ZFx86™

System-on-a-Chip

Data Book

Version 1.0 Rev C

November 28, 2001

THIS DOCUMENT AND THE INFORMATION CONTAINED THEREIN IS PROVIDED "AS-IS" AND WITHOUT A WARRANTY OF ANY KIND. YOU, THE USER, ACCEPT FULL RESPONSIBILITY FOR PROPER USE OF THE MATERIAL. ZF MICRO DEVICES, INC. MAKES NO REPRESENTATIONS OR WARRANTIES THAT THIS DATA BOOK OR THE INFORMATION CONTAINED THERE-IN IS ERROR FREE OR THAT THE USE THEREOF WILL NOT INFRINGE ANY PATENTS, COPYRIGHT OR TRADEMARKS OF THIRD PARTIES. ZF MICRO DEVICES, INC. EXPLICITLY ASSUMES NO LIABILITY FOR ANY DAMAGES WHATSOEVER RELATING TO ITS USE.

LIFE SUPPORT POLICY

ZF MICRO DEVICES' PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZF MICRO DEVICES, INC.

As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.

2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

1. Overview

The “ZFx86” System-On-a-Chip (SOC) is a complete processor and peripheral subsystem requiring only external clocks, SDRAM, and BIOS ROM/Flash. It is illustrated in ZFx86 Fail-Safe PC-on-a-Chip Block Diagram and consists of the following major blocks:

1) Industry standard 32 bit processor core with integrated floating point co-processor and 8K byte write-back level 1 cache. The clock multiplier design allows the core to run at a multiple of the system bus. For example, a 3x multiplier delivers a system running at 100 MHz with a 33 MHz PCI bus.

2) A North Bridge (system controller) with “Frontside” PCI Master / Slave Arbitration interface and SDRAM interface. See ‘North Bridge ’ on page 108.

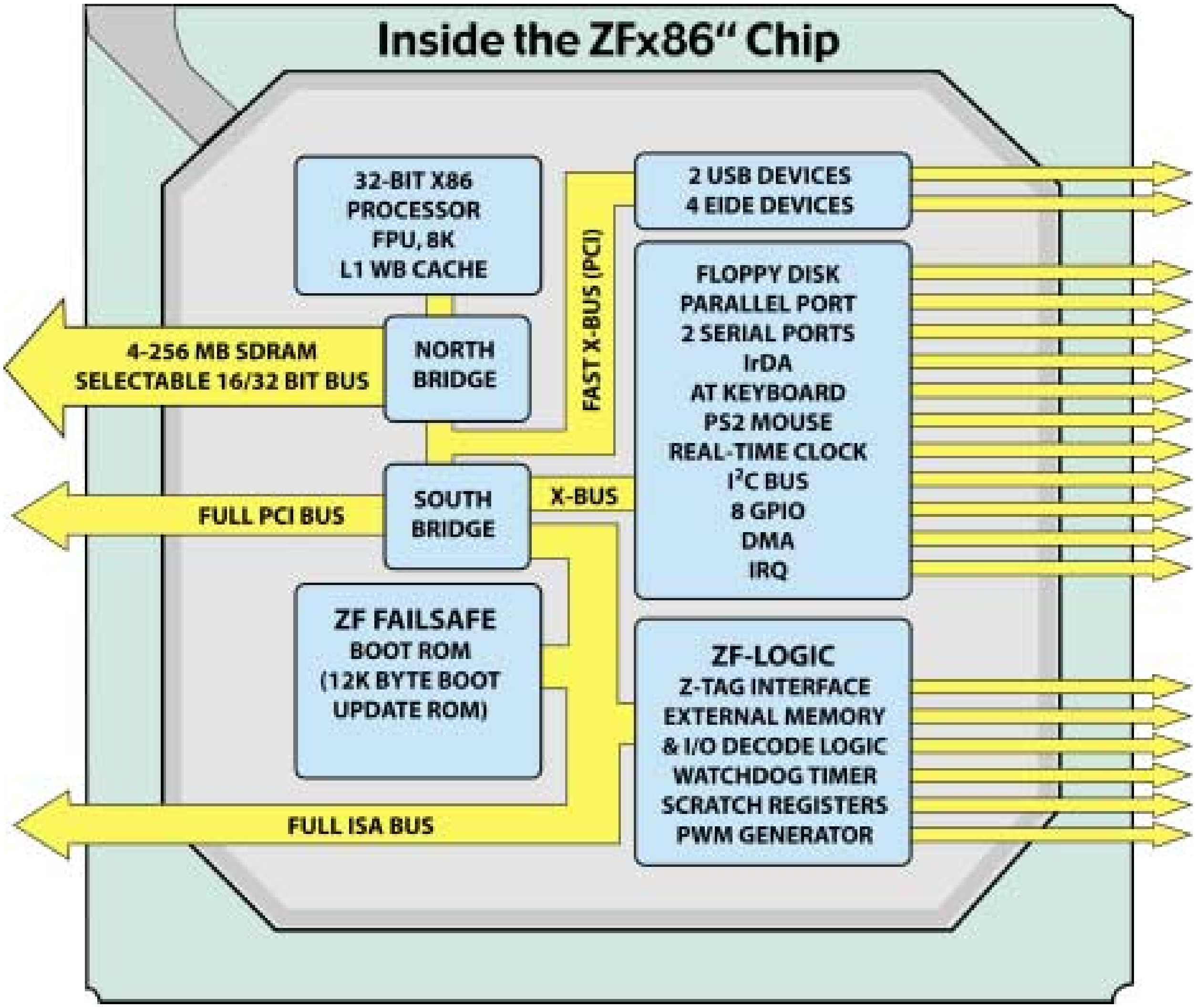
3) A custom South Bridge with “Frontside” PCI interface to the North Bridge and “Backside” PCI Master/Slave system interface, enhanced IDE controller supporting four devices on two channels, USB controller with two hub ports, real time clock (RTC), floppy disk controller, serial ports, access bus, 8042 compatible keyboard and mouse controller, parallel port, general purpose programmable I/O’s and counters, PC/AT system components, and power management. The PC/AT system components include 8237 compatible DMA controllers, two 8259 compatible interrupt controllers, 8254 compatible system timer, and ISA bus interface. See ‘South Bridge ’ on page 160.

4) 12K Bytes of ROM with ZF proprietary code. This BIOS Update ROM (BUR) is used in a special mode which allows a flash based BIOS to be updated without removal of any system components or peripherals. See ‘BUR (BIOS Update ROM) ’ on page 456.

5) ZF proprietary digital logic including specific and general purpose chip selects, watchdog timer, and flash controller. See ‘ZF-Logic and Clocking ’ on page 397.

The above functions are packaged in a 35 MM. 388 pin Ball Grid Array (BGA). See ‘Pinout Summary ’ on page 533.

Figure 1-1 ZFx86 Fail-Safe PC-on-a-Chip Block Diagram



- **Processor**: 486+ CPU at 128 MHz
- **North Bridge**: DRAM Controller and FrontSide 32 MHz PCI Bus
- **South Bridge**: Generates BackSide PCI and ISA Buses.
- **USB + Extended IDE Device Interface**: on the FrontSide PCI Bus
- **SuperIO**: Industry Standard X86 I/O + I^2C
- **ZF-Logic**: ISA Additions for Embedded Systems, Low BOM cost, and FailSafe

2. 32-bit x86 Processor

2.1. Overview

The Processor is an industry standard 32-bit x86 compatible microprocessor.

Configure the 8 KB cache to run in traditional write-through mode or in the higher performance write-back mode. Write-back mode eliminates unnecessary external memory write cycles offering higher overall performance than write-through mode.

The processor supports 8-, 16- and 32-bit data types and operates in real, virtual 8086 and protected modes. The CPU accesses up to 256 MB of physical memory using a 32-bit burst mode bus. Floating point instructions are parallel processed using an on-chip math coprocessor.

The processor is an ideal design solution for low-powered applications. Due to its static design, it features a low current drain while the input clock is stopped in suspend mode. SMM (System Management Mode) allows the implementation of transparent system power management or the software emulation of I/O peripheral devices.

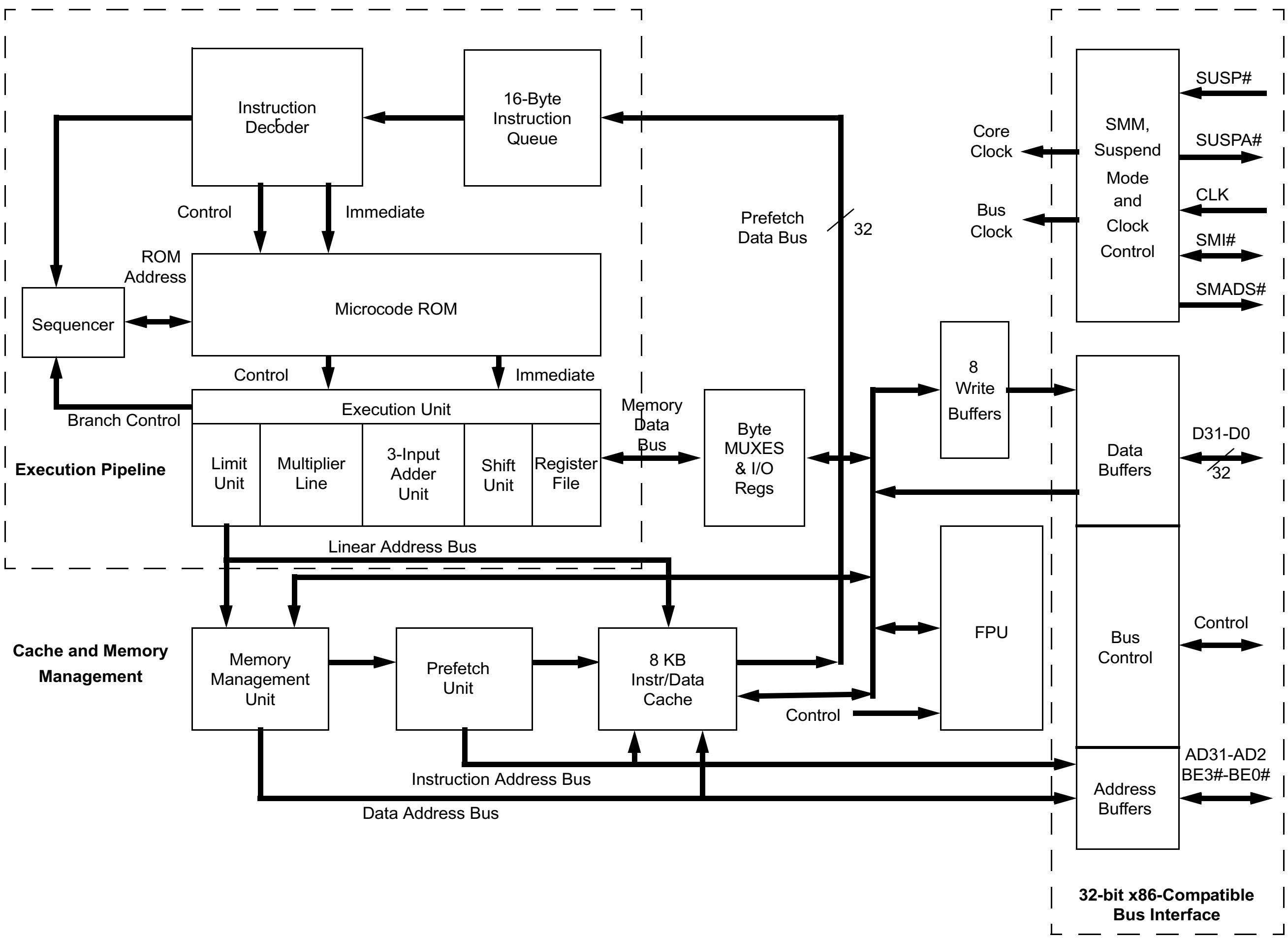


Figure 2-1 Processor Block Diagram

2.1.1. Internal Clock Logic

The processor operates in 4 clock rate modes and 3 clock operation modes. These modes are controlled by 4 signals, CLKMODE0, CLKMODE1, PLLMODE and RAWCLK. Additionally 3 signals, CLKDEL[2:0] control the duty cycle of the internal clock signal.

As you look at the clock modes, please also reference Table 5.42 "Composite BootStrap Register Map" on page 432, and Figure 5-9 "System Clocking and Control" on page 437.

2.1.1.1. Clock Rate Modes

The internal clock rate can be 1, 2, 3 or 4 times the input clock rate as controlled by the CLKMODE[1:0] signals.

RATE	CLKMODE[1:0]
1X	00b
2X	01b
3X	11b
4X	10b

2.1.1.2. Clock Operation Modes

The source of the internal clock is determined by the PLLMODE and RAWCLK signals. Three modes of operation are supported, PLL Mode, Delay Mode and Raw Clock Mode.

Mode	PLLMODE	RAWCLK
PLL Mode	1	1
Delay Mode	0	1
Rawclock Mode	0	0
NOT SUPPORTED	1	0

PLL Mode (DLL Mode)

In PLL Mode the source of the internal clock is from the Digital Locked Loop. Clock modes 1x, 2x, 3x and 4x are supported. The duty cycle of the internal clock is determined by the state of the CLKDEL[2:0] signals.

Delay Mode

In Delay Mode the source of the internal clock is from the Clock Delay circuitry. Modes 1x, 2x, 3x and 4x are supported. The duty cycle and frequency of the internal clock is determined by the state of the CLKDEL[2:0] signals. The exact operation of this mode is beyond the scope of this document.

Raw Clock Mode

Raw Clock Mode is normally used for test purposes only. In Raw Clock Mode the source of the internal clock is from the CLK port. Only clock rate mode 1x is supported in this mode.

Clock Delay Signals

The CLKDEL[2:0] signals effect the duty cycle of the internal clock. The exact effect is beyond the scope of this document. The setting of these signals is determined during early production testing experimentally. The setting which results in the best performance over voltage, temperature and frequency is normally used as a bond out option in the final package.

2.1.2. On-Chip Write-Back Cache

The processor on-chip cache can be configured to run in traditional write-through mode or in a higher performance write-back mode. The write-back cache mode was specifically designed to optimize performance of the CPU core by eliminating bus bottlenecks caused by unnecessary external write cycles. This write-back architecture is especially effective in improving performance of the clock-tripled processor.

Traditional write-through cache architectures require that all writes to the cache also update external memory simultaneously. These unnecessary write cycles create bottlenecks which result in CPU stalls that adversely impact performance. In contrast, a write-back architecture allows data to be written to the cache without updating external memory. With a write-back cache, external write cycles are only required when a cache miss occurs, a modified line is replaced in the cache, or when an external bus master requires access to data.

The processor cache is an 8 KB unified instruction. Data cache is implemented using a four-way set associative architecture and an LRU (Least Recently Used) replacement algorithm. The cache is designed for optimum performance in write-back mode; however, the cache can be operated in write-through mode. The cache line size is 16 bytes and new lines are only allocated during memory read cycles. Valid status is maintained on a 16-byte cache line basis, but modified or “dirty” status for write-back mode is maintained on a 4-byte DWORD (Double Word) basis. Therefore, only the DWORDs that have been modified are written back to external memory when a line is replaced in the cache. The CPU core can access the cache in a single internal clock cycle for both reads and writes.

2.1.3. System Management Mode

System Management Mode (SMM) provides an additional interrupt and a separate address space that can be used for system power management or software transparent emulation of I/O peripherals. SMM is entered using the SMI# (System Management Interrupt) or SMINT instruction. While running in isolated SMM address space, the SMI interrupt routine can execute without interfering with the operating system or application programs.

After entering SMM, portions of the CPU state are automatically saved. Program execution begins at the base of SMM address space. The location and size of the SMM memory are programmable within the processor. Eight SMM instructions have been added to the processor instruction set that permit software entry into SMM, as well as saving and restoring the total CPU state when in SMM mode.

2.1.4. Power Management

The processor power management features allow for a dramatic improvement in battery life over systems designed with non-static processors. During suspend mode the typical current consumption is far less than full operation current.

Suspend mode is entered by either a hardware or a software initiated action. Using the hardware method to initiate suspend mode involves a two-signal handshake between the SUSP# and SUSPA# signals. The software can initiate suspend mode through the execution of the HALT instruction. Once in suspend mode, power consumption is further reduced by stopping the external clock input. Since the processor is static, no internal data is lost when the clock is stopped.

2.1.5. Signal Summary

The processor interface signal set includes five cache interface signals, two coprocessor interface signals, two power management signals, and two system management mode signals.

2.2. Programming Interface

In this chapter the internal operations of the Processor are described from an application programmer’s point of view. Included in this chapter are descriptions of processor initialization, the register set, memory addressing, various types of interrupts and the shutdown and halt process. An overview is provided of real, virtual 8086 and protected operating modes. FPU operations are described separately at the end of this chapter.

2.2.1. Processor Initialization

The processor is initialized when the RESET# signal is asserted. The processor is placed in real mode and the registers listed in Table 2.1 are set to their initialized values. RESET invalidates and disables the processor cache, and turns off paging. When RESET# is asserted the processor terminates all local bus activity and all internal execution. During the entire time that RESET# is asserted the internal pipeline is flushed, and no instruction execution or bus activity occurs.

Approximately 150 to 250 external clock cycles (additional 2^{20} + 60 if self-test is requested) after RESET is negated, the processor begins executing instructions at the top of physical memory (address location FFFF FFF0h). When the first intersegment JMP or CALL is executed, address lines AD31-AD20 are driven low for code segment-relative memory access cycles. While AD31-AD20 are low, the processor executes instructions only in the lowest 1 MB of physical address space until system-specific initialization occurs via program execution.

Table 2.1 Initialized Register Controls

Register	Register Name	Initialized Contents	Comments
EAX	Accumulator	xxxx xxxxh	0000 0000h indicates self-test passed
EBX	Base	xxxx xxxxh	
ECX	Count	xxxx xxxxh	
EDX	Data	xxxx 0400h + Device ID	Device ID = xxh
EBP	Base Pointer	xxxx xxxxh	
ESI	Source Index	xxxx xxxxh	
EDI	Destination Index	xxxx xxxxh	
ESP	Stack Pointer	xxxx xxxxh	
EFLAGS	Flag Word	0000 0002h	
EIP	Instruction Pointer	0000 FFF0h	
ES	Extra Segment	0000h	Base address set to 0000 0000h. Limit set to FFFFh
CS	Code Segment	F000h	Base address set to FFFF 0000h. Limit set to FFFFh
SS	Stack Segment	0000h	Base address set to 0000 0000h. Limit set to FFFFh
DS	Data Segment	0000h	Base address set to 0000 0000h. Limit set to FFFFh
FS	Extra Segment	0000h	Base address set to 0000 0000h. Limit set to FFFFh
GS	Extra Segment	0000h	Base address set to 0000 0000h. Limit set to FFFFh
IDTR	Interrupt Descriptor Table Register	Base = 0, Limit = 3FFh	
CR0	Machine Status Word	6000 0010h	
CCR1	Configuration Control 1	00h	
CCR2	Configuration Control 2	00h	
CCR3	Configuration Control 3	00h	
SMAR	SMM Address Region	0000h	
DIR0	Device Identification 0	processor = xxh	
DIR1	Device Identification 1	Step ID + Revision ID	
DR7	Debug Register 7	0000 0400h	

Note: x = Undefined value

2.2.1.1. Warm Reset

The WM_RESET input signal is used to support write back caching policy on the processor. The WM_RESET signal will reset the entire processor except for the CD and NW bits in the CR0 register, the CFG0 register, the CFG1 register and the valid and dirty bits in the cache. The WM_RESET signal is always enabled and included a pull-down resistor to keep the pin inactive when not used.

2.2.2. Instruction Set Overview

The processor instruction set can be divided into eight types of operations:

- Arithmetic
- Bit Manipulation
- Control Transfer
- Data Transfer
- Floating Point
- High-Level Language Support
- Operating System Support
- Shift/Rotate
- String Manipulation

All processor instructions operate on as few as 0 operands and as many as 3 operands. An NOP instruction (no operation) is an example of a 0 operand instruction. Two operand instructions allow the specification of an explicit source and destination pair as part of the instruction. These two operand instructions can be divided into eight groups according to operand types:

- Register to Register
- Register to Memory
- Memory to Register
- Memory to Memory
- Register to I/O
- I/O to Register
- Immediate Data to Register
- Immediate Data to Memory

An operand can be held in the instruction itself (as in the case of an immediate operand), in a register, in an I/O port or in memory. An immediate operand is prefetched as part of the opcode for the instruction.

- Operand lengths of 8, 16, or 32 bits are supported as well as 64 or 80 bit associated with floating point instructions.
- Operand lengths of 8 or 32 bits are generally used when executing code written for x86 32-bit code processors.
- Operand lengths of 8 or 16 bits are generally used when executing existing 8086 or 80286 code (16-bit code).

The default length of an operand can be overridden by placing one or more instruction prefixes in front of the opcode. For example, by using prefixes, a 32-bit operand can be used with 16-bit code or a 16-bit operand can be used with 32-bit code.

Section 2.3. ‘Instruction Set’ of this manual lists each instruction in the processor instruction set along with the associated opcodes, execution clock counts and effects on the FLAGS register.

2.2.2.1. Lock Prefix

The LOCK prefix may be placed before certain instructions that read, modify, then write back to memory. The prefix asserts the LOCK# signal to indicate to the external hardware that the CPU is in the process of running multiple indivisible memory accesses. The LOCK prefix can be used with the following instructions:

- Bit Test Instructions (BTS, BTR, BTC)
- Exchange Instructions (XADD, XCHG, CMPXCHG)
- One-operand Arithmetic and Logical Instructions (DEC, INC, NEG, NOT)

- Two-operand Arithmetic and Logical Instructions (ADC, ADD, AND, OR, SBB, SUB, XOR)

An invalid opcode exception is generated if the LOCK prefix is used with any other instruction, or with the above instructions when no write operation to memory occurs (i.e., the destination is a register). The LOCK prefix function may be disabled by setting the NO_LOCK bit in Configuration Control Register 1 (CCR1).

If No_Lock (bit 4 in CCR1) is set, locked cycles are inhibited for some locked instructions. These instructions include interrupt acknowledge cycles, descriptor loads, and updates and accesses to the interrupt descriptor table. However, locked cycles are not inhibited by No_Lock bit for TLB table lookups, XCHG instructions to memory, or any instruction that includes a lock prefix.

If No_Lock = 0, locked cycles occur for all locked instructions.

2.2.3. Register Set

There are 40 accessible registers in the processor, and these registers are grouped into two sets:

- The Application Register Set contains eight general purpose registers, six segment registers, a flag register and an instruction pointer register, and are typically used by application programmers.
- The System Register Set contains the remaining registers which include three control registers, four system address registers, six debug registers, six configuration registers and five test registers, and are typically used by operating system programmers.

Each of the registers is discussed in detail in the following sections.

2.2.3.1. Application Register Set

The Application Register Set, shown in Table 2.2, are generally accessible and are not protected from read or write access.

The contents of the eight General Purpose Registers are frequently modified by assembly language instructions and typically contain arithmetic and logical instruction operands.

In real mode the six Segment Registers contain the base address for each segment. In protected mode the Segment Registers contain segment selectors. The segment selectors provide indexing for tables (located in memory) that contain the base address for each segment, as well as other memory addressing information.

The Flag Register contains control bits used to reflect the status of previously executed instructions. This register also contains control bits that affect the operation of some instructions.

The Instruction Pointer register points to the next instruction that the processor will execute. This register is automatically incremented by the processor as execution progresses.

Table 2.2 Application Register Set

31 … 16	15 … 8	7 … 0	Application Register Set
	AX		General Purpose Registers
	AH	AL	
EAX (Extended A Register)			
	BX		
	BH	BL	
EBX (Extended B Register)			
	CX		
	CH	CL	
ECX (Extended C Register)			
	DX		
	DH	DL	
EDX (Extended D Register)			
	SI (Source Index)		
ESI (Extended Source Index)			
	DI (Destination Index)		
EDI (Extended Destination Index)			
	BP (Base Pointer)		
EBP (Extended Base Pointer)			
	SP (Stack Pointer)		
ESP (Extended Stack Pointer)			
	CS (Code Segment)		Segment (Selector) Registers
	SS (Stack Segment)		
	DS (D Data Segment)		
	ES (E Data Segment)		
	FS (F Data Segment)		
	GS (G Data Segment)		
EIP (Extended Instruction Pointer Register)			Instruction Pointer and Flags Register
EFLAGS (Extended Flags Register)			

General Purpose Registers (eight)

The General Purpose Registers are divided into four data registers, two pointer registers, and two index registers.

Data Registers are used by the applications programmer to manipulate data structures and to hold the results of logical and arithmetic operations. Different portions of the general data registers can be addressed by using different names. An “E” prefix identifies the complete 32-bit register. An “X” suffix without the “E” prefix identifies the lower 16 bits of the register. The lower two bytes of the register can be addressed with an “H” suffix to identify the upper byte or an “L” suffix to identify the lower byte. When a destination operand size specified by an instruction is smaller than the specified destination register, the other bytes of the destination register are not affected when the operand is written to the register.

The Pointer and Index Registers are listed as follows:

- SI or ESI   Source Index
- DI or EDI   Destination Index
- SP or ESP   Stack Pointer
- BP or EBP   Base Pointer

These registers can be addressed as 16- or 32-bit registers, with the "E" prefix indicating 32 bits. The pointer and index registers can be used as general purpose registers, however, some instructions use a fixed assignment of these registers. For example, repeated string operations always use ESI as the source pointer, EDI as the destination pointer and ECX as a counter. The instructions using fixed registers include multiply and divide, I/O access, string operations, translate, loop, variable shift and rotate and stack operations instructions.

The processor implements a stack using the ESP register. This stack is accessed during the PUSH and POP instructions, procedure calls, procedure returns, interrupts, exceptions, and interrupt/exception returns. The microprocessor automatically adjusts the value of the ESP during operation of these instructions.

The EBP register may be used to reference data passed on the stack during procedure calls. Local data may also be placed on the stack and referenced relative to BP. This register provides a mechanism to access stack data in high-level languages.

Segment Registers and Selectors (six)

Segmentation provides a means of defining data structures inside the memory space of the microprocessor. There are three basic types of segments: code, data, and stack. Segments are used automatically by the processor to determine the location in memory of code, data and stack references.

There are six 16-bit segment registers:

- CS – Code Segment
- DS – Data Segment
- ES – Extra Segment
- SS – Stack Segment
- FS – Additional Data Segment
- GS – Additional Data Segment

In real and virtual 8086 operating modes, a segment register holds a 16-bit segment base. The 16-bit segment base is multiplied by 16 and a 16-bit or 32-bit offset is then added to it to create a linear address. The offset size is dependent on the current address size. In real mode and in virtual 8086 mode with paging disabled, the linear address is also the physical address. In virtual 8086 mode with paging enabled, the linear address is translated to the physical address using the current page tables.

In protected mode, a segment register holds a Segment Selector containing a 13-bit Index, a Table Indicator (TI) bit, and a two-bit Requested Privilege Level (RPL) field, as illustrated in Figure 2-1.

TI = Table Indicator
RPL = Request Privilege Level

Figure 2-1 Segment Selector

The Index Register points into a descriptor table in memory and selects one of 8192 (2^{13}) segment descriptors contained in the descriptor table. A segment descriptor is an 8-byte value used to describe a memory segment by defining the segment base, the segment limit, and access control information.

To address data within a segment, a 16-bit or 32-bit offset is added to the segment's base address. Once a segment selector has been loaded into a segment register, an instruction needs to specify the offset only. The Table Indicator (TI) bit of the selector defines which descriptor table the index points to. If TI = 0, the index references the Global Descriptor Table (GDT). If TI = 1, the index references the Local Descriptor Table (LDT). The GDT and LDT are described in more detail later in Section 'Descriptor Table Registers and Descriptors' on page 38.

The Requested Privilege Level (RPL) field contains a 2-bit segment privilege level (0 = most privileged, 3 = least privileged). The RPL bits are used when the segment register is loaded to determine the Effective Privilege Level (EPL). If the RPL bits indicate less privilege than the Current Program Level (CPL), the RPL overrides the CPL and the EPL is the less privileged level. If the RPL bits indicate more privilege than the program, the CPL overrides the RPL and again the EPL is the less privileged level.

When a segment register is loaded with a segment selector, the segment base, segment limit and access rights are also loaded from the descriptor table into a user-invisible or hidden portion of the segment register i.e., cached on-chip. The CPU does not access the descriptor table again until another segment register load occurs. If the descriptor tables are modified in memory, the segment registers must be reloaded with the new selector values by the software.

The processor automatically selects a default segment register for memory references. Table 2.3 describes the selection rules. In general, data references use the selector contained in the DS register, stack references use the SS register and instruction fetches use the CS register. While some of these selections may be overridden, instruction fetches, stack operations, and the destination write of string operations cannot be overridden. Special segment override prefixes allow the use of alternate segment registers including the use of the ES, FS, and GS segment registers.

Table 2.3 Segment Register Selection Rules

Type of Memory Reference	Implied (Default) Segment	Segment-Override Prefix
Code Fetch	CS	None
Destination of PUSH, PUSHF, INT, CALL, PUSHA instructions	SS	None
Source of POP, POPA, POPF, IRET, RET instructions	SS	None
Destination of STOS, MOVS, REP STOS, REP MOVS instructions	ES	None
Other data references with effective address using base registers of: EAX, EBX, ECX, EDX, ESI, EDI, EBP, ESP	DS SS	CS, ES, FS, GS, SS CS, DS, ES, FS, GS

Instruction Pointer Register (one)

The Instruction Pointer (EIP) register contains the offset into the current code segment of the next instruction to be executed. The register is normally incremented with each instruction execution unless implicitly modified through an interrupt, exception or an instruction that changes the sequential execution flow (e.g., JMP, CALL).

Flags Register (one)

The Extended Flags Register, EFLAGS, contains status information and controls certain operations on the processor. The lower 16 bits of this register are referred to as the FLAGS register that is used when executing 8086 or 80286 code. The flag bits are illustrated in Table 2.4

Table 2.4 EFLAGS Register

Bit	Name	Flag Type	Description
31:19	RSVD	--	**Reserved** — Set to 0.
18	AC	System	**Alignment Check Enable** — In conjunction with the AM flag in CR0, the AC flag determines whether or not misaligned accesses to memory cause a fault. If AC is set, alignment faults are enabled.
17	VM	System	**Virtual 8086 Mode** — If set while in protected mode, the processor switches to virtual 8086 operation handling segment loads as the 8086 does, but generating exception 13 faults on privileged opcodes. The VM bit can be set by the IRET instruction (if current privilege level is 0) or by task switches at any privilege level.
16	RF	Debug	**Resume Flag** — Used in conjunction with debug register breakpoints. RF is checked at instruction boundaries before breakpoint exception processing. If set, any debug fault is ignored on the next instruction.
15	RSVD	--	**Reserved** — Set to 0.
14	NT	System	**Nested Task** — While executing in protected mode, NT indicates that the execution of the current task is nested within another task.
13:12	IOPL	System	**I/O Privilege Level** — While executing in protected mode, IOPL indicates the maximum current privilege level (CPL) permitted to execute I/O instructions without generating an exception 13 fault or consulting the I/O permission bit map. IOPL also indicates the maximum CPL allowing alteration of the IF bit when new values are popped into the EFLAGS register.
11	OF	Arithmetic	**Overflow Flag** — Set if the operation resulted in a carry or borrow into the sign bit of the result but did not result in a carry or borrow out of the high-order bit. Also set if the operation resulted in a carry or borrow out of the high-order bit but did not result in a carry or borrow into the sign bit of the result.
10	DF	Control	**Direction Flag** — When cleared, DF causes string instructions to auto-increment (default) the appropriate index registers (ESI and/or EDI). Setting DF causes auto-decrement of the index registers to occur.
9	IF	System	**Interrupt Enable Flag** — When set, maskable interrupts (INTR input signal) are acknowledged and serviced by the CPU.
8	TF	Debug	**Trap Enable Flag** — Once set, a single-step interrupt occurs after the next instruction completes execution. TF is cleared by the single-step interrupt.
7	SF	Arithmetic	**Sign Flag** — Set equal to high-order bit of result (0 indicates positive, 1 indicates negative).
6	ZF	Arithmetic	**Zero Flag** — Set if result is zero; cleared otherwise.
5	RSVD	--	**Reserved** — Set to 0.
4	AF	Arithmetic	**Auxiliary Carry Flag** — Set when a carry out of (addition) or borrow into (subtraction) bit position 3 of the result occurs; cleared otherwise.
3	RSVD	--	**Reserved** — Set to 0.
2	PF	Arithmetic	**Parity Flag** — Set when the low-order 8 bits of the result contain an even number of ones; otherwise PF is cleared.
1	RSVD	--	**Reserved** — Set to 1.
0	CF	Arithmetic	**Carry Flag** — Set when a carry out of (addition) or borrow into (subtraction) the most significant bit of the result occurs; cleared otherwise.

2.2.3.2. System Register Set

The System Register Set, shown in Table 2.5, consists of registers not generally used by application programmers. These registers are typically employed by system level programmers who generate operating systems and memory management programs.

The **Control Registers** control certain aspects of the processor such as paging, coprocessor functions, and segment protection. When a paging exception occurs while paging is enabled, the control registers retain the linear address of the access that caused the exception.

The **Descriptor Table Registers** and the **Task Register** can also be referred to as system address or memory management registers. These registers consist of two 48-bit and two 16-bit registers. These registers specify the location of the data structures that control the segmentation used by the processor. Segmentation is one available method of memory management.

The **Configuration Registers** are used to configure the processor on-chip cache operation, coprocessor interface, power management features and System Management Mode. The configuration registers also provide information on the CPU device type and revision.

The **Debug Registers** provide debugging facilities for the processor and enable the use of data access breakpoints and code execution breakpoints.

The **Test Registers** provide a mechanism to test the contents of both the on-chip 8 KB cache and the Translation Lookaside Buffer (TLB). The TLB is used as a cache for the tables which are used in translating linear addresses to physical addresses while paging is enabled. In the following sections, the system register set is described in greater detail.

Table 2.5 System Register Set

Group	Name	Function	Width (Bits)
Control Registers	CR0	System Control Register	32
	CR2	Page Fault Linear Address Register	32
	CR3	Page Directory Base Register	32
Descriptor Table and Task Registers	GDTR	GDT Register	48
	IDTR	IDT Register	48
	LDTR	LDT Register	16
	TR	Task Register Setup	16
Configuration Registers	CCR1	Configuration Control Register 1	8
	CCR2	Configuration Control Register 2	8
	CCR3	Configuration Control Register 3	8
	SMAR	SMM Address Region Register	24
	DIR0	Device Identification Register 0	8
	DIR1	Device Identification Register 1	8
Debug Registers	DR0	Linear Breakpoint Address 0	32
	DR1	Linear Breakpoint Address 1	32
	DR2	Linear Breakpoint Address 2	32
	DR3	Linear Breakpoint Address 3	32
	DR6	Breakpoint Status	32
	DR7	Breakpoint Control	32
Test Registers	TR3	Cache Test	32
	TR4	Cache Test	32
	TR5	Cache Test	32
	TR6	TLB Test Control	32
	TR7	TLB Test Status	32

2.2.3.3. Control Registers (3)

A map of the Control Registers (CR3, CR2 and CR0) is shown in Table 2.6 and the bit definitions are given in Table 2.7. The CR0 register contains system control flags which control operating modes and indicate the general state of the CPU. The lower 16 bits of CR0 are referred to as the Machine Status Word (MSW). The reserved bits in CR0 should not be modified.

When paging is enabled and a page fault is generated, the CR2 register retains the 32-bit linear address of the address that caused the fault. Register CR3 contains the 20 most significant bits of the physical base address of the page directory. The page directory must always be aligned to a 4 KB page boundary; therefore, the lower 12 bits of CR3 are not required to specify the base address.

When operating in protected mode, any program can read the control registers; however, only privilege level 0 (most privileged) programs can modify the contents of these registers.

Table 2.6 Control Registers Map

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="32">CR3 Register</td></tr>
<tr><td colspan="20">PDBR (Page Directory Base Register)</td><td colspan="7">RSVD</td><td>PCD</td><td>PWT</td><td colspan="3">RSVD</td></tr>
<tr><td colspan="32">CR2 Register</td></tr>
<tr><td colspan="32">PFLA (Page Fault Linear Address)</td></tr>
<tr><td colspan="32">CR0 Register</td></tr>
<tr><td>PG</td><td>CD</td><td>NW</td><td colspan="10">RSVD</td><td>AM</td><td>RSVD</td><td>WP</td><td colspan="10">RSVD</td><td>NE</td><td>1</td><td>TS</td><td>EM</td><td>MP</td><td>PE</td></tr>
<tr><td colspan="16"></td><td colspan="16">Machine Status Word (MSW)</td></tr>
</table>

Table 2.7 CR3, CR2, and CR0 Bit Definitions

Bit	Name	Description
CR3 Register		
31:12	PDBR	**Page Directory Base Register:** Identifies page directory base address on a 4 KB page boundary.
11:4	RSVD	**Reserved**
4	PCD	**Page-level Cache Disable**: Affects the operation of internal cache.
3	PWT	**Page Write Through:** Drives output pins for controlling external caches.
2:0	RSVD	**Reserved**
CR2 Register		
31:0	PFLA	**Page Fault Linear Address:** With paging enabled and after a page fault, PFLA contains the linear address of the address that caused the page fault.
CR0 Register		

Table 2.7 CR3, CR2, and CR0 Bit Definitions (cont.)

Bit	Name	Description
31	PG	**Paging Enable Bit:** If PG = 1 and protected mode is enabled (PE = 1), paging is enabled.
30	CD	**Cache Disable:** If CD = 1, no further cache line fills occur; however data already present in the cache continues to be used if the requested address hits in the cache. The cache must also be invalidated to completely disable any cache activity.
29	NW	**Not Write-Through:** If NW = 1, the on-chip cache operates in write-back mode. In write-back mode, writes are issued to the external bus only for a cache miss, a line replacement of a modified line, or as the result of a cache inquiry cycle. If NW = 0, the on-chip cache operates in write-through mode. In write-through mode, all writes (including cache hits) are issued to the external bus.
28:19	RSVD	**Reserved**
18	AM	**Alignment Check Mask:** If AM = 1, the AC bit in the EFLAGS register is unmasked and allowed to enable alignment check faults. Setting AM = 0 prevents AC faults from occurring.
17	RSVD	**Reserved**
16	WP	**Write Protect:** Protects read-only pages from supervisor write access. WP = 0 allows a read-only page to be written from privilege level 0-2. WP = 1 forces a fault on a write to a read-only page from any privilege level.
15:6	RSVD	**Reserved**
5	NE	**Numerics Exception:** NE = 1 to allow FPU exceptions to be handled by interrupt 16. NE = 0 if FPU exceptions are to be handled by external interrupts.
4	1	**Reserved:** Do not attempt to modify.
3	TS	**Task Switched:** Set whenever a task switch operation is performed. Execution of a floating point instruction with TS = 1 causes a DNA (Device Not Available) fault. If MP = 1 and TS = 1, a WAIT instruction also causes a DNA fault.
2	EM	**Emulate Processor Extension:** If EM = 1, all floating point instructions cause a DNA fault 7.
1	MP	**Monitor Processor Extension:** If MP = 1 and TS = 1, a WAIT instruction causes a DNA fault 7. The TS bit is set to 1 on task switches by the CPU. Floating point instructions are not affected by the state of the MP bit. The MP bit should be set to one during normal operations.
0	PE	**Protected Mode Enable:** Enables the segment based protection mechanism. If PE = 1, protected mode is enabled. If PE = 0, the CPU operates in real mode, with segment based protection disabled, and addresses are formed as in an 8086-style CPU.

Table 2.8 Effects of Various Combinations of TS, EM and MP Bits

CR0[3:1]			Instruction Type	
TS	**EM**	**MP**	**WAIT**	**ESC**
0	0	0	Execute	Execute
0	0	1	Execute	Execute
1	0	0	Execute	Fault 7
1	0	1	Fault 7	Fault 7
0	1	0	Execute	Fault 7
0	1	1	Execute	Fault 7
1	1	0	Execute	Fault 7
1	1	1	Fault 7	Fault 7

Descriptor Table Registers and Descriptors

The Global, Interrupt and Local Descriptor Table Registers (GDTR, IDTR and LDTR), shown in Figure 2-2 "Task Register", are used to specify the location of the data structures that control segmented memory management. The GDTR, IDTR and LDTR are loaded using the LGDT, LIDT and LLDT instructions, respectively. The values of these registers are stored using the corresponding store instructions. The GDTR and IDTR load instructions are privileged instructions when operating in protected mode. The LDTR can only be accessed in protected mode.

The **Global Descriptor Table Register (GDTR)** holds a 32-bit linear base address and 16-bit limit for the Global Descriptor Table (GDT). The GDT is an array of up to 8192 8-byte descriptors. When a segment register is loaded from memory, the TI bit in the segment selector chooses either the GDT or the Local Descriptor Table (LDT) to locate a descriptor. If TI = 0, the index portion of the selector is used to locate a given descriptor within the GDT. The contents of the GDTR are completely visible to the programmer. The first descriptor in the GDT (location 0) is not used by the CPU and is referred to as the “null descriptor”. If the GDTR is loaded while operating in 16-bit operand mode, the processor accesses a 32-bit base value but the upper 8 bits are ignored resulting in a 24-bit base address.

The **Interrupt Descriptor Table Register (IDTR)** holds a 32-bit linear base address and 16-bit limit for the Interrupt Descriptor Table (IDT). The IDT is an array of 256 8-byte interrupt descriptors, each of which is used to point to an interrupt service routine. Every interrupt that may occur in the system must have an associated entry in the IDT. The contents of the IDTR are completely visible to the programmer.

The **Local Descriptor Table Register (LDTR)** holds a 16-bit selector for the Local Descriptor Table (LDT). The LDT is an array of up to 8192 8-byte descriptors. When the LDTR is loaded, the LDTR selector indexes an LDT descriptor that must reside in the Global Descriptor Table (GDT). The contents of the selected descriptor are cached on-chip in the hidden portion of the LDTR. The CPU does not access the GDT again until the LDTR is reloaded. If the LDT descriptor is modified in memory in the GDT, the LDTR must be reloaded to update the hidden portion of the LDTR.

When a segment register is loaded from memory, the TI bit in the segment selector chooses either the GDT or the LDT to locate a segment descriptor. If TI = 1, the index portion of the selector is used to locate a given descriptor within the LDT. Each task in the system may be given its own LDT, managed by the operating system. The LDTs provide a method of isolating a given task’s segments from other tasks in the system.

48 — 16	15 — 0	
Base Address	Limit	GDTR
Base Address	Limit	IDTR
	Selector	LDTR

Figure 2-2 Descriptor Table Registers

Descriptors

There are three types of descriptors:

- Application Segment Descriptors that define code, data and stack segments.
- System Segment Descriptors that define an LDT segment or a Task State Segment (TSS) table described later in this text.
- Gate Descriptors that define task gates, interrupt gates, trap gates and call gates.

Application Segment Descriptors are located in either the LDT or GDT; System Segment Descriptors can only be located in the GDT. Dependent on gate type, gate descriptors are located in either the GDT, LDT or Interrupt Descriptor Table (IDT). Table 2.9 illustrates the descriptor format for both Application Segment Descriptors and System Segment Descriptors.

Table 2.9 Application and System Segment Descriptors

31	31	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Memory Offset +4																															
BASE[31:24]								G	D	0	A V L	LIMIT[19:16]				P	DPL		D T	TYPE				BASE[23:16]							
Memory Offset +0																															
BASE[15:0]																LIMIT[15:0]															

Gate Descriptors provide protection for executable segments operating at different privilege levels. Table 2.10 illustrates the format for Gate Descriptors and Table 2.11 lists the corresponding bit definitions.

Task Gate Descriptors (TGD) are used to switch the CPU’s context during a task switch. The selector portion of the TGD locates a Task State Segment. TGDs can be located in the GDT, LDT or IDT tables.

Interrupt Gate Descriptors are used to enter a hardware interrupt service routine. Trap Gate Descriptors are used to enter exceptions or software interrupt service routines. Trap Gate and Interrupt Gate Descriptors can only be located in the IDT.

Call Gate Descriptors are used to enter a procedure (subroutine) that executes at the same or a more privileged level. A Call Gate Descriptor primarily defines the procedure entry point and the procedure’s privilege level.

Table 2.10 Gate Descriptors

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Memory Offset +4																															
OFFSET[31:16]																P	DPL		0	TYPE				0	0	0	PARAMETERS				
Memory Offset +0																															
SELECTOR[15:0]																OFFSET[15:0]															

Table 2.11 Gate Descriptor Bit Definitions

Bit	Memory Offset	Name	Description
31:16 15:0	+4 +0	OFFSET	**Offset** — Used during a call gate to calculate the branch target.
31:16	+0	SELECTOR	**Selector** — Segment selector used during a call gate to calculate the branch target.
15	+4	P	**Segment present**.
14:13	+4	DPL	**Descriptor privilege level**.
11:8	+4	TYPE	**Segment type**: 0100 = 16-bit call gate 0101 = task gate 0110 = 16-bit interrupt gate 0111 = 16-bit trap gate 1100 = 32-bit call gate 1110 = 32-bit interrupt gate 1111 = 32-bit trap gate.
4:0	+4	PARAMETERS	**Parameters** — Number of 32-bit parameters to copy from the caller's stack to the called procedure's stack.

Task Register

The Task Register (TR) holds a 16-bit selector for the current Task State Segment (TSS) table as shown in xxx. The TR is loaded and stored via the LTR and STR instructions, respectively. The TR can only be accessed during protected mode and can only be loaded when the privilege level is 0 (most privileged). When the TR is loaded, the TR selector field indexes a TSS descriptor that must reside in the Global Descriptor Table (GDT). The contents of the selected descriptor are cached on-chip in the hidden portion of the TR

During task switching, the processor saves the current CPU state in the TSS before starting a new task. The TR points to the current TSS. The TSS can be either a 386/486-type 32-bit TSS as shown in Table 2.12 or a 286-type 16-bit TSS type as shown on Table 2.13. An I/O permission bit map is referenced in the 32-bit TSS by the I/O Map Base Address.

15	0
SELECTOR	

Figure 2-2 Task Register

Table 2.12 32-Bit Task State Segment (TSS) Table

31 … 16	15 … 0	
I/O Map Base Address	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 T	+64h
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Selector for Task's LDT	+60h
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	GS	+5Ch
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	FS	+58h
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	DS	+54h
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SS	+50h
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	CS	+4Ch
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	ES	+48h

Table 2.12 32-Bit Task State Segment (TSS) Table (cont.)

31 — 16	15 — 0	
EDI		+44h
ESI		+40h
EBP		+3Ch
ESP		+38h
EBX		+34h
EDX		+30h
ECX		+2Ch
EAX		+28h
EFLAGS		+24h
EIP		+20h
CR3		+1Ch
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SS for CPL = 2	+18h
ESP for CPL = 2		+14h
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SS for CPL = 1	+10h
ESP for CPL = 1		+Ch
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	SS for CPL = 0	+8h
ESP for CPL = 0		+4h
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	Back Link (Old TSS Selector)	+0h

Note: 0 = Reserved

Table 2.13 16-Bit Task State Segment (TSS) Table

15 — 0	
Selector for Task's LDT	+2Ah
DS	+28h
SS	+26h
CS	+24h
ES	+22h
DI	+20h
SI	+1Eh
BP	+1Ch
SP	+1Ah
BX	+18h
DX	+16h
CX	+14h
AX	+12h
FLAGS	+10h
IP	+Eh
SS for Privilege Level 2	+Ch
SP for Privilege Level 2	+Ah
SS for Privilege Level 1	+8h
SP for Privilege Level 1	+6h
SS for Privilege Level 0	+4h
SP for Privilege Level 0	+2h
Back Link (Old TSS Selector)	+0h

Configuration Registers (six)

The processor provides three 8-bit Configuration Control Registers (CCR1, CCR2 and CCR3) used to control the on-chip write-back cache, the coprocessor interface signals and SMM features. The processor also provides two 8-bit internal read-only device identification registers (DIR0 and DIR1) and one 24-bit SMM Address Region Register (SMAR). The CCR, DIR, and SMAR registers exist in I/O memory space and are selected by a “register index” number as listed in ‘Table 2.14 Configuration Register Map’ on page 42.

Access to these registers is achieved by writing the index of the register to I/O port 22h. I/O port 23h is then used for data transfer. Each I/O port 23h data transfer must be preceded by an I/O port 22h register index selection, otherwise the second and later I/O port 23h operations are directed off-chip and produce external I/O cycles. If the register index number is outside the C0h-CFh, FEh-FFh range, external I/O cycles will also occur.

The CCR1 register, Table 2.15 on page 43, controls SMM features and enables SMM and cache interface signals.

The CCR2 register, Table 2.16 on page 44, is used to setup internal cache operation and enable suspend control signals.

The CCR3 register, Table 2.17 on page 44, controls additional SMM features.

The SMAR register, Table 2.18 on page 45, is used to define the location and size of the memory region associated with SMM memory space. The starting address of the SMM address region must be on a block size boundary. For example, a 128 KB block is allowed to have a starting address of 0 KB, 128 KB, 256 KB, etc. The SMM block size must be defined for SMI# to be recognized.

‘Table 2.19 DIR0 Bit Definitions’ on page 45 contains an 8-bit value that defines the device type.

‘Table 2.20 DIR1 Bit Definitions’ on page 45 contains additional device type information. The upper 4 bits of DIR1 represent the stepping number of the device and the lower 4 bits of DIR1 represent the particular revision number of the stepping. Actual values for DIR0 and DIR1 are shown in Table 2.1 "Initialized Register Controls" on page 28.

Table 2.14 Configuration Register Map

Register Index	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Control Registers								
CCR1 (C1h)	**RSVD**			NO_LOCK	MMAC	SMAC	SMI	RPL
CCR2 (C2h)	SUSP	BWRT	BARB	WTI	HALT	LOCK_NW	WBAK	**RSVD**
CCR3 (C3h)	**RSVD**	**RSVD**	**RSVD**	**RSVD**	SMM_MODE	**RSVD**	**RSVD**	**RSVD**
SMM Address Region Registers (24 bits)								
CDh	A31	A30	A29	A28	A27	A26	A25	A24
CEh	A23	A22	A21	A20	A19	A18	A17	A16
CFh	A15	A14	A13	A12	SIZE			
Device ID Registers								
DIR0	DEVID 7	DEVID 6	DEVID 5	DEVID4	DEVID3	DEVID2	DEVID1	DEVID0
DIR1	SID[3:0]				RID[3:0]			
Note: The following register index numbers are reserved for future use: C0h through CFh and FEh, FFh.								

Example

```
; Enable CPU warm reset (WM_RST). See Table 2.16, 'CCR2 Bit Defi-
nitions,' on page 44, and see 'Configuration Registers (six)' on
page 42. Compare North Bridge Configuration Registers in 'I/O
Address Map' on page 114.

        mov   al,   0C2h    ; select CCR2
        out   22h,  al      ; set address pointer to CCR2
        in    al,   23h     ; read data from CCR2
        or    al,   2       ; or in WBAK bit Enable WM_RST
        mov   ah,   al
        mov   al,   0C2h    ; select CCR2 again (see prev page)
        out   22h,  al      ; set address pointer to CCR2
        mov   al,   ah      ; or in bit 1 WBAK
        out   23h,  al      ; write data to CCR2
```

Table 2.15 CCR1 Bit Definitions

Bit	Name	Description
7:5	RSVD	**Reserved.**
4	NO_LOCK	**Negate LOCK#** — If = 1: All bus cycles are issued with LOCK# signal negated except page table accesses. Interrupt acknowledge cycles are executed as locked cycles even though LOCK# is negated. With NO_LOCK set, previously noncacheable locked cycles are executed as unlocked cycles and, therefore, may be cached. This results in higher CPU performance.
3	MMAC	**Main Memory Access** — If = 1: All data accesses which occur within an SMI service routine (or when SMAC = 1) access main memory instead of SMM memory space. If = 0: No effect on access.
2	SMAC	**System Management Memory Access** — If = 1: Any access to addresses within the SMM memory space cause external bus cycles to be issued with SMADS# output active. SMI# input is ignored. If = 0: No effect on access.
1	SMI	**Enable SMM Signals** — If = 1: SMI# input/output signal and SMADS# output signal are enabled. If = 0: SMI# input signal ignored and SMADS# output signal floats.
0	RPL	**Enable RPL Signals** — If = 1: Enable output signals RPLSET(1-0) and RPLVAL#. If = 0: Output signals RPLSET(1-0) and RPLVAL# float.
Note: Bits [4:0] are cleared to 0 at reset.		

Table 2.16 CCR2 Bit Definitions

Bit	Name	Description
7	SUSP	**Enable Suspend Signals** — If = 1: SUSP# input and SUSPA# output are enabled. If = 0: SUSP# input is ignored and SUSPA# output floats.
6	BWRT	**Enable Burst Write Cycles** — If = 1: Enables use of 16-byte burst write-back cycles.
5	BARB	**Enable Cache Coherency on Bus Arbitration** — If = 1: Enable write-back of all dirty cache data when HOLD is requested and prior to asserting HLDA.
4	WT1	**Write-Through Region 1** — If = 1: Forces all writes to the 640 KB to 1 MB address region that hit in the on-chip cache to be issued on the external bus.
3	HALT	**Suspend on HALT** — If = 1: CPU enters suspend mode following execution of a HALT instruction.
2	LOCK_NW	**LOCK NW Bit** — If = 1: Prohibits changing the state of the NW bit in CR0.
1	WBAK	**Enable Write-Back Cache Interface Signals** — If = 1: Enable INVAL and WM_RST input signals, and HITM# output signal. If = 0: INVAL and WM_RST input signals are ignored, and HITM# output signal floats.
0	RSVD	**Reserved.**
Note: All bits are cleared to zero at reset.		

Table 2.17 CCR3 Bit Definitions

Bit	Name	Description
7:4	RSVD	**Reserved.**
3	SMM_MODE	**SL-enhanced compatible mode.** If = 1: SL compatible mode enabled. If = 0: SL compatible mode disabled. **NOTE:** Once the SMI_Lock bit is set, the CPU must be reset in order to modify SMI_Lock and SMM_Mode.
2	RSVD	**Reserved**
1	NMIEN	**NMI Enable** — If = 1: NMI is enabled during SMM. If = 0: NMI is not recognized during SMM.
0	SMI_LOCK	**SMM Register Lock** — If = 1: the following SMM control bits cannot be modified: CCR1 bits: 1, 2, and 3 CCR3 bit: 1 all SMAR bits While operating within an SMI handler, these SMM control bits can be modified. Once set, the SMI_LOCK bit can only be cleared by asserting the RESET signal.
Note: Bits [1:0] are cleared to zero at reset.		

Table 2.18 SMAR Size Field

Bits [3:0]	Block Size	Bits [3:0]	Block Size
0h	Disabled	8h	512 KB
1h	4 KB	9h	1 MB
2h	8 KB	Ah	2 MB
3h	16 KB	Bh	4 MB
4h	32 KB	Ch	8 MB
5h	64 KB	Dh	16 MB
6h	128 KB	Eh	32 MB
7h	256 KB	Fh	4 KB (Same as 1h)

Table 2.19 DIR0 Bit Definitions

Bit	Name	Description
7:0	DEVID[7:0]	**Device Identification** — DEVID[7:0] bits define the CPU type. These bits are read only. processor = xxh

Table 2.20 DIR1 Bit Definitions

Bit	Name	Description
7:4	SID[3:0]	**Stepping Identification** — SID[3:0] are read only and indicate device stepping number.
3:0	RID[3:0]	**Revision Identification** — RID[3:0] are read only and indicate device revision number.

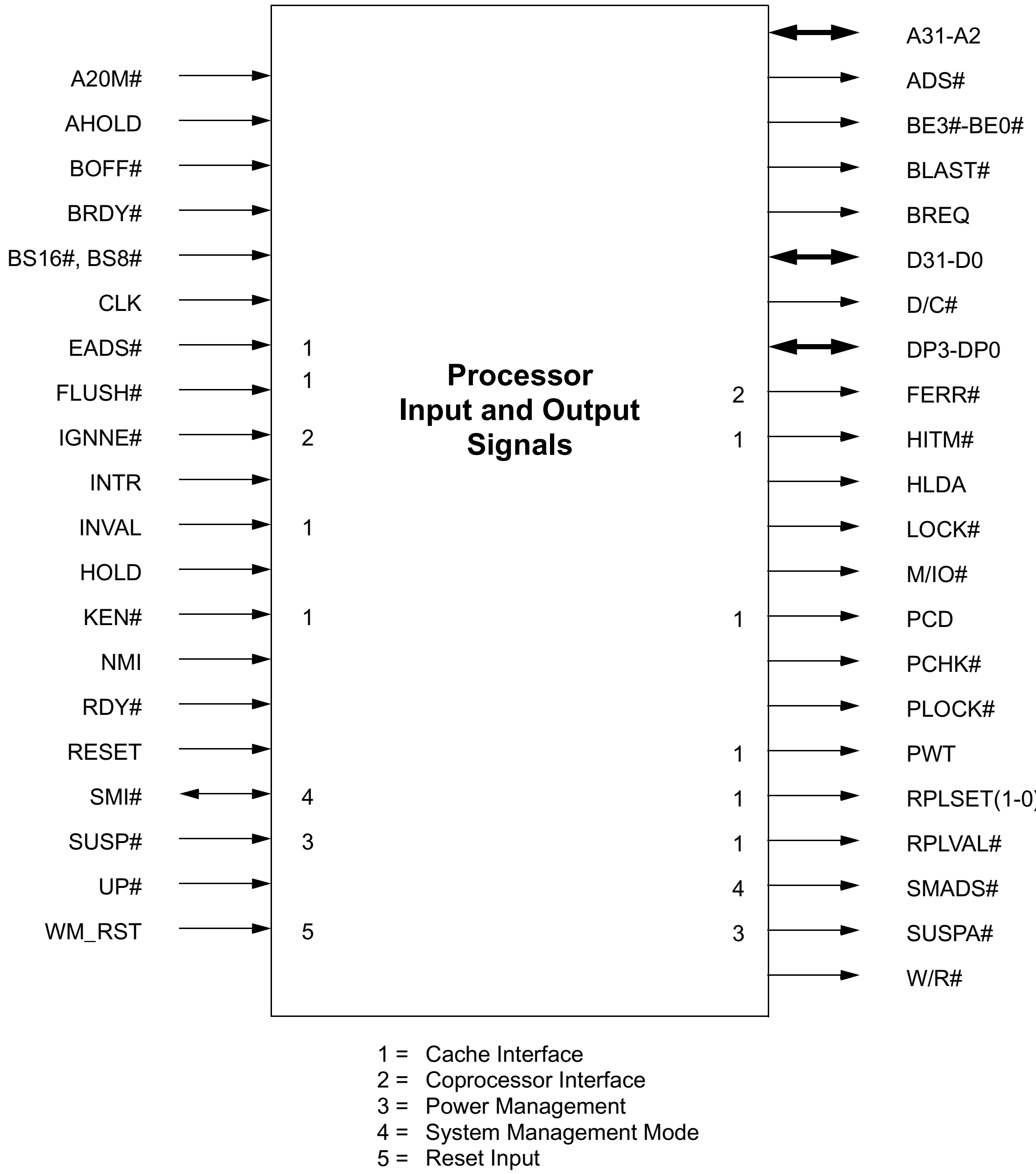


Figure 2-3 Processor Internal I/O Interface Signals

Debug Registers

Six debug registers (DR0-DR3, DR6 and DR7), shown in Table 2.21, support debugging on the processor. Memory addresses loaded in the debug registers, referred to as “breakpoints”, generate a debug exception when a memory access of the specified type occurs to the specified address. A breakpoint can be specified for a particular kind of memory access such as a read or a write. Code and data breakpoints can also be set allowing debug exceptions to occur whenever a given data access (read or write) or code access (execute) occurs. The size of the debug target can be set to 1-, 2-, or 4-bytes. The debug registers are accessed via MOV instructions which can be executed only at privilege level 0.

Debug Address Registers (DR0-DR3) contain the linear address for one of four possible breakpoints. Each breakpoint is specified by bits in Debug Control Register (DR7). For each breakpoint address in DR0-DR3, there are corresponding fields (L, R/W, and LEN) in DR7 that specify the memory access type associated with the breakpoint.

The R/W field can be used to specify instruction execution as well as data access breakpoints. Instruction execution breakpoints are always taken before execution of the instruction that matches the breakpoint.

The Debug Status Register (DR6) reflects conditions that were in effect at the time the debug exception occurred. The contents of the DR6 register are not automatically cleared by the processor after a debug exception occurs and, therefore, should be cleared by software at the appropriate time. Table 2.22 on page 48 lists the field definitions for the DR6 and DR7 registers.

Code execution breakpoints may also be generated by placing the breakpoint instruction (INT 3) at the location where control is to be regained. The single-step feature may be enabled by setting the TF flag in the EFLAGS register. This causes the processor to perform a debug exception after the execution of every instruction

Table 2.21 Debug Registers

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR7 Register	LEN3		R/W3		LEN2		R/W2		LEN1		R/W1		LEN0		R/W0		0	0	GD	0	0	1	GE	LE	G3	L3	G2	L2	G1	L1	G0	L0
DR6 Register	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	BT	BS	0	0	1	1	1	1	1	1	1	1	B3	B2	B1	B0
DR3 Register	Breakpoint 3 Linear Address																															
DR2 Register	Breakpoint 2 Linear Address																															
DR1 Register	Breakpoint 1 Linear Address																															
DR0 Register	Breakpoint 0 Linear Address																															

Note: All bits marked as 0 or 1 are reserved and should not be modified.

Table 2.22 DR6 and DR7 Field Definitions

Register	Field	Number Of Bits	Description
DR6	Bi	1	Bi is set by the processor if the conditions described by DRi, R/Wi, and LENi occurred when the debug exception occurred, even if the breakpoint is not enabled via the Gi or Li bits.
	BT	1	BT is set by the processor before entering the debug handler if a task switch has occurred to a task with the T bit in the TSS set.
	BS	1	BS is set by the processor if the debug exception was triggered by the single-step execution mode (TF flag in EFLAGS set).
DR7	R/Wi	2	Applies to the DRi breakpoint address register: 00 - Break on instruction execution only 01 - Break on data writes only 10 - Not used 11 - Break on data reads or writes.
	LENi	2	Applies to the DRi breakpoint address register: 00 - One byte length 01 - Two byte length 10 - Not used 11 - Four byte length.
	Gi	1	If set to a 1, breakpoint in DRi is globally enabled for all tasks and is not cleared by the processor as the result of a task switch.
	Li	1	If set to 1, breakpoint in DRi is locally enabled for the current task and is cleared by the processor as the result of a task switch.
	GD	1	Global disable of debug register access. GD bit is cleared whenever a debug exception occurs.

Test Registers

The five test registers, shown in Table 2.23, are used to test the CPU's Translation Look-aside Buffer (TLB) and on-chip cache. TR6 and TR7 are used for TLB testing, and TR3-TR5 are used for cache testing. Table 2.24 on page 49 lists the bit definitions for the TR6 and TR7 registers.

The processor TLB is a four-way set associative memory with eight entries per set. Each TLB entry consists of a 24-bit tag and 20-bit data. The 24-bit tag represents the high-order 20 bits of the linear address, a valid bit, and three attribute bits. The 20-bit data portion represents the upper 20 bits of the physical address that corresponds to the linear address.

The TLB Test Control Register (TR6) contains a command bit, the upper 20 bits of a linear address, a valid bit and the attribute bits used in the test operation. The contents of TR6 is used to create the 24-bit TLB tag during both write and read (TLB lookup) test operations. The command bit defines whether the test operation is a read or a write.

The TLB Test Data Register (TR7) contains the upper 20 bits of the physical address (TLB data field), three LRU bits and a control bit. During TLB write operations, the physical address in TR7 is written into the TLB entry selected by the contents of TR6. During TLB lookup operations, the TLB data selected by the contents of TR6 is loaded into TR7.

Table 2.23 Test Registers

Register	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR7 Register	TLB Physical Address																				PCD	PWT	TLB LRU			0	0	PL	REP		0	0
TR6 Register	TLB Linear Address																				V	D	D#	U	U#	R	R#	0	0	0	0	C
TR5 Register	RSVD																				Line Selection							Set/ DWORD			CTL	
TR4 Register	Cache Tag Address																				0	V	Cache LRU Bits			Dirty Bits				RSVD		0
TR3 Register	Cache Data																															

Table 2.24 TR7 and TR6 Bit Definitions

Register Name	Bit	Description
TR7	31:12	Physical address. TLB lookup: data field from the TLB. TLB write: data field written into the TLB.
	11	Page-level cache disable bit (PCD). Corresponds to the PCD bit of a page table entry.
	10	Page-level cache write-through bit (PWT). Corresponds to the PWT bit of a page table entry.
	9:7	LRU bits. TLB lookup: LRU bits associated with the TLB entry prior to the TLB lookup. TLB write: ignored.
	4	PL bit. TLB lookup: If = 1, read hit occurred. If = 0, read miss occurred. TLB write: If = 1, REP field is used to select the set. If = 0, the pseudo-LRU replacement algorithm is used to select the set.
	3:2	Set selection (REP). TLB lookup: If PL = 1, set in which the tag was found. If PL = 0, undefined data. TLB write: If PL = 1, selects one of the four sets for replacement. If PL = 0, ignored.

Table 2.24 TR7 and TR6 Bit Definitions

Register Name	Bit	Description
TR6	31:12	Linear address. TLB lookup: The TLB is interrogated per this address. If one and only one match occurs in the TLB, the rest of the fields in TR6 and TR7 are updated per the matching TLB entry. TLB write: A TLB entry is allocated to this linear address.
	11	Valid bit (V). TLB write: If set, indicates that the TLB entry contains valid data. If clear, target entry is invalidated.
	10:9	Dirty attribute bit and its complement (D, D#). Refer to Table 2-17 on page 30.
	8:7	User/supervisor attribute bit and its complement (U, U#). Refer to Table 2-17 on page 30.
	6:5	Read/write attribute bit and its complement (R, R#). Refer to Table 2-17 on page 30.
	0	Command bit (C). If = 0: TLB write. If = 1: TLB lookup.

Table 2.25 TR6 Attribute Bit Pairs

Bit (D, U or R)	Bit Complement (D#, U#, or R#)	Effect On TLB Lookup	Effect On TLB Write
0	0	Do not match.	Undefined.
0	1	Match if D, U or R bit = 0.	Clear the bit.
1	0	Match if D, U or R bit = 1.	Set the bit.
1	1	Match if D, U or R bit = either 1 or 0.	Undefined.

Cache Test Registers

The processor 8 KB on-chip cache is a four-way set associative memory that can be configured as either write-back or write-through. Each cache set contains 128 entries. Each entry consists of a 21-bit tag address, a 16-byte data field, a valid bit, and four dirty bits.

The 21-bit tag represents the high-order 21 bits of the physical address. The 16-byte data represents the 16 bytes of data currently in memory at the physical address represented by the tag. The valid bit indicates whether the data bytes in the cache actually contain valid data. The four dirty bits indicate if the data bytes in the cache have been modified internally without updating external memory (write-back configuration). Each dirty bit indicates the status for one double-word (4 bytes) within the 16-byte data field.

For each line in the cache there are three LRU bits that indicate which of the four sets was most recently accessed. A line is selected using bits10-4 of the physical address. Figure 2-4 illustrates the processor cache architecture.

The processor contains three test registers that allow testing of its internal cache. Bit definitions for the cache test registers are shown in Table 2.26. Using these registers, cache writes and reads may be performed.

Cache test writes cause the data in the cache fill buffer to be written to the selected set and entry in the cache. Data must be written to TR3 (32-bit register) four times in order to fill the cache fill buffer. Once the cache fill buffer has been loaded, a cache test write can be performed. For data to be written to the allocated entry, the valid bit for the entry must be set prior to the write of the data.

Cache test reads cause the data in the selected set and entry to be loaded into the cache flush buffer. Once the buffer has been loaded, data must be read from TR3 four times in order to empty the cache flush buffer. For proper operation, cache tests should be performed only when the cache is disabled (CD bit in CR0 = 1).

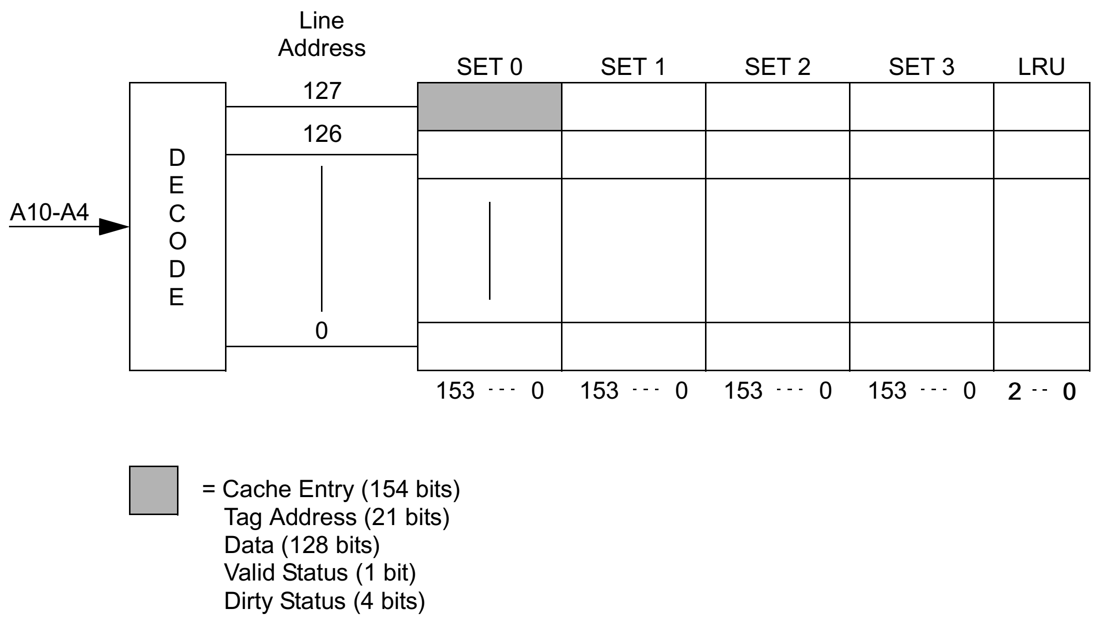


Figure 2-4   Processor Cache Architecture

Table 2.26 TR3-TR5 Bit Definitions

Bit	Name	Description
31:0	TR3	**Cache data** — Flush buffer read: data accessed from the cache flush buffer. Fill buffer write: data to be written into the cache fill buffer.
31:12	TR4	**Upper Tag Address** — Cache read: upper 21 bits of tag address of the selected entry. Cache write: data written into the upper 21 bits of the tag address of the selected entry.
10		**Valid Bit** — Cache read: valid bit for the selected entry. Cache write: data written into the valid bit for the selected entry.
9:7		**LRU Bits** — Cache read: the LRU bits for the selected line. xx1 = Set 0 or Set 1 most recently accessed. xx0 = Set 2 or Set 3 most recently accessed. x1x = Most recent access to Set 0 or Set 1 was to Set 0. x0x = Most recent access to Set 0 or Set 1 was to Set 1. 1xx = Most recent access to Set 2 or Set 3 was to Set 2. 0xx = Most recent access to Set 2 or Set 3 was to Set 3. Cache write: ignored.
6:3		**Dirty Bits** — Cache read: the dirty bits for the selected entry (one bit per DWORD). Cache write: data written into the dirty bits for the selected entry.
10:4	TR5	**Line Selection** — Physical address bits 10:4 used to select one of 128 lines.
3:2		**Set/DWORD Selection** — Cache read: selects which of the four sets is used as the source for data transferred to the cache flush buffer. Cache write: selects which of the four sets is used as the destination for data transferred from the cache fill buffer. Flush buffer read: selects which of the four Words in the flush buffer is loaded into TR3. Fill buffer write: selects which of the four DWORDs in TR3 is written to the fill buffer.
1:0		**Control Bits** — If = 00: flush read or fill buffer write. Writing to TR3 fill buffer write. Reading TR3 initiates flush buffer read. If = 01: cache write. If = 10: cache read. If = 11: cache flush.

2.2.4. Address Spaces

The CPU can directly address either memory or I/O space. Figure 2-5 illustrates the range of addresses available for memory address space and I/O address space. For the processor, the addresses for physical memory range between 0000 0000h and FFFF FFFFh (4 GB). However, the address bus capability of the ZFx86 limits external memory address space to 256 MB.The accessible I/O addresses space ranges between 0000 0000h and 0000 FFFFh (64 KB). The processor does not use coprocessor communication space in upper I/O space between 8000 00F8h and 8000 00FFh as do the 386-style CPU's. The I/O locations 22h and 23h are used for the processor configuration register access.

2.2.4.1. I/O Address Space

The processor I/O address space is accessed using IN and OUT instructions to addresses referred to as "ports". The accessible I/O address space is 64 KB and can be accessed as 8-bit, 16-bit or 32-bit ports. The execution of any IN or OUT instruction causes the M/IO# signal to be driven low, thereby selecting the I/O space instead of memory space.

The processor configuration registers reside within the I/O address space at port addresses 22h and 23h and are accessed using standard IN and OUT instructions. The configuration registers are modified by writing the index of the configuration register to port 22h, then transferring the data through port 23h. Accesses to the on-chip configuration registers do not generate external I/O cycles. Each port 23h operation must be preceded by a port 22h write with a valid index value. Otherwise, the second and later port 23h operations are directed off-chip and generate external I/O cycles without modifying the on-chip configuration registers. Writes to port 22h outside of the processor index range (C0h-CFh and FEh-FFh) result in external I/O cycles and do not affect the on-chip configuration registers. Reads of port 22h are always directed off-chip.

2.2.4.2. Memory Address Space

The processor directly addresses up to 4 GB of physical memory. However, the address bus capability of the ZFx86 limits external memory address space to 256 MB. Memory address space is accessed as bytes, WORDS (16-bits) or DWORDs (32-bits). WORDS and DWORDs are stored in consecutive memory bytes with the low-order byte located in the lowest address. The physical address of a word or DWORD is the byte address of the low-order byte.

Memory can be addressed using nine different addressing modes. These addressing modes are used to calculate an offset address often referred to as an effective address. Depending on the operating mode of the CPU, the offset is then combined using memory management mechanisms to create a physical address that actually addresses the physical memory devices.

Memory management mechanisms on the CPU consist of segmentation and paging. Segmentation allows each program to use several independent, protected address spaces. Paging supports a memory subsystem that simulates a large address space using a small amount of RAM and disk storage for physical memory. Either or both of these mechanisms can be used for management of the processor memory address space

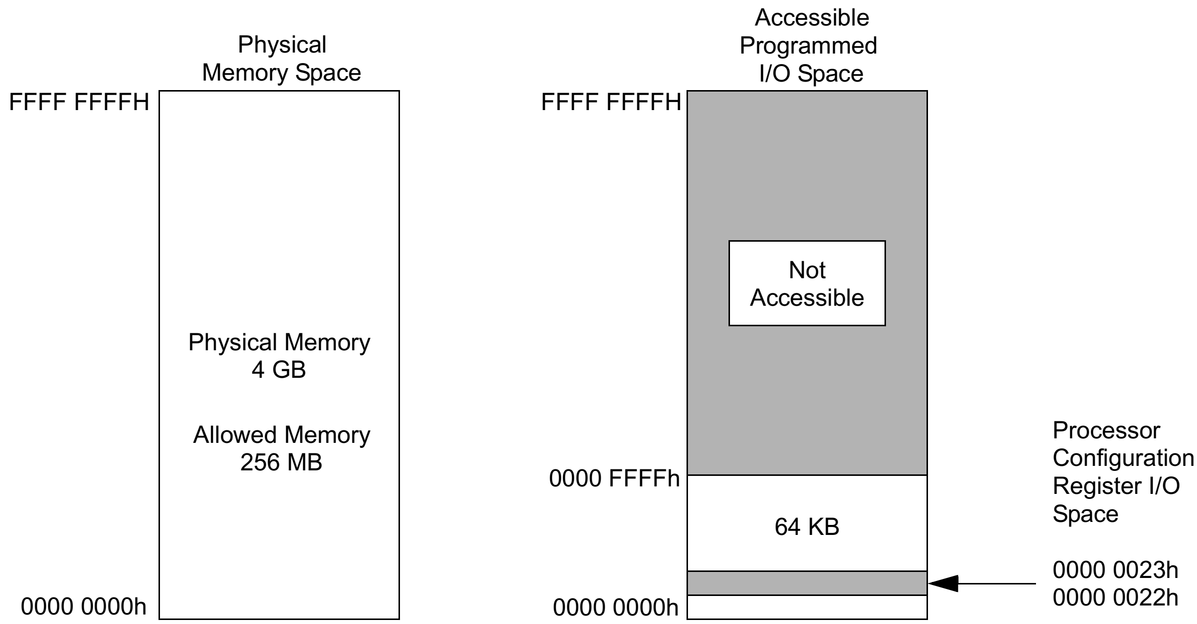


Figure 2-5 Memory and I/O Address Spaces

Offset Mechanism

The offset mechanism computes an offset (effective) address by adding together up to three values: a base, an index and a displacement. The base, if present, is the value in one of eight 32-bit general registers at the time of the execution of the instruction. The index, like the base, is a value that is contained in one of the 32-bit general registers (except the ESP register) when the instruction is executed. The index differs from the base in that the index is first multiplied by a scale factor of 1, 2, 4 or 8 before the summation is made. The third component added to the memory address calculation is the displacement which is a value of up to 32-bits in length supplied as part of the instruction. See Figure 2-6 "Offset Address Calculation".

Nine valid combinations of the base, index, scale factor and displacement can be used with the processor instruction set. These combinations are listed in Table 2.27. The base and index both refer to contents of a register as indicated by [Base] and [Index]

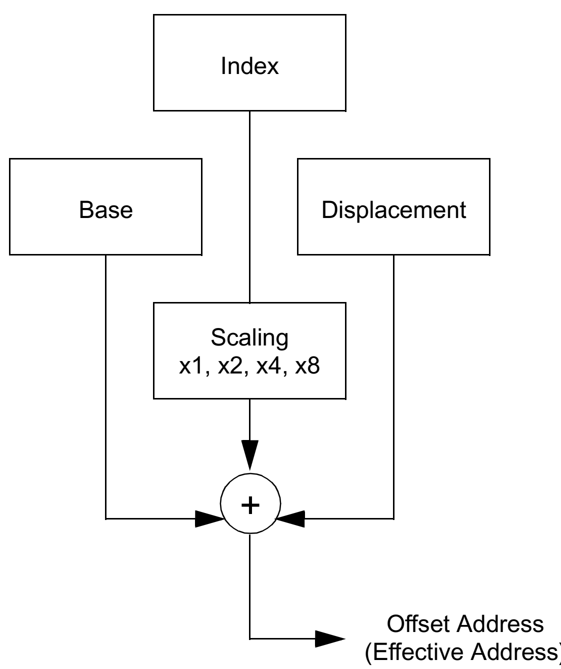


Figure 2-6 Offset Address Calculation

Table 2.27 Memory Addressing Modes

Addressing Mode	Base	Index	Scale Factor	Displacement	Offset Address (OA) Calculation
Direct				x	OA = DP
Register Indirect	x				OA = [BASE]
Based	x			x	OA = [BASE] + DP
Index		x		x	OA = [INDEX] + DP
Scaled Index		x	x	x	OA = ([INDEX] * SF) + DP
Based Index	x	x			OA = [BASE] + [INDEX]
Based Scaled Index	x	x	x		OA = [BASE] + ([INDEX] * SF)
Based Index with Displacement	x	x		x	OA = [BASE] + [INDEX] + DP
Based Scaled Index with Displacement	x	x	x	x	OA = [BASE] + ([INDEX] * SF) + DP

Real Mode Memory Addressing

In real mode operation, the CPU only addresses the lowest 1 MB of memory. To calculate a physical memory address, the 16-bit segment base address located in the selected segment register is multiplied by 16 and then the 16-bit offset address is added. The resulting 20-bit address is then extended with twelve zeros in the upper address bits to create the 32-bit physical address. Figure 2-13 illustrates the real mode address calculation.

The addition of the base address and the offset address may result in a carry. Therefore, the resulting address may actually contain up to 21 significant address bits that can address memory in the first 64 KB above 1 MB.

Protected Mode Memory Addressing

In protected mode three mechanisms calculate a physical memory address:

- Offset Mechanism that produces the offset or effective address as in real mode.
- Selector Mechanism that produces the base address.
- Optional Paging Mechanism that translates a linear address to the physical memory address.

The offset and base address are added together to produce the linear address. If paging is not used, the linear address is used as the physical memory address. If paging is enabled, the paging mechanism is used to translate the linear address into the physical address. The offset mechanism is described in ‘Offset Mechanism’ on page 54. and applies to both real and protected mode. The selector and paging mechanisms are described in the following paragraphs.

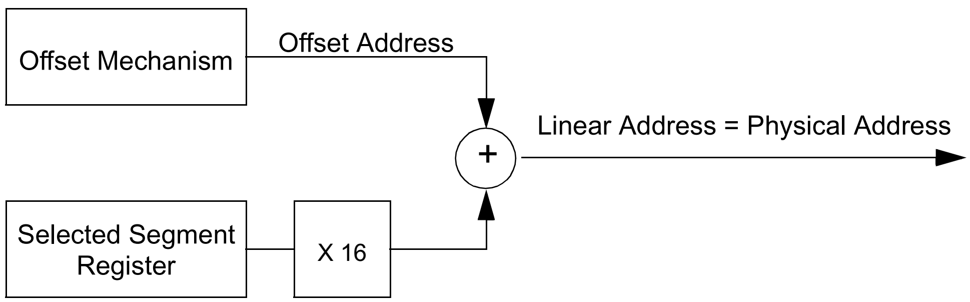


Figure 2-7 Real Mode Address Calculation

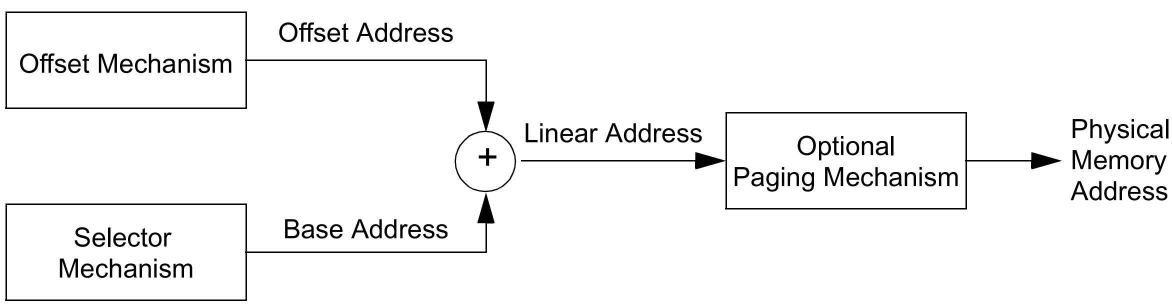


Figure 2-8 Protected Mode Address Calculation

Selector Mechanism

Memory is divided into an arbitrary number of segments, each containing less than the 2^{32} byte (4 GB) maximum.

The six segment registers (CS, DS, SS, ES, FS and GS) each contain a 16-bit selector that is used when the register is loaded to locate a segment descriptor in either the global descriptor table (GDT) or the local descriptor table (LDT). The segment descriptor defines the base address, limit and attributes of the selected segment and is cached on the processor as a result of loading the selector.

The cached descriptor contents are not visible to the programmer. When a memory reference occurs in protected mode, the linear address is generated by adding the segment base address in the hidden portion of the segment register to the offset address. If paging is not enabled, this linear address is used as the physical memory address. Figure 2-9 "Selector Mechanism" illustrates the operation of the selector mechanism.

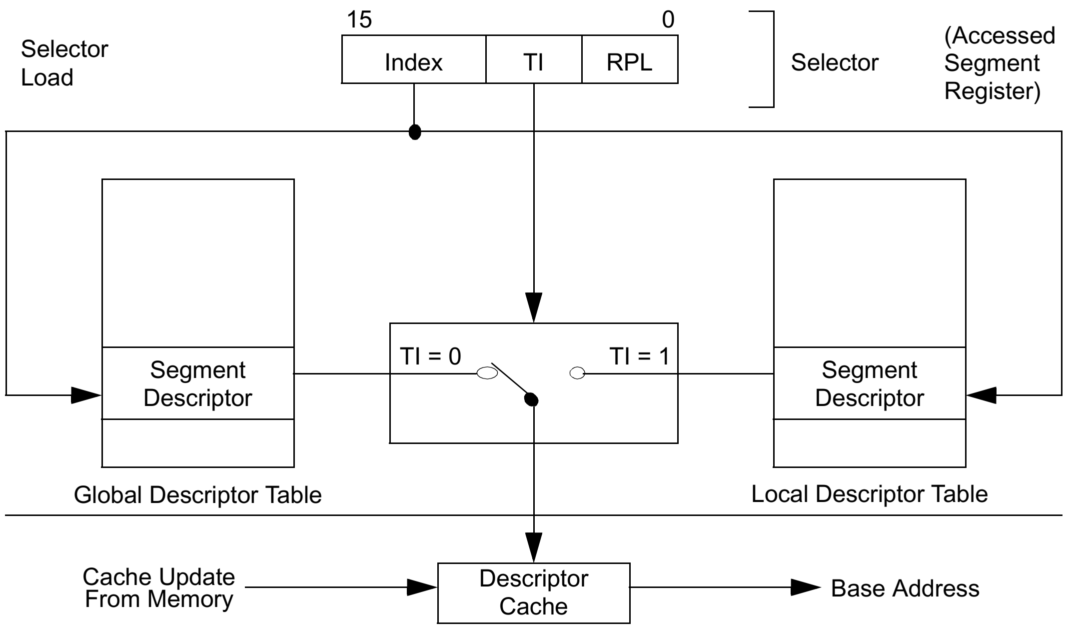


Figure 2-9 Selector Mechanism

Paging Mechanism

The paging mechanism supports a memory subsystem that simulates a large address space with a small amount of RAM and disk storage. The paging mechanism either translates a linear address to its corresponding physical address or generates an exception if the required page is not currently present in RAM. When the operating system services the exception, the required page is loaded into memory and the instruction is then restarted. Pages are either 4 KB or 1 MB in size. The CPU defaults to 4 KB pages that are aligned to 4 KB boundaries.

A page is addressed by using two levels of tables as illustrated in Figure 2-10 on page 58. The upper 10 bits of the 32-bit linear address are used to locate an entry in the page directory table. The page directory table acts as a 32-bit master index to up to 1 KB individual second-level page tables. The selected entry in the page directory table, referred to as the directory table entry, identifies the starting address of the second-level page table. The page directory table itself is a page and is, therefore, aligned to a 4 KB boundary. The physical address of the current page directory table is stored in the CR3 control register, also referred to as the Page Directory Base Register (PDBR).

Bits 21:12 of the 32-bit linear address, the Page Table Index, locate a 32-bit entry in the second-level page table. The Page Table Entry (PTE) contains the base address of the page frame. The second-level page table addresses up to 1 KB individual page frames. A second-level page table is 4 KB in size and is itself a page. The lower 12 bits of the 32-bit linear address, the Page Frame Offset (PFO), locate the desired physical data within the page frame.

Since the page directory table can point to 1 KB page tables, and each page table can point to 1 KB of page frames, a total of 1 MB of page frames can be implemented. Since each page frame contains 4 KB, up to 4 GB of virtual memory can be addressed by the processor with a single page directory table.

In addition to the base address of the page table or the page frame, each directory table entry or page table entry contains attribute bits and a Present (P) Flag bit as illustrated in Figure 2-10 and listed in Table 2.28.

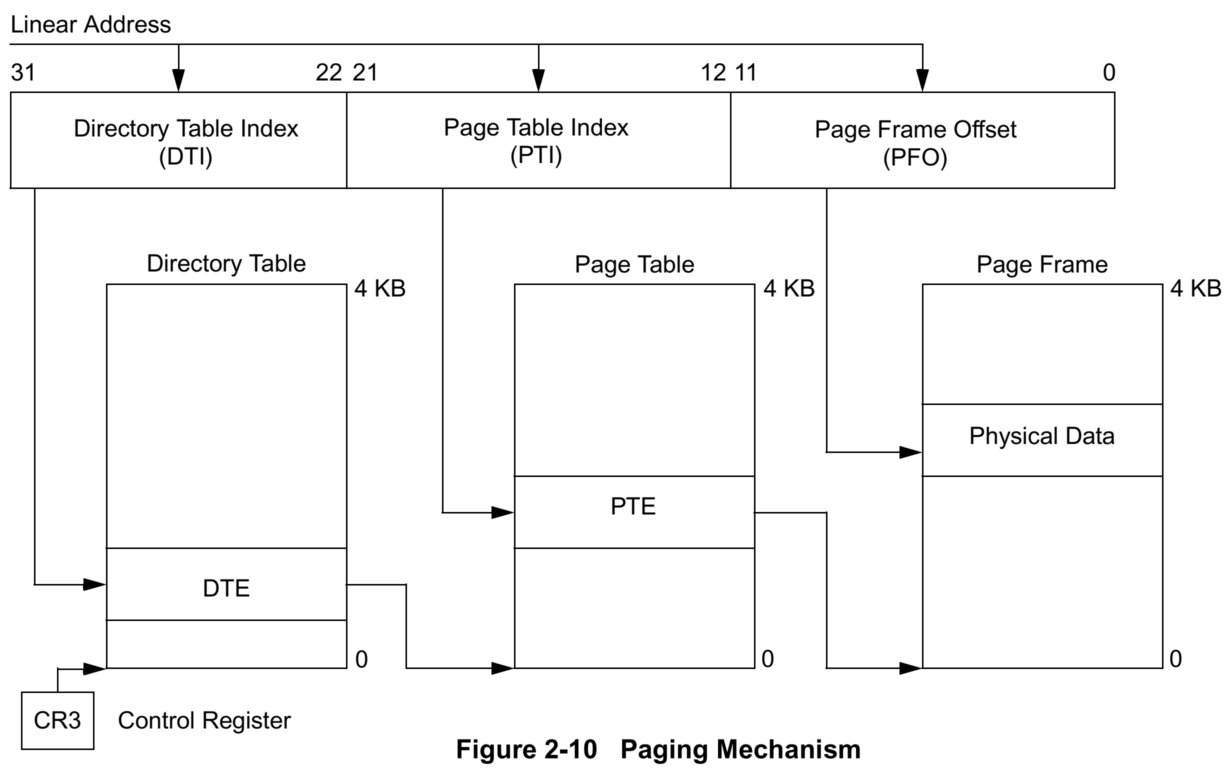


Figure 2-10 Paging Mechanism

If the P bit is set in the DTE, the page table is present and the appropriate page table entry is read. If P = 1 in the corresponding PTE (indicating that the page is in memory), the accessed and dirty bits are updated, if necessary, and the operand is fetched. Both accessed bits are set (DTE and PTE), if necessary, to indicate that the table and the page have been used to translate a linear address. The dirty bit (D) is set before the first write is made to a page.

The P bits must be set to validate the remaining bits in the DTE and PTE. If either of the P bits is not set, a page fault is generated when the DTE or PTE is accessed. If P = 0, the remaining DTE/PTE bits are available for use by the operating system. For example, the operating system can use these bits to record where on the hard disk the pages are located. A page fault is also generated if the memory reference violates the page protection attributes.

Translation Look-Aside Buffer

The translation look-aside buffer (TLB) is a cache for the paging mechanism and replaces the two-level page table lookup procedure for TLB hits. The TLB is a four-way set associative 32-entry page table cache that automatically keeps the most commonly used page table entries in the processor. The 32-entry TLB, coupled with a 4 KB page size, results in coverage of 128 KB of memory addresses.

The TLB must be flushed when entries in the page tables are changed. The TLB is flushed whenever the CR3 register is loaded. An individual entry in the TLB can be flushed using the INVLPG instruction.

Table 2.28 Directory and Page Table Entry (DTE and PTE) Bit Definitions

Bit	Name	Description
31:12	Base Address	Specifies the base address of the page or page table.
11:9	--	Undefined and available to the programmer.
8:7	RSVD	Reserved and not available to the programmer.
6	D	**Dirty Bit** — If set, indicates that a write access has occurred to the page (PTE only; undefined in DTE).
5	A	**Accessed Flag** — If set, indicates that a read access or write access has occurred to the page.
4	PCD	**Page Caching Disable Flag** — If set, indicates that the page is not cacheable in the on-chip cache.
3	PWT	**Page Write-Through Flag** — If set, indicates that writes to the page or page tables that hit in the on-chip cache must update both the cache and external memory.
2	U/S	**User/Supervisor Attribute** — If set (user), page is accessible at privilege level 3. If clear (supervisor), page is accessible only when CPL ≤ 2.
1	W/R	**Write/Read Attribute** — If set (write), page is writable. If clear (read), page is read only.
0	P	**Present Flag** — If set, indicates that the page is present in RAM, and validates the remaining DTE/PTE bits. If clear, indicates that the page is not present in memory and the remaining DTE/PTE bits can be used by the programmer.

2.2.5. Interrupts and Exceptions

The processing of either an interrupt or an exception changes the normal sequential flow of a program by transferring program control to a selected service routine. Except for SMM interrupts, the location of the selected service routine is determined by one of the interrupt vectors stored in the interrupt descriptor table.

True interrupts are hardware interrupts and are generated by signal sources external to the CPU. All exceptions (including so-called software interrupts) are produced internally by the CPU.

2.2.5.1. Interrupts

External events can interrupt normal program execution by using one of the three interrupt signals on the CPU.

- Non-maskable Interrupt (NMI signal)
- Maskable Interrupt (INTR signal)
- SMM Interrupt (SMI# signal).

For most interrupts, program transfer to the interrupt routine occurs after the current instruction has been completed. When the execution returns to the original program, it begins immediately following the interrupted instruction.

The NMI interrupt cannot be masked by software and always uses interrupt vector 2 to locate its service routine. Since the interrupt vector is fixed and is supplied internally, no interrupt acknowledge bus cycles are performed. This interrupt is normally reserved for unusual situations such as parity errors and has priority over INTR interrupts.

Once NMI processing has started no additional NMIs are processed until an IRET instruction is executed, typically at the end of the NMI service routine. If NMI is re-asserted prior to execution of the IRET instruction, one and only one NMI rising edge is stored and then processed after execution of the next IRET.

During the NMI service routine, maskable interrupts may be enabled. If an unmasked INTR occurs during the NMI service routine, the INTR is serviced and execution returns to the NMI service routine following the next IRET. If a HALT instruction is executed within the NMI service routine, the processor restarts execution only in response to RESET, an unmasked INTR or an SMM interrupt. NMI does not restart CPU execution under this condition.

The INTR interrupt is unmasked when the Interrupt Enable Flag (IF) in the EFLAGS register is set to 1. With the exception of string operations, INTR interrupts are acknowledged between instructions. Long string operations have interrupt windows between memory moves that allow INTR interrupts to be acknowledged.

When an INTR interrupt occurs, the CPU performs two locked interrupt acknowledge bus cycles. During the second cycle, the CPU reads an 8-bit vector which is supplied by an external interrupt controller. This vector selects which of the 256 possible interrupt handlers will be executed in response to the interrupt.

The SMM interrupt has higher priority than either INTR or NMI. After SMI# is asserted, program execution is passed to an SMI service routine which runs in SMM address space reserved for this purpose. The remainder of this section does not apply to the SMM interrupts. SMM interrupts are described in greater detail in Section 2.2.5.4. 'Interrupt and Exception Priorities' on page 62.

2.2.5.2.Exceptions

Exceptions are generated by an interrupt instruction or a program error. Exceptions are classified as traps, faults or aborts depending on the mechanism used to report them and the ability to restart of the instruction which first caused the exception.

A Trap Exception is reported immediately following the instruction that generated the trap exception. Trap exceptions are generated by execution of a software interrupt instruction (INTO, INT 3, INT n, BOUND), by a single- step operation or by a data breakpoint.

Software interrupts can be used to simulate hardware interrupts. For example, an INT n instruction causes the processor to execute the interrupt service routine pointed to by the nth vector in the interrupt table. Execution of the interrupt service routine occurs regardless of the state of the IF flag in the EFLAGS register.

The one byte INT 3, or breakpoint interrupt (vector 3), is a particular case of the INT n instruction. By inserting this one byte instruction in a program, the user can set breakpoints in the code that can be used during debug.

Single-step operation is enabled by setting the TF bit in the EFLAGS register. When TF is set, the CPU generates a debug exception (vector 1) after the execution of every instruction. Data breakpoints also generate a debug exception and are specified by loading the debug registers (DR0-DR7) with the appropriate values.

A Fault Exception is reported prior to completion of the instruction that generated the exception. By reporting the fault prior to instruction completion, the CPU is left in a state which allows the instruction to be restarted and the effects of the faulting instruction to be nullified. Fault exceptions include divide-by-zero errors, invalid opcodes, page faults and coprocessor errors. Debug exceptions (vector 1) are also handled as faults (except for data breakpoints and single-step operations). After execution of the fault service routine, the instruction pointer points to the instruction that caused the fault.

An Abort Exception is a type of fault exception that is severe enough that the CPU cannot restart the program at the faulting instruction. The double fault (vector 8) is the only abort exception that occurs on the processor.

2.2.5.3.Interrupt Vectors

When the CPU services an interrupt or exception, the current program's instruction pointer and flags are pushed onto the stack to allow resumption of execution of the interrupted program. In protected mode, the processor also saves an error code for some exceptions. Program control is then transferred to the interrupt handler (also called the interrupt service routine). Upon execution of an IRET at the end of the service routine, program execution resumes at the instruction pointer address saved on the stack when the interrupt was serviced.

Interrupt Vector Assignments

Each interrupt (except SMI#) and exception is assigned one of 256 interrupt vector numbers listed in Table 2.29. The first 32 interrupt vector assignments are defined or reserved. INT instructions acting as software interrupts may use any of interrupt vectors, 0 through 255.

The non-maskable hardware interrupt (NMI) is assigned vector 2. Illegal opcodes including faulty FPU instructions will cause an invalid opcode fault, Interrupt Vector 6.

Table 2.29 Interrupt Vector Assignments

Interrupt Vectors	Function	Exception Type
0	Divide error	FAULT
1	Debug exception	TRAP/FAULT (see note)
2	NMI interrupt	
3	Breakpoint	TRAP
4	Interrupt on overflow	TRAP
5	BOUND range exceeded	FAULT
6	Invalid opcode	FAULT
7	Device not available	FAULT
8	Double fault	ABORT
9	Reserved	
10	Invalid TSS	FAULT
11	Segment not present	FAULT
12	Stack fault	FAULT
13	General protection fault	TRAP/FAULT
14	Page fault	FAULT
15	Reserved	
16	FPU error	FAULT
17	Alignment check exception	FAULT
18-31	Reserved	
32-255	Maskable hardware interrupts	TRAP
0-255	Programmed interrupt	TRAP

Note: Data breakpoints and single steps are traps. All are debug exceptions are faults.

In response to a maskable hardware interrupt (INTR), the processor issues interrupt acknowledge bus cycles used to read the vector number from external hardware. These vectors should be in the range 32-255 as vectors 0-31 are pre-defined.

Interrupt Descriptor Table

The interrupt vector number is used by the CPU to locate an entry in the interrupt descriptor table (IDT). In real mode, each IDT entry consists of a 4-byte far pointer to the beginning of the corresponding interrupt service routine. In protected mode, each IDT entry is an 8-byte descriptor. The Interrupt Descriptor Table Register (IDTR) specifies the beginning address and limit of the IDT. Following RESET, the IDTR contains a base address of 0h with a limit of 3FFh.

The IDT can be located anywhere in physical memory as determined by the IDTR register. The IDT may contain different types of descriptors: interrupt gates, trap gates and task gates. Interrupt gates are used primarily to enter a hardware interrupt handler. Trap gates are generally used to enter an exception handler or software interrupt handler. If an interrupt gate is used, the Interrupt Enable Flag (IF) in the EFLAGS register is cleared before the interrupt handler is entered. Task gates are used to make the transition to a new task.

2.2.5.4. Interrupt and Exception Priorities

As the processor executes instructions, it follows a consistent policy for prioritizing exceptions and hardware interrupts. The priorities for competing interrupts and exceptions are listed in Table 2-31 “Interrupt and Exception Priorities”. SMM interrupts always take precedence. Debug traps for the previous instruction and next instructions are handled as the next priority. When NMI and maskable INTR interrupts are both detected at the same instruction boundary, the processor services the NMI interrupt first.

The processor checks for exceptions in parallel with instruction decoding and execution. Several exceptions can result from a single instruction. However, only one exception is generated upon each attempt to execute the instruction. Each exception service routine should make the appropriate corrections to the instruction and then restart the instruction. In this way, exceptions can be serviced until the instruction executes properly.

The processor supports instruction restart after all faults, except when an instruction causes a task switch to a task whose task state segment (TSS) is partially not present. A TSS can be partially not present if the TSS is not page aligned and one of the pages where the TSS resides is not currently in memory.

Table 2.30 Interrupt and Exception Priorities

Priority	Description	Notes
0	SMM hardware interrupt.	SMM interrupts are caused by SMI# asserted and always have highest priority.
1	Debug traps and faults from previous instruction.	Includes single-step trap and data breakpoints specified in the debug registers.
2	Debug traps for next instruction.	Includes instruction execution breakpoints specified in the debug registers.
3	Non-maskable hardware interrupt.	Caused by NMI asserted.
4	Maskable hardware interrupt.	Caused by INTR asserted and IF = 1.
5	Faults resulting from fetching the next instruction.	Includes segment not present, general protection fault and page fault.
6	Faults resulting from instruction decoding.	Includes illegal opcode, instruction too long, or privilege violation.

Table 2.30 Interrupt and Exception Priorities

Priority	Description	Notes
7	WAIT instruction and TS = 1 and MP = 1.	Device not available. Exception generated.
8	ESC instruction and EM = 1 or TS = 1.	Device not available. Exception generated.
9	Floating point error exception.	Caused by unmasked floating point exception with NE = 1.
10	Segmentation faults (for each memory reference required by the instruction) that prevent transferring the entire memory operand.	Includes segment not present, stack fault, and general protection fault.
11	Page Faults that prevent transferring the entire memory operand.	
12	Alignment check fault.	

2.2.5.5. Exceptions in Real Mode

Many of the exceptions described in the 'Interrupt and Exception Priorities' on page 62 are not applicable in real mode. Exceptions 10, 11, and 14 do not occur in real mode. Other exceptions have slightly different meanings in real mode as listed in 'Exception Changes in Real Mode' on page 63

Table 2.31 Exception Changes in Real Mode

Vector	Protected Mode Function	Real Mode Function
8	Double fault	Interrupt table limit overrun.
10	Invalid TSS	--
11	Segment not present	--
12	Stack fault	SS segment limit overrun
13	General protection fault	CS, DS, ES, FS, GS segment limit overrun
14	Page fault	--
Note: -- means "does not occur".		

2.2.5.6. Error Codes

When operating in protected mode, the following exceptions generate a 16-bit error code:

- Double Fault
- Alignment Check
- Invalid TSS
- Segment Not Present
- Stack Fault
- General Protection Fault
- Page Fault

The error code format is shown in Figure 2-18 and the error code bit definitions are listed in Table 2-33. Bits [15:3] (selector index) are not meaningful if the error code was generated as the result of a page fault. The error code is always zero for double faults and alignment check exceptions.

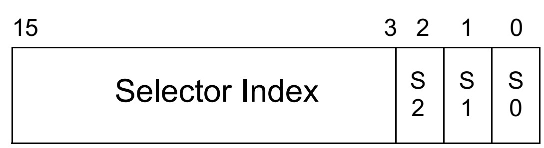


Figure 2-11 Error Code Format

Table 2.32 Error Code Bit Definitions

Fault Type	Selector Index (Bits [15:3])	S2 (Bit 2)	S1 (Bit 1)	S0 (Bit 0)
Page Fault	Reserved	Fault caused by: 0 = not present page 1 = page-level protection violation	Fault occurred during: 0 = read access 1 = write access	Fault occurred during: 0 = supervisor access 1 = user access
IDT Fault	Index of faulty IDT selector	Reserved	1	If = 1, exception occurred while trying to invoke exception or hardware interrupt handler.
Segment Fault	Index of faulty selector	TI bit of faulty selector	0	If = 1, exception occurred while trying to invoke exception or hardware interrupt handler.

2.2.6. System Management Mode

System Management Mode (SMM) provides an additional interrupt which can be used for system power management or software transparent emulation of I/O peripherals. SMM is entered using the System Management Interrupt (SMI#) that has a higher priority than any other interrupt, including NMI. An SMI interrupt can also be triggered via the software using an SMINT instruction. After an SMI interrupt, portions of the CPU state are automatically saved, SMM is entered, and program execution begins at the base of SMM address space (Figure 2-12). Running in protected SMM address space, the interrupt routine does not interfere with the operating system or any application program.

Eight SMM instructions are included in the processor instruction set that permit software initiated SMM, and saving and restoring of the total CPU state when in SMM mode. The signals SMI# and SMADS# support SMM functions.

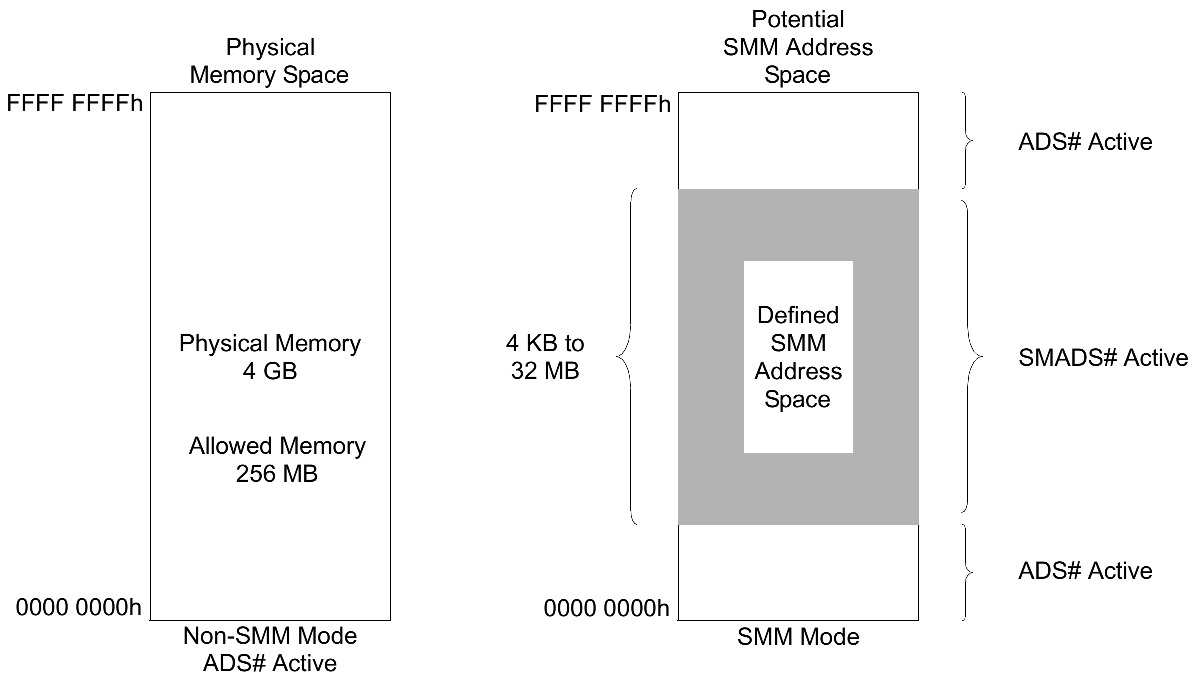


Figure 2-12   System Management Memory Address Space

2.2.6.1. SMM Operation

SMM operation is summarized in Figure 2-20. Entering SMM requires the assertion of the SMI# signal for at least two CLK periods or execution of the SMINT instruction. For the SMI# or SMINT instruction to be recognized, the following configuration register bits must be set as shown in Table 2.33. The configuration registers are discussed in detail in Section "Configuration Registers" on page 25.

Table 2.33 Requirement for Recognizing SMI# and SMINT

Register (Bit)		SMI#	SMINT
SMI	CCR1 [1]	1	1
SMAC	CCR1 [2]	0	1
SMAR	SIZE [3-0]	> 0	> 0

After recognizing SMI# or SMINT and prior to executing the SMI service routine, some of the CPU state information is changed. Prior to modification, this information is automatically saved in the SMM memory space header located at the top of SMM memory space. After the header is saved, the CPU enters real mode and begins executing the SMI service routine starting at the SMM memory base address.

The SMI service routine is user definable and may contain system or power management software. If the power management software forces the CPU to power down, or the SMI service routine modifies more than what is automatically saved, the complete CPU state information can be saved.

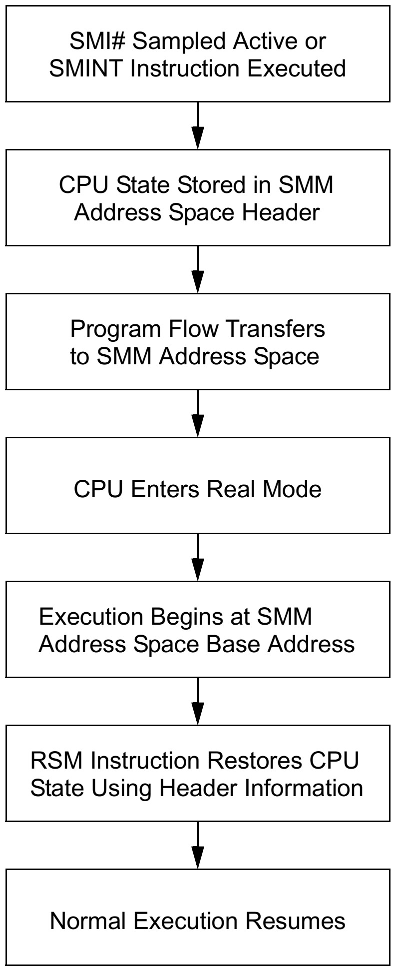


Figure 2-13 SMI Execution Flow Diagram

2.2.6.2. SMM Memory Space Header

With every SMI interrupt or SMINT instruction, certain CPU state information is automatically saved in the SMM memory space header located at the top of SMM address space. See Figure 2-14. The header contains CPU state information that is modified when servicing an SMI interrupt. Included in this information are two pointers. The Current IP points to the instruction executing when the SMI was detected.

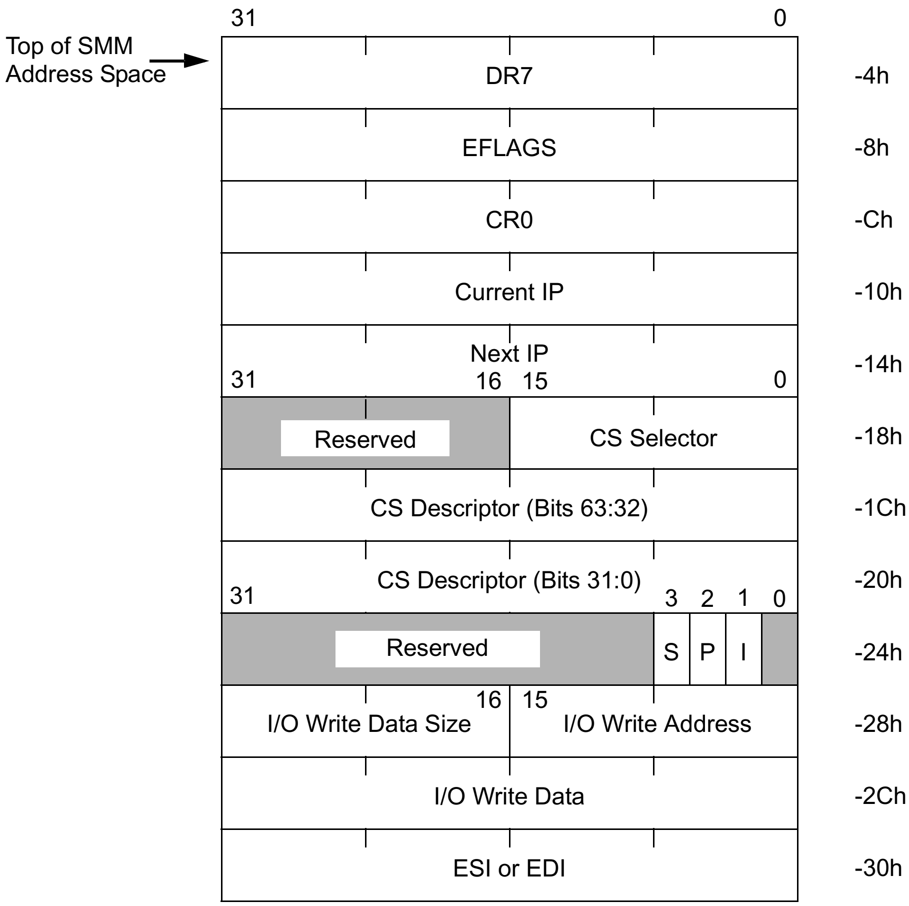


Figure 2-14 SMM Memory Space Header

The Next IP points to the instruction that will be executed after exiting SMM. Also saved are the contents of debug register 7 (DR7), the extended flags register (EFLAGS), and control register 0 (CR0). If SMM has been entered due to an I/O trap for a REP INSx or REP OUTSx instruction, the Current IP and Next IP fields contain the same addresses and the I and P field contain valid information.

If entry into SMM was caused by an I/O trap (“I/O Trapping” on page 99.), it is useful for the programmer to know the port address, data size and data value associated with that I/O operation. This information is also saved in the header and is only valid for an I/O write operation. The I/O write information is not restored within the CPU when executing an RSM instruction.

Table 2.34 SMM Memory Space Header

Name	Size	Description
DR7	4 Bytes	The contents of Debug Register 7.
EFLAGS	4 Bytes	The contents of Extended Flags Register.
CR0	4 Bytes	The contents of Control Register 0.
Current IP	4 Bytes	The address of the instruction executed prior to servicing SMI interrupt.
Next IP	4 Bytes	The address of the next instruction that will be executed after exiting SMM mode.
CS Selector	2 Bytes	Code segment register selector for the current code segment.
CS Descriptor	8 Bytes	Code segment register descriptor for the current code segment.
S	1 Bit	Software SMM Entry Indicator. S = 1 if current SMM is the result of an SMINT instruction. S = 0 if current SMM is not the result of an SMINT instruction.
P	1 Bit	REP INSx/OUTSx Indicator. P = 1 if current instruction has a REP prefix. P = 0 if current instruction does not have a REP prefix.
I	1 Bit	IN, INSx, OUT, or OUTSx Indicator. I = 1 if current instruction performed is an I/O WRITE. I = 0 if current instruction performed is an I/O READ.
I/O Write Data Size	2 Bytes	Indicates size of data for the trapped I/O write. 01h = Byte 03h = WORD 0Fh = DWORD
I/O Write Address	2 Bytes	Address of the trapped I/O write.
I/O Write Data	4 Bytes	Data associated with the trapped I/O write.
ESI or EDI	4 Bytes	Restored ESI or EDI value. Used when it is necessary to repeat a REP OUTSx or REP INSx instruction when one of the I/O cycles caused an SMI# trap.
Note: INSx = INS, INSB, INSW or INSD instruction. OUTSx = OUTS, OUTSB, OUTSW and OUTSD instruction.		

2.2.6.3. SMM Instructions

The processor automatically saves the minimal amount of CPU state information when entering SMM which allows fast SMI service routine entry and exit. After entering the SMI service routine, the MOV, SVDC, SVLDT and SVTS instructions can be used to save the complete CPU state information. If the SMI service routine modifies more than what is automatically saved or forces the CPU to power down, the complete CPU state information must be saved. Since the CPU is a static device, its internal state is retained when the input clock is stopped. Therefore, an entire CPU state save is not necessary prior to stopping the input clock.

The SMM instructions, listed in Table 2-36, can only be executed if the following conditions are met:

SMI# is enabled **and**
SMAR SIZE > 0 **and**
[the Current Privilege Level (CPL) = 0 **and**
the SMAC bit (CCR1, bit 2) is set] **or**
[the Current Privilege Level (CPL) = 0 **and**
the CPU is in an SMI service routine (SMI# = 0)].

If the above conditions are not met and an attempt is made to execute an SVDC, RSDC, SVLDT, RSLDT, SVTS, RSTS, SMINT or RSM instruction, an invalid opcode exception is generated. These instructions can be executed outside of defined SMM space provided the above conditions are met.

The SMINT instruction can be used by software to enter SMM. The CPU will not drive the SMI# output low during the software initiated SMM.

Table 2.35 SMM Instruction Set

Instruction	OPCODE	Format	Description
SVDC	0F 78 [mod sreg3 r/m]	SVDC mem80, sreg3	Save Segment Register and Descriptor: Saves reg (DS, ES, FS, GS, or SS) to mem80.
RSDC	0F 79 [mod sreg3 r/m]	RSDC sreg3, mem80	Restore Segment Register and Descriptor: Restores reg (DS, ES, FS, GS, or SS) from mem80 Use RSM to restore CS. **Note:** Processing "RSDC CS, Mem80" will produce an exception.
SVLDT	0F 7A [mod 000 r/m]	SVLDT mem80	Save LDTR and Descriptor: Saves Local Descriptor Table (LDTR) to mem80.
RSLDT	0F 7B [mod 000 r/m]	RSLDT mem80	Restore LDTR and Descriptor: Restores Local Descriptor Table (LDTR) from mem80
SVTS	0F 7C [mod 000 r/m]	SVTS mem80	Save TSR and Descriptor: Saves Task State Register (TSR) to mem80.
RSTS	0F 7D [mod 000 r/m]	RSTS mem80	Restore TSR and Descriptor: Restores Task State Register (TSR) from mem80.
SMINT	0F 7E	SMINT	Software SMM Entry: CPU enters SMM mode. CPU state information is saved in SMM memory space header and execution begins at SMM base address.
RSM	0F AA	RSM	Resume Normal Mode: Exits SMM mode. The CPU state is restored using the SMM memory space header and execution resumes at interrupted point.
Note: mem80 = 80-bit memory location			

If the SMI# is asserted to the CPU during a software SMM, the SMI# handshake occurs normally. The hardware SMI# is serviced after the software SMM has been exited by execution of the RSM instruction.

All of the SMM instructions (except RSM and SMINT) save or restore 80 bits of data, allowing the saved values to include the hidden portion of the register contents.

2.2.6.4. SMM Memory Space

SMM memory space is defined by specifying the base address and size of the SMM memory space in the SMAR register. The base address must be a multiple of the SMM memory space size. For example, a 32 KB SMM memory space must be located at a 32 KB address boundary. The memory space size can range from 4 KB to 32 MB.

SMM memory space accesses are always non-cacheable. SMM accesses ignore the state of the A20M# input signal and drive the A20 address bit to the unmasked value.

Access to the SMM memory space can be made even though not in SMM mode by setting the SMAC bit in the CCR1 register. This feature may be used to initialize the SMM memory space.

While in SMM mode, SMADS# address strobes are generated instead of ADS# for SMM memory accesses. Any memory accesses outside the defined SMM space result in normal memory accesses and ADS# strobes. Data (non-code) accesses to main memory that overlap with defined SMM memory space are allowed if MMAC in CCR1 is set. In this case, ADS# strobes are generated for data accesses only and SMADS# strobes continue to be generated for code accesses.

2.2.6.5. SMI Service Routine Execution

After the SMM header has been saved, upon entry into SMM the CR0, EFLAGS, and DR7 registers are set to their reset values. The Code Segment (CS) register is loaded with the base, as defined by the SMAR register, and a limit of 4 GB. The SMI service routine then begins execution at the SMM base address in real mode.

The programmer must save the value of any registers that may be changed by the SMI service routine. For data accesses immediately after entering the SMI service routine, the programmer must use CS as a segment override. I/O port access is possible during the routine, but care must be taken to save registers modified by the I/O instructions. Before using a segment register, the register and the register's descriptor cache contents should be saved using the SVDC instruction. While executing in the SMM space, execution flow can transfer to normal memory locations.

Hardware interrupts, (INTRs and NMIs), may be serviced during a SMI service routine. If interrupts are to be serviced while executing in the SMM memory space, the SMM memory space must be within the 0 to 1 MB address range to guarantee proper return to the SMI service routine after handling the interrupt.

INTRs are automatically disabled when entering SMM since the IF flag is set to its reset value. Once in SMM, the INTR can be enabled by setting the IF flag. An NMI event in SMM mode can be enabled by setting NMIEN in the CCR3 register. If NMI is not enabled while in SMM mode, the CPU latches one NMI event and services the interrupt after NMI has been enabled or after exiting SMM mode through the RSM instruction.

Within the SMI service routine, protected mode may be entered and exited as required, and real or protected mode device drivers may be called.

To exit the SMI service routine, a Resume (RSM) instruction, rather than an IRET, is executed. The RSM instruction causes the processor to restore the CPU state using the SMM header information and resume execution at the interrupted point. If the full CPU state was saved by the programmer, the stored values should be reloaded prior to executing the RSM instruction using the MOV, RSDC, RSLDT and RSTS instructions.

CPU States Related to SMM and Suspend Mode

Figure 2-15 illustrates the various CPU states associated with SMM and suspend mode. While in the SMI service routine, the processor can enter suspend mode either by (1) executing a halt (HLT) instruction or (2) by asserting the SUSP# input.

During SMM operations and while in SUSP# initiated suspend mode, an occurrence of either NMI or INTR is latched. (In order for INTR to be latched, the IF flag must be set.)

The INTR or NMI is serviced after exiting suspend mode. If suspend mode is entered via a HLT instruction from the operating system or application software, the reception of an SMI# interrupt causes the CPU to exit suspend mode and enter SMM. If suspend mode is entered via the hardware (SUSP# = 0) while the operating system or application software is active, the CPU latches one occurrence of INTR, NMI and SMI#.

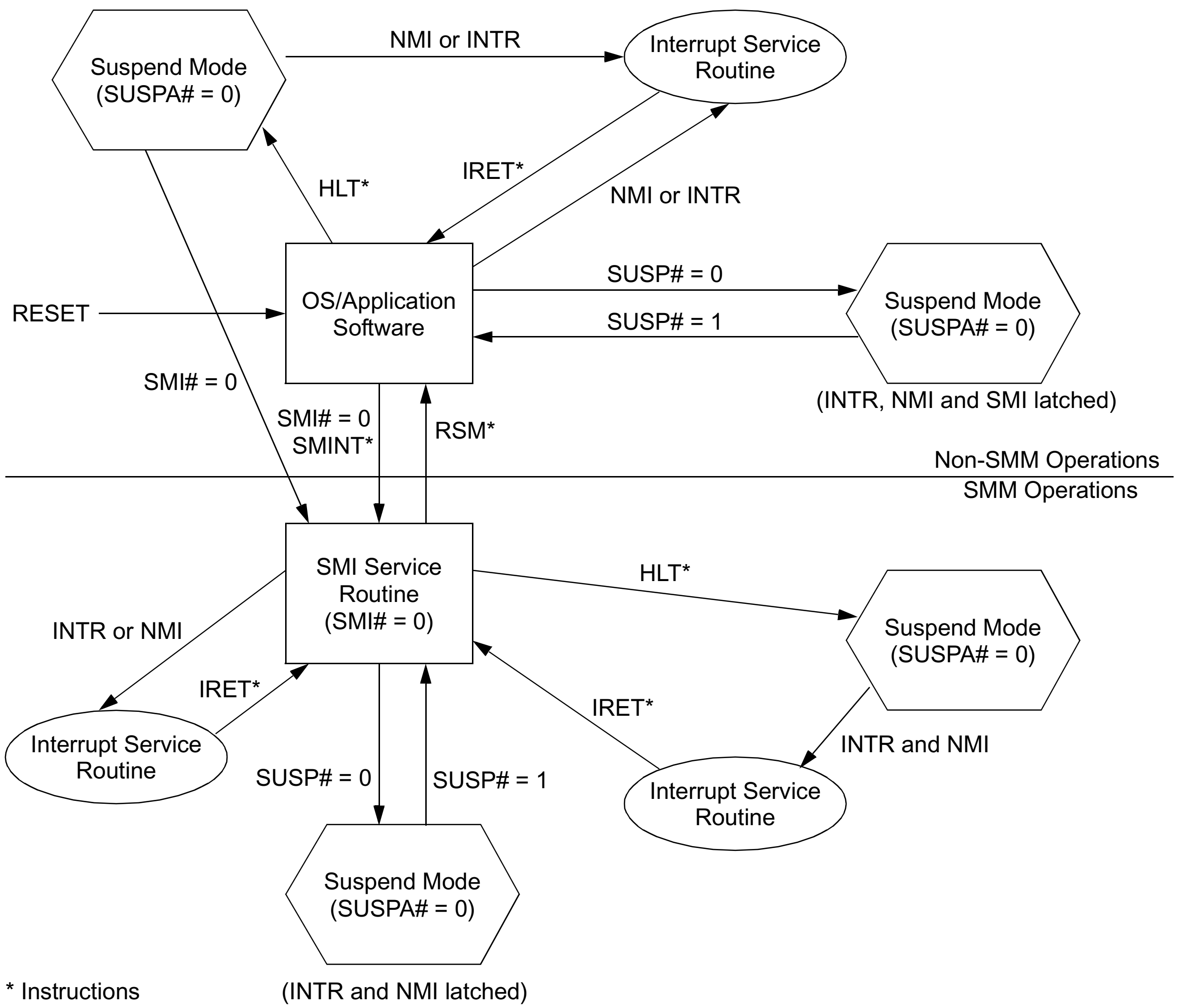


Figure 2-15 SMM and Suspend Mode State Diagram

SL-enhanced Compatibility Mode

Following power-up or RESET, the CPU SMM interface pins are disabled. Once enabled, these two pins can either function as defined previously (SMI and SMADS) or can be programmed to function with a signalling protocol compatible with the 32-bit x86-enhanced CPUs (SMI, SMIACT). This section describes the operation of the SMM interface pins when operating in the SL-compatible mode.

SMM Control Bit

The SMM_Mode bit in the configuration register (CCR3 bit 3) controls the SMM interface mode. Once the SMI_Lock bit is set, the CPU must be reset in order to modify SMI_Lock and SMM_Mode.

Pin Definitions

The two pins that change function in SL-compatible mode are SMI and SMADS.

Table 2.36 SMM Pin Definitions

Non-SLCompatible Mode	SL-Compatible Mode
SMI: Bidirectional System Management Interrupt pin. Asserted by the system logic to request an SMI interrupt. Sampled by the CPU on each rising clock edge. Causes I/O trap to occur if sampled and found asserted at least two clocks prior to ready sampled asserted for an I/O cycle. Asserted by the CPU during execution of an SMI service routine or in response to SMINT if SMAC is set.	SMI: System Management Interrupt input pin. Asserted by the system logic to request an SMI interrupt. Sampled by the CPU on each rising clock edge. SMI is falling edge sensitive and causes an I/O trap to occur if sampled and found asserted at least three clocks prior to RDY/BRDY sampled asserted for any I/O cycle.
SMADS: SMI Address Strobe output used to indicate that the current bus cycle is an SMM memory access.	SMIACT: SMI Active output asserted by the CPU during execution of an SMI service routine.

Nested SMI

In Non-SLCompatible mode, nested SMI's cannot occur due to the fact that the SMI pin becomes an output during SMI servicing. In SL-Compatible mode, if an SMI occurs during an SMI service routine, one and only one SMI is latched. The latched SMI is then serviced immediately following execution of a RSM instruction (used to exit the original SMI service routine).

SMM Features Not Used with SL-Compatible Interface.

The SMAC and MMAC functions are disabled when in SL-Compatible mode. Additionally, SMIACT remains asserted while executing an SMI service routine regardless of the address being accessed. In other words, if the SMI service routine accesses memory outside the defined SMM memory space, SMIACT remains asserted. Also, the SMINT instruction should not be used in SL-Compatible mode.

Write-Back Caching and SMM

The CPU allows caching of SMM memory accesses. The SMM memory caching may cause coherency problems in systems where SMM memory space and normal memory space overlap. Therefore, one of the following options is recommended:

1) Flush the cache when entering and exiting an SMI service routine.

OR

2) Flush the cache when entering an SMI service routine and then make all SMM accesses non-cacheable using the KEN pin.

In either case it is recommended to assert the FLUSH input pin when the SMIACT pin is asserted. Asserting FLUSH in this manner is acceptable for a CPU with write-through cache as the flush invalidates the cache in a single clock.

However, on CPUs with write-back cache, asserting FLUSH requires the writing of all dirty data to external memory prior to invalidating the cache contents. Bus cycles that address normal memory addresses that overlap with SMM memory space should not be issued while SMIACT is asserted.

Therefore, while in SL-Compatible mode, the CPU automatically writes all dirty data to memory and then invalidates the cache prior to asserting SMIACT. This guarantees that no dirty data exists in the CPU at the time that SMIACT is asserted.

SMM accesses are always non-cacheable and the cache is flushed before entering the SMI service routine. For these reasons, a bus snoop that occurs while SMIACT is asserted can never hit on a dirty line that is in SMM space or the overlapped normal memory space.Therefore, bus snoops that occur, while SMIACT is asserted, never result in memory incoherences.

2.2.7. Shutdown and Halt

The halt instruction (HLT) stops program execution and prevents the processor from using the local bus until restarted. The processor then enters a low-power suspend mode if the HLT bit in CCR2 is set. SMI, NMI, INTR with interrupts enabled (IF bit in EFLAGS = 1), or RESET forces the CPU out of the halt state. If interrupted, the saved code segment and instruction pointer specify the instruction following the HLT.

Shutdown occurs when a severe error is detected that prevents further processing. An NMI input can bring the processor out of shutdown if the IDT limit is large enough to contain the NMI interrupt vector (at least 000Fh) and the stack has enough room to contain the vector and flag information (i.e., stack pointer is greater than 0005h). Otherwise, shutdown can only be exited by a processor reset.

2.2.8. Protection

Segment protection and page protection are safeguards built into the CPU protected mode architecture which deny unauthorized or incorrect access to selected memory addresses. These safeguards allow multitasking programs to be isolated from each other and from the operating system. Page protection is discussed earlier in this chapter in 'Paging Mechanism' on page 57. This section concentrates on segment protection.

Selectors and descriptors are the key elements in the segment protection mechanism. The segment base address, size, and privilege level are established by a segment descriptor. Privilege levels control the use of privileged instructions, I/O instructions and access to segments and segment descriptors. Selectors are used to locate segment descriptors.

Segment accesses are divided into two basic types, those involving code segments (e.g., control transfers) and those involving data accesses. The ability of a task to access a segment depends on:

- the segment type
- the instruction requesting access
- the type of descriptor used to define the segment
- the associated privilege levels (described below).

Data stored in a segment can be accessed only by code executing at the same or a more privileged level. A code segment or procedure can only be called by a task executing at the same or a less privileged level.

2.2.8.1. Privilege Levels

The values for privilege levels range between 0 and 3. Level 0 is the highest privilege level (most privileged), and level 3 is the lowest privilege level (least privileged). The privilege level in real mode is effectively 0.

The **Descriptor Privilege Level (DPL)** is the privilege level defined for a segment in the segment descriptor. The DPL field specifies the minimum privilege level needed to access the memory segment pointed to by the descriptor.

The **Current Privilege Level (CPL)** is defined as the current task's privilege level. The CPL of an executing task is stored in the hidden portion of the code segment register and essentially is the DPL for the current code segment.

The **Requested Privilege Level (RPL)** specifies a selector's privilege level and is used to distinguish between the privilege level of a routine actually accessing memory (the CPL), and the privilege level of the original requestor (the RPL) of the memory access. The lesser of the RPL and CPL is called the effective privilege level (EPL). Therefore, if RPL = 0 in a segment selector, the effective privilege level is always determined by the CPL. If RPL = 3, the effective privilege level is always 3 regardless of the CPL.

For a memory access to succeed, the effective privilege level (EPL) must be at least as privileged as the descriptor privilege level (EPL £ DPL). If the EPL is less privileged than the DPL (EPL > DPL), a general protection fault is generated. For example, if a segment has a DPL = 2, an instruction accessing the segment only succeeds if executed with an EPL £ 2.

2.2.8.2. I/O Privilege Levels

The I/O Privilege Level (IOPL) allows the operating system executing at CPL = 0 to define the least privileged level at which IOPL-sensitive instructions can unconditionally be used. The IOPL-sensitive instructions include CLI, IN, OUT, INS, OUTS, REP INS, REP OUTS, and STI. Modification of the IF bit in the EFLAGS register is also sensitive to the I/O privilege level.

The IOPL is stored in the EFLAGS register. An I/O permission bit map is available as defined by the 32-bit Task State Segment (TSS). Since each task can have its own TSS, access to individual I/O ports can be granted through separate I/O permission bit maps.

If CPL <= IOPL, IOPL-sensitive operations can be performed. If CPL > IOPL, a general protection fault is generated if the current task is associated with a 16-bit TSS. If the current task is associated with a 32-bit TSS and CPL > IOPL, the CPU consults the I/O permission bitmap in the TSS to determine on a port-by-port basis whether or not I/O instructions (IN, OUT, INS, OUTS, REP INS, REP OUTS) are permitted, and the remaining IOPL-sensitive operations generate a general protection fault.

2.2.8.3. Privilege Level Transfers

A task's CPL can be changed only through intersegment control transfers using gates or task switches to a code segment with a different privilege level. Control transfers result from exception and interrupt servicing and from execution of the CALL, JMP, INT, IRET and RET instructions.

There are five types of control transfers that are summarized in Table 2.37. Control transfers can be made only when the operation causing the control transfer references the correct descriptor type. Any violation of these descriptor usage rules causes a general protection fault.

Any control transfer that changes the CPL within a task results in a change of stack. The initial values for the stack segment (SS) and stack pointer (ESP) for privilege levels 0, 1, and 2 are stored in the TSS. During a JMP or

CALL control transfer, the SS and ESP are loaded with the new stack pointer and the previous stack pointer is saved on the new stack. When returning to the original privilege level, the RET or IRET instruction restores the less-privileged stack.

Table 2.37 Descriptor Types Used for Control Transfer

Control Transfer Type	Operation Types	Descriptor	Descriptor Table
Intersegment within the same privilege level.	JMP, CALL, RET, IRET*	Code Segment	GDT or LDT
Intersegment to the same or a more privileged level. Interrupt within task (could change CPL level).	CALL	Gate Call	GDT or LDT
	Interrupt Instruction, Exception External Interrupt	Trap or Interrupt Gate	LDT
Intersegment to a less privileged level (changes task CPL).	RET, IRET*	Code Segment	GDT or LDT
Task Switch via TSS.	CALL, JMP	Task State Segment	GDT
Task Switch via Task Gate.	CALL, JMP	Task Gate	GDT or LDT
	IRET** Interrupt Instruction, Exception, External Interrupt	Task Gate	IDT

* NT (Nested Task bit in EFLAGS) = 0
** NT (Nested Task bit in EFLAGS) = 1

Gates

Gate descriptors provide protection for privilege transfers among executable segments. Gates are used to transition to routines of the same or a more privileged level. Call gates, interrupt gates and trap gates are used for privilege transfers within a task. Task gates are used to transfer between tasks.

Gates conform to the standard rules of privilege. In other words, gates can be accessed by a task if the effective privilege level (EPL) is the same or more privileged than the gate descriptor's privilege level (DPL).

2.2.8.4. Initialization and Transition to Protected Mode

The processor switches to real mode immediately after RESET. While operating in real mode, the system tables and registers should be initialized. The GDTR and IDTR must point to a valid GDT and IDT, respectively. The size of the IDT should be at least 256 bytes, and the GDT must contain descriptors which describe the initial code and data segments.

The processor can be placed in protected mode by setting the PE bit in the CR0 register. After enabling protected mode, the CS register should be loaded and the instruction decode queue should be flushed by executing an intersegment JMP. Finally, all data segment registers should be initialized with appropriate selector values.

2.2.9. Virtual 8086 Mode

Both real mode and virtual 8086 (V86) mode are supported by the CPU, allowing execution of 8086 application programs and 8086 operating systems. V86 mode allows the execution of 8086-type applications, yet still permits use of the processor protection mechanism. V86 tasks run at privilege level 3. Upon entry, all segment limits are set to FFFFh (64 KB) as in real mode.

2.2.9.1. Memory Addressing

While in V86 mode, segment registers are used in an identical fashion to real mode. The contents of the segment register are multiplied by 16 and added to the offset to form the segment base linear address. The CPU permits the operating system to select which programs use the V86 address mechanism and which programs use protected mode addressing for each task.

The processor also permits the use of paging when operating in V86 mode. Using paging, the 1 MB address space of the V86 task can be mapped to anywhere in the 4 GB linear address space of the CPU.

The paging hardware allows multiple V86 tasks to run concurrently and provides protection and operating system isolation. The paging hardware must be enabled to run multiple V86 tasks or to relocate the address space of a V86 task to physical address space greater than 1 MB.

2.2.9.2. Protection

All V86 tasks operate with the least amount of privilege (level 3) and are subject to all of the protected mode protection checks. As a result, any attempt to execute a privileged instruction within a V86 task results in a general protection fault.

In V86 mode a slightly different set of instructions is sensitive to the I/O privilege level (IOPL) than in protected mode. The instructions are: CLI, INTn, IRET, POPF, PUSHF, and STI. The INT3, INTO and BOUND variations of the INT instruction are not IOPL sensitive.

2.2.9.3. Interrupt Handling

To fully support the emulation of an 8086-type machine, interrupts in V86 mode are handled as follows. When an interrupt or exception is serviced in V86 mode, program execution transfers to the interrupt service routine at privilege level 0 (i.e., transition from V86 to protected mode occurs) and the VM bit in the EFLAGS register is cleared. The protected mode interrupt service routine then determines if the interrupt came from a protected mode or V86 application by examining the VM bit in the EFLAGS image stored on the stack. The interrupt service routine may then choose to allow the 8086 operating system to handle the interrupt or may emulate the function of the interrupt handler. Following completion of the interrupt service routine, an IRET instruction restores the EFLAGS register (restores VM = 1) and segment selectors and control returns to the interrupted V86 task.

2.2.9.4. Entering and Leaving V86 Mode

V86 mode is entered from protected mode by either executing an IRET instruction at CPL = 0 or by task switching. If an IRET is used, the stack must contain an EFLAGS image with VM = 1. If a task switch is used, the TSS must contain an EFLAGS image containing a 1 in the VM bit position. The POPF instruction cannot be used to enter V86 mode since the state of the VM bit is not affected. V86 mode can only be exited as the result of an interrupt or exception. The transition out must use a 32-bit trap or interrupt gate which must point to a non-conforming privilege level 0 segment (DPL = 0), or a 32-bit TSS. These restrictions are required to permit the trap handler to IRET back to the V86 program.

2.2.10. FPU Operations

2.2.10.1. FPU Register Set

In addition to the registers described to this point, the FPU circuitry within the CPU provides the user eight data registers (accessed in a stack-like manner), a control register, and a status register. The CPU also provides a data register tag word which improves context switching and stack performance by maintaining empty/non-empty status for each of the eight data registers. In addition, registers in the CPU contain pointers to (a) the memory location containing the current instruction word and (b) the memory location containing the operand associated with the current instruction word (if any).

FPU Tag Word Register. The processor maintains a tag word register comprised of two bits for each physical data register. Tag Word fields assume one of four values depending on the contents of their associated data registers, Valid (00), Zero (01), Special (10), and Empty (11). Note: Denormal, Infinity, QNaN, SNaN and unsupported formats are tagged as "Special". Tag values are maintained transparently by the processor and are only available to the programmer indirectly through the FSTENV and FSAVE instructions. The tag word with tag fields for each associated physical register, tag(n), is shown in Figure 2-16.

15	13	11	9	7	5	3	1
Tag 7	Tag 6	Tag 5	Tag 4	Tag 3	Tag 2	Tag 1	Tag 0

Figure 2-16 Tag Word Register

FPU Status Register. The FPU circuitry communicates information about its status and the results of operations to the CPU via the status register. The FPU status register, illustrated in Figure 2-17, is comprised of bit fields that reflect exception status, operation execution status, register status, operand class, and comparison results. This register is continuously accessible to the CPU regardless of the state of the Control or Execution Units. The Status Register's bit definitions are given in Table 2.38.

15				11				7				3			
B	C3	S	S	S	C2	C1	C0	ES	SF	P	U	O	Z	D	I

Figure 2-17 FPU Status Register

FPU Mode Control Register. The FPU Mode Control Register (MCR), Figure 2-18, is used by the CPU to specify the operating mode of the FPU. The MCR contains bit fields which specify the rounding mode to be used, the precision by which to calculate results, and the exception conditions which should be reported to the CPU via traps. The user controls precision, rounding, and exception reporting by setting or clearing appropriate bits in the MCR. The Mode Control Register's bit definitions are given in Table 2.39.

15				11				7				3			
-	-	-	-	RC	RC	PC	PC	-	-	P	U	O	Z	D	I

Figure 2-18 FPU Mode Control Register

Table 2.38 Status Control Register Bit Definitions

Bit	Name	Description
15	B	Copy of the ES bit. (ES is bit 7 in this table.)
14, 10:8	C3-C0	Condition code bits.
13:11	SSS	Top of stack register number which points to the current TOS.
7	ES	Error indicator. Set to 1 if an unmasked exception is detected.
6	SF	Stack Fault or invalid register operation bit.
5	P	Precision error exception bit.
4	U	Underflow error exception bit.
3	O	Overflow error exception bit.
2	Z	Divide by zero exception bit.

Table 2.38 Status Control Register Bit Definitions

Bit	Name	Description
1	D	Denormalized operand error exception bit.
0	I	Invalid operation exception bit.

Table 2.39 Mode Control Register Bit Definition

Bit	Name	Description
15:12	RSVD	Reserved.
11:10	RC	Rounding Control bits: 00 = Round to nearest or even 01 = Round towards minus infinity 10 = Round towards plus infinity 11 = Truncate
9:8	PC	Precision Control bits: 00 = 24-bit mantissa 01 = Reserved 10 = 53-bit mantissa 11 = 64-bit mantissa
5	P	Precision error exception bit mask.
4	U	Underflow error exception bit mask.
3	O	Overflow error exception bit mask.
2	Z	Divide by zero exception bit mask.
1	D	Denormalized operand error exception bit mask.
0	I	Invalid operation exception bit mask.

2.3. Instruction Set

This section summarizes the Processor instruction set and provides detailed information on the instruction encodings. All instructions are listed in the CPU Instruction Set Summary Table Table 2.56 on page 87, and the FPU Instruction Set Summary Table Table 2.58 on page 101. These tables provide information on the instruction encoding, and the instruction clock counts for each instruction. The clock count values for both tables are based on the assumptions described in the Section 2.3.2.1. 'Assumptions Made in Determining Instruction Clock Count' on page 86.

Depending on the instruction, the CPU instructions follow the general instruction format shown in Figure 2-19. These instructions vary in length and can start at any byte address. An instruction consists of one or more bytes that can include: prefix byte(s), at least one opcode byte(s), mod r/m byte, s-i-b byte, address displacement byte(s) and immediate data byte(s). An instruction can be as short as one byte and as long as 15 bytes. If there are more than 15 bytes in the instruction a general protection fault (error code 0) is generated.

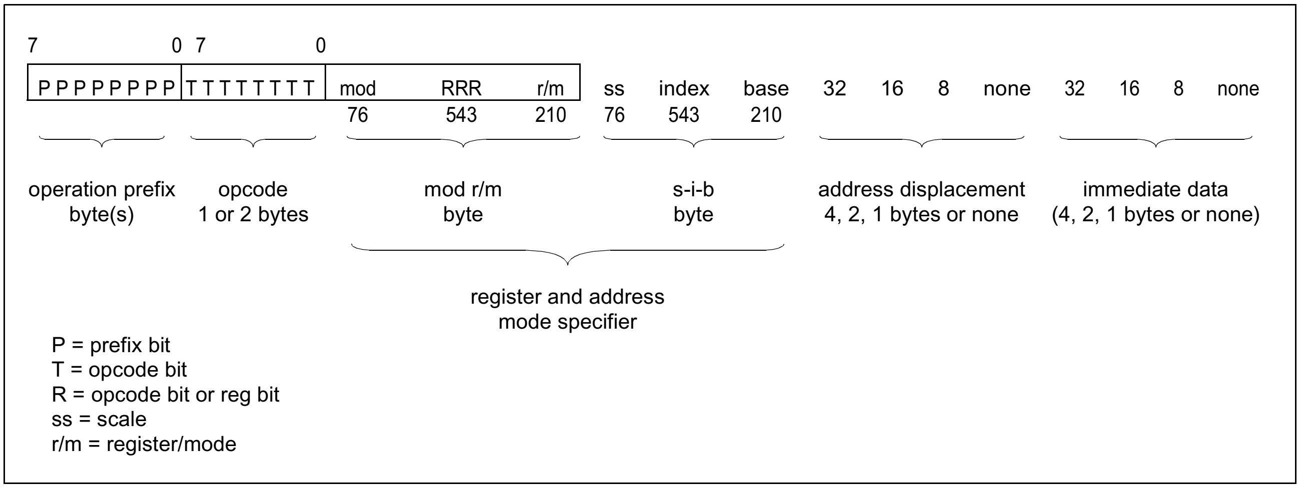


Figure 2-19 Instruction Set Format

2.3.1. General Instruction Fields

The fields in the general instruction format at the byte level are listed in Table 2.40.

Table 2.40 Instruction Fields

Field Name	Description	Width
Optional Prefix Byte(s)	Specifies segment register override, address and operand size, repeat elements in string instruction, LOCK# assertion.	1 or more bytes
Opcode Byte(s)	Identifies instruction operation.	1 or 2 bytes
mod and r/m Byte	Address mode specifier.	1 byte
s-i-b Byte	Scale factor, Index and Base fields.	1 byte
Address Displace-ment	Address displacement operand.	1, 2 or 4 bytes
Immediate data	Immediate data operand.	1, 2 or 4 bytes

2.3.1.1. Optional Prefix Byte(s)

Prefix bytes can be placed in front of any instruction. The prefix modifies the operation of the next instruction only. When more than one prefix is used, the order is not important. There are five types of prefixes as follows:

- Segment Override explicitly specifies which segment register an instruction will use for effective address calculation.
- Address Size switches between 16- and 32-bit addressing. Selects the inverse of the default.
- Operand Size switches between 16- and 32-bit operand size. Selects the inverse of the default.
- Repeat is used with a string instruction which causes the instruction to be repeated for each element of the string.
- Lock is used to assert the hardware LOCK# signal during execution of the instruction.

Table 2.41 lists the encodings for each of the available prefix bytes. The operand size and address size prefixes allow the individual overriding of the default value for operand size and effective address size. The presence of these prefixes selects the opposite (non-default) operand size and/or effective address size.

Table 2.41 Instruction Prefix Summary

Prefix	Encoding	Description
ES:	26h	Override segment default, use ES for memory operand
CS:	2Eh	Override segment default, use CS for memory operand
SS:	36h	Override segment default, use SS for memory operand
DS:	3Eh	Override segment default, use DS for memory operand
FS:	64h	Override segment default, use FS for memory operand
GS:	65h	Override segment default, use GS for memory operand

Table 2.41 Instruction Prefix Summary

Prefix	Encoding	Description
Operand Size	66h	Make operand size attribute the inverse of the default
Address Size	67h	Make address size attribute the inverse of the default
LOCK	F0h	Assert LOCK# hardware signal.
REPNE	F2h	Repeat the following string instruction.
REP/REPE	F3h	Repeat the following string instruction.

2.3.1.2. Opcode Byte(s)

The opcode field is either one or two bytes in length and may be further defined by additional bits in the mod r/m byte. The opcode field specifies the operation to be performed by the instruction. Some operations have more than one opcode, each specifying a different form of the operation. Some opcodes name instruction groups. For example, opcode 0x80 names a group of operations that has an immediate operand, and a register or memory operand. The opcodes are given in hex values unless shown within brackets ([ ]). Values within brackets are given in binary. The reg field may appear in the second opcode byte or in the mod r/m byte.

2.3.1.3. w Field

The 1-bit w field selects the operand size during 16 and 32 bit data operations.

Table 2.42 w Field Encoding

W Field	Operand Size 16-bit Data Operations	Operand Size 32-bit Data Operations
0	8 Bits	8 Bits
1	16 Bits	32 Bits

2.3.1.4. d Field

The d field determines which operand is taken as the source operand and which operand is taken as the destination.

Table 2.43 d Field Encoding

d Field	Direction of Operation	Source Operand	Destination Operand
0	Register --> Register or Register --> Memory	reg	mod r/m or mod s-i-b
1	Register --> Register or Memory --> Register	mod r/m or mod s-i-b	reg

2.3.1.5. eee Field

The eee field is used to select the control, debug and test registers in the MOV instructions. The type of register and base registers selected by the eee field are listed in Table 2.44. The values shown are the only valid encodings for the eee bits.

Table 2.44 eee Field Encoding

eee Field	Register Type	Base Register
000	Control Register	CR0
010	Control Register	CR2
011	Control Register	CR3
000	Debug Register	DR0
001	Debug Register	DR1
010	Debug Register	DR2
011	Debug Register	DR3
110	Debug Register	DR6
111	Debug Register	DR7

Table 2.44 eee Field Encoding (cont.)

eee Field	Register Type	Base Register
011	Test Register	TR3
100	Test Register	TR4
101	Test Register	TR5
110	Test Register	TR6
111	Test Register	TR7

2.3.1.6. mod and r/m Byte

The mod and r/m fields, within the mod r/m byte, select the type of memory addressing to be used. Some instructions use a fixed addressing mode (e.g., PUSH or POP) and therefore, these fields are not present. Table 2.45 lists the addressing method when 16-bit addressing is used and a mod r/m byte is present. Some mod r/m field encodings are dependent on the w field and are shown in Table 2.46.

Table 2.45 mod r/m Field Encoding

mod and r/m fields	**16-bit Address Mode** with mod r/m byte	**32-bit Address Mode** with mod r/m byte and no s-i-b byte present
00 000	DS:[BX+SI]	DS:[EAX]
00 001	DS:[BX+DI]	DS:[ECX]
00 010	DS:[BP+SI]	DS:[EDX]
00 011	DS:[BP+DI]	DS:[EBX]
00 100	DS:[SI]	s-i-b is present
00 101	DS:[DI]	DS:[d32]
00 110	DS:[d16]	DS:[ESI]
00 111	DS:[BX]	DS:[EDI]
01 000	DS:[BX+SI+d8]	DS:[EAX+d8]
01 001	DS:[BX+DI+d8]	DS:[ECX+d8]
01 010	DS:[BP+SI+d8]	DS:[EDX+d8]
01 011	DS:[BP+DI+d8]	DS:[EBX+d8]
01 100	DS:[SI+d8]	s-i-b is present
01 101	DS:[DI+d8]	SS:[EBP+d8]
01 110	SS:[BP+d8]	DS:[ESI+d8]
01 111	DS:[BX+d8]	DS:[EDI+d8]
10 000	DS:[BX+SI+d16]	DS:[EAX+d32]
10 001	DS:[BX+DI+d16]	DS:[ECX+d32]
10 010	DS:[BP+SI+d16]	DS:[EDX+d32]
10 011	DS:[BP+DI+d16]	DS:[EBX+d32]
10 100	DS:[SI+d16]	s-i-b is present
10 101	DS:[DI+d16]	SS:[EBP+d32]
10 110	SS:[BP+d16]	DS:[ESI+d32]
10 111	DS:[BX+d16]	DS:[EDI+d32]
11000-11111	See Table 5-7	See Table 5-7

Table 2.46 mod r/m Field Encoding Dependent on w Field

mod r/m	16-bit Operation w = 0	16-bit Operation w = 1	32-bit Operation w = 0	32-bit Operation w = 1
11 000	AL	AX	AL	EAX
11 001	CL	CX	CL	ECX
11 010	DL	DX	DL	EDX
11 011	BL	BX	BL	EBX
11 100	AH	SP	AH	ESP
11 101	CH	BP	CH	EBP
11 110	DH	SI	DH	ESI
11 111	BH	DI	BH	EDI

2.3.1.7. reg Field

The reg field determines which general registers are to be used. The selected register is dependent on whether a 16 or 32 bit operation is current and the status of the w bit.

Table 2.47 reg Field

reg	16-bit Operation w Field Not Present	32-bit Operation w Field Not Present	16-bit Operation w = 0	16-bit Operation w = 1	32-bit Operation w = 0	32-bit Operation w = 1
000	AX	EAX	AL	AX	AL	EAX
001	CX	ECX	CL	CX	CL	ECX
010	DX	EDX	DL	DX	DL	EDX
011	BX	EBX	BL	BX	BL	EBX
100	SP	ESP	AH	SP	AH	ESP
101	BP	EBP	CH	BP	CH	EBP
110	SI	ESI	DH	SI	DH	ESI
111	DI	EDI	BH	DI	BH	EDI

2.3.1.8. sreg3 Field

The sreg3 field (Table 2.43) is 3-bit field that is similar to the sreg2 field, but allows use of the FS and GS segment registers.

Table 2.48 sreg3 Field Encoding

sreg3 Field	Segment Register Selected
000	ES
001	CS

Table 2.48 sreg3 Field Encoding

sreg3 Field	Segment Register Selected
010	SS
011	DS
100	FS
101	GS
110	undefined
111	undefined

2.3.1.9. sreg2 Field

The sreg2 field (Table 2.42) is a 2-bit field that allows one of the four 286 type segment registers to be specified.

Table 2.49 sreg2 Field Encoding

sreg2 FIELD	Segment Register Selected
00	ES
01	CS
10	SS
11	DS

2.3.1.10. s-i-b Byte

The s-i-b fields provide scale factor, indexing and a base field for address selection.

2.3.1.11. ss Field

The ss field (Table 2.50) specifies the scale factor used in the offset mechanism for address calculation. The scale factor multiplies the index value to provide one of the components used to calculate the offset address.

Table 2.50 ss Field Encoding

ss Field	Scale Factor
00	x1

Table 2.50 ss Field Encoding

ss Field	Scale Factor
01	x2
01	x4
11	x8

2.3.1.12. index Field

The index field (Table 2.51) specifies the index register used by the offset mechanism for offset address calculation. When no index register is used (index field = 100), the ss value must be 00 or the effective address is undefined.

Table 2.51 index Field Encoding

Index Field	Index Register
000	EAX
001	ECX
010	EDX
011	EBX
100	none
101	EBP
110	ESI
111	EDI

Base Field

In Table 2.45, the note “s-i-b present” for certain entries forces the use of the mod and base field as listed in Table 2.52. The first two digits in the first column of this table identifies the mod bits in the mod r/m byte. The last three digits in the first column identify the base fields in the s-i-b byte.

Table 2.52 mod base Field Encoding

mod Field within mode/rm Byte	base Field within s-i-b Byte	32-Bit Address Mode with mod r/m and s-i-b Bytes Present
00	000	DS:[EAX+(scaled index)]
00	001	DS:[ECX+(scaled index)]

Table 2.52 mod base Field Encoding (cont.)

mod Field within mode/rm Byte	base Field within s-i-b Byte	32-Bit Address Mode with mod r/m and s-i-b Bytes Present
00	010	DS:[EDX+(scaled index)]
00	011	DS:[EBX+(scaled index)]
00	100	SS:[ESP+(scaled index)]
00	101	DS:[d32+(scaled index)]
00	110	DS:[ESI+(scaled index)]
00	111	DS:[EDI+(scaled index)]
01	000	DS:[EAX+(scaled index)+d8]
01	001	DS:[ECX+(scaled index)+d8]
01	010	DS:[EDX+(scaled index)+d8]
01	011	DS:[EBX+(scaled index)+d8]
01	100	SS:[ESP+(scaled index)+d8]
01	101	SS:[EBP+(scaled index)+d8]
01	110	DS:[ESI+(scaled index)+d8]
01	111	DS:[EDI+(scaled index)+d8]
10	000	DS:[EAX+(scaled index)+d32]
10	001	DS:[ECX+(scaled index)+d32]
10	010	DS:[EDX+(scaled index)+d32]
10	011	DS:[EBX+(scaled index)+d32]
10	100	SS:[ESP+(scaled index)+d32]
10	101	SS:[EBP+(scaled index)+d32]
10	110	DS:[ESI+(scaled index)+d32]
10	111	DS:[EDI+(scaled index)+d32]

2.3.2. Instruction Set Tables

The instruction set is presented in two tables, the CPU Instruction Set (Table 2.56 on page 87) and the FPU Instruction Set (Table 2.58 on page 101). Additional information concerning the FPU Clock Counts is presented on page 100.

2.3.2.1. Assumptions Made in Determining Instruction Clock Count

The following assumptions have been made in presenting the clock count values for the individual instructions:

- All clock counts refer to the internal CPU internal clock frequency. For example, the clock counts for a clock-doubled processor refer to 50 MHz clocks while the external clock is 25 MHz.
- The instruction has been prefetched, decoded and is ready for execution.
- Bus cycles do not require wait states.
- There are no local bus HOLD requests delaying processor access to the bus.
- No exceptions are detected during instruction execution.
- If an effective address is calculated, it does not use two general register components. One register, scaling and displacement can be used within the clock count shown. However, if the effective address calculation uses two general register components, add one clock to the clock count shown.
- All clock counts assume aligned 32-bit memory/IO operands.
- If instructions access a 32-bit operand on odd addresses, add one clock for read or write and add two clocks for read and write.
- For non-cached memory accesses, add two clocks (DX) or four clocks (DX2), assuming zero wait state memory accesses.
- Locked cycles are not cacheable. Therefore, using the LOCK prefix with an instruction adds additional clocks as specified in instruction 9 above.

2.3.2.2. CPU Instruction Set Summary Table Abbreviations

The clock counts listed in the CPU Instruction Set Summary Table are grouped by operating mode and whether there is a register/cache hit or a cache miss. In some cases, more than one clock count is shown in a column for a given instruction, or a variable is used in the clock count. The abbreviations used for these conditions are listed in Table 2.53.

Table 2.53 CPU Clock Count Abbreviations

Clock Count Symbol	Explanation
/	Register operand/memory operand.
n	Number of times operation is repeated.
L	Level of the stack frame.
|	Conditional jump taken | Conditional jump not taken. (e.g. "4|1" = 4 clocks if jump taken, 1 clock if jump not taken)
\	CPL ≤IOPL \ CPL > IOPL (where CPL = Current Privilege Level, IOPL = I/O Privilege Level)

2.3.2.3. CPU Instruction Set Summary Table Flags Table

The CPU Instruction Set Summary Table lists nine flags that are affected by the execution of instructions. The conventions shown in Table 2.54 are used to identify the different flags. Table 2.55 lists the conventions used to indicate what action the instruction has on the particular flag.

Table 2.54 Flag Abbreviations

Abbreviation	Name Of Flag
OF	Overflow Flag
DF	Direction Flag
IF	Interrupt Enable Flag
TF	Trap Flag
SF	Sign Flag
ZF	Zero Flag
AF	Auxiliary Flag
PF	Parity Flag
CF	Carry Flag

Table 2.55 Action of Instruction on Flag

Instruction Table Symbol	Action
x	Flag is modified by the instruction.
-	Flag is not changed by the instruction.
0	Flag is reset to “0”.
1	Flag is set to “1”.

Table 2.56 Processor Core Instruction Set Summary

Instruction	Opcode	Flags									Real Mode	Prot'd Mode	Real Mode	Prot'd Mode
		OF	DF	IF	TF	SF	ZF	AF	PF	CF	Clock Count (Reg/Cache Hit)		Notes	
AAA *ASCII Adjust AL after Add*	37	-	-	-	-	-	-	x	-	x	4	4		
AAD *ASCII Adjust AX before Divide*	D5 0A	-	-	-	-	x	x	-	x	-	4	4		
AAM *ASCII Adjust AX after Multiply*	D4 0A	-	-	-	-	x	x	-	x	-	16	16		
AAS *ASCII Adjust AL after Subtract*	3F	-	-	-	-	-	-	x	-	x	4	4		
ADC *Add with Carry*														
Register to Register	1 [00dw] [11 reg r/m]	x	-	-	-	x	x	x	x	x	1	1	b	h
Register to Memory	1 [000w] [mod reg r/m]										3	3		
Memory to Register	1 [001w] [mod reg r/m]										3	3		
Immediate to Register/Memory	8 [00sw] [mod 010 r/m]#										1/3	1/3		
Immediate to Accumulator	1 [010w] #										1	1		
ADD *Integer Add*														
Register to Register	0 [00dw] [11 reg r/m]	x	-	-	-	x	x	x	x	x	1	1	b	h
Register to Memory	0 [000w] [mod reg r/m]										3	3		
Memory to Register	0 [001w] [mod reg r/m]										3	3		
Immediate to Register/Memory	8 [00sw] [mod 000 r/m]#										1/3	1/3		
Immediate to Accumulator	0 [010w] #										1	1		
AND *Boolean AND*														
Register to Register	2 [00dw] [11 reg r/m]	0	-	-	-	x	x	-	x	0	1	1	b	h
Register to Memory	2 [000w] [mod reg r/m]										3	3		
Memory to Register	2 [001w] [mod reg r/m]										3	3		
Immediate to Register/Memory	8 [00sw] [mod 100 r/m]#										1/3	1/3		
Immediate to Accumulator	2 [010w] #										1	1		
ARPL *Adjust Requested Privilege Level*														

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags O D I T S Z A P C F F F F F F F F F	Real Mode Clock Count (Reg/Cache Hit)	Prot'd Mode Clock Count (Reg/Cache Hit)	Real Mode Notes	Prot'd Mode Notes
From Register/Memory	63 [mod reg r/m]	- - - - - x - - -		6/10	a	h
BOUND *Check Array Boundaries*						
If Out of Range (Int 5)	62 [mod reg r/m]	- - - - - - - - -	11+INT	11+INT	b, e	g,h,j,k,r
If In Range			11	11		
BSF *Scan Bit Forward*						
Register, Register/Memory	0F BC [mod reg r/m]	- - - - - x - - -	5/7+n	5/7+n	b	h
BSR *Scan Bit Reverse*						
Register, Register/Memory	0F BC [mod reg r/m]	- - - - - x - - -	5/7+n	5/7+n	b	h
BSWAP *Byte Swap*	0F C[1 reg]	- - - - - - - - -	4	4		
BT *Test Bit*						
Register/Memory, Immediate	0F BA [mod 100 r/m]#	- - - - - - - - x	3/4	3/4	b	h
Register/Memory, Register	0F A3 [mod reg r/m]		3/6	3/6		
BTC *Test Bit and Complement*						
Register/Memory, Immediate	0F BA [mod 111 r/m]#	- - - - - - - - x	4/5	4/5	b	h
Register/Memory, Register	0F BB [mod reg r/m]		5/8	5/8		
BTR *Test Bit and Reset*						
Register/Memory, Immediate	0F BA [mod 110 r/m]#	- - - - - - - - x	4/5	4/5	b	h
Register/Memory, Register	0F B3 [mod reg r/m]		5/8	5/8		
BTS *Test Bit and Set*						
Register/Memory	0F BA [mod 101 r/m	- - - - - - - - x	3/5	3/5	b	h
Register (short form)	0F AB [mod reg r/m]		4/7	4/7		
CALL *Subroutine Call*						
Direct Within Segment	E8 +++	- - - - - - - - -	7	7	b	h,j,k,r
Register/Memory Indirect Within Segment	FF [mod 010 r/m]		8/9	8/9		
Direct Intersegment -Call Gate to Same Privilege -Call Gate to Different Privilege No Par's -Call Gate to Different Privilege m Par's -16-bit Task to 16-bit TSS -16-bit Task to 32-bit TSS -16-bit Task to V86 Task -32-bit Task to 16-bit TSS -32-bit Task to 32-bit TSS -32-bit Task to V86 Task	9A [unsigned full offset, selector]		12	30 41 83 81+4x 235 262 179 238 265 182		
Indirect Intersegment -Call Gate to Same Privilege -Call Gate to Different Privilege No Par's -Call Gate to Different Privilege m Par's -16-bit Task to 16-bit TSS -16-bit Task to 32-bit TSS -16-bit Task to V86 Task -32-bit Task to 16-bit TSS -32-bit Task to 32-bit TSS -32-bit Task to V86 Task	FF [mod 011 r/m]		14	14 43 85 86+4x 237 264 181 240 267 184		
CBW *Convert Byte to Word*	98	- - - - - - - - -	3	3		
CDQ *Convert DWORD to Quadword*	99	- - - - - - - - -	1	1		
CLC *Clear Carry Flag*	F8	- - - - - - - - 0	1	1		
CLD *Clear Direction Flag*	FC	- 0 - - - - - - -	1	1		
CLI *Clear Interrupt Flag*	FA	- - 0 - - - - - -	7	7		m

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags									Real Mode	Prot'd Mode	Real Mode	Prot'd Mode
		O F	D F	I F	T F	S F	Z F	A F	P F	C F	Clock Count (Reg/Cache Hit)		Notes	
CLTS *Clear Task Switched Flag*	0F 06	-	-	-	-	-	-	-	-	-	5	5	c	l
CMC *Complement the Carry Flag*	F5	-	-	-	-	-	-	-	-	x	1	1		
CMP *Compare Integers*														
Register to Register	3 [10dw] [11 reg r/m]	x	-	-	-	x	x	x	x	x			b	h
Register to Memory	3 [101w] [mod reg r/m]										1	1		
Memory to Register	3 [100w] [mod reg r/m]										3	3		
Immediate to Register/Memory	8 [00sw] [mod 111 r/m] #										3	3		
Immediate to Accumulator	3 [110w] ###										1/3	1/3		
CMPS *Compare String*	A [011w]	x	-	-	-	x	x	x	x	x	7	7	b	h
CMPXCHG *Compare and Exchange*														
Register1, Register2	0F B [000w] [11 reg2 reg1]	x	-	-	-	x	x	x	x	x	5	5		
Memory, Register	0F B [000w] [mod reg r/m]										7	7		
CWD *Convert Word to DWORD*	99	-	-	-	-	-	-	-	-	-	1	1		
CWDE *Convert Word to DWORD Extended*	98	-	-	-	-	-	-	-	-	-	3	3		
DAA *Decimal Adjust AL after Add*	27	-	-	-	-	x	x	x	x	x	4	4		
DAS *Decimal Adjust AL after Subtract*	2F	-	-	-	-	x	x	x	x	x	4	4		
DEC *Decrement by 1*														
Register/Memory	F [111w] [mod 001 r/m]	x	-	-	-	x	x	x	x	-	1/3	1/3	b	h
Register (short form)	4 [1 reg]										1	1		
DIV *Unsigned Divide*														
Accumulator by Register/Memory Divisor: Byte WORD DWORD	F [011w] [mod 110 r/m]	-	-	-	-	-	-	-	-	-	 14/15 22/23 38/39	 14/15 22/23 38/39	b,e	e,h
ENTER *Enter New Stack Frame*														
Level = 0	C8 ++[8-bit Level]	-	-	-	-	-	-	-	-	-	7	7	b	h
Level = 1											10	10		
Level (L) > 1											6+4*L	6+4*L		
HLT *Halt*	F4	-	-	-	-	-	-	-	-	-	3	3		l
IDIV *Integer (Signed) Divide*														
Accumulator by Register/Memory Divisor: Byte WORD DWORD	F [011w] [mod 111 r/m]	-	-	-	-	-	-	-	-	-	 19/20 27/28 43/44	 19/20 27/28 43/44	b,e	e,h
IMUL *Integer (Signed) Multiply*														

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags: O F	D F	I F	T F	S F	Z F	A F	P F	C F	Real Mode Clock Count (Reg/Cache Hit)	Prot'd Mode Clock Count (Reg/Cache Hit)	Real Mode Notes	Prot'd Mode Notes
Accumulator by Register/Memory Multiplier: Byte WORD DWORD	F [011w] [mod 101 r/m]	x	-	-	-	-	-	-	-	x	 3/5 3/5 7/9	 3/5 3/5 7/9	b	h
Register with Register/Memory Multiplier: WORD Doubleword	0F AF [mod reg r/m]										3/5 3/5 7/9	3/5 3/5 7/9		
Register/Memory with Immediate to Register2 Multiplier: Word DWORD	6 [10s1] [mod reg r/m] #										3/5 3/5 7/9	3/5 3/5 7/9		
IN *Input from I/O Port*														
Fixed Port	E [010w] [port number]	-	-	-	-	-	-	-	-	-	16	6/19		m
Variable Port	E [110w]										16	6/19		
INS *Input String from I/O Port*	6 [110w]	-	-	-	-	-	-	-	-	-	20	6/19	b	h,m
INC *Increment by 1*														
Register/Memory	F [111w] [mod 000 r/m]	x	-	-	-	x	x	x	x	-	1/3	1/3	b	h
Register (short form)	4 [0 reg]										1	1		
INT *Software Interrupt*														

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags									Real Mode	Prot'd Mode	Real Mode	Prot'd Mode
		OF	DF	IF	TF	SF	ZF	AF	PF	CF	Clock Count (Reg/Cache Hit)		Notes	
INT i	CD [i]	-	x	0		-	-	-	-	-			b,e	g,j,k,r
Protected Mode:											14			
-Interrupt or Trap to Same Privilege												49		
-Interrupt or Trap to Different Privilege												77		
-16-bit Task to 16-bit TSS by Task Gate												233		
-16-bit Task to 32-bit TSS by Task Gate												260		
-16-bit Task to V86 by Task Gate												177		
-16-bit Task to 16-bit TSS by Task Gate												236		
-32-bit Task to 32-bit TSS by Task Gate												263		
-32-bit Task to V86 by Task Gate												180		
-V86 to 16-bit TSS by Task Gate												236		
-V86 to 32-bit TSS by Task Gate												263		
-V86 to Privilege 0 by Trap Gate/Int Gate												93		
INT 3	CC										14			
Protected Mode														
Interrupt or Trap to Same Privilege												49		
Interrupt or Trap to Different Priv.												77		
16-bit Tsk - 16-bit TSS by Tsk Gate												233		
16-bit Tsk - 32-bit TSS by Tsk Gate												260		
16-bit Task to V86 by Task Gate												177		
32-bit Tsk - 16-bit TSS by Tsk Gate												236		
32-bit Tsk - 32-bit TSS by Tsk Gate												263		
32-bit Task to V86 by Task Gate												180		
V86 to 16-bit TSS by Task Gate												236		
V86 to 32-bit TSS by Task Gate												263		
V86 to Priv. 0 by Trap Gate/Int Gate												93		
INTO	CE													
If OF==0											1	1		
If OF==1 (INT 4)											15			
Protected Mode														
Interrupt or Trap to Same Privilege												49		
Interrupt or Trap to Different Priv.												77		
16-bit Tsk - 16-bit TSS by Tsk Gate												233		
16-bit Tsk - 32-bit TSS by Tsk Gate												260		
16-bit Task to V86 by Task Gate												177		
32-bit Tsk - 16-bit TSS by Tsk Gate												236		
32-bit Tsk - 32-bit TSS by Tsk Gate												263		
32-bit Task to V86 by Task Gate												180		
V86 to 16-bit TSS by Task Gate												236		
V86 to 32-bit TSS by Task Gate												263		
V86 to Priv. 0 by Trap Gate/Int Gate												93		
INVD *Invalidate Cache*	0F 08	-	-	-	-	-	-	-	-	-	4	4		
INVLPG *Invalidate TLB Entry*	0F 01 [mod 111 r/m]	-	-	-	-	-	-	-	-	-	4	4		

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags	Real Mode	Prot'd Mode	Real Mode	Prot'd Mode
		O D I T S Z A P C F F F F F F F F F	Clock Count (Reg/Cache Hit)		Notes	
IRET *Interrupt Return*	CF	x x x x x x x x x				g,h,j,k,r
Real Mode			14			
Protected Mode:						
-Within Task to Same Privilege				31		
-Within Task to Different Privilege				66		
-16-bit Task to 16-bit Task				229		
-16-bit Task to 32-bit TSS				256		
-16-bit Task to V86 Task				173		
-32-bit Task to 16-bit TSS				232		
-32-bit Task to 32-bit TSS				259		
-32-bit Task to V86 Task				176		
JB/JNAE/JC *Jump on Below/Not Above or Equal/Carry*						
8-bit Displacement	72 +	- - - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 82 +++		4 | 1	6 | 1		
JBE/JNA *Jump on Below or Equal/Not Above*						
8-bit Displacement	76 +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 86 +++		4 | 1	6 | 1		
JCXZ/JECXZ *Jump on CX/ECX Zero*	E3 +	- - - - - - - - -	7 | 3	7 | 3		r
JE/JZ *Jump on Equal/Zero*						
8-bit Displacement	74 +	- - - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 84 +++		4 | 1	6 | 1		
JECXZ *Jump on EXC Zero*	E3 +	- - - - - - - - -	7 | 3	7 | 3		r
JL/JNGE *Jump on Less/Not Greater or Equal*						
8-bit Displacement	7C +	- - - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 8C +++		4 | 1	6 | 1		
JLE/JNG *Jump on Less or Equal/Not Greater*						
8-bit Displacement	7E +	- - - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 8E +++		4 | 1	6 | 1		
JMP *Unconditional Jump*						
Short	EB +	- - - - - - - - -	4	6	b	h,j,k,r
Direct within Segment	E9 +++		4	6		
Register/Memory Indirect Within Segment	FF [mod 100 r/m]		6/8	6/8		
Direct Intersegment	EA [full offset, selector]		9	26		
Call Gate Same Privilege Level				37		
16-bit Task to 16-bit TSS				238		
16-bit Task to 32-bit TSS				265		
16-bit Task to V86 Task				182		
32-bit Task to 16-bit TSS				241		
32-bit Task to 32-bit TSS				268		
32-bit Task to V86 Task				185		
Indirect Intersegment	FF [mod 101 r/m]			30		
Call Gate Same Privilege Level				39		
16-bit Task to 16-bit TSS				240		
16-bit Task to 32-bit TSS				267		
16-bit Task to V86 Task				184		
32-bit Task to 16-bit TSS				243		
32-bit Task to 32-bit TSS				270		
32-bit Task to V86 Task				187		
JNB/JAE/JNC *Jump on Not Below/Above or Equal/Not Carry*						
8-bit Displacement	73 +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 83 +++		4 | 1	6 | 1		
JNBE/JA *Jump on Not Below or Equal/Above*						

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags OF DF IF TF SF ZF AF PF CF	Real Mode Clock Count (Reg/Cache Hit)	Prot'd Mode Clock Count (Reg/Cache Hit)	Real Mode Notes	Prot'd Mode Notes
8-bit Displacement	77 +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 87 +++		4 | 1	6 | 1		
JNE/JNZ *Jump on Not Equal/Not Zero*						
8-bit Displacement	75 +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 85 +++		4 | 1	6 | 1		
JNL/JGE *Jump on Not Less/Greater or Equal*						
8-bit Displacement	7D +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 8D +++		4 | 1	6 | 1		
JNLE/JG *Jump on Not Less or Equal/Greater*						
8-bit Displacement	7F +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 8F +++		4 | 1	6 | 1		
JNO *Jump on Not Overflow*						
8-bit Displacement	71 +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 81 +++		4 | 1	6 | 1		
JNP/JPO *Jump on Not Parity/Parity Odd*						
8-bit Displacement	7B +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 8B +++		4 | 1	6 | 1		
JNS *Jump on Not Sign*						
8-bit Displacement	79 +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 89 +++		4 | 1	6 | 1		
JO *Jump on Overflow*						
8-bit Displacement	70 +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 80 +++		4 | 1	6 | 1		
JP/JPE *Jump on Parity/Parity Even*						
8-bit Displacement	7A +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 8A +++		4 | 1	6 | 1		
JS *Jump on Sign*						
8-bit Displacement	78 +	- - - - - - - -	4 | 1	6 | 1		r
Full Displacement	0F 88 +++		4 | 1	6 | 1		
LAHF *Load AH with Flags*	9F	- - - - - - - - -	2	2		
LAR *Load Access Rights*						
From Register/Memory	0F 02 [mod reg r/m]	- - - - - x - - -		11/12	a	g,h,j,p
LDS *Load Pointer to DS*	C5 [mod reg r/m]	- - - - - - - - -	6	19	b	h,i,j
LEA *Load Effective Address*						
No Index Register	8D [mod reg r/m]	- - - - - - - - -	2	2		
With Index Register			3	3		
LEAVE *Leave Current Stack Frame*	C9	- - - - - - - - -	3	3	b	h
LES *Load Pointer to ES*	C4 [mod reg r/m]	- - - - - - - - -	6	19	b	h,i,j
LFS *Load Pointer to FS*	0F B4 [mod reg r/m]	- - - - - - - - -	6	19	b	h,i,j
LGDT *Load GDT Register*	0F 01 [mod 010 r/m]	- - - - - - - - -	9	9	b,c	h,l
LGS *Load Pointer to GS*	0F B5 [mod reg r/m]	- - - - - - - - -	6	19	b	h,i,j
LIDT *Load IDT Register*	0F 01 [mod 011 r/m]	- - - - - - - - -	9	9	b,c	h,l
LLDT *Load LDT Register*						
From Register/Memory	0F 00 [mod 010 r/m]	- - - - - - - - -		16/17	a	g,h,j,l
LMSW *Load Machine Status Word*						

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags O D I T S Z A P C F F F F F F F F F	Real Mode Clock Count (Reg/Cache Hit)	Prot'd Mode Clock Count (Reg/Cache Hit)	Real Mode Notes	Prot'd Mode Notes
From Register/Memory	0F 01 [mod 110 r/m]	- - - - - - - - -	5	5	b,c	h,l
LODS *Load String*	A [110 w]	- - - - - - - - -	4	4	b	h
LOOP — Offset Loop/No Loop	E2 +	- - - - - - - - -	7 | 3	9 | 3		r
LOOPNZ/LOOPNE — Offset	E0 +	- - - - - - - - -	7 | 3	9 | 3		r
LOOPZ/LOOPE — Offset	E1 +	- - - - - - - - -	7 | 3	9 | 3		r
LSL *Load Segment Limit*						
From Register/Memory	0F 03 [mod reg r/m]	- - - - - x - - -		14/15	a	g,h,j,p
LSS *Load Pointer to SS*	0F B2 [mod reg r/m]	- - - - - - - - -	6	19	a	h,i,j
LTR *Load Task Register*						
From Register/Memory	0F 00 [mod 011 r/m]	- - - - - - - - -		16/17	a	g,h,j,l
MOV *Move Data*						
Register to Register/Memory	8 [100w] [mod reg r/m]	- - - - - - - - -	1/2	1/2	b	h,i,j
Register/Memory to Register	8 [100w] [mod reg r/m]		1/2	1/2		
Immediate to Register/Memory	C [011w] [mod 000 r/m] ###		1/2	1/2		
Immediate to Register (short form)	B [w reg] ###		1	1		
Memory to Accumulator (short form)	A [000w] +++		2	2		
Accumulator to Memory (short form)	A [001w] +++		1/2	1/2		
Register/Memory to Segment Register	8E [mod sreg3 r/m]		2/3	15/16		
Segment Register to Register/Memory	8C [mod sreg3 r/m]		1/2	1/2		
MOV *Move to/from Control/Debug/Test Regs*		- - - - - - - - -				l
Register to CR0/CR2/CR3	0F 22 [11 eee reg]	- - - - - - - - -	11/3/3	11/3/3		l
CR0/CR2/CR3 to Register	0F 20 [11 eee reg]		1/3/3	1/3/3		
Register to DR0-DR3	0F 23 [11 eee reg]		1	1		
DR0-DR3 to Register	0F 21 [11 eee reg]		3	3		
Register to DR6-DR7	0F 23 [11 eee reg]		1	1		
DR6-DR7 to Register	0F 21 [11 eee reg]		3	3		
Register to TR3-5	0F 26 [11 eee reg]		5	5		
TR3-5 to Register	0F 24 [11 eee reg]		5	5		
Register to TR6-TR7	0F 26 [11 eee reg]		1	1		
TR6-TR7 to Register	0F 24 [11 eee reg]		3	3		
MOVS *Move String*	A [010w]	- - - - - - - - -	5	5	b	h
MOVSX *Move with Sign Extension*						
Register from Register/Memory	0F B[111w] [mod reg r/m]	- - - - - - - - -	1/3	1/3	b	h
MOVZX *Move with Zero Extension*						
Register from Register/Memory	0F B[011w] [mod reg r/m]	- - - - - - - - -	2/3	2/3	b	h
MUL *Unsigned Multiply*						
Accumulator with Register/Memory Multiplier: Byte WORD DWORD	F [011w] [mod 100 r/m]	x - - - x	 3/5 3/5 7/9	 3/5 3/5 7/9	b	h
NEG *Negate Integer*	F [011w] [mod 011 r/m]	x - - - x x x x x	1/3	1/3	b	h
NOP *No Operation*	90	- - - - - - - - -	1	1		
NOT *Boolean Complement*	F [011w] [mod 010 r/m]	- - - - - - - - -	1/3	1/3	b	h
OIO *Official Invalid Opcode*	0F FF	- - 0 - - - - - -	15	49-300	t	t

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags O D I T S Z A P C F F F F F F F F F	Real Mode Clock Count (Reg/Cache Hit)	Prot'd Mode Clock Count (Reg/Cache Hit)	Real Mode Notes	Prot'd Mode Notes
OR *Boolean OR*						
Register to Register	0 [10dw] [11 reg r/m]	0 - - - x x x x 0	1	1	b	h
Register to Memory	0 [100w] [mod reg r/m]		3	3		
Memory to Register	0 [101w] [mod reg r/m]		3	3		
Immediate to Register/Memory	8 [00sw] [mod 001 r/m] ###		1/3	1/3		
Immediate to Accumulator	0 [110w] ###		1	1		
OUT *Output to Port*						
Fixed Port	E [011w] #	- - - - - - - - -	18	4/17		m
Variable Port	E [111w]		18	4/17		
OUTS *Output String*	6 [111w]	- - - - - - - - -	20	6/19	b	h,m
POP *Pop Value off Stack*						
Register/Memory	8F [mod 000 r/m]	- - - - - - - - -	3/5	3/5	b	h,i,j
Register (short form)	5 [1 reg]		3	3		
Segment Register (ES, CS, SS, DS)	[000 sreg2 111]		4	18		
Segment Register (ES, CS, SS, DS, FS, GS)	0F [10 sreg3 001]		4	18		
POPA *Pop All General Registers*	61	- - - - - - - - -	18	18	b	h
POPF *Pop Stack into FLAGS*	9D	x x x x x x x x x	4	4	b	h,n
PREFIX BYTES						
Assert Hardware LOCK Prefix	F0	- - - - - - - - -				m
Address Size Prefix	67					
Operand Size Prefix	66					
Segment Override Prefix -CS -DS -ES -FS -GS -SS	 2E 3E 26 64 65 36					
PUSH *Push Value onto Stack*						
Register/Memory	FF [mod 110 r/m]	- - - - - - - - -	2/4	2/4	b	h
Register (short form)	5 [0 reg]		2	2		
Segment Register (ES, CS, SS, DS)	[000 sreg2 110]		2	2		
Segment Register (ES. CS. SS, DS, FS, GS)	0F [10 sreg3 000]		2	2		
Immediate	6 [10s0] ###		2	2		
PUSHA *Push All General Registers*	60	- - - - - - - - -	17	17	b	h
PUSHF *Push FLAGS Register*	9C	- - - - - - - - -	2	2	b	h
RCL *Rotate Through Carry Left*						
Register/Memory by 1	D [000w] [mod 010 r/m]	x - - - - - - - x	9/9	9/9	b	h
Register/Memory by CL	D [001w] [mod 010 r/m]	u - - - - - - - x	9/9	9/9		
Register/Memory by Immediate	C [000w] [mod 010 r/m] #	u - - - - - - - x	9/9	9/9		
RCR *Rotate Through Carry Right*						
Register/Memory by 1	D [000w] [mod 011 r/m]	x - - - - - - - x	9/9	9/9	b	h
Register/Memory by CL	D [001w] [mod 011 r/m]	u - - - - - - - x	9/9	9/9		
Register/Memory by Immediate	C [000w] [mod 011 r/m] #	u - - - - - - - x	9/9	9/9		

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags O D I T S Z A P C F F F F F F F F F	Real Mode Clock Count (Reg/Cache Hit)	Prot'd Mode Clock Count (Reg/Cache Hit)	Real Mode Notes	Prot'd Mode Notes
REP INS *Input String*	F2 6[110w]	- - - - - - - - -	20+9n	5+9n\ 18+9n	b	h,m
REP LODS *Load String*	F2 A[110w]	- - - - - - - - -	4+5n	4+5n	b	h
REP MOVS *Move String*	F2 A[010w]	- - - - - - - - -	5+4n	5+4n	b	h
REP OUTS *Output String*	F2 6[111w]	- - - - - - - - -	20+4n	5+4n\ 18+4n	b	h,m
REP STOS *Store String*	F2 A[101w]	- - - - - - - - -	3+4n	3+4n	b	h
REPE CMPS *Compare String*						
Find non-match	F3 A[011w]	x - - - x x x x x	5+8n	5+8n	b	h
REPE SCAS *Scan String*						
Find non-AL/AX/EAX	F3 A[111w]	x - - - x x x x x	4+5n	4+5n	b	h
REPNE CMPS *Compare String*						
Find match	F2 A[011w]	x - - - x x x x x	5+8n	5+8n	b	h
REPNE SCAS *Scan String*						
Find AL/AX/EAX	F2 A[111w]	x - - - x x x x x	4+5n	4+5n	b	h
RET *Return from Subroutine*						
Within Segment	C3	- - - - - - - - -	10	10	b	g,h,j,k,r
Within Segment Adding Immediate to SP	C2 ##		10	10		
Intersegment	CB		13	26		
Intersegment Adding Immediate to SP	CA ##		13	26		
Protected Mode: Different Privilege Level -Intersegment -Intersegment Adding Immediate to SP				 61 61		
ROL *Rotate Left*						
Register/Memory by 1	D[000w] [mod 000 r/m]	x - - - - - - - x	2/4	2/4	b	h
Register/Memory by CL	D[001w] [mod 000 r/m]		3/5	3/5		
Register/Memory by Immediate	C[000w] [mod 000 r/m] #		2/4	2/4		
ROR *Rotate Right*						
Register/Memory by 1	D[000w] [mod 001 r/m]	x - - - - - - - x	2/4	2/4	b	h
Register/Memory by CL	D[001w] [mod 001 r/m]		3/5	3/5		
Register/Memory by Immediate	C[000w] [mod 001 r/m] #		2/4	2/4		
RSDC *Restore Segment Register and Descriptor*	0F 79 [mod sreg3 r/m]	- - - - - - - - -	10	10	s	s
RSLDT *Restore LDTR and Descriptor*	0F 7B [mod 000 r/m]	- - - - - - - - -	10	10	s	s
RSM *Resume from SMM Mode*	0F AA	-x - - - - - - - -	76	76	s	s
RSTS *Restore TSR and Descriptor*	0F 7D [mod 000 r/m]	- - - - - - - - -	10	10	s	s
SAHF *Store AH in FLAGS*	9E	x - - - x x - x x	2	2		
SAL *Shift Left Arithmetic*						
Register/Memory by 1	D[000w] [mod 100 r/m]	x - - - x x - x x	2/4	2/4	b	h
Register/Memory by CL	D[001w] [mod 100 r/m]	u - - - x x u x x	3/5	3/5		
Register/Memory by Immediate	C[000w] [mod 100 r/m] #	u - - - x x u x x	2/4	2/4		
SAR *Shift Right Arithmetic*						
Register/Memory by 1	D[000w] [mod 111 r/m]	x - - - x x x x x	2/4	2/4	b	h
Register/Memory by CL	D[001w] [mod 111 r/m]		3/5	3/5		
Register/Memory by Immediate	C[000w] [mod 111 r/m] #		2/4	2/4		

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags: OF DF IF TF SF ZF AF PF CF	Real Mode Clock Count (Reg/Cache Hit)	Prot'd Mode Clock Count (Reg/Cache Hit)	Real Mode Notes	Prot'd Mode Notes
SBB *Integer Subtract with Borrow*						
Register to Register	1[10dw] [11 reg r/m]	x - - - x x x x x	1	1	b	h
Register to Memory	1[100w] [mod reg r/m]		3	3		
Memory to Register	1[101w] [mod reg r/m]		3	3		
Immediate to Register/Memory	8[00sw] [mod 011 r/m] ###		1/3	1/3		
Immediate to Accumulator (short form)	1[110w] ###		1	1		
SCAS *Scan String*	A [111w]	x - - - x x x x x	5	5	b	h
SETB/SETNAE/SETC *Set Byte on Below/Not Above or Equal to Register/Memory*						
	0F 92 [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETBE/SETNA *Set Byte on Below or Equal/Not Above to Register/Memory*						
	0F 96 [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETE/SETZ *Set Byte on Equal/Zero to Register/Memory*						
	0F 94 [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETL/SETNGE *Set Byte on Less/Not Greater or Equal to Register/Memory*						
	0F 9C [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETLE/SETNG *Set Byte on Less or Equal/Not Greater to Register/Memory*						
	0F 9E [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETNB/SETAE/SETNC *Set Byte on Not Below/Above or Equal/Not Carry to Register/Memory*						
	0F 93 [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETNBE/SETA *Set Byte on Not Below or Equal/Above to Register/Memory*						
	0F 97 [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETNE/SETNZ *Set Byte on Not Equal/Not Zero to Register/Memory*						
	0F 95 [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETNL/SETGE *Set Byte on Not Less/Greater or Equal to Register/Memory*						
	0F 9D [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETNLE/SETG *Set Byte on Not Less or Equal/Greater to Register/Memory*						
	0F 9F [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETNO *Set Byte on Not Overflow to Register/Memory*						
	0F 91 [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETNP/SETPO *Set Byte on Not Parity/Parity Odd to Register/Memory*						
	0F 9B [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETNS *Set Byte on Not Sign to Register/Memory*						
	0F 99 [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETO *Set Byte on Overflow to Register/Memory*						
	0F 90 [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETP/SETPE *Set Byte on Parity/Parity Even to Register/Memory*						
	0F 9A [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SETS *Set Byte on Sign to Register/Memory to Register/Memory*						
	0F 98 [mod 000 r/m]	- - - - - - - - -	2/2	2/2		h
SGDT *Store GDT Register to Register/Memory*						
	0F 01 [mod 000 r/m]	- - - - - - - - -	6	6	b,c	h
SIDT *Store IDT Register to Register/Memory*						
	0F 01 [mod 001 r/m]	- - - - - - - - -	6	6	b,c	h
SLDT *Store LDT Register to Register/Memory*						
	0F 01 [mod 000 r/m]	- - - - - - - - -		1/2	a	h
STR *Store Task Register to Register/Memory*						

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags O D I T S Z A P C F F F F F F F F F	Real Mode Clock Count (Reg/Cache Hit)	Prot'd Mode Clock Count (Reg/Cache Hit)	Real Mode Notes	Prot'd Mode Notes
	0F 00 [mod 001 r/m]	- - - - - - - - - -		1/2	a	h
SMSW *Store Machine Status Word*	0F 01 [mod 100 r/m]	- - - - - - - - - -	1/2	1/2	b,c	h
STOS *Store String*	A [101w]	- - - - - - - - - -	3	3	b	h
SHL *Shift Left Logical*						
Register/Memory by 1	D [000w] [mod 100 r/m]	x - - - x x - x x	1/3	1/3	b	h
Register/Memory by CL	D [001w] [mod 100 r/m]		2/4	2/4		
Register/Memory by Immediate	C [000w] [mod 100 r/m] #		1/3	1/3		
SHLD *Shift Left Double*						
Register/Memory by Immediate	0F A4 [mod reg r/m] #	- - - - x x - x x	1/3	1/3		
Register/Memory by CL	0F A5 [mod reg r/m]		3/5	3/5		
SHR *Shift Right Logical*						
Register/Memory by 1	D [000w] [mod 101 r/m]	x - - - x x - x x	1/3	1/3		
Register/Memory by CL	D [001w] [mod 101 r/m]		2/4	2/4		
Register/Memory by Immediate	C [000w] [mod 101 r/m] #		1/3	1/3		
SHRD *Shift Right Double*						
Register/Memory by Immediate	0F AC [mod reg r/m] #	- - - - x x - x x	1/3	1/3		
Register/Memory by CL	0F AD [mod reg r/m]		3/5	3/5		
SMINT *Software SMM Entry*	0F 7E	- - - - - - - - - -	24	24	s	s
STC *Set Carry Flag*	F9	- - - - - - - - - 1	1	1		
STD *Set Direction Flag*	FD	- 1 - - - - - - - -	1	1		
STI *Set Interrupt Flag*	FB	- - 1 - - - - - - -	7	7		m
SUB *Integer Subtract*						
Register to Register	2 [10dw] [11 reg r/m]	x - - - x x x x x	1	1	b	h
Register to Memory	2 [100w] [mod reg r/m]		3	3		
Memory to Register	2 [101w] [mod reg r/m]		3	3		
Immediate to Register/Memory	8 [00sw] [mod 101 r/m] ###		1/3	1/3		
Immediate to Accumulator (short form)	2 [110w] ###		1	1		
SVDC *Save Segment Register and Descriptor*	0F 78 [mod sreg3 r/m]	- - - - - - - - - -	18	18	s	s
SVLDT *Save LDTR and Descriptor*	0F 7A [mod 000 r/m]	- - - - - - - - - -	18	18	s	s
SVTS *Save TSR and Descriptor*	0F 7C [mod 000 r/m]	- - - - - - - - - -	18	18	s	s
TEST *Test Bits*						
Register/Memory and Register	8 [010w] [mod reg r/m]	0 - - - x x - x 0	1/3	1/3	b	h
Immediate Data and Register/Memory	F [011w] [mod 000 r/m] #		1/3	1/3		
Immediate Data and Accumulator	A [100w] #		1	1		
VERR *Verify Read Acces to Register/Memory*						
	0F 00 [mod 100 r/m]	- - - - - x - - -		9/10	a	g,h,j,p
VERW *Verify Write Access to Register/Memory*						
	0F 00 [mod 101 r/m]	- - - - - x - - -		9/10	a	g,h,j,p
WAIT *Wait Until FPU Not Busy*	9B	- - - - - - - - - -	5	5		
WBINVD *Write-Back and Invalidate Cache*	0F 09	- - - - - - - - - -	4	4		
XADD *Exchange and Add*						

Table 2.56 Processor Core Instruction Set Summary (cont.)

Instruction	Opcode	Flags O F	D F	I F	T F	S F	Z F	A F	P F	C F	Real Mode Clock Count (Reg/Cache Hit)	Prot'd Mode Clock Count (Reg/Cache Hit)	Real Mode Notes	Prot'd Mode Notes
Register1, Register2	0F C[000w] [11 reg2 reg1]	x	-	-	-	x	x	x	x	x	3	3		
Memory, Register	0F C[000w] [mod reg r/m]										6	6		
XCHG *Exchange*														
Register/Memory with Register	8[011w] [mod reg r/m]	-	-	-	-	-	-	-	-	-	3/4	3/4	b,f	f,h
Register with Accumulator	9[0 reg]										3	3		
XLAT *Translate Byte*	D7	-	-	-	-	-	-	-	-	-	3	3		h
XOR *Boolean Exclusive OR*														
Register to Register	3 [00dw] [11 reg r/m]	0	-	-	-	x	x	-	x	0	1	1	b	h
Register to Memory	3 [000w] [mod reg r/m]										3	3		
Memory to Register	3 [001w] [mod reg r/m]										3	3		
Immediate to Register/Memory	8 [00sw] [mod 110 r/m] #										1/3	1/3		
Immediate to Accumulator (short form)	3 [010w] #										1	1		

Instruction Notes for Instruction Set Summary

Notes a through c apply to Real Address Mode only:

a. This is a Protected Mode instruction. Attempted execution in Real Mode will result in exception 6 (invalid op-code).

b. Exception 13 fault (general protection) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum CS, DS, ES, FS, or GS segment limit (FFFFH). Exception 12 fault (stack segment limit violation or not present) will occur in Real Mode if an operand reference is made that partially or fully extends beyond the maximum SS limit.

c. This instruction may be executed in Real Mode. In Real Mode, its purpose is primarily to initialize the CPU for Protected Mode.

Note: E through g apply to Real Address Mode and Protected Virtual Address Mode:

d. An exception may occur, depending on the value of the operand.

e. LOCK# is automatically asserted, regardless of the presence or absence of the LOCK prefix.

f. LOCK# is asserted during descriptor table accesses.

Note: H through r apply to Protected Virtual Address Mode only:

g. Exception 13 fault will occur if the memory operand in CS, DS, ES, FS, or GS cannot be used due to either a segment limit violation or an access rights violation. If a stack limit is violated, an exception 12 occurs.

h. For segment load operations, the CPL, RPL, and DPL must agree with the privilege rules to avoid an exception 13 fault. The segment's descriptor must indicate "present" or exception 11 (CS, DS, ES, FS, GS not present). If the SS register is loaded and a stack segment not present is detected, an exception 12 occurs.

i. All segment descriptor accesses in the GDT or LDT made by this instruction will automatically assert LOCK# to maintain descriptor integrity in multiprocessor systems.

j. JMP, CALL, INT, RET, and IRET instructions referring to another code segment will cause an exception 13, if an applicable privilege rule is violated.

k. An exception 13 fault occurs if CPL is greater than 0 (0 is the most privileged level).

l. An exception 13 fault occurs if CPL is greater than IOPL.

m. The IF[9] bit of the EFLAG register is not updated if CPL is greater than IOPL. The IOPL and VM fields of the flag register are updated only if CPL = 0.

n. The PE bit of the MSW (CR0) cannot be reset by this instruction. Use MOV into CRO if desiring to reset the PE bit.

o. Any violation of privilege rules as apply to the selector operand does not cause a Protection exception, rather, the zero flag is cleared.

p. If the coprocessor's memory operand violates a segment limit or segment access rights, an exception 13 fault will occur before the ESC instruction is executed. An exception 12 fault will occur if the stack limit is violated by the operand's starting address.

q. The destination of a JMP, CALL, INT, RET, or IRET must be in the defined limit of a code segment or an exception 13 fault will occur.

Note: S applies to processor specific SMM instructions:

r. All memory accesses to SMM space are non-cacheable. An invalid opcode exception 6 occurs unless SMI is enabled and SMAR size > 0, and CPL = 0 and [SMAC is set or if in an SMI handler].

s. As requested by Microsoft® Corporation.

2.3.2.4. FPU Clock Counts

The CPU can be divided into the FPU which processes floating point instructions and the remaining circuity collectively called the integer unit. The FPU can execute instructions independently of the integer unit. For example, the integer unit can issue a floating point instruction without memory operands, in two clock cycles and then pass the operation to the FPU to execute. The integer unit will continue to execute instructions until the next floating point instruction is encountered. The FPU loads from memory are similar in that the integer unit issues the FPU instruction, transfers data to the FPU and then is free to execute integer instructions. However, when executing a floating point store, the resources of both the FPU and integer unit are used.

2.3.2.5. Instruction Set Summary

Table 2.58 summarizes the operation and allowed forms of the FPU instruction set.

2.3.2.6. Abbreviations

The abbreviations used in Table 2.57 are listed in the table below:

Table 2.57 FPU Table Abbreviations

Abbreviation	Meaning
n	Stack register number
TOS	Top of stack register pointed to by SSS in the status register.
ST(1)	FPU register next to TOS
ST(n)	A specific FPU register, relative to TOS
M.WI	16-bit integer operand from memory
M.SI	32-bit integer operand from memory
M.LI	64-bit integer operand from memory
M.SR	32-bit real operand from memory
M.DR	64-bit real operand from memory

Table 2.57 FPU Table Abbreviations

M.XR	80-bit real operand from memory
M.BCD	18-digit BCD integer operand from memory
CC	FPU condition code
Env Regs	Status, Mode Control and Tag Registers, Instruction Pointer and Operand Pointer

Table 2.58 MMX Instruction Set Summary

MMX Instructions	Opcode	Operation	Clock Count	Notes
F2XM1 *Function Evaluation 2^x-1*	D9 F0	TOS <— 2^{TOS}-1	98 - 114	Note 2
FABS *Floating Absolute Value*	D9 E1	TOS <— | TOS |	5	
FADD *Floating Point Add*				
Top of Stack	DC [1100 0 n]	ST(n) <— ST(n) + TOS	10 - 16	
80-bit Register	D8 [1100 0 n]	TOS <— TOS + ST(n)	10 - 16	
64-bit Real	DC [mod 000 r/m]	TOS <— TOS + M.DR	13 - 19	
32-bit Real	D8 [mod 000 r/m]	TOS <— TOS + M.SR	11 - 17	
FADDP *Floating Point Add, Pop*	DE [1100 0 n]	ST(n) <— ST(n) + TOS; then pop TOS	10 - 16	
FIADD *Floating Point Integer Add*				
32-bit integer	DA [mod 000 r/m]	TOS <— TOS + M.SI	18 - 27	
16-bit integer	DE [mod 000 r/m]	TOS <— TOS + M.WI	18 - 26	
FCHS *Floating Change Sign*	D9 E0	TOS <— TOS	5	
FCLEX *Clear Exceptions*	(9B) DB E2	Wait then Clear Exceptions	8	
FNCLEX *Clear Exceptions*	DB E2	Clear Exceptions	5	
FCOM *Floating Point Compare*				
80-bit Register	D8 [1101 0 n	CC set by TOS - ST(n)	8	
64-bit Real	DC [mod 010 r/m]	CC set by TOS - M.DR	12	
32-bit Real	D8 [mod 010 r/m]	CC set by TOS - M.SR	10	
FCOMP *Floating Point Compare, Pop*				
80-bit Register	D8 [1101 1 n	CC set by TOS - ST(n); then pop TOS	8	
64-bit Real	DC [mod 011 r/m]	CC set by TOS - ST(n); then pop TOS	12	
32-bit Real	D8 [mod 011 r/m]	CC set by TOS - ST(n); then pop TOS	10	
FCOMPP *Floating Point Compare , Pop Two Stack Elements*				
	DE D9	CC set by TOS-ST(1) then pop TOS and ST(1)	8	
FICOM *Floating Point Compare*				
32-bit integer	DA [mod 011 r/m]	CC set by TOS - M.WI	15 - 17	
16-bit integer	DE [mod 011 r/m]	CC set by TOS - M.SI	15 - 16	
FICOMP *Floating Point Compare*				
32-bit integer	DA [mod 011 r/m]	CC set by TOS - M.WI; then pop TOS	15 - 17	
16-bit integer	DE [mod 011 r/m]	CC set by TOS - M.SI; then pop TOS	15 - 16	
FCOS *Function Evaluation: Cos(x)*	D9 FF	TOS <— COS(TOS)	98 - 143	Note 1
FDECSTP *Decrement Stack Pointer*	D9 F6	Decrement top of stack pointer	5	

Table 2.58 MMX Instruction Set Summary (cont.)

MMX Instructions	Opcode	Operation	Clock Count	Notes
FDIV *Floating Point Divide*				
Top of Stack	DC [1111 1 n]	ST(n) <— ST(n) / TOS	28 - 34	
80-bit Register	D8 [1111 0 n]	TOS <— TOS / ST(n)	28 - 34	
64-bit Real	DC [mod 110 r/m]	TOS <— TOS / M.DR	35 - 41	
32-bit Real	D8 [mod 110 r/m]	TOS <— TOS / M.SR	33 - 39	
FDIVP *Floating Point Divide, Pop*	DE [1111 1 n]	ST(n) <— ST(n) / TOS; then pop TOS	28 - 34	
FDIVR *Floating Point Divide Reversed*				
Top of Stack	DC [1111 0 n]	TOS <— ST(n) / TOS	28 - 34	
80-bit Register	D8 [1111 1 n	ST(n) <— TOS / ST(n)	28 - 34	
64-bit Real	DC [mod 111 r/m]	TOS <— M.DR / TOS	35 - 41	
32-bit Real	D8 [mod 111 r/m]	TOS <— M.SR / TOS	33 - 39	
FIDIVRP *Floating Point Integer Divide, Reversed, Pop*				
	DE [1111 0 n]	ST(n) <— TOS / ST(n); then pop TOS	28 - 34	
FIDIV *Floating Point Integer Divide*				
32-bit Integer	DA [mod 110 r/m]	TOS <— TOS / M.SI	36 - 44	
16-bit Integer	DE [mod 110 r/m]	OS <— TOS / M.WI	36 - 43	
FIDIVR *Floating Point Integer Divide Reversed*				
32-bit Integer	DA [mod 111 r/m]	TOS <— M.SI / TOS	36 - 43	
16-bit Integer	DE [mod 111 r/m]	TOS <— M.WI / TOS	36 - 43	
FFREE *Free Floating Point Register*	DD [1100 0 n]	TAG(n) <—Empty	5	
FINCSTP *Increment Stack Pointer*	D9 F7	Increment top of stack pointer	5	
FINIT *Initialize FPU*	(9B)DB E3	Wait then initialize	8	
FNINIT *Initialize FPU*	DB E3	Initialize	5	
FLD *Load Data to FPU Register*				
Top of Stack	D9 [1100 0 n]	Push ST(n) onto stack	4	
80-bit Register	DB [mod 101 r/m]	Push M.XR onto stack	9	
64-bit Real	DD [mod 000 r/m	Push M.DR onto stack	7	
32-bit Real	D9 [mod 000 r/m]	Push M.SR onto stack	5	
FBLD *Load Packed BCD Data to FPU Register*				
	DF [mod 100 r/m]	Push MBCD onto stack	49 - 53	
FILD *Load Integer Data to FPU Register*				
64-bit Integer	DF [mod 101 r/m]	Push M.LI onto stack	9 - 13	
32-bit Integer	DB [mod 000 r/m]	Push M.SI onto stack	8 - 10	
16-bit Integer	DF [mod 000 r/m]	Push 1.0 onto stack	8 - 9	
FLD1 *Load Floating Const. = 1.0*	D9 E8	Push 1.0 onto stack	6	
FLDCW *Load FPU Mode Control Register*				
	D9 [mod 101 r/m]	CTL Word <— Memory	6	
FLDENV *Load FPU Environment*	D9 [mod 100 r/m]	Env Regs <— Memory	28 - 38	
FLDL2E *Load Floating Const. = Log$_2$(e)*	D9 EA	Push Log$_2$(e) onto stack	6	
FLDL2T *Load Floating Const. = Log$_2$(10)*	D9 E9	Push Log$_2$(10) onto stack	6	

Table 2.58 MMX Instruction Set Summary (cont.)

MMX Instructions	Opcode	Operation	Clock Count	Notes
FLDLG2 *Load Floating Const. = $Log_{10}(2)$*	D9 EC	Push $Log_{10}(2)$ onto stack	6	
FLDLN2 *Load Floating Const. = $L_n(2)$*	D9 ED	Push $Log_e(2)$ onto stack	6	
FLDPI *Load Floating Const. = p*	D9 EB	Push π onto stack	6	
FLDZ *Load Floating Const. = 0.0*	D9 EE	Push 0.0 onto stack	6	
FMUL *Floating Point Multiply*				
Top of Stack	DC [1100 1 n]	ST(n) <— ST(n) x TOS	12	
80-bit Register	D8 [1100 1 n]	TOS <— TOS x ST(n)	12	
64-bit Real	DC [mod 001 r/m]	TOS <— TOS x M.DR	15	
32-bit Real	D8 [mod 001 r/m]	TOS <— TOS x M.SR	13	
FMULP *Floating Point Multiply & Pop*	DE [1100 1 n	ST(n) <— ST(n) x TOS, then pop TOS	12	
FIMUL *Floating Point Integer Multiply*				
32-bit Integer	DA [mod 001 r/m]	TOS <— TOS x M.SI	21 - 54	
16-bit Integer	DE [mod 001 r/m]	TOS <— TOS x M.WI	21 - 24	
FNOP *No Operation*	D9 D0	No Operation	3	
FPATAN *Function Eval: $Tan^{-1}(y/x)$*	D9 F3	ST(1) <— ATAN[ST(1) / TOS]; then pop TOS	97 - 161	Note 3
FPREM *Floating Point Remainder*	D9 F8	TOS <— Rem[TOS / ST(1)]	82 - 93	
FPREM1 *Floating Point Remainder IEEE*	D9 F5	TOS <— Rem[TOS / ST(1)]	82 - 93	
FPTAN *Function Eval: Tan(x)*	D9 F2	TOS <— TAN(TOS); then push 1.0 onto stack	123 - 140	Note 1
FRNDINT *Round to Integer*	D9 FC	TOS <— Round(TOS)	12 - 21	
FRSTOR *Load FPU Environment and Register*				
	DD [mod 100 r/m]	Restore state	110 - 120	
FSAVE *Save FPU Environment and Reg*	(9B)DD[mod 110 r/m]	Wait then save state	143 -153	
FNSAVE *Save FPU Environment and Register*				
	DD [mod 110 r/m]	Save state	140 - 150	
FSCALE *Floating Multiply by 2^n*	D9 FD	TOS <— TOS x $2^{(ST(1))}$	10 - 15	
FSIN *Function Evaluation: Sin(x)*	D9 FE	TOS <— SIN(TOS)	81 - 159	Note 1
FSINCOS *Function Eval.: Sin(x)& Cos(x)*	D9 FB	temp <— TOS; TOS <— SIN(temp); then push COS (temp) onto stack	150 - 165	
FSQRT *Floating Point Square Root*	D9 FA	TOS <— Square Root of TOS	61 - 62	
FST *Store FPU Register*				
80-bit Register	DD [1101 0 n]	ST(n) <— TOS	5	
80-bit Real	DB [mod 111 r/m]	M.XR <— TOS	15	
64-bit Real	DD [mod 010 r/m]	M.DR <— TOS	12	
32-bit Real	D9 [mod 010 r/m]	M.SR <— TOS	9	
FSTP *—tore FPU Register, Pop*				
Top of Stack	DB [1101 1 n]	ST(n) <— TOS; then pop TOS	5	
80-bit Real	DB [mod 111 r/m]	M.XR <— TOS; then pop TOS	15	
64-bit Real	DD [mod 011 r/m]	M.DR <— TOS; then pop TOS	12	
32-bit Real	D9 [mod 011 r/m]	M.SR <— TOS; then pop TOS	9	

Table 2.58 MMX Instruction Set Summary (cont.)

MMX Instructions	Opcode	Operation	Clock Count	Notes
FBSTP *Store BCD Data, Pop*	DF [mod 110 r/m]	M.BCD<—TOS; then pop TOS	77 - 82	
FIST *Store Integer FPU Register*				
32-bit Integer	DB [mod 010 r/m]	M.SI <— TOS	16 - 22	
16-bit Integer	DF [mod 010 r/m]	M.WI <— TOS	12 - 18	
FISTP *Store Integer FPU Register, Pop*				
64-bit Integer	DF [mod 111 r/m]	M.LI <— TOS; then pop TOS	19 - 27	
32-bit Integer	DB [mod 011 r/m]	M.SI <— TOS; then pop TOS	16 - 22	
16-bit Integer	DF [mod 011 r/m]	M.WI <— TOS; then pop TOS	12 - 18	
FSTCW *Store FPU Mode Control Register*				
	9B)D9[mod 111 r/m]	Wait Memory <— Control Mode Register	6	
FNSTCW *Store FPU Mode Control Register*				
	D9 [mod 111 r/m]	Memory <— Control Mode Register	3	
FSTENV *Store FPU Environment*	(9B)D9[mod 110 r/m	Wait Memory <— Env. Register	30 - 40	
FNSTENV *Store FPU Environment*	D9 [mod 110 r/m]	Memory <— Env. Registers	27 - 37	
FSTSW *Store FPU Status Register*	(9B)DD[mod 111 r/m]	Wait Memory <— Status Register	6	
FNSTSW *Store FPU Status Register*	DD [mod 111 r/m]	Memory <— Status Register	3	
FSTSW AX *Store FPU Status Reg. to AX*	(9B)DF E0	Wait AX <— Status Register	6	
FNSTSW AX *Store FPU Status Reg to AX*	DF E0	AX <— Status Register	3	
FSUB *Floating Point Subtract*				
Top of Stack	DC [1110 1 n]	ST(n) <— ST(n) - TOS	10 - 16	
80-bit Register	D8 [1110 0 n]	TOS <— TOS - ST(n)	10 - 16	
64-bit Real	DC [mod 100 r/m]	TOS <— TOS - M.DR	13 - 19	
2-bit Real	D8 [mod 100 r/m	TOS <— TOS - M.SR	11 - 17	
FSUBP *Floating Point Subtract, Pop*	DE [1110 1 n]	ST(n) <— ST(n) - TOS; then pop TOS	10 - 16	
FSUBR *Floating Point Subtract Reverse*				
Top of Stack	DC [1110 0 n]	TOS <— ST(n) - TOS	10 - 16	
80-bit Register	D8 [1110 1 n	ST(n) <— TOS - ST(n)	10 - 16	
64-bit Real	DC [mod 101 r/m]	TOS <— TOS - M.DR - TOS	13 - 19	
32-bit Real	D8 [mod 101 r/m]	TOS <— TOS - M.SR - TOS	11 - 17	
FSUBRP *Floating Point Subtract Reverse, Pop*	DE [1110 0 n]	TOS <— TOS - ST(n); then pop TOS	10 - 16	
FISUB *Floating Point Integer Subtract*				
32-bit Integer	DA [mod 100 r/m]	TOS <— TOS - M.SI	18 - 27	
16-bit Integer	DE [mod 100 r/m]	TOS <— TOS - M.WI	18 - 26	
FISUBR *Floating Point Integer Subtract Reverse*				
32-bit Integer Reversed	DA [mod 101 r/m]	TOS <— M.SI - TOS	18 - 27	
16-bit Integer Reversed	DE [mod 101 r/m]	TOS <— M.WI - TOS	18 - 26	
FTST *Test Top of Stack*	D9 E4	CC set by TOS - 0.0	10	
FUCOM *Unordered Compare*	DD [1110 0 n]	CC set by TOS - ST(n)	8	
FUCOMP *Unordered Compare, Pop*	DD [1110 1 n]	CC set by TOS - ST(n); then pop TOS	8	
FUCOMPP *Unordered Compare, Pop two elements*				
	DA E9	CC set by TOS - ST(1); then pop TOS & ST(1)	8	

Table 2.58 MMX Instruction Set Summary (cont.)

MMX Instructions	Opcode	Operation	Clock Count	Notes
FWAIT *Wait*	9B	Wait for FPU not busy	3	
FXAM *Report Class of Operand*	D9 E5	CC <— Class of TOS	4	
FXCH *Exchange Register with TOS*	D9 [1100 1 n]	TOS <—>ST(n) Exchange	9	
FXTRACT *Extract Exponent*	D9 F4	temp <— TOS TOS <— exponent (temp); then push significant (temp) onto stack		
FLY2X *Function Eval. y × Log2(x)*	D9 F1	ST(1) <— ST(1) x Log_2(TOS); then pop TOS	145 - 154	
FLY2XP1 *Function Eval. y × Log2(x+1)*	D9 F9	ST(1) <— ST(1) x Log_2(1+TOS);then pop TOS	131 - 133	Note 4

FPU Instruction Summary Notes

All references to TOS and ST(n) refer to stack layout prior to execution.

Notes:

1. Values popped off the stack are discarded.
2. A pop from the stack increments the top of stack pointer.
3. A push to the stack decrements the top of stack pointer.
4. For FCOS, FSIN, FSINCOS and FPTAN, time shown is for absolute value of TOS < 3p/4.
   Add 90 clock counts for argument reduction if outside this range.

   - For FCOS, clock count is 143 if TOS < p/4 and clock count is 98 if p/4 < TOS > p/2.
   - For FSIN, clock count is 81 to 82 if absolute value of TOS < p/4.
5. For F2XM1, clock count is 98 if absolute value of TOS < 0.5.
6. For FPATAN, clock count is 97 if ST(1)/TOS < p/32.
7. For FYL2XP1, clock count is 170 if TOS is out of range and regular FYL2X is called.

2

3. North Bridge

The North Bridge (illustrated in Figure 3-1 Data Paths) is a high performance 32 bit controller. Through the use of synchronous DRAM, high processor to/from system memory bandwidth is supported. This minimizes the need for second level cache. PCI read and write buffers allow system memory accesses to be handled in bursts and hence maximizes system availability to the processor. Lastly, a CPU to PCI write buffer allows the processor to post writes to the PCI and then continue to other tasks.

3.1. North Bridge Features

The North Bridge controller is based on a PC87550 PCI System Controller (North Bridge) designed for a Pentium class processor. Following are the features of the North Bridge Controller.

- CPU single cycle and burst bus transactions support
- Cache coherency support
- CPU level one write back and write through cache support.
- Support for SMM bus cycles
- Support for Level 1 cache flush via CPU FLUSH# pin
- Programmable cache, non-cache and Read-only regions support
- 32 bit data bus structure though out
- Memory Controller to support Synchronous DRAM (SDRAM). Memory can be configured as 16 or 32 bits wide.
- Support for up to four banks of SDRAM and 256 Mbytes of memory space
- CPU bus to PCI bus bridge with PCI arbiter. Support for three external masters and one internal master. Any on chip or off chip master must connect via this interface.
- External PCI bus mastership. External bus mastership of System Controller internal bus. Mastership allows access to system controller memory devices.
- SDRAM Write Buffer – 32 Bytes
- CPU to PCI Write Buffer – 32 Bytes
- PCI Write Buffer – 16 Bytes
- PCI Read Pre-fetch Buffer – Dual 16 Bytes
- Support for Power Management signals from the South Bridge

Figure 3-1 "Data Paths" on page 109 & ‘Block Diagram’ on page 110 provide a detailed look at the control paths of the North Bridge and how the data flows through it.

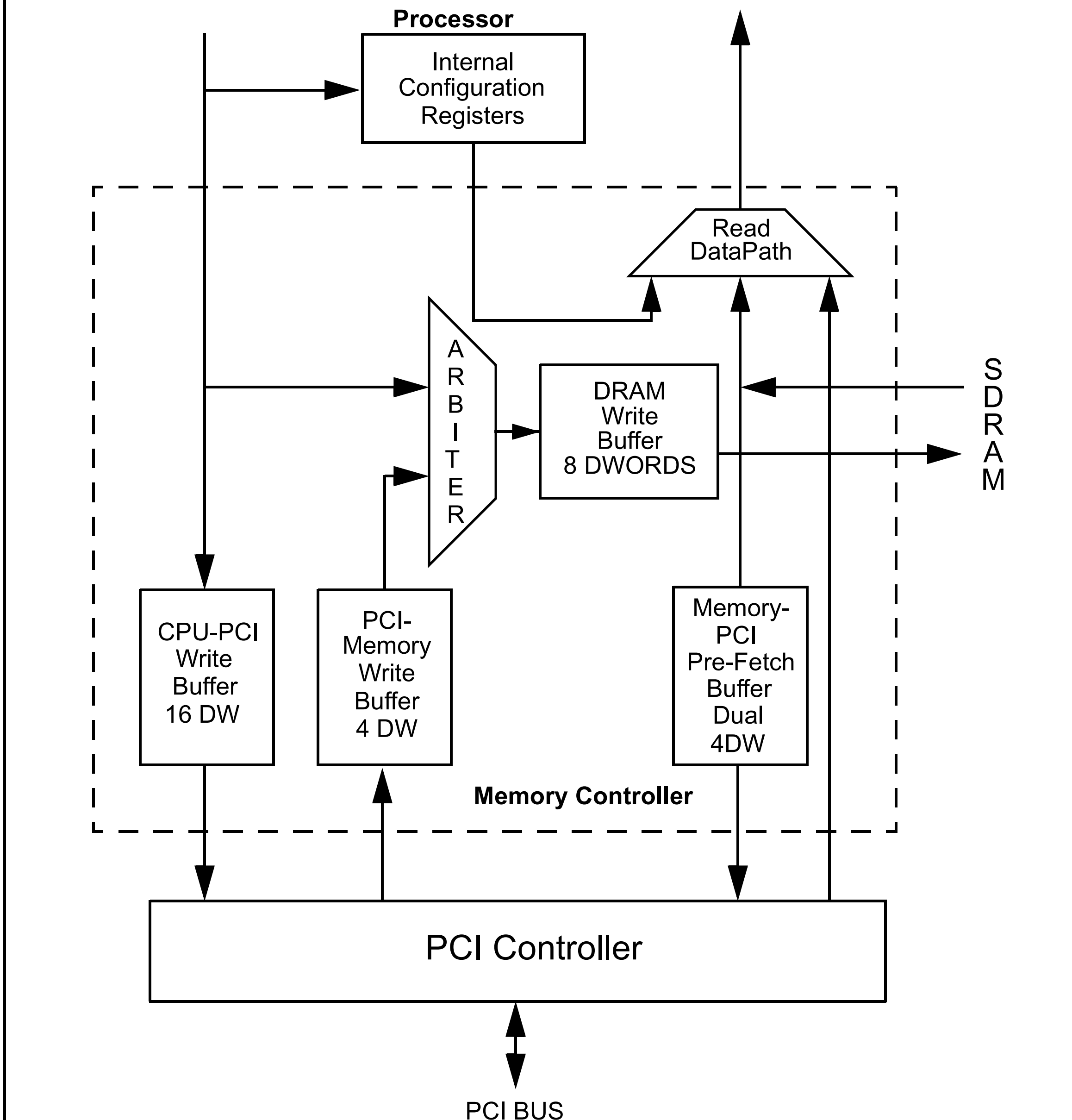


Figure 3-1 Data Paths

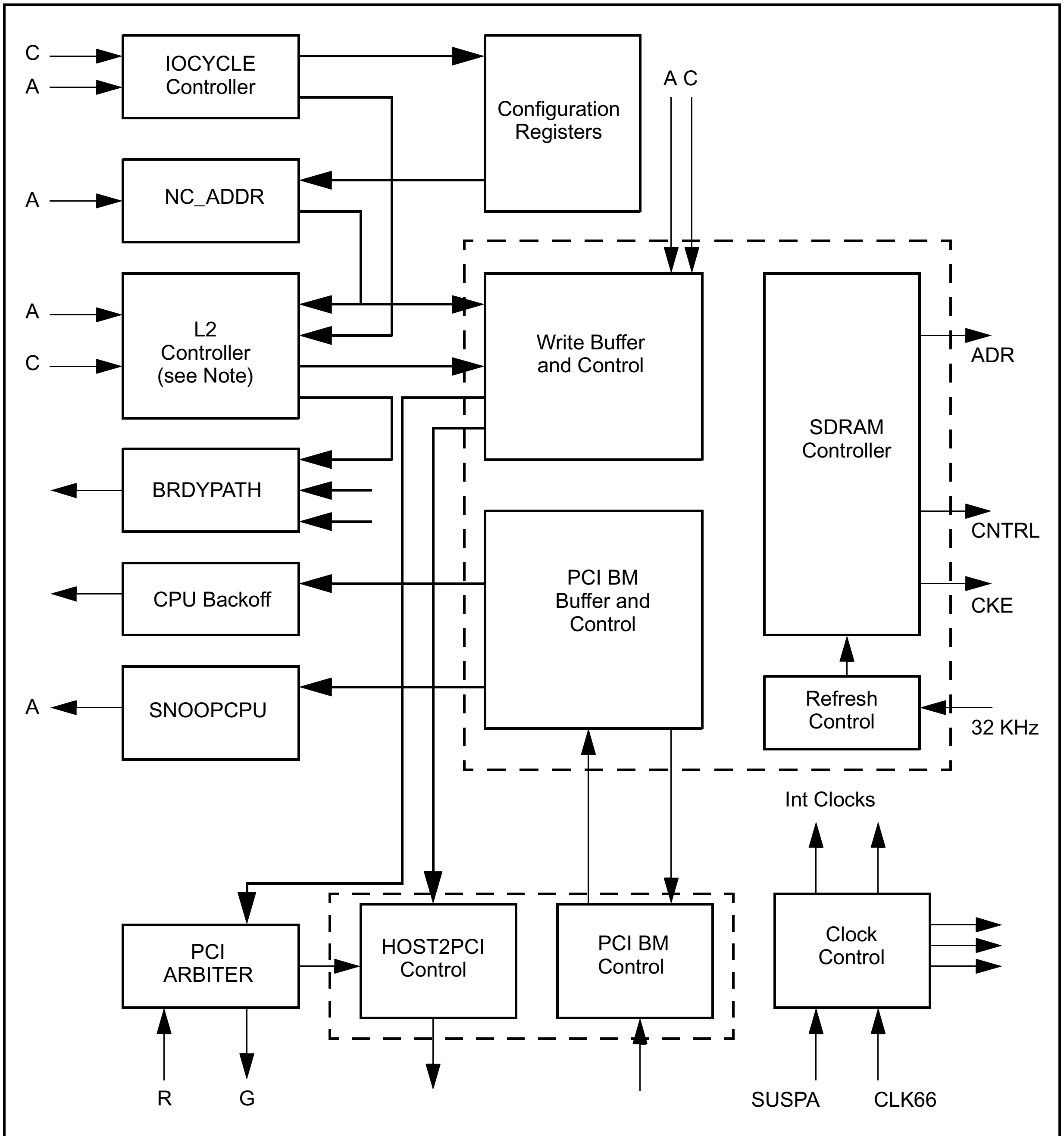


Figure 3-2 Block Diagram

Note: The L2 Controller is not operational in the ZFx86 chip.

3.2. Interface Signals

This section provides the description of signals between the North Bridge Core and other cores or pads. The signal descriptions are divided into various tables according to the functionality and interface they relate to for easy reference.

Table 3.1 SDRAM Interface Signals

Signal	Pin No. (PU/PD)	Type	Description
SDRAM_CSn_N[3:0]	0 = B25 1 = A25 2 = A24 3 = B24	O	Chip Select - drives CSn inputs of four pairs of SDRAM banks 7/6, 5/4, 3/2 and 1/0. Disables or enables device operation by masking or enabling all inputs except CLK, CKE (clock enable) and DQM.
SDRAM_DQM[3:0]	0 = C23 1 = B23 2 = D22 3 = A23	O	Data Input/Output Mask - Drives four DQM inputs of SDRAM bank pair. They correspond to inverse of BEn[3:0] in 32 bits mode, in 16 bits mode DQM[1:0] is used to multiplex the four byte enables. When DQM is high during a write no data is written and during a read data pins are tri-stated.
SDRAM_WE_N	C22	O	Data Write Enable - This output drives write enables to all SDRAMs. Enables write operation and row pre-charge. Latches data in starting from CAS, WE active. (dram_Wenn)
MA[11:0]	0 = A15 1 = C14 2 = B15 3 = C15 4 = B16 5 = A16 6 = C16 7 = B17 8 = C17 9 = A18 10 = C18 11 = B18	O	Memory Address - Twelve bits of address for SDRAM. Row/column addresses are multiplexed on the same pin. And it correspond to Row address : RA11 ~ RA0, Column address : CA9 ~ CA0
MA[13:12]	12 = A19 13 = D18	O	Memory Address - SDRAM chips are organized internally in banks, and there may be 4 banks{x4, x8 and x16 SDRAMs) or 2 banks (x32 SDRAM). These banks should not confuse with system level banks of SDRAMs.

Table 3.1 SDRAM Interface Signals (cont.)

Signal	Pin No. (PU/PD)	Type	Description
D[31:0]	0 = C24 1 = A26 2 = B26 3 = C25 4 = D24 5 = C26 6 = E23 7 = D25 8 = E24 9 = D26 10 = E25 11 = E26 12 = F24 13 = F25 14 = G23 15 = F26 16 = G25 17 = G24 18 = G26 19 = H24 20 = H25 21 = H26 22 = J24 23 = J23 24 = J25 25 = J26 26 = K25 27 = K24 28 = K26 29 = L23 30 = L24 31 = L25	I/O	Memory Data Bus -
SDRAM_RAS_N	C20	O	Row Address Strobe - connects RASn input of SDRAM chips.
SDRAM_CAS_N	D20	O	Column Address Strobe - connects to CASn input of all SDRAM chips.
SDRAMCLK[0:3]	0 = B22 1 = A22 2 = B21 3 = A21	O	SDRAM Clock - The frequency will be x1 of the System Clock (CLKIN). The SDRAM captures inputs at the positive edge of the clock.
SDRAM_CKE	C21	O	SDRAM Clock Enable - Masks clock to freeze operation from the next clock cycle. SDRAM_CKE should be enabled at least one cycle prior to new command. Disables input buffers for power down mode.

Table 3.2 PCI Sideband Signals

Signal	Type	Description
SB_REQnn	I	South Bridge PCI Request - This is a request signal from the South bridge for PCI bus into the arbiter.
SB_GNTnn	O	South Bridge PCI Grant - This is bus grant signal from the arbiter in response to SBREQn.
SB_PCICLK	O	South Bridge PCI Clock - This is a clock signal output to the South Bridge.
IRQ13 Internal Only	O	Floating Point Interrupt - Causes SB to send INT to the CPU.
WRM_RESET	I	Warm Reset - This is an input to the core. This pin allows an external source to initiate a warm reset to the CPU, by setting this pin HIGH. If not used it should be tied LOW.

Table 3.3 Test Signals (JTAG)

Signal	Type	Description
TESTMODE	I	Enable Test Mode.
SCAN_MODE	I	Scan Mode
SCAN_EN	I	Scan Enable
TEST_SI1–TEST_SI4	I	Scan In ports - for four scan chains
TEST_SO1–TEST_SO4	O	Scan Out ports - for four scan chains
CPU_SYNC[3:0]	O	CPU SYNC bus - For syncing Processor test with PCI Masters in simulation and with PCI Bus Testers in hardware. Only CPU_SYNC[0] actually goes out off chip.

3.3. Functional Description

3.3.1. Processor Interface

Processor Interface will support memory, I/O and special bus cycles from the processor to the onboard resource, system memory and PCI bus. It will also handle snoop cycles to the processor on behalf of the PCI bus master.

3.3.1.1. Address Map

Following Memory and I/O address map specifies how address decoding is done to determine if local (within the core) or external resource is being accessed. Access to the PCI configuration address register is always local, while access to configuration data register is special as it could be within the core or external depending on the contents of the configuration address register. This is explained in more detail in the PCI Configuration section.

Table 3.4 Memory Access Map

Memory	Local/ PCI	Access Size	Address Range
Maximum local DRAM	Local[a]	All	0FFF FFFF (256 MB max)
PCI Memory Space	PCI	All	Top-of-Memory => 0FFF FFFF
PCI Memory Space	PCI[b]	All	1000 0000 FDFF FFFF
SMM Space	Local	All	1DFD 0000 — 1DFE FFFF
Upper ROM (Presently this also goes to PCI)	PCI	All	FE00 0000 — FFFF FFFF (32 MB)

a.Some holes may exist in the range of 000A 0000 – 000F FFFF for the Video Memory and ROMs may not be shadowed
b.A hole of 128 KB may be used by SMM, which would be local, if programmed in SMMC

Table 3.5 I/O Address Map

Register	Local/ PCI	Access Size	Address
NB Configuration Index Register	Local	16 bits	24H
NB Configuration Data Register	Local	16 bits	26H
PCI Configuration Cycle Address Register	Local	32 bits	CF8H
PCI Configuration Cycle Data Register (If Configuration Enable =1 in PCI Configuration Address Register)	Local or PCI	All	**CFCH**

3.3.1.2. Special Regions

To allow system flexibility, 4 regions have been defined that can be individually programmed to be non-cacheable. The region sizes should be allowed to be 32KB, 64KB, 128KB, 256KB, 512KB, or 1MB. The starting address and size are programmed in the registers PR3-PR1, and the mode is programmed in the register PRC.

Refer to ‘Register Set’ on page 131 for programming information. In a typical system configuration, these regions are not required since standard non-cacheable regions — non-shadowed ROM regions between 000C0000H-000FFFFFH — are automatically marked as non-cacheable.

3.3.1.3. Invalidate ROM Shadowed Region in L1

If ROM region is cached in L1, then whenever write to the ROM region is detected will cause the NB to run an invalidate cycle back to the processor.

3.3.1.4. ROM Regions

Since the CPU is not capable of handling write-back/write-through modes on cache line basis, all ROM regions shadowed in the memory should be marked as non-cacheable, if the L1 cache is operating in write-back mode. If L1 is operating in write-through mode then ROM may be cacheable, if L1WBEN is cleared in the PROC register and SMM RAM is not overlaid to this ROM region.

The reading from this region of DRAM is controlled by SHADRC and SHADWC register controls the writing. To load the DRAM with ROM information, that region should be marked write-able and disable the reading. This will cause the reads to happen from the ROM and write to the DRAM. After loading is complete the bits may be reversed in SHADRC and SHADWC for that region. Programming of SHADRC and SHADWC is explained in more detail in 3.4. "Register Set" on page 131.

3.3.1.5. Special cycles

The North Bridge will always swallow special cycles, with the exception of shutdown and halt cycles, which it will broadcast to the PCI bus followed by a master-abort cycle.

3.3.1.6. Burst sequence

The North Bridge core will support only linear burst sequence

Table 3.6 North Bridge Core Burst Sequence

Burst Cycle First Address A3A2	Linear assumed burst cycle address sequence A3A2
00	00-01-10-11
01	01-10-11-00
10	10-11-00-01
11	11-00-01-10

3.3.1.7. Concurrent CPU and PCI busses

The North Bridge will support an external master and the CPU running concurrently. This mandates that the PCI bus be (TRDY#) stalled on external master cycles until we run the cycle up to the CPU (if snooping is required). This may result in a write-back cycle, and thus the DRAM must be updated with the CPU data. The CPU should only be snooped for External Master accesses to a new 16 byte line (A[31:4] change). The initiator of the snoop request (Memory Controller) will manage this.

3.3.1.8. PCI Master Deadlock Issues

In the event of a PCI deadlock condition, i.e. where the Target is continually retrying cycles, we need to take the following steps.

- Condition our response with a large time-out counter.
- CPU interface snoop logic must be able to generate BOFF# to the CPU to retry the cycle. Special care must be taken when there are cycles already been posted to the write buffer.
- If the deadlock is on a PCI read, the CPU can be backed off, the external master granted the bus and the corresponding

address snooped to the CPU. In the event of a write-back cycle, the DRAM is updated and the external master continues with the cycle.

- In the event of a deadlock on a PCI write, with the PCI write buffer full, we must block the post to the write buffer, assert BOFF# to the CPU, force the CPU Interface/L2 FASTEN's to idle and let the CPU retry the cycle. If the interrupted cycle was to DRAM, interrupting the cycle in the middle of a burst and rewriting part or all of the data is non-destructive (b/c it is linear memory). This means that in the clock that BOFF# is asserted to the CPU, all writes to the Write Buffer are blocked (in the Write Buffer modules)
- While snooping, AHOLD to the processor will be used, so that a write-back cycle can occur, even if BOFF# was generated.

3.3.1.9. Conditions when a memory address is not cacheable in L1

If any one of the conditions below is true, that memory address must be made non-cacheable in L1.

- KENEN bit in the PROC register is a '0'.
- The address is not within the local DRAM area.
- The address matches the value programmed in any of the four programmable regions and the cacheability bit for the programmable region indicates non-cacheable.
- DIS23RMAP bit in SMMC register is a '0' and access is to 20000-3FFFFh region.
- SMM memory is mapped to lower SMM RAM region.
- When in SMM mode, and SMM memory is mapped to lower region, D0000-EFFFFh. Or KDISSMMRAM bit is set to a '1' in the SMMC register. "SMM Control Register (SMMC) - Configuration Index 118H" on page 135

3.3.1.10. Conditions when a memory address is marked as WT

Since the Processor does not support WB and WT regions in the memory, entire cache can be Write-back (WB) or Write-through (WT). The corresponding bits in the Programmable Region Control Register will be treated as don't cares.

3.3.1.11. Conditions when there is no write-posting to PCI buffer

All IO cycles will cause the write buffers to flush ***and*** finish any outstanding cycles. If any external masters currently occupy the PCI bus the CPU must be stalled until it regains arbitration priority for the PCI bus.

LOCK and Pseudo-LOCK

CPU drives the LOCK# signal when it is executing read-modify-write type of instruction. The LOCK# is driven with the ADS# of the first read bus cycle and stays on till the RDY# for the last write cycle. During the lock cycle the access to SDRAM from PCI is blocked till the LOCK# is de-asserted.

Pseudo Lock indicated by PSLOCK# signal is asserted by the CPU when doing multi dword read. NB uses this signal only during 64-bit write cycle to lock the bus to the SDRAM, just like LOCK#, till the second write is complete. The use of PSLOCK# is controlled by a bit, DIS_PSLOCK, in the PROC register. When this bit is set, PSLOCK# signal will be ignored.

3.3.1.12. Address Translation

When the processor does a memory access (MIO# = high) the address goes through a translation logic before it is fed to the decoders to determine if the address belongs to local memory or it is off-chip

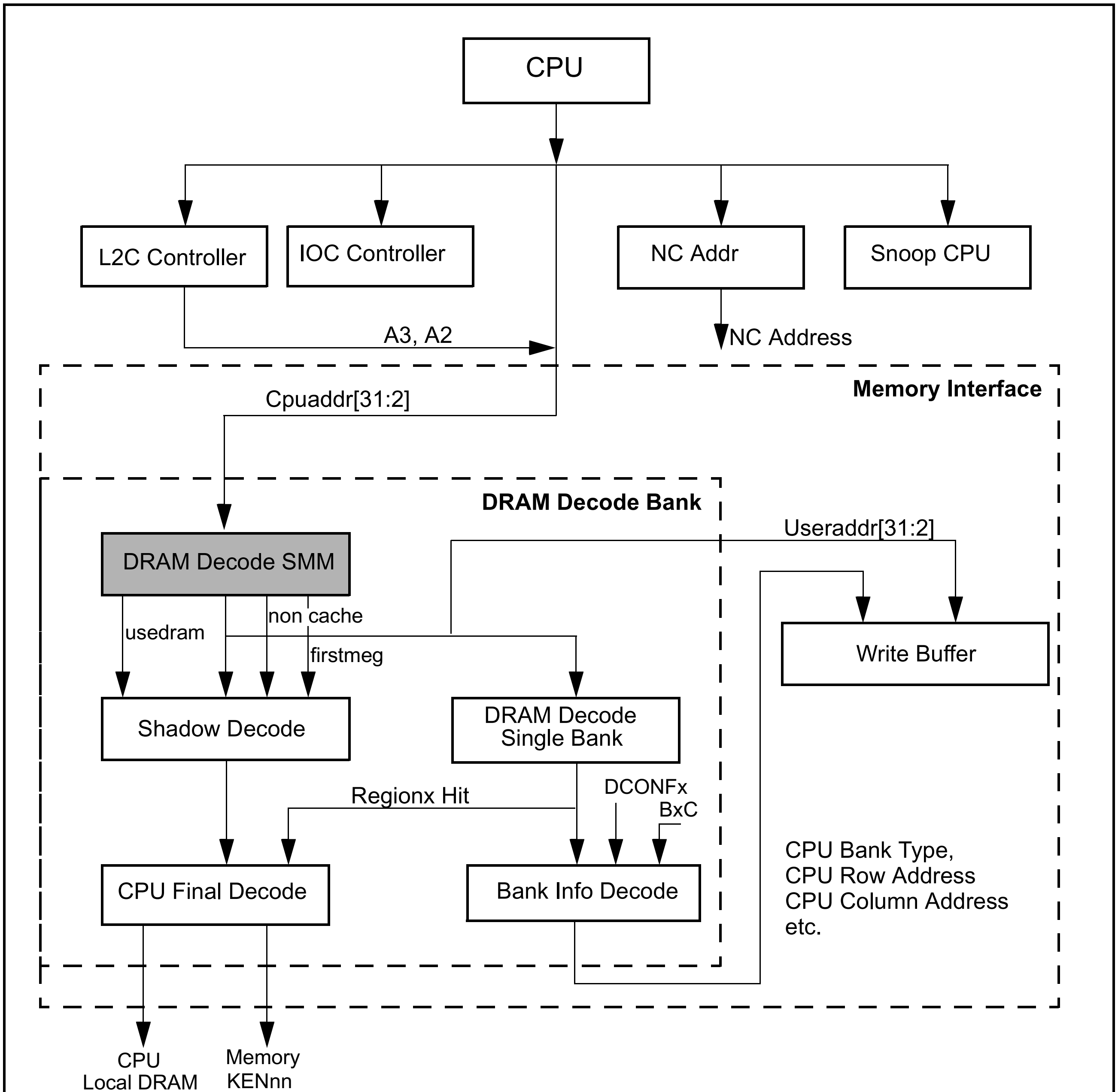


Figure 3-3 CPU Address Translation and Decode

3.3.2. DRAM Controller

3.3.2.1. DRAM architecture

NB DRAM controller will only support 16 Mb, 64 Mb and 128Mb Synchronous DRAM. It will have control for up to four banks of 32-bit memory or 16-bit memory. Each bank can be made up of a single or multiple SDRAM chip(s). Each 32-bit bank can have capacity from 8 MB to 64 MB of memory, with four banks giving a total of 256 MB for the NB. With 16-bit banks lower memory size (2MB) can be achieved for a low entry cost system design. DRAM controller will support Single Data Strobe (SDR) SDRAM of the following configurations:

Table 3.7 SDRAM Configurations

SDRAM Type	SDRAM Configuration	SDRAM Int. Bank Size	SDRAM Row Size	SDRAM Column Size	Number of SDRAM per Memory Bank		Max Memory Bank Size (Mbytes)	
					32Bit	**16Bit**	**32Bit**	**16Bit**
16Mbit	2M x 4 x 2	1	11	10	8	4	16	8
	1M x 8 x 2	1	11	9	4	2	8	4
	512K x 16 x 2	1	11	8	2	1	4	2
64Mbit	4M x 4 x 4	2	12	10	8	4	64	32
	2M x 8 x 4	2	12	9	4	2	32	16
	1M x 16 x 4	2	12	8	2	1	16	8
	512K x 32 x 4	2	11	8	1	1	8	4*
128Mbit	2M x 16 x 4	2	12	9	2	1	32	16

3.3.2.2. DRAM memory map

Address mapping is done to reduce the encoding logic and cover wide range of memory devices. Row address is presented with a RAS and column address is presented with a CAS, bank address need to be stable and same, during both RAS and CAS times. “P” bit during CAS indicates to the SDRAM to enable or disable Auto Pre-Charge. The memory address MA[11:0] connect to MA[11:0], and MA[13:12] connect to the internal-bank address SDRAMs. The exact mapping of address connection between NB and the SDRAMs will depend on the technology and the configuration of the SDRAM.

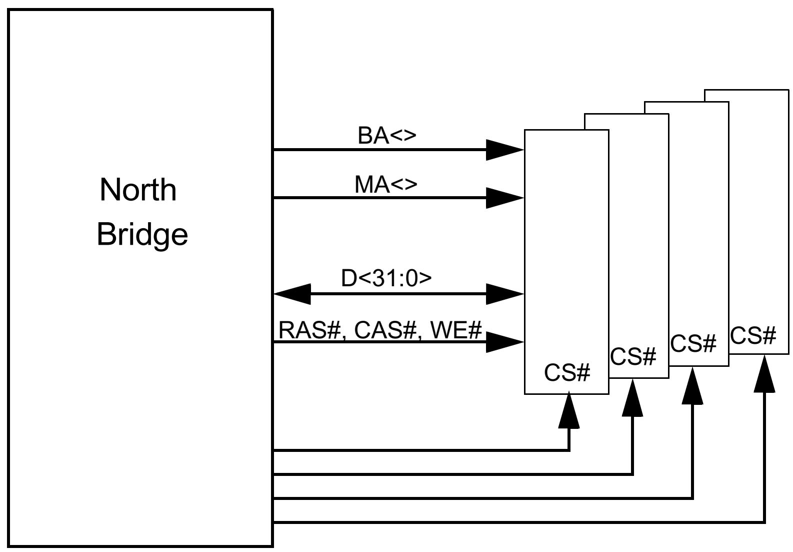


Figure 3-4 32-Bit Banks Connection

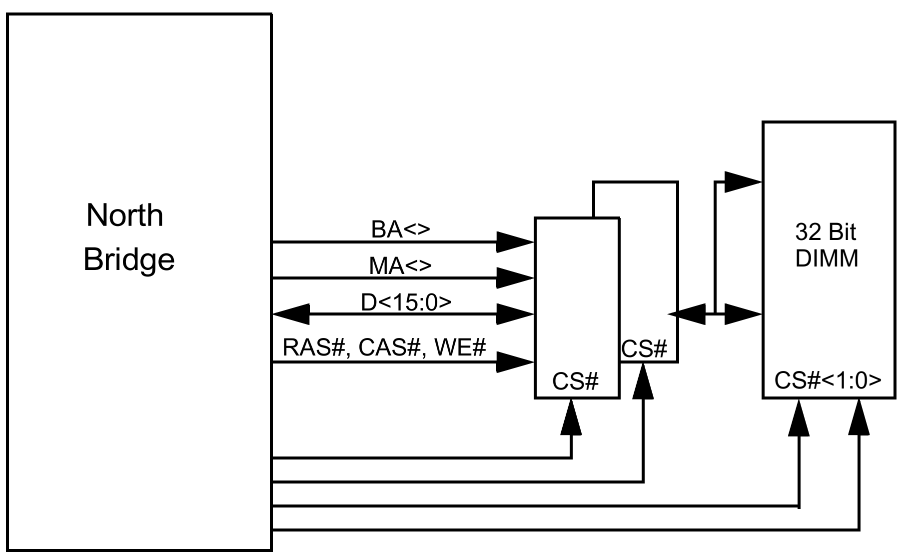


Figure 3-5 16-Bit Bank Connections

3.3.2.3. DRAM Bank location flexibility

All DRAMs, regardless of sizes, must be able to be placed in any Bank and at any starting address locations (in 2 Meg resolution for 16 bit banks). If there are gaps in the mapping, the gaps will go to PCI. There is no checking for the overlapped mapping, and this is done in the BIOS firmware.

3.3.2.4. Mixing SDRAM banks

Mixing of SDRAM banks of different size, speed and/or manufacturers will be allowed through SDRAM bank control and timing control registers for each bank. The 16-bit banks only exist on the lower 16 bits of the MD<> bus. Also a bank can be of 16-bit and another bank of 32-bit in a system will be supported.

3.3.2.5. DRAM refresh

16 Mb, 64 Mb and 128Mb SDRAM need refresh every 64 ms/row regardless of the width. Using Auto Refresh command every 15.625 usec satisfies this requirement. SDRAM refresh period is generated from 32KHz clock, using both edges of the clock to generate refresh request pulse. Though 50% duty cycle clock is not required, the ratio should be reasonable to allow the system clock to synchronize this 32KHz clock.

3.3.2.6. CPU write to 16 bit DRAM bank

If a 16 bit DRAM bank is in the system and the processor is writing to either the upper word or lower word only, there should be just one DRAM write cycle. The DRAM should filter writes based on the BE[3:0]# and ONLY run 2 cycles if both BE2# AND BE3# are active (i.e. crossing a WORD boundary).

3.3.2.7. Treat bank miss as page miss

Because bank miss cycles are very infrequent, bank misses will be treated as page misses. This will save logic by allowing the use of one block of logic (i.e. RAS precharge logic) for all DRAM banks, instead of one for each DRAM bank.

3.3.2.8. Pre-charge Time

Pre-charge command is used instead of Auto Pre-charge to pre-charge the rows of the DRAM. It is initiated at the start of the Mode Register setting, auto-refresh, self-refresh, RAS time-out, page-miss and on power-on.

3.3.2.9. Decode all necessary signals before writing to FIFO

Instead of latching the address into the write FIFO in the NB chip and then decoding the necessary signals when the actual write cycle is about to occur, all necessary signals will be decoded and then written into the FIFO. This will allow faster DRAM signal generation.

3.3.2.10. ROM shadowing

When ROM is shadowed into DRAM, only CPU is allowed access to shadowed RAM. The DRAM controller should treat all non-CPU access to ROM regions as non-local DRAM access.

The ROM shadowing is applicable to regions 000C0000H-000FFFFFH. Reg. 200 and 201 specify READ/WRITE ability of the shadowed ROM. Illustrated below is a table specifying the 16KB block from D4000H-D7FFFH (1MB-160KB).

Table 3.8 ROM Shadow Illustration

SHADRC	SHADWC	Description
0	0	Read from 000D4000H:000D7FFH comes from ROM Write to 000D4000H:000D7FFH is ignored Read from 4GB–4GB-32MB comes from ROM Write to 4GB–4GB-32MB is sent to PCI bus
0	1	Read from 000D4000H:000D7FFH comes from ROM Write to 000D4000H:000D7FFH is to Shadow RAM Read from 4GB–4GB-32MB comes from ROM Write to 4GB–4GB-32MB is sent to PCI bus
1	0	Read from 000D4000H:000D7FFH comes from Shadow RAM Write to 000D4000H:000D7FFH is ignored Read from 4GB–4GB-32MB comes from ROM Write to 4GB–4GB-32MB is sent to PCI bus
1	1	Read from 000D4000H:000D7FFH comes from Shadow RAM Write to 000D4000H:000D7FFH is to Shadow RAM Read from FFFD4000H:FFFD7FFH comes from ROM Write to FFFD4000H:FFFD7FFH is sent to PCI bus

3.3.3. Configuration and Testability

3.3.3.1. North Bridge Register Programming

IO address **24H** will be used as an index register and IO address **26H** will be used as a data register. All 16 bit register accesses to these addresses will be absorbed by North Bridge (NB) and will not appear on the PCI bus. Any byte accesses to IO 24/26 will be ignored by NB and passed onto the PCI bus where the South Bridge will pick up the request.

Table 3.9 North Bridge Registers

Index range	Function
0100H-01FFH	Reset sampling and Miscellaneous Registers
0200H-02FFH	SDRAM Registers
0300H-03FFH	Power Management Registers
0400H-04FFH	L2 Controller Registers[a]

a.The ZFx86 does not support an L2 Cache.

3.3.3.2. PCI Registers

The PCI specification requires that all devices connected to the PCI bus must implement a minimum set of 64bytes of PCI configuration space registers. In order to support this, two double-word IO addresses, **0CF8H and 0CFCH** are used as CONFIG_ADDRESS and CONFIG_DATA registers, respectively. Only double-word access to 0CF8H address will be trapped for local access and it accesses PCI_CONFIG_ADDRESS register, other size accesses will go to PCI bus. To access local PCI configuration registers, address has to be loaded into PCI_CONFIG_ADDRESS bits 7:0 and with bits 23:11 equal to 0s, and bit31 equal to 1. If bits 23:11 is other than 0s or bit31 is not a 1, or access is not a double-word a PCI cycle would be run.

See ‘PCI Hardware and Software Architecture & Design, 4th Edition, Solari & Willse, Annabooks, IBSN 092939259-0.

3.3.4. PCI bus interface and arbiter

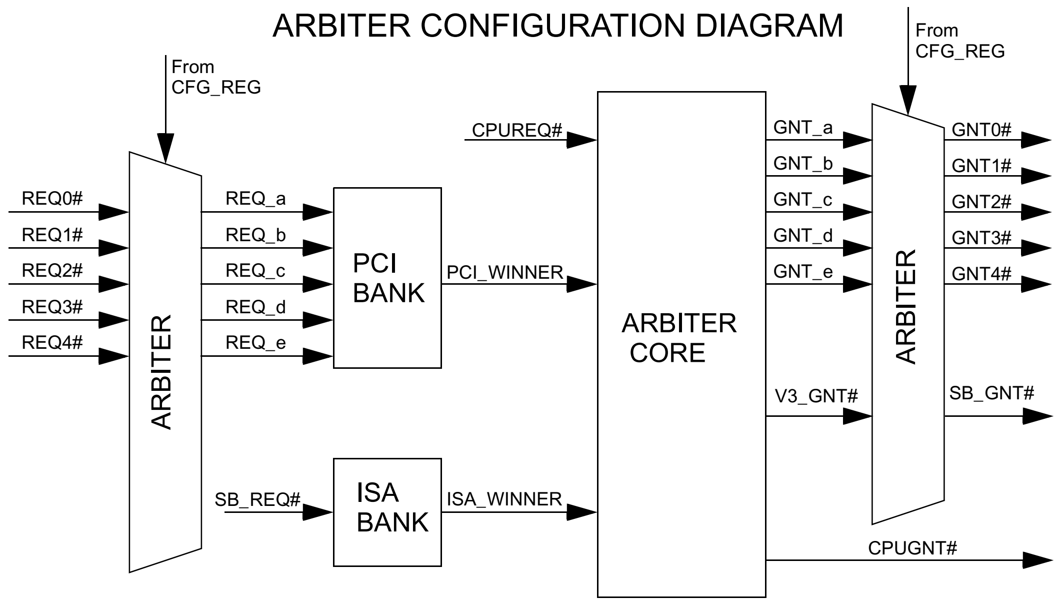
3.3.4.1. PCI arbiter

NB will support up to 3 external PCI masters, a South Bridge request, and the CPU request. The arbiter will allow a rotating priority scheme between the PCI-ISA bridge, the CPU, and one of the PCI REQ/GNT# pairs. It can be configured to give the PCI REQ#/GNT# pairs every other arbitration (see GRNT_BANK diagram below). The arbitration is set up on a 4-3 tier formation. The starting arbitration chooses evenly (round robin), or prioritizes between a “PCI Winner”, the South Bridge and the CPU. The beginning arbitration chooses a “PCI Winner” amongst 3 requesters. These 4 requesters can be mapped to any of the REQ#/GNT# pins.

Figure 3-6 shows the arbitration priority scheme between requesters on the PCI bank. in order to determine a “PCI Winner”. Note Requester A (mapped from any REQ#/GNT# pair) *can* win every other PCI arbitration. This will give priority access to certain masters (ex. video or IDE) if they are continually requesting. Although the Requester A can get every other PCI arbitration, it still needs to arbitrate with an “ISA winner” and the CPU.

If the PCI Winner is allowed every other arbitration (see GNT_BANK Priority Diagram below) and the CPU and ISA are all requesting, the Requester A will get 1/2 of all PCI Arbitration but only get access (to DRAM) 1/4 of the arbitration. If the PCI Winner is programmed for Round Robin (fair arbitration) the Requester A would have a maximum of 1/6 of all arbitration to DRAM.

The PCI Arbiter should allow PCI Peer to Peer access (we decode the address as non-local DRAM) and allow the CPU to access the DRAM while the Peer to Peer transaction is taking place. In the event of a CPU request for the PCI bus, it must again wait until it is arbitrated the bus, and then the cycle can complete. Note that for PCI peer to peer transactions we do NOT need to snoop the CPU because this memory is considered non-cacheable.



SB_REQ#: Can be routed via SB_REQ pin or one of the PCI REQ# pins.

PCI_BANK: Only 5 requests as 6th is routed to V4. Priority scheme as explained.

ISA_BANK: Always ROTATING priority between masters. If SIO_HIPRI bit (Reg111EH[6]) is set, then V4 is always higher priority.

ARBITER CORE: Priority scheme as explained.

Figure 3-6 PCI Bus Arbiter Block Diagram

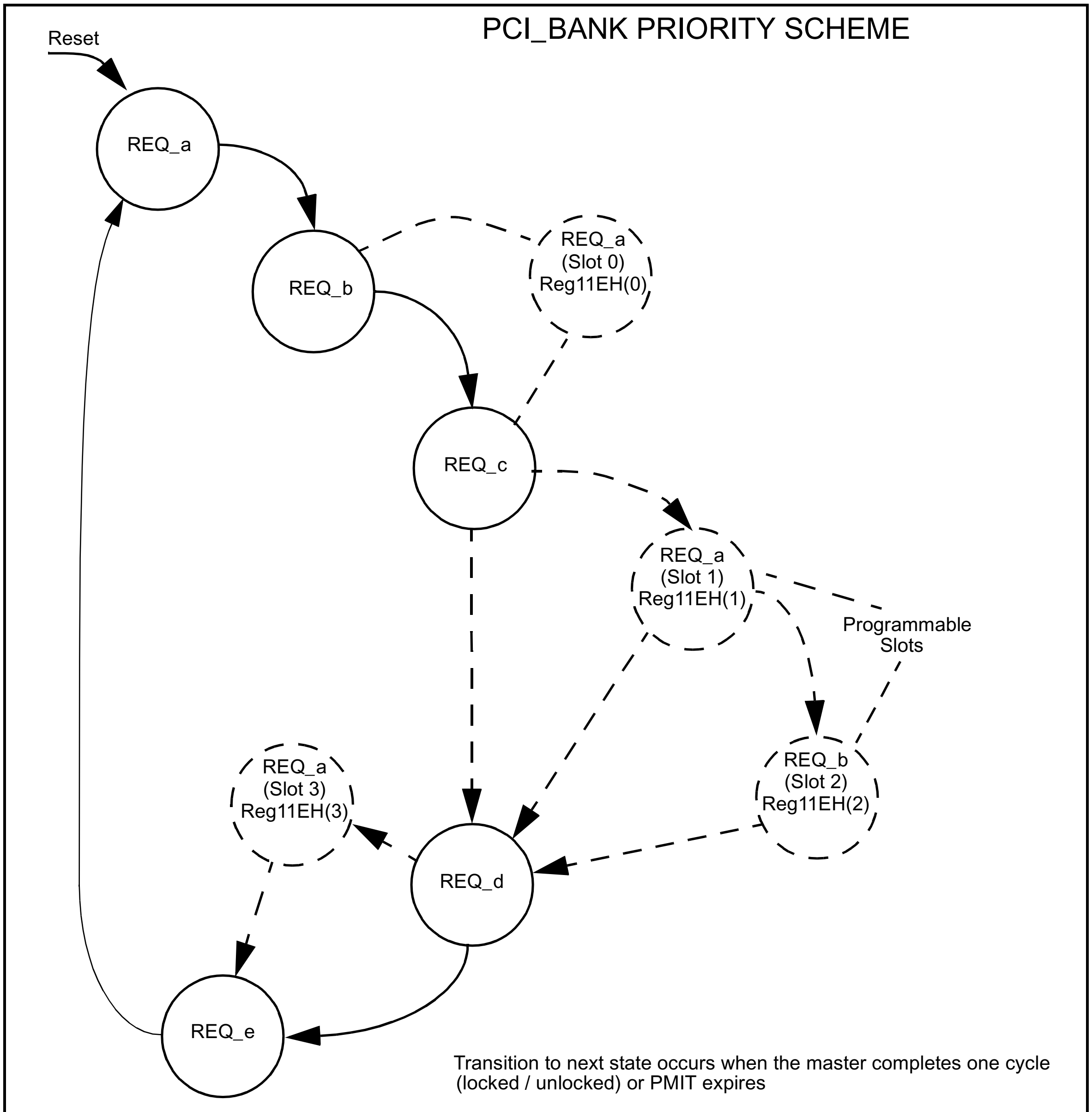


Figure 3-7 PCI Bank Arbiter State Diagram

3.3.5. PCI Write Buffer and Bursts

In order to meet the 100Mb/s PCI transfer rate requirements, the PCI needs to sustain a continual burst of data to linearly increasing addresses. Thus the PCI block must look ahead 3 entries (from the current address and data) in order to determine if it can sustain the burst cycle. The 3 entry requirement is based on the need to de-assert FRAME# on the _next_ to last data transfer. The PCI write buffer will be 16 DWORD entries long. Read Reordering is NOT allowed for PCI cycles.

The conditions for preventing or terminating a burst sequence are:

- The next cycle is to a different (non-contiguous address).
- There are less than 3 entries in the PCI write buffer.
- The PCI burst enable bit is turned off in the PCIC register.
- The cycle is to a ROM region.
- The cycle is to I/O space.

3.3.5.1. CPU write/read to PCI

Since the CPU (32-bit X86) and PCI are of same data widths, CPU memory and I/O cycles are translated into corresponding PCI cycles of the same type, except for writes and reads to configuration address which is explained in the next section.

3.3.5.2. PCI configuration address and data registers

The PCI configuration address and data registers have been defined as **0CF8H** and **0CFCH** respectively. The North Bridge (NB) is implementing the PCI configuration mechanism #1 with 64 bytes of configuration space (00H-3FH). Since some registers within this 64 bytes are reserved, the PCI specification requires any writes to these areas to be ignored and any reads from these areas to be returned as all 0s.

Writes and reads to the Configuration Address port 0CF8H have to be full double-word and the composition of the Configuration Address register is in Figure 3-8 Translation of Type 0 Configuration Cycle, and Figure 3-9 Translation of Type 1 Configuration Cycle. Any access other than full double-word access will turn into an I/O cycle on the PCI bus. When the CPU makes an access to the Configuration Data port 0CFCH, bit 31 of the Configuration Address register decides if its going to be a configuration cycle or a ordinary I/O cycle. Configuration cycles to the PCI register space (Device Address = 0) are trapped to local PCI registers; other accesses go out to the PCI as Configuration cycles.

Configuration cycles can be Type 0 or Type 1, and are differentiated in the way the mapping of device address is done before it is presented on the PCI bus.

In Type 0 cycles the device address, CONFIG_ADDR Register[15:11], is decoded and presented on the AD[31:11] bits, the CONFIG_ADDR Register[10:2] will go as is on AD[10:2] and AD[1:0] are forced to '00'. In Type 1 cycles the contents of the CONF Register[31:2] are copied to AD[31:2] and AD[1:0] are forced to '01'. Both Type0 and Type1 configuration cycles are supported, and the type of cycle run will depend on the value of the "Bus Number" field in the Configuration Address register. If it's zero then Type 0 will be run otherwise Type 1 would be run.

```
; Check to see if the NB is in it's default state.

      mov   eax,   80009048h       ; (EAX) = Function register
      mov   dx,    0CF8h           ; (DX) = PCI register.
      out   dx,    eax             ; Write it on out (full width)
      add   dl,    4               ; DX = 0CFCH
      in    al,    dx              ; Get the current value (I/O cycle)
      sub   dl,    4               ; DX = 0CF8H
      cmp   al,    04h             ; Check to see if it's the
                                   ; default value or we have been
                                   ; programming it.
```

```
; Check to see if NB is in it's default state.

      mov   eax,   80009050h        ; (EAX) = Function register
      mov   dx,    0CF8h            ; (DX) = PCI register.
      out   dx,    eax              ; Write it on out.
      add   dl,    4                ; DX - 0CFCH
      in    eax,   dx               ; Get the current value.
      sub   dl,    4                ; DX = 0CF8H
      shr   eax,   16               ; shift reg52 to AL
      cmp   al,    98h              ; default value? (ROM read-write gets
                                    ; changed at table programming (bit1)
      jz    SkipReset               ; Skip reseting PCI if default


            ; Do reset, if register content was not as
            ; defaults are supposed to be
            ;

DoReset:
      POSTCODE 02h

      mov   eax, 80009044h          ; (EAX) = reset control register, 5540.
      out   dx, eax                 ; Select the reset control register.
      add   dl, 4                   ; Move to PCI index register.
      in    al, dx                  ; Get the reset status.
      or    al, 0Eh                 ; Reset PCI, IDE etc.
      out   dx, al
      nop
      nop
      nop
      and   al, 0F0h                ; That reset was edge sensitive.
      out   dx, al                  ; Clear reset bits
      or    al, 1                   ; Do X-Bus reset (resets entire system)
      out   dx, al                  ; do-it
      sub   dl, 4
      jmp   DoReset                 ; keep spinning until hardware reset


SkipReset:
```

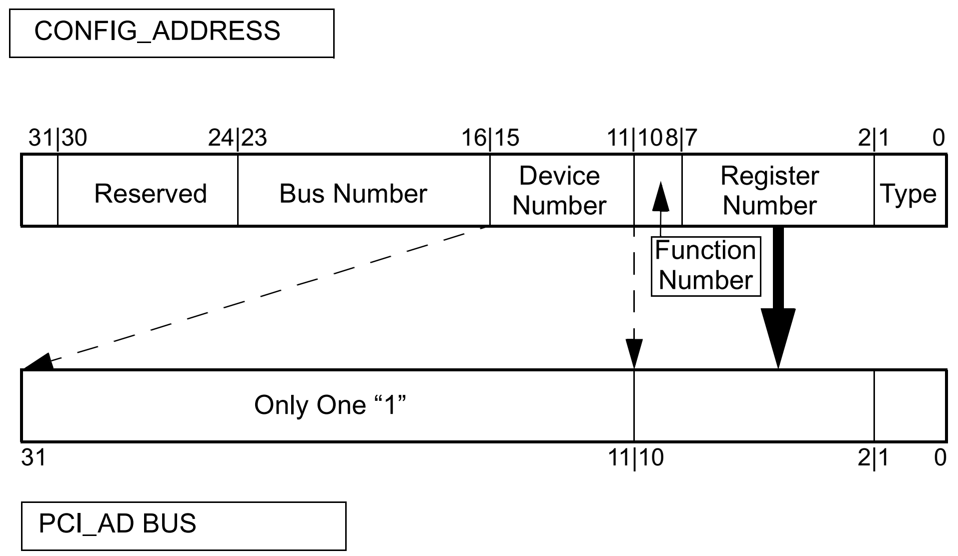


Figure 3-8 Translation of Type 0 Configuration Cycle

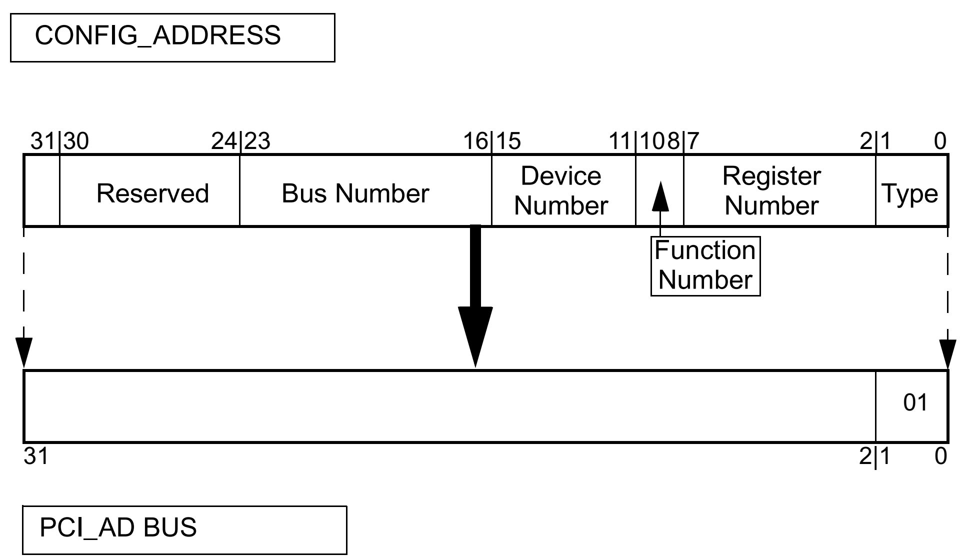


Figure 3-9 Translation of Type 1 Configuration Cycle

3.3.5.3. CPU to PCI bus cycle conversion

The Following table illustrates the conversion of CPU cycles to PCI cycles. In most cases it is one to one except where the configuration space is involved.

Table 3.10 CPU-PCI Cycle Conversion

M/IO#	D/C#	W/R#	CPU bus definition	C/BE[3:0]#	PCI bus definition
0	0	0	Interrupt Acknowledge	0000	Interrupt Acknowledge
0	0	1	Special Cycle	0001	Special Cycle
0	1	0	I/O Read	0010	I/O Read
0	1	1	I/O Write	0011	I/O Write
0	1	0	I/O Read to CFCH	1010	Configuration Read
0	1	1	I/O Write to CFCH	1011	Configuration Write
1	X	0	Memory Read	0110	Memory Read
1	1	1	Memory Write	0111	Memory Write

3.3.5.4. PCI interrupt acknowledge cycle

The CPU generates 2 INTA cycles, for any assertion of INTR. However, the PCI bus specifies that only one INTA cycle is to be run on the PCI bus. NB will swallow the first INTA cycle from the CPU, and pass only the second INTA cycle.

3.3.5.5. AD[31:0] must be actively driven

If the CPU is the master, the NB must actively drive out AD[31:0], C/BE[3:0]#, and PAR.

3.3.5.6. Hardware generated PCI special cycle

Hardware generated special cycles on the PCI bus come from CPU generated special cycles only. In particular, the CPU ‘SHUTDOWN’ and ‘HALT’ special cycles will be propagated to the PCI bus and become PCI special cycles. All other CPU special cycles should not propagate to the PCI bus. For the CPU ‘SHUTDOWN’ cycle, the value 0000H should be driven out on AD[15:0], until the end of the PCI cycle. For CPU the ‘HALT’ cycle, the value 0001H should be driven out on AD[15:0] until the end of the PCI cycle. The normal PCI cycle will have an address phase, followed by the data phase, while a hardware PCI special cycle will have an address phase only, and no data phase.

3.3.5.7. PCI configuration access

For a PCI configuration access, AD[31:0] (in address phase) needs to be driven out one PCI clock before FRAME# is asserted. This will allow enough precharge time for devices that resistively connect their IDSEL signals to the AD bus.

3.3.5.8. Master Abort

When the CPU is doing a I/O or memory read from the PCI bus and a master abort condition happens, NB must drive out all ‘1’s to the CPU bus. This is the prescribed behavior for a CPU bridge according to the PCI Spec. and this event is recorded in the PCI Configuration

Status register.

3.3.6. Write buffer architecture

3.3.6.1. Write FIFO depth

There are two 8 level deep write FIFO's in NB. The independent FIFO's are for buffering DRAM and PCI writes from the CPU.

3.3.6.2. Read re-ordering support for DRAM

DRAM read around write should be allowed as an option for performance purposes.

3.3.6.3. Empty write buffers before disabling

When the write buffer enable bit has been programmed to disable, the internal logic must ensure that the write buffer is emptied before disabling the write buffer. This should be guaranteed by design, since a configuration cycle (I/O space) will flush the buffers before disabling them.

3.3.6.4. All IO writes should not be posted

All IO writes should not be posted. This mean that BRDY# to the processor should not be returned until the IO cycle is actually finished.

3.3.6.5. PCI Reads should wait for Empty PCI buffers

All PCI reads should wait for the PCI write buffer to empty.

3.3.6.6. Concurrent PCI and DRAM operation

Due to the presence of 2 split write buffer's, allowing PCI memory writes and DRAM memory writes/reads to occur in parallel enhances NB performance. For example, if the CPU posts 6 PCI memory writes, these should all be taken from the CPU at 0 or 1 Wait State (dependent on L2). If the CPU follows with 5 DRAM writes and then a read, the read should be serviced while the PCI memory writes are in process and while the DRAM writes are being stored (read-reordering).

3.3.7. System Management Mode

System Management Mode (SMM) provides system designers with a means of adding new software controlled features to their computer products that always operate transparently to the operating system (OS) and software applications. SMM is intended for use only by system firmware, not by applications software or general purpose system software.

System Management Mode is entered when the processor detects an SMI# (generated by South Bridge). SMM is used for special power management software or for transparent emulation of I/O devices. Special SMM memory space is allocated to protect this software from getting corrupted by the application or OS.

The processor after a RESET needs to be put in the SL-mode by writing to CCR3 internal CPU register. In SL-mode the processor generates SMIACT# after it gets an SMI# signal, and it remains asserted until the RSMI instruction is executed by the SMM software. SMIACT# is generated on the pin marked SMADS#, as in the default mode of the processor (ST-mode) special System Mode Address Strobe# (SMADS#) is generated to access SMM space. When SMIACT# is asserted NB logic enables the access to the special SMM memory during this time, which is usually mapped over some normal memory addresses explained in the following sections.

3.3.7.1. SMM base address

The SMM base address in the NB system is defined as either 000D 0000H-000E FFFFH or 1DFD 0000H-1DFE FFFFH region, selected via SMMC register in NB. The region selected and its size should be programmed into the CPU SMAR register. The size of the SMM memory can be a multiple of 32Kbytes, to maximum of 128KBytes

All physical memory space used for SMM memory is at A0000H-BFFFFH in DRAM. The reason for physically locating SMM memory here is that normally a Video memory space exists here, which is never shadowed in the local DRAM memory.

The address re-mapping logic for DRAM also re-maps 20000H-3FFFFH to A0000H-BFFFFH (physical memory) during the very first SMM mode access after reset, when SMMC[2]=0 to allow copying the SMM code.

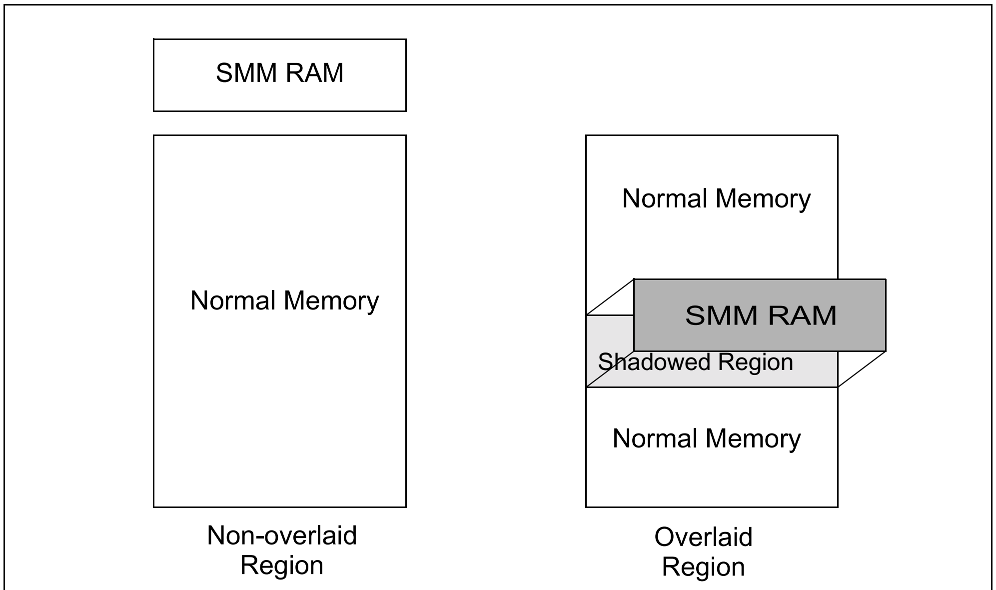


Figure 3-10 SMM RAM Location

3.3.7.2. L1 Cacheability of SMM RAM

The NB does not allow the SMM RAM to be cached in the L1 cache if SMM RAM is mapped to the lower SMM region by keeping KEN# HIGH during SMM space access. If the SMM RAM is mapped to the upper region i.e. 1DFD0000-1DFEFFFFh then SMM may be cached if SMMC[1]=0 and the processor was operating in the write-through mode. If the L1 was operating in the write-back mode then BIOS should set SMMC[1] to a ‘1’ to disable the caching to avoid dirty data remaining in the L1 cache. "SMM Control Register (SMMC) - Configuration Index 118H" on page 135

Reasons why the SMM cannot and should not be cached in L1:

- If the CPU was operating in write-back mode, the SMM RAM cannot be cached in L1 as the CPU lacks a WB/WT# pin to put SMM data in write-through mode. The SMM data needs to be marked WT, as after the SMI handler is finished the dirty data in the L1 cache cannot be written back to the memory as that SMM space will not be available to the NB in the normal mode to write back the data. This is a CPU limitation.
- If the CPU was operating in the write-through mode and SMM was mapped in the lower region which may contain shadowed ROM, the L1 needs to be flushed on the **entry** of the SMI handler by asserting FLUSH# pin of the CPU. Flushing the SMM code at the end of SMI handler by

asserting the FLUSH# pin again immediately after de-assertion of SMI#. Presently the NB does not have this function to assert FLUSH# on the start or end of SMI handler.

The only condition under which SMM memory can be cached in L1:

- If the CPU was in Write-through mode and the SMM RAM is mapped to the upper region, then it is not overlaid on a cacheable memory. Here no cache flushing is needed at the start or at the end of SMI handler.

3.3.7.3. SMM code copying to SMM RAM in non-SMM mode

To copy the SMI handler in the SMM memory space, LDSMIHLDER bit in the SMMC register should be set to '1' and DIS23RMAP bit (SMMC[2]) should be set to a '0'. See "SMM Control Register (SMMC) - Configuration Index 118H" on page 135. Then write to address 20000H to 3FFFFH with the SMM code, which actually writes to the physical A0000H to BFFFFH area in the local DRAM. Which means that BIOS or the loader software should not be using the 20000H-3FFFFH space for code or data storage during this time. After the loading is complete BIOS should set bit SMMC[2] to a '1', which disables the re-mapping of 20000H–3FFFFH to A0000H-BFFFFH address for normal operation, and clear LDSMHLDER bit. Also the SMIHLDERLOCK bit should be set in SMMC after the loading to prevent access to this area in the normal mode.

3.3.7.4. Ban non-CPU masters from accessing SMM RAM

When the PCI master accesses the memory space reserved for SMM memory, NB will not re-map the address and it will access the normal memory at that location.

3.3.8. Power Management

3.3.8.1. CPU clock management

The South Bridge Core or Chip will do all the power management. The NB will receive SUSP# and SUPA# signals from the South Bridge. On detection of SUSPA# assertion, if the EN_STOP_CPU_CLK bit is a '1' in the Clock Control2 (CC2) register, the clock will be stopped to the CPU within 2-4 clocks. On detection of SUSP# de-assertion the clock to the CPU will restart within 2-4 clocks.

3.3.8.2. PCI clock management

PCI clock to the PCI bus is free running.

3.3.8.3. Power Management for SDRAMs

The CKE (clock enable) pin for SDRAM puts the device into a low power state. If EN_SDRAM_CKE_RST bit is a '1' in the Clock Control2 register, CKE will be driven low on detection of P_SUSPA# assertion and will be driven back HIGH on detection of SB_SUSP# pin de-assertion.

Another bit EN_STOP_SDRAM_CLK in the CC2 when set to '1' will stop the clocks going to SDRAM for even lower power consumption under the conditions described above.

3.4. Register Set

Registers are divided into six groups:

- Reset Sampling
- Memory Configurations Registers
- Power Management Registers
- L2 Cache Controller Registers
- PCI Configuration Registers
- Miscellaneous Registers

3.4.1. Register Address Map

The address of these registers will be maintained to be same as in 87550 chip. The register composition may change as certain bits may be eliminated or meaning changed and some new ones added.

Note: Some registers discussed here may not be consecutively numbered; registers reserved for future expansion are not shown in this chapter. Ports 24h and 26h are used as index and data register respectively. Note that 16-bit access to Port 24h and 26h will be directed to the North Bridge and not passed to the PCI bus. However, any byte access to Ports 24h and 26h will be ignored by the North Bridge and passed to the PCI bus.

This table shows the index numbers and abbreviations, detailed register descriptions follows.

Table 3.11 Configuration Registers

Bit	Name	Function	Def.
NB RevisionID Register (RID): Configuration Index **100H**			
3:0	North Bridge ID	North Bridge version ID.	0H
15:4	**Reserved**		All 0s
Programmable Region 1 Register (PR1): Configuration Index **110H**			
2:0	PREG1S<2:0>	Programmable region 1 block size: 000 = 32KB 001 = 64KB 010 = 128KB 011 = 256KB 100 = 512KB 101 = 1 MB 11X = Reserved	0H
15:3	PREG1S<27:15>	Programmable region 1 starting address: The programmable region starting address must be a multiple of the block size.	000H
Programmable Region 2 Register (PR2): Configuration Index **111H**			
2:0	PREG2S<2:0>	Programmable region 2 block size: 000 = 32KB 001 = 64KB 010 = 128KB 011 = 256KB 100 = 512KB 101 = 1 MB 11X = Reserved	0H
15:3	PREG2S<27:15>	Programmable region 2 starting address: The programmable region starting address must be a multiple of the block size.	000H

Table 3.11 Configuration Registers (cont.)

Bit	Name	Function	Def.
Programmable Region 3 Register (PR3): Configuration Index **112H**			
2:0	PREG3S<2:0>	Programmable region 3 block size: 000 = 32KB 001 = 64KB 010 = 128KB 011 = 256KB 100 = 512KB 101 = 1 MB 11X = Reserved	0H
15:3	PREG3S<27:15>	Programmable region 3 starting address: The programmable region starting address must be a multiple of the block size.	000H
Programmable Region 4 Register (PR4): Configuration Index **113H**			
2:0	PREG4S<2:0>	Programmable region 4 block size: 000 = 32KB 001 = 64KB 010 = 128KB 011 = 256KB 100 = 512KB 101 = 1 MB 11X = Reserved	0H
15:3	PREG4S<27:15>	Programmable region 4 starting address: The programmable region starting address must be a multiple of the block size.	000H
Programmable Region Control Register (PRC): Configuration Index **114H**			
1:0	PRGREG1_SEL<1:0>	Programmable region 1 select <1:0>: 00 = Disable 01 = Reserved 10 = non-cacheable 11 = Reserved	00
3:2	PRGREG2_SEL<1:0>	Programmable region 2 select <1:0>: 00 = Disable 01 = Reserved 10 = non-cacheable 11 = Reserved	00
5:4	PRGREG3_SEL<1:0>	Programmable region 3 select <1:0>: 00 = Disable 01 = Reserved 10 = non-cacheable 11 = Reserved	00

Table 3.11 Configuration Registers (cont.)

Bit	Name	Function	Def.
7:6	PRGREG4_SEL<1:0>	Programmable region 4 select <1:0>: 00 = Disable 01 = Reserved 10 = non-cacheable 11 = Reserved	00
15:8	**Reserved**		All 0s
Cacheability Override Register (COR): Configuration Index **115H**			
0	CACHE_OVR_A24	Cacheability Override A24: When set, all address with A<24> high is marked non-cacheable. This corresponds to addresses in the range X1000000H-X1FFFFFFH.	0
1	CACHE_OVR_A25	Cacheability Override A25: When set, all address with A<25> high is marked non-cacheable. This corresponds to addresses in the range X2000000H-X3FFFFFFH.	0
2	CACHE_OVR_A26	Cacheability Override A26: When set, all address with A<26> high is marked non-cacheable. This corresponds to addresses in the range X4000000H-X7FFFFFFH.	0
3	CACHE_OVR_A27	Cacheability Override A27: When set, all address with A<27> high is marked non-cacheable. This corresponds to addresses in the range X8000000H-XFFFFFFFH.	0
4	CACHE_OVR_A28	Cacheability Override A28: When set, all address with A<28> high is marked non-cacheable. This corresponds to addresses in the range 10000000H-1FFFFFFFH.	0
5	CACHE_OVR_A29	Cacheability Override A29: When set, all address with A<29> high is marked non-cacheable. This corresponds to addresses in the range 20000000H-3FFFFFFFH.	0
6	CACHE_OVR_A30	Cacheability Override A30: When set, all address with A<30> high is marked non-cacheable. This corresponds to addresses in the range 40000000H-7FFFFFFFH.	0
7	CACHE_OVR_A31	Cacheability Override A31: When set, all address with A<31> high is marked non-cacheable. This corresponds to addresses in the range 80000000H-FFFFFFFFH.	0
15:8	**Reserved**		All 0s

Table 3.11 Configuration Registers (cont.)

Bit	Name	Function	Def.
Back-off Control Register (BCR): Configuration Index **117H**			
1:0	NONPOST_RETRY_CNT<1:0>	Non-posted PCI cycle retry count <1:0> 00 = 3 01 = 7 10 = 11 11 = 15	0
2	NONPOST_RETRY_DIS	Disable PCI retry counter for non-posted cycle: 0 = enable, 1 = disable	
3	**Reserved**		
5:4	POST_RETRY_CNCT<1:0>	Posted PCI cycle retry count<1:0> 00 = 3 01 = 7 10 = 11 11 = 15	
6	POST_RETRYCNT_DIS	Disable PCI retry counter for posted cycle: 0 = enable, 1 = disable	
7	**Reserved**		0
8	RESET_CNT_ON_GNT	RESET retry counter on any bus master grant: 0 = not reset on gnt, 1 = reset on gnt.	
9	HLD_RETRY_ON_REQ	Hold retry on any PCI Bus Master Request: 0 = initiate retry once before backoff, 1 = initiate retry only after all pending PCI bus master requests have been serviced	
15:10	**Reserved**		0

Table 3.12 SMM Control Register (SMMC)

Bit	Name	Function	Def.
SMM Control Register (SMMC) - Configuration Index **118H**			
0	**Reserved**		0
1	KDISSMMRAM	SMM RAM KEN disable: 1= KEN# held inactive (high) during access to SMM RAM, 0 = KEN# function normally within SMM RAM. Should always be set to ‘1’, to disallow caching.	0
2	DIS23RMAP	Disable 20000H-3FFFFH remap to A0000H-BFFFFH physical memory in SMM mode: 0 = enabled, 1 = disabled. **Note**: This bit can only be used while both L1 and L2 are disabled.	0

Table 3.12 SMM Control Register (SMMC) (cont.)

Bit	Name	Function	Def.
3	FRCREMAP	Enables the SMM remapped address to be used in a non-SMM cycle. This is used during loading of the SMM code to the memory. It works in conjunction with bit 14 and 15 of this register, and they need to be in the correct state to allow the loading.	0
5:4	SMM_DL_SEL[1:0]	SMM D0000H-D7FFFH select<1:0>: 00 XXXD0000H-XXXD7FFFH is not used as SMM space. 01 reserved 10 000D0000H-000D7FFFH is used as SMM space. remap to 000A0000H-000A7FFFH in physical DRAM space.) 11 1DFD0000H-1DFD7FFFH is used as SMM space. (remap to 000A0000H-000A7FFFH in physical DRAM space.) ***Note**: When programmed to 10, 000D0000H-000D7FFFH will be automatically be set to non-cacheable.	00
7:6	SMM_DH_SEL[1:0]	SMM D8000H-DFFFFH select<1:0>: 00 XXXD8000H-XXXDFFFFH is not used as SMM space. 01 reserved 10 000D8000H-000DFFFFH is used as SMM space. (remap to 000A8000H-000AFFFFH in physical DRAM space.) 11 1DFD8000H-1DFDFFFFH is used as SMM space. (remap to 000A8000H-000AFFFFH in physical DRAM space.) ***Note**: When programmed to 10, 000D8000H-000DFFFFH will be automatically bet set to non-cacheable.	00
9:8	SMM_EL_SEL[1:0]	SMM E0000H-E7FFFH select<1:0>: 00 XXXE0000H-XXXE7FFFH is not used as SMM space. 01 reserved 10 000E0000H-000E7FFFH is used as SMM space. (remap to 000B0000H-000B7FFFH in physical DRAM space.) 11 1DFE0000H-1DFE7FFFH is used as SMM space. (remap to 000B0000H-000B7FFFH in physical DRAM space.) *Note: When programmed to 10, 000E0000H-000E7FFFH will be automatically bet set to non-cacheable.	00
11:10	SMM_EH_SEL[1:0]	SMM E8000H-EFFFFH select<1:0>: 00 XXXE8000H-XXXEFFFFH is not used as SMM space. 01 reserved 10 000E8000H-000EFFFFH is used as SMM space. (remap to 000B8000H-000BFFFFH in physical DRAM space.) 11 1DFE8000H-1DFEFFFFH is used as SMM space. (remap to 000B8000H-000BFFFFH in physical DRAM space.) ***Note**: When programmed to 10, 000E8000H-000EFFFFH will be automatically bet set to non-cacheable.	00

Table 3.12 SMM Control Register (SMMC) (cont.)

Bit	Name	Function	Def.
12	SWAP_23_MAP	Swap SMM 2/3 mapping: 0 = 2/3 will be mapped to A/B, 1 = 2/3 will be mapped to B/A. Here 2/3 and A/B refer to the address bits 19-16, 0 = 2XXXX access will be mapped to AXXXX and 3XXXX to BXXXX 1 = 2XXXX access will be mapped to BXXXX and 3XXXX to AXXXX	0
13	SWAP_DE_MAP	Swap SMM D/E mapping: 0 = D/E will be mapped to A/B, 1 = D/E will be mapped to B/A. Here again D/E and A/B refer to the address bits 19-16, 0 = DXXXX access will be mapped to AXXXX and EXXXX to BXXXX 1 = DXXXX access will be mapped to BXXXX and EXXXX to AXXXX	0
14	LDSMIHLDER	Load SMI handler into SMM RAM: 1 = enable access to SMM RAM during normal cycle, 0 = disable access to SMM RAM during normal cycle.	0
15	SMIHLDERLOCK	SMM RAM access in normal mode lock: This bit provides an option to lock bit 14 in a disabled state, thereby prohibiting any further access to SMM RAM from normal mode. This bit can only be written once. Reading a 0 from this bit indicates that bit 14 above is not locked. Reading a 1 from this bit indicates that bit 14 above is locked to disable state.	0

Table 3.13 Processor Control Register (PROC)

Bit	Name	Function	Def.
Processor Control Register (PROC) - Configuration Index **119H**			
0	KENEN	KEN enable: When low, KEN# will be forced to inactive state for all cycles. When high, KEN# will be generated for all local memory cycles.	0
1	L1WBEN	L1 write-back enable: When low, WB_WT# will be in write through state always. When high, WB_WT# will be in write back state whenever is possible. NO WB_WT# pin on CPU	1
2	**Reserved**		0
3	**Reserved**	Set to 1 in hardware.	1
4	**Reserved**		0

Table 3.13 Processor Control Register (PROC)

Bit	Name	Function	Def.
5	WRFIFO_EN	Enable write FIFO: 0 = disable, 1 = enable. This bit controls buffer depth of CPU-PCI write buffer and CPU-SDRAM write buffer. When disabled, CPU-PCI depth = 2 CPU-SDRAM depth = 1 When enabled, CPU-PCI depth is controlled by PCIWFIFOC register CPU-SDRAM depth is controlled by WFIFOC registers	0
6	DIS_PSLOCK	Disable PSLOCK – When set to ‘1’, will disable the PSLOCK signal from being used.	0
7	FLUSH	Setting this bit from 0->1, causes the core to set FLUSHnn pin to the CPU to go LOW for 1 clock. To do another flush this bit should be reset to ‘0’ and then set to 1.	0
8	DIS_FPUCLR_BY_F0	Disable clearing of FPU error by writing to IO port F0H: 0 = enable clearing, 1 = disable clearing.	0
9	DIS_FPUCLR_BY_F1	Disable clearing of FPU error by writing to IO port F1H: 0 = enable clearing, 1 = disable clearing.	0
10	WRM_RST	Warm Reset – When a ‘1’ is written to this bit, a warm reset sequence initiates. It works the same as FLUSH bit. For example, to do another warm reset, clear this bit to ‘0’ and then set it to 1.	0
11	A20M	Address 20 Mask – Used for DOS compatibility.	0
15:12	**Reserved**		All 0’s

Table 3.14 Write FIFO Control Register (WFIFOC)

Bit	Name	Function	Def.
Write FIFO Control Register (WFIFOC) - Configuration Index **11AH**			
2:0	FIFOD<2:0>	Write FIFO depth 000 8 dwords 001 7 dwords 010 6 dwords 011 5 dwords 100 4 dwords 101 3 dwords 110 2 dwords 111 1 dword	000
3	DRMRDREODEREN	DRAM read re-ordering enable: 0 = disable, 1 = enable. When this bit is set and when there’s pending DRAM write cycle, a DRAM read operation will be performed before a DRAM write operation.	0
4	**Reserved**		

Table 3.14 Write FIFO Control Register (WFIFOC) (cont.)

Bit	Name	Function	Def.
5	CPU&EM_DRAM ARBITRATION	CPU/External Master DRAM Arbitration Priority Scheme: 0 CPU has NO Write Buffer access while Ext. Master is accessing DRAM 1 CPU has Write Buffer access' while Ext. Master is accessing DRAM	0
7:6	**Reserved**		00
11:8	RD2WR_LAT<3:0>	Read to write pending latency<3:0>: These bits indicate the number of clocks to delay before switching from a read cycle back to pending cycles in the write buffer. These bits have no effect if the read re-ordering is disabled. Bits<3:0> Number of CPUCLKs 0H reserved 1H 1 2H 2 3H 3 4H 4 5H 5 6H 6 7H 7 8H 8 9H 9 AH 10 BH 11 CH 12 DH 13 EH 14 FH 15	2H
13:12	**Reserved**		00
15:14	WR_LATENCY<1:0>	DRAM write latency<1:0>: These bits indicate the number of processor clocks write are stalled before being issued to DRAM controller. Bits<1:0> number of clocks 00 1 01 2 10 3 11 4	00

Table 3.15 PCI Control Register (PCIC)

Bit	Name	Function	Def.
PCI Control Register (PCIC) - Configuration Index **11BH**			
0	CPU2PCI_BURST_EN	CPU to PCI burst enable: When 0, the North Bridge only does a single PCI transfer when CPU is accessing PCI bus. When 1, the North Bridge will try to burst to PCI when CPU is master.	0
1	PCIM2DRM_BRST_EN	PCI master to DRAM burst enable: When 0, the North Bridge only does a single DRAM transfer when PCI master is accessing DRAM. When 1, the North Bridge tries to do a burst to DRAM when PCI master is accessing.	0
2	BM_BURSTRD_ALWYS	PCI master read prefetch always: When 0, only PCI read line or PCI read multiple starts a burst read request. For PCI single read, a burst read request initiates only after the first data phase completes and PCI master indicates that it wants a burst access. When 1, any PCI read cycle initiates a burst read request. Note: In order to enable this feature, bit[1] must be enabled.	0
3	DISC_ON_LN_BOUNDARY	Disconnect from PCI master on CACHE line boundary: 0 = no disconnect, 1 = disconnect.	0
4	EN_PCI_FAST_DECDE	Enable PCI fast decode when accessing DRAM: 0 = disable, 1 = enable.	0
5	EN_ADCBE_FLT_IDLE	Enable AD/CBE/PAR float when PCI is idle and CPU is the bus master: 0 = disable float, 1 = enable float.	0
6	DIS_RESOURCE_LOCK	Disable Resource Lock: 0 = enable, 1 = disable. Note: when EN_BUS_LOCK(bit 7) is set to 1, this bit ignored.	0
7	EN_BUS_LOCK	Enable Bus Lock: 0 = disable, 1 = enable. When enabled, GNT# to a particular PCI master remains asserted until LOCK# is deasserted. Note: When this bit is set to 1, DIS_RESOURCE_LOCK(BIT 6) ignored.	0
8	LCK_RDBURST_EN	Enable the locking of PCI bus during a 64-bit processor read access to the PCI bus. 0 = disable, 1 = enable.	0
9	CNFCY_AD_STEP_DIS	PCI configuration cycle address stepping disable: 0 = enable, 1 = disable.	0
10	BM_DONE_DIS	Disable the waiting of PCI master cycle is done before starting processor initated PCI cycle. 0 = enable, 1 = disable. When enabled, the North Bridge's PCI master controller will not start until, 1)PCI master initiated cycle is done, 2)PCI master write buffer is empty, and 3) PCI master read prefetch is done.	0
11	**Reserved**		0
12	**Reserved**		0

Table 3.15 PCI Control Register (PCIC) (cont.)

Bit	Name	Function	Def.
15:12	**Reserved**		All 0's

Table 3.16 Clock Skew Adjust Register (CSA)

Bit	Name	Function	Def.
Clock Skew Adjust Register (CSA) - Configuration Index **11CH**			
2:0	**Reserved**		000
5:3	SDRAMCLK_SKEW	There three bits control the skew between the core clock and the SDRAM clock. 000 – Nominal 001 – Minus 1 nsec 010 – Minus 2 nsec 100 – Plus 1 nsec 101 – Plus 2 nsec Reset – Default to Nominal	000
15:6	**Reserved**		All 0's

Table 3.17 BUS MASTER And Snooping Control Register (SNOOPCTRL)

Bit	Name	Function	Def.
BUS MASTER And Snooping Control Register (SNOOPCTRL) - Configuration Index **11DH**			
0	DIS_SNOOP	Disable Snooping: 0 = enable snoop, 1 = disable snoop.	0
1	DIS_CHK_HITM	Disable the check of HITM#: 0 = enable the checking of HITM# during snooping. 1 = disable the checking of HITM# during snooping. In either case, the L1 cache may be invalidated with INVAL signal.	0
3:2	CK_HITM_WS<1:0>	Check HITM# wait state: 00 2 clock after EADS# is deasserted. 01 3 clocks after EADS# is deasserted. 10 4 clocks after EADS# is deasserted. 11 5 clocks after EADS# is deasserted.	00
4	ADP_PREF_DIS	Adaptive Prefetch Disable: 0 = enable, 1 = disable. When enabled, the North Bridge will monitor the average burst transfer length of a master access and than control the number of speculative prefetches accordingly.	0
5	**Reserved**		0

Table 3.17 BUS MASTER And Snooping Control Register (SNOOPCTRL) (cont.)

Bit	Name	Function	Def.
6	DIS_WB_MERGE	0 = Merge CPU/L2 Write-back data with External Master writes. The External Master's valid bytes overwrite the data cast-out from the CPU/L2 and subsequently limit the bandwidth requirements to the s/dram. 1 = Do not merge External Master write data bytes with CPU/L2 write-back cycle.	0
7	DIS_EM_PREFETCH	0 = Prefetch next "cache" line on EM accesses, and store in prefetch buffer 1 = Disable prefetch logic for External Masters (CPU clock based)	0
8	DIS_CONCURRENCY	CPU/PCI master concurrency disable: 0 = enable, 1 = disable.	0
9	FAST_TRDY	0 = Normal TRDY# timings 1 = Enable Fast TRDY# timings to EM. Improves path from prefetch data ready (from CPU write-back, yes we snarf-see bit 10, or from DRAM)	0
10	DIS_BUS_SNARF	0 = Snarf CPU write-back data and return it to the requesting External Master (read), concurrent with it's retirement into DRAM. 1 = Disable bus snarfing and create 2nd cycle to get data after the write-back has retired it to DRAM.	0
11	FORCE_DRM_PM_PCIM	Force DRAM page miss in bus master cycle: 0 = Disable force page miss mode 1 = Enable force page miss mode.	0
12	**Reserved**		0
13	DISPCIM_ELY_DRM_CY	Speculatively start DRAM cycle for PCI External Master Request and restart it in the event of an L1 or L2 write-back: 0 = enable, 1 = disable. **Errata**: LNB will corrupt system memory with 2 PCI masters. This bug can be eliminated by setting bit 13 -- DISPCIM_ELY_DRM_CY, in the SNOOPCTRL register.	0
14	**Reserved**		0
15	DISPCIM_SHADOWRAM	0 = Claim cycle for PCI Master access to 000C0000-000F0000 region: 1 = Do not Claim cycle for PCI Master access to ROM space (shadowed RAM) **Note**: All DRAM Write/Read protect bits are still applicable	0

Table 3.18 Arbiter Control Register (ARBCTRL)

Bit	Name	Function	Def.
Arbiter Control Register (ARBCTRL) - Configuration Index **11EH** (see also PCI register section REG 41H)			
0	REQa_slot0	0= Disable Slot 0 for REQa 1= Enable Slot 0 for REQa	0
1	REQa_slot1	0= Disable Slot 1 for REQa 1= Enable Slot 1 for REQa	0
2	REQb_slot2	0= Disable Slot 2 for REQb 1= Enable Slot 2 for REQb	0
3	REQa_slot3	0= Disable Slot 3 for REQa 1= Enable Slot 3 for REQa	0
4	REQpci_slot4	0= Disable Slot 1 for 2nd Arbitration of PCI (see diagram) 1= Enable Slot 1 for 2nd Arbitration of PCI (see diagram)	0
5	PC98_support	0= V4REQ#/V4GNT# pair treated as such. 1= The V4REQ#/V4GNT# pair is treated as PHOLD#/PHLDA#	0
6	SIO_HIPRI	0= Fair Arbitration between V3 & V4 REQ# pins 1= Always give priority to V3 REQ#	0
7	**Reserved**		0
[15:8]	CPU_BUSY_TIMER	Number of PCI bus clocks that the CPU can “own” of the PCI bus before it is preempted by any other active requesters 00H = Never preempt the CPU 01H = 4 clks 02H = 8 clks... FFH = 1024 clks	00H

Table 3.19 PCI Write FIFO Control Register (PCIWFIFOC)

Bit	Name	Function	Def.
PCI Write FIFO Control Register (PCIWFIFOC) - Configuration Index **120H**			
2:0	FIFOD[2:0]	PCI Write FIFO depth Bits FIFO depth 000 16 dwords 001 14 dwords 010 12 dwords 011 10 dwords 100 8 dwords 101 6 dwords 110 4 dwords 111 2 dwords	000
3	**Reserved**		0
4	**Reserved**		0

Table 3.19 PCI Write FIFO Control Register (PCIWFIFOC)

Bit	Name	Function	Def.
5	PCI BM_FREERUNMODE	PCI master write buffer PCI entry count free running mode bit. Transfer loop which copies CPU clocked write buffer entry count to PCI clocked entry count normally operates in an on-demand mode. This forces a free running mode which update the PCI every 6 or 8 CPU clocks (see slow transfer bit below).	0
6	PCI_BM_SLOWRUNMODE	PCI master write buffer PCI entry count slow transfer mode. Increases transfer loop period from 6 CPU clocks to 8 CPU clocks. Transfer loop period is defined as how often the PCI side entry count is updated from the CPU entry count. 0= 6 CPU clocks 1 = 8 CPU clocks	0
8:7	STALL_PCI_BM_POST	Stall PCI master posting: Bits<1:0> # of clocks 00 no stall 01 1 clock 10 3 clocks 11 7 clocks	00
11:9	**Reserved**		All 0's
15:12			0

3.4.2. DRAM registers

Table 3.20 Shadow RAM Read Enable Control Register (SHADRC)

Bit	Name	Function	Def.
Shadow RAM Read Enable Control Register (SHADRC) - Configuration Index **200H**			
0	LMEMRDEN0	Local memory C0000H-C3FFFH read enable: 0 = disable, 1 = enable.	0
1	LMEMRDEN1	Local memory C4000H-C7FFFH read enable: 0 = disable, 1 = enable.	0
2	LMEMRDEN2	Local memory C8000H-CBFFFH read enable: 0 = disable, 1 = enable.	0
3	LMEMRDEN3	Local memory CC000H-CFFFFH read enable: 0 = disable, 1 = enable.	0
4	LMEMRDEN4	Local memory D0000H-D3FFFH read enable: 0 = disable, 1 = enable.	0
5	LMEMRDEN5	Local memory D4000H-D7FFFH read enable: 0 = disable, 1 = enable.	0
6	LMEMRDEN6	Local memory D8000H-DBFFFH read enable: 0 = disable, 1 = enable.	0
7	LMEMRDEN7	Local memory DC000H-DFFFFH read enable: 0 = disable, 1 = enable.	0

Table 3.20 Shadow RAM Read Enable Control Register (SHADRC) (cont.)

Bit	Name	Function	Def.
8	LMEMRDEN8	Local memory E0000H-E3FFFH read enable: 0 = disable, 1 = enable.	0
9	LMEMRDEN9	Local memory E4000H-E7FFFH read enable: 0 = disable, 1 = enable.	0
10	LMEMRDEN10	Local memory E8000H-EBFFFH read enable: 0 = disable, 1 = enable.	0
11	LMEMRDEN11	Local memory EC000H-EFFFFH read enable: 0 = disable, 1 = enable.	0
12	LMEMRDEN12	Local memory F0000H-F3FFFH read enable: 0 = disable, 1 = enable.	0
13	LMEMRDEN13	Local memory F4000H-F7FFFH read enable: 0 = disable, 1 = enable.	0
14	LMEMRDEN14	Local memory F8000H-FBFFFH read enable: 0 = disable, 1 = enable.	0
15	LMEMRDEN15	Local memory FC000H-FFFFFH read enable: 0 = disable, 1 = enable.	0

Table 3.21 Shadow RAM Write Enable Control Register

Bit	Name	Function	Def.
Shadow RAM Write Enable Control Register (SHADWC) - Configuration Index **201H**			
0	LMEMWREN0	Local memory C0000H-C3FFFH write enable: 0 = disable, 1 = enable.	0
1	LMEMWREN1	Local memory C4000H-C7FFFH write enable: 0 = disable, 1 = enable.	0
2	LMEMWREN2	Local memory C8000H-CBFFFH write enable: 0 = disable, 1 = enable.	0
3	LMEMWREN3	Local memory CC000H-CFFFFH write enable: 0 = disable, 1 = enable.	0
4	LMEMWREN4	Local memory D0000H-D3FFFH write enable: 0 = disable, 1 = enable.	0
5	LMEMWREN5	Local memory D4000H-D7FFFH write enable: 0 = disable, 1 = enable.	0
6	LMEMWREN6	Local memory D8000H-DBFFFH write enable: 0 = disable, 1 = enable.	0
7	LMEMWREN7	Local memory DC000H-DFFFFH write enable: 0 = disable, 1 = enable.	0
8	LMEMWREN8	Local memory E0000H-E3FFFH write enable: 0 = disable, 1 = enable.	0
9	LMEMWREN9	Local memory E4000H-E7FFFH write enable: 0 = disable, 1 = enable.	0

Table 3.21 Shadow RAM Write Enable Control Register

Bit	Name	Function	Def.
10	LMEMWREN10	Local memory E8000H-EBFFFH write enable: 0 = disable, 1 = enable.	0
11	LMEMWREN11	Local memory EC000H-EFFFFH write enable: 0 = disable, 1 = enable.	0
12	LMEMWREN12	Local memory F0000H-F3FFFH write enable: 0 = disable, 1 = enable.	0
13	LMEMWREN13	Local memory F4000H-F7FFFH write enable: 0 = disable, 1 = enable.	0
14	LMEMWREN14	Local memory F8000H-FBFFFH write enable: 0 = disable, 1 = enable.	0
15	LMEMWREN15	Local memory FC000H-FFFFFH write enable: 0 = disable, 1 = enable.	0

Table 3.22 Bank 0 Control Register (N_B0C)

Bit	Name	Function	Def.
Bank 0 Control Register (N_B0C) - Configuration Index **202H**			
7:0	B0A<27:20>	Bank 0 starting address <27:20>	00H
8	**Reserved**		0
11:9	B0S<2:0>	Bank 0 DRAM size Bits<2:0> DRAM bank size 000 2MB 001 4MB 010 8MB 011 16MB 100 32MB 101 64MB 110 Reserved 111 Reserved	000
14:12	COLADR0<2:0>	Number of column address bits for Bank 0<2:0> Bits<2:0>Column address 000 8 bits 001 9 bits 010 10 bits all othersReserved	000
15	**Reserved**		0

Table 3.23 Bank 0 Timing Control Register (N_B0TC)

Bit	Name	Function	Def.
Bank 0 Timing Control Register (N_B0TC) - Synchronous DRAM - Configuration Index **204H**			
1:0	B0_TRP	SDRAM Pre-charge cmd to ACT cmd Bits<1:0> Time 00 Reserved 01 2T 10 3T 11 4T	11
3:2	B0_TRC	SDRAM ACT cmd to ACT cmd (same bank) Bits<3:2> Addr. hold time 00 6T 01 7T 10 8T 11 9T	11
6:4	**Reserved**		111
7	B0_CAS_LATCY	SDRAM CAS Latency: 0 = 2T, 1 = 3T	1
8	B0_TRCD	SDRAM ACT cmd to R/W cmd delay: 0 = 2T, 1 = 3T	1
9	B0_TCCD	SDRAM R/W cmd to R/W cmd: 0 = 1T, 1 = 2T	1
15:10	**Reserved**		All ‘1’s

Table 3.24 Bank 1 Control Register (N_B1C)

Bit	Name	Function	Def.
Bank 1 Control Register (N_B1C) - Configuration Index **205H**			
7:0	B1A<27:20>	Bank 1 starting address <27:20>	00H
8	**Reserved**		0
11:9	B1S<2:0>	Bank 21DRAM size Bits<2:0> DRAM bank size 000 2MB 001 4MB 010 8MB 011 16MB 100 32MB 101 64MB 110 Reserved 111 Reserved	000

Table 3.24 Bank 1 Control Register (N_B1C) (cont.)

Bit	Name	Function	Def.
14:12	COLADR1<2:0>	Number of column address bits for Bank 2 Bits<2:0>Column address 000 8 bits 001 9 bits 010 10 bits all othersReserved	000
15	**Reserved**		0

Table 3.25 Bank 1 Timing Control Register (N_B1TC)

Bit	Name	Function	Def.
Bank 1 Timing Control Register (N_B1TC) - Synchronous DRAM - Configuration Index **207H**			
1:0	B1_TRP	SDRAM Pre-charge cmd to ACT cmd Bits<1:0> Time 00 Reserved 01 2T 10 3T 11 4T	11
3:2	B1_TRC	SDRAM ACT cmd to ACT cmd (same bank) Bits<3:2>Addr. hold time 00 6T 01 7T 10 8T 11 9T	11
6:4	**Reserved**		
7	B1_CAS_LATCY	SDRAM CAS Latency: 0 = 2T, 1 = 3T	1
8	B1_TRCD	SDRAM ACT cmd to R/W cmd delay: 0 = 2T, 1 = 3T	1
9	B1_TCCD	SDRAM R/W cmd to R/W cmd: 0 = 1T, 1 = 2T	1
15:10	**Reserved**		All ‘1’s

Table 3.26 Bank 2 Control Register (N_B2C)

Bit	Name	Function	Def.
Bank 2 Control Register (N_B2C) - Configuration Index **208H**			
7:0	B2A<27:20>	Bank 2 starting address <27:20>	00H
8	**Reserved**		0

Table 3.26 Bank 2 Control Register (N_B2C)

Bit	Name	Function	Def.
11:9	B2S<2:0>	Bank 2 DRAM size Bits<2:0>DRAM bank size 000 2MB 010 8MB 011 16MB 100 32MB 101 64MB 110 Reserved 111 Reserved	000
14:12	COLADR2<2:0>	Number of column address bits for Bank 4 Bits<2:0>Column address 000 8 bits 001 9 bits 010 10 bits all others Reserved	000
15	**Reserved**		0

Table 3.27 Bank 2 Timing Control Register (N_B2TC)

Bit	Name	Function	Def.
Bank 2 Timing Control Register (N_B2TC) - Synchronous DRAM - Configuration Index **20AH**			
1:0	B2_TRP	SDRAM Pre-charge cmd to ACT cmd Bits <1:0> Time 00 Reserved 01 2T 10 3T 11 4T	11
3:2	B2_TRC	SDRAM ACT cmd to ACT cmd (same bank) Bits <3:2> Addr. hold time 00 6T 01 7T 10 8T 11 9T	11
6:4	**Reserved**		
7	B2_CAS_LATCY	SDRAM CAS Latency: 0 = 2T, 1 = 3T	1
8	B2_TRCD	SDRAM ACT cmd to R/W cmd delay: 0 = 2T, 1 = 3T	1
9	B2_TCCD	SDRAM R/W cmd to R/W cmd: 0 = 1T, 1 = 2T	1
15:10	**Reserved**		All ‘1’s

Table 3.28 Bank 3 Control Register (N_B3C)

Bit	Name	Function	Def.
Bank 3 Control Register (N_B3C) - Configuration Index **20BH**			
7:0	B3A<27:20>	Bank 3 starting address <27:20>	00H
8	**Reserved**		0
11:9	B3S<2:0>	Bank 3 DRAM size Bits <2:0> DRAM bank size 000 2MB 001 4MB 010 8MB 011 16MB 100 32MB 101 64MB 110 Reserved 111 Reserved	000
14:12	COLADR3<2:0>	Number of column address bits for Bank 6 Bits<2:0> Column address 000 8 bits 001 9 bits 010 10 bits 011 11 bits 100 12 bits all others Reserved	000
15	**Reserved**		0

Table 3.29 Bank 3 Timing Control Register (N_B3TC)

Bit	Name	Function	Def.
Bank 3 Timing Control Register (N_B3TC) - Synchronous DRAM - Configuration Index **20DH**			
1:0	B3_TRP	SDRAM Pre-charge cmd to ACT cmd Bits<1:0> Time 00 Reserved 01 2T 10 3T 11 4T	11
3:2	B3_TRC	SDRAM ACT cmd to ACT cmd (same bank) Bits<3:2> Addr. hold time 00 6T 01 7T 10 8T 11 9T	11
6:4	**Reserved**		
7	B3_CAS_LATCY	SDRAM CAS Latency: 0 = 2T, 1 = 3T	1

Table 3.29 Bank 3 Timing Control Register (N_B3TC) (cont.)

Bit	Name	Function	Def.
8	B3_TRCD	SDRAM ACT cmd to R/W cmd delay: 0 = 2T, 1 = 3T	1
9	B3_TCCD	SDRAM R/W cmd to R/W cmd: 0 = 1T, 1 = 2T	1
15:10	**Reserved**		All ‘1’s

Table 3.30 DRAM Configuration Register 1 (DCONF1)

Bit	Name	Function	Def.
DRAM Configuration Register 1 (DCONF1) - Configuration Index **20EH**			
2:0	**Reserved**		000
5:3	DRAM_INAT_TO	DRAM inactive time-out<2:0> Bits<2:0> Page size 000 never 001 8T 010 32T 011 128T 100 512T 101 reserved 110 reserved 111 immediate If SDRAM interface is inactive for the set amount of time, a Pre-charge cycle is generated at the end of timeout. Pre-charge cycle de-activates the DRAM row which may be in “ACTIVE” state. Doing a Pre-charge cycle when SDRAM is in-active for a while saves power. But the next memory cycle may be to the row which was just closed, and takes a hit of running a RAS cycle causing lower performance.	000
7:6	**Reserved**		00
8	**Reserved**	Fixed to ‘0’ in the hardware	0
9	**Reserved**		
10	**Reserved**		0
11	SDRAM_CMD_PIPELINE	SDRAM command pipeline enable: 0 = disable the pipelining of SDRAM command cycle. 1 = enable the pipelining of SDRAM command cycle.	0

Table 3.30 DRAM Configuration Register 1 (DCONF1) (cont.)

Bit	Name	Function	Def.
12	EN_RELAX_SDRM_CMD_TMING	Enable relax timing for SDRAM command cycle. 0 = disable, 1 = enable relax timing to the SDRAM command cycle. =1 MA, RAS, CAS and WE are asserted 1 clk before CS is asserted. Note: setting this bit to 1 will not affect performance but at the same time, allow the potential of not buffering MA, SDRAM_RAS, SDRAM_CAS, and WE# externally.	0
13	EN_SDRM_PWRDN	Enable SDRAM power-down mode during mix DRAM type configuration: 0 = disable, 1 = enable SDRAM to get into power-down mode during mix DRAM type configuration and when access is to anywhere other than SDRAM. FW should always set this bit to ‘0’	0
14	FST_SDRM_RD_L2_EN	Enable fast SDRAM read access when L2 is on: 0 = disable, 1 = enable. FW should always set this bit to ‘0’	0
15:14	**Reserved**		00

Table 3.31 DRAM Configuration Register 2 (DCONF2)

Bit	Name	Function	Def.
DRAM Configuration Register 2 (DCONF2) - Configuration Index **20FH**			
0	BANK0_16EN	Bank 0 enable: 0 = disable, 1 = enable. When enabled, bank 0 operates as a 16bit bank.	1
1	BANK1_16EN	Bank 1 enable: 0 = disable, 1 = enable. When enabled, bank 1 operates as a 16bit bank.	0
2	BANK2_16EN	Bank 2 enable: 0 = disable, 1 = enable. When enabled, bank 2 operates as a 16 bit bank.	0
3	BANK3_16EN	Bank 3 enable: 0 = disable, 1 = enable. When enabled, bank 3 operates as a 32 bit bank.	0
7:4	**Reserved**		0000
8	BANK0_32EN	=0 Bank 0 disabled (bit0 overides this) =1 Bank 1 enabled as 32 bit bank (this bit overides bit 0)	0
9	BANK1_32EN	=0 Bank 0 disabled (bit1 overides this) =1 Bank 1 enabled as 32 bit bank (this bit overides bit 1)	0
10	BANK2_32EN	=0 Bank 0 disabled (bit2 overides this) =1 Bank 1 enabled as 32 bit bank (this bit overides bit 2)	0
11	BANK3_32EN	=0 Bank 0 disabled (bit3 overides this) =1 Bank 1 enabled as 32 bit bank (this bit overides bit 3)	0
15:12	**Reserved**		0H

Table 3.32 DRAM Refresh Control Register (DRFSHC)

Bit	Name	Function	Def.
DRAM Refresh Control Register (DRFSHC) - Configuration Index **211H**			
4:0	**Reserved**		00000
7:5	REFRPRD<2:0>	Refresh period: These bits determine the refresh period for local DRAM. Bits<2:0> Refresh period 000 15us 001 15us 010 15us 011 30us all others stopped	101
10:8	**Reserved**		000
11	MANUAL_REFRESH	Manual refresh control: A 1-> 0->1 generates a refresh cycle after 128 process clocks. Also, this bit forces normal refresh disabled while left at the 1 setting.	0
13:12	**Reserved**		11
15:14	**Reserved**		00

Table 3.33 SDRAM Mode Program Register (SDRAMMPR)

Bit	Name	Function	Def.
SDRAM Mode Program Register (SDRAMMPR) - Configuration Index **213H**			
0	EN_SDRAM_CONFIG	Enable SDRAM MRS configuration cycle: 0 = disable, 1 = enable.	0
2:1	SDRAM_BANK_CONFIG[1:0]	SDRAM bank configuration select <1:0> programming options as follows: Bits<1:0> DRAM bank 00 Bank 0 01 Bank 1 10 Bank 2 11 Bank 3	00
4:3	POWERON_SEQ[1:0]	SDRAM Power-on initialization sequence bits Bits<1:0> Function 00 Normal 01 Pre-charge SDRAM bank specified by BANK_CONFIG[1:0] 10 Trigger Mode Program Register Command 11 Trigger CBR refresh cycle	00

Table 3.33 SDRAM Mode Program Register (SDRAMMPR)

Bit	Name	Function	Def.
15:5	WCBR_MA[11:1]=	MA[0] comes from the SDRAMMPEX register as it is needed to handle 16 bit banks to do 8 burst cycles [2:0] set to '010' corresponding to burst length of 4 for 32 bit banks set to '011' corresponding to burst length of 8 for 16 bit banks [3] always set to 0 linear burst type (Fixed in hardware) [6:4] 010 CAS Latency 2 011 CAS Latency 3 Others Reserved [11:7] Always leave at '00000' **Note**: MA[13:12] will be forced to 0 always during SDRAM configuration cycle	**019**H

Table 3.34 SDRAM Mode Program Register (SDRAMMPREX)

Bit	Name	Function	Def.
SDRAM Mode Program Register (SDRAMMPREX) - Configuration Index **214H**			
0	WCBR_MA[0]	SDRAM Mode Register bit 0 used together with bits 11:1 defined earlier.	0
15:1	**Reserved**		0000

Table 3.35 SDRAM Slew Control Register (SDRAMSLEW)

Bit	Name	Function	Def.
SDRAM Slew Control Register (SDRAMSLEW) - Configuration Index **239H**			
2:0	MD_DQM_SLEW	32 Bit Data and 4 Bit Mask Bus: MD[31:0], DQM[3:0] 000 = Force Tri-State 001 = 2*N-ch + 4*P-ch 010 = 3*N-ch + 6*P-ch 011 = 5*N-ch + 10*P-ch 100 = 4*N-ch + 8*P-ch 101 = 6*N-ch + 12*P-ch 110 = 7*N-ch + 14*P-ch 111 = 8*N-ch + 16*P-ch	111
5:3	MA_SLEW	14 Bit Address Bus: BA[1:0], MA[11:0] Encoding same as for 2:0 bits	111
8:6	RAS_CAS_SLEW	RASnn and CASnn Encoding same as for 2:0 bits	111

Table 3.35 SDRAM Slew Control Register (SDRAMSLEW) (cont.)

Bit	Name	Function	Def.
11:9	WE_SLEW	Write Enable: Wenn Encoding same as for 2:0 bits	111
14:12	CS_SLEW	4 Chip Select: CSnn[3:0] Encoding same as for 2:0 bits	111
15	DATA_BUS_HOLD	Data Bus Holder enabled when LOW	1

3.4.3. Power Management registers

Table 3.36 Clock Control Register (CC)

Bit	Name	Function	Def.
Clock Control Register (CC) - Configuration Index **300H**			
15:0	**Reserved**		All 0's

Table 3.37 Clock Control2 Register (CC2)

Bit	Name	Function	Def.
Clock Control 2 Register (CC2) - Configuration Index **3FFH**			
0	EN_STOP_CPU_CLK	Enables stopping of CPU clock during Suspend mode. Stopping the clock conserves more power than placing the CPU in suspend mode. This bit allows the BIOS decide.	0
1	EN_SDRAM_CKE_RST	Enables resetting of SDRAM CKE (clock enable) input during suspend mode.	0
2	EN_STOP_SDRAM_CLK	Enables stopping of SDRAM CLK during Suspend mode. Different from SDRAMCLK disable bit in CSA, which always disables the clock. Only use this bit during suspend mode.	0
3	EN_STOP_CORE_CLK	Enable stopping the core clock during suspend mode. When set to 1, it stops the clocks to most of the cores except the clocks needed to detect the end of suspend mode.	0
15:4	**Reserved**		All "0"s

3.4.4. Test Signals

Table 3.38 CPU-SYNC Register (CPUSYNC)

Bit	Name	Function	Def.
CPU-SYNC Register (CPUSYNC) - Configuration Index **238H**			
0	SYNC_OUT	This bit when set to 1, generates a pulse on the SYNC_OUT port of the NB of 2 processor clock wide. This bit is not sticky i.e. writing a 1 will be cleared to zero within two clocks.	0H
15:1	**Reserved**		00H

3.4.5. PCI configuration registers

Note that any read to PCI registers between 00H-FFH which are not described below must return all 0s.

Table 3.39 Vendor ID Register (VID)

Bit	Name	Function	Def.
Vendor ID Register (VID) - Configuration Index **00H**			
15:0	VENDOR_ID	Vendor ID number. These bits are hard-wired.	1066H

Table 3.40 Device ID Register (DID)

Bit	Name	Function	Def.
Device ID Register (DID) - Configuration Index **02H**			
31:16	DEVICE_ID	Device ID number. These bits are hard-wired.	0005H

Table 3.41 Command Register (COMMD)

Bit	Name	Function	Def.
Command Register (COMMD) - Configuration Index **04H**			
0	**Reserved**		0
1	MEM_RESPOND	Memory space enable: 0 = PCI master access to main memory disabled. 1 = PCI master access to main memory is enabled.	1
5:2	**Reserved**		1H
6	PARERR_REP	Parity Error Respond: 0 = the North Bridge does not assert PERR# when a PCI parity error detected 1 = the North Bridge asserts PERR# when a PCI parity error detected. .	0
15:7	**Reserved**		0

Table 3.42 Status Register (STAT)

Bit	Name	Function	Def.
Status Register (STAT) - Configuration Index **06H**			
22:16	**Reserved**		All ‘0’s
23	FAST_B2B_STAT	Fast Back-to-Back status – Use this bit when EN_PCI_FAST_DCD bit in the PCIC register is set. This bit indicates that NB as a target can accept fast back-to-back cycles from another master.	0
24	DATA_PAR_DET	Data Parity Detected: Set this bit when operating as a bus master and either the PERR# output is driven low by the North Bridge or the target asserts PERR# and bit 6 of the Device Control Register is set. Reset this bit writing a 1.	0
26:25	DEVSEL_TIM	DEVSEL Timing: These bits indicate the slowest time that NB will return DEVSEL#. 00 = fast, 01 = medium, 10 = slow, 11 = reserved. Note that these bits are hard-wired to 01	01
27	**Reserved**		0
28	REC_TAG_ABRT	Receive Target Abort: Reading a 1 indicates receiving a target abort condition. Reset this bit by writing a 1.	0
29	REC_MST_ABRT	Receive Master Abort: Reading a 1 indicates receiving a master abort condition(not including master abort generated from a special cycle). Reset this bit by writing a 1.	0
30	**Reserved**		0
31	DET_PAR_ERR	Detect parity error: When the North Bridge detects a PCI parity error, this bit will be set to 1. Reset this bit by writing a 1.	0

Table 3.43 Revision ID Register (RID)

Bit	Name	Function	Def.
Revision ID Register (RID) - Configuration Index **08H**			
7:0	REVISION_ID	Revision ID number. These bits are hard wired.	00H

Table 3.44 Class Register (CLASS)

Bit	Name	Function	Def.
Class Register (CLASS) - Configuration Index **09H**			
31:8	CLASS_CODE	Class Code. These bits are hard-wired.	060000H

3

4

4. South Bridge

4.1. South Bridge Module

The South Bridge module is an enhanced PCI-to-ISA bridge module that provides AT/ISA functionality. The module also contains a bus mastering IDE controller for support of up to four ATA-compliant devices. A two-port Universal Serial Bus (USB) host controller provides high speed, Plug & Play expansion for a variety of new consumer peripheral devices.

4.1.1. South Bridge Features

4.1.1.1. Front-PCI Interface

- PCI protocol for transfers on Front-PCI
- Up to 33 MHz operation
- Point-to-point only connection to North Bridge enables higher bandwidth
- Uses non-preemptable arbiter connection to North Bridge
- Subtractive decode handled internally in conjunction with back-side PCI bus
- Front-PCI signals do not include SERR# and PERR#

4.1.1.2. PCI Interface

- Off chip “back-side” PCI interface
- PCI 2.1 compliant
- Up to 33 MHz operation
- Internal PCI master for IDE and USB controllers
- Subtractive agent for unclaimed transactions
- Supports PCI initiator-to-Front-PCI
- PCI-to-ISA interrupt mapper/translator
- Non-supported modes:
  - Devices internal to South Bridge (that is, IDE, USB, ISA, etc.) cannot master to memory on back-side PCI bus
  - Legacy DMA not supported to memory located on back-side PCI bus
  - South Bridge bit-buckets subtractively decoded I/O cycles originating from back-side PCI bus

4.1.1.3. Bus Mastering IDE Controller

- One channel with support for up to two IDE devices
- Second IDE channel for two more devices off GPIO
- Independent timing for master and slave devices
- PCI bus master burst reads and writes
- Ultra DMA (ATA-4) support
- Multiword DMA support
- Programmed I/O (PIO) Modes 0-4 support

4.1.1.4. Universal Serial Bus

- Two independent USB interfaces
- USB 1.1 specification compliant
- Open Host Controller Interface (OpenHCI) 1.0 specification compliant
- Second generation proven core design
- Overcurrent and power control support
- PCI bus master burst reads and writes

4.1.1.5. Integrated SuperI/O

- Floppy disk controller
- Two standard serial ports fully compatible with 16550A and 16450
- Infrared communication port

- Parallel Port: Extended Capabilities Port (ECP) that is IEEE 1284 compliant, including level 2
- Real-time clock compatible with DS1287, MC146818, and CP87911
- 8042 keyboard controller
- ACCESS.bus interface (compatible with the physical layer of SMBus and I^2C)

4.1.1.6. AT Compatibility

- 8259A-equivalent interrupt controllers
- 8254-equivalent timer
- 8237-equivalent DMA controllers
- Port A, B, and NMI logic
- Positive decode for AT I/O space

4.1.1.7. ISA Interface

- Boot ROM and keyboard chip select
- Extended ROM to 16 MB
- Two general purpose chip selects
- NAND Flash support

4.1.1.8. Power Management

- Automated CPU Suspend modulation
- I/O Traps and Idle Timers for peripheral power management
- Software SMI and Stop Clock for APM support
- Keyboard and mouse activity detect for screen wake-up

4.1.1.9. GPIOs

- Eight GPIOs: All have the capability to generate Power Management Events (PMEs)

4.2. Architecture

The South Bridge architecture provides the internal functional blocks shown in Figure 4-1 "Internal Block Diagram".

- Front-PCI interface / Back-side PCI bus
- PCI configuration registers
- IDE controller (UDMA-33)
- USB controller
- Integrated SuperI/O
- ISA bus interface
- AT compatibility logic
- Power management
- GPIOs
- ZF Logic

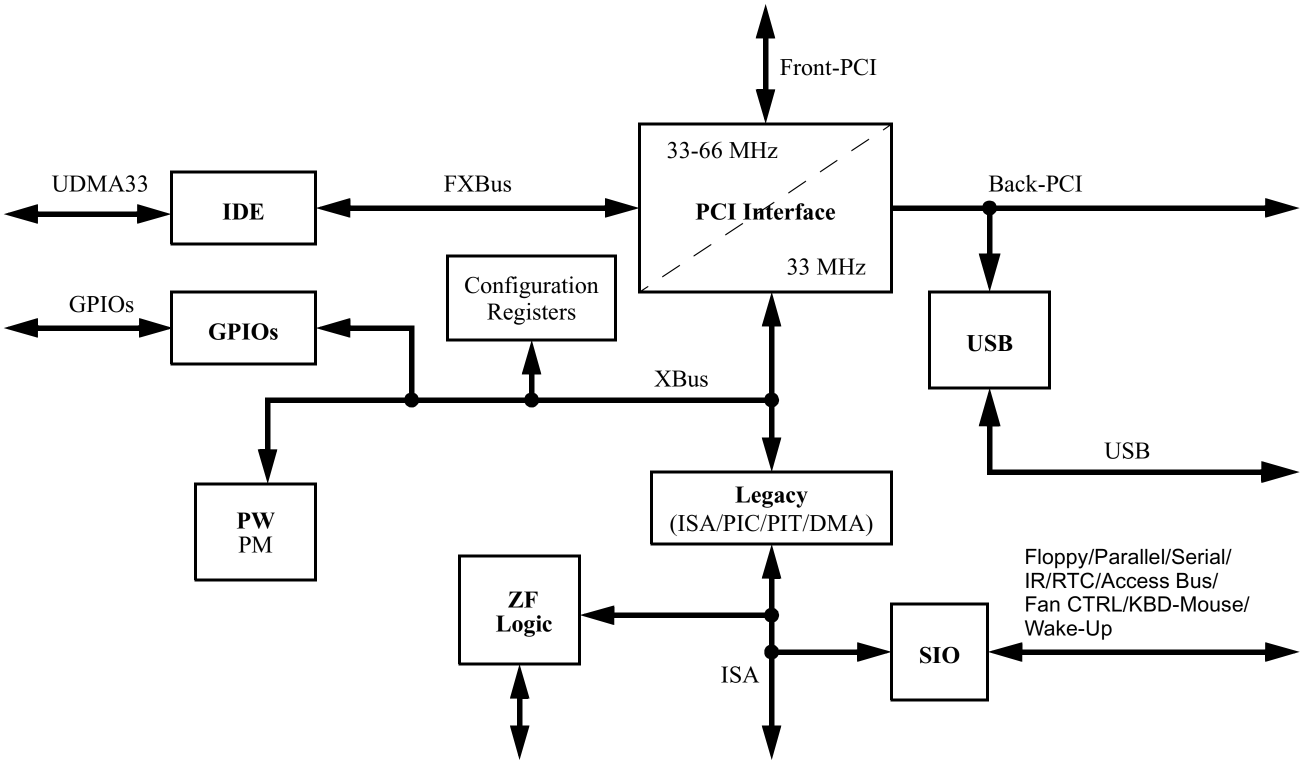


Figure 4-1 Internal Block Diagram

4.2.1. Front-side PCI / Back-Side PCI Bus

The South Bridge provides a PCI bus interface that is both a slave for PCI cycles initiated by the North Bridge or other PCI master devices, and a non-preemptable master for DMA transfer cycles. The module is also a standard PCI master for the IDE controller. The South Bridge supports positive decode for configurable memory and I/O regions, and implements a subtractive decode option for unclaimed PCI accesses. The South Bridge also generates address and data parity, and performs parity checking. The arbiter for the Front-PCI interface is located in North Bridge.

Configuration registers are accessed through the PCI interface using the PCI Bus Type 1 configuration mechanism as described in the PCI 2.1 Specification.

The main objective of the Front-PCI interface is two-fold:

- Provide a fast PCI compliant internal bus between the North and South Bridges.
- Enable a higher bandwidth interface to system memory for high bandwidth devices.

To achieve these goals, the following describes how the Front-PCI interface manages PCI cycles.

4.2.1.1. North Bridge Mastered Cycles on Front-PCI

All North Bridge initiated cycles are acted upon by the South Bridge following the normal PCI rules for active/subtractive decode using DEVSEL. Memory writes are automatically posted. Reads will be retried if they are *not* destined for actively decoded devices on the FXBus or the XBus. This means that a read to back-side PCI or ISA devices are automatically treated as a delayed transaction through the PCI retry mechanism. This allows the high bandwidth devices access to the Front-PCI interface while the response from a possibly slow device can be accumulated.

All types of configuration cycles are supported and handled appropriately according to the PCI specification.

4.2.1.2. Back-Side PCI Master Cycles to Front PCI

Memory cycles mastered by external PCI devices on the back-side PCI bus are actively taken if they are within the system memory address range. Memory cycles to system memory are forwarded to the Front-PCI interface. Burst transfers are stopped on every cache line boundary to allow efficient buffering in the Front-PCI interface block.

I/O and configuration cycles mastered by external back-side PCI devices which are subtractively decoded by the South Bridge will not be handled. They will be discarded.

4.2.1.3. South Bridge Internal or ISA Master Cycles to Front PCI

Only memory cycles (not I/O cycles) are supported by the internal ISA or legacy DMA masters. These memory cycles are always forwarded to the Front-PCI interface.

4.2.1.4. Back-Side PCI Bus

The Back-Side PCI bus is a fully-compliant 5V tolerant PCI bus. PCI slots are connected to this bus. Support for up to three bus masters is provided. The arbiter is in the South Bridge.

4.2.2. IDE Controller

The South Bridge integrates a PCI bus mastering ATA-4 compatible IDE controller. The IDE controller supports Ultra DMA, Multi-word DMA, and Programmed I/O (PIO) modes. The controller contains two channels with two devices supported per channel, for a total of up to four devices. To enable the second channel, GPIO pins must be reconfigured. Program the data-transfer speed for each device independently. This allows high-speed IDE peripherals to coexist on the same channel as lower speed devices. Faster devices must be ATA-4 compatible.

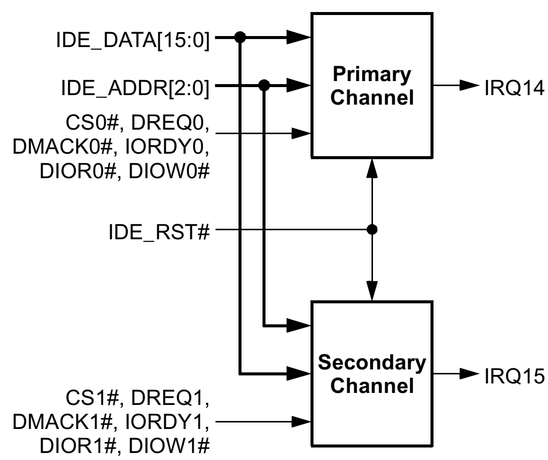


Figure 4-2 IDE Channel Connections

4.2.3. Universal Serial Bus

The South Bridge provides two complete, independent USB ports. Each port contains a Data “–” and a Data “+” pin.

The USB ports are Open Host Controller Interface (OpenHCI) version 1.0 compliant. The OpenHCI specification provides a register-level description for a host controller, as well as common industry hardware/software interface and drivers.

The USB host controller masters Front-PCI to fetch setup and control information related to OpenHCI. The USB host controller also masters the PCI bus performing read and write bursts to move, transmit, and receive packet data to system memory.

4.2.4. Integrated SuperI/O

The integrated SuperI/O is based on 11 logical devices (shown in Table 4.1), the host interface, and a configuration register set, all built around a central, internal 8-bit bus.

Table 4.1 Logical Devices

LDN (Logical Device Number)	Functional Block
00h	Floppy Disk Controller (FDC)
01h	Parallel Port (PP)
02h	Serial Port 2 (SP2)
03h	Serial Port 1 (SP1)
04h	System Wake-Up Control (SWC)
05h	Keyboard and Mouse Controller (KBC) — Mouse interface
06h	Keyboard and Mouse Controller (KBC) — Keyboard interface
07h	Infrared Communication Port (IRCP)
08h	ACCESS.Bus (ACB)
09h	Reserved
0Ah	Real Time Clock (RTC)

The host interface serves as a bridge between the external ISA interface and the internal bus. It supports 8-bit I/O read, 8-bit I/O write, and 8-bit DMA transactions as defined in Personal Computer Bus Standard P996.

The central configuration registers are structured as a subset of the Plug and Play Standard Registers, defined by Intel® and Microsoft® in Appendix A of the Plug and Play ISA Specification Version 1.0a. All system resources assigned to the functional blocks (I/O address space, DMA channels, and IRQ lines) are configured in, and managed by the central configuration register set. In addition, some function-specific parameters are configurable through this unit and distributed to the functional blocks through special control signals

4.2.5. ISA Bus Interface

The South Bridge provides an ISA bus interface for subtractive-decoded memory and I/O cycles on PCI. The South Bridge is the default subtractive decoding agent and forwards all unclaimed memory and I/O cycles to the ISA interface; however, the South Bridge may be configured to ignore either I/O, memory, or all unclaimed cycles (subtractive decode disabled).

DRAM is not supported on the ISA.

When you disable the ZF-Logic, the ISA interface of the South Bridge remaps pin functions to include the followings signals in addition to the signals used for an ISA interface (chip pin):

- IOCS0# (mem_cs1), IOCS1# (mem_cs2)
  - Asserted on I/O read/write transactions from/to a programmable address range.
- DOCCS# (mem_cs3)
  - Asserted on memory read/write transactions from/to an 8 KB programmable window.
- ROMCS# (mem_cs0)
  - Asserted on memory read/write to upper 16 MB of address space. Configurable via the ROM Mask Register (F0 Index 6Eh).

The Boot Flash supported by the South Bridge can be up to 16 MB using the ROMCS# signal.

Forward all unclaimed memory and I/O cycles to the Boot Flash and I/O Peripheral interface when subtractive decode is enabled.

Assert the Disk-On-Chip chip-select signal (DOCCS#) on any memory read or memory write transaction from/to a predefined 8 KB window in the address range 0C0000h-0EFFFFh. Program the 8 KB window via the DOCBASE register (F0 Index 78h). The window's base address must be on an 8 KB address boundary.

4.2.5.1. AT Compatibility Logic

The South Bridge integrates:

- Two 8237-equivalent DMA controllers with full 32-bit addressing
- Two 8259-equivalent interrupt controllers providing 13 individually programmable external interrupts
- An 8254-equivalent timer for refresh, timer, and speaker logic
- NMI control and generation for PCI system errors and all parity errors
- Support for standard AT keyboard controllers
- Positive decode for the AT I/O register space
- Reset control

4.2.5.2. DMA Controller

The South Bridge supports the industry standard DMA architecture using two 8237-compatible DMA controllers in cascaded configuration. South Bridge supported DMA functions include:

- Standard seven-channel DMA support
- 32-bit address range support via high page registers
- IOCHRDY extended cycles for compatible timing transfers
- ISA bus master device support using Cascade mode

When there is at least one active DMA request (DRQ), DACK[7:0] indicates which DRQ is granted. When there is no active DRQ, DACK[7:0] are encoded with 00010000b, which is an unused DACK.

4.2.5.3. Programmable Interval Timer

The South Bridge contains an 8254-equivalent programmable interval timer. This device has three timers, each with an input frequency of 1.193 MHz. Each timer can be individually programmed to different modes.

4.2.5.4. Programmable Interrupt Controller

The South Bridge contains two 8259-equivalent programmable interrupt controllers, with eight interrupt request lines each, for a total of 16 interrupts. The two controllers cascade internally, and two of the interrupt request inputs are connected to the internal circuitry. This allows a total of 13 externally available interrupt requests.

Individually select any South Bridge IRQ signal as an edge- or level-sensitive type. Route the four PCI interrupt signals internally to any PIC IRQ.

4.2.6. Power Management

The South Bridge integrates advanced power management features including idle timers for common system peripherals, address trap registers for programmable address ranges for I/O or memory accesses, clock throttling with automatic speedup for the CPU clock, and software CPU stop clock.

4.2.7. GPIO Interface

Up to eight GPIOs in the South Bridge provide for system control. The ZFx86 contains 8 GPIO pins. The features include power management event (PME) generation. This means that any of the 8 GPIO pins set in input mode can be used to wake up the processor. That is, each GPIO pin can be programmed to generate an SMI or SCI. The features of the GPIO pins include the following:

- **PME Debounce Enable** — Enables or disables IRQ debounce (debounce period = 16 ms):
- **PME Polarity** — Selects the signal polarity of the signal that issues a PME from the corresponding GPIO pin (falling/low or rising/high)
- **PME Edge/Level Select** — Selects the signal type (edge or level) that issues a PME from the corresponding GPIO pin.
- **Lock** — This bit locks the corresponding GPIO pin. Once set to 1 by software, it can only be cleared to 0 by system reset or power-off.
- **Pull-Up Control** — Enables/disables the internal pull-up capability of the corresponding GPIO pin. It supports open-drain output signals with internal pull-ups and TTL input signals.
- **Output Type** — Controls the output buffer type (open-drain or push-pull) of the corresponding GPIO pin.
- **Output Enable** — Indicates the GPIO pin output state. It has no effect on input.

Note: To clear the GPIO PME source, complete these two items: (1) write a “1” to the source bit in the GPIO Status Register (F0BAR0+0Ch), and (2) perform a read to that same register. For more information, see Table 4.31 on page 225.

4.2.8. ZF-Logic

See Chapter 5. "ZF-Logic and Clocking" on page 397.

4.3. Signal Descriptions

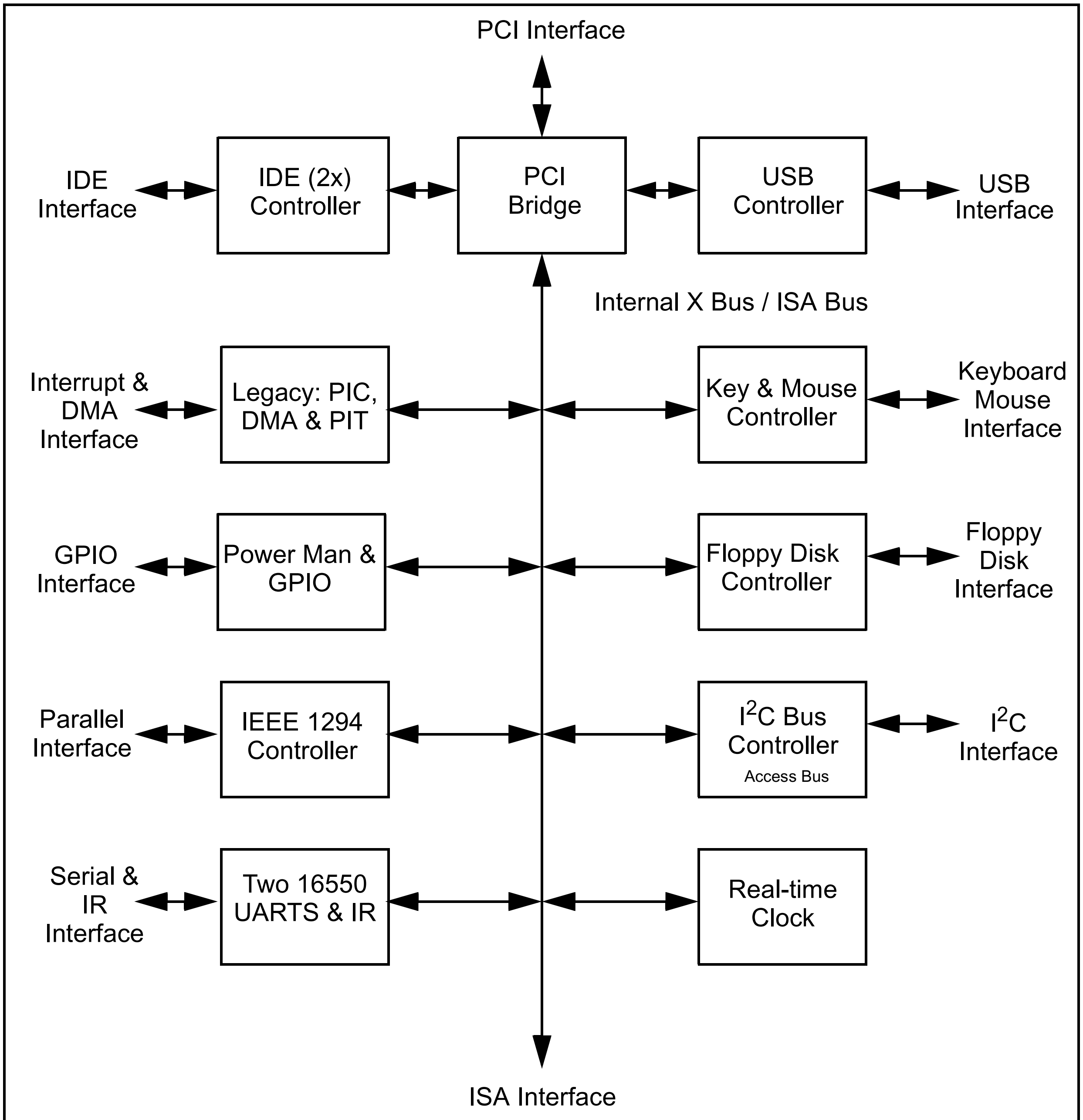


Figure 4-3 South Bridge Block Diagram

4.3.1. System Interface Signals

Table 4.2 System Interface Signals

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
MEM_CS[0]	B04	O (8mA)	---	**ROM Chip Select** ROMCS# is the enable pin for the BIOS ROM and is asserted for ISA memory accesses that are in the BIOS address range. ROM size selection is made via F0 Index 52h[2]. This is part of the ZF-Logic described in 5.3. "ISA Memory Mapper for Flash/SRAM" on page 404.
POR_N	C19	I smt	---	**POR_N** POR# is the system reset signal generated from the power supply to indicate that the system should be reset.

Table 4.3 Clock and Crystal Interface Signals

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
CLK_14MHZ	AF16	I	---	**14MHz Clock** This buffered 14.31818MHz input is used for the ISA bus OSC. This signal is derived internally if the RTC is enabled.
CLK_48MHZ	AE15	I	---	**48MHz Clock** This input connects to an external 48MHz clock source and is used by the SIO and USB.
KHZ32_C	AE01	I	---	**32KHz Crystal Oscillator Connection** X1C and X2C are used with a capacitor and resistor to create a 32KHz oscillator connection.
KHZZ32	AF01	I	---	**32KHz Clock Connection** If a 32KHz crystal is not used in the system design, this signal can be directly connected to a 32KHz external clock.

Table 4.4 CPU Interface Signals

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
CPU_RST	Internal Pin	O (8mA)	---	**CPU Reset** CPU_RST resets the CPU and is asserted for approximately 100µs after the negation of POR.
INTR	Internal Pin	O (4mA)	---	**CPU Interrupt Request** INTR is the level output from the integrated 8259 PICs and is asserted if an unmasked interrupt request (IRQ_n) is sampled active.
IRQ13	Internal Pin	I	F0 Index 53h[1] = 1	**Interrupt Request 13/Floating Point Error Interrupt** IRQ13 is the input from floating point unit indicating that an error was detected and that INTR should be asserted.
FERR#	Internal Pin		F0 Index 53h[1] = 0	**Floating Point Error Interrupt** FERR# is an input from a processor that supports the FERR# signal. It indicates that a floating point error was detected and that IGNNE# should be asserted. In order for the above event to occur, pin E11 must be programmed to function as: IGNNE# (F0 Index 4Ah[3] = 0). Note that the IGNNE# (pin E11) is only available in the 456 PBGA.
SMI#	Internal Pin	I/O (4mA)	---	**System Management Interrupt** SMI# is a level-sensitive interrupt to the CPU that can be configured to assert on a number of different system events. After an SMI# assertion, System Management Mode (SMM) is entered, and program execution begins at the base of SMM address space. Once asserted, SMI# remains active until the SMI source is cleared.
NMI	Internal Pin	O (8mA)	---	**Non-Maskable Interrupt Request** Non-maskable Interrupt Request is an output that causes the processor to suspend execution of the current instruction stream and begin execution of an NMI interrupt service routine.
SUSPA#		I	---	**CPU Suspend Acknowledge** SUSPA# is a level input from the processor. When asserted it indicates the CPU is in Suspend mode as a result of SUSP# assertion or execution of a HALT instruction.

Table 4.4 CPU Interface Signals (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
SUSP#		O (4mA)	---	**CPU Suspend** SUSP# asserted requests that the CPU enter Suspend mode. The CPU then asserts SUSPA# to complete the handshake, but only after executing a HALT instruction and turning off the appropriate internal clocks. The SUSP# pin is then deasserted by the South Bridge upon detection of a pre-defined Speedup or Wakeup/Resume event. If the SUSP#/SUSPA# handshake is configured as a system 3 Volt Suspend, the deassertion of SUSP# will be delayed to allow the system clock chip and the processor's internal clocks to stabilize. The SUSP#/SUSPA# handshake can occur as a result of a write to the Suspend Notebook Command Register (F0 Index AFh), or an expiration of the Suspend Modulation OFF Count Register (F0 Index 94h) when Suspend Modulation is enabled. Suspend Modulation is enabled via F0 Index 96[0].

4.3.2. Back-Side PCI Interface Signals

Table 4.5 Back-Side PCI Bus Interface Signals

Signal Name	**Pin No. (PU/PD)**	**Type (Drive)**	**Function Selection**	**Description**
PCICLK_C	U25	I	---	**PCI Clock** An input clock signal to the backside PCI interface of the South Bridge. It runs at the PCI clock frequency and is used to drive most of the South Bridge circuitry.
PCI_RST_N	U26	O (14mA)	---	**PCI Reset** PCI_RST# is the reset signal for the PCI bus. Asserted for approximately 100µs after the negation of POR.
CLKRUN#	not supported	I/O (PCI)	---	**Clock Run** CLKRUN# is an I/O that follows the PCI 2.2 defined protocol.

Table 4.5 Back-Side PCI Bus Interface Signals (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
AD[31:0]	0 = U24 1 = V26 2 = V24 3 = V25 4 = W26 5 = V23 6 = W25 7 = W24 8 = Y26 9 = Y25 10 = Y23 11 = Y24 12 = AA26 13 = AA25 14 = AA24 15 = AB26 16 = AB25 17 = AB24 18 = AC26 19 = AB23 20 = AC25 21 = AC24 22 = AD25 23 = AD26 24 = AE26 25 = AE25 26 = AD24 27 = AF26 28 = AF25 29 = AE24 30 = AD23 31 = AF24	I/O, t/s (PCI)	---	**PCI Address/Data** AD[31:0] is a physical address during the first clock of a PCI transaction. It is the data during subsequent clocks. When the South Bridge is a PCI master, AD[31:0] are outputs during the address and write data phases, and are inputs during the read data phase of a transaction. When the South Bridge is a PCI slave, AD[31:0] are inputs during the address and write data phases, and are outputs during the read data phase of a transaction.

Table 4.5 Back-Side PCI Bus Interface Signals (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
C/BE[3:0]_N	0 = T24 1 = T26 2 = T25 3 = R24	I/O, t/s (PCI)	---	**PCI Bus Command and Byte Enables** During the address phase of a PCI transaction, C/BE[3:0]# define the bus command. During the data phase of a transaction, C/BE[3:0]# are the data byte enables. C/BE[3:0]# are outputs when the South Bridge is a PCI master and are inputs when it is a PCI slave. The command encoding and types are: 0000 = Interrupt Acknowledge 0001 = Special Cycle 0010 = I/O Read 0011 = I/O Write 0100 = Reserved 0101 = Reserved 0110 = Memory Read 0111 = Memory Write 1000 = Reserved 1001 = Reserved 1010 = Configuration Read 1011 = Configuration Write 1100 = Memory Read Multiple 1101 = Dual Address Cycle (Rsvd) 1110 = Memory Read Line 1111 = Memory Write and Invalidate
INT9# PCI_INT_A	D02	I	---	**PCI Interrupt Pins** The South Bridge provides inputs for the optional “level-sensitive” PCI interrupts (also known in industry terms as PIRQx#). These interrupts may be mapped to IRQs of the internal 8259s using PCI Interrupt Steering Registers 1 and 2 (F0 Index 5Ch and 5Dh). Optionally routed internally. For detailed information about routing PCI interrupts using the ZFx86 BIOS, refer to the “Routing ZFx86 PCI Interrupts” document (P/N 9150-0015-00) on the ZF Micro Device website: *http://www.zfmicro.com*
INT10# PCI_INT_B	E04			
INT11# PCI_INT_C	D01			
INT12# PCI_INT_D	E03			
REQ1_N	N23	I	---	**PCI Bus Requests** The South Bridge asserts REQ# in response to a DMA request or ISA master request to gain ownership of the PCI bus. The REQ# and GNT# signals are used to arbitrate for the PCI bus.
REQ0_N	M25			
GNT1_N	M24	O (PCI)	---	**PCI Bus Grants** GNT# is asserted by an arbiter that indicates to the South Bridge that access to the PCI bus has been granted.
GNT0_N	L26			

Table 4.5 Back-Side PCI Bus Interface Signals (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
FRAME_N	P24	I/O, t/s (PCI)	---	**PCI Cycle Frame** FRAME# is asserted to indicate the start and duration of a transaction. It is deasserted on the final data phase. FRAME# is an input when the South Bridge is a PCI slave.
IRDY_N	P25	I/O, t/s (PCI)	---	**PCI Initiator Ready** IRDY# is driven by the master to indicate valid data on a write transaction, or that it is ready to receive data on a read transaction. When the South Bridge is a PCI slave, IRDY# is an input that can delay the beginning of a write transaction or the completion of a read transaction. Wait cycles are inserted until both IRDY# and TRDY# are asserted together.
TRDY_N	R26	I/O, t/s (PCI)	---	**PCI Target Ready** TRDY# is asserted by a PCI slave to indicate it is ready to complete the current data transfer. TRDY# is an input that indicates a PCI slave has driven valid data on a read or a PCI slave is ready to accept data from the South Bridge on a write. TRDY# is an output that indicates the South Bridge has placed valid data on AD[31:0] during a read or is ready to accept the data from a PCI master on a write. Wait cycles are inserted until both IRDY# and TRDY# are asserted together.
STOP_N	P26	I/O, t/s (PCI)	---	**PCI Stop** As an input, STOP# indicates that a PCI slave wants to terminate the current transfer. The transfer will either be aborted or retried. As an output, STOP# is asserted with TRDY# to indicate a target disconnect, or without TRDY# to indicate a target retry. The South Bridge will assert STOP# during any cache line crossings if in single transfer DMA mode or if busy.
LOCK_N	N25	I/O, t/s (PCI)	---	**PCI Lock** LOCK# indicates an atomic operation that may require multiple transactions to complete. If the South Bridge is currently the target of a LOCKed transaction, any other PCI master request with the South Bridge as the target will be forced to retry the transfer. The South Bridge does not generate LOCKed transactions.

Table 4.5 Back-Side PCI Bus Interface Signals (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
DEVSEL_N	R25	I/O, t/s (PCI)	---	**PCI Device Select** DEVSEL# is asserted by a PCI slave, to indicate to a PCI master and subtractive decoder that it is the target of the current transaction. As an input, DEVSEL# indicates a PCI slave has responded to the current address. As an output, DEVSEL# is asserted one cycle after the assertion of FRAME#, and remains asserted to the end of a transaction as the result of a positive decode. DEVSEL# is asserted four cycles after the assertion of FRAME# if the South Bridge is selected as the result of a subtractive decode. The subtractive decode sample point can be configured in F0 Index 41h[2:1]. These cycles are passed to the ISA bus.
PAR	N26	I/O, t/s (PCI)	---	**PCI Parity** PAR is the parity signal driven to maintain even parity across AD[31:0] and C/BE[3:0]#. The South Bridge drives PAR one clock after the address phase and one clock after each completed data phase of write transactions as a PCI master. It also drives PAR one clock after each completed data phase of read transactions as a PCI slave.
PERR_N	N24	I/O, t/s (PCI)	---	**PCI Parity Error** PERR# is pulsed by a PCI device to indicate that a parity error was detected. If a parity error was detected, PERR# is asserted by a PCI slave during a write data phase and by a PCI master during a read data phase. When the South Bridge is a PCI master, PERR# is an output during read transfers and an input during write transfers. When the South Bridge is a PCI slave, PERR# is an input during read transfers and an output during write transfers. Parity detection is enabled through F0 Index 04h[6]. An NMI is generated if I/O Port 061h[2] is set. PERR# can assert SERR# if F0 Index 40h[1] is set.

Table 4.5 Back-Side PCI Bus Interface Signals (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
SERR_N	M26	I/O, OD (PCI)	---	**PCI System Error** SERR# is pulsed by a PCI device to indicate an address parity error, data parity error on a special cycle command, or other fatal system errors. SERR# is an open drain output reporting an error condition, and an input indicating that the South Bridge should generate an NMI. As an input, SERR# is asserted for a single clock by the slave reporting the error. System error detection is enabled with F0 Index 04h[8]. An NMI is generated if I/O Port 061h[2] is set. PERR# can assert SERR# if F0 Index 40h[1] is set.

Table 4.6 IDE Interface Signals

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
IDE_CS0_N	AE20	O (IDE)	---	**IDE Chip Select for Channels 0 and 1** The chip select signals are used to select the command block registers in an IDE device.
IDE_CS1_N	AF21			
IDE_DIOR0_N	AC20	O (IDE)	---	**IDE I/O Read for Channels 0 and 1** IDE_IOR0# is the read signal for Channel 0 and IDE_IOR1# is the read signal for Channel 1. Each signal is asserted on read accesses to the corresponding IDE port addresses.
IDE_DIOR1_N GPIO[3][a]	AD11			
IDE_DIOW0_N	AE21	O (IDE)	---	**IDE I/O Write for Channels 0 and 1** IDE_IOW0# is the write signal for Channel 0 and IDE_IOW1# is the read signal for Channel 1. Each signal is asserted on write accesses to corresponding the IDE port addresses.
IDE_DIOW1_N GPIO[2][a]	AF11			
IDE_DMA_ACK0_N	AC22		---	**DMA Acknowledge Channels 0 and 1** The DMACK# acknowledges the DREQ request to initiate DMA transfers.
IDE_DMA_ACK1_N GPIO[1][a]	AE11			
IDE_DMA_REQ0_N	AE23	I	---	**DMA Request Channels 0 and 1** The DREQ is used to request a DMA transfer from the South Bridge. The direction of the transfers is determined by the IDE_IOR/IOW signals.
IDE_DMA_REQ1_N GPIO[5][a]	AE10			
IDE_IORDY0	AD22	I	---	**I/O Ready Channels 0 and 1** When deasserted, these signals extend the transfer cycle of any host register access when the device is not ready to respond to the data transfer request.
IDE_IORDY1 GPIO[6][a]	AD10			

Table 4.6 IDE Interface Signals (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
IDE_ADDR[0]	AD21	O (IDE)	---	**IDE Address Bits** These address bits are used to access a register or data port in a device on the IDE bus.
IDE_ADDR[1]	AF22			
IDE_ADDR[2]	AE22			
IDE_DATA[15:0]	0 = AD20 1 = AF20 2 = AD19 3 = AE19 4 = AF19 5 = AD18 6 = AC18 7 = AE18 8 = AF18 9 = AE17 10 = AD17 11 = AF17 12 = AC16 13 = AD16 14 = AE16 15 = AD15	I/O (IDE)	---	**IDE Data Lines** IDE_DATA[15:0] transfers data to/from the IDE devices.
IDE_RST_N	AF23	O (IDE)	---	**IDE Reset** This signal resets all devices attached to the IDE interface.
IRQ14	E02	I	---	**Interrupt Request 14** Normally connected to the primary IDE channel.
IRQ15	E01	I	---	**Interrupt Request 15** Normally connected to the secondary IDE channel.

a. See Table 4.8 "GPIO Interface Signals"

Table 4.7 USB Interface Signals

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
POWER_EN	AE12	O (4mA)	---	**Power Enable** Enables the power to a bus-powered USB hub.
OVER_CUR1#	AD13	I	---	**Over Current** Indicates the USB hub detected an overcurrent on the USB.
OVER_CUR2#	AF12			
PORT1_P	AE13	I/O (USB)	---	**USB Port 1 Data Positive** The Universal Serial Bus Data Positive for port 1.
PORT1_M	AF13	I/O (USB)	---	**USB Port 1 Data Minus** The Universal Serial Bus Data Minus for port 1.
PORT2_P	AF14	I/O (USB)	---	**USB Port 2 Data Positive** The Universal Serial Bus Data Positive for port 2.
PORT2_M	AE14	I/O (USB)	---	**USB Port 2 Data Minus** The Universal Serial Bus Data Minus for port 2.

Table 4.8 GPIO Interface Signals

Signal Name	Pin No. (PU/PD)	Type (Drive)	Description
gpio[0]	AD12	I/O	GPIO/0 (optional 32KHz out). Reference IO_CLK32K_OE[ca]
gpio[1]	AE11	I/O	GPIO/1 (optional 2nd IDE IDE_DMA_ACK1_N)[b]
gpio[2]	AF11	I/O	GPIO/2 (optional 2nd IDE IDE_DIOW1_N)[b]
gpio[3]	AD11	I/O	GPIO/3 (optional 2nd IDE IDE_DIOR1_N)[b]
gpio[4]	AF10	I/O	GPIO/4 (can set GPIO[0] to 32 Khz Out)[c]
gpio[5]	AE10	I/O	GPIO/5 (optional 2nd IDE IDE_DMA_REQ1_N)[b]
gpio[6]	AD10	I/O	GPIO/6 (optional 2nd IDE IDE_IORDY1)[b]
gpio[7]	AF09	I/O	GPIO/7

a. GPIO[0] can also be Chip Select for External BUR. See Table 5.42 "Composite BootStrap Register Map" on page 432.
b. See 'IO_IDE_ON_GPIO — Drive IDE channel 2 onto gpio. Must also have gpio conditioned to correct direction corresponding to IDE pin functionality.' on page 240.
c. See Figure 5-9 "System Clocking and Control" on page 437

Example: Setting the ISA Bus Clock

Although the only legal values of the ISA Bus Clock are 2 and 3, this example steps through all eight values. It illustrates how to access the South Bridge configuration registers. The “outpd” on line 4 writes a 32-bit value to the port address specified.

For the pattern output to CF8, see “PCI Configuration Space and Access Methods” on page 191, and “PCI Configuration Address Register (0CF8h)” on page 191

```
1     printf ("\r\n\n\n Step Through ISA Bus Clock Divisors\r\n");
2        for (uii = 0; uii < 8; uii++)
3            {
4            outpd (0xCF8, 0x80009050);
5            outp (0xCFC, ((inp (0xCFC) & 0xF0) | uii));
6            printf ("Divisor is Now %i - Press Key to Step", uii);
7            getch();
8            }
```

Table 4.9 Full ISA Interface

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
ISACLK_N	ACO2	O (14mA)	F0 Index 4Ah[4] = 0	**ISA Bus Clock** ISACLK is derived from PCICLK and is typically programmed for 8.33MHz. F0 Index 50h[2:0] is used to program the ISA clock divisor. These bits determine the divisor of the PCI clock used to make the 8.33MHz ISA bus clock. If F0 Index 50h[2:0] is set to: 010 = Divide by three (PCI clock = 25MHz) 011 = Divide by four (PCI clock = 33MHz) All other values are invalid and can produce unexpected results. See Table 4.30 "F0 Index xxh: PCI Header and Bridge Configuration Registers".
MASTER_N	Not Supported	I	F0 Index 4Ah[4] = 0	**Master** The MASTER# input asserted indicates an ISA bus master is driving the ISA bus and that it may access any device on the system board.
TC	A12	O (8mA)	F0 Index 4Ah[4] = 0	**Terminal Count** TC signals the final data transfer of a DMA transfer. TC is accepted only when a DACK signal is active.
AEN	C13	O (8mA)	F0 Index 4Ah[4] = 0	**Address Enable** AEN asserted indicates to ISA memory devices that a valid address for a DMA transfer is present on SA[23:0], and for I/O devices to ignore this address and any data on the ISA bus.
BALE	AD2	O (8mA)	F0 Index 4Ah[4] = 0	**Buffered Address Latch Enable** BALE indicates when SA[23:0] and SBHE# are valid and may be latched. For DMA transfers, BALE remains asserted until the transfer is complete.

Table 4.9 Full ISA Interface (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
SD[15:0]	0 = AC07 1 = AD07 2 = AF06 3 = AE06 4 = AD06 5 = AF06 6 = AE05 7 = AD05 8 = AF04 9 = AC05 10 = AEO4 11 = AD04 12 = AF03 13 = AE03 14 = AF02 15 = AE02	I/O (8mA)	F0 Index 4Ah[5] = 0	**System Data Bus Bits 15-0**
SA[23:0]	0 = AC01 1 = AB02 2 = AB01 3 = AA03 4 = AA02 5 = Y04 6 = AA01 7 = Y02 8 = Y03 9 = Y01 10 = W03 11 = W02 12 = W01 13 = V03 14 = V04 15 = V02 16 = V01 17 = U02 18 = U03 19 = U01 20 = T04 21 = T03 22 = T02 23 = R03	I/O (8mA)	F0 Index 4Ah[5] = 0	**System Address Bus Lines 23-20** The SA[23:20] signals provide the address for memory and I/O accesses on the ISA bus. The addresses are outputs when the South Bridge owns the ISA bus and are inputs when an external ISA master owns the ISA bus.
IRQ1	Internal	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 1** Keyboard / Mouse

Table 4.9 Full ISA Interface (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
IRQ3	B02	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 3** Refer to IRQ1 signal description.
IRQ4	C01	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 4** Refer to IRQ1 signal description.
IRQ5	C02	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 5** Refer to IRQ1 signal description.
IRQ6	Internal	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 6** Refer to IRQ1 signal description.
IRQ7	D03	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 7** Refer to IRQ1 signal description.
IRQ8#	Internal	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 8** RTC
IRQ9, PCI_INT_A	D02	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 9 / PCI Int A** Refer to IRQ1 signal description.
IRQ10, PCI_INT_B	E04	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 10 / PCI Int B** Refer to IRQ1 signal description.
IRQ11, PCI_INT_C	D01	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 11 / PCI Int C** Refer to IRQ1 signal description.
IRQ12, PCI_INT_D	E03	I/O (4mA)	F0 Index 47h[0] = 0	**Interrupt Request 12 / PCI Int D** PCI Int D
IRQ14	E02	I	---	**Interrupt Request 14** Normally connected to the primary IDE channel.
IRQ15	E01	I	---	**Interrupt Request 15** Normally connected to the secondary IDE channel.
DRQ1	B14	I	F0 Index 4Ah[4] = 0	**DMA Requests** DRQ inputs are asserted by ISA DMA devices to request a DMA transfer. The request must remain asserted until the corresponding DACK is asserted. IDE DMA does not use ISA DMA.
DRQ5	B13			
DRQ3	Internal	I	F0 Index 47h[0] = 0	**DMA Request 3** The DRQ is used to request DMA service from the DMA controller.
DRQ6	Internal	I	F0 Index 47h[0] = 0	**DMA Request 6** **The DRQ is used to request DMA service from the DMA controller.**

Table 4.9 Full ISA Interface (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
DRQ7	Internal	I	F0 Index 47h[0] = 0	**DMA Request 7** The DRQ is used to request DMA service from the DMA controller.
DACK0	Internal	O (8mA)	F0 Index 47h[0] = 0	**DMA Acknowledge 0** IDE DMA does not use ISA DMA.
DACK1_N	A14	O (8mA)	F0 Index 47h[0] = 0	**DMA Acknowledge 1**
DACK5_N	A13	O (8mA)	F0 Index 47h[0] = 0	**DMA Acknowledge 5**
ZWS_N	AB04	I	F0 Index 47h[0] = 0	**Zero Wait States** ZWS_N asserted indicates that an ISA 8- or 16-bit memory slave can shorten the current cycle. The South Bridge samples this signal in the phase after BALE is asserted. If asserted, it shortens 8-bit cycles to three ISACLKs and 16-bit cycles to two ISACLKs.
SBHE_N	AC03	I/O (8mA)	F0 Index 47h[0] = 0	**System Bus High Enable** The South Bridge or ISA master asserts SBHE_N to indicate that SD[15:8] will be used to transfer a byte at an odd address. SBHE_N is an output during non-ISA master DMA operations. It is driven as the inversion of AD0 during 8-bit DMA cycles. It is forced low for all 16-bit DMA cycles. SBHE_N is an input during ISA master operations.
IOCS16_N	AF07	I	F0 Index 47h[0] = 0	**I/O Chip Select 16** IOCS16_N is asserted by 16-bit ISA I/O devices based on an asynchronous decode of SA[15:0] to indicate that SD[15:0] may be used to transfer data (8-bit ISA I/O devices use SD[7:0]).
MEMCS16_N	AE07	I/O (8mA)	F0 Index 47h[0] = 0	**Memory Chip Select 16** MEMCS_N is asserted by 16-bit ISA memory devices based on an asynchronous decode of SA[23:17] to indicate that SD[15:0] may be used to transfer data (8-bit ISA memory devices use SD[7:0]).
SMEMR_N	AE08	O (8mA)	F0 Index 47h[0] = 0	**System Memory Read** SMEMR_N is asserted for memory read accesses below 1MB. It enables 16-bit memory slaves to decode the memory address on SA[23:0].

Table 4.9 Full ISA Interface (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Function Selection	Description
SMEMW_N	AD08	O (8mA)	F0 Index 47h[0] = 0	**System Memory Write** **SMEMW_N is asserted for all memory write accesses below 1MB. It enables 16-bit memory slaves to decode the memory address on SA[23:0].**
MR	B20			**Master Reset** An active high MR input signal resets the device with its default settings.
MEMW_N	AC09	I/O (8mA)	---	**Memory Write** MEMW_N is asserted for all memory write accesses.
MEMR_N	AF08	I/O (8mA)	---	**Memory Read** MEMR_N is asserted for all memory read accesses.
IOR_N	AD09	I/O (8mA)	---	**I/O Read** IOR_N is asserted to request an ISA I/O slave to drive data onto the data bus.
IOW_N	AE09	I/O (8mA)	---	**I/O Write** IOW_N is asserted to request an ISA I/O slave to accept data from the data bus.
IOCHRDY	AD01	I/O, OD (8mA)	---	**I/O Channel Ready** IOCHRDY deasserted indicates that an ISA slave requires additional wait states. When the South Bridge is an ISA slave, IOCHRDY is an output indicating additional wait states are required.

4.3.3. Integrated SuperI/O Interface Signals

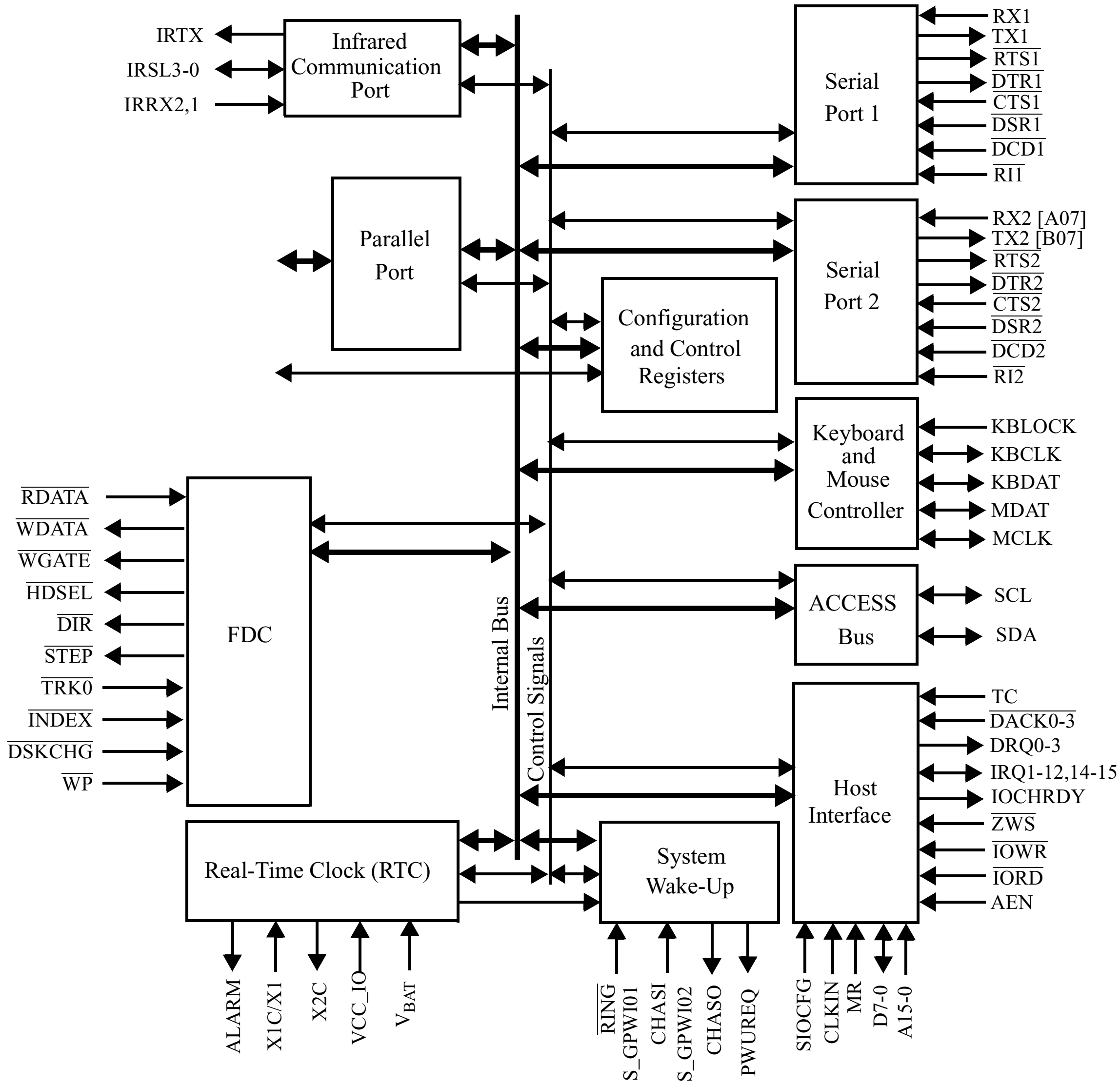


Table 4.10 Access Bus

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
SCL_C	B12	I/O	IN_{SM}/OD_6	V_{DD}	**ACCESS.bus Clock Signal** An internal pull-up is optional, depending upon the ACCESS.bus configuration register.
SDA	D13	I/O	IN_{SM}/OD_6	V_{DD}	**ACCESS.bus Data Signal** An internal pull-up is optional, depending upon the ACCESS.bus configuration register.

Table 4.11 Clock

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
CLKIN	Internal	I	IN_T	V_{DD}	**Clock In** A 48MHz clock input.

Table 4.12 Floppy Disk Controller

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
DIR_N	G03	O	OD_{14}-$O_{14/14}$	V_{DD}	**Direction** Determines the direction of the Floppy Disk Drive (FDD) head movement (active = step in, inactive = step out) during a seek operation. During reads or writes, $\overline{\text{DIR}}$ is inactive.
DR0_N	F01	O	OD_{14}-$O_{14/14}$	V_{DD}	**Drive Select 0** Decoded drive select output signal. $\overline{\text{DR0}}$ is controlled by Digital Output Register (DOR) bit 0.
DSKCHG_N	J02	I	IN_T	V_{DD}	**Disk Change** Indicates if the drive door has been opened. The state of this pin is stored in the Digital Input Register (DIR). This pin can also be configured as the RGATE data separator diagnostic input signal via the MODE command.
HDSEL	H01	O	OD_{14}, $O_{14/14}$	V_{DD}	**Head Select** Determines which side of the FDD is accessed. Active low selects side 1, inactive selects side 0.
INDEX_N	F03	I	IN_T	V_{DD}	**Index** Indicates the beginning of an FDD track.

Table 4.12 Floppy Disk Controller (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
MTR0_N	F02	O	OD_{14}, $O_{14/14}$	V_{DD}	**Motor Select 0** Active low, motor enable line for drive 0, controlled by bits D7-4 of the Digital Output Register (DOR). This signal is not available on the PPM, assuming that the external FDD is either drive 1 or 3.
RDATA_N	J04	I	IN_T	V_{DD}	**Read Data** Raw serial input data stream read from the FDD.
STEP_N	G04	O	OD_{14}, $O_{14/14}$	V_{DD}	**Step** Issues pulses to the disk drive at a software programmable rate to move the head during a seek operation.
TRK0_N	H03	I	IN_T	V_{DD}	**Track 0** Indicates to the controller that the head of the selected floppy disk drive is at track 0.
WDATA_N	G02	O	OD_{14}, $O_{14/14}$	V_{DD}	**Write Data** Carries out the write pre-compensated serial data that is written to the selected floppy disk drive. Pre-compensation is software selectable.
WGATE_N	G01	O	OD_{14}, $O_{14/14}$	V_{DD}	**Write Gate** Enables the write circuitry of the selected disk drive. WGATE is designed to prevent glitches during power up and power down. This prevents writing to the disk when power is cycled.
WRPRT_N	H02	I	IN_T	V_{DD}	**Write Protected** Indicates that the disk in the selected drive is write protected. A software programmable configuration bit (FDC configuration at Index F0h, Logical Device 0) can force an active write-protect indication to the FDC regardless of the status of this pin.

Table 4.13 Keyboard and Mouse Controller (KBC)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
KBCLK	C12	I/O	IN_{TS}/OD_2	V_{DD}, V_{CC_IO}	**Keyboard Clock** Transfers the keyboard clock between the SuperI/O chip and the external keyboard using the PS/2 protocol. Driven by the internal, inverted KBC P26 signal, and connected internally to the T0 signal of the KBC. External pull-up resistor to 5V required (for PS/2 compliance). The pin is monitored for wake-up event detection. However, to enable this activity during power off, it must be pulled up to Keyboard and Mouse standby voltage.

Table 4.13 Keyboard and Mouse Controller (KBC) (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
KBDAT	A11	I/O	IN_{TS}/OD_2	V_{DD}, VCC_IO	**Keyboard Data** Transfers the keyboard data between the SuperI/O chip and the external keyboard using the PS/2 protocol. Driven by the internal, inverted KBC P27 signal, and connected internally to KBC P10. External pull-up resistor to 5V required (for PS/2 compliance). The pin is monitored for wake-up event detection. To enable this activity, it must be pulled up to Keyboard and Mouse standby voltage.
KBLOCK	B11	I	IN_{TS}	V_{DD}	**Keyboard Lock** P17 input.
MCLK_C	D11	I/O	IN_{TS}/OD_2	V_{DD}, VCC_IO	**Mouse Clock** Transfers the mouse clock between the SuperI/O chip and the external keyboard using the PS/2 protocol. Driven by the internal, inverted KBC P23 signal, and connected internally to KBC’s T1. External pull-up resistor to 5V required (for PS/2 compliance). The pin is monitored for wake-up event detection. To enable the activity, it must be pulled up to Keyboard and Mouse standby voltage.
MDAT	C11	I/O	IN_{TS}/OD_2	V_{DD}, VCC_IO	**Mouse Data** Transfers the mouse data between the SuperI/O chip and the external keyboard using the PS/2 protocol. This pin is driven by the internal, inverted KBC P22 signal, and is connected internally to KBC’s P11. External pull-up resistor to 5V is required (for PS/2 compliance). The pin is monitored for wake-up event detection. To enable the activity, it must be pulled up to Keyboard and Mouse standby voltage.

Table 4.14 Parallel Port

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
ACK_N	K02	I	IN_T	V_{DD}	**Acknowledge** Pulsed low by the printer to indicate that it has received data from the Parallel Port.
AFD_N	L02	O	OD_{14}, $O_{14/14}$	V_{DD}	**Automatic Feed - AFD** When low, instructs the printer to automatically feed a line after printing each line. This pin is in TRI-STATE after a 0 is loaded into the corresponding control register bit. An external 4.7 KΩ pull-up resistor should be attached to this pin. **Data Strobe (EPP) - DSTRB** Active low, used in EPP mode to denote a data cycle. When the cycle is aborted, $\overline{\text{DSTRB}}$ becomes inactive (high).
BUSY	K03	I	IN_T	V_{DD}	**Busy** Set high by the printer when it cannot accept another character. **Wait** In EPP mode, the Parallel Port device uses this active low signal to extend its access cycle.
ERR_N	L01	I	IN_T	V_{DD}	**Error** Set active low by the printer when it detects an error.
INIT	L03	O	OD_{14}, $O_{14/14}$	V_{DD}	**Initialize** When low, initializes the printer. This signal is in TRI-STATE after a 1 is loaded into the corresponding control register bit. Use an external 4.7 KΩ pull-up resistor.
PD[7:0]	0 = P03 1 = P01 2 = P02 3 = N01 4 = N02 5 = M01 6 = N03 7 = M02	I/O	IN_T/ OD_{14}, $O_{14/14}$	V_{DD}	**Parallel Port Data** Transfer data to and from the peripheral data bus and the appropriate Parallel Port data register. These signals have a high current drive capability.
PE	J01	I	IN_T	V_{DD}	**Paper End** Set high by the printer when it is out of paper. This pin has an internal weak pull-up or pull-down resistor.
SLCT	J03	I	IN_T	V_{DD}	**Select** Set active high by the printer when the printer is selected.

Table 4.14 Parallel Port (cont.)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
SLIN_N	K01	O	OD_{14}, $O_{14/14}$	V_{DD}	**Select Input - SLIN** When low, selects the printer. This signal is in TRI-STATE after a 0 is loaded into the corresponding control register bit. Uses an external 4.7 KΩ pull-up resistor. **Address Strobe (EPP) - ASTRB** Active low, used in EPP mode to denote an address or data cycle. When the cycle is aborted, $\overline{\text{ASTRB}}$ becomes inactive (high).
STB_N	M03	O	OD_{14}, $O_{14/14}$	V_{DD}	**Data Strobe - STB** When low, Indicates to the printer that valid data is available at the printer port. This signal is in TRI-STATE after a 0 is loaded into the corresponding control register bit. An external 4.7 KΩ pull-up resistor should be employed. **Write Strobe - WRITE** Active low, used in EPP mode to denote an address or data cycle. When the cycle is aborted, $\overline{\text{WRITE}}$ becomes inactive (high).

Table 4.15 Power and Ground

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
V_{BAT}	AD3		IN_{ULR}	-	**Battery Power Supply** Provides battery back-up to the System Wake-Up Control registers. The pin is connected to the internal logic through a series resistor for UL protection. **Note:** The ZFx86 contains no reverse polarity protection.
V_{DD}			PWR	-	**Main 3.3V Power Supply** See Pin description Table 8.4 on page 564.
GND			GND	-	**Ground** See Table 8.4 on page 564

Table 4.16 Serial Port 1 and Serial Port 2 (Shared with I/R Port)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
CTS1_N CTS2_N	1 = D09 2 = A06	I	IN_{TS}	V_{DD}	**Clear to Send** When low, indicates that the modem or other data transfer device is ready to exchange data.
DCD1_N DCD2_N	1 = A10 2 = B08	I	IN_{TS}	V_{DD}	**Data Carrier Detected** When low, indicate that the modem or other data transfer device has detected the data carrier.
DSR1_N DSR2_N	1 = C10 2 = C08	I	IN_{TS}	V_{DD}	**Data Set Ready** When low, indicate that the data transfer device, e.g., modem, is ready to establish a communications link.
DTR1_N, BOUT1 DTR2_N, BOUT2	C09 D07	O	$O_{8/8}$	V_{DD}	**Data Terminal Ready** When low, indicate to the modem or other data transfer device that the Serial Port is ready to establish a communications link. After system reset, these pins provides the $\overline{\text{DTR}}$ function, sets these signals to inactive high, and loopback operation holds them inactive. **Baud Output** Provides the associated serial channel baud rate generator output signal if test mode is selected, that is, bit 7 of the EXCR1 Register is set.
RI1_N RI2_N	A08 B06	I	IN_{TS}	V_{DD}, VCC_IO	**Ring Indicators (Modem)** When low, indicates that a telephone ring signal has been received by the modem. These pins may issue a wake-up event.
RTS1_N RTS2_N	A09 C07	O	$O_{8/8}$	V_{DD}	**Request to Send** When low, indicates to the modem or other data transfer device that the corresponding Serial Port is ready to exchange data. A system reset sets these signals to inactive high, and loopback operation holds them inactive.
RX1 RX2	B10 A07	I	IN_{TS}	V_{DD}	**Serial Input** Receive composite serial data from the communications link (peripheral device, modem, or other data transfer device).
TX1 TX2	B09 B07	O	$O_{8/8}$	V_{DD}	**Serial Output** Send composite serial data to the communications link (peripheral device, modem, or other data transfer device). The SOUT2,1 signals are set active high after system reset.

Table 4.17 Infrared Communication Port (Shared W/COM2)

Signal Name	Pin No. (PU/PD)	Type (Drive)	Buffer Type	Power Well	Description
IRRX1	C06	I	Generic 2	VDD_IO	**IR Reception 1** Primary input to receive serial data from the IR transceiver module.
IRSL0	C07	I/O	Generic 2	VDD_IO	**IR Select 0 - IRSL0** Input/Output to control the IR analog front end.
IRSL1	C08	I/O	Generic 2	VDD_IO	**IR Select 1 - IRSL1** Input/Output to control the IR analog front end.
IRSL2	B08	I/O	Generic 2	VDD_IO	**IR Select 2 - IRSL2** Input/Output to control the IR analog front end.
IRSL3	A06	I	Generic 2	VDD_IO	**IR Select 3 - IRSL3** Input to control the IR analog front end.
IRTX	A05	O	Generic 2	VDD_IO	**IR Transmit** IR serial output data.

4.4. Register Descriptions

The South Bridge module is a multi-function device. Its register space can be broadly divided into three categories in which specific types of registers are located:

- Chipset Register Space (F0-F3)
- USB Controller Register Space (PCIUSB)
- ISA Legacy Register Space (I/O Port)

The Chipset and USB Controller Register Spaces are accessed through the PCI interface using the PCI Type One Configuration Mechanism.

The **Chipset Register Space** of the South Bridge module is comprised of four separate functions; each with its own register space consisting of PCI header registers and configuration registers.

The PCI header is a 256-byte region used for configuring a PCI device or function. The first 64 bytes are the same for all PCI devices and are predefined by the PCI specification. Use these registers to configure the PCI for the device. Use the rest of the 256-byte region to configure the device or function itself.

The **USB Controller Register Space** consists of the standard PCI header registers. The USB controller supports three ports and is OpenHCI compliant.

The **ISA Legacy I/O Register Space** contains all the legacy compatibility I/O ports that are internal, trapped, shadowed, or snooped.

The remaining subsections of this chapter contains the following:

- A brief discussion on how to access the registers located in PCI Configuration Space.
- Register summaries
- Bit formats for all registers

4.4.1. PCI Configuration Space and Access Methods

Configuration cycles are generated from the North Bridge. All configuration registers in the South Bridge module are accessed through the PCI interface using the PCI Configuration Mechanism 1. This mechanism uses two DWORD I/O locations at 0CF8h and 0CFCh.

- 0CF8h references the Configuration Address Register.
- 0CFCh references the Configuration Data Register.

Note: The write to 0CF8h for configuration cycles *must* be a 32 bit IO operation.

To access PCI configuration space, write the Configuration Address (0CF8h) Register with data that specifies the South Bridge module as the device on PCI being accessed, along with the configuration register offset. On the following cycle, a read or write to the Configuration Data Register (CDR) causes a PCI configuration cycle to the South Bridge module. Byte, word, or double word accesses are allowed to the CDR at 0CFCh, 0CFDh, 0CFEh, or 0CFFh.

The South Bridge module has five PCI configuration register sets, one for each function (F0-F3) and USB (PCIUSB). Base Address Registers (BARx) in F0-F3 and PCIUSB set the base addresses for additional I/O or memory mapped configuration registers for each respective function.

Table 4.18 shows the PCI Configuration Address Register (0CF8h) and how to access the PCI header registers.

Table 4.18 PCI Configuration Address Register (0CF8h)

31	30 24	23 16	15 11	10 8	7 2	1 0
Configuration Space Mapping	**RSVD**	**Bus Number**	**Device Number**	**Function**	**Index**	**Doubleword 00**
1 (Enable)	000 000	0000 0000	xxxx x **(Note)**	xxx	xxxx xx	00 (Always)
Function 0 (F0): Bridge Configuration and GPIO Configuration Register Space						
80h		0000 0000	1001 0	000	Index	
Function 1 (F1): SMI Status and Power Management Timer Configuration Register Space						
80h		0000 0000	1001 0	001	Index	
Function 2 (F2): IDE Controller Configuration Register Space						
80h		0000 0000	1001 0	010	Index	
Function 3 (F3): XBus Expansion Register Space						
80h		0000 0000	1001 0	011	Index	
PCIUSB: USB Controller Register Configuration Space						
80h		0000 0000	1001 1	000	Index	

See "Example: Setting the ISA Bus Clock" on page 178.

4.4.2. Register Summaries

The tables in this subsection summarize all the registers of the South Bridge module. Included in the tables are the register's reset values and page references where the bit formats are found.

Table 4.19 F0: PCI Header/Bridge and GPIO Configuration Register Summary

F0 Index	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.30)
00h-01h	16	RO	Vendor Identification Register	1078h	Page 202
02h-03h	16	RO	Device Identification Register	0400h	Page 202
04h-05h	16	R/W	PCI Command Register	000Fh	Page 202
06h-07h	16	R/W	PCI Status Register	0280h	Page 202
08h	8	RO	Device Revision ID Register	00h	Page 203
09h-0Bh	24	RO	PCI Class Code Register	060100h	Page 203
0Ch	8	R/W	PCI Cache Line Size Register	00h	Page 203
0Dh	8	R/W	PCI Latency Timer Register	00h	Page 203
0Eh	8	RO	PCI Header Type Register	80h	Page 203
0Fh	8	RO	PCI BIST Register	00h	Page 203
10h-13h	32	R/W	**Base Address Register 0 (F0BAR0)** — Sets the base address for the I/O mapped GPIO Runtime and Configuration Registers (summarized in Table 4.20).	00000001h	Page 203
14h-2Bh	--	--	Reserved	--	
2Ch-2Dh	16	RO	Subsystem Vendor ID	1078h	Page 204
2Eh-2Fh	16	RO	Subsystem ID	0400h	Page 204
30h-3Fh	--	--	Reserved	--	
40h	8	R/W	PCI Function Control Register 1	39h	Page 204
41h	8	R/W	PCI Function Control Register 2	00h	Page 204
42h	--	--	Reserved	--	
43h	8	R/W	PIT Delayed Transactions Register	02h	Page 205
44h	8	R/W	Reset Control Register	01h	Page 205
45h	--	--	Reserved	--	
46h	8	R/W	PCI Functions Enable Register	FEh	Page 205
47h	8	R/W	Miscellaneous Enable Register	00h	Page 205
48h-4Bh	--	--	Reserved	--	
4Ch-4Fh	32	R/W	Top of System Memory	FFFFFFFFh	Page 206
50h	8	R/W	PIT Control/ISA CLK Divider	7Bh	Page 206
51h	8	R/W	ISA I/O Recovery Control Register	40h	Page 206
52h	8	R/W	ROM/AT Logic Control Register	98h	Page 206
53h	8	R/W	Alternate CPU Support Register	00h	Page 207
54h-59h	--	--	Reserved	--	
5Ah	8	R/W	Decode Control Register 1	01h	Page 207
5Bh	8	R/W	Decode Control Register 2	20h	Page 208
5Ch	8	R/W	PCI Interrupt Steering Register 1	00h	Page 208
5Dh	8	R/W	PCI Interrupt Steering Register 2	00h	Page 209
5Eh-6Dh	--	--	Reserved	--	
6Eh-6Fh	16	R/W	ROM Mask Register	FFF0h	Page 209
70h-71h	16	R/W	IOCS1# Base Address Register	0000h	Page 209

Table 4.19 F0: PCI Header/Bridge and GPIO Configuration Register Summary (cont.)

F0 Index	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.30)
72h	8	R/W	IOCS1# Control Register	00h	Page 209
73h	--	--	Reserved	--	
74h-75h	16	R/W	IOCS0# Base Address Register	0000h	Page 210
76h	8	R/W	IOCS0# Control Register	00h	Page 210
77h	--	--	Reserved	--	
78h-7Bh	32	R/W	DOCCS# Base Address Register	00000000h	Page 210
7Ch-7Fh	32	R/W	DOCCS# Control Register	00000000h	Page 210
80h	8	R/W	Power Management Enable Register 1	00h	Page 211
81h	8	R/W	Power Management Enable Register 2	00h	Page 211
82h	8	R/W	Power Management Enable Register 3	00h	Page 212
83h	8	R/W	Power Management Enable Register 4	00h	Page 213
84h	--	--	Reserved	--	
85h	8	RO	Second Level PME/SMI Status Mirror Register 2	00h	Page 214
86h	8	RO	Second Level PME/SMI Status Mirror Register 3	00h	Page 214
87h	8	RO	Second Level PME/SMI Status Mirror Register 4	00h	Page 215
88h	8	R/W	General Purpose Timer 1 Count Register	00h	Page 215
89h	8	R/W	General Purpose Timer 1 Control Register	00h	Page 215
8Ah	8	R/W	General Purpose Timer 2 Count Register	00h	Page 216
8Bh	8	R/W	General Purpose Timer 2 Control Register	00h	Page 216
8Ch	8	R/W	IRQ Speedup Timer Count Register	00h	Page 217
8Dh-92h	--	--	Reserved	--	
93h	8	R/W	Miscellaneous Device Control Register	00h	Page 217
94h	8	R/W	Suspend Modulation OFF Count Register	00h	Page 217
95h	8	R/W	Suspend Modulation ON Count Register	00h	Page 217
96h	8	R/W	Suspend Configuration Register	00h	Page 218
97h	--	--	Reserved	--	
98h-99h	16	R/W	Hard Disk Idle Timer Count Register — Primary Channel	0000h	Page 218
9Ah-9Bh	16	R/W	Floppy Disk Idle Timer Count Register	0000h	Page 218
9Ch-9Dh	16	R/W	Parallel / Serial Idle Timer Count Register	0000h	Page 218
9Eh-9Fh	16	R/W	Keyboard / Mouse Idle Timer Count Register	0000h	Page 219
A0h-A1h	16	R/W	User Defined Device 1 Idle Timer Count Register	0000h	Page 219
A2h-A3h	16	R/W	User Defined Device 2 Idle Timer Count Register	0000h	Page 219
A4h-A5h	16	R/W	User Defined Device 3 Idle Timer Count Register	0000h	Page 219
A6h-ABh	--	--	Reserved	--	
ACh-ADh	16	R/W	Hard Disk Idle Timer Count Register — Secondary Channel	0000h	Page 219
AEh	8	WO	CPU Suspend Command Register	00h	Page 220
AFh	8	WO	Suspend Notebook Command Register	00h	Page 220
B0h-B7h	--	--	Reserved	--	
B8h	8	RO	DMA Shadow Register	xxh	Page 220
B9h	8	RO	PIC Shadow Register	xxh	Page 220
BAh	8	RO	PIT Shadow Register	xxh	Page 221
BBh	8	RO	RTC Index Shadow Register	xxh	Page 221
BCh	8	R/W	Clock Stop Control Register	00h	Page 221

Table 4.19 F0: PCI Header/Bridge and GPIO Configuration Register Summary (cont.)

F0 Index	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.30)
BDh-BFh	--	--	Reserved	--	
C0h-C3h	32	R/W	User Defined Device 1 Base Address Register	00000000h	Page 221
C4h-C7h	32	R/W	User Defined Device 2 Base Address Register	00000000h	Page 221
C8h-CBh	32	R/W	User Defined Device 3 Base Address Register	00000000h	Page 222
CCh	8	R/W	User Defined Device 1 Control Register	00h	Page 222
CDh	8	R/W	User Defined Device 2 Control Register	00h	Page 222
CEh	8	R/W	User Defined Device 3 Control Register	00h	Page 222
CFh	--	--	Reserved	--	
D0h	8	WO	Software SMI Register	00h	Page 223
D1h-EBh	--	--	Reserved	--	
ECh	8	R/W	Timer Test Register	00h	Page 223
EDh-F4h	--	--	Reserved	--	
F5h	8	RC	Second Level PME/SMI Status Register 2	00h	Page 223
F6h	8	RC	Second Level PME/SMI Status Register 3	00h	Page 223
F7h	8	RC	Second Level PME/SMI Status Register 4	00h	Page 224
F8h-FFh	--	--	Reserved	--	

Table 4.20 F0BAR0: GPIO Support Registers Summary

F0BAR0+ I/O Offset	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.31)
00h	8	R/W	GPIO Data Out 0 Register	FFh	Page 225
01h-03h	--	--	Reserved	--	
04h	8	RO	GPIO Data In 0 Register	FFh	Page 225
05h-07h	--	--	Reserved	--	
08h	8	R/W	GPIO Interrupt Enable 0 Register	00h	Page 225
09h-0Bh	--	--	Reserved	--	
0Ch	8	R/W1C	GPIO Status 0 Register	00h	Page 225
0Dh-1Fh	--	--	Reserved	--	
20h	8	R/W	GPIO Pin Configuration Select Register	00h	Page 225
21h-23h	--	--	Reserved	--	
24h	8	R/W	GPIO Pin Configuration Access Register	44h	Page 226
25h-27h	--	--	Reserved	--	
28h	8	R/W	GPIO Reset Control Register	00h	Page 226
29h-2Bh	--	--	Reserved	--	

Table 4.21 F1: PCI Header Registers for SMI Status Summary

F1 Index	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.32)
00h-01h	16	RO	Vendor Identification Register	1078h	Page 227
02h-03h	16	RO	Device Identification Register	0401h	Page 227
04h-05h	16	R/W	PCI Command Register	0000h	Page 227
06h-07h	16	RO	PCI Status Register	0280h	Page 227
08h	8	RO	Device Revision ID Register	00h	Page 227
09h-0Bh	24	RO	PCI Class Code Register	000000h	Page 227
0Ch	8	RO	PCI Cache Line Size Register	00h	Page 227
0Dh	8	RO	PCI Latency Timer Register	00h	Page 227
0Eh	8	RO	PCI Header Type Register	00h	Page 227
0Fh	8	RO	PCI BIST Register	00h	Page 227
10h-13h	32	R/W	**Base Address Register 0 (F1BAR0)** — Sets the base address for the I/O mapped SMI Status Registers (summarized in Table 4.22).	00000001h	Page 227
14h-2Bh	--	--	Reserved	--	
2Ch-2Dh	16	RO	Subsystem Vendor ID	1078h	Page 227
2Eh-2Fh	16	RO	Subsystem ID	0401h	Page 227
30h-FFh	--	--	Reserved	--	

Table 4.22 F1BAR0: SMI Status Registers Summary

F1BAR0+ I/O Offset	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.33)
00h-01h	16	RO	Top Level PME/SMI Status Mirror Register	0000h	Page 228
02h-03h	16	RC	Top Level PME/SMI Status Register	0000h	Page 228
04h-05h	16	RO	Second Level General Traps & Timers PME/SMI Status Mirror Register	0000h	Page 229
06h-07h	16	RC	Second Level General Traps & Timers PME/SMI Status Register	0000h	Page 230
08h-09h	16	Read to Enable	SMI Speedup Disable Register	0000h	Page 230
0Ah-4Fh	--	--	Reserved	--	
50h-FFh	--	--	The I/O mapped registers located here (F1BAR0+Offset 50h-FFh) are also accessible at F0 Index 50h-FFh. The preferred method is to program these registers through the F0 register space.		

Table 4.23 F2: PCI Header Registers for IDE Controller Support Summary

F2 Index	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.34)
00h-01h	16	RO	Vendor Identification Register	1078h	Page 231
02h-03h	16	RO	Device Identification Register	0402h	Page 231
04h-05h	16	R/W	PCI Command Register	0000h	Page 231
06h-07h	16	RO	PCI Status Register	0280h	Page 231
08h	8	RO	Device Revision ID Register	01h	Page 231
09h-0Bh	24	RO	PCI Class Code Register	010180h	Page 231
0Ch	8	RO	PCI Cache Line Size Register	00h	Page 231
0Dh	8	RO	PCI Latency Timer Register	00h	Page 231
0Eh	8	RO	PCI Header Type Register	00h	Page 231
0Fh	8	RO	PCI BIST Register	00h	Page 231
10h-13h	32	RO	**Base Address Register 0 (F2BAR0)** — Reserved for possible future use by the South Bridge module.	00000000h	Page 231
14h-17h	32	RO	**Base Address Register 1 (F2BAR1)** — Reserved for possible future use by the South Bridge module.	00000000h	Page 231
18h-1Bh	32	RO	**Base Address Register 2 (F2BAR2)** — Reserved for possible future use by the South Bridge module.	00000000h	Page 231
1Ch-1Fh	32	RO	**Base Address Register 3 (F2BAR3)** — Reserved for possible future use by the South Bridge module.	00000000h	Page 232
20h-23h	32	R/W	**Base Address Register 4 (F2BAR4)** — Sets the base address for the I/O mapped Bus Master IDE Registers (summarized in Table 4.25)	00000001h	Page 232
24h-2Bh	--	--	Reserved	--	Page 232
2Ch-2Dh	16	RO	Subsystem Vendor ID	1078h	Page 232
2Eh-2Fh	16	RO	Subsystem ID	0402h	Page 232
30h-3Fh	--	--	Reserved	--	Page 232
40h-43h	32	R/W	Channel 0 Drive 0 PIO Register	00009172h	Page 232
44h-47h	32	R/W	Channel 0 Drive 0 DMA Control Register	00077771h	Page 233
48h-4Bh	32	R/W	Channel 0 Drive 1 PIO Register	00009172h	Page 233
4Ch-4Fh	32	R/W	Channel 0 Drive 1 DMA Control Register	00077771h	Page 233
50h-53h	32	R/W	Channel 1 Drive 0 PIO Register	00009172h	Page 233
54h-57h	32	R/W	Channel 1 Drive 0 DMA Control Register	00077771h	Page 233
58h-5Bh	32	R/W	Channel 1 Drive 1 PIO Register	00009172h	Page 233
5Ch-5Fh	32	R/W	Channel 1 Drive 1 DMA Control Register	00077771h	Page 234
60h-FFh	--	--	Reserved	--	

Table 4.24 IDE Controller Configuration Summary

F2BAR4+ I/O Offset	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.35)
00h	8	R/W	IDE Bus Master 0 Command Register — Primary	00h	Page 234
01h	--	--	Not Used	--	
02h	8	R/W	IDE Bus Master 0 Status Register — Primary	00h	Page 234
03h	--	--	Not Used	--	
04h-07h	32	R/W	IDE Bus Master 0 PRD Table Address — Primary	00000000h	Page 235
08h	8	R/W	IDE Bus Master 1 Command Register — Secondary	00h	Page 235
09h	--	--	Not Used	--	
0Ah	8	R/W	IDE Bus Master 1 Status Register — Secondary	00h	Page 235
0Bh	--	--	Not Used	--	
0Ch-0Fh	32	R/W	IDE Bus Master 1 PRD Table Address — Secondary	00000000h	Page 235

Table 4.25 F3: PCI Header Registers for XBus Expansion Summary

F3 Index	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.36)
00h-01h	16	RO	Vendor Identification Register	1078h	Page 236
02h-03h	16	RO	Device Identification Register	0403h	Page 236
04h-05h	16	R/W	PCI Command Register	0000h	Page 236
06h-07h	16	RO	PCI Status Register	0280h	Page 236
08h	8	RO	Device Revision ID Register	00h	Page 236
09h-0Bh	24	RO	PCI Class Code Register	000000h	Page 236
0Ch	8	RO	PCI Cache Line Size Register	00h	Page 236
0Dh	8	RO	PCI Latency Timer Register	00h	Page 236
0Eh	8	RO	PCI Header Type Register	00h	Page 236
0Fh	8	RO	PCI BIST Register	00h	Page 236
10h-13h	32	R/W	**Base Address Register 0 (F3BAR0)** — Sets the base address for the XBus Expansion support registers (summarized in Table 4.26).	00000000h	Page 236
14h-17h	32	R/W	**Base Address Register 1 (F3BAR1)** — Reserved for possible future use by the South Bridge module.	00000000h	Page 237
18h-1Bh	32	R/W	**Base Address Register 2 (F3BAR2)** — Reserved for possible future use by the South Bridge module.	00000000h	Page 237
1Ch-1Fh	32	R/W	**Base Address Register 3 (F3BAR3)** — Reserved for possible future use by the South Bridge module.	00000000h	Page 237
20h-23h	32	R/W	**Base Address Register 4 (F3BAR4)** — Reserved for possible future use by the South Bridge module.	00000000h	Page 237
24h-27h	32	R/W	**Base Address Register 5 (F3BAR5)** — Reserved for possible future use by the South Bridge module.	00000000h	Page 237
28h-2Bh	--	--	Reserved	--	
2Ch-2Dh	16	RO	Subsystem Vendor ID	1078h	Page 237
2Eh-2Fh	16	RO	Subsystem ID	0405h	Page 237
30h-3Fh	--	--	Reserved	--	

Table 4.25 F3: PCI Header Registers for XBus Expansion Summary (cont.)

F3 Index	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.36)
40h-43h	32	R/W	F3BAR0 Base Address Register Mask	00000000h	Page 237
44h-47h	32	R/W	F3BAR1 Base Address Register Mask	00000000h	Page 238
48h-4Bh	32	R/W	F3BAR2 Base Address Register Mask	00000000h	Page 238
4Ch-4Fh	32	R/W	F3BAR3 Base Address Register Mask	00000000h	Page 238
50h-53h	32	R/W	F3BAR4 Base Address Register Mask	00000000h	Page 238
54h-57h	32	R/W	F3BAR5 Base Address Register Mask	00000000h	Page 238
58h	8	R/W	F3BARx Initialized Register	00h	Page 238
58h-FFh	--	--	Reserved	--	

Table 4.26 F3BAR0: XBus Expansion Registers Summary

F3BAR0+ I/O Offset	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.37)
00h-03h	32	R/W	I/O Control Register 1	010C0007h	Page 239
04h-07h	32	R/W	I/O Control Register 2	00000000h	Page 240
08h-0Bh	32	R/W	I/O Control Register 3	00009000h	Page 240

Table 4.27 PCIUSB: USB Controller Register Summary

PCIUSB Index	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.38)
00h-01h	16	RO	Vendor Identification	0E11h	Page 241
02h-03h	16	RO	Device Identification	A0F8h	Page 241
04h-05h	16	R/W	Command Register	00h	Page 241
06h-07h	16	R/W	Status Register	0280h	Page 241
08h	8	RO	Device Revision ID	07h	Page 242
09h-0Bh	24	RO	Class Code	0C0310h	Page 242
0Ch	8	R/W	Cache Line Size	00h	Page 242
0Dh	8	R/W	Latency Timer	00h	Page 242
0Eh	8	RO	Header Type	00h	Page 242
0Fh	8	RO	BIST Register	00h	Page 242
10h-13h	32	R/W	Base Address 0	00000000h	Page 242
14h-2Bh	--	--	Reserved	--	
2Ch-2Dh	16	R/W	Subsystem Vendor ID	0E11h	Page 242
2Eh-2Fh	16	R/W	Subsystem ID	A0F8h	Page 242
30h-3Bh	--	--	Reserved	--	
3Ch	8	R/W	Interrupt Line Register	00h	Page 242
3Dh	8	RO	Interrupt Pin Register	01h	Page 243
3Eh	8	RO	Min. Grant Register	00h	Page 243
3Fh	8	RO	Max. Latency Register	50h	Page 243

Table 4.27 PCIUSB: USB Controller Register Summary (cont.)

PCIUSB Index	Width (Bits)	Type	Name	Reset Value	Reference (Table 4.38)
40h-43h	32	R/W	ASIC Test Mode Enable Register	000F0000h	Page 243
44h	8	R/W	ASIC Operational Mode Enable	00h	Page 243
45h-FFh	--	--	Reserved	--	

Table 4.28 ZF-Logic Register Summary

ISA Index	Width (Bits)	Type	Name	Reset Value	Reference
218-21Ch	40	--	Reserved for ZF-Logic - See 'ZF-Logic and Clocking' on page 397.	--	Page 397

Table 4.29 Legacy I/O Register Summary

I/O Port	Type	Name	Reference
DMA Channel Control Registers (Table 4.39)			
000h	R/W	DMA Channel 0 Address Register	Page 243
001h	R/W	DMA Channel 0 Transfer Count Register	Page 243
002h	R/W	DMA Channel 1 Address Register	Page 244
003h	R/W	DMA Channel 1 Transfer Count Register	Page 244
004h	R/W	DMA Channel 2 Address Register	Page 244
005h	R/W	DMA Channel 2 Transfer Count Register	Page 244
006h	R/W	DMA Channel 3 Address Register	Page 244
007h	R/W	DMA Channel 3 Transfer Count Register	Page 244
008h	Read	DMA Status Register, Channels 3:0	Page 244
	Write	DMA Command Register, Channels 3:0	Page 244
009h	WO	Software DMA Request Register, Channels 3:0	Page 244
00Ah	R/W	DMA Channel Mask Register, Channels 3:0	Page 244
00Bh	WO	DMA Channel Mode Register, Channels 3:0	Page 245
00Ch	WO	DMA Clear Byte Pointer Command, Channels 3:0	Page 245
00Dh	WO	DMA Master Clear Command, Channels 3:0	Page 245
00Eh	WO	DMA Clear Mask Register Command, Channels 3:0	Page 245
00Fh	WO	DMA Write Mask Register Command, Channels 3:0	Page 245
0C0h	R/W	DMA Channel 4 Address Register (Not used)	Page 245
0C2h	R/W	DMA Channel 4 Transfer Count Register (Not Used)	Page 245
0C4h	R/W	DMA Channel 5 Address Register	Page 245
0C6h	R/W	DMA Channel 5 Transfer Count Register	Page 245
0C8h	R/W	DMA Channel 6 Address Register	Page 245
0CAh	R/W	DMA Channel 6 Transfer Count Register	Page 245
0CCh	R/W	DMA Channel 7 Address Register	Page 245
0CEh	R/W	DMA Channel 7 Transfer Count Register	Page 245

Table 4.29 Legacy I/O Register Summary (cont.)

I/O Port	Type	Name	Reference
0D0h	Read	DMA Status Register, Channels 7:4	Page 246
	Write	DMA Command Register, Channels 7:4	Page 246
0D2h	WO	Software DMA Request Register, Channels 7:4	Page 246
0D4h	R/W	DMA Channel Mask Register, Channels 7:0	Page 246
0D6h	WO	DMA Channel Mode Register, Channels 7:4	Page 246
0D8h	WO	DMA Clear Byte Pointer Command, Channels 7:4	Page 246
0DAh	WO	DMA Master Clear Command, Channels 7:4	Page 246
0DCh	WO	DMA Clear Mask Register Command, Channels 7:4	Page 246
0DEh	WO	DMA Write Mask Register Command, Channels 7:4	Page 246
DMA Page Registers (Table 4.40)			
081h	R/W	DMA Channel 2 Low Page Register	Page 247
082h	R/W	DMA Channel 3 Low Page Register	Page 247
083h	R/W	DMA Channel 1 Low Page Register	Page 247
087h	R/W	DMA Channel 0 Low Page Register	Page 247
089h	R/W	DMA Channel 6 Low Page Register	Page 247
08Ah	R/W	DMA Channel 7 Low Page Register	Page 247
08Bh	R/W	DMA Channel 5 Low Page Register	Page 247
08Fh	R/W	ISA Refresh Low Page Register	Page 247
481h	R/W	DMA Channel 2 High Page Register	Page 247
482h	R/W	DMA Channel 3 High Page Register	Page 247
483h	R/W	DMA Channel 1 High Page Register	Page 247
487h	R/W	DMA Channel 0 High Page Register	Page 247
489h	R/W	DMA Channel 6 High Page Register	Page 247
48Ah	R/W	DMA Channel 7 High Page Register	Page 248
48Bh	R/W	DMA Channel 5 High Page Register	Page 248
Programmable Interval Timer Registers (Table 4.41)			
040h	Write	PIT Timer 0 Counter	Page 248
	Read	PIT Timer 0 Status	Page 248
041h	Write	PIT Timer 1 Counter (Refresh)	Page 248
	Read	PIT Timer 1 Status (Refresh)	Page 248
042h	Write	PIT Timer 2 Counter (Speaker)	Page 248
	Read	PIT Timer 2 Status (Speaker)	Page 248
043h	R/W	PIT Mode Control Word Register	Page 249
Programmable Interrupt Controller Registers (Table 4.42)			
020h / 0A0h	WO	Master / Slave PCI IWC1	Page 249
021h / 0A1h	WO	Master / Slave PIC ICW2	Page 249
021h / 0A1h	WO	Master / Slave PIC ICW3	Page 249
021h / 0A1h	WO	Master / Slave PIC ICW4	Page 249
021h / 0A1h	R/W	Master / Slave PIC OCW1	Page 250
020h / 0A0h	WO	Master / Slave PIC OCW2	Page 250
020h / 0A0h	WO	Master / Slave PIC OCW3	Page 250
020h / 0A0h	RO	Master / Slave PIC Interrupt Request and Service Registers for OCW3 Commands	Page 250

Table 4.29 Legacy I/O Register Summary (cont.)

I/O Port	Type	Name	Reference
Keyboard Controller Registers (Table 4.43)			
060h	R/W	External Keyboard Controller Data Register	Page 251
061h	R/W	Port B Control Register	Page 251
062h	R/W	External Keyboard Controller Mailbox Register	Page 251
064h	R/W	External Keyboard Controller Command Register	Page 251
066h	R/W	External Keyboard Controller Mailbox Register	Page 252
092h	R/W	Port A Control Register	Page 252
Real Time Clock Registers (Table 4.44)			
070h	WO	RTC Address Register	Page 252
071h	R/W	RTC Data Register	Page 252
Miscellaneous Registers (Table 4.45)			
0F0h, 0F1h	WO	Coprocessor Error Register	Page 252
170h-177h/ 376h-377h	R/W	Secondary IDE Registers	Page 252
1F0-1F7h/ 3F6h-3F7h	R/W	Primary IDE Registers	Page 252
4D0h	R/W	Interrupt Edge/Level Select Register 1	Page 252
4D1h	R/W	Interrupt Edge/Level Select Register 2	Page 253

4.4.3. Chipset Register Space

The Chipset Register Space of the South Bridge module is comprised of four separate functions (F0-F3), each with its own register space. Base Address Registers (BARs) in each PCI header register space set the base address for the configuration registers for each respective function. The configuration registers accessed through BARs are I/O or memory mapped. The PCI header registers in all functions are very similar.

- Function 0 (F0): PCI Header/Bridge Configuration Registers and GPIO Support
- Function 1 (F1): PCI Header Registers for SMI Status
- Function 2 (F2): PCI Header/Channel 0 and 1 Configuration Registers for IDE Controller Support
- Function 3 (F3): PCI Header Registers for XBus Expansion.
  F3 contains six BARs in their standard PCI header locations (that is, Index 10h, 14h, 18h, 1Ch, 20, and 24h). In addition there are six mask registers that allow the six BARs to be fully programmed; I/O versus memory space and size:
  --from 4 GB to 16 bytes for memory and
  --from 4 GB to 4 bytes for I/O.

4.4.3.1. Bridge, GPIO Registers - Function 0

The register space designated as Function 0 (F0) is used to configure Bridge features and functionality unique to the South Bridge module. In addition, it configures the PCI portion of support hardware for the GPIO support registers. Table 4.30 defines the bit formats for the PCI Header Registers and Bridge Configuration.

Note: The registers at F0 Index 50h-FFh can also be accessed at F1BAR0+I/O Offset 50h-FFh. However, the preferred method is to program these registers through the F0 register space.

Located in the PCI Header Registers of F0 is Base Address Register (F0BAR0) used for pointing to the register spaces designated for GPIO configuration, described in Section 4.4.3.2.

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers

Bit	Description
Index 00h-01h	**Vendor Identification Register (RO)** — **Reset Value = 1078h**
Index 02h-03h	**Device Identification Register (RO)** — **Reset Value = 0400h**
Index 04h-05h	**PCI Command Register (R/W)** — **Reset Value = 000Fh**
15:10	**Reserved** — Set to 0.
9	**Fast Back-to-Back Enable (Read Only)** — Always reads 0.
8	**SERR#** — Allow SERR# assertion on detection of special errors: 0 = Disable **(Default)**; 1 = Enable.
7	**Wait Cycle Control (Read Only)** — Always reads 0.
6	**Parity Error** — Allow the South Bridge module to check for parity errors on PCI cycles for which it is a target and to assert PERR# when a parity error is detected: 0 = Disable **(Default)**; 1 = Enable.
5	**VGA Palette Snoop Enable (Read Only)** — Always disabled. Reads 0.
4	**Memory Write and Invalidate** — Allow the South Bridge module to do memory write and invalidate cycles, if the PCI Cache Line Register (F0 Index 0Ch) is set to 32 bytes (08h). 0 = Disable **(Default)**; 1 = Enable.
3	**Special Cycles** — Allow the South Bridge module to respond to special cycles: 0 = Disable; 1 = Enable **(Default)**. This bit must be enabled to allow the CPU Warm Reset signal to be triggered from a CPU Shutdown cycle.
2	**Bus Master** — Allow the South Bridge module bus mastering capabilities: 0 = Disable; 1 = Enable **(Default)**. This bit must be set to 1.
1	**Memory Space** — Allow the South Bridge module to respond to memory cycles from the PCI bus: 0 = Disable; 1 = Enable **(Default)**.
0	**I/O Space** —Allow the South Bridge module to respond to I/O cycles from the PCI bus: 0 = Disable; 1 = Enable **(Default)**. This bit must be enabled to access I/O offsets through F0BAR0 and F0BAR1 (see F0 Index 10h and 14h).
Index 06h-07h	**PCI Status Register (R/W)** — **Reset Value = 0280h**
15	**Detected Parity Error** — This bit is set whenever a parity error is detected. Write 1 to clear.
14	**Signaled System Error** — This bit is set whenever the South Bridge module asserts SERR# active. Write 1 to clear.
13	**Received Master Abort** — This bit is set whenever a master abort cycle occurs. A master abort will occur when a PCI cycle is not claimed, except for special cycles. Write 1 to clear.
12	**Received Target Abort** — This bit is set whenever a target abort is received while the South Bridge module is the master for the PCI cycle. Write 1 to clear.
11	**Signaled Target Abort** — This bit is set whenever the South Bridge module signals a target abort. This occurs when an address parity error occurs for an address that hits in the active address decode space of the South Bridge module. Write 1 to clear.
10:9	**DEVSEL# Timing (Read Only)** — These bits are always 01, as the South Bridge module will always respond to cycles for which it is an active target with medium DEVSEL# timing. 00 = Fast; 01 = Medium; 10 = Slow; 11 = Reserved.

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
8	**Data Parity Detected** —This bit is set when: 1) The South Bridge module asserted PERR# or observed PERR# asserted. 2) The South Bridge module is the master for the cycle in which the PERR# occurred and the Parity Error bit is set (F0 Index 04h[6] = 1). Write 1 to clear.
7	**Fast Back-to-Back Capable (Read Only)** — As a target, the South Bridge module is capable of accepting fast back-to-back transactions: 0 = Disable; 1 = Enable. This bit is always 1.
6:0	**Reserved**
Index 08h	**Device Revision ID Register (RO)** — **Reset Value = 00h**
Index 09h-0Bh	**PCI Class Code Register (RO)** — **Reset Value = 060100h**
Index 0Ch	**PCI Cache Line Size Register (R/W)** — **Reset Value = 00h**
7:0	**PCI Cache Line Size Register** — This register sets the size of the PCI cache line, in increments of four bytes. For memory write and invalidate cycles, the PCI cache line size must be set to 16 bytes (04h) and the Memory Write and Invalidate bit must be set (F0 Index 04h[4] = 1).
Index 0Dh	**PCI Latency Timer Register (R/W)** — **Reset Value = 00h**
7:4	**Reserved**
3:0	**PCI Latency Timer Value** — The PCI Latency Timer Register prevents system lockup when a slave does not respond to a cycle that the South Bridge module masters. If the value is set to 00h (**default**), the timer is disabled. If the timer is written with any other value, bits [3:0] become the four most significant bytes in a timer that counts PCI clocks for slave response. The timer is reset on each valid data transfer. If the counter expires before the next assertion of TRDY# is received, the South Bridge module stops the transaction with a master abort and asserts SERR#, if enabled to do so (F0 Index 04h[8] = 1).
Index 0Eh	**PCI Header Type (RO)** — **Reset Value = 80h**
7:0	**PCI Header Type Register** — This register defines the format of this header. This header is of type format 0. Additionally, bit 7 defines whether this PCI device is a multifunction device (bit 7 = 1) or not (bit 7 = 0).
Index 0Fh	**PCI BIST Register (RO)** — **Reset Value = 00h**
7	**BIST Capable (Read Only)** — Is device capable of running a built-in self-test (BIST)? 0 = No; 1 = Yes,
6	**Reserved**
5:4	**Reserved**
3:0	**BIST Completion Code (Read Only)** — Upon completion of the BIST, the completion code will be stored in these bits. A completion code of zero indicates the BIST has successfully been completed. All other values indicate some type of BIST failure.
Index 10h-13h	**Base Address Register 0 - F0BAR0 (R/W)** — **Reset Value = 00000001h**
	This register allows access to I/O mapped GPIO runtime and configuration registers. Bits [5:0] are read only (000001), indicating a 64-byte aligned I/O address space. Table 4.31 gives the bit formats and reset values of the GPIO registers.
31:6	**GPIO Base Address**
5:0	**Address Range (Read Only)**
Index 14h-17h	**Reserved**

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 18h-2Bh	**Reserved**
Index 2Ch-2Dh	**Subsystem Vendor ID (RO)** **Reset Value = 1078h**
Index 2Eh-2Fh	**Subsystem ID (RO)** **Reset Value = 0400h**
Index 30h-3Fh	**Reserved**
Index 40h	**PCI Function Control Register 1 (R/W)** **Reset Value = 39h**
7:5	Reserved
4	**PCI Subtractive Decode** — Allow the South Bridge module to act as a subtractive decode agent on the Front-PCI bus: 0 = Disable; 1 = Enable.
3:2	**Reserved**
1	**PERR# Signals SERR#** — Assert SERR# any time that PERR# is asserted or detected active by the South Bridge module (allows PERR# assertion to be cascaded to NMI (SMI) generation in the system): 0 = Disable; 1 = Enable.
0	**PCI Interrupt Acknowledge Cycle Response** — Allow the South Bridge module to respond to PCI interrupt acknowledge cycles: 0 = Disable; 1 = Enable.
Index 41h	**PCI Function Control Register 2 (R/W)** **Reset Value = 00h**
7:4	Reserved
3	**XBus Configuration Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs to one of the configuration registers in PCI Function 3 (F3) register space, an SMI is generated. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[9]. Second level SMI status is reported at F1BAR0+I/O Offset 04h/06h[5].
2	**IDE Configuration Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs to one of the configuration registers in PCI Function 2 (F2) register space, an SMI is generated. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[9]. Second level SMI status is reported at F1BAR0+I/O Offset 04h/06h[5].
1	**Power Management Configuration Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs to one of the configuration registers in PCI Function 1 (F1) register space, an SMI is generated. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[9]. Second level SMI status is reported at F1BAR0+I/O Offset 04h/06h[5].
0	**Legacy Configuration SMI** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs to one of the configuration registers in the ISA Legacy I/O register space, an SMI is generated. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[9]. Second level SMI status is reported at F1BAR0+I/O Offset 04h/06h[5]. **Note:** It is not recommended
Index 42h	**Reserved**

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 43h	**PIT Delayed Transactions Register (R/W)** **Reset Value = 02h**
7:2	**Reserved** — Set to 0.
1	**Enable PCI Delayed Transactions for AT legacy PIT I/O Addresses —** Some x86 programs (certain benchmarks/diagnostics) assume a particular latency for PIT accesses; this bit will allow that code to work. 0 = PIT I/O addresses complete as fast as possible on PCI. 1 = Accesses to PIT I/O addresses are delayed transactions on PCI. **(Default)** For best performance when running Windows™ this bit should be set to 0.
0	**Reserved**
Index 44h	**Reset Control Register (R/W)** **Reset Value = 01h**
7:4	**Reserved**
3	**IDE Controller Reset** — Reset the IDE controller: 0 = Disable; 1 = Enable. Write 0 to clear. This bit is level-sensitive and must be cleared after the reset is enabled.
2	**IDE Reset** — Reset IDE bus: 0 = Disable; 1 = Enable (will drive outputs to zero). Write 0 to clear. This bit is level-sensitive and must be cleared after the reset is enabled.
1	**PCI Reset** — Reset PCI bus: 0 = Disable; 1 = Enable. When set, the South Bridge module PCI_RST# output pin is asserted and all devices on the PCI bus including PCIUSB reset. No other function within the South Bridge module is affected by this bit. Write 0 to clear. This bit is level-sensitive and must be cleared after the reset is enabled.
0	**XBus Warm Start** — Reading/writing this bit has two different meanings/functions: Read: Has a warm start occurred since power-up? 0 = Yes; 1 = No Write: 0 = NOP; 1 = Execute system wide reset
Index 45h	**Reserved**
Index 46h	**PCI Functions Enable Register (R/W)** **Reset Value = FEh**
7:4	**Reserved**
3	**F3 (PCI Function 3)** —F3 register space: 0 = Disable; 1 = Enable (**Default**). This bit must always be set to 1.
2	**F2 (PCI Function 2)** — F2 register space: 0 = Disable; 1 = Enable (**Default**). This bit must always be set to 1.
1	**F1 (PCI Function 1)** — F1 register space: 0 = Disable; 1 = Enable (**Default**). This bit must always be set to 1.
0	**Reserved**
Index 47h	**Miscellaneous Enable Register (R/W)** **Reset Value = 00h**
7:2	**Reserved**
1	**F0BAR0 (Function 0, Base Address Register 0)** — F0BAR0, pointer to I/O mapped GPIO runtime and configuration registers: 0 = Disable; 1 = Enable.
0	Reserved
Index 48h-4Bh	**Reserved**

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 4Ch-4Fh	**Top of System Memory (R/W)** — **Reset Value = FFFFFFFFh**
31:0	**Top of System Memory** — Highest address in system used to determine active decode for external PCI mastered memory cycles. ***Errata***: The last nibble must always remain set else external master bursted writes will be disabled. If an external PCI master requests a memory address below the value programmed in this register, the cycle is transferred from the external PCI bus interface to the Front-PCI interface for servicing by the processor module.
Index 50h	**PIT Control/ISA CLK Divider (R/W)** — **Reset Value = 7Bh**
7	**PIT Software Reset** — 0 = Disable; 1 = Enable.
6	**PIT Counter 1** — 0 = Forces Counter 1 output (OUT1) to zero; 1 = Allows Counter 1 output (OUT1) to pass to the I/O Port 061h[4].
5	**PIT Counter 1 Enable** — 0 = Sets GATE1 input low; 1 = Sets GATE1 input high.
4	**PIT Counter 0** — 0 = Forces Counter 0 output (OUT0) to zero; 1 = Allows Counter 0 output (OUT0) to pass to IRQ0.
3	**PIT Counter 0 Enable** — 0 = Sets GATE0 input low; 1 = Sets GATE0 input high.
2:0	**ISA Clock Divisor** — Determines the divisor of the PCI clock used to make the ISA clock, which is typically programmed for approximately 8 MHz: 000 = Divide by 1; 010 = Divide by 3; 100 = Divide by 5; 110 = Divide by 7 001 = Divide by 2; 011 = Divide by 4; 101 = Divide by 6; 111 = Divide by 8 If PCI clock = 25 MHz, use setting of 010 (divide by 3). It PCI clock = 30 or 33 MHz, use a setting of 011 (divide by 4).
Index 51h	**ISA I/O Recovery Control Register (R/W)** — **Reset Value = 40h**
7:4	**8-Bit I/O Recovery** — These bits determine the number of ISA bus clocks between back-to-back 8-bit I/O read cycles. This count is in addition to a preset one-clock delay built into the controller. 0000 = 1 PCI clock; 0100 = 5 PCI clocks; 1000 = 9 PCI clocks; 1100 = 13 PCI clocks 0001 = 2 PCI clocks; 0101 = 6 PCI clocks; 1001 = 10 PCI clocks; 1101 = 14 PCI clocks 0010 = 3 PCI clocks; 0110 = 7 PCI clocks; 1010 = 11 PCI clocks; 1110 = 15 PCI clocks 0011 = 4 PCI clocks; 0111 = 8 PCI clocks; 1011 = 12 PCI clocks; 1111 = 16 PCI clocks
3:0	**16-Bit I/O Recovery** — These bits determine the number of ISA bus clocks between back-to-back 16-bit I/O cycles. This count is in addition to a preset one-clock delay built into the controller. 0000 = 1 PCI clock; 0100 = 5 PCI clocks; 1000 = 9 PCI clocks; 1100 = 13 PCI clocks 0001 = 2 PCI clocks; 0101 = 6 PCI clocks; 1001 = 10 PCI clocks; 1101 = 14 PCI clocks 0010 = 3 PCI clocks; 0110 = 7 PCI clocks; 1010 = 11 PCI clocks; 1110 = 15 PCI clocks 0011 = 4 PCI clocks; 0111 = 8 PCI clocks; 1011 = 12 PCI clocks; 1111 = 16 PCI clocks
Index 52h	**ROM/AT Logic Control Register (R/W)** — **Reset Value = 98h**
7	**Snoop Fast Keyboard Gate A20 and Fast Reset** — Enables the snoop logic associated with keyboard commands for A20 Mask and Reset: 0 = Disable; 1 = Enable (snooping). If disabled, the keyboard controller handles the commands.
6:5	**Reserved**
4	**Enable A20M# Deassertion on Warm Reset** — Force A20M# high during a Warm Reset (guarantees that A20M# is deasserted regardless of the state of A20): 0 = Disable; 1 = Enable.
3	**Enable Port 092h (Port A)** — I/O Port 092h decode and the logical functions: 0 = Disable; 1 = Enable.
2	**Upper ROM Size** — Selects upper ROM addressing size: 0 = 256 KB (FFFC0000h-FFFFFFFFh); 1 = Use ROM Mask Register (F0 Index 6Eh) ROMCS# goes active for the above ranges. Refer to F0BAR1+I/O Offset 10h[15] for further strapping/programming details. **Note:** The selected range can then be either positively or subtractively decoded through F0 index 5Bh[5].

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
1	**ROM Write Enable** — Enable writes to ROM space, allowing Flash programming: 0 = Disable; 1 = Enable. If strapped for ISA and this bit is set, writes to the configured ROM space will assert ROMCS#, enabling the write cycle to the Flash device on the ISA bus. Otherwise, ROMCS# is inhibited for writes. Refer to F0BAR1+I/O Offset 10h[15] for further LPC strapping/programming details.
0	**Lower ROM Size** — Selects lower ROM addressing size in which ROMCS# goes active: 0 = 000F0000h-000FFFFFh (64 KB) **(Default)** 1 = 000E0000h-000FFFFFh (128 KB) ROMCS# goes active for the above ranges. Refer to F0BAR1+I/O Offset 10h[15] for further strapping/programming details. **Note:** The selected range can then be either positively or subtractively decoded through F0 Index 5Bh[5].
Index 53h	**Alternate CPU Support Register (R/W)** — **Reset Value = 00h**
7	**Enable Keyboard Chip Select** — Allow the ROMCS# signal to fire on keyboard controller I/O accesses. 0 = Disable **(Default)**; 1 = Enable. Note that even if this bit is enabled, F0 Index 81h[3] will prevent the ROMCS# from firing.
6	**Reserved**
5	**Bidirectional SMI Enable** — 0 = Disable; 1 = Enable. This bit must be set to 0.
4:2	**Reserved**
1	**IRQ13 Pin Function Selection** — Selects function of IRQ13/FERR# pin: 0 = FERR#; 1 = IRQ13. This bit must be set to 1.
0	**Generate SMI on A20M# toggle** — 0 = Disable; 1 = Enable. SMI status is reported at F1BAR0+I/O Offset 00h/02h[7].
Index 54h-59h	**Reserved**
Index 5Ah	**Decode Control Register 1 (R/W)** — **Reset Value = 01h**
7	**Secondary Floppy Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 372h-375h and 377h: 0 = Subtractive; 1 = Positive.
6	**Primary Floppy Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 3F2h-3F5h and 3F7h: 0 = Subtractive; 1 = Positive.
5	**COM4 Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 2E8h-2EFh: 0 = Subtractive; 1 = Positive.
4	**COM3 Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 3E8h-3EFh: 0 = Subtractive; 1 = Positive.
3	**COM2 Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 2F8h-2FFh: 0 = Subtractive; 1 = Positive.
2	**COM1 Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 3F8h-3FFh: 0 = Subtractive; 1 = Positive.
1	**Keyboard Controller Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 060h and 064h (and I/O Port 062h/066h if enabled): 0 = Subtractive; 1 = Positive.
0	**Real Time Clock Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 070h and 071h: 0 = Subtractive; 1 = Positive.
Note:	Positive decoding by the South Bridge module speeds up the I/O cycle time. These I/O Ports do not exist in the South Bridge module. It is assumed that if positive decode is enabled, the port exists on the ISA bus.

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 5Bh	**Decode Control Register 2 (R/W)** — **Reset Value = 20h**
7	**Keyboard Port 062h/066h Decode** — This alternate port to the keyboard controller is provided in support of the 8051SL notebook keyboard controller mailbox**:** 0 = Disable; 1 = Enable.
6	**Reserved**
5	**BIOS ROM Positive Decode** — Selects positive or subtractive decoding for accesses to the configured ROM space: 0 = Subtractive; 1 = Positive. ROM configuration is at F0 Index 52h[2:0].
4	**Secondary IDE Controller Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 170h-177h and 376h-377h (excluding writes to 377h): 0 = Subtractive; 1 = Positive. Positively decoded IDE addresses are forwarded to the internal IDE controller and then to the IDE bus. Subtractively decoded IDE addresses are forwarded to the PCI slot bus. If a master abort occurs, they are then forwarded to ISA.
3	**Primary IDE Controller Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 1F0h-1F7h and 3F6h-3F7h (excluding writes to 3F7h): 0 = Subtractive; 1 = Positive. Positively decoded IDE addresses are forwarded to the internal IDE controller and then to the IDE bus. Subtractively decoded IDE addresses are forwarded to the PCI slot bus. If a master abort occurs, they are then forwarded to ISA.
2	**LPT3 Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 3BCh-3BFh: 0 = Subtractive; 1 = Positive.
1	**LPT2 Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 378h-37Fh: 0 = Subtractive; 1 = Positive.
0	**LPT1 Positive Decode** — Selects PCI positive or subtractive decoding for accesses to I/O Port 278h-27Fh: 0 = Subtractive; 1 = Positive.
Note:	Positive decoding by the South Bridge module speeds up the I/O cycle time. The Keyboard, LPT3, LPT2, and LPT1 I/O Ports do not exist in the South Bridge module. It is assumed that if positive decode is enabled, the port exists on the ISA bus.
Index 5Ch	**PCI Interrupt Steering Register 1 (R/W)** — **Reset Value = 00h**
7:4	**INTB# Target Interrupt** 0000 = Disable; 0100 = IRQ4; 1000 = RSVD; 1100 = IRQ12 0001 = IRQ1; 0101 = IRQ5; 1001 = IRQ9; 1101 = RSVD 0010 = RSVD; 0110 = IRQ6; 1010 = IRQ10; 1110 = IRQ14 0011 = IRQ3; 0111 = IRQ7; 1011 = IRQ11; 1111 = IRQ15
3:0	**INTA# Target Interrupt** 0000 = Disable; 0100 = IRQ4; 1000 = RSVD; 1100 = IRQ12 0001 = IRQ1; 0101 = IRQ5; 1001 = IRQ9; 1101 = RSVD 0010 = RSVD; 0110 = IRQ6; 1010 = IRQ10; 1110 = IRQ14 0011 = IRQ3; 0111 = IRQ7; 1011 = IRQ11; 1111 = IRQ15
Note:	The target interrupt must first be configured as level sensitive via I/O Port 4D0h and 4D1h in order to maintain PCI interrupt compatibility

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 5Dh	**PCI Interrupt Steering Register 2 (R/W)** **Reset Value = 00h**
7:4	**INTD# Target Interrupt** 0000 = Disable; 0100 = IRQ4; 1000 = RSVD; 1100 = IRQ12 0001 = IRQ1; 0101 = IRQ5; 1001 = IRQ9; 1101 = RSVD 0010 = RSVD; 0110 = IRQ6; 1010 = IRQ10; 1110 = IRQ14 0011 = IRQ3; 0111 = IRQ7; 1011 = IRQ11; 1111 = IRQ15
3:0	**INTC# Target Interrupt** 0000 = Disable; 0100 = IRQ4; 1000 = RSVD; 1100 = IRQ12 0001 = IRQ1; 0101 = IRQ5; 1001 = IRQ9; 1101 = RSVD 0010 = RSVD; 0110 = IRQ6; 1010 = IRQ10; 1110 = IRQ14 0011 = IRQ3; 0111 = IRQ7; 1011 = IRQ11; 1111 = IRQ15
Note:	The target interrupt must first be configured as level sensitive via I/O Port 4D0h and 4D1h in order to maintain PCI interrupt compatibility
Index 5Eh-5Fh	**Reserved**
Index 60h-63h	**Reserved**
Index 64h-6Dh	**Reserved**
Index 6Eh-6Fh	**ROM Mask Register (R/W)** **Reset Value = FFF0h**
15:8	**Reserved** — Read/modify/write.
7:4	**ROM Size — If F0 Index 52h[2] = 1, these bits select the upper ROM size:** 0000 = 1 MB: FFF00000h-FFFFFFFFh; 1000 = RSVD 0001 = 2 MB: FFE00000h-FFFFFFFFh; 1001 = RSVD 0010 = RSVD; 1010 = RSVD 0011 = 4 MB: FFC00000h-FFFFFFFFh; 1011 = RSVD 0100 = RSVD; 1100 = RSVD 0101 = RSVD; 1101 = RSVD 0110 = RSVD; 1110 = RSVD 0111 = 8 MB: FF800000h-FFFFFFFFh; 1111 = 16 MB: FF000000h-FFFFFFFFh
3:0	Reserved
Index 70h-71h	**IOCS1# Base Address Register (R/W)** **Reset Value = 0000h**
15:0	**I/O Chip Select 1 Base Address** — This 16-bit value represents the I/O base address used to enable the assertion of the IOCS1# signal. This register, together with F0 Index 72h (control register) is used to configure the operation of the IOCS1# pin.
Index 72h	**IOCS1# Control Register (R/W)** **Reset Value = 00h**
7	**I/O Chip Select 1** — IOCS1#: 0 = Disable; 1 = Enable.
6	**Writes Result in Chip Select** — Writes to configured I/O address (base address configured in F0 Index 70h and range configured in bits [4:0]) causes IOCS1# signal to be asserted: 0 = Disable; 1 = Enable.
5	**Reads Result in Chip Select** — Reads from configured I/O address (base address configured in F0 Index 70h and range configured in bits [4:0]) causes IOCS1# signal to be asserted: 0 = Disable; 1 = Enable.
4:0	**IOCS1# I/O Address Range** — This 5-bit field is used to select the range of IOCS0# signal: 00000 = 1 byte; 01111 = 16 bytes 00001 = 2 bytes; 11111 = 32 bytes 00011 = 4 bytes; All other combinations are reserved. 00111 = 8 bytes
Note:	This register, together with F0 Index 70h (base address register) is used to configure the operation of the IOCS1# pin.

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 74h-75h	**IOCS0# Base Address Register (R/W)** — **Reset Value = 0000h**
15:0	**I/O Chip Select 0 Base Address** — This 16-bit value represents the I/O base address used to enable the assertion of the IOCS0# signal. This register, together with F0 Index 76h (control register) is used to configure the operation of the IOCS0# pin.
Index 76h	**IOCS0# Control Register (R/W)** — **Reset Value = 00h**
7	**I/O Chip Select 0** — IOCS0#: 0 = Disable; 1 = Enable.
6	**Writes Result in Chip Select** — Writes to configured I/O address (base address configured in F0 Index 74h and range configured in bits [4:0]) causes IOCS0# signal to be asserted: 0 = Disable; 1 = Enable.
5	**Reads Result in Chip Select** — Reads from configured I/O address (base address configured in F0 Index 74h and range configured in bits [4:0]) causes IOCS0# signal to be asserted: 0 = Disable; 1 = Enable.
4:0	**IOCS0# I/O Address Range** — This 5-bit field is used to select the range of IOCS0# signal: 00000 = 1 byte; 00001 = 2 bytes; 00011 = 4 bytes; 00111 = 8 bytes 01111 = 16 bytes; 11111 = 32 bytes All other combinations are reserved.
Note: This register together with F0 Index 74h (base address register) is used to configure the operation of the IOCS0# pin.	
Index 77h	**Reserved**
Index 78h-7Bh	**DOCCS# Base Address Register (R/W)** — **Reset Value = 00000000h**
31:0	**Disk-On-Chip Chip Select Base Address** — This 32-bit value represents the memory base address used to enable the assertion of the DOCCS# signal. This register, together with F0 Index 7Ch (control register) is used to configure the operation of the DOCCS# pin.
Index 7Ch-7Fh	**DOCCS# Control Register (R/W)** — **Reset Value = 00000000h**
31:27	**Reserved**
26	**Disk-On-Chip Chip Select** — DOCCS#: 0 = Disable; 1 = Enable.
25	**Writes Result in Chip Select** — Writes to configured memory address (base address configured in F0 Index 78h and range configured in bits [18:0]) causes DOCCS# signal to be asserted: 0 = Disable; 1 = Enable.
24	**Reads Result in Chip Select** — Reads from configured memory address (base address configured in F0 Index 78h and range configured in bits [18:0]) causes DOCCS# signal to be asserted: 0 = Disable; 1 = Enable.
23:19	**Reserved**
18:0	**DOCCS# Memory Address Range** — This 19-bit mask is used to further qualify accesses on which the DOCCS# signal is asserted. It is used to mask the upper 19 bits of the incoming PCI address (AD[31:13]).
Note: This register together with F0 Index 78h (base address register) is used to configure the operation of the DOCCS# pin.	

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 80h	**Power Management Enable Register 1 (R/W)** **Reset Value = 00h**
7:4	**Reserved**
3	**IRQ Speedup** — Any unmasked IRQ (per I/O Ports 021h/0A1h) or SMI disables clock throttling (via SUSP#/SUSPA# handshake) for a configurable duration when system is power managed using CPU Suspend modulation: 0 = Disable; 1 = Enable. The duration of the speedup is configured in the IRQ Speedup Timer Count Register (F0 Index 8Ch).
2	**Traps** — Globally enable all power management I/O traps: 0 = Disable; 1 = Enable.
1	**Idle Timers** — Device idle timers: 0 = Disable; 1 = Enable. Note, disable at this level does not reload the timers on the enable. The timers are disabled at their current counts. This bit has no affect on the Suspend Modulation OFF/ON Timers (F0 Index 94h/95h). Only applicable when in APM mode.
0	**Power Management** — Global power management: 0 = Disable; 1 = Enabled. This bit must be set (1) immediately after POST for power management resources to function.
Index 81h	**Power Management Enable Register 2 (R/W)** **Reset Value = 00h**
7	**Reserved**
6	**User Defined Device 3 (UDEF3) Idle Timer Enable** — Turn on UDEF3 **Idle** Timer Count Register (F0 Index A4h) and generate an SMI when the timer expires: 0 = Disable; 1 = Enable. If an access occurs in the programmed address range, the timer is reloaded with the programmed count. UDEF3 address programming is at F0 Index C8h (base address register) and CEh (control register). Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[6].
5	**User Defined Device 2 (UDEF2) Idle Timer Enable** — Turn on UDEF2 **Idle** Timer Count Register (F0 Index A2h) and generate an SMI when the timer expires: 0 = Disable; 1 = Enable. If an access occurs in the programmed address range, the timer is reloaded with the programmed count. UDEF2 address programming is at F0 Index C4h (base address register) and CDh (control register). Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[5].
4	**User Defined Device 1 (UDEF1) Idle Timer Enable** — Turn on UDEF1 **Idle** Timer Count Register (F0 Index A0h) and generate an SMI when the timer expires: 0 = Disable; 1 = Enable. If an access occurs in the programmed address range, the timer is reloaded with the programmed count. UDEF1 address programming is at F0 Index C0h (base address register) and CCh (control register). Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[4].
3	**Keyboard/Mouse Idle Timer Enable** — Turn on Keyboard/Mouse Idle Timer Count Register (F0 Index 9Eh) and generate an SMI when the timer expires: 0 = Disable; 1 = Enable. If an access occurs in the address ranges listed below, the timer is reloaded with the programmed count. Keyboard Controller: I/O Ports 060h/064h COM1: I/O Port 3F8h-3FFh (if F0 Index 93h[1:0] = 10 this range is included) COM2: I/O Port 2F8h-2FFh (if F0 Index 93h[1:0] = 11 this range is included) Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[3].

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
2	**Parallel/Serial Idle Timer Enable** — Turn on Parallel/Serial Port Idle Timer Count Register (F0 Index 9Ch) and generate an SMI when the timer expires: 0 = Disable; 1 = Enable. If an access occurs in the address ranges listed below, the timer is reloaded with the programmed count. LPT1: I/O Port 378h-37Fh, 778h-77Ah LPT2: I/O Port 278h-27Fh, 678h-67Ah COM1: I/O Port 3F8h-3FFh (if F0 Index 93h[1:0] = 10 this range is excluded) COM2: I/O Port 2F8h-2FFh (if F0 Index 93h[1:0] = 11 this range is excluded) COM3: I/O Port 3E8h-3EFh COM4: I/O Port 2E8h-2EFh Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[2].
1	**Floppy Disk Idle Timer Enable** — Turn on Floppy Disk **Idle** Timer Count Register (F0 Index 9Ah) and generate an SMI when the timer expires: 0 = Disable; 1 = Enable. If an access occurs in the address ranges (listed below, the timer is reloaded with the programmed count. Primary floppy disk: I/O Port 3F2h-3F5h, 3F7h, Secondary floppy disk: I/O Port 372h-375h, 377h Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[1].
0	**Primary Hard Disk Idle Timer Enable** — Turn on Primary Hard Disk **Idle** Timer Count Register (F0 Index 98h) and generate an SMI when the timer expires: 0 = Disable; 1 = Enable. If an access occurs in the address ranges selected in F0 Index 93h[5], the timer reloads with the programmed count. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[0].
Index 82h	**Power Management Enable Register 3 (R/W)** — **Reset Value = 00h**
7	**Reserved**
6	**User Defined Device 3 (UDEF3) Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs in the programmed address range, an SMI generates. UDEF3 address programming is at F0 Index C8h (base address register) and CEh (control register). Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[9]. Second level SMI status is reported at F1BAR0+I/O Offset 04h/06h[4].
5	**User Defined Device 2 (UDEF2) Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs in the programmed address range, an SMI generates. UDEF2 address programming is at F0 Index C4h (base address register) and CDh (control register). Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[9]. Second level SMI status is reported at F1BAR0+I/O Offset 04h/06h[3].
4	**User Defined Device 1 (UDEF1) Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs in the programmed address range, an SMI generates. UDEF1 address programming is at F0 Index C0h (base address register), and CCh (control register). Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[9]. Second level SMI status is reported at F1BAR0+I/O Offset 04h/06h[2].
3	**Keyboard/Mouse Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs in the address ranges listed below, an SMI generates. Keyboard Controller: I/O Ports 060h/064h COM1: I/O Port 3F8h-3FFh (if F0 Index 93h[1:0] = 10 this range is included) COM2: I/O Port 2F8h-2FFh (if F0 Index 93h[1:0] = 11 this range is included) Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 86h/F6h[3].

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
2	**Parallel/Serial Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs in the address ranges listed below, an SMI is generated. LPT1: I/O Port 378h-37Fh, 778h-77Ah LPT2: I/O Port 278h-27Fh, 678h-67Ah COM1: I/O Port 3F8h-3FFh (if F0 Index 93h[1:0] = 10 this range is excluded) COM2: I/O Port 2F8h-2FFh (if F0 Index 93h[1:0] = 11 this range is excluded) COM3: I/O Port 3E8h-3EFh COM4: I/O Port 2E8h-2EFh Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 86h/F6h[2].
1	**Floppy Disk Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs in the address ranges listed below, an SMI generates. Primary floppy disk: I/O Port 3F2h-3F5h, 3F7h, Secondary floppy disk: I/O Port 372h-375h, 377h Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 86h/F6h[1].
0	**Primary Hard Disk Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs in the address ranges selected in F0 Index 93h[5], an SMI generates. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 86h/F6h[0]. **Note:** Does not work when the internal IDE is used.
Index 83h	**Power Management Enable Register 4 (R/W)** **Reset Value = 00h**
7	**Secondary Hard Disk Idle Timer Enable** — Turn on **Secondary** Hard Disk **Idle** Timer Count Register (F0 Index ACh) and generate an SMI when the timer expires: 0 = Disable; 1 = Enable. If an access occurs in the address ranges selected in F0 Index 93h[4], the timer is reloaded with the programmed count. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 86h/F6h[4].
6	**Secondary Hard Disk Trap** — 0 = Disable; 1 = Enable. If this bit is enabled and an access occurs in the address ranges selected in F0 Index 93h[4], an SMI is generated. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 86h/F6h[5]. **Note:** Does not work when the internal IDE is used.
5:2	**Reserved**
1	**General Purpose Timer 2 Enable** — Turn on GP Timer 2 Count Register (F0 Index 8Ah) and generate an SMI when the timer expires: 0 = Disable; 1 = Enable. This idle timer is reloaded from the assertion of GPIO7 (if programmed to do so). GP Timer 2 programming is at F0 Index 8Bh[5,3,2]. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[9]. Second level SMI status is reported at F1BAR0+I/O Offset 04h/06h[1].
0	**General Purpose Timer 1 Enable** — Turn on GP Timer 1 Count Register (F0 Index 88h) and generate an SMI when the timer expires: 0 = Disable; 1 = Enable. This idle timer's load is multi-sourced and gets reloaded any time an enabled event (F0 Index 89h[6:0]) occurs. GP Timer 1 programming is at F0 Index 8Bh[4]. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[9]. Second level SMI status is reported at F1BAR0+I/O Offset 04h/06h[0].
Index 84h	**Reserved**

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 85h	**Second Level PME/SMI Status Mirror Register 2 (RO, see Note) — Reset Value = 00h**
7	**Reserved**
6	**User Defined Device 3 Idle Timer (UDEF3) SMI Status (Read Only)** — Was SMI caused by expiration of UDEF3 Idle Timer Count Register (F0 Index A4h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[6] = 1.
5	**User Defined Device 2 Idle Timer (UDEF2) SMI Status (Read Only)** — Was SMI caused by expiration of UDEF2 Idle Timer Count Register (F0 Index A2h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[5] = 1.
4	**User Defined Device 1 Idle Timer (UDEF1) SMI Status (Read Only)** — Was SMI caused by expiration of UDEF1 Idle Timer Count Register (F0 Index A0h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[4] = 1.
3	**Keyboard/Mouse Idle Timer SMI Status (Read Only)** — Was SMI caused by expiration of Keyboard/Mouse Idle Timer Count Register (F0 Index 9Eh)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[3] = 1.
2	**Parallel/Serial Idle Timer SMI Status (Read Only)** — Was SMI caused by expiration of Parallel/Serial Port Idle Timer Count Register (F0 Index 9Ch)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[2] = 1.
1	**Floppy Disk Idle Timer SMI Status (Read Only)** — Was SMI caused by expiration of Floppy Disk Idle Timer Count Register (F0 Index 9Ah)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[1] = 1.
0	**Primary Hard Disk Idle Timer SMI Status (Read Only)** — Was SMI caused by expiration of Primary Hard Disk Idle Timer Count Register (F0 Index 98h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[0] = 1.
Note: Second level of status reporting. Top level status reported at F1BAR0+I/O Offset 00h/02h[0]. If any SMI source described in Index 85h occurs, then bit 0 – SMI Source is Power Management Event of F1BAR+I/O Offset 00h/02h sets. This register is called a “Mirror” register since an identical register exists at F0 Index F5h. Reading this register does not clear the status, while reading its counterpart at F0 Index F5h clears the status at both the second and top levels.	
Index 86h	**Second Level PME/SMI Status Mirror Register 3 (RO, see Note) — Reset Value = 00h**
7:6	**Reserved**
5	**Secondary Hard Disk Access Trap SMI Status (Read Only)** — Was SMI caused by a trapped I/O access to the secondary hard disk? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 83h[6] = 1.
4	**Secondary Hard Disk Idle Timer SMI Status (Read Only)** — Was SMI caused by expiration of Hard Disk Idle Timer Count Register (F0 Index ACh)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 83h[7] = 1.
3	**Keyboard/Mouse Access Trap SMI Status (Read Only)** — Was SMI caused by a trapped I/O access to the keyboard or mouse? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[3] = 1.
2	**Parallel/Serial Access Trap SMI Status (Read Only)** — Was SMI caused by a trapped I/O access to either the serial or parallel ports? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[2] = 1.
1	**Floppy Disk Access Trap SMI Status (Read Only)** — Was SMI caused by a trapped I/O access to the floppy disk? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[1] = 1.
0	**Primary Hard Disk Access Trap SMI Status (Read Only)** — Was SMI caused by a trapped I/O access to the primary hard disk? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[0] = 1.

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Note:	Second level of status reporting. Top level status reported at F1BAR0+I/O Offset 00h/02h[0]. If any SMI source described in Index 86h occurs, then bit 0 – SMI Source is Power Management Event of F1BAR+I/O Offset 00h/02h sets. This register is called a “Mirror” register since an identical register exists at F0 Index F6h. Reading this register does not clear the status, while reading its counterpart at F0 Index F6h clears the status at both the second and top levels.
Index 87h	**Second Level PME/SMI Status Mirror Register 4 (RO, see Note) — Reset Value = 00h**
7	**GPIO Event SMI Status (Read Only)** — Was SMI caused by a transition of any of the GPIOs? 0 = No; 1 = Yes. Note that F0BAR0+I/O Offset 08h/18h selects which GPIOs are enabled to generate a PME. In addition, the selected GPIO must be enabled as an input (F0BAR0+I/O Offset 20h and 24h). The next level (third level) of SMI status is at F0BAR0+I/O 0Ch/1Ch.
6:4	**Reserved**
3	**SIO PWUREQ SMI Status (Read Only)** — Was SMI caused by a power-up event from the SIO? 0 = No; 1 = Yes. A power-up event is defined as any of the following events/activity: Modem, Telephone, Keyboard, Mouse, CEIR (Consumer Electronic Infrared).
2	**Reserved**
1	**Reserved**
0	**Reserved**
Note:	This is the second level of status reporting. The top level status is reported at F1BAR0+I/O Offset 00h/02h[0]. If any SMI source described in Index 87h occurs, then bit 0 – SMI Source is Power Management Event of F1BAR+I/O Offset 00h/02h is set. This register is called a “Mirror” register since an identical register exists at F0 Index F7h. Reading this register does not clear the status, while reading its counterpart at F0 Index F7h clears the status at both the second and top levels except for bit 7 which has a third level of SMI status reporting at F0BAR0+I/O 0Ch/1Ch.
Index 88h	**General Purpose Timer 1 Count Register (R/W) — Reset Value = 00h**
7:0	**General Purpose Timer 1 Count** — This field represents the load value for General Purpose Timer 1. This value can represent either an 8-bit or 16-bit counter (selected in F0 Index 8Bh[4]). It is loaded into the counter when the timer is enabled (F0 Index 83h[0] = 1). Once enabled, an enabled event (configured in F0 Index 89h[6:0]) reloads the timer. The counter is decremented with each clock of the configured timebase (1 msec or 1 sec selected at F0 Index 89h[7]). Upon expiration of the counter, an SMI is generated and the top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[9]. The second level SMI status is reported at F1BAR0+I/O Offset 04h/06h[0]). Once expired, this counter must be re-initialized by either disabling and enabling it, or writing a new count value here.
Index 89h	**General Purpose Timer 1 Control Register (R/W) — Reset Value = 00h**
7	**General Purpose Timer 1 Timebase** — Selects timebase for General Purpose Timer 1 (F0 Index 88h): 0 = 1 sec; 1 = 1 msec.
6	**Re-trigger General Purpose Timer 1 on User Defined Device 3 (UDEF3) Activity** — 0 = Disable; 1 = Enable. Any access to the configured (memory or I/O) address range for UDEF3 (configured in F0 Index C8h and CEh) reloads General Purpose Timer 1.
5	**Re-trigger General Purpose Timer 1 on User Defined Device 2 (UDEF2) Activity** — 0 = Disable; 1 = Enable. Any access to the configured (memory or I/O) address range for UDEF2 (configured in F0 Index C4h and CDh) reloads General Purpose Timer 1.
4	**Re-trigger General Purpose Timer 1 on User Defined Device 1 (UDEF1) Activity** — 0 = Disable; 1 = Enable. Any access to the configured (memory or I/O) address range for UDEF1 (configured in F0 Index C0h and CCh) reloads General Purpose Timer 1.
3	**Re-trigger General Purpose Timer 1 on Keyboard or Mouse Activity** — 0 = Disable; 1 = Enable Any access to the keyboard or mouse I/O address range listed below reloads General Purpose Timer 1. Keyboard Controller: I/O Ports 060h/064h COM1: I/O Port 3F8h-3FFh (if F0 Index 93h[1:0] = 10 this range is included) COM2: I/O Port 2F8h-2FFh (if F0 Index 93h[1:0] = 11 this range is included)

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
2	**Re-trigger General Purpose Timer 1 on Parallel/Serial Port Activity** — 0 = Disable; 1 = Enable. Any access to the parallel or serial port I/O address range listed below reloads the General Purpose Timer 1. LPT1: I/O Port 378h-37Fh, 778h-77Ah LPT2: I/O Port 278h-27Fh, 678h-67Ah COM1: I/O Port 3F8h-3FFh (if F0 Index 93h[1:0] = 10 this range is excluded) COM2: I/O Port 2F8h-2FFh (if F0 Index 93h[1:0] = 11 this range is excluded) COM3: I/O Port 3E8h-3EFh COM4: I/O Port 2E8h-2EFh
1	**Re-trigger General Purpose Timer 1 on Floppy Disk Activity** — 0 = Disable; 1 = Enable. Any access to the floppy disk drive address ranges listed below reloads General Purpose Timer 1. Primary floppy disk: I/O Port 3F2h-3F5h, 3F7h Secondary floppy disk: I/O Port 372h-375h, 377h The active floppy disk drive is configured via F0 Index 93h[7].
0	**Re-trigger General Purpose Timer 1 on Primary Hard Disk Activity** — 0 = Disable; 1 = Enable. Any access to the primary hard disk address range selected in F0 Index 93h[5], reloads General Purpose Timer 1. **Note:** Does not work when the internal IDE is used.
Index 8Ah	**General Purpose Timer 2 Count Register (R/W)** — **Reset Value = 00h**
7:0	**General Purpose Timer 2 Count** — This field represents the load value for General Purpose Timer 2. This value can represent either an 8-bit or 16-bit counter (configured in F0 Index 8Bh[5]). It is loaded into the counter when the timer is enabled (F0 Index 83h[1] = 1). Once the timer is enabled and a transition occurs on GPIO7, the timer is re-loaded. The counter is decremented with each clock of the configured timebase (1 msec or 1 sec selected at F0 Index 8Bh[3]. Upon expiration of the counter, an SMI is generated and the top level of status is F1BAR0+I/O Offset 00h/02h[9]. The second level of status is reported at F1BAR0+I/O Offset 04h/06h[1]). Once expired, this counter must be re-initialized by either disabling and enabling it, or writing a new count value here. For GPIO7 to act as the reload for this counter, it must be enabled as such (F0 Index 8Bh[2]) and be configured as an input. (GPIO pin programming is at F0BAR0+I/O Offset 20h and 24h.)
Index 8Bh	**General Purpose Timer 2 Control Register (R/W)** — **Reset Value = 00h**
7	**Re-trigger General Purpose Timer 1 (GP Timer 1) on Secondary Hard Disk Activity** — 0 = Disable; 1 = Enable. Any access to the secondary hard disk address range selected in F0 Index 93h[4] reloads GP Timer 1.
6	**Reserved**
5	**General Purpose Timer 2 (GP Timer 2) Shift** — GP Timer 2 is treated as an 8-bit or 16-bit timer: 0 = 8-bit; 1 = 16-bit. As an 8-bit timer, the count value is loaded into GP Timer 2 Count Register (F0 Index 8Ah). As a 16-bit timer, the value loaded into GP Timer 2 Count Register is shifted left by eight bits, the lower eight bits become zero, and this 16-bit value is used as the count for GP Timer 2.
4	**General Purpose Timer 1 (GP Timer 1) Shift** — GP Timer 1 is treated as an 8-bit or 16-bit timer: 0 = 8-bit; 1 = 16-bit. As an 8-bit timer, the count value is loaded into GP Timer 1 Count Register (F0 Index 88h). As a 16-bit timer, the value loaded into GP Timer 1 Count Register is shifted left by eight bit, the lower eight bits become zero, and this 16-bit value is used as the count for GP Timer 1.
3	**General Purpose Timer 2 (GP Timer 2) Timebase** — Selects timebase for GP Timer 2 (F0 Index 8Ah): 0 = 1 sec; 1 = 1 msec.
2	**Re-trigger Timer on GPIO7 Pin Transition** — A rising-edge transition on the GPIO7 pin reloads GP Timer 2 (F0 Index 8Ah): 0 = Disable; 1 = Enable. For GPIO7 to work here, it must first be configured as an input. (GPIO pin programming is at F0BAR0+I/O Offset 20h and 24h.)
1:0	**Reserved**

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 8Ch	**IRQ Speedup Timer Count Register (R/W)** **Reset Value = 00h**
7:0	**IRQ Speedup Timer Load Value** — This field represents the load value for the IRQ speedup timer. It is loaded into the counter when Suspend Modulation is enabled (F0 Index 96h[0] = 1) and an INTR or an access to I/O Port 061h occurs. When the event occurs, the Suspend Modulation logic is inhibited, permitting full performance operation of the CPU. Upon expiration, no SMI is generated; the Suspend Modulation begins again. The IRQ speedup timer's timebase is 1 msec. This speedup mechanism allows instantaneous response to system interrupts for full-speed interrupt processing. A typical value here would be 2 to 4 msec.
Index 8Dh-92h	**Reserved**
Index 93h	**Miscellaneous Device Control Register (R/W)** **Reset Value = 00h**
7	**Floppy Disk Port Select** — All system resources used to power manage the floppy disk drive use the primary or secondary FDC addresses for decode: 0 = Secondary; 1 = Primary.
6	**Reserved — This bit must always be set to 1 if written.**
5	**Partial Primary Hard Disk Decode** — This bit is used to restrict the addresses which are decoded as primary hard disk accesses. 0 = Power management monitors all reads and writes I/O Port 1F0h-1F7h, 3F6h-3F7h (excludes writes to 3F7h) 1 = Power management monitors only writes to I/O Port 1F6h and 1F7h
4	**Partial Secondary Hard Disk Decode** — This bit is used to restrict the addresses which are decoded as secondary hard disk accesses. 0 = Power management monitors all reads and writes I/O Port 170h-177h, 376h-377h (excludes writes to 377h) 1 = Power management monitors only writes to I/O Port 176h and 177h
3:2	**Reserved**
1	**Mouse on Serial Enable** — Mouse is present on a Serial Port: 0 = No; 1 = Yes. (Note)
0	**Mouse Port Select** — Selects which serial port the mouse is attached to: 0 = COM1; 1 = COM2. (Note)
Note:	Bits 1 and 0 - If a mouse is attached to a serial port (bit 1 = 1), that port is removed from the serial device list being used to monitor serial port access for power management purposes and added to the keyboard/mouse decode. This is done because a mouse, along with the keyboard, is considered an input device and is used only to determine when to blank the screen. These bits determine the decode used for the Keyboard/Mouse Idle Timer Count Register (F0 Index 9Eh) as well as the Parallel/Serial Idle Timer Count Register (F0 Index 9Ch).
Index 94h	**Suspend Modulation OFF Count Register (R/W)** **Reset Value = 00h**
7:0	**Suspend Signal Deasserted Count** — This 8-bit counter represents the number of 32 µs intervals that the SUSP# pin will be deasserted to the processor. This counter, together with the Suspend Modulation ON Count Register (F0 Index 95h), perform the Suspend Modulation function for CPU power management. The ratio of the on-to-off count sets up an effective (emulated) clock frequency, allowing the power manager to reduce CPU power consumption. This counter is prematurely reset if an enabled speedup event occurs. The speedup events are IRQ speedups and video speedups.
Index 95h	**Suspend Modulation ON Count Register (R/W)** **Reset Value = 00h**
7:0	**Suspend Signal Asserted Count** — This 8-bit counter represents the number of 32 µs intervals that the SUSP# pin will be asserted. This counter, together with the Suspend Modulation OFF Count Register (F0 Index 94h), perform the Suspend Modulation function for CPU power management. The ratio of the on-to-off count sets up an effective (emulated) clock frequency, allowing the power manager to reduce CPU power consumption. This counter is prematurely reset if an enabled speedup event occurs. The speedup events are IRQ speedups and video speedups.

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 96h	**Suspend Configuration Register (R/W)** **Reset Value = 00h**
7:2	**Reserved**
1	**SMI Speedup Configuration** — Selects how Suspend Modulation function reacts when an SMI occurs: 0 = Use the IRQ Speedup Timer Count Register (F0 Index 8Ch) to temporarily disable Suspend Modulation when an SMI occurs. 1 = Disable Suspend Modulation when an SMI occurs until a read to the SMI Speedup Disable Register occurs. (F1BAR0+I/O Offset 08h). The goal of this bit is to disable Suspend Modulation while the CPU is in the System Management Mode so that VSA and Power Management operations occur at full speed. Two methods for accomplishing this are either to map the SMI into the IRQ Speedup Timer Count Register (F0 Index 8Ch), or to have the SMI disable Suspend Modulation until the SMI handler reads the SMI Speedup Disable Register (F1BAR0+I/O Offset 08h). The latter is the preferred method. The IRQ speedup method is provided for software compatibility with earlier revisions of the South Bridge module. This bit has no affect if the Suspend Modulation feature is disabled (bit 0 = 0).
0	**Suspend Modulation Feature Enable** — Suspend Modulation feature: 0 = Disable; 1 = Enable. When enabled, the SUSP# pin will be asserted and deasserted for the durations programmed in the Suspend Modulation OFF/ON Count Registers (F0 Index 94h/95h). This bit setting is mirrored in the Top Level PME/SMI Status Register (F1BAR0+I/O Offset 00h/02h[15]. It is used by the SMI handler to determine if the SMI Speedup Disable Register (F1BAR0+I/O Offset 08h) must be cleared on exit.
Index 97h	**Reserved**
Index 98h-99h	**Primary Hard Disk Idle Timer Count Register (R/W)** **Reset Value = 0000h**
15:0	**Primary Hard Disk Idle Timer Count** — This idle timer is used to determine when the hard disk is not in use so that it can be powered down. The 16-bit value programmed here represents the period of hard disk inactivity after which the system is alerted via an SMI. The timer is automatically reloaded with the count value whenever an access occurs to the configured hard disk’s data port (I/O Port 1F0h or 170h). The counter uses a 1 second timebase. To enable this timer set F0 Index 81h[0] = 1. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[0].
Index 9Ah-9Bh	**Floppy Disk Idle Timer Count Register (R/W)** **Reset Value = 0000h**
15:0	**Floppy Disk Idle Timer Count** — This idle timer is used to determine when the floppy disk drive is not in use so that it can be powered down. The 16-bit value programmed here represents the period of floppy disk drive inactivity after which the system is alerted via an SMI. The timer is automatically reloaded with the count value whenever an access occurs to the configured floppy drive’s data port (I/O Port 3F5h or 375h). The counter uses a 1 second timebase. To enable this timer set F0 Index 81h[1] = 1. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[1].
Index 9Ch-9Dh	**Parallel / Serial Idle Timer Count Register (R/W)** **Reset Value = 0000h**
15:0	**Parallel / Serial Idle Timer Count** — This idle timer is used to determine when the parallel and serial ports are not in use so that the ports can be power managed. The 16-bit value programmed here represents the period of inactivity for these ports after which the system is alerted via an SMI. The timer is automatically reloaded with the count value whenever an access occurs to the parallel (LPT) or serial (COM) I/O address spaces. If the mouse is enabled on a serial port, that port is not considered here. The counter uses a 1 second timebase. To enable this timer set F0 Index 81h[2] = 1. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[2].

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index 9Eh-9Fh	**Keyboard / Mouse Idle Timer Count Register (R/W)** **Reset Value = 0000h**
15:0	**Keyboard / Mouse Idle Timer Count** — This idle timer determines when the keyboard and mouse are not in use so that the LCD screen can be blanked. The 16-bit value programmed here represents the period of inactivity for these ports after which the system is alerted via an SMI. The timer is automatically reloaded with the count value whenever an access occurs to either the keyboard or mouse I/O address spaces, including the mouse serial port address space when a mouse is enabled on a serial port. The counter uses a 1 second timebase. To enable this timer set F0 Index 81h[3] = 1. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[3].
Index A0h-A1h	**User Defined Device 1 Idle Timer Count Register (R/W)** **Reset Value = 0000h**
15:0	**User Defined Device 1 (UDEF1) Idle Timer Count** — This idle timer determines when the device configured as UDEF1 is not in use so that it can be power managed. The 16-bit value programmed here represents the period of inactivity for this device after which the system is alerted via an SMI. The timer is automatically reloaded with the count value whenever an access occurs to memory or I/O address space configured in F0 Index C0h (base address register) and F0 Index CCh (control register). The counter uses a 1 second timebase. To enable this timer set F0 Index 81h[4] = 1. Top level SMI status is reported at F1BAR+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[4].
Index A2h-A3h	**User Defined Device 2 Idle Timer Count Register (R/W)** **Reset Value = 0000h**
15:0	**User Defined Device 2 (UDEF2) Idle Timer Count** — This idle timer determines when the device configured as UDEF2 is not in use so that it can be power managed. The 16-bit value programmed here represents the period of inactivity for this device after which the system is alerted via an SMI. The timer is automatically reloaded with the count value whenever an access occurs to memory or I/O address space configured in the F0 Index C4h (base address register) and F0 Index CDh (control register). The counter uses a 1 second timebase. To enable this timer set F0 Index 81h[5] = 1. Top level SMI status is reported at F1BAR+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[5].
Index A4h-A5h	**User Defined Device 3 Idle Timer Count Register (R/W)** **Reset Value = 0000h**
15:0	**User Defined Device 3 (UDEF3) Idle Timer Count** — This idle timer determines when the device configured as UDEF3 is not in use so that it can be power managed. The 16-bit value programmed here represents the period of inactivity for this device after which the system is alerted via an SMI. The timer is automatically reloaded with the count value whenever an access occurs to memory or I/O address space configured in the UDEF3 Base Address Register (F0 Index C8h) and UDEF3 Control Register (F0 Index CEh). The counter uses a 1 second timebase. To enable this timer set F0 Index 81h[6] = 1. Top level SMI status is reported at F1BAR+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 85h/F5h[6].
Index A6h-ABh	**Reserved**
Index ACh-ADh	**Secondary Hard Disk Idle Timer Count Register (R/W)** **Reset Value = 0000h**
15:0	**Secondary Hard Disk Idle Timer Count** — This idle timer is used to determine when the hard disk is not in use so that it can be powered down. The 16-bit value programmed here represents the period of hard disk inactivity after which the system is alerted via an SMI. The timer is automatically reloaded with the count value whenever an access occurs to the configured hard disk's data port (I/O Port 1F0h or 170h). The counter uses a 1 second timebase. To enable this timer set F0 Index 83h[7] = 1. Top level SMI status is reported at F1BAR0+I/O Offset 00h/02h[0]. Second level SMI status is reported at F0 Index 86h/F6h[4].

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index AEh	**CPU Suspend Command Register (WO)** **Reset Value = 00h**
7:0	**Software CPU Suspend Command (Write Only)** — If bit 0 in the Clock Stop Control Register is set low (F0 Index BCh[0] = 0), a write to this register causes a SUSP#/SUSPA# handshake with the CPU, placing the CPU in a low-power state. The data written is irrelevant. Once in this state, any unmasked IRQ or SMI releases the CPU halt condition. If F0 Index BCh[0] = 1, writing to this register invokes a full system Suspend.
Index AFh	**Suspend Notebook Command Register (WO)** **Reset Value = 00h**
7:0	**Software CPU Stop Clock Suspend (Write Only)** — A write to this register causes a SUSP#/SUSPA# handshake with the CPU, placing the CPU in a low-power state.
Index B0h-B7h	**Reserved**
Index B8h	**DMA Shadow Register (RO)** **Reset Value = xxh**
7:0	**DMA Shadow (Read Only)** — This 8-bit port sequences through the following list of shadowed DMA Controller registers. At power on, a pointer starts at the first register in the list and consecutively reads incrementally through it. A write to this register resets the read sequence to the first register. Each shadow register in the sequence contains the last data written to that location. The read sequence for this register is: 1. DMA Channel 0 Mode Register 2. DMA Channel 1 Mode Register 3. DMA Channel 2 Mode Register 4. DMA Channel 3 Mode Register 5. DMA Channel 4 Mode Register 6. DMA Channel 5 Mode Register 7. DMA Channel 6 Mode Register 8. DMA Channel 7 Mode Register 9. DMA Channel Mask Register (bit 0 is channel 0 mask, etc.) 10. DMA Busy Register (bit 0 or 1 means a DMA occurred within last 1 msec, all other bits are 0)
Index B9h	**PIC Shadow Register (RO)** **Reset Value = xxh**
7:0	**PIC Shadow (Read Only)** — This 8-bit port sequences through the following list of shadowed Interrupt Controller registers. At power on, a pointer starts at the first register in the list and consecutively reads incrementally through it. A write to this register resets the read sequence to the first register. Each shadow register in the sequence contains the last data written to that location. The read sequence for this register is: 1. PIC1 ICW1 2. PIC1 ICW2 3. PIC1 ICW3 4. PIC1 ICW4 - Bits [7:5] of ICW4 are always 0 5. PIC1 OCW2 - Bits [6:3] of OCW2 are always 0 (See note below.) 6. PIC1 OCW3 - Bits [7, 4] are 0 and bit [6, 3] are 1 7. PIC2 ICW1 8. PIC2 ICW2 9. PIC2 ICW3 10. PIC2 ICW4 - Bits [7:5] of ICW4 are always 0 11. PIC2 OCW2 - Bits [6:3] of OCW2 are always 0 (Note) 12. PIC2 OCW3 - Bits [7, 4] are 0 and bit [6, 3] are 1 **Note:** To restore OCW2 to shadow register value, write the appropriate address twice. First with the shadow register value, then with the shadow register value ORed with C0h.

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index BAh	**PIT Shadow Register (RO)** — **Reset Value = xxh**
7:0	**PIT Shadow (Read Only)** — This 8-bit port sequences through the following list of shadowed Programmable Interval Timer registers. At power on, a pointer starts at the first register in the list and consecutively reads to increment through it. A write to this register resets the read sequence to the first register. Each shadow register in the sequence contains the last data written to that location. The read sequence for this register is: 1. Counter 0 LSB (least significant byte) 2. Counter 0 MSB 3. Counter 1 LSB 4. Counter 1 MSB 5. Counter 2 LSB 6. Counter 2 MSB 7. Counter 0 Command Word 8. Counter 1 Command Word 9. Counter 2 Command Word **Note:** The LSB/MSB of the count is the Counter base value, not the current value. Bits [7:6] of the command words are not used.
Index BBh	**RTC Index Shadow Register (RO)** — **Reset Value = xxh**
7:0	**RTC Index Shadow (Read Only)** — The RTC Shadow register contains the last written value of the RTC Index register (I/O Port 070h).
Index BCh	**Clock Stop Control Register (R/W)** — **Reset Value = 00h**
7:4	**PLL Delay** — The programmed value in this field sets the delay (in milliseconds) after a break event occurs before the SUSP# pin is deasserted to the CPU. This delay is designed to allow the clock chip and CPU clock to stabilize before starting execution. This delay is only invoked if the STP_CLK bit was set. The four-bit field allows values from 0 to 15 msec. 0000 = 0 msec; 0100 = 4 msec; 1000 = 8 msec; 1100 = 12 msec 0001 = 1 msec; 0101 = 5 msec; 1001 = 9 msec; 1101 = 13 msec 0010 = 2 msec; 0110 = 6 msec; 1010 = 10 msec; 1110 = 14 msec 0011 = 3 msec; 0111 = 7 msec; 1011 = 11 msec; 1111 = 15 msec
3:1	**Reserved**
0	**CPU Clock Stop** — 0 = Normal SUSP#/ SUSPA# handshake; 1 = Full system Suspend.
Index BDh-BFh	**Reserved**
Index C0h-C3h	**User Defined Device 1 Base Address Register (R/W)** — **Reset Value = 00000000h**
31:0	**User Defined Device 1 (UDEF1) Base Address [31:0]** — This 32-bit register supports power management (trap and idle timer resources) for a PCMCIA slot or some other device in the system. The value written is used as the address comparator for the device trap/timer logic. The device can be memory or I/O mapped (configured in F0 Index CCh). The South Bridge module can not snoop addresses on the Front-PCI bus unless the South Bridge module actually claims the cycle. Therefore, traps and idle timers can not support power management of devices on the Front-PCI bus.
Index C4h-C7h	**User Defined Device 2 Base Address Register (R/W)** — **Reset Value = 00000000h**
31:0	**User Defined Device 2 (UDEF2) Base Address [31:0]** — This 32-bit register supports power management (trap and idle timer resources) for a PCMCIA slot or some other device in the system. The value written is used as the address comparator for the device trap/timer logic. The device can be memory or I/O mapped (configured in F0 Index CDh). The South Bridge module can not snoop addresses on the Front-PCI bus unless the South Bridge module actually claims the cycle. Therefore, traps and idle timers can not support power management of devices on the Front-PCI bus.

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index C8h-CBh	**User Defined Device 3 Base Address Register (R/W)** **Reset Value = 00000000h**
31:0	**User Defined Device 3 (UDEF3) Base Address [31:0]** — This 32-bit register supports power management (trap and idle timer resources) for a PCMCIA slot or some other device in the system. The value written is used as the address comparator for the device trap/timer logic. The device can be memory or I/O mapped (configured in F0 Index CEh). The South Bridge module can not snoop addresses on the Front-PCI bus unless the South Bridge module actually claims the cycle. Therefore, traps and idle timers can not support power management of devices on the Front-PCI bus.
Index CCh	**User Defined Device 1 Control Register (R/W)** **Reset Value = 00h**
7	**Memory or I/O Mapped** — User Defined Device 1 is: 0 = I/O; 1 = Memory.
6:0	**Mask** If bit 7 = 0 (I/O): Bit 6 0 = Disable write cycle tracking 1 = Enable write cycle tracking Bit 5 0 = Disable read cycle tracking 1 = Enable read cycle tracking Bits 4:0 Mask for address bits A[4:0] If bit 7 = 1 (Memory): Bits 6:0 Mask for address memory bits A[15:9] (512 bytes min. and 64 KB max.) A[8:0] are ignored. **Note:** A "1" in a mask bit means that the address bit is ignored for comparison.
Index CDh	**User Defined Device 2 Control Register (R/W)** **Reset Value = 00h**
7	**Memory or I/O Mapped** — User Defined Device 2 is: 0 = I/O; 1 = Memory.
6:0	**Mask** If bit 7 = 0 (I/O): Bit 6 0 = Disable write cycle tracking 1 = Enable write cycle tracking Bit 5 0 = Disable read cycle tracking 1 = Enable read cycle tracking Bits 4:0 Mask for address bits A[4:0] If bit 7 = 1 (Memory): Bits 6:0 Mask for address memory bits A[15:9] (512 bytes min. and 64 KB max.) A[8:0] are ignored. **Note:** A "1" in a mask bit means that the address bit is ignored for comparison.
Index CEh	**User Defined Device 3 Control Register (R/W)** **Reset Value = 00h**
7	**Memory or I/O Mapped** — User Defined Device 3 is: 0 = I/O; 1 = Memory.
6:0	**Mask** If bit 7 = 0 (I/O): Bit 6 0 = Disable write cycle tracking 1 = Enable write cycle tracking Bit 5 0 = Disable read cycle tracking 1 = Enable read cycle tracking Bits 4:0 Mask for address bits A[4:0] If bit 7 = 1 (Memory): Bits 6:0 Mask for address memory bits A[15:9] (512 bytes min. and 64 KB max.) A[8:0] are ignored. **Note:** A "1" in a mask bit means that the address bit is ignored for comparison.
Index CFh	**Reserved**

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
Index D0h	**Software SMI Register (WO)** **Reset Value = 00h**
7:0	**Software SMI (Write Only)** — A write to this location generates an SMI. The data written is irrelevant. This register allows software entry into SMM via normal bus access instructions.
Index D1h-EBh	**Reserved**
Index ECh	**Timer Test Register (R/W)** **Reset Value = 00h**
7:0	**Timer Test Value** — The Timer Test Register is intended only for test and debug purposes. It is not intended for setting operational timebases.
Index EDh-F4h	**Reserved**
Index F5h	**Second Level PME/SMI Status Register 2 (RC, see Note)** **Reset Value = 00h**
7	**Reserved**
6	**User Defined Device3 Idle Timer (UDEF3) SMI Status (Read to Clear)** — Was SMI caused by expiration of UDEF3 Idle Timer Count Register (F0 Index A4h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[6] = 1.
5	**User Defined Device 2 Idle Timer (UDEF2) SMI Status (Read to Clear)** — Was SMI caused by expiration of UDEF2 Idle Timer Count Register (F0 Index A2h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[5] = 1.
4	**User Defined Device Idle 1 Timer (UDEF1) SMI Status (Read to Clear)** — Was SMI caused by expiration of UDEF1 Idle Timer Count Register (F0 Index A0h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[4] = 1.
3	**Keyboard/Mouse SMI Status Idle Timer (Read to Clear)** — Was SMI caused by expiration of Keyboard/Mouse Idle Timer Count Register (F0 Index 9Eh)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[3] = 1.
2	**Parallel/Serial SMI Status Idle Timer (Read to Clear)** — Was SMI caused by expiration of Parallel/Serial Port Idle Timer Count Register (F0 Index 9Ch)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[2] = 1.
1	**Floppy Disk SMI Status Idle Timer (Read to Clear)** — Was SMI caused by expiration of Floppy Disk Idle Timer Count Register (F0 Index 9Ah)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[1] = 1.
0	**Primary Hard Disk SMI Status Idle Timer (Read to Clear)** — Was SMI caused by expiration of Primary Hard Disk Idle Timer Count Register (F0 Index 98h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 81h[0] = 1.
Note:	This is the second level of status reporting. The top level status is reported at F1BAR0+I/O Offset 00h/02h[0]. If any SMI source described in Index F5h occurs, then bit 0 – SMI Source is Power Management Event of F1BAR+I/O Offset 00h/02h sets. Reading register F5h clears the status at both the second and top levels. A read-only “Mirror” version of this register exists at F0 Index 85h. If the value of the register must be read without clearing the SMI source (and consequently deasserting SMI), F0 Index 85h may be read instead.
Index F6h	**Second Level PME/SMI Status Register 3 (RC, see Note)** **Reset Value = 00h**
7:6	**Reserved**
5	**Secondary Hard Disk Access Trap SMI Status (Read to Clear)** — Was SMI caused by a trapped I/O access to the secondary hard disk? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 83h[6] = 1.
4	**Secondary Hard Disk Idle Timer SMI Status (Read to Clear)** — Was SMI caused by expiration of Hard Disk Idle Timer Count Register (F0 Index ACh)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 83h[7] = 1.

Table 4.30 F0 Index xxh: PCI Header and Bridge Configuration Registers (cont.)

Bit	Description
3	**Keyboard/Mouse Access Trap SMI Status (Read to Clear)** — Was SMI caused by a trapped I/O access to the keyboard or mouse? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[3] = 1.
2	**Parallel/Serial Access Trap SMI Status (Read to Clear)** — Was SMI caused by a trapped I/O access to either the serial or parallel ports? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[2] = 1.
1	**Floppy Disk Access Trap SMI Status (Read to Clear)** — Was SMI caused by a trapped I/O access to the floppy disk? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[1] = 1.
0	**Primary Hard Disk Access Trap SMI Status (Read to Clear)** — Was SMI caused by a trapped I/O access to the primary hard disk? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[0] = 1.
Note:	This is the second level of status reporting. The top level status is reported at F1BAR0+I/O Offset 00h/02h[0]. If any SMI source described in Index F6h occurs, then bit 0 – SMI Source is Power Management Event of F1BAR+I/O Offset 00h/02h sets. Reading register F6h clears the status at both the second and top levels. A read-only “Mirror” version of this register exists at F0 Index 86h. If the value of the register must be read without clearing the SMI source (and consequently deasserting SMI), F0 Index 86h may be read instead.
Index F7h	**Second Level PME/SMI Status Register 4 (RO/RC, see Note)** **Reset Value = 00h**
7	**GPIO Event SMI Status** (Read Only, Read does not Clear) — Was SMI caused by a transition of any of the GPIOs? 0 = No; 1 = Yes. Note that F0BAR0+I/O Offset 08h/18h selects which GPIOs are enabled to generate a PME. In addition, the selected GPIO must be enabled as an input (F0BAR0+I/O Offset 20h and 24h). The next level (third level) of SMI status is at F0BAR0+I/O 0Ch/1Ch.
6:4	**Reserved**
3	**SIO PWUREQ SMI Status (Read to Clear)** — Was SMI caused by a power-up event from the SIO? 0 = No; 1 = Yes. A power-up event is defined as any of the following events/activity: Modem, Telephone, Keyboard, Mouse, CEIR (Consumer Electronic Infrared).
2	**Reserved**
1	**Reserved**
0	**Reserved**
Note:	This is the second level of status reporting. Top level status is reported at F1BAR0+I/O Offset 00h/02h[0]. If any SMI source described in Index F7h occurs, then bit 0 – SMI Source is Power Management Event of F1BAR+I/O Offset 00h/02h sets. Reading register F7h clears the status at both the second and top levels except for bit 7 which has a third level of status reporting at F0BAR0+I/O 0Ch/1Ch. A read-only “Mirror” version of this register exists at F0 Index 87h. If the value of the register must be read without clearing the SMI source (and consequently deasserting SMI), F0 Index 87h may be read instead.
Index F8h-FFh	**Reserved**

4.4.3.2. GPIO Support Registers

F0 Index 10h, Base Address Register 0 (F0BAR0) points to the base address of where the GPIO runtime and configuration registers are located. Table 4.31 gives the bit formats of the I/O mapped registers accessed through F0BAR0.

Table 4.31 F0BAR0+I/O Offset xxh: GPIO Runtime and Configuration Registers

Bit	
Offset 00h	**GPDO0 — GPIO Data Out 0 Register (R/W)** **Reset Value = FFFFFFFFh**
31:8	**Reserved**
7:0	**GPIO Data Out** — Bits [7:0] correspond to GPIO7-GPIO0 pins, respectively. The value of each bit determines the value driven on the corresponding GPIO pin when its output buffer is enabled. Writing to the bit latches the written data unless the bit is locked by the GPIO Configuration Register Lock Bit (F0BAR0+I/O Offset 24h[3]). Reading the bit returns the value, regardless of the pin value and configuration. 0 = Corresponding GPIO pin driven to low when output enabled. 1 = Corresponding GPIO pin driven or released to high (according to buffer type and static pull-up selection) when output enabled.
Offset 04h	**GPDI0 — GPIO Data In 0 Register (RO)** **Reset Value = FFFFFFFFh**
31:8	Reserved
7:0	**GPIO Data In (Read Only)** — Bits [7:0] correspond to GPI07-GPIO0 pins, respectively. Reading each bit returns the value of the corresponding GPIO pin, regardless of the pin configuration and the GPDO0 Register value. Writes are ignored. 0 = Corresponding GPIO pin level low; 1 = Corresponding GPIO pin level high.
Offset 08h	**GPIEN0 — GPIO Interrupt Enable 0 Register (R/W)** **Reset Value = 00000000h**
31:8	**Reserved**
7:0	**GPIO Power Management Event (PME) Enable** — Bits [7:0] correspond to GPIO7-GPIO0 pins, respectively. Each bit allows PME generation by the corresponding GPIO pin. 0 = Disable PME generation; 1 = Enable PME generation. **Note:** The individually selected GPIO PMEs generate an SMI and the status is reported at F1BAR0+I/O Offset 00h/02h[0].
Offset 0Ch	**GPST0 — GPIO Status 0 Register (R/W1C)** **Reset Value = 00000000h**
31:8	Reserved
7:0	**GPIO Status** — Bits [7:0] correspond to GPIO7-GPIO0 pins, respectively. Each bit reports a 1 when the hardware detects the edge (rising/falling on the GPIO pin) programmed in F0BAR0+I/O Offset 24h[5]. If the corresponding bit in F0BAR0+I/O Offset 08h is set, this edge generates a PME. 0 = No active edge detected since last cleared; 1 = Active edge detected. **Note:** Writing a 1 FOLLOWED by reading the Status bit clears it to 0. This is the third level of SMI status reporting to the second level at F0 Index 87h/F7h[7] and the top level at F1BAR0+I/O Offset 00h/02h[0]. Clearing the third level also clears the second and top levels.
Offset 10h-1Fh	**Reserved**
Offset 20h	**GPIO Pin Configuration Select Register (R/W)** **Reset Value = 00000000h**
31:6	Reserved
5	**Bank Select** — Selects the GPIO bank to be configured: 0 = Bank 0 for GPIO0-GPIO7 pins.

Table 4.31 F0BAR0+I/O Offset xxh: GPIO Runtime and Configuration Registers (cont.)

Bit	
4:0	**Pin Select** — Selects the GPIO pin to be configured in the Bank selected via bit 5 setting (that is, Bank 0). **If bit 5 = 0; Bank 0** 00000 = GPIO0 00001 = GPIO1 00010 = GPIO2 00011 = GPIO3 00100 = GPIO4 00101 = GPIO5 00110 = GPIO6 00111 = GPIO7
Offset 24h	**GPIO Pin Configuration Access Register (R/W)** **Reset Value = 00000044h**
31:7	**Reserved**
6	**PME Debounce Enable** — Enables/disables IRQ debounce (debounce period = 16 ms): 0 = Disable; 1 = Enable.
5	**PME Polarity** — Selects the polarity of the signal that issues a PME from the corresponding GPIO pin (falling/low or rising/high): 0 = Falling edge or low level input. 1 = Rising edge or high level input.
4	**PME Edge/Level Select** — Selects the type (edge or level) of the signal that issues a PME from the corresponding GPIO pin: 0 = Edge input; 1 = Level input. For normal operation always set this bit to 0 (edge input). Erratic system behavior will result if this bit is set to 1.
3	**Lock** — This bit locks the corresponding GPIO pin. Once this bit is set to 1 by software, it can only be cleared to 0 by system reset or power-off. 0 = No effect **(Default)**; 1 = Direction, output type, pull-up and output value locked.
2	**Pull-Up Control** — Enables/disables the internal pull-up capability of the corresponding GPIO pin. It supports open-drain output signals with internal pull-ups and TTL input signals. 0 = Disable; 1 = Enable **(Default)**. Bits [1:0] must = 01 for this bit to have effect.
1	**Output Type** — Controls the output buffer type (open-drain or push-pull) of the corresponding GPIO pin. 0 = Open-drain **(Default)**; 1 = Push-pull Bit 0 must = 1 for this bit to have effect.
0	**Output Enable** — Indicates the GPIO pin output state. It has no effect on input. 0 = TRI-STATE **(Default)**; 1 = Output enabled.
Offset 28h	**GPIO Reset Control Register (R/W)** **Reset Value = 00000000h**
31:1	**Reserved**
0	**GPIO Reset** — Reset the GPIO logic: 0 = Disable; 1 = Enable. Write 0 to clear. This bit is level-sensitive and must be cleared after the reset is enabled (normal operation requires this bit to be 0).

4.4.3.3. SMI Status Registers - Function

The register space designated as Function 1 (F1) is used to configure the PCI portion of support hardware for the SMI Status Registers. The bit formats for the PCI Header Registers are given in Table 4.32.

Located in the PCI Header Registers of F1 is Base Address Register (F1BAR0) used for pointing to the register spaces designated for SMI Status, described later in this section.

Table 4.32 F1 Index xxh: PCI Header Registers for SMI Status

Bit	Description		
Index 00h-01h	**Vendor Identification Register (RO)**		**Reset Value = 1078h**
Index 02h-03h	**Device Identification Register (RO)**		**Reset Value = 0401h**
Index 04h-05h	**PCI Command Register (R/W)**		**Reset Value = 0000h**
15:1	**Reserved**		
0	**I/O Space** — Allow South Bridge module to respond to I/O cycles from the PCI bus: 0 = Disable; 1 = Enable. This bit must be enabled to access I/O offsets through F1BAR0 (see F1 Index 10h).		
Index 06h-07h	**PCI Status Register (RO)**		**Reset Value = 0280h**
Index 08h	**Device Revision ID Register (RO)**		**Reset Value = 00h**
Index 09h-0Bh	**PCI Class Code Register (RO)**		**Reset Value = 000000h**
Index 0Ch	**PCI Cache Line Size Register (RO)**		**Reset Value = 00h**
Index 0Dh	**PCI Latency Timer Register (RO)**		**Reset Value = 00h**
Index 0Eh	**PCI Header Type (RO)**		**Reset Value = 00h**
Index 0Fh	**PCI BIST Register (RO)**		**Reset Value = 00h**
Index 10h-13h	**Base Address Register 0 - F1BAR0 (R/W)**		**Reset Value = 00000001h**
This register allows access to I/O mapped SMI status related registers. Bits [7:0] are read only (0000 0001), indicating a 256-byte I/O address range. Refer to Table 4.33 for the SMI status registers bit formats and reset values.			
31:8	**SMI Status Base Address**		
7:0	**Address Range (Read Only)**		
Index 14h-2Bh	**Reserved**		
Index 2Ch-2Dh	**Subsystem Vendor ID (RO)**		**Reset Value = 1078h**
Index 2Eh-2Fh	**Subsystem ID (RO)**		**Reset Value = 0401h**
Index 30h-FFh	**Reserved**		

4.4.3.4. SMI Status Support Registers

F1 Index 10h, Base Address Register 0 (F1BAR0), points to the base address of where the SMI Status Registers are located. Table 4.33 gives the bit formats of I/O mapped SMI Status Registers accessed through F1BAR0.

Note: The registers at F1BAR0+I/O Offset 50h-can also be accessed F0 Index 50h-FFh. The preferred method is to program these registers through the F0 register space.

Table 4.33 F1BAR0+I/O Offset xxh: SMI Status Registers

Bit	Description
Offset 00h-01h	**Top Level PME/SMI Status Mirror Register (RO, see Note)** **Reset Value = 0000h**
15	**Suspend Modulation Enable Mirror (Read Only)** — This bit mirrors the Suspend Mode Configuration bit (F0 Index 96h[0]). It is used by the SMI handler to determine if the SMI Speedup Disable Register (F1BAR0+I/O Offset 08h) must be cleared on exit.
14	**SMI Source is USB (Read Only)** — Was SMI caused by USB activity? 0 = No; 1 = Yes. To enable SMI generation set F5BAR0+I/O Offset 00h[20:19] = 11.
13	**SMI Source is Warm Reset Command (Read Only)** — Was SMI caused by Warm Reset command? 0 = No; 1 = Yes.
12	**Reserved**.
11	**SMI Source is SIO (Read Only)** — Was SMI caused by SIO? 0 = No; 1 = Yes. The next level (second level) of SMI status is reported in the SIO module.
10	**SMI Source is EXT_SMI[7:0] (Read Only)** — Was SMI caused by a negative-edge event on EXT_SMI[7:0]? 0 = No; 1 = Yes. The next level (second level) of SMI status is at F1BAR0+I/O Offset 24h[23:8].
9	**SMI Source is GP Timer/UDEF/PCI/ISA Function Trap (Read Only)** — Was SMI caused by expiration of GP Timer 1/2, trapped access to UDEF3/2/1, and/or trapped access to F1-F3 or ISA Legacy Register Space? 0 = No; 1 = Yes. The next level (second level) of SMI status is at F1BAR0+I/O Offset 04h/06h.
8	**SMI Source is Software Generated (Read Only)** — Was SMI caused by software? 0 = No; 1 = Yes.
7	**SMI on an A20M# Toggle (Read Only)** — Was SMI caused by an access to either Port 92h or the keyboard command which initiates an A20M# SMI? 0 = No; 1 = Yes. This method of controlling the internal A20M# in the processor is used instead of a pin. To enable SMI generation set F0 Index 53h[0] = 1.
6:1	**Reserved**
0	**SMI Source is Power Management Event (Read Only)** — Was SMI caused by one of the power management resources (except for the GP Timers, UDEF, and PCI/ISA Function traps that are reported in bit 9): 0 = No; 1 = Yes. The next level (second level) of SMI status is at F0 Index 84h/F4h and 87h/F7h.
Note: Reading this register does not clear the status bits. See F1BAR0+I/O Offset 02h.	
Offset 02h-03h	**Top Level PME/SMI Status Register (RO/RC, see Note)** **Reset Value = 0000h**
15	**Suspend Modulation Enable Mirror (Read to Clear)** — This bit mirrors the Suspend Mode Configuration bit (F0 Index 96h[0]). It is used by the SMI handler to determine if the SMI Speedup Disable Register (F1BAR0+I/O Offset 08h) must be cleared on exit.
14	**SMI Source is USB (Read to Clear)** — Was SMI caused by USB activity? 0 = No; 1 = Yes. To enable SMI generation set F5BAR0+I/O Offset 00h[20:19] = 11.
13	**SMI Source is Warm Reset Command (Read to Clear)** — Was SMI caused by Warm Reset command? 0 = No; 1 = Yes.
12	**SMI Source is NMI (Read to Clear)** — Was SMI caused by NMI activity? 0 = No; 1 = Yes.
11	**SMI Source is SIO (Read to Clear)** — Was SMI caused by SIO? 0 = No; 1 = Yes. The next level (second level) of SMI status is reported in the SIO module.

Table 4.33 F1BAR0+I/O Offset xxh: SMI Status Registers (cont.)

Bit	Description
10	**SMI Source is EXT_SMI[7:0]** (Read to Clear in Rev A and Read Only, Read does not Clear in Rev B) — Was SMI caused by a negative-edge event on EXT_SMI[7:0]? 0 = No; 1 = Yes. The next level (second level) of SMI status is at F1BAR0+I/O Offset 24h[23:8].
9	**SMI Source is General Timers/Traps (Read Only, Read does not Clear)** — Was SMI caused by the expiration of one of the General Purpose Timers or one of the User Defined Traps? 0 = No; 1 = Yes. The next level (second level) of SMI status is at F1BAR0+I/O Offset 04h/06h.
8	**SMI Source is Software Generated (Read to Clear)** — Was SMI caused by software? 0 = No; 1 = Yes.
7	**SMI on an A20M# Toggle (Read to Clear)** — Was SMI caused by an access to either Port 92h or the keyboard command which initiates an A20M# SMI? 0 = No; 1 = Yes. This method of controlling the internal A20M# in the processor is used instead of a pin. To enable SMI generation set F0 Index 53h[0] = 1.
6:1	**Reserved**
0	**SMI Source is Power Management Event (Read Only, Read does not Clear)** — Was SMI caused by one of the power management resources (except for the GP Timers, UDEF, and PCI/ISA Function traps are reported in bit 9): 0 = No; 1 = Yes. The next level (second level) of SMI status is at F0 Index 84h/F4h-87h/F7h.
Note:	Reading this register clears all the SMI status bits except for the "read only" bits because they have a second level of status reporting. Clearing the second level status bits also clears the top level with the exception of GPIOs. GPIO SMIs have a third level of SMI status reporting at F0BAR0+I/O Offset 0Ch/1Ch. Clearing the third level GPIO status bits also clears the second and top levels. A read-only “Mirror” version of this register exists at F1BAR0+I/O Offset 00h. If the value of the register must be read without clearing the SMI source (and consequently deasserting SMI), F1BAR0+I/O Offset 00h may be read instead.

Offset 04h-05h	**Second Level General Traps & Timers PME/SMI Status Mirror Register (RO, See Note)** — **Reset Value = 0000h**
15:6	**Reserved**
5	**PCI/ISA Function Trap (Read Only)** — Was SMI caused by a trapped PCI/ISA configuration cycle? 0 = No; 1 = Yes. To enable SMI generation for: Trapped access to ISA Legacy I/O register space set F0 Index 41h[0] = 1. Trapped access to F1 register space set F0 Index 41h[1] = 1. Trapped access to F2 register space set F0 Index 41h[2] = 1. Trapped access to F3 register space set F0 Index 41h[3] = 1.
4	**SMI Source is Trapped Access to User Defined Device 3 (Read Only)** — Was SMI caused by a trapped I/O or memory access to the User Defined Device 3 (F0 Index C8h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[6] = 1.
3	**SMI Source is Trapped Access to User Defined Device 2 (Read Only)** — Was SMI caused by a trapped I/O or memory access to the User Defined Device 2 (F0 Index C4h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[5] = 1.
2	**SMI Source is Trapped Access to User Defined Device 1 (Read Only)** — Was SMI caused by a trapped I/O or memory access to the User Defined Device 1 (F0 Index C0h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[4] = 1.
1	**SMI Source is Expired General Purpose Timer 2 (Read Only)** — Was SMI caused by the expiration of General Purpose Timer 2 (F0 Index 8Ah)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 83h[1] = 1.
0	**SMI Source is Expired General Purpose Timer 1 (Read Only)** — Was SMI caused by the expiration of General Purpose Timer 1 (F0 Index 88h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 83h[0] = 1.
Note:	This is the second level of status reporting. The top level status is reported at F1BAR0+I/O Offset 00h/02h[9]. Reading this register does not clear the SMI. See F1BAR0+I/O Offset 06h.

Table 4.33 F1BAR0+I/O Offset xxh: SMI Status Registers (cont.)

Bit	Description
Offset 06h-07h	**Second Level General Traps & Timers PME/SMI Status Register (RC, see Note)** **Reset Value = 0000h**
15:6	**Reserved**
5	**PCI/ISA Function Trap (Read to Clear)** — Was SMI caused by a trapped PCI/ISA configuration cycle? 0 = No; 1 = Yes. Trapped Access to ISA Legacy I/O register space; to enable SMI generation set F0 Index 41h[0] = 1. Trapped Access to F1 register space; to enable SMI generation set F0 Index 41h[1] = 1. Trapped Access to F2 register space; to enable SMI generation set F0 Index 41h[2] = 1. Trapped Access to F3 register space; to enable SMI generation set F0 Index 41h[3] = 1.
4	**SMI Source is Trapped Access to User Defined Device 3 (Read to Clear)** — Was SMI caused by a trapped I/O or memory access to the User Defined Device 3 (F0 Index C8h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[6] = 1.
3	**SMI Source is Trapped Access to User Defined Device 2 (Read to Clear)** — Was SMI caused by a trapped I/O or memory access to the User Defined Device 2 (F0 Index C4h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[5] = 1.
2	**SMI Source is Trapped Access to User Defined Device 1 (Read to Clear)** — Was SMI caused by a trapped I/O or memory access to the User Defined Device 1 (F0 Index C0h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 82h[4] = 1.
1	**SMI Source is Expired General Purpose Timer 2 (Read to Clear)** — Was SMI caused by the expiration of General Purpose Timer 2 (F0 Index 8Ah)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 83h[1] = 1.
0	**SMI Source is Expired General Purpose Timer 1 (Read to Clear)** — Was SMI caused by the expiration of General Purpose Timer 1 (F0 Index 88h)? 0 = No; 1 = Yes. To enable SMI generation set F0 Index 83h[0] = 1.
Note:	This is the second level of status reporting. The top level status is reported in F1BAR0+I/O Offset 00h/02h[9]. Reading this register clears the status at both the second and top levels. A read-only “Mirror” version of this register exists at F1BAR0+I/O Offset 04h. If the value of the register must be read without clearing the SMI source (and consequently deasserting SMI), F1BAR0+I/O Offset 04h may be read instead.
Offset 08h-09h	**SMI Speedup Disable Register (Read to Enable)** **Reset Value = 0000h**
15:0	**SMI Speedup Disable** — If bit 1 in the Suspend Configuration Register is set (F0 Index 96h[1] = 1), a read of this register invokes the SMI handler to re-enable Suspend Modulation. The data read from this register can be ignored. If the Suspend Modulation feature is disabled, reading this I/O location has no effect.
Offset 0Fh-1Bh	**Reserved**
Offset 1Ch-1Fh	**Reserved**
Offset 20h-21h	**Reserved**
Offset 22h-23h	**Reserved**
Offset 24h-27h	**Reserved**
Offset 28h-4Fh	**Not Used**
Offset 50h-FFh	The I/O mapped registers located here (F1BAR0+I/O Offset 50h-FFh) can also be accessed at F0 Index 50h-FFh. The preferred method is to program these register through the F0 register space.

4.4.3.5. IDE Controller Registers - Function 2

The register space designated as Function 2 (F2) is used to configure Channels 0 and 1 and the PCI portion of support hardware for the IDE controllers. The bit formats for the PCI Header/Channels 0 and 1 Registers are given in Table 4.34.

Located in the PCI Header Registers of F2 is a Base Address Register (F2BAR4) used for pointing to the register space designated for support of the IDE controllers, described later in this section.

Table 4.34 F2 Index xxh: PCI Header/Channels 0 & 1 Registers for IDE Controller Config

Bit	Description
Index 00h-01h	**Vendor Identification Register (RO)** **Reset Value = 1078h**
Index 02h-03h	**Device Identification Register (RO)** **Reset Value = 0402h**
Index 04h-05h	**PCI Command Register (R/W)** **Reset Value = 0000h**
15:3	**Reserved**
2	**Bus Master** — Allow the South Bridge module bus mastering capabilities: 0 = Disable; 1 = Enable **(Default)**. This bit must be set to 1.
1	**Reserved**
0	**I/O Space** — Allow South Bridge module to respond to I/O cycles from the PCI bus: 0 = Disable; 1 = Enable. This bit must be enabled to access I/O offsets through F2BAR4 (see F2 Index 20h).
Index 06h-07h	**PCI Status Register (RO)** **Reset Value = 0280h**
Index 08h	**Device Revision ID Register (RO)** **Reset Value = 01h**
Index 09h-0Bh	**PCI Class Code Register (RO)** **Reset Value = 010180h**
Index 0Ch	**PCI Cache Line Size Register (RO)** **Reset Value = 00h**
Index 0Dh	**PCI Latency Timer Register (RO)** **Reset Value = 00h**
Index 0Eh	**PCI Header Type (RO)** **Reset Value = 00h**
Index 0Fh	**PCI BIST Register (RO)** **Reset Value = 00h**
Index 10h-13h	**Base Address Register 0 - F2BAR0 (RO)** **Reset Value = 00000000h**
Reserved — Reserved for possible future use by the South Bridge module.	
Index 14h-17h	**Base Address Register 1 - F2BAR1 (RO)** **Reset Value = 00000000h**
Reserved — Reserved for possible future use by the South Bridge module.	
Index 18h-1Bh	**Base Address Register 2 - F2BAR2 (RO)** **Reset Value = 00000000h**
Reserved — Reserved for possible future use by the South Bridge module.	

Table 4.34 F2 Index xxh: PCI Header/Channels 0 & 1 Registers for IDE Controller Config

Bit	Description
Index 1Ch-1Fh	**Base Address Register 3 - F2BAR3 (RO)** — **Reset Value = 00000000h**
	Reserved — Reserved for possible future use by the South Bridge module.
Index 20h-23h	**Base Address Register 4 - F2BAR4 (R/W)** — **Reset Value = 00000001h**
	Base Address 0 Register — This register allows access to I/O mapped Bus Mastering IDE registers. Bits [3:0] are read only (0001), indicating a 16-byte I/O address range. Refer to Table 4.35 for the IDE controller registers bit formats and reset values.
31:4	Bus Mastering IDE Base Address
3:0	Address Range (Read Only)
Index 24h-2Bh	**Reserved**
Index 2Ch-2Dh	**Subsystem Vendor ID (RO)** — **Reset Value = 1078h**
Index 2Eh-2Fh	**Subsystem ID (RO)** — **Reset Value = 0402h**
Index 30h-3Fh	**Reserved**
Index 40h-43h	**Channel 0 Drive 0 PIO Register (R/W)** — **Reset Value =** 00009172h
	If Index 44h[31] = 0, Format 0 — Selects slowest PIOMODE per channel for commands. Format 0 settings for: PIO Mode 0 = 00009172h PIO Mode 1 = 00012171h PIO Mode 2 = 00020080h PIO Mode 3 = 00032010h PIO Mode 4 = 00040010h
31:20	**Reserved**
19:16	**PIOMODE** — PIO mode
15:12	**t2I** — Recovery time (value + 1 cycle)
11:8	**t3** — IDE_IOW# data setup time (value + 1 cycle)
7:4	**t2W** — IDE_IOW# width minus t3 (value + 1 cycle)
3:0	**t1** — **Address Setup Time** (value + 1 cycle)
	If Index 44h[31] = 1, Format 1 — Allows independent control of command and data. Format 1 settings for: PIO Mode 0 = 9172D132h PIO Mode 1 = 21717121h PIO Mode 2 = 00803020h PIO Mode 3 = 20102010h PIO Mode 4 = 00100010h
31:28	**t2IC** — Command cycle recovery time (value + 1 cycle)
27:24	**t3C** — Command cycle IDE_IOW# data setup (value + 1 cycle)
23:20	**t2WC** — Command cycle IDE_IOW# pulse width minus t3 (value + 1 cycle)
19:16	**t1C** — Command cycle address setup time (value + 1 cycle)
15:12	**t2ID** — Data cycle recovery time (value + 1 cycle)
11:8	**t3D** — Data cycle IDE_IOW# data setup (value + 1 cycle)
7:4	**t2WD** — Data cycle IDE_IOW# pulse width minus t3 (value + 1 cycle)
3:0	**t1D** — **Data cycle address Setup Time** (value + 1 cycle)

Table 4.34 F2 Index xxh: PCI Header/Channels 0 & 1 Registers for IDE Controller Config

Bit	Description
Index 44h-47h	**Channel 0 Drive 0 DMA Control Register (R/W)** **Reset Value =** 00077771h
If bit 20 = 0, Multiword DMA	Settings for: Multiword DMA Mode 0 = 00077771h Multiword DMA Mode 1 = 00012121h Multiword DMA Mode 2 = 00002020h
31	**PIO Mode Format** — 0 = Format 0; 1 = Format 1
30:21	**Reserved**
20	**DMA Select** — DMA operation: 0 = Multiword DMA, 1 = Ultra DMA.
19:16	**tKR** — IDE_IOR# recovery time (4-bit) (value + 1 cycle)
15:12	**tDR** — IDE_IOR# pulse width (value + 1 cycle)
11:8	**tKW** — IDE_IOW# recovery time (4-bit) (value + 1 cycle)
7:4	**tDW** — IDE_IOW# pulse width (value + 1 cycle)
3:0	**tM** — IDE_CS0#/CS1# to IDE_IOR#/IOW# setup; IDE_CS0#/CS1# setup to IDE_DACK0#/DACK1#
If bit 20 = 1, Ultra DMA	Settings for: Ultra DMA Mode 0 = 00921250h Ultra DMA Mode 1 = 00911140h Ultra DMA Mode 2 = 00911030h
31	**PIO Mode Format** — 0 = Format 0; 1 = Format 1
30:21	**Reserved**
20	**DMA Select** — DMA operation: 0 = Multiword DMA, 1 = Ultra DMA.
19:16	**tCRC** — CRC setup UDMA in IDE_DACK# (value + 1 cycle) (for host terminate CRC setup = tMLI + tSS)
15:12	**tSS** — UDMA out (value + 1 cycle)
11:8	**tCYC** — Data setup and cycle time UDMA out (value + 2 cycles)
7:4	**tRP** — Ready to pause time (value + 1 cycle). Note: tRFS + 1 tRP on next clock.
3:0	**tACK** — IDE_CS0#/CS1# setup to IDE_DACK0#/DACK1# (value + 1 cycle)
Index 48h-4Bh	**Channel 0 Drive 1 PIO Register (R/W)** **Reset Value =** 00009172h
	Channel 0 Drive 1 Programmed I/O Control Register — Refer to F2 Index 40h for bit descriptions.
Index 4Ch-4Fh	**Channel 0 Drive 1 DMA Control Register (R/W)** **Reset Value =** 00077771h
	Channel 0 Drive 1 MDMA/UDMA Control Register — See F2 Index 44h for bit descriptions. **Note:** Once the PIO Mode Format is selected in F2 Index 44h[31], bit 31 of this register is defined as reserved, read only.
Index 50h-53h	**Channel 1 Drive 0 PIO Register (R/W)** **Reset Value =** 00009172h
	Channel 1 Drive 0 Programmed I/O Control Register — Refer to F2 Index 40h for bit descriptions.
Index 54h-57h	**Channel 1 Drive 0 DMA Control Register (R/W)** **Reset Value =** 00077771h
	Channel 1 Drive 0 MDMA/UDMA Control Register — See F2 Index 44h for bit descriptions. **Note:** Once the PIO Mode Format is selected in F2 Index 44h[31], bit 31 of this register is defined as reserved, read only.
Index 58h-5Bh	**Channel 1 Drive 1 PIO Register (R/W)** **Reset Value =** 00009172h
	Channel 1 Drive 1 Programmed I/O Control Register — Refer to F2 Index 40h for bit descriptions.

Table 4.34 F2 Index xxh: PCI Header/Channels 0 & 1 Registers for IDE Controller Config

Bit	Description
Index 5Ch-5Fh	**Channel 1 Drive 1 DMA Control Register (R/W)** **Reset Value =** 00077771h
	Channel 1 Drive 1 MDMA/UDMA Control Register — See F2 Index 44h for bit descriptions. **Note:** Once the PIO Mode Format is selected in F2 Index 44h[31], bit 31 of this register is defined as reserved, read only.
Index 60h-FFh	**Reserved**

4.4.3.6. IDE Controller Support Registers

F2 Index 20h, Base Address Register 4 (F2BAR4), points to the base address of where the registers for IDE controller configuration are located. Table 4.35 gives the bit formats of the I/O mapped IDE Controller Configuration Registers accessed through F2BAR4.

Note: For proper operation the register must be read before the Bus Master Control at F2BAR4+0h is cleared, else the status may be lost.

Table 4.35 F2BAR4+I/O Offset xxh: IDE Controller Configuration Registers

Bit	Description
Offset 00h	**IDE Bus Master 0 Command Register — Primary (R/W)** **Reset Value = 00h**
7:4	**Reserved** — Set to 0. Must return 0 on reads.
3	**Read or Write Control** — Sets the direction of bus master transfers: 0 = PCI reads performed; 1 = PCI writes performed. This bit should not be changed when the bus master is active.
2:1	**Reserved**— Set to 0. Must return 0 on reads.
0	**Bus Master Control** — Controls the state of the bus master: 0 = Disable master; 1 = Enable master Hault bus master operationsn by setting bit 0 to 0. Once an operation halts, it can not be resumed. If bit 0 is set to 0 while a bus master operation is active, the command aborts and the data transferred from the drive is discarded. This bit should be reset after completion of data transfer.
Offset 01h	**Reserved**
Offset 02h	**IDE Bus Master 0 Status Register — Primary (R/W)** **Reset Value = 00h**
7	**Simplex Mode (Read Only)** — Can both the primary and secondary channel operate independently? 0 = Yes; 1 = No (simplex mode)
6	**Drive 1 DMA Capable** — Allow Drive 1 to be capable of DMA transfers: 0 = Disable; 1 = Enable.
5	**Drive 0 DMA Capable** — Allow Drive 0 to be capable of DMA transfers: 0 = Disable; 1 = Enable.
4:3	**Reserved** — Set to 0. Must return 0 on reads.
2	**Bus Master Interrupt** — Has the bus master detected an interrupt? 0 = No; 1 = Yes. Write 1 to clear.
1	**Bus Master Error** — Has the bus master detected an error during data transfer? 0 = No; 1 = Yes. Write 1 to clear.
0	**Bus Master Active** — Is the bus master active? 0 = No; 1 = Yes.
Offset 03h	**Reserved**

Table 4.35 F2BAR4+I/O Offset xxh: IDE Controller Configuration Registers

Bit	Description
Offset 04h-07h	**IDE Bus Master 0 PRD Table Address — Primary (R/W)** **Reset Value = 00000000h**
31:2	**Pointer to the Physical Region Descriptor Table** — This register is a PRD table pointer for IDE Bus Master 0. When written, this register points to the first entry in a PRD table. Once IDE Bus Master 0 is enabled (Command Register bit 0 = 1], it loads the pointer and updates this register to the next PRD by adding 08h. When read, this register points to the next PRD. **Note:** Entries in the PRD must be 32 byte aligned.
1:0	**Reserved** — Set to 0.
Offset 08h	**IDE Bus Master 1 Command Register — Secondary (R/W)** **Reset Value = 00h**
7:4	**Reserved** — Set to 0. Must return 0 on reads.
3	**Read or Write Control** — Sets the direction of bus master transfers: 0 = PCI reads performed; 1 = PCI writes performed. This bit should not be changed when the bus master is active.
2:1	**Reserved** — Set to 0. Must return 0 on reads.
0	**Bus Master Control** — Controls the state of the bus master: 0 = Disable master; 1 = Enable master Hault bus master operations by setting bit 0 to 0. Once an operation halts, it can not be resumed. If bit 0 is set to 0 while a bus master operation is active, the command aborts and the data transferred from the drive is discarded. This bit should be reset after completion of data transfer.
Offset 09h	**Reserved**
Offset 0Ah	**IDE Bus Master 1 Status Register — Secondary (R/W)** **Reset Value = 00h**
7	**Simplex Mode** — Can both the primary and secondary channel operate independently? 0 = Yes; 1 = No (simplex mode).
6	**Drive 1 DMA Capable** — Allow Drive 1 to be capable of DMA transfers: 0 = Disable; 1 = Enable.
5	**Drive 0 DMA Capable** — Allow Drive 0 to be capable of DMA transfers: 0 = Disable; 1 = Enable.
4:3	**Reserved** — Set to 0. Must return 0 on reads.
2	**Bus Master Interrupt** — Has the bus master detected an interrupt? 0 = No; 1 = Yes. Write 1 to clear.
1	**Bus Master Error** — Has the bus master detected an error during data transfer? 0 = No; 1 = Yes. Write 1 to clear.
0	**Bus Master Active** — Is the bus master active? 0 = No; 1 = Yes.
Offset 0Bh	**Reserved**
Offset 0Ch-0Fh	**IDE Bus Master 1 PRD Table Address — Secondary (R/W)** **Reset Value = 00000000h**
31:2	**Pointer to the Physical Region Descriptor Table** — This register is a PRD table pointer for IDE Bus Master 1. When written, this register points to the first entry in a PRD table. Once IDE Bus Master 1 is enabled (Command Register bit 0 = 1], it loads the pointer and updates this register to the next PRD by adding 08h. When read, this register points to the next PRD. **Note:** Entries in the PRD must be 32 byte aligned.
1:0	**Reserved** — Set to 0.

4.4.3.7. XBus Expansion - Function 3

The register space designated as Function 3 (F3) is used to configure the PCI portion of support hardware for accessing the XBus Expansion support registers. The bit formats for the PCI Header Registers are given in Table 4.36.

Located in the PCI Header Registers of F3 are five Base Address Registers (F3BARx) used for pointing to the register spaces designated for XBus Expansion, described later in this section.

Table 4.36 F3 Index xxh: PCI Header Registers for XBus Expansion

Bit	Description
Index 00h-01h	**Vendor Identification Register (RO) — Reset Value = 1078h**
Index 02h-03h	**Device Identification Register (RO) — Reset Value = 0403h**
Index 04h-05h	**PCI Command Register (R/W) — Reset Value = 0000h**
15:2	**Reserved (Read Only)**
1	**Memory Space** — Allow South Bridge module to respond to memory cycles from the PCI bus: 0 = Disable; 1 = Enable. If any of F3BAR1, F3BAR2, F3BAR3, F3BAR4, or F3BAR5 (see F3 Index 10h, 14h, 18h, 1Ch, 20h, 24h) are defined as allowing access to memory mapped registers, this bit must be set to 1. BAR configuration is programmed through the corresponding mask register (see F3 Index 40h, 44h, 48h, 4Ch, 50h, and 54h).
0	**I/O Space** — Allow South Bridge module to respond to I/O cycles from the PCI bus: 0 = Disable; 1 = Enable. This bit must be enabled to access I/O offsets through F3BAR0 (see F3 Index 10h). If any of F3BAR1, F3BAR2, F3BAR3, F3BAR4, or F3BAR5 (see F3 Index 10h, 14h, 18h, 1Ch, 20h, 24h) are defined as allowing access to I/O mapped registers, this bit must be set to 1. BAR configuration is programmed through the corresponding mask register (see F3 Index 40h, 44h, 48h, 4Ch, 50h, and 54h).
Index 06h-07h	**PCI Status Register (RO) — Reset Value = 0280h**
Index 08h	**Device Revision ID Register (RO) — Reset Value = 00h**
Index 09h-0Bh	**PCI Class Code Register (RO) — Reset Value = 000000h**
Index 0Ch	**PCI Cache Line Size Register (RO) — Reset Value = 00h**
Index 0Dh	**PCI Latency Timer Register (RO) — Reset Value = 00h**
Index 0Eh	**PCI Header Type (RO) — Reset Value = 00h**
Index 0Fh	**PCI BIST Register (RO) — Reset Value = 00h**
Index 10h-13h	**Base Address Register 0 - F3BAR0 (R/W) — Reset Value = 00000000h** **XBus Expansion Address Space** — This register allows PCI access to I/O mapped XBus Expansion registers. Bits [5:0] must be set to 000001, indicating a 64-byte aligned I/O address space. Refer to Table 4.36 for the XBus Expansion configuration registers bit formats and reset values. **Note:** The size and type of accessed offsets can be re-programmed through F3BAR0 Mask Register (F3 Index 40h).
31:6	XBus Expansion **Base Address**
5:0	**Address Range** — These bits must be set to 000001 for this register operate correctly.

Table 4.36 F3 Index xxh: PCI Header Registers for XBus Expansion

Bit	Description
Index 14h-17h	**Base Address Register 1 - F3BAR1 (R/W)** — **Reset Value = 00000000h**
	Reserved— Reserved for possible future use by the South Bridge module. Configuration of this register is programmed through the F3BAR1 Mask Register (F3 Index 44h).
Index 18h-1Bh	**Base Address Register 2 - F3BAR2 (R/W)** — **Reset Value = 00000000h**
	Reserved— Reserved for possible future use by the South Bridge module. Configuration of this register is programmed through the F3BAR2 Mask Register (F3 Index 48h).
Index 1Ch-1Fh	**Base Address Register 3 - F3BAR3 (R/W)** — **Reset Value = 00000000h**
	Reserved -- Reserved for possible future use by the South Bridge module. Configuration of this register is programmed through the F3BAR3 Mask Register (F3 Index 4Ch).
Index 20h-23h	**Base Address Register 4 - F3BAR4 (R/W)** — **Reset Value = 00000000h**
	Reserved — Reserved for possible future use by the South Bridge module. Configuration of this register is programmed through the F3BAR4 Mask Register (F3 Index 50h).
Index 24h-27h	**Base Address Register 5 - F3BAR5 (R/W)** — **Reset Value = 00000000h**
	Reserved — Reserved for possible future use by the South Bridge module. Configuration of this register is programmed through the F3BAR5 Mask Register (F3 Index 54h).
Index 28h-2Bh	**Reserved**
Index 2Ch-2Dh	**Subsystem Vendor ID (RO)** — **Reset Value = 1078h**
Index 2Eh-2Fh	**Subsystem ID (RO)** — **Reset Value = 0405h**
Index 30h-3Fh	**Reserved**
Index 40h-43h	**F3BAR0 Mask Address Register (R/W)** — **Reset Value = 00000000h**
	To use F3BAR0, the mask register should be programmed first. The mask register defines the size of F3BAR0 and whether the accessed offset registers are memory or I/O mapped. Note that whenever this mask register is written to, F3BAR0 must also be rewritten even if the value of F3BAR0 does not change.
	Memory Base Address Register (Bit 0 == 0)
31:4	**Address Mask** — Use to determine the size of the BAR. Every bit that is a 1 is programmable in the BAR. Every bit that is a 0 will be fixed 0 in the BAR. Since the address mask goes down to bit 4, the smallest memory region is 16 bytes, however, the PCI Specification suggests not using less than 4 KB address range.
3	**Prefetchable**
2:1	**Type** 00 = Located anywhere in 32-bit address space 01 = Located below 1 MB 10 = Located anywhere in 64-bit address spaced 11 = Reserved
0	Must = 0 for memory

Table 4.36 F3 Index xxh: PCI Header Registers for XBus Expansion

Bit	Description
I/O Base Address Register (Bit 0 == 1)	
31:2	**Address Mask** — Use to determine in the size of the BAR. Every bit that is a 1 is programmable in the BAR. Every bit that is a 0 will be fixed 0 in the BAR. Since the address mask goes down to bit 2, the smallest I/O region is 4 bytes, however, the PCI Specification suggests not using less than 4 KB address range.
1	**Reserved** — Must be 0.
0	Must = 1 for I/O
Index 44h-47h	**F3BAR1 Mask Address Register (R/W)** — **Reset Value = 00000000h**
	To use F3BAR1, the mask register should be programmed first. The mask register defines the size of F3BAR1 and whether the accessed offset registers are memory or I/O mapped. Note that whenever this mask register is written to, F3BAR1 must also be rewritten even if the value of F3BAR1 does not change. See F3 Index 40h (F3BAR0 Mask Address Register) for bit descriptions.
Index 48h-4Bh	**F3BAR2 Mask Address Register (R/W)** — **Reset Value = 00000000h**
	To use F3BAR2, the mask register should be programmed first. The mask register defines the size of F3BAR2 and whether the accessed offset registers are memory or I/O mapped. Note that whenever this mask register is written to, F3BAR2 must also be rewritten even if the value of F3BAR2 does not change. See F3 Index 40h (F3BAR0 Mask Address Register) for bit descriptions.
Index 4Ch-4Fh	**F3BAR3 Mask Address Register (R/W)** — **Reset Value = 00000000h**
	To use F3BAR3, the mask register should be programmed first. The mask register defines the size of F3BAR3 and whether the accessed offset registers are memory or I/O mapped. Note that whenever this mask register is written to, F3BAR3 must also be rewritten even if the value of F3BAR3 does not change. See F3 Index 40h (F3BAR0 Mask Address Register) for bit descriptions.
Index 50h-53h	**F3BAR4 Mask Address Register (R/W)** — **Reset Value = 00000000h**
	To use F3BAR4, the mask register should be programmed first. The mask register defines the size of F3BAR4 and whether the accessed offset registers are memory or I/O mapped. Note that whenever this mask register is written to, F3BAR4 must also be rewritten even if the value of F3BAR4 does not change. See F3 Index 40h (F3BAR0 Mask Address Register) for bit descriptions.
Index 54h-57h	**F3BAR5 Mask Address Register (R/W)** — **Reset Value = 00000000h**
	To use F3BAR5, the mask register should be programmed first. The mask register defines the size of F3BAR5 and whether the accessed offset registers are memory or I/O mapped. Note that whenever this mask register is written to, F3BAR5 must also be rewritten even if the value of F3BAR5 does not change. See F3 Index 40h (F3BAR0 Mask Address Register) on page 118 for bit descriptions.
Index 58h	**F3BARx Initialized Register (R/W)** — **Reset Value = 00h**
7:6	**Reserved** — Set to 0.
5	**F3BAR5 Initialized** — This bit reflects if F3BAR5 (F3 Index 24h) has been initialized. At reset this bit is cleared (0). Writing F3BAR5 sets (1) this bit. If this bit programmed to 0, the decoding of F3BAR5 will be disabled until either this bit is programmed to 1 or F3BAR5 is written.
4	**F3BAR4 Initialized** — This bit reflects if F3BAR4 (F3 Index 20h) has been initialized. At reset this bit is cleared (0). Writing F3BAR4 sets (1) this bit. If this bit programmed to 0, the decoding of F3BAR4 will be disabled until either this bit is programmed to 1 or F3BAR4 is written.
3	**F3BAR3 Initialized** — This bit reflects if F3BAR3 (F3 Index 1Ch) has been initialized. At reset this bit is cleared (0). Writing F3BAR3 sets (1) this bit. If this bit is programmed to 0, the decoding of F3BAR3 will be disabled until either this bit is programmed to 1 or F3BAR3 is written.

Table 4.36 F3 Index xxh: PCI Header Registers for XBus Expansion

Bit	Description
2	**F3BAR2 Initialized** — This bit reflects if F3BAR2 (F3 Index 18h) has been initialized. At reset this bit is cleared (0). Writing F3BAR2 sets (1) this bit. If this bit is programmed to 0, the decoding of F3BAR2 will be disabled until either this bit is programmed to 1 or F3BAR2 is written.
1	**F3BAR1 Initialized** — This bit reflects if F3BAR1 (F3 Index 14h) has been initialized. At reset this bit is cleared (0). Writing F3BAR1 sets (1) this bit. If this bit is programmed to 0, the decoding of F3BAR1 will be disabled until either this bit is programmed to 1 or F3BAR1 is written.
0	**F3BAR0 Initialized** — This bit reflects if F3BAR0 (F3 Index 10h) has been initialized. At reset this bit is cleared (0). Writing F3BAR0 sets (1) this bit. If this bit is programmed to 0, the decoding of F3BAR0 will be disabled until either this bit is programmed to 1 or F3BAR0 is written.
Index 59h-FFh	**Reserved**

4.4.3.8. XBus Expansion Support Registers

F3 Index 10h, Base Address Register 0 (F3BAR0) set the base address that allows PCI access to additional I/O Control support registers. Table 4.37 shows the support registers accessed through F3BAR0.

Table 4.37 F3BAR0+I/O Offset xxh: XBus Expansion Registers

Bit	Description
Offset 00h-03h	**I/O Control Register 1 (R/W)** **Reset Value = 010C0007h**
31:28	**Reserved**
27	**Enable Integrated SIO Infrared (IO_ENABLE_SIO_IR)** — 0 = Disable; 1 = Enable.
26:25	**Integrated SIO Input Configuration (IO_SIOCFG_IN) —** These two bits can be used to disable the integrated SIO totally or limit/control the base address: 00 = Integrated SIO disable 01 = Integrated SIO configuration access disable 10 = Integrated SIO base address 02Eh/02Fh enable 11 = Integrated SIO base address 015Ch/015Dh enable
24	**Enable Integrated SIO ISA Bus Control (IO_ENABLE_SIO_DRIVING_ISA_BUS)** — Allow the integrated SIO to drive the internal and external ISA bus: 0 = Disable; 1 = Enable **(Default)**.
23	**Reserved**
22	**IO_CLK32K_OE** **This bit is set to drive 32Khz clock out on to GPIO[0]. Reset to 0.**
21	**IO_RTC_32K** This bit selects which 32K clock source is used. Resets to 0. 0 = use SIO generated 32Khz. Clock is driven by RTC. 1 = use internally generated 32Khz. Clock is derived by dividing the 48Mhz by 1484. **Note**: bootstrap[6] = 1 can also be used to select internally generated 32Khz clock.
20	**USB Internal SMI (IO_USB_SMI_PWM_EN)** — Route USB-generated SMI through the Top Level SMI Status Register at F1BAR0+I/O Offset 00h[14]: 0 = Disable; 1 = Enable. Bit 19 must be enabled to allow the USB to generate an SMI for status reporting.
19	**USB SMI I/O Configuration (IO_USB_SMI_PIN_EN)** — Route USB-generated SMI directly to the SMI# pin: 0 = Disable 1 = Enable, USB-generated SMI pulls SMI# pin active (low) If bits 19 and 20 are enabled, the SMI generated by the USB is reported through the Top Level SMI Status Register at F1BAR0+I/O Offset 00h[14]. If only bit 19 is enabled, the USB can generate an SMI but there is no status reporting.

Table 4.37 F3BAR0+I/O Offset xxh: XBus Expansion Registers

Bit	Description	
18	**USB (IO_USB_PCI_EN)** — USB ports: 0 = Disable; 1 = Enable.	
17	External KBC -- Must be left at 0.	
16	External RTC -- Must be left at 0.	
15:0	**Reserved**	
Offset 04h-07h	**I/O Control Register 2 (R/W)**	**Reset Value = 00000000h**
31:8	**Reserved**	
7	**IO_CLK_14M_OE** — Set to drive the internally generated 12Mhz clock out on GPIO[4]. Resets to 0.	
6	**Reserved**	
5	**IO_ZT_EN** Set to enable the ZF-Logic ROM interface. Resets to 0. **Note:** bootstrap[23] is also used to enable ZF-Logic ROM interface.[a]	
4	**IO_ZFL_EN** Set to enable the ZF-Logic. Resets to 0. **Note**: bootstrap[22] is also used to enable the ZF-Logic.[a]	
3	**IO_FUNC_ON_SIO** — Used in design verification only. Resets to 0.	
2	**IO_BUR_ON_SIO** — Used in design verification only. Resets to 0.	
1	**IO_IDE_ON_GPIO** — Drive IDE channel 2 onto gpio. Must also have gpio conditioned to correct direction corresponding to IDE pin functionality. See also Table 4.6 "IDE Interface Signals" on page 175. 0 = Do not drive IDE onto gpio. **(default)** 1 = Drive IDE into gpio. gpio[1] is dmackx, gpio must be configured as output. gpio[2] is diowx, gpio must be configured as output. gpio[3] is diorx, gpio must be configured as output. gpio[5] is dreq, gpio must be configured as input. gpio[6] is iordy, gpio must be configured as input.	
0	**IO_EXT_CLK_14M** Select 14.3Mhz clock source. Select either internally or externally generated 14.3Mhz clock source. If internal source is selected the actual clock frequency is 12Mhz (48Mhz / 4). 0 = Select external 14.3Mhz input as source. **(default)** 1 = Select internally generated 14.3Mhz clock source. **Note:** bootstrap[5] = 1 can also be used to select internally generated 14.3Mhz clock.[a]	
Offset 08h-0Bh	**I/O Control Register 3 (R/W)**	**Reset Value = 00009000h**
31:16	**Reserved**	
15:13	**USB Voltage Adjustment Connection (IO_USB_XCVR_VADJ)** — These bits connect to the voltage adjustment interface on the three USB transceivers. Default = 100.	
12:8	**USB Current Adjustment (IO_USB_XCVT_CADJ)** — These bits connect to the current adjustment interface on the three USB transceivers. Default value = 10000.	
7:0	**Reserved**.	

a. See Table 5.42 "Composite BootStrap Register Map" on page 432

4.4.4. USB Controller Registers - PCIUSB

The registers designated as PCIUSB are 32-bit registers decoded from the PCI address bits 7 through 2 and C/BE[3:0]#, when IDSEL is high, AD[10:8] select the appropriate function, and AD[1:0] are '00'. Bytes within a 32-bit address are selected with the valid byte enables. All registers can be accessed via 8-, 16-, or 32-bit cycles (that is, each byte is individually selected by the byte enables.) Registers marked as reserved, and reserved bits within a register are not implemented and should return 0s when read. Writes have no effect for reserved registers.

Table 4.38 gives the bit formats for the USB controller's PCI header registers. For complete register/bit formats, refer to Revision 1.0 of the OpenHCI Specification.

Table 4.38 PCIUSB: USB Controller Registers

Bit	Description
Index 00h-01h	**Vendor Identification Register (RO) — Reset Value = 0E11h**
Index 02h-03h	**Device Identification Register (RO) — Reset Value = A0F8h**
Index 04h-05h	**Command Register (R/W) — Reset Value = 00h**
15:10	**Reserved** — Set to 0.
9	**Fast Back-to-Back Enable (Read Only)** — USB only acts as a master to a single device, so this functionality is not needed. It is always disabled (must always be set to 0).
8	**SERR#** — USB asserts SERR# when it detects an address parity error: 0 = Disable; 1 = Enable.
7	**Wait Cycle Control** — USB does not need to insert a wait state between the address and data on the AD lines. It is always disabled (bit is set to 0).
6	**Parity Error** — USB asserts PERR# when it is the agent receiving data, and it detects a data parity error: 0 = Disable; 1 = Enable.
5	**VGA Palette Snoop Enable (Read Only)** — USB does not support this function. It is always disabled (bit is set to 0).
4	**Memory Write and Invalidate** — Allow USB to run Memory Write and Invalidate commands: 0 = Disable; 1 = Enable. The Memory Write and Invalidate Command only occurs if the cacheline size is set to 32 bytes and the memory write is exactly one cache line. This bit should be left = 0.
3	**Special Cycles** — USB does not run special cycles on PCI. It is always disabled (bit is set to 0).
2	**PCI Master Enable** — Allow USB to run PCI master cycles: 0 = Disable; 1 = Enable.
1	**Memory Space** — Allow USB to respond as a target to memory cycles: 0 = Disable; 1 = Enable.
0	**I/O Space** — Allow USB to respond as a target to I/O cycles: 0 = Disable; 1 = Enable.
Index 06h-07h	**Status Register (R/W) — Reset Value = 0280h**
15	**Detected Parity Error** — This bit is set whenever the USB detects a parity error, even if the Parity Error (Response) Detection Enable Bit (Command Register, bit 6) is disabled. Write 1 to clear.
14	**SERR# Status** — This bit is set whenever the USB detects a PCI address error. Write 1 to clear.
13	**Received Master Abort Status** — This bit is set when the USB, acting as a PCI master, aborts a PCI bus memory cycle. Write 1 to clear.
12	**Received Target Abort Status** — This bit is set when a USB generated PCI cycle (USB is the PCI master) is aborted by a PCI target. Write 1 to clear.
11	**Signaled Target Abort Status** — This bit is set whenever the USB signals a target abort. Write 1 to clear.
10:9	**DEVSEL# Timing (Read Only)** — These bits indicate the DEVSEL# timing when performing a positive decode. Since DEVSEL# is asserted to meet the medium timing, these bits are encoded as 01b.

Table 4.38 PCIUSB: USB Controller Registers

Bit	Description
8	**Data Parity Reported** — Set to 1 if the Parity Error Response bit (Command Register bit 6) is set, and USB detects PERR# asserted while acting as PCI master (whether PERR# was driven by USB or not).
7	**Fast Back-to-Back Capable** — USB does support fast back-to-back transactions when the transactions are not to the same agent. This bit is always 1.
6:0	**Reserved** — Set to 0.
Note: The PCI Specification defines this register to record status information for PCI related events. This is a read/write register. However, writes can only reset bits. A bit is reset whenever the register is written, and the data in the corresponding bit location is a 1.	
Index 08h	**Device Revision ID Register (RO)** — **Reset Value = 07h**
Index 09h-0Bh	**PCI Class Code Register (RO)** — **Reset Value = 0C0310h**
	This register identifies the generic function of USB the specific register level programming interface. The Base Class is 0Ch (Serial Bus Controller). The Sub Class is 03h (Universal Serial Bus). The Programming Interface is 10h (OpenHCI).
Index 0Ch	**Cache Line Size Register (R/W)** — **Reset Value = 00h**
	This register identifies the system cacheline size in units of 32-bit words. USB will only store the value of bit 3 in this register since the cacheline size of 32 bytes is the only value applicable to the design. Any value other than 08h written to this register will be read back as 00h.
Index 0Dh	**Latency Timer Register (R/W)** — **Reset Value = 00h**
	This register identifies the value of the latency timer in PCI clocks for PCI bus master cycles.
Index 0Eh	**Header Type Register (RO)** — **Reset Value = 00h**
	This register identifies the type of the predefined header in the configuration space. Since USB is a single function device and not a PCI-to-PCI bridge, this byte should be read as 00h.
Index 0Fh	**BIST Register (RO)** — **Reset Value = 00h**
	This register identifies the control and status of Built In Self Test. USB does not implement BIST, so this register is read only.
Index 10h-13h	**Base Address Register (R/W)** — **Reset Value = 00000000h**
31:12	**Base Address** — POST writes the value of the memory base address to this register.
11:4	**Always 0** — Indicates a 4 KB address range is requested.
3	**Always 0** — Indicates there is no support for prefetchable memory.
2:1	**Always 0** — Indicates that the base register is 32-bits wide and can be placed anywhere in 32-bit memory space.
0	**Always 0** — Indicates that the operational registers are mapped into memory space.
Index 14h-2Bh	**Reserved**
Index 2Ch-2Dh	**Subsystem Vendor ID (R/W)** — **Reset Value = 0E11h**
Index 2Eh-2Fh	**Subsystem ID (R/W)** — **Reset Value = A0F8h**
Index 30h-3Bh	**Reserved**
Index 3Ch	**Interrupt Line Register (R/W)** — **Reset Value = 00h**
	This register identifies to which of the system interrupt controllers the devices interrupt pin is connected. The value of this register is used by device drivers and has no direct meaning to USB.

Table 4.38 PCIUSB: USB Controller Registers

Bit	Description
Index 3Dh	**Interrupt Pin Register (RO)** **Reset Value = 01h**
	This register identifies which interrupt pin a device uses. Since USB uses INTA#, this value is set to 01h.
Index 3Eh	**Min. Grant Register (RO)** **Reset Value = 00h**
	This register specifies the desired settings for how long of a burst USB needs assuming a clock rate of 33 MHz. The value specifies a period of time in units of 1/4 microsecond.
Index 3Fh	**Max. Latency Register (RO)** **Reset Value = 50h**
	This register specifies the desired settings for how often USB needs access to the PCI bus assuming a clock rate of 33 MHz. The value specifies a period of time in units of 1/4 microsecond.
Index 40h-43h	**ASIC Test Mode Enable Register (R/W)** **Reset Value = 000F0000h**
	Used for internal debug and test purposes only.
Index 44h	**ASIC Operational Mode Enable Register (R/W)** **Reset Value = 00h**
7:1	**Write Only** — Read as 0s.
0	**Data Buffer Region 16** — When set the size of the region for the data buffer is 16 bytes. Otherwise, the size is 32 bytes.
Index 45h-FFh	**Reserved**

4.4.5. ISA Legacy Register Space

The ISA Legacy registers reside in the ISA I/O address space in the address range from 000h to FFFh and are accessed through typical input/output instructions (that is, CPU direct R/W) with the designated I/O port address and 8-bit data.

The bit formats for the ISA Legacy I/O Registers plus two chipset-specific configuration registers used for interrupt mapping in the South Bridge module core logic are given in this section. The ISA Legacy registers are separated into the following categories:

- DMA Channel Control Registers, see Table 4.39
- DMA Page Registers, see Table 4.40
- Programmable Interval Timer Registers, see Table 4.41
- Programmable Interrupt Controller Registers, see Table 4.42
- Keyboard Controller Registers, see Table 4.43
- Real Time Clock Registers, see Table 4.44
- Miscellaneous Registers, see Table 4.45 (includes 4D0h and 4D1h Interrupt Edge/Level Select Registers)

Table 4.39 DMA Channel Control Registers

Bit	Description
I/O Port 000h (R/W)	**DMA Channel 0 Address Register**
	Written as two successive bytes, byte 0, 1.
I/O Port 001h (R/W)	**DMA Channel 0 Transfer Count Register**
	Written as two successive bytes, byte 0, 1.

Table 4.39 DMA Channel Control Registers (cont.)

Bit	Description
I/O Port 002h (R/W)	**DMA Channel 1 Address Register**
	Written as two successive bytes, byte 0, 1.
I/O Port 003h (R/W)	**DMA Channel 1 Transfer Count Register**
	Written as two successive bytes, byte 0, 1.
I/O Port 004h (R/W)	**DMA Channel 2 Address Register**
	Written as two successive bytes, byte 0, 1.
I/O Port 005h (R/W)	**DMA Channel 2 Transfer Count Register**
	Written as two successive bytes, byte 0, 1.
I/O Port 006h (R/W)	**DMA Channel 3 Address Register**
	Written as two successive bytes, byte 0, 1.
I/O Port 007h (R/W)	**DMA Channel 3 Transfer Count Register**
	Written as two successive bytes, byte 0, 1.
I/O Port 008h (R/W)	
Read	**DMA Status Register, Channels 3:0**
7	**Channel 3 Request** — Request pending? 0 = No; 1 = Yes.
6	**Channel 2 Request** — Request pending? 0 = No; 1 = Yes.
5	**Channel 1 Request** — Request pending? 0 = No; 1 = Yes.
4	**Channel 0 Request** — Request pending? 0 = No; 1 = Yes.
3	**Channel 3 Terminal Count** — TC reached? 0 = No; 1 = Yes.
2	**Channel 2 Terminal Count** — TC reached? 0 = No; 1 = Yes.
1	**Channel 1 Terminal Count** — TC reached? 0 = No; 1 = Yes.
0	**Channel 0 Terminal Count** — TC reached? 0 = No; 1 = Yes.
Write	**DMA Command Register, Channels 3:0**
7	**DACK Sense** — 0 = Active high; 1 = Active low.
6	**DREQ Sense** — 0 = Active high; 1 = Active low.
5	**Write Selection** — 0 = Late write; 1 = Extended write.
4	**Priority Mode** — 0 = Fixed; 1 = Rotating.
3	**Timing Mode** — 0 = Normal; 1 = Compressed.
2	**Channels 3:0** — 0 = Disable; 1 = Enable.
1:0	**Reserved** — Set to 0.
I/O Port 009h (W)	**Software DMA Request Register, Channels 3:0**
7:3	**Reserved** — Set to 0.
2	**Request Type** — 0 = Reset; 1 = Set.
1:0	**Channel Number Request Select** — 00 = Channel 0; 01 = Channel 1; 10 = Channel 2; 11 = Channel 3.
I/O Port 00Ah (R/W)	**DMA Channel Mask Register, Channels 3:0**
7:3	**Reserved** — Set to 0.
2	**Channel Mask** — 0 = Not masked; 1 = Masked

Table 4.39 DMA Channel Control Registers (cont.)

Bit	Description
1:0	**Channel Number Mask Select** — 00 = Channel 0; 01 = Channel 1; 10 = Channel 2; 11 = Channel 3
I/O Port 00Bh (W)	**DMA Channel Mode Register, Channels 3:0**
7:6	**Transfer Mode** — 00 = Demand; 01 = Single; 10 = Block; 11 = Cascade.
5	**Address Direction** — 0 = Increment; 1 = Decrement.
4	**Auto-initialize** — 0 = Disable; 1 = Enable.
3:2	**Transfer Type** — 00 = Verify; 01 = Memory read; 10 = Memory write; 11 = Reserved.
1:0	**Channel Number Mode Select** — 00 = Channel 0; 01 = Channel 1; 10 = Channel 2; 11 = Channel 3.
I/O Port 00Ch (W)	**DMA Clear Byte Pointer Command, Channels 3:0**
I/O Port 00Dh (W)	**DMA Master Clear Command, Channels 3:0**
I/O Port 00Eh (W)	**DMA Clear Mask Register Command, Channels 3:0**
I/O Port 00Fh (W)	**DMA Write Mask Register Command, Channels 3:0**
I/O Port 0C0h (R/W)	**DMA Channel 4 Address Register**
Not used.	
I/O Port 0C2h (R/W)	**DMA Channel 4 Transfer Count Register**
Not used.	
I/O Port 0C4h (R/W)	**DMA Channel 5 Address Register**
Memory address bytes 1 and 0.	
I/O Port 0C6h (R/W)	**DMA Channel 5 Transfer Count Register**
Transfer count bytes 1 and 0	
I/O Port 0C8h (R/W)	**DMA Channel 6 Address Register**
Memory address bytes 1 and 0.	
I/O Port 0CAh (R/W)	**DMA Channel 6 Transfer Count Register**
Transfer count bytes 1 and 0.	
I/O Port 0CCh (R/W)	**DMA Channel 7 Address Register**
Memory address bytes 1 and 0.	
I/O Port 0CEh (R/W)	**DMA Channel 7 Transfer Count Register**
Transfer count bytes 1 and 0.	

Table 4.39 DMA Channel Control Registers (cont.)

Bit	Description
I/O Port 0D0h (R/W)	
Read	**DMA Status Register, Channels 7:4**
7	**Channel 7 Request** — Request pending? 0 = No; 1 = Yes.
6	**Channel 6 Request** — Request pending? 0 = No; 1 = Yes.
5	**Channel 5 Request** — Request pending? 0 = No; 1 = Yes.
4	**Undefined**
3	**Channel 7 Terminal Count** — TC reached? 0 = No; 1 = Yes.
2	**Channel 6 Terminal Count** — TC reached? 0 = No; 1 = Yes.
1	**Channel 5 Terminal Count** — TC reached? 0 = No; 1 = Yes.
0	**Undefined**
Write	**DMA Command Register, Channels 7:4**
7	**DACK Sense** — 0 = Active high; 1 = Active low.
6	**DREQ Sense** — 0 = Active high; 1 = Active low.
5	**Write Selection** — 0 = Late write; 1 = Extended write.
4	**Priority Mode** — 0 = Fixed; 1 = Rotating.
3	**Timing Mode** — 0 = Normal; 1 = Compressed.
2	**Channels 7:4** — 0 = Disable; 1 = Enable.
1:0	**Reserved** — Set to 0.
I/O Port 0D2h (W)	**Software DMA Request Register, Channels 7:4**
7:3	**Reserved** — Set to 0.
2	**Request Type** — 0 = Reset; 1 = Set.
1:0	**Channel Number Request Select** — 00 = Illegal; 01 = Channel 5; 10 = Channel 6; 11 = Channel 7.
I/O Port 0D4h (R/W)	**DMA Channel Mask Register, Channels 7:0**
7:3	**Reserved** — Set to 0.
2	**Channel Mask** — 0 = Not masked; 1 = Masked.
1:0	**Channel Number Mask Select** — 00 = Channel 4; 01 = Channel 5; 10 = Channel 6; 11 = Channel 7.
I/O Port 0D6h (W)	**DMA Channel Mode Register, Channels 7:4**
7:6	**Transfer Mode** — 00 = Demand; 01 = Single; 10 = Block; 11 = Cascade.
5	**Address Direction** — 0 = Increment; 1 = Decrement.
4	**Auto-initialize** — 0 = Disabled; 1 = Enable.
3:2	**Transfer Type** — 00 = Verify; 01 = Memory read; 10 = Memory write; 11 = Reserved.
1:0	**Channel Number Mode Select** — 00 = Channel 4; 01 = Channel 5; 10 = Channel 6; 11 = Channel 7. Channel 4 must be programmed in cascade mode. This mode is not the default.
I/O Port 0D8h (W)	**DMA Clear Byte Pointer Command, Channels 7:4**
I/O Port 0DAh (W)	**DMA Master Clear Command, Channels 7:4**
I/O Port 0DCh (W)	**DMA Clear Mask Register Command, Channels 7:4**
I/O Port 0DEh (W)	**DMA Write Mask Register Command, Channels 7:4**

Table 4.40 DMA Page Registers

Bit	Description
I/O Port 081h (R/W)	**DMA Channel 2 Low Page Register** Address bits [23:16] (byte 2).
I/O Port 082h (R/W)	**DMA Channel 3 Low Page Register** Address bits [23:16] (byte 2).
I/O Port 083h (R/W)	**DMA Channel 1 Low Page Register** Address bits [23:16] (byte 2).
I/O Port 087h (R/W)	**DMA Channel 0 Low Page Register** Address bits [23:16] (byte 2).
I/O Port 089h (R/W)	**DMA Channel 6 Low Page Register** Address bits [23:16] (byte 2).
I/O Port 08Ah (R/W)	**DMA Channel 7 Low Page Register** Address bits [23:16] (byte 2).
I/O Port 08Bh (R/W)	**DMA Channel 5 Low Page Register** Address bits [23:16] (byte 2).
I/O Port 08Fh (R/W)	**ISA Refresh Low Page Register** Refresh address.
I/O Port 481h (R/W)	**DMA Channel 2 High Page Register** Address bits [31:24] (byte 3). **Note:** This register is reset to 00h on any access to Port 081h.
I/O Port 482h (R/W)	**DMA Channel 3 High Page Register** Address bits [31:24] (byte 3). **Note:** This register is reset to 00h on any access to Port 082h.
I/O Port 483h (R/W)	**DMA Channel 1 High Page Register** Address bits [31:24] (byte 3). **Note:** This register is reset to 00h on any access to Port 083h.
I/O Port 487h (R/W)	**DMA Channel 0 High Page Register** Address bits [31:24] (byte 3). **Note:** This register is reset to 00h on any access to Port 087h.
I/O Port 489h (R/W)	**DMA Channel 6 High Page Register** Address bits [31:24] (byte 3). **Note:** This register is reset to 00h on any access to Port 089h.

Table 4.40 DMA Page Registers (cont.)

Bit	Description
I/O Port 48Ah (R/W)	**DMA Channel 7 High Page Register**
	Address bits [31:24] (byte 3). **Note:** This register is reset to 00h on any access to Port 08Ah.
I/O Port 48Bh (R/W)	**DMA Channel 5 High Page Register**
	Address bits [31:24] (byte 3). **Note:** This register is reset to 00h on any access to Port 08Bh.

Table 4.41 Programmable Interval Timer Registers

Bit	Description
I/O Port 040h	
Write	**PIT Timer 0 Counter**
7:0	Counter Value
Read	**PIT Timer 0 Status**
7	**Counter Output** — State of counter output signal.
6	**Counter Loaded** — Last count written is loaded? 0 = Yes; 1 = No.
5:4	**Current Read/Write Mode** — 00 = Counter latch command; 01 = R/W LSB only; 10 = R/W MSB only; 11 = R/W LSB, followed by MSB.
3:1	**Current Counter Mode** — 0-5.
0	**BCD mode** — 0 = Binary; 1 = BCD (binary coded decimal).
I/O Port 041h	
Write	**PIT Timer 1 Counter (Refresh)**
7:0	Counter Value
Read	**PIT Timer 1 Status (Refresh)**
7	**Counter Output** — State of counter output signal.
6	**Counter Loaded** — Last count written is loaded? 0 = Yes; 1 = No.
5:4	**Current Read/Write Mode** — 00 = Counter latch command; 01 = R/W LSB only; 10 = R/W MSB only; 11 = R/W LSB, followed by MSB.
3:1	**Current Counter Mode** — 0-5.
0	**BCD mode** — 0 = Binary; 1 = BCD (binary coded decimal).
I/O Port 042h	
Write	**PIT Timer 2 Counter (Speaker)**
7:0	Counter Value
Read	**PIT Timer 2 Status (Speaker)**
7	**Counter Output** — State of counter output signal.
6	**Counter Loaded** — Last count written is loaded? 0 = Yes; 1 = No.
5:4	**Current Read/Write Mode** — 00 = Counter latch command; 01 = R/W LSB only; 10 = R/W MSB only; 11 = R/W LSB, followed by MSB.
3:1	**Current Counter Mode** — 0-5.

Table 4.41 Programmable Interval Timer Registers (cont.)

Bit	Description
0	**BCD mode** — 0 = Binary; 1 = BCD (binary coded decimal)
I/O Port 043h (R/W)	**PIT Mode Control Word Register**
7:6	**Counter Select** — 00 = Counter 0; 01 = Counter 1; 10 = Counter 2; 11 = Read-back command (Note 1).
5:4	**Current Read/Write Mode** — 00 = Counter latch command (Note 2); 01 = R/W LSB only; 10 = R/W MSB only; 11 = R/W LSB, followed by MSB.
3:1	**Current Counter Mode** — 0-5.
0	**BCD mode** — 0 = Binary; 1 = BCD (binary coded decimal).
Notes: 1. If bits [7:6] = 11: Register functions as Read Status Command and: Bit 5 = Latch Count Bit 4 = Latch Status Bit 3 = Select Counter 2 Bit 2 = Select Counter 1 Bit 1 = Select Counter 0 Bit 0 = Reserved 2. If bits [5:4] = 00: Register functions as Counter Latch Command and: Bits [7:6] = Selects Counter Bits [3:0] = Don't care	

Table 4.42 Programmable Interrupt Controller Registers

Bit	Description
I/O Port 020h / 0A0h (WO)	**Master / Slave PIC IWC1**
7:5	**Reserved** — Set to 0.
4	**Reserved** — Set to 1.
3	**Trigger Mode** — 0 = Edge; 1 = Level.
2	**Vector Address Interval** — 0 = 8 byte intervals; 1 = 4 byte intervals.
1	**Reserved** — Set to 0 (cascade mode).
0	**Reserved** — Set to 1 (ICW4 must be programmed).
I/O Port 021h / 0A1h (WO)	**Master / Slave PIC ICW2 (after ICW1 is written)**
7:3	**A[7:3]** — Address lines 7:3 for base vector for interrupt controller.
2:0	**Reserved** — Set to 0.
I/O Port 021h / 0A1h (WO)	**Master / Slave PIC ICW3 (after ICW2 is written)**
Master PIC ICW3	
7:0	**Cascade IRQ** — Must be 04h.
Slave PIC ICW3	
7:0	**Slave ID** — Must be 02h.
I/O Port 021h / 0A1h (WO)	**Master / Slave PIC ICW4 (after ICW3 is written)**
7:5	**Reserved** — Set to 0.
4	**Special Fully Nested Mode** — 0 = Disable; 1 = Enable.
3:2	**Reserved** — Set to 0.
1	**Auto EOI** — 0 = Normal EOI; 1 = Auto EOI.

Table 4.42 Programmable Interrupt Controller Registers (cont.)

Bit	Description
0	**Reserved** — Set to 1 (8086/8088 mode).
I/O Port 021h / 0A1h (R/W)	**Master / Slave PIC OCW1 (except immediately after ICW1 is written)**
7	**IRQ7 / IRQ15 Mask** — 0 = Not Masked; 1 = Mask.
6	**IRQ6 / IRQ14 Mask** — 0 = Not Masked; 1 = Mask.
5	**IRQ5 / IRQ13 Mask** — 0 = Not Masked; 1 = Mask.
4	**IRQ4 / IRQ12 Mask** — 0 = Not Masked; 1 = Mask.
3	**IRQ3 / IRQ11 Mask** — 0 = Not Masked; 1 = Mask.
2	**IRQ2 / IRQ10 Mask** — 0 = Not Masked; 1 = Mask.
1	**IRQ1 / IRQ9 Mask** — 0 = Not Masked; 1 = Mask.
0	**IRQ0 / IRQ8 Mask** — 0 = Not Masked; 1 = Mask.
I/O Port 020h / 0A0h (WO)	**Master / Slave PIC OCW2**
7:5	**Rotate/EOI Codes** 000 = Clear rotate in Auto EOI mode 001 = Non-specific EOI 010 = No operation 011 = Specific EOI (bits [2:0] must be valid) 100 = Set rotate in Auto EOI mode 101 = Rotate on non-specific EOI command 110 = Set priority command (bits [2:0] must be valid) 111 = Rotate on specific EOI command
4:3	**Reserved** — Set to 0.
2:0	**IRQ number (000-111)**
I/O Port 020h / 0A0h (WO)	**Master / Slave PIC OCW3**
7	**Reserved** — Set to 0.
6:5	**Special Mask Mode** 00 = No operation 01 = No operation 10 = Reset Special Mask Mode 11 = Set Special Mask Mode
4	**Reserved** — Set to 0.
3	**Reserved** — Set to 1.
2	**Poll Command** — 0 = Disable; 1 = Enable.
1:0	**Register Read Mode** 00 = No operation 01 = No operation 10 = Read interrupt request register on next read of Port 20h 11 = Read interrupt service register on next read of Port 20h.
I/O Port 020h / 0A0h (RO)	**Master / Slave PIC Interrupt Request and Service Registers for OCW3 Commands**
Interrupt Request Register	
7	**IRQ7 / IRQ15 Pending** — 0 = Yes; 1 = No.
6	**IRQ6 / IRQ14 Pending** — 0 = Yes; 1 = No.
5	**IRQ5 / IRQ13 Pending** — 0 = Yes; 1 = No.
4	**IRQ4 / IRQ12 Pending** — 0 = Yes; 1 = No.
3	**IRQ3 / IRQ11 Pending** — 0 = Yes; 1 = No.
2	**IRQ2 / IRQ10 Pending** — 0 = Yes; 1 = No.
1	**IRQ1 / IRQ9 Pending** — 0 = Yes; 1 = No.
0	**IRQ0 / IRQ8 Pending** — 0 = Yes; 1 = No.

Table 4.42 Programmable Interrupt Controller Registers (cont.)

Bit	Description
Interrupt Service Register	
7	**IRQ7 / IRQ15 In-Service** — 0 = No; 1 = Yes.
6	**IRQ6 / IRQ14 In-Service** — 0 = No; 1 = Yes.
5	**IRQ5 / IRQ13 In-Service** — 0 = No; 1 = Yes.
4	**IRQ4 / IRQ12 In-Service** — 0 = No; 1 = Yes.
3	**IRQ3 / IRQ11 In-Service** — 0 = No; 1 = Yes.
2	**IRQ2 / IRQ10 In-Service** — 0 = No; 1 = Yes.
1	**IRQ1 / IRQ9 In-Service** — 0 = No; 1 = Yes.
0	**IRQ0 / IRQ8 In-Service** — 0 = No; 1 = Yes.
Note: The function of this register is set with bits 1:0 in a write to 020h.	

Table 4.43 Keyboard Controller Registers

Bit	Description
I/O Port 060h (R/W) **External Keyboard Controller Data Register**	
Keyboard Controller Data Register — All accesses to this port are passed to the ISA bus. If the fast keyboard gate A20 and reset features are enabled through bit 7 of the ROM/AT Logic Control Register (F0 Index 52h[7]), the respective sequences of writes to this port assert the A20M# pin or cause a warm CPU reset.	
I/O Port 061h (R/W) **Port B Control Register** **Reset Value = 00x01100b**	
7	**PERR#/SERR# Status (Read Only)** — Was a PCI bus error (PERR#/ SERR#) asserted by PCI device? 0 = No; 1=Yes. This bit can only be set if ERR_EN is set 0. Set this bit to 0 after a write to ERR_EN with a 1 or after reset.
6	**IOCHK# Status (Read Only)** — Is an I/O device reporting an error? 0 = No; 1 = Yes. This bit can only be set if IOCHK_EN is set 0. This bit is set 0 after a write to IOCHK_EN with a 1 or after reset.
5	**PIT OUT2 State (Read Only)** — Reflects the current status of the PIT Timer2-OUT2.
4	**Toggle (Read Only)** — Toggles on every falling edge of Counter 1 (OUT1).
3	**IOCHK Enable** 0 = Generates an NMI if IOCHK# is driven low by an I/O device to report an error. Note that NMI is under SMI control. 1 = Ignores the IOCHK# input signal and does not generate NMI.
2	**PERR#/ SERR# Enable** — Generate an NMI if PERR#/ SERR# is driven active to report an error: 0 = Enable; 1 = Disable
1	**PIT Counter2 (SPKR)** — 0 = Forces Counter 2 output (OUT2) to zero. 1 = Allows Counter 2 output (OUT2) to pass to the speaker
0	**PIT Counter2 Enable** — 0 = Sets GATE2 input low. 1 = Sets GATE2 input high.
I/O Port 062h (R/W) **External Keyboard Controller Mailbox Register**	
Keyboard Controller Mailbox Register — Accesses to this port will assert ROMCS# if the Port 062h/066h decode is enabled through bit 7 of the Decode Control Register 2 (F0 Index 5Bh[7]).	
I/O Port 064h (R/W) **External Keyboard Controller Command Register**	
Keyboard Controller Command Register — All accesses to this port are passed to the ISA bus. If the fast keyboard gate A20 and reset features are enabled through bit 7 of the ROM/AT Logic Control Register (F0 Index 52h[7]), the respective sequences of writes to this port assert the A20M# pin or cause a warm CPU reset.	

Table 4.43 Keyboard Controller Registers (cont.)

Bit	Description
I/O Port 066h (R/W)	**External Keyboard Controller Mailbox Register**
	Keyboard Controller Mailbox Register — Accesses to this port will assert KBROMCS# if the Port 062h/066h decode is enabled through bit 7 of the Decode Control Register 2 (F0 Index 5Bh[7]).
I/O Port 092h (R/W)	**Port A Control Register** **Reset Value = 02h**
7:2	**Reserved** — Set to 0.
1	**A20M# SMI Assertion** — Assert A20# SMI: 0 = Enable; 1 = Disable.
0	**Fast CPU Reset** — WM_RST SMI is asserted to the BIOS: 0 = Disable; 1 = Enable. Clear this bit before the generation of another reset.

Table 4.44 Real-Time Clock Registers

Bit	Description
I/O Port 070h (WO)	**RTC Address Register**
7	**NMI Mask** — 0 = Enable; 1 = Mask.
6:0	**RTC Register Index** — A write of this register sends the data out on the ISA bus.
	Note: This register is shadowed within the South Bridge module and is read through the RTC Shadow Register (PCIDV2F1 10h+Memory Offset BBh).
I/O Port 071h (R/W)	**RTC Data Register**
	A read of this register returns the value of the register indexed by the RTC Address Register. A write of this register sets the value into the register indexed by the RTC Address Register.

Table 4.45 Miscellaneous Registers

Bit	Description
I/O Port 0F0h, 0F1h	**Coprocessor Error Register (W)** **Reset Value = F0h**
	A write to either port when the FERR# signal is asserted causes the South Bridge module to assert IGNNE#. IGNNE# remains asserted until the FERR# deasserts.
I/O Ports 170h-177h/376h-377h	**Secondary IDE Registers (R/W)**
	When the local IDE functions are enabled, reads or writes to these registers cause the local IDE interface signals to operate according to their configuration rather than generating standard ISA bus cycles.
I/O Ports 1F0h-1F7h/3F6h-3F7h	**Primary IDE Registers (R/W)**
	When the local IDE functions are enabled, reads or writes to these registers cause the local IDE interface signals to operate according to their configuration rather than generating standard ISA bus cycles.
I/O Port 4D0h	**Interrupt Edge/Level Select Register 1 (R/W)** **Reset Value = 00h**
7	**IRQ7 Edge or Level Sensitive Select** — Selects PIC IRQ7 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.
6	**IRQ6 Edge or Level Sensitive Select** — Selects PIC IRQ6 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.

Table 4.45 Miscellaneous Registers (cont.)

Bit	Description
5	**IRQ5 Edge or Level Sensitive Select** — Selects PIC IRQ5 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.
4	**IRQ4 Edge or Level Sensitive Select** — Selects PIC IRQ4 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.
3	**IRQ3 Edge or Level Sensitive Select** — Selects PIC IRQ3 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.
2:0	**Reserved** — Set to 0.
Notes: 1. If ICW1 - bit 3 in the PIC is set as level, it overrides this setting. See I/O Port 020h/0A0h. 2. This bit is provided to configure a PCI interrupt mapped to IRQ[x] on the PIC as level-sensitive (shared).	
I/O Port 4D1h	**Interrupt Edge/Level Select Register 2 (R/W)** **Reset Value = 00h**
7	**IRQ15 Edge or Level Sensitive Select** — Selects PIC IRQ15 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.
6	**IRQ14 Edge or Level Sensitive Select** — Selects PIC IRQ14 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.
5	**Reserved** — Set to 0.
4	**IRQ12 Edge or Level Sensitive Select** — Selects PIC IRQ12 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.
3	**IRQ11 Edge or Level Sensitive Select** — Selects PIC IRQ11 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.
2	**IRQ10 Edge or Level Sensitive Select** — Selects PIC IRQ10 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.
1	**IRQ9 Edge or Level Sensitive Select** — Selects PIC IRQ9 sensitivity configuration: 0 = Edge; 1 = Level. Notes 1 and 2.
0	**Reserved** — Set to 0.
Notes: 1. If ICW1 - bit 3 in the PIC is set as level, it overrides this setting. 2. This bit is provided to configure a PCI interrupt mapped to IRQ[x] on the PIC as level-sensitive (shared).	

4.5. SuperI/O - A PC98 Compliant Cell

The SuperI/O is a PC98 compliant component that offers a complete solution to the most commonly used ISA peripherals.

The SuperI/O incorporates: a Floppy Disk Controller (FDC), two enhanced Serial Ports, an Infrared Communication Port that supports FIR, MIR, HP-SIR, Sharp-IR, and Consumer Electronics-IR, a full IEEE 1284 Parallel Port, an ACCESS.bus Interface (ACB), a Keyboard and Mouse Controller (KBC), System Wake-Up Control (SWC), a Real-Time Clock (RTC) that provides both RTC timekeeping and Advanced Power Control (APC) functionality.

4.5.1. Outstanding Features

- System Wake-Up Control generates a power-up request in response to pre programmed keyboard or mouse sequence, modem, telephone ring, and two general-purpose events
- Programmable write protect for Floppy Disk Controller
- Advanced RTC and APC, Y2K compliant

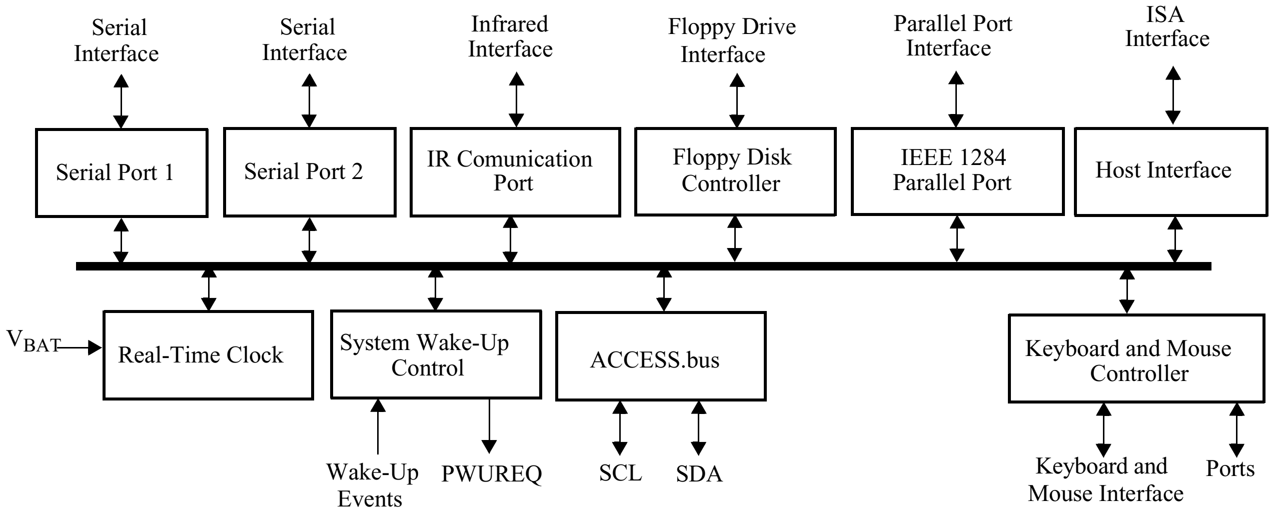


Figure 4-4 Super I/O Block Diagram

4.5.2. Features

- PC98 Compliant
  - — PnP Configuration Register structure
  - — Flexible resource allocation for all logical devices
    - — Relocatable base address
    - — 9 Parallel IRQ routing options
    - — 3 optional 8-bit DMA channels (where applicable)
- Floppy Disk Controller (FDC)
  - — Programmable write protect
  - — FM and MFM mode support
  - — Enhanced mode command for three-mode Floppy Disk Drive (FDD) support
  - — Perpendicular recording drive support for 2.88 MB
  - — Burst and non-burst modes
  - — Full support for IBM Tape Drive register (TDR) implementation of AT and PS/2 drive types
  - — 16-byte FIFO
  - — Software compatible with the PC8477, which contains a superset of the FDC functions in the microDP8473, the NEC microPD765A and the N82077
  - — High-performance, digital separator
  - — Standard 5.25” and 3.5” FDD support
- Parallel Port
  - — Software or hardware control
  - — Enhanced Parallel Port (EPP) compatible with new version EPP 1.9 and IEEE 1284 compliant
  - — EPP support for version EPP 1.7 of the Xircom specification
  - — EPP support as mode 4 of the Extended Capabilities Port (ECP)
  - — IEEE 1284 compliant ECP, including level 2
  - — Selection of internal pull-up or pull-down resistor for Paper End (PE) pin
  - — PCI bus utilization reduction by supporting a demand DMA mode mechanism and a DMA fairness mechanism

— Protection circuit that prevents damage to the parallel port when a printer connected to it powers up or is operated at high voltages, even if the device is in power-down
— Output buffers that can sink and source 14 mA

- Serial Ports 1 and 2

— Software compatible with the 16550A and the 16450
— Shadow register support for write-only bit monitoring
— UART data rates up to 1.5 Mbps

- Infrared Communication Port

— Data rate of up to 115.2 Kbps (SIR)
— Data rate of 1.152 Mbps (MIR)
— Data rate of 4.0 Mbps (FIR)
— Selectable internal or external modulation/demodulation (Sharp-IR)
— Consumer-IR (TV-Remote) mode
— Software compatible with the 16550A and the 16450
— Shadow register support for write-only bit monitoring
— HP-SIR
— ASK-IR option of SHARP-IR
— DASK-IR option of SHARP-IR
— Consumer Remote Control supports RC-5, RC-6, NEC, RCA and RECS 80
— Non-standard DMA support - 1 or 2 channels

- Keyboard and Mouse Controller (KBC)

— 8-bit microcontroller
— Software compatible with the 8042AH and PC87911 microcontrollers
— 2 KB custom-designed program ROM
— 256 bytes RAM for data
— Four programmable dedicated open-drain I/O lines
— Asynchronous access to two data registers and one status register during normal operation
— Support for both interrupt and polling
— 93 instructions
— 8-bit timer/counter
— Support for binary and BCD arithmetic
— Operation at 8 MHz,12 MHz or 16 MHz (programmable option)
— Can be customized by using the PC87323, which includes a RAM-based KBC as a development platform for KBC code

- System Wake-Up Control (SWC)

— Power-up request upon detection of Keyboard, Mouse, RI1, RI2, RING, PME1 and PME2 activity, as follows:
  — Preprogrammed Keyboard or Mouse sequence
  — External modem ring on serial ports
  — Ring pulse or pulse train on the RING input
  — General-purpose events, PME1 and PME2
— Battery-backed wake-up setup
— Power-fail recovery support

- Real-Time Clock

— A modifiable address that is referenced by a 16-bit programmable register
— 13 IRQ options, with programmable polarity
— DS1287, MC146818 and PC87911 compatibility
— 242 bytes of battery backed up CMOS RAM in two banks
— Selective lock mechanisms for the RTC RAM
— Battery backed up century calendar in days, day of the week, date of month, months, years and century, with automatic leap-year adjustment

- — Battery backed-up time of day in seconds, minutes and hours that allows a 12 or 24 hour format and adjustments for daylight savings time
- — BCD or binary format for time keeping
- — Three different maskable interrupt flags:
  - — Periodic interrupts - At intervals from 122 msec to 500 msec
  - — Time-of-Month alarm - At intervals from once per second to once per Month
  - — Updated Ended Interrupt - Once per second upon completion of update
- — Separate battery pin, 3.0V operation that includes an internal UL protection resistor
- — 2 mA maximum power consumption during power down
- — Double-buffer time registers

- Clock Sources
  - — 48 MHz clock input
  - — On-chip low frequency clock generator for wake-up
  - — 32.768 KHz crystal with an internal frequency multiplier to generates all required internal frequencies

- Y2K Compliant

4.5.3. SIGNAL/PIN Descriptions

Table 4.46 ACCESS.bus Interface (ACB)

Signal	Pin(s)	I/O	Buffer Type	Power Well	Description
SCL_N	AA4	I/O	IN_{SM}/OD_6	V_{DD}	**ACCESS.bus Clock Signal** An internal pull-up is optional, depending upon the ACCESS.bus configuration register.
SDA	AB4	I/O	IN_{SM}/OD_6	V_{DD}	**ACCESS.bus Data Signal** An internal pull-up is optional, depending upon the ACCESS.bus configuration register.

4.5.4. Device Architecture and Configuration

The SuperI/O device comprises a collection of generic functional blocks. Each functional block is described in a separate chapter in this book. However, some parameters in the implementation of each functional block may vary per SuperI/O device. This chapter describes the SuperI/O structure and provides all device specific information, including special implementation of generic blocks, system interface and device configuration.

4.5.4.1. Overview

The SuperI/O is based on 10 logical devices, the host interface, and a central configuration register set, all built around a central, internal 8-bit bus.

The host interface serves as a bridge between the external ISA interface and the internal bus. It supports 8-bit I/O read, 8-bit I/O write and 8-bit DMA transactions, as defined in *Personal Computer Bus Standard P996*.

The central configuration registers are structured as a subset of the Plug and Play Standard Registers, defined in Appendix A of the *Plug and Play ISA Specification* Version 1.0a by Intel and Microsoft. All system resources assigned to the functional blocks (I/O address space, DMA channels, and IRQ lines) are configured in, and managed by, the central configuration register set. In addition, some function-specific parameters are configurable through this unit and distributed to the functional blocks through special control signals.

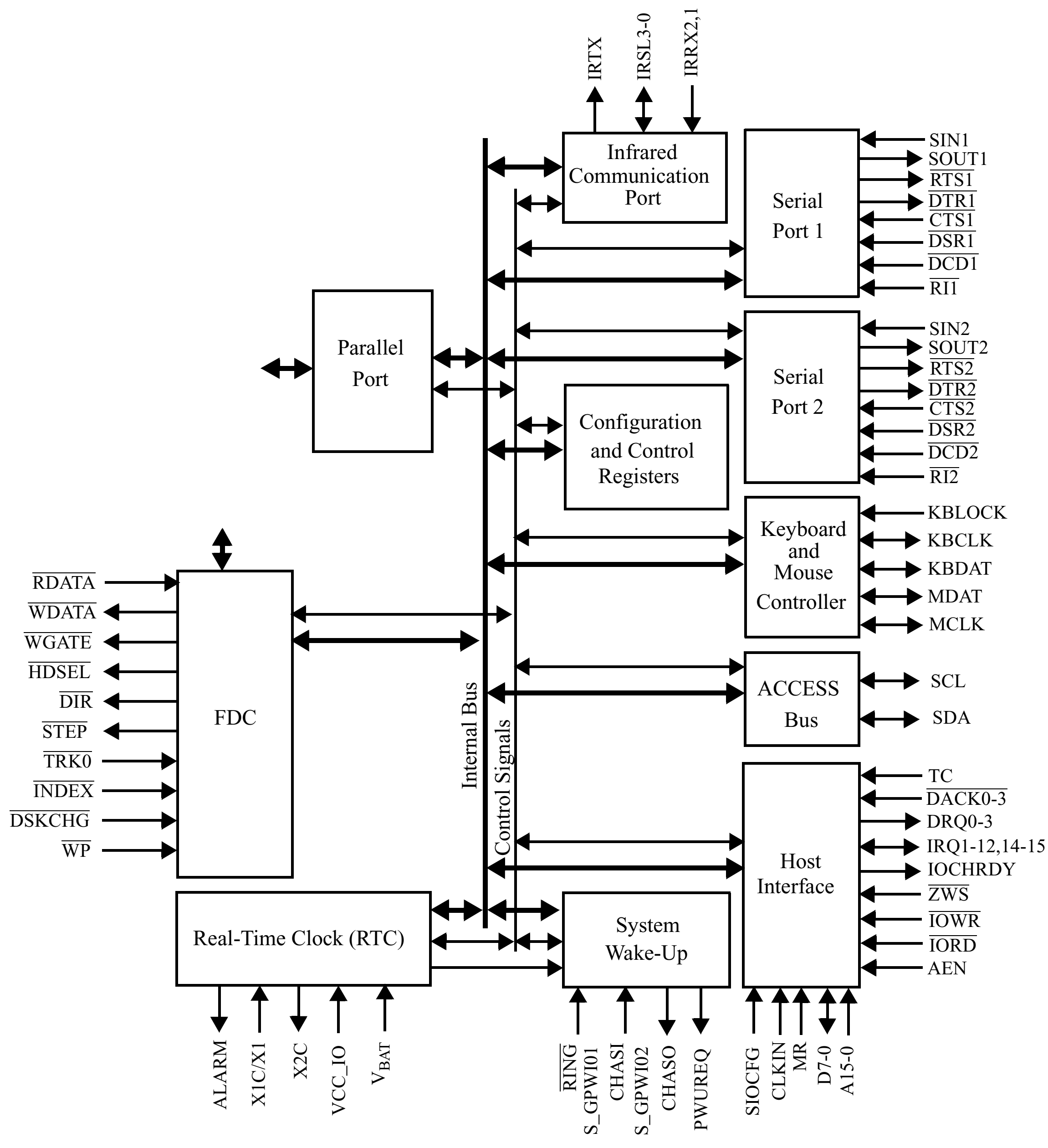


Figure 4-5 Detailed SuperI/O Block Diagram

4.5.4.2. Coinfiguration Structure And Access

This section describes the structure of the configuration register file, and the method of accessing the configuration registers.

4.5.4.3. The Index-Data Register Pair

The SuperI/O configuration access is performed via an Index-Data register pair, using only two system I/O byte locations. The base address of this register pair is determined according to the state on the SIOCFG input pins.

The Index Register is an 8-bit R/W register located at the selected base address (Base+0). It is used as a pointer to the configuration register file, and holds the contents of the configuration register that is currently accessible via the Data Register. Reading the Index Register returns the last value written to the Data Register (or the default of 00h after reset).

The Data Register is an 8-bit virtual register, used as a data path to any configuration register. Accessing the data register results in a physical access of the configuration register that is currently being pointed to by the index register.

Table 4.47 shows the selected base addresses as a function of SIOCFG.

Table 4.47 SuperI/O Configuration Options

SIOCFG F3BAR0 Offset 00h-03h [26:25] Settings	I/O Address		Description
	Index Register	Data Register	
00	-	-	SuperI/O disabled
01	-	-	Configuration access disabled
10	002Eh	002Fh	Base address 1 selected
11	015Ch	015Dh	Base address 2 selected

4.5.4.4. Banked Logical Device Registers

Each functional block is associated with a Logical Device Number (LDN). The configuration registers are grouped into banks, where each bank holds the standard configuration registers of the corresponding logical device. Table 4.47 shows the LDN values of the device functional blocks.

Figure 4-6 shows the structure of the standard configuration register file. The SuperI/O control and configuration registers are not banked and are accessed by the Index-Data register pair only, as described above. However, the device control and device configuration registers are duplicated over 10 banks for 10 logical devices. Therefore, accessing a specific register in a specific bank is performed by two dimensional indexing, where the LDN register selects the bank (or logical device) and the Index register selects the register within the bank.

Accessing the Data register while the Index register holds a value of 30h or higher results in a physical access to the Logical Device Configuration registers currently pointed to by the Index register, within the logical device bank currently selected by the LDN register.

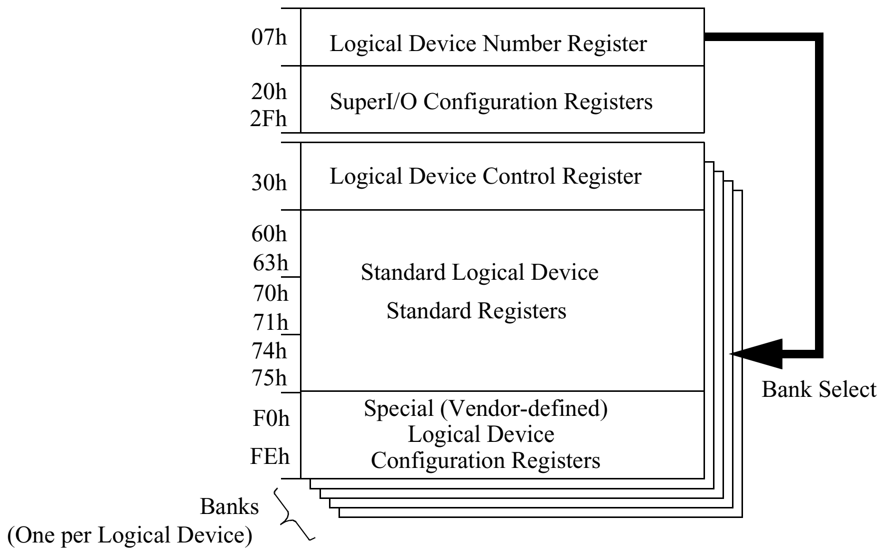


Figure 4-6 Structure of the Standard Configuration Register File

Table 4.48 Logical Device Number (LDN) Assignments

LDN	Functional Block
00h	Floppy Disk Controller (FDC)
01h	Parallel Port (PP)
02h	Serial Port 2 (SP2)
03h	Serial Port 1 (SP1)
04h	System Wake-Up Control (SWC)
05h	Keyboard and Mouse Controller (KBC) - Mouse interface
06h	Keyboard and Mouse Controller (KBC) - Keyboard interface
07h	Infrared Communication Port (IRCP)
08h	ACCESS.Bus (ACB)
09h	Reserved
0Ah	Real-Time Clock (RTC)

Write accesses to unimplemented registers (that is accessing the Data register while the Index register points to a non-existing register), are ignored and read returns 00h on all addresses except for 74h and 75h (DMA configuration registers) which returns 04h (indicating no DMA channel is active). The configuration registers are accessible immediately after reset.

4.5.5. Standard Logical Device Configuration Register Definitions

Unless otherwise noted in Table 4.49 through Table 4.54:

- All registers are read/write.
- All reserved bits return 0 on reads, except where noted otherwise. They must not be modified as it may cause unpredictable results. *Use read-modify-write to prevent the values of reserved bits from being changed during write.*
- Write only registers should not use read-modify-write during updates.

Table 4.49 Standard Control Registers

Index	Register Name	Description
07h	Logical Device Number	This register selects the current logical device. See Table 4.48 for valid numbers. All other values are reserved.
20h - 2Fh	SuperI/O Configuration	SuperI/O configuration registers and ID registers

Table 4.50 Logical Device Activate Register

Index	Register Name	Description
30h	Activate	Bit 0 - Logical device activation control 0 = Disabled 1 = Enabled Bits 7-1 - Reserved

Table 4.51 I/O Space Configuration Registers

Index	Register Name	Description
60h	I/O Port Base Address Bits (15-8) Descriptor 0	Indicates selected I/O lower limit address bits 15-8 for I/O Descriptor 0.
61h	I/O Port Base Address Bits (7-0) Descriptor 0	Indicates selected I/O lower limit address bits 7-0 for I/O Descriptor 0.
62h	I/O Port Base Address Bits (15-8) Descriptor 1	Indicates selected I/O lower limit address bits 15-8 for I/O Descriptor 1.
63h	I/O Port Base Address Bits (7-0) Descriptor 1	Indicates selected I/O lower limit address bits 7-0 for I/O Descriptor 1.

Table 4.52 Interrupt Configuration Registers

Index	Register Name	Description
70h	Interrupt Number	Bits 3-0 select the interrupt number. A value of 1 selects IRQ1, a value of 2 selects IRQ2, etc. (up to IRQ15). IRQ0 is not a valid interrupt selection.
71h	Interrupt Request Type Select	Indicates the type and level of the interrupt request number selected in the previous register. Bit 0 - Type of interrupt request selected in previous register 0 = Edge 1 = Level Bit 1 - Level of interrupt request selected in previous register 0 = Low polarity 1 = High polarity

Table 4.53 DMA Configuration Registers

Index	Register Name	Description
74h	DMA Channel Select 0	Indicates selected DMA channel for DMA 0 of the logical device (0 - The first DMA channel in the case of using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.
75h	DMA Channel Select 1	Indicates selected DMA channel for DMA 1 of the logical device (1 - The second DMA channel in the case of using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 1. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.

Table 4.54 Special Logical Device Configuration Registers

Index	Register Name	Description
F0h-FEh	Logical Device Configuration	Special (vendor-defined) configuration options.

4.5.6. Standard Configuration Registers

SuperI/O Control and Configuration Registers

Logical Device Control and Configuration Registers - one per logical device (some are optional)

Index	Register Name
07h	Logical Device Number
20h	SuperI/O ID
21h	SuperI/O Configuration 1
22h	SuperI/O Configuration 2
27h	SuperI/O Revision ID
2Eh	Reserved
30h	Logical Device Control (Activate)
60h	I/O Port Base Address Descriptor 0 Bits 15-8
61h	I/O Port Base Address Descriptor 0 Bits 7-0
62h	I/O Port Base Address Descriptor 1 Bits 15-8
63h	I/O Port Base Address Descriptor 1 Bits 7-0
70h	Interrupt Number Select
71h	Interrupt Type Select
74h	DMA Channel Select 0
75h	DMA Channel Select 1
F0h	Device Specific Logical Device Configuration 1
F1h	Device Specific Logical Device Configuration 2
F2h	Device Specific Logical Device Configuration 3
F3h	Device Specific Logical Device Configuration 4

Figure 4-7 Configuration Register Map

SuperI/O Control and Configuration Registers (20h/27h)

The SuperI/O Configuration registers at indexes 20h to 27h are mainly used for part identification, global power management and the selection of pin multiplexing options.

Logical Device Control and Configuration Registers (30h)

A subset of these registers is implemented for each logical device.

Control

The only implemented logical device control register is the activate register at index 30h. Bit 0 of the activate register, together with the Global Device Enable bit (except for the RTC and the SWC) control the activation of the associated function block. Activation of the block enables access to the block's registers, and attaches its system resources, which are unused as long as the block is not activated. Other effects may apply, on a function-specific basis (such as clock enable and active pinout signaling).

Standard Configuration

The standard configuration registers are used to manage the resource allocation to the functional blocks. The I/O port base address descriptor 0 is a pair of registers at Index 60-61, holding the (first or only) 16-bit base address for the register set of the functional block. An optional second base-address (descriptor 1) at index 62-63 is used for devices with more than one continuous register set. Interrupt Number Select (index 70h) and Interrupt Type Select (index 71h) allocate an IRQ line to the block and control its type. DMA Channel Select 0 (index 74h) allocates a DMA channel to the block, where applicable. DMA Channel Select 1 (index 75h) allocates a second DMA channel, where applicable.

Special Configuration

The vendor identification registers, starting at index F0h are used to control function-specific parameters such as operation modes, power saving modes, pin TRI-STATE, clock rate selection, and non-standard extensions to generic functions.

4.5.6.1. Default Configuration Setup

The device has three reset types:

- **Software Reset**
  Generated by bit 1 of the SIOCF1 register, which resets all logical devices. A software reset also resets most bits in the SuperI/O Configuration and Control registers (see Section 4.6.1 for the bits not affected). This reset does not affect register bits that are locked for write access.
- **Hardware Reset**
  Activated by the system reset signal. This resets all logical devices, with the exception of the RTC and the SWC, and all SuperI/O Configuration and Control registers, with the exception of the SIOCF2 register. It also resets all SuperI/O control and configuration registers, except for those that are battery-backed.
- **VPP Power-Up Reset**
  Activated when V_{BAT} is powered on after both have been off. V_{PP} is an internal voltage which is V_{BAT}. This reset resets all registers whose values are retained by V_{PP}.

The SuperI/O wakes up with the default setup, as follows:

- In the event of a hardware reset:
  - The configuration base address is 2Eh, 15Ch or None, according to the SIOCFG pin values, as shown in Table 4.47 on page 258.
  - The Keyboard Controller (KBC) may be activated and all other logical devices are disabled, with the exception of the RTC and the SWC, which remains functional but whose registers cannot be accessed.
- In the event of either a hardware or a software reset:
  - The legacy devices are assigned with their legacy system resource allocation.
  - Proprietary functions are not assigned with any default resources and the default values of their base addresses are all 00h.

4.5.6.2. Address Decoding

A full 16-bit address decoding is applied when accessing the configuration I/O space, as well as the registers of the functional blocks. However, the number of configurable bits in the base address registers vary for each device.

The lower 1, 2, 3 or 4 address bits are decoded within the functional block to determine the offset of the accessed register, within the device's I/O range of 2, 4, 8 or 16 bytes, respectively. The rest of the bits are matched with the base address register to decode the entire I/O range allocated to the device. Therefore the lower bits of the base address register are forced to 0 (read-only), and the base address is forced to be 2, 4, 8 or 16 byte aligned, according to the size of the IO range.

- The base address of the FDC, RTC, Serial Port 1, Serial Port 2, Infrared Communication Port and KBC are limited to the I/O address range of 00h to 7FXh only (bits 11-15 are forced to 0).
- The Parallel Port base address is limited to the I/O address range of 00h to 3F8h.
- The addresses of the non-legacy devices are configurable within the full 16-bit address range (up to FFFXh).

In some special cases, other address bits are used for internal decoding (such as bit 2 in the KBC and bit 10 in the Parallel Port). The KBC has two I/O descriptors with some implied dependency between them. For more details, please see the detailed description of the base address register for each specific logical device.

4.5.6.3. The Internal Clocks

The source of the device internal clocks is a 48 MHz clock signal, which is routed through the CLKIN pin. Wake-up on KBD, Mouse and RING pulse train detection operates on 32 KHz clock.

4.6. SuperI/O Configuration Registers

This section describes the SuperI/O configuration and ID registers (those registers with first level indexes in the range of 20h - 2Eh). See Table 4.56 for a summary and directory of these registers.

4.6.1. Register Type Abbreviations

The register maps in this chapter use the following abbreviations:

Table 4.55 Register Type Abbreviations

Symbol	Meaning
R/W	Read/Write
R	Read from a specific address returns the value of a specific register. Write to the same address is to a different register.
W	Write
RO	Read Only
R/W1C	Read/Write 1 to Clear. Writing 1 to a bit clears it to 0. Writing 0 has no effect.

Table 4.56 SuperI/O Configuration Registers

Index	Mnemonic	Register Name	Power Well	Type	Reference
20h	SID	SuperI/O ID	V_{DD}	RO	Table 4.57
21h	SIOCF1	SuperI/O Configuration 1	V_{DD}	R/W	Table 4.58
22h	SIOCF2	SuperI/O Configuration 2	V_{PP}	R/W	Table 4.59
27h	SRID	SuperI/O Revision ID	V_{DD}	RO	Table 4.60
2Eh	Reserved				

Table 4.57 SuperI/O ID Register (SID) - Index 20H

Bit	7	6	5	4	3	2	1	0
Name	Chip ID							
Reset	1	1	1	0	0	0	1	1
Notes: RO. This register contains the identity number of the chip. The SuperI/O is identified by the value E3H.								

Table 4.58 SuperI/O Configuration 1 Register (SIOCF1) - Index 21H

Bit	7	6	5	4	3	2	1	0
Name	**General Purpose Scratch**		**Lock Scratch**	**PNF Status**	**Reserved**	**Pin Function Lock**	**SW Reset**	**Global Device Enable**
Reset	0	0	0	X	0	0	0	1
This is a read/write register.								

Bit	Description
7-6	**General Purpose Scratch** When bit 5 is set to 1, these bits are read only. After reset, these bits can be read or write. Once changed to read only, the bits can be changed back to read/write only by a hardware reset.
5	**Lock Scratch** This bit controls bits 7 and 6 of this register. Once this bit is set to 1 by software, it can only be cleared to 0 by a hardware reset. 0 = Bits 7 and 6 of this register are read/write bits (default). 1 = Bits 7 and 6 of this register are read only bits.
4	Reserved**.**
3	Reserved
2	**Pin Function Lock** When bit set to 1, all function selection on the associated pins lock (Bits 0 and 2 of the SIOCF2 register). When set to 1 by software, it can only be cleared to 0 by Master Reset or power-off. 0 = No effect (default) 1 = Pin function locked
1	**SW Reset** Read always returns 0. 0 = Ignored (default) 1 = Resets all the devices that are reset by MR (with the exception of the lock bits) and the registers of the SWC
0	**Global Device Enable** This bit controls the function enable of all the logical devices in the SuperI/O, except the SWC and the RTC. It allows them to be disabled simultaneously by writing to a single bit. 0 = All logical devices in the SuperI/O disabled, except the SWC and the RTC. 1 = Each logical device enabled according to its Activate register at index 30h (default)

Table 4.59 SuperI/O Configuration 2 Register (SIOCF2) - Index 22H

Bit	7	6	5	4	3	2	1	0
Name	Reserved	General Purpose Scratch			S_GPWI02 Debounce Enable	Reserved	S_GPWI01 Debounce Enable	S_GPWI01 Function Select
Reset	1	0	0	0	1	0	1	0
This register controls the multiplexing of two pins. Its value is retained by V_{PP}, and is not affected by either hardware or software reset.								

Bit	Description
7	**Reserved**
6-4	**General Purpose Scratch** Battery-backed.
3	**S_GPWI02 Debounce Enable** 0 = Debounce disabled 1 = 16mS debounce enabled (V_{PP} power-up default)
2	**Reserved**
1	**S_GPWI01 Debounce Enable** 0 = Debounce disabled 1 = 16mS debounce enabled (V_{PP} power-up default)
0	**S_GPWI01 Function Select** 0 = $\overline{\text{RING}}$ (V_{PP} power-up default) 1 = S_GPWI01

Table 4.60 SuperI/O Revision ID Register (SRID) - Index 27H

Bit	7	6	5	4	3	2	1	0
Name	Chassis Intrusion	General Purpose Scratch			S_GPWI02 Debounce Enable	S_GPWI02 Function Select	S_GPWI01 Debounce Enable	S_GPWI01 Function Select
Reset	1	0	0	0	1	0	1	0
This read only register contains the identity number of the chip revision. SRID is incremented on each revision.								

4.7. Floppy Disk Controller (FDC) Configuration

The generic FDC is a standard FDC with a digital data separator, and is DP8473 and N82077 software compatible. The FDC supports 14 of the 17 standard FDC signals including:

- FM and MFM modes are supported. To select either mode, set bit 6 of the first command byte when writing to/reading from a diskette, where:
  0 = FM mode
  1 = MFM mode
- A logic 1 is returned for all floating (TRI-STATE) FDC register bits upon read cycles.
- Exceptions to standard FDC support include:
- DRATE1 is not supported.

Table 4.61 lists the FDC functional block registers.

Table 4.61 FDC Registers

Offset	Mnemonic	Register Name	Type
00h	SRA	Status A	RO
01h	SRB	Status B	RO
02h	DOR	Digital Output	R/W
03h	TDR	Tape Drive	R/W
04h	MSR	Main Status	R
	DSR	Data Rate Select	W
05h	FIFO	Data (FIFO)	R/W
06h		N/A	X
07h	DIR	Digital Input	R
	CCR	Configuration Control	W
All Registers are described in the Programmers Manual			

Table 4.62 Logical Device 0 (FDC) Configuration

Index	Configuration Register or Action	Type	Reset
30h	Activate. See also bit 0 of the SIOCF1 register.	R/W	00h
60h	Base Address MSB register. Bits 7-3 (for A15-11) are read only, 00000b.	R/W	03h
61h	Base Address LSB register. Bits 2 and 0 (for A2 and A0) are read only, 00b.	R/W	F2h
70h	Interrupt Number	R/W	06h
71h	Interrupt Type. Bit 1 is read/write; other bits are read only.	R/W	03h
74h	DMA Channel Select	R/W	02h
75h	Report no second DMA assignment	RO	04h
F0h	FDC Configuration register	R/W	24h
F1h	Drive ID register	R/W	00h

Table 4.63 FDC Configuration Register - Index F0H

Bit	7	6	5	4	3	2	1	0
Name	Four Drive Control	TDR Register Mode		Reserved	Write Protect	Reserved		TRI-STATE Control
Reset	0	0	1	0	0	1	0	0
Required				0				
RW. This register is reset by hardware to 24h.								

Bit	Description
7	**Four Drive Control** 0 = Two floppy drives directly controlled by $\overline{\text{DR1-0}}$ and $\overline{\text{MTR1-0}}$ (default) 1 = Four floppy drives controlled with the aid of external logic Only one floppy present in ZFx86.
6	**TDR Register Mode** 0 = PC-AT compatible drive mode; that is, bits 7-2 of the TDR are ignored (default) 1 = Enhanced drive mode
5	Reserved
4	**Reserved —** Must be set to 0.
3	**Write Protect** This bit allows forcing of write protect by software. When set, write to the floppy disk drive is disabled. This effect is identical to $\overline{\text{WP}}$ when it is active. 0 = Write protected according to $\overline{\text{WP}}$ signal (default) 1 = Write protected regardless of value of $\overline{\text{WP}}$ signal
2-1	**Reserved.** Reset value of bit 2 is 1.

Bit	Description
0	**TRI-STATE Control** When enabled and the device is inactive, the logical device output pins are in TRI-STATE. 0 = Disabled (default) 1 = Enabled

Table 4.64 Drive ID Register - Index F1H

Bit	7	6	5	4	3	2	1	0
Name	Reserved				Drive 1 ID		Drive 0 ID	
Reset	0				0		0	
RW. This read/write register is reset by hardware to 00h. This register controls bits 5 and 4 of the TDR register in the Enhanced mode. Usage Hints: Some BIOS implementations support automatic media sense FDDs, in which case bit 5 of the TDR register in the Enhanced mode is interpreted as valid media sense when it is cleared to 0. If drive 0 and/or drive 1 do not support automatic media sense, bits 1 and/or 3 of the Drive ID register should be set to 1 respectively (to indicate non-valid media sense) when the corresponding drive is selected and the Drive ID bit is reflected on bit 5 of the TDR register in the Enhanced mode.								

Bit	Description
7-4	Reserved
3-2	Drive 1 ID. When drive 1 is accessed, these bits are reflected on bits 5-4 of the TDR register, respectively.
1-0	Drive 0 ID. When drive 0 is accessed, these bits are reflected on bits 5-4 of the TDR register, respectively.

4.8. Parallel Port Configuration

The SuperI/O Parallel Port supports all IEEE1284 standard communication modes:

- Compatibility (known also as Standard or SPP)
- Bidirectional (known also as PS/2)
- FIFO, EPP (known also as Mode 4)
- ECP (with an optional Extended ECP mode).

The Parallel Port includes two groups of runtime registers:

- A group of 21 registers at first level offset, sharing 14 entries. Three of these registers (at offsets 403h, 404h, and 405h) use only the Extended ECP mode.
- A group of four registers, used only in the Extended ECP mode, are accessed by a second level offset.

The desired mode is selected by the ECR runtime register (offset 402h). The selected mode determines which runtime registers are used and which address bits are used for the base address.

4.8.1. Logical Device 1 (PP) Configuration

Table 4.65 lists the configuration registers, their offset addresses, and the associated modes which affect the Parallel Port. Only the last register (F0h) is described here.

Table 4.65 Parallel Port Configuration Registers

Index	Configuration Register or Action	Type	Reset
30h	**Activate**. See also bit 0 of the SIOCF1 register.	R/W	00h
60h	**Base Address MSB register**. Bits 7-3 (for A15-11) are read only, 00000b. Bit 2 (for A10) should be set to 0b.	R/W	02h
61h	**Base Address LSB register**. Bits 1 and 0 (A1 and A0) are read only, 00b. For ECP Mode 4 (EPP) or when using the Extended registers, bit 2 (A2) should be set to 0b.	R/W	78h
70h	Interrupt Number	R/W	07h
71h	Interrupt Type Bits 7-2 are read only. Bit 1 is a read/write bit. Bit 0 is read only. It reflects the interrupt type dictated by the Parallel Port operation mode. This bit is set to 1 (level interrupt) in Extended Mode and cleared (edge interrupt) in all other modes.	R/W	02h
74h	DMA Channel Select	R/W	04h
75h	Report no second DMA assignment	RO	04h
F0h	Parallel Port Configuration register	R/W	F2h

Table 4.66 Parallel Port Configuration Register - F0H

Bit	7	6	5	4	3	2	1	0
Name	Reserved			Extended Register Access	Reserved		Power Mode Control	TRI-STATE Control
Reset	1	1	1	1	0	0	1	0
Required	1	1	1					
RW. This read/write register is reset by hardware to F2h.								

Bit	Description
7-5	**Reserved.** Must be 111.
4	**Extended Register Access** 0 = Registers at base (address) + 403h, base + 404h and base + 405h are not accessible (reads and writes are ignored). 1 = Registers at base (address) + 403h, base + 404h and base + 405h are accessible. This option supports run-time configuration within the Parallel Port address space.
3-2	**Reserved**

Bit	Description
1	**Power Mode Control** When the logical device is active: 0 = Parallel port clock disabled. ECP modes and EPP time-out are not functional when the logical device is active. Registers are maintained. 1 = Parallel port clock enabled. All operation modes are functional when the logical device is active (default).
0	**TRI-STATE Control** When enabled and the device is inactive, the logical device output pins are in TRI-STATE. 0 = Disabled (default) 1 = Enabled

4.9. System Wake-Up Control (SWC)

4.9.1. Overview

The SWC wakes up the system by sending a power-up request to the controller, in response to the following maskable system events:

- Modem ring (RI1 and RI2 pins)
- Telephone ring (RING input pin)
- Keyboard activity or specific programmable key sequence
- Mouse activity or specific programmable sequence of clicks and movements
- Programmable Consumer Electronics IR (CEIR) address
- General purpose events (S_GPWI01 and S_GPWI02).

4.9.2. Functional Description

The SWC monitors eight system events or activities. Each one of them is fed into a dedicated detector that decides when this event is active, according to predetermined (either fixed or programmable) criteria. A set of dedicated registers is used to determine the wake-up criteria, including the CEIR address and the Keyboard sequence.

A Wake-Up Events Status Register (WKSR) and a Wake-Up Events Control Register (WKCR) hold a Status bit and Enable bit, respectively, for each one of the events.

Upon detection of any active event, the corresponding Status bit is set to 1. If the event is enabled (the corresponding Enable bit is set to 1), a power-up request is issued to the controller. In addition, detection of an active wake-up event may be also routed to any arbitrary IRQ.

Disabling an event prevents it from issuing power-up requests, but does not affect the Status bits. A power-up reset is issued when both the Status and Enable bits equal 1 for at least one event.

The SWC logic is powered by V_{CC_IO}. The SWC control and configuration registers are battery backed, powered by V_{PP}. The setup of the wake-up events, including programmable sequences, is retained throughout power failures (no V_{CC_IO}) as long as the battery is connected. V_{PP} is taken from V_{CC_IO} if V_{CC_IO} is greater than the minimum (Min) value; otherwise, V_{BAT} is used as the V_{PP} source.

Hardware reset does not affect these registers. They are reset only by SuperI/O software reset or power-up of V_{PP}.

4.9.3. Event Detection

4.9.3.1. Modem Ring

High to low transitions on RI1 or RI2 indicate the detection of ring in external modem connected to Serial Port 1 or Serial Port 2 respectively and can be used as a wake-up event.

4.9.3.2. Telephone Ring

A telephone ring is detected by the SWC by processing the raw signal coming directly from the telephone line into the RING input pin. Detection of a pulse-train with a frequency higher than 16 Hz that lasts at least 0.19 sec, is used as a wake-up event.

The RING pulse-train detection is achieved by monitoring the falling edges on RING in time slots of 62.5 msec (a 16 Hz cycle). A positive detection occurs if falling edges of RING are detected in three consecutive time slots, following a time slot in which no RING falling edge is detected. This detection method guarantees the detection of a RING pulse-train with frequencies higher than 16 Hz. It filters out (does not detect) pulses of less than 10 Hz, and may detect pulses between 10 Hz to 16 Hz.

4.9.3.3. Keyboard and Mouse Activity

The detection of either any activity or a specific predetermined Keyboard or Mouse activity can be used as a wake-up event.

The Keyboard wake-up detection can be programmed to detect:

- Any keystroke
- A specific programmable sequence of up to eight alphanumeric keystrokes
- Any programmable sequence of up to 8 bytes of data received from the keyboard.

The Mouse wake-up detection can be programmed to detect either any Mouse click or movement, or a specific programmable click (left or right) or double-clicks.

The keyboard or mouse event detection operates independently of the KBC (which is powered down with the rest of the system).

4.9.3.4. CEIR Address

A CEIR transmission received on an IRRX pin in a pre-selected standard (NEC, RCA or RC-5) is matched against a programmable CEIR address. Detection of matching can be used as a wake-up event.

Whenever an IR signal is detected, the receiver immediately enters the active state. When this happens, the receiver keeps sampling the IR input signal and generates a bit string where a logic 1 indicates an idle condition and a logic 0 indicates the presence of IR energy. The received bit string is de-serialized and assembled into 8-bit characters.

The expected CEIR protocol of the received signal should be configured through bits 5,4 at the CEIR Wake-Up Control register (see Table 4.77 on page 280).

The CEIR Wake-Up Address register (IRWAD) holds the unique address to be compared with the address contained in the incoming CEIR message. If CEIR is enabled (bit 0 of the IRWCR register is 1) and an address match occurs, then the CEIR Event Status bit of the WKSR register is set to 1 (see Table 4.70 on page 275).

The CEIR Address Shift register holds the received address which is compared with the address contained in the IRWAD. The comparison is affected also by the CEIR Wake-Up Address Mask register (IRWAM) in which each bit determines whether to ignore the corresponding bit in the IRWAD.

If CEIR routing to interrupt request is enabled, the assigned SWC interrupt request may be used to indicate that a complete address has been received. To get this interrupt when the address is completely received, the IRWAM should be written with FFh. Once the interrupt is received, the value of the address can be read from the ADSR register.

Another parameter that is used to determine whether a CEIR signal is to be considered valid is the bit cell time width. There are four time ranges for the different protocols and carrier frequencies. Four pairs of registers define the low and high limits of each time range. (See 'CEIR Wake-Up Range 0 Registers' on page 281 (and following) for more details regarding the recommended values for each protocol.)

The CEIR address detection operates independently of the serial port with the IR (which is powered down with the rest of the system).

4.9.3.5. General-Purpose Events

A general-purpose event is defined as the detection of falling edge, rising edge, low level, or high level on a specific signal. Each signal's event is configurable via software. S_GPWI01 and S_GPWI02 may wake up the system from power-off state, or generate an interrupt if the system is in power-on state.

A debouncer of 16 mS is enabled (default) on each event. It may be disabled by software.

The SWC registers are organized in two banks. The offsets are related to a base address that is determined by the SWC Base Address Register in the device configuration. The lower three registers are common to the two banks while the upper registers (03-0fh) are divided as follows:

- Bank 0 holds the Keyboard/Mouse Control Registers.
- Bank 1 holds the CEIR Control Registers.

The active bank is selected through the Configuration Bank Select field (bits 1-0) in the Wake-Up Configuration Register (WKCFG). See Table 4.72 on page 277.

4.9.3.6. SWC Register Map

Table 4.67 Banks 0 and 1 - The Common Control and Status Register Map

Offset	Mnemonic	Name	Type	Reference
00h	WKSR	Wake-Up Events Status Register	R/W	Page 275
01h	WKCR	Wake-Up Events Control Register	R/W	Page 276
02h	WKCFG	Wake-Up Configuration Register	R/W	Page 277

Table 4.68 Bank 0 - PS/2 KBD/MOUSE Wake-Up Config/Control Register Map

Offset	Mnemonic	Name	Type	Reference
03h	PS2CTL	PS/2 Protocol Control Register	R/W	Page 278
04h-05h	**Reserved**			
06h	KDSR	Keyboard Data Shift Register	RO	Page 279
07h	MDSR	Mouse Data Shift Register	RO	Page 279
08h-0Fh	PS2KEY0–PS2KEY7	PS/2 Keyboard Key Data Registers	R/W	Page 280

Table 4.69 Bank 1 - CEIR Wake-Up Config/Control Register Map

Offset	Mnemonic	Name	Type	Reference
03h	IRWCR	CEIR Wake-Up Control Register	R/W	Page 280
04h	**Reserved**			
05h	IRWAD	CEIR Wake-Up Address Register	R/W	Page 281
06h	IRWAM	CEIR Wake-Up Address Mask Register	R/W	Page 281
07h	ADSR	CEIR Address Shift Register	R/O	Page 281
08h	IRWTR0L	CEIR Wake-Up, Range 0, Low Limit Register	R/W	Page 282
09h	IRWTR0H	CEIR Wake-Up, Range 0, High Limit Register	R/W	Page 282
0Ah	IRWTR1L	CEIR Wake-Up, Range 1, Low Limit Register	R/W	Page 282
0Bh	IRWTR1H	CEIR Wake-Up, Range 1, High Limit Register	R/W	Page 283
0Ch	IRWTR2L	CEIR Wake-Up, Range 2, Low Limit Register	R/W	Page 284
0Dh	IRWTR2H	CEIR Wake-Up, Range 2, High Limit Register	R/W	Page 284
0Eh	IRWTR3L	CEIR Wake-Up, Range 3, Low Limit Register	R/W	Page 284
0Fh	IRWTR3H	CEIR Wake-Up, Range 3, High Limit Register	R/W	Page 285

Table 4.70 Wake-Up Events Status Register (WKSR) - 00H

Bit	7	6	5	4	3	2	1	0
Name	**S_GPWI02 Event Status**	**S_GPWI01 Event Status**	**CEIR Event Status**	**Mouse Event Status**	**KBD Event Status**	**$\overline{\text{RING}}$ Event Status**	**$\overline{\text{RI2}}$ Event Status**	**$\overline{\text{RI1}}$ Event Status**
Reset	0	0	0	0	0	0	0	0
R/W: This register is set to 00h on power-up of V_{PP} or software reset. It indicates which wake-up events occurred.								

Bit	Description
7	**S_GPWI02 Event Status** This sticky bit shows the status of the S_GPWI02 event detection. 0 = Event not detected (default) 1 = Event detected
6	**S_GPWI01 Event Status** This sticky bit shows the status of the S_GPWI01 event detection. 0 = Event not detected (default) 1 = Event detected
5	**CEIR Event Status** This sticky bit shows the status of the CEIR event detection. 0 = Event not detected (default) 1 = Event detected
4	**Mouse Event Status** This sticky bit shows the status of the Mouse event detection. 0 = Event not detected (default) 1 = Event detected
3	**KBD Event Status** This sticky bit shows the status of the KBD event detection. 0 = Event not detected (default) 1 = Event detected
2	**$\overline{\text{RING}}$ Event Status** This sticky bit shows the status of the $\overline{\text{RING}}$ event detection. 0 = Event not detected (default) 1 = Event detected
1	**$\overline{\text{RI2}}$ Event Status** This sticky bit shows the status of $\overline{\text{RI2}}$ event detection. 0 = Event not detected (default) 1 = Event detected
0	**$\overline{\text{RI1}}$ Event Status** This sticky bit shows the status of $\overline{\text{RI1}}$ event detection. 0 = Event not detected (default) 1 = Event detected

Table 4.71 Wake-Up Events Control Register (WKCR) - 01H

Bit	7	6	5	4	3	2	1	0
Name	**S_GPWI02 Event Enable**	**S_GPWI01 Event Enable**	**CEIR Event Enable**	**Mouse Event Enable**	**KBD Event Enable**	**$\overline{\text{RING}}$ Event Enable**	**$\overline{\text{RI2}}$ Event Enable**	**$\overline{\text{RI1}}$ Event Enable**
Reset	0	0	0	0	0	1	1	1
R/W: This register is set to 07h on power-up of V_{PP} or software reset. Detected wake-up events that are enabled activate issue a power-up request signal to the controller.								

Bit	Description
7	**S_GPWI02 Event Enable** 0 = Disabled (default) 1 = Enabled
6	**S_GPWI01 Event Enable** 0 = Disabled (default) 1 = Enabled
5	**CEIR Event Enable** 0 = Disabled (default) 1 = Enabled
4	**Mouse Event Enable** 0 = Disabled (default) 1 = Enabled
3	**KBD Event Enable** 0 = Disabled (default) 1 = Enabled.
2	**$\overline{\text{RING}}$ Event Enable** 0 = Disabled 1 = Enabled (default)
1	**$\overline{\text{RI2}}$ Event Enable** 0 = Disabled 1 = Enabled (default)
0	**$\overline{\text{RI1}}$ Event Enable** 0 = Disabled 1 = Enabled (default)

Table 4.72 Wake-Up Configuration Register (WKCFG) - 02H

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**	**S_GPWI02 Event Type**	**S_GPWI02 Event Polarity**	**S_GPWI01 Event Type**	**S_GPWI01 Event Polarity**	**Swap KBC Inputs**	**Configuration Bank Select**	
Reset	0	0	0	0	0	0	0	0
R/W: This register is set to 00h on power-up of V_{PP} or software reset. It enables access to CEIR registers or to Keyboard/Mouse registers.								

Bit	Description
7	**Reserved**
6	**S_GPWI02 Event Type** 0 = Edge 1 = Level
5	**S_GPWI02 Event Polarity** 0 = Falling edge, low level 1 = Rising edge, high level
4	**S_GPWI01 Event Type** 0 = Edge 1 = Level
3	**S_GPWI01 Event Polarity** 0 = Falling edge, low level 1 = Rising edge, high level
2	**Swap KBC Inputs** 0 = No swapping (default) 1 = KBD (KBCLK, KBDAT) and Mouse (MCLK, MDAT) inputs swapped
1-0	**Configuration Bank Select** Bits 1 0 Bank Selected 0 0 Keyboard/Mouse Registers (Bank 0) 0 1 CEIR Registers (Bank 1) 1 X Reserved

4.9.3.7. PS/2 Keyboard and Mouse Wake-Up Events

The SWC can be configured to detect any predetermined PS/2 keyboard or mouse activity.

The detection mechanisms for keyboard and mouse events are independent. Therefore, they can be operated simultaneously with no interference. Since both mechanisms are implemented by hardware which is independent of the device's keyboard controller, the keyboard controller itself need not be activated to detect either keyboard or mouse events. See the ZFx86 User's Guide for more information.

Mouse Wake-Up Events

Program the mouse wake-up detection mechanism to detect either any mouse click or movement, or a specific programmable click (left or right) or double-click.

To program this mechanism to wake-up on a specific event, set bits 6-4 of the PS2CTL register to the required value, according to the description of these bits in Table 4.73.

Table 4.73 PS/2 Protocol Control Register (PS2CTL) (Bank 0 Offset 03H)

Bit	7	6	5	4	3	2	1	0
Name	**Disable Parity Check**	**Mouse Wake-Up Configuration**			**Keyboard Wake-Up Configuration**			
Reset	0	0	0	0	0	0	0	0
R/W: This register is set to 00h on power-up of V_{PP} or software reset. It configures the PS/2 Keyboard and Mouse wake-up features								

Bit	Description
7	**Disable Parity Check**
6-4	**Mouse Wake-Up Configuration** **Bits** **6 5 4 Configuration** 0 0 0 Disable Mouse wake-up detection 0 0 1 Wake-up on any Mouse movement or button click 0 1 0 Wake-up on left button click 0 1 1 Wake-up on left button double-click 1 0 0 Wake-up on right button click 1 0 1 Wake-up on right button double-click 1 1 0 Wake-up on any button single-click (left, right or middle) 1 1 1 Wake-up on any button double-click (left, right or middle)

Bit	Description
3-0	**Keyboard Wake-Up Configuration** Bits 3 2 1 0 Configuration 0 0 0 0 Disable Keyboard wake-up detection 0 0 0 1 to 0 1 1 1 } Special key seq 2-8 PS/2 scan codes, “Make” and “Break” (incl Shift, Alt keys) 1 0 0 0 to 1 1 1 1 } Password enabled with 1-8 keys “Make” code (excluding Shift and Alt keys)

Table 4.74 Keyboard Data Shift Register (KDSR) - Bank 0 Offset 06H

Bit	7	6	5	4	3	2	1	0
Name	**Keyboard Data**							
Reset	0							
R/O: This register is set to 00h on power-up of V_{PP} or software reset. It stores the Keyboard data shifted in from the Keyboard during transmission, only when Keyboard wake-up detection is enabled.								

Table 4.75 Mouse Data Shift Register (MDSR) 07H

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**					**Mouse Data**		
Reset	0					0		
R/O: This register is set to 00h on power-up of V_{CC_IO} or software reset. It stores the Mouse data shifted in from the Mouse during transmission, only when Mouse wake-up detection is enabled.								

4.9.3.8. PS/2 Keyboard Key Data Registers (PS2KEY0 - PS2KEY7)

Eight registers (PS2KEY0-PS2KEY7) store the scan codes for the password or key sequence of the keyboard wake-up feature, as follows:

- PS2KEY0 register stores the scan code for the first key in the password/key sequence.
- PS2KEY1 register stores the scan code for the second key in the password/key sequence.
- PS2KEY2 - PS2KEY7 registers store the scan codes for the third to eighth keys in the password/key sequence.

When one of these registers is set to 00h, it indicates that the value of the corresponding scan code byte is ignored (not compared). These registers are set to 00h on power-up of V_{PP} or software reset.

Location: Bank 0, Offset 08h-0Fh

Type: R/W

Table 4.76 PS/2 Keyboard Key Data Registers (PS2KEY0 - PS2KEY7)

Bit	7	6	5	4	3	2	1	0
Name	**Scan Code of Keys 0-7**							
Reset	0	0	0	0	0	0	0	0

Table 4.77 CEIR Wake-Up Control Register (IRWCR) - Bank 1 Offset 3

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**		**CEIR Protocol Select**		**Select IRRX2 Input**	**Invert IRRXn Input**	**Reserved**	**CEIR Enable**
Reset	0	0	0	0	0	0	0	0
R/W: This register is set to 00h on power-up of V_{PP} or software reset.								

Bit	Description
7-6	**Reserved**
5-4	**CEIR Protocol Select** **Bits** **5 4 Protocol** 0 0 RC5 0 1 NEC/RCA 1 X Reserved
3	**Select IRRX2 Input.** This selects the IRRX input. 0 = IRRX1 (default) 1 = IRRX2
2	**Invert IRRXn Input** 0 = Not inverted (default) 1 = Inverted
1	**Reserved**
0	**CEIR Enable** 0 = Disabled (default) 1 = Enabled

Table 4.78 CEIR Wake-Up Address Register (IRWAD) - Bank 1 Offset 05H

Bit	7	6	5	4	3	2	1	0
Name	**CEIR Wake-Up Address**							
Reset	0							
R/W: This register defines the station address to be compared with the address contained in the incoming CEIR message. If CEIR is enabled (bit 0 of the IRWCR Register is 1) and an address match occurs, then bit 5 of the WKSR Register is set to 1 (see Table 4.70, "Wake-Up Events Status Register (WKSR) - 00H," on page 275). This register is set to 00h on power-up of V_{PP} or software reset.								

Table 4.79 CEIR Wake-Up Address Mask Register (IRWAM) - Bank 1 Offset 6

Bit	7	6	5	4	3	2	1	0
Name	**CEIR Wake-Up Address Mask**							
Reset	1	1	1	0	0	0	0	0
R/W: Each bit in this register determines whether the corresponding bit in the IRWAD Register takes part in the address comparison. Bits 5, 6 and 7 must be set to 1 if the RC-5 protocol is selected. This register is set to E0h on power-up of V_{PP} or software reset.								

Bit	Description
7-0	**CEIR Wake-Up Address Mask** If the corresponding bit is 0, the address bit is not masked (enabled for compare). If the corresponding bit is 1, the address bit is masked (ignored during compare).

Table 4.80 CEIR Address Shift Register (ADSR) - Bank 1 Offset 7

Bit	7	6	5	4	3	2	1	0
Name	**CEIR Address**							
Reset	0							
R/O: This register holds the received address to be compared with the address contained in the IRWAD register. This register is set to 00h on power-up of V_{PP} or software reset.								

CEIR Wake-Up Range 0 Registers

These registers define the low and high limits of time range 0. The values are represented in units of 0.1 msec.

For the RC-5 protocol, the bit cell width must fall within this range for the cell to be considered valid. The nominal cell width is 1.778 msec for a 36 KHz carrier. IRWTR0L and IRWTR0H should be set to 10h and 14h respectively (default).

For the NEC protocol, the time distance between two consecutive CEIR pulses that encodes a bit value of 0 must fall within this range. The nominal distance for a 0 is1.125 msec for a 38 KHz carrier. IRWTR0L and IRWTR0H should be set to 09h and 0Dh respectively.

Table 4.81 CEIR Wake-Up Range 0 Registers - IRWTR0L– Bank 1 Offset 8

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**			**CEIR Pulse Change, Range 0, Low Limit**				
Reset	0	0	0	1	0	0	0	0
R/W: This register is set to 10h on power-up of V_{PP} or software reset.								

Table 4.82 CEIR Wake-Up Range 0 Registers - IRWTR0H – Bank 1 Offset 9

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**			**CEIR Pulse Change, Range 0, High Limit**				
Reset	0	0	0	1	0	1	0	0
R/W: This register is set to 14h on power-up of V_{PP} or software reset.								

CEIR Wake-Up Range 1 Registers

These registers define the low and high limits of time range 1. The values are represented in units of 0.1 msec.

For the RC-5 protocol, the pulse width defining a half-bit cell must fall within this range in order for the cell to be considered valid. The nominal pulse width is 0.889 for a 38 KHz carrier. IRWTR1L and IRWTR1H should be set to 07h and 0Bh respectively (default).

For the NEC/RCA protocol, the time between two consecutive CEIR pulses that encodes a bit value of 1 must fall within this range. The nominal time for a 1 is 2.25 msec for a 36 KHz carrier. IRWTR1L and IRWTR1H should be set to 14h and 19h respectively.

Table 4.83 CEIR Wake-Up Range 1 Registers - IRWTR1L – Bank 1 Offset 0AH

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**			**CEIR Pulse Change, Range 1, Low Limit**				
Reset	0			0	0	1	1	1
R/W: This register is set to 07h on power-up of V_{PP} or software reset.								

Table 4.84 CEIR Wake-Up Range 1 Registers - IRWTR1H – Bank 1 Offset 0BH

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**			**CEIR Pulse Change, Range 1, High Limit**				
Reset	0			0	1	0	1	1
R/W: This register is set to 0Bh on power-up of V_{PP} or software reset.								

CEIR Wake-Up Range 2 Registers

These registers define the low and high limits of time range 2. The values are represented in units of 0.1 msec. These registers are not used when the RC-5 protocol is selected.

For the NEC/RCA protocol, the header pulse width must fall within this range in order for the header to be considered valid. The nominal value is 9 msec for a 38 KHz carrier. IRWTR2L and IRWTR2H should be set to 50h and 64h respectively (default).

Table 4.85 CEIR Wake-Up Range 2 Registers - IRWTR2L – Bank 1 0CH)

Bit	7	6	5	4	3	2	1	0
Name	**CEIR Pulse Change, Range 2, Low Limit**							
Reset	0	1	0	1	0	0	0	0
R/W: This register is set to 50h on power-up of V_{pp} or software reset.								

Table 4.86 CEIR Wake-Up Range 2 Registers - IRWTR2H – Bank 1 0DH

Bit	7	6	5	4	3	2	1	0
Name	**CEIR Pulse Change, Range 2, High Limit**							
Reset	0	1	1	0	0	1	0	0
R/W: This register is set to 64h on power-up of V_{pp} or software reset.								

CEIR Wake-Up Range 3 Registers

These registers define the low and high limits of time range 3. The values are represented in units of 0.1 msec. These registers are not used when the RC-5 protocol is selected.

For the NEC protocol, the post header gap width must fall within this range in order for the gap to be considered valid. The nominal value is 4.5 msec for a 36 KHz carrier. IRWTR3L and IRWTR3H should be set to 28h and 32h respectively (default).

Table 4.87 CEIR Wake-Up Range 3 Registers - IRWTR3L – Bank 1 OEH

Bit	7	6	5	4	3	2	1	0
Name	**CEIR Pulse Change, Range 3, Low Limit**							
Reset	0	0	1	0	1	0	0	0
R/WS: This register is set to 28h on power-up of V_{pp} or software reset.								

Table 4.88 CEIR Wake-Up Range 3 Registers - IRWTR3H – Bank 1 OFH

Bit	7	6	5	4	3	2	1	0
Name	**CEIR Pulse Change, Range 3, High Limit**							
Reset	0	0	1	1	0	0	1	0
R/W: This register is set to 32h on power-up of V_{pp} or software reset.								

CEIR Recommended Values

Table 4.89 lists the recommended time ranges limits for the different protocols and their four applicable ranges. The values are represented in hexadecimal code where the units are of 0.1 msec.

Table 4.89 Time Range Limits for CEIR Protocols

Range	RC-5		NEC		RCA	
	Low Limit	High Limit	Low Limit	High Limit	Low Limit	High Limit
0	10h	14h	09h	0Dh	0Ch	12h
1	07h	0Bh	14h	19h	16h	1Ch
2	-	-	50h	64h	B4h	DCh
3	-	-	28h	32h	23h	2Dh

4.9.4. SWC Register Bitmap

Table 4.90 Banks 0 and 1 - The Common Three-Register Map

Register		Bits							
Offset	Mnemonic	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
00h	WKSR	**S_GPWI02 Event Status**	**S_GPWI01 Event Status**	**CEIR Event Status**	**Mouse Event Status**	**KBD Event Status**	**RING Event Status**	**RI2 Event Status**	**RI1 Event Status**
01h	WKCR	**S_GPWI02 Event Enable**	**S_GPWI01 Event Enable**	**CEIR Event Enable**	**Mouse Event Enable**	**KBD Event Enable**	**RING Event Enable**	**RI2 Event Enable**	**RI1 Event Enable**
02h	WKCFG	Reserved	S_GPWI02 Event Type	S_GPWI02 Event Polarity	S_GPWI01 Event Type	S_GPWI01 Event Polarity	Swap KBC Inputs	**Configuration Bank Select**	

Table 4.91 Bank 0 - PS/2 Keyboard/Mouse Wake-Up Config/Ctrl Registers

Register		Bits							
Offset	Mnemonic	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
03h	PS2CTL	**Disable Parity**	**Mouse Wake-Up Configuration**			**Keyboard Wake-Up Configuration**			
04-05	Reserved								
06h	KDSR	**Keyboard Data**							
07h	MDSR	**Reserved**					**Mouse Data**		
08-0F	PS2KEY0-PS2KEY7	**Scan Code of Keys 0-7**							

Table 4.92 CEIR Wake-Up Configuration and Control Registers

Register		Bits							
Offset	Mnemonic	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
03h	IRWCR	**Reserved**		**CEIR Protocol Select**		**Select IRRX2 Input**	**Invert IRRXn Input**	**Reserved**	**CEIR Enable**
04h	Reserved								
05h	IRWAD	CEIR Wake-Up Address							
06h	IRWAM	CEIR Wake-Up Address Mask							

Table 4.92 CEIR Wake-Up Configuration and Control Registers

07h	ADSR	CEIR Address	
08h	IRWTR0L	Reserved	**CEIR Pulse Change, Range 0, Low Limit**
09h	IRWTR0H	Reserved	**CEIR Pulse Change, Range 0, High Limit**
0Ah	IRWTR1L	Reserved	**CEIR Pulse Change, Range 1, Low Limit**
0Bh	IRWTR1H	Reserved	**CEIR Pulse Change, Range 1, High Limit**
0Ch	IRWTR2L	**CEIR Pulse Change, Range 2, Low Limit**	
0Dh	IRWTR2H	**CEIR Pulse Change, Range 2, High Limit**	
0Eh	IRWTR3L	**CEIR Pulse Change, Range 3, Low Limit**	
0Fh	IRWTR3H	**CEIR Pulse Change, Range 3, High Limit**	

4.9.4.1. Serial Port 1 And Serial Port 2 Configuration

Serial Ports 1 and 2 are identical, except for their reset values as shown in Table 4.93 below.

Logical Devices 2 and 3 (SP2 and SP1) Configuration

Serial Port 1 is designated as logical device 3, and Serial Port 2 as logical device 2. Table 4.93 lists the configuration registers which affect Serial Ports 1 and 2. Only the last register (F0h) is described here. See Sections 4.5.5 and 4.5.6 for descriptions of the others.

Table 4.93 Serial Ports 1 and 2 Configuration Registers

Index	Configuration Register or Action	Type	Reset Port 1	Reset Port 2
30h	Activate. See also bit 0 of the SIOCF1 register.	R/W	00h	00h
60h	Base Address MSB register. Bits 7-3 (for A15-11) are read only, 00000b.	R/W	03h	02h
61h	Base Address LSB register. Bit 2-0 (for A2-0) are read only, 000b.	R/W	F8h	F8h
70h	Interrupt Number	R/W	04h	03h
71h	Interrupt Type. Bit 1 is R/W; other bits are read only.	R/W	03h	03h
74h	Report no DMA Assignment	RO	04h	04h
75h	Report no DMA Assignment	RO	04h	04h
F0h	Serial Ports 1 and 2 Configuration register	R/W	02h	02h

Table 4.94 Serial Ports 1 and 2 Configuration Register - F0H

Bit	7	6	5	4	3	2	1	0
Name	Bank Select Enable	Reserved				Busy Indicator	Power Mode Control	TRI-STATE Control
Reset	0	0	0	0	0	0	1	0
RW. This register is reset by hardware to 02								

Bit	Description
7	**Bank Select Enable** Enables bank switching for Serial Ports 1 and 2. 0 = Disabled (default). 1 = Enabled
6-3	**Reserved**
2	**Busy Indicator** Read only bit used by power management software to decide when to power-down Serial Ports 1 and 2 logical devices. 0 = No transfer in progress (default). 1 = Transfer in progress.
1	**Power Mode Control** When the logical device is active in: 0 = Low power mode Serial Ports 1 and 2 Clock disabled. The output signals are set to their default states. The $\overline{RI}$ input signal can be programmed to generate an interrupt. Registers are maintained. (Unlike Active bit in Index 30 that also prevents access to Serial Ports 1 or 2 registers.) 1 = Normal power mode Serial Ports 1 and 2 clock enabled. Serial Ports 1 and 2 are functional when the respective logical devices are active (default).
0	**TRI-STATE Control** Controls the TRI-STATE status of the device output pins when it is inactive (disabled). 0 = Disabled (default) 1 = Enabled when device inactive

4.9.4.2. System Wake-Up Control (SWC) — Logical Device 4

Table 4.95 System Wake-Up Control (SWC) Configuration

Index	Configuration Register or Action	Type	Reset
30h	Activate. When bit 0 is cleared, the registers of this logical device are not accessible.[a]	R/W	00h
60h	Base Address MSB register	R/W	00h
61h	Base Address LSB register. Bits 3-0 (for A3-0) are read only, 0000b.	R/W	00h
70h	Interrupt Number (For routing the internal PWUREQ signal).	R/W	00h
71h	Interrupt Type. Bit 1 is read/write. Other bits are read only.	R/W	03h
74h	Report no DMA assignment	RO	04h
75h	Report no DMA assignment	RO	04h

a. The logical device registers are maintained, and all wake-up detection mechanisms are functional.

4.9.5. Keyboard/Mouse Control

The KBC is implemented physically as a single hardware module and houses two separate logical devices: a Mouse controller (logical device 5) and a Keyboard controller (logical device 6).

The hardware KBC module is integrated to provide the following pin functions: KBLOCK (P17), KBDAT, KBCLK, MDAT, and MCLK. KBLOCK is implemented as bi-directional, open-drain pins. The Keyboard and Mouse interfaces are implemented as bi-directional, open-drain pins. Their internal connections are shown in Figure 4-8.

P10, P11, P13-P16, P22-P27 of the KBC core are not available on dedicated pins; neither are T0 and T1, P10, P11, P22, P23, P26, P27, T0 and T1 are used to implement the Keyboard and Mouse interface.

The KBC executes a program fetched from an on-chip 2Kbyte ROM. The code programmed in this ROM is user-customizable. The KBC has two interrupt request signals: one for the Keyboard and one for the Mouse. The interrupt requests are implemented using ports P24 and P25 of the KBC core. The interrupt requests are controlled exclusively by the KBC firmware, except for the type and number, which are affected by configuration registers.

The interrupt requests are implemented as bi-directional signals. When an I/O port is read, all unused bits return the value latched in the output registers of the ports.

For KBC firmware that implements interrupt-on-OBF schemes, it is recommended to implement it as follows:

- Put the data in DBBOUT.
- Set the appropriate port bit to issue an interrupt request..

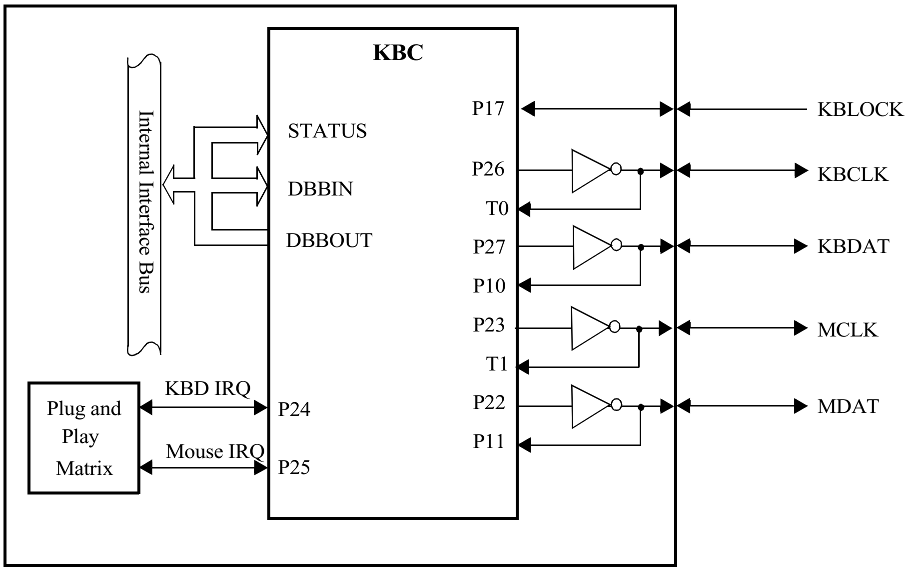


Figure 4-8 Keyboard and Mouse Interfaces

4.9.5.1. Logical Devices 5 and 6 (Mouse and Keyboard) Configuration

Tables 4.96 and 4.97 list the configuration registers which affect the Mouse and the Keyboard respectively. Only the last register (F0h) is described here. See Sections 4.5.5 and 4.5.6 for descriptions of the others.

Table 4.96 Mouse Configuration Registers

Index	Mouse Configuration Register or Action	Type	Reset
30h	Activate. See also bit 0 of the SIOCF1. When the Mouse of the KBC is inactive, the IRQ selected by the Mouse Interrupt Number register (index 70h) is not asserted. This register has no effect on host KBC commands handling the PS/2 Mouse.	R/W	00h
70h	Mouse Interrupt Number	R/W	0Ch
71h	Mouse Interrupt Type. Bits 1,0 are read/write; other bits are read only.	R/W	02h
74h	Report no DMA assignment	RO	04h
75h	Report no DMA assignment	RO	04h

Table 4.97 Keyboard Configuration Registers

Index	Keyboard Configuration Register or Action	Type	Reset
30h	Activate. See also bit 0 of the SIOCF1.	R/W	01h
60h	Data Port Base Address MSB register. Bits 7-3 (for A15-11) are read only, 00000b.	R/W	00h
61h	Data Port Base Address LSB register. Bits 2-0 are read only 000b.	R/W	60h
62h	Command Port Base Address MSB register. Bits 7-3 (for A15-11) are read only, 00000b.	R/W	00h
63h	Command Port Base Address LSB. Bits 2-0 are read only 100b.	R/W	64h
70h	KBC Interrupt Number	R/W	01h
71h	KBC Interrupt Type. Bits 1,0 are read/write; others are read only.	R/W	02h
74h	Report no DMA assignment	RO	04h
75h	Report no DMA assignment	RO	04h
F0h	KBC Configuration register	R/W	40h

Table 4.98 iKBC Configuration Register - F0H

Bit	7	6	5	4	3	2	1	0
Name	KBC Clock Source		Reserved					TRI-STATE Control
Reset	0	1	0	0	0	0	0	0
Required						0		

RW. This register is reset by hardware to 40H. To change the clock frequency of the KBC, perform the following:

1. Disable the KBC logical devices.
2. Change the frequency setting.
3. Enable the KBC logical devices.

Bit	Description
7-6	**KBC Clock Source** The clock source can be changed only when the KBC is inactive (disabled). **Bits** **7 6 Function** 0 0 8 MHz 0 1 12 MHz (default) 1 0 16 MHz 1 1 Reserved
5-1	**Reserved**. Use read-modify-write to change the value of the register. Do not change the value of these bits. Bit 2 must be 0.

Bit	Description
0	**TRI-STATE Control** If KBD is inactive (disabled) when this bit is set, the KBD pins (KBCLK and KBDAT) are in TRI-STATE. If Mouse is inactive (disabled) when this bit is set, the Mouse pins (MCLK and MDAT) are in TRI-STATE. 0 = Disabled (default) 1 = Enabled

4.9.6. Infrared Communication Port Configuration

Table 4.99 lists the configuration registers which affect the Infrared Communication Port. Only the last register (F0h) is described here. See Sections 4.5.5 and 4.5.6 for descriptions of the others.

Table 4.99 Infrared Communication Port Configuration Registers

Index	Configuration Register or Action	Type	Reset
30h	Activate. See also bit 0 of the SIOCF1 register.	R/W	00h
60h	Base Address MSB register. Bits 7-3 (for A15-11) are read only, 00000b.	R/W	03h
61h	Base Address LSB register. Bit 2-0 (for A2-0) are read only, 000b.	R/W	E8h
70h	Interrupt Number	R/W	00h
71h	Interrupt Type. Bit 1 is R/W; other bits are read only.	R/W	03h
74h	DMA Channel Select 0 (RX_DMA)	R/W	04h
75h	DMA Channel Select 1 (TX_DMA)	R/W	04h
F0h	Infrared Communication Port Configuration register	R/W	02h

Table 4.100 Infrared Communication Port Configuration Register - F0H

Bit	7	6	5	4	3	2	1	0
Name	Bank Select Enable	Reserved				Busy Indicator	Power Mode Control	TRI-STATE Control
Reset	0	0	0	0	0	0	1	0
RW. This register is reset by hardware to 02H								

Bit	Description
7	**Bank Select Enable** Enables bank switching. 0 = All attempts to access the extended registers are ignored (default). 1 = Enables bank switching.
6-3	**Reserved**

Bit	Description
2	**Busy Indicator** This read only bit can be used by power management software to decide when to power-down the device. 0 = No transfer in progress (default). 1 = Transfer in progress.
1	**Power Mode Control** When the logical device is active in: 0 = Low power mode Clock disabled. The output signals are set to their default states. The $\overline{\text{RI}}$ input signal can be programmed to generate an interrupt. Registers are maintained. (Unlike Active bit in Index 30 that also prevents access to device registers.) 1 = Normal power mode Clock enabled. The device is functional when the logical device is active (default).
0	**TRI-STATE Control** When enabled and the device is inactive, the logical device output pins are in TRI-STATE. One exception is the IRTX pin, which is driven to 0 when Serial Port 2 is inactive and is not affected by this bit. 0 = Disabled (default) 1 = Enabled

4.10. ACCESS.Bus Interface (ACB) Configuration

The ACB is a two-wire synchronous serial interface compatible with the ACCESS.bus physical layer.

The ACB uses a 24 MHz internal clock. The six runtime registers are shown below.

Table 4.101 ACB Runtime Registers

Offset	Mnemonic	Register Name	Type
00h	ACBSDA	ACB Serial Data	R/W
01h	ACBST	ACB Status	Varies per bit
02h	ACBCST	ACB Control Status	Varies per bit
03h	ACBCTL1	ACB Control 1	R/W
04h	ACBADDR	ACB Own Address	R/W
05h	ACBCTL2	ACB Control 2	R/W

Table 4.102 Access Bus Interface (ACB) Configuration

Index	Configuration Register or Action	Type	Reset
30h	Activate. See also bit 0 of the SIOCF1 register	R/W	00h
60h	Base Address MSB register	R/W	00h
61h	Base Address LSB register. Bits 2-0 (for A2-0) are read only, 000b.	R/W	00h
70h	Interrupt Number	R/W	00h
71h	Interrupt Type. Bit 1 is read/write. Other bits are read only.	R/W	03h
74h	Report no DMA assignment	RO	04h
75h	Report no DMA assignment	RO	04h
F0h	ACB Configuration register	R/W	00h

Table 4.103 ACB Configuration Register – F0H

Bit	7	6	5	4	3	2	1	0
Name	Reserved					Internal Pull-Up Enable	Reserved	
Reset	0	0	0	0	0	0	0	0
This register is reset by hardware to 00H.								

Bit	Description
7-3	Reserved
2	**Internal Pull-Up Enable** 0 = No internal pull-up resistors on SCL and SDA (default) 1 = Internal pull-up resistors on SCL and SDA
1-0	Reserved

4.11. Real-time Clock (RTC)

Table 4.104 shows the logical Device A real time clock configuration settings.

Table 4.104 Logical Device A (RTC) Configuration

Index	Configuration Register or Action	Type	Reset
30h	Activate. When bit 0 is cleared, the registers of this logical device are not accessible.[a]	R/W	00h
60h	Standard Base Address MSB register. Bits 7-3 (for A15-11) are read only, 00000b.	R/W	00h
61h	Standard Base Address LSB register. Bit 0 (for A0) is read only, 0b.	R/W	70h
62h	Extended Base Address MSB register. Bits 7-3 (for A15-11) are read only, 00000b.	R/W	00h
63h	Extended Base Address LSB register. Bit 0 (for A0) is read only, 0b.	R/W	72h
70h	Interrupt Number	R/W	08h
71h	Interrupt Type. Bit 1 is R/W; other bits are read only.	R/W	00h
74h	Report no DMA Assignment	R/W	04h
75h	Report no DMA Assignment	R/W	04h
F0h	RAM Lock register (RLR)	R/W	00h
F1h	Date of Month Alarm Offset register (DOMAO)	R/W	00h
F2h	Month Alarm Offset register (MONAO)	R/W	00h
F3h	Century Offset register (CENO)	R/W	00h

a. The logical device registers are maintained, and all RTC mechanisms are functional.

Table 4.105 RAM Lock Register (RLR) - F0H

Bit	7	6	5	4	3	2	1	0
Name	**Block Standard RAM**	**Block RAM Write**	**Block Extended RAM Write**	**Block Extended RAM Read**	**Block Extended RAM**	**Reserved**		
Reset	0	0	0	0	0	0	0	0
R/W: When a non-reserved bit is set to 1, it can be cleared only by hardware reset.								

Bit	Description
7	**Block Standard RAM** 0 = No effect on Standard RAM access (default) 1 = Read and write to locations 38h-3Fh of the Standard RAM are blocked, writes ignored, and reads return FFh

Bit	Description
6	**Block RAM Write** 0 = No effect on RAM access (default) 1 = Writes to RAM (Standard and Extended) are ignored
5	**Block Extended RAM Write.** This bit controls writes to bytes 00h-1Fh of the Extended RAM. 0 = No effect on the Extended RAM access (default) 1 = Writes to bytes 00h-1Fh of the Extended RAM are ignored
4	**Block Extended RAM Read** This bit controls read from bytes 00h-1Fh of the Extended RAM. 0 = No effect on Extended RAM access (default) 1 = Reads to bytes 00h-1Fh of the Extended RAM are ignored
3	**Block Extended RAM** This bit controls access to the Extended RAM 128 bytes. 0 = No effect on Extended RAM access (default) 1 = Read and write to the Extended RAM are blocked: writes are ignored and reads return FFh
2-0	**Reserved**

Table 4.106 Date Of Month Alarm Register Offset (DOMAO) – F1H

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**	**Date of Month Alarm Register Offset Value**						
Reset	0	0	0	0	0	0	0	0
R/W								

Bit	Description
7	**Reserved**
6-0	**Date of Month Alarm Register Offset Value**

Table 4.107 Month Alarm Register Offset (MAO) – F2H

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**	**Month Alarm Register Offset Value**						
Reset	0	0	0	0	0	0	0	0
R/W								

Bit	Description
7	**Reserved**
6-0	Month Alarm Register Offset Value

Table 4.108 Century Register Offset (CENO0) – F3H

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**	**Century Register Offset Value**						
Reset	0	0	0	0	0	0	0	0
R/W								

Bit	Description
7	**Reserved**
6-0	Century Register Offset Value

4.11.1. RTC Overview

The RTC provides timekeeping and calendar management capabilities. The RTC uses a 32.768 KHz signal as the basic clock for time-keeping. It also includes 242 bytes of battery-backed RAM for general-purpose use.

The RTC provides the following functions:

- Accurate timekeeping and calendar management
- Alarm at a predetermined time and/or date
- Three programmable interrupt sources
- Valid timekeeping during power-down, by utilizing external battery backup
- 242 bytes of battery-backed RAM
- RAM lock schemes to protect its content
- Internal oscillator circuit (the crystal itself is off-chip), or external clock supply for the 32.768KHz clock
- A century counter
- PnP support
- Relocatable index and data registers
- Module access enable/disable option
- Host interrupt enable/disable option
- Additional low-power features such as:
- Automatic switching from battery to VCC_IO
- Internal power monitoring on the VRT bit
- Oscillator disabling to save battery during storage
- Software compatible with the DS1287 and MC146818

4.11.2. Functional Description

4.11.2.1. Bus Interface

The RTC function is initially mapped to the default SuperI/O locations at indexes 70h to 73h (two Index/Data pairs). These locations may be reassigned, in compliance with Plug and Play requirements.

To access Bank 1, you most set the RAMSEL bit to ONE. RAMSEL bit is located at the general configuration registers, at bit 5 of the KRR (Keyboard and RTC Control Register), at index 05_{16}.

4.11.2.2. RTC Clock Generation

The RTC uses a 32.768 KHz clock signal as the basic clock for timekeeping. The 32.768 KHz clock can be supplied by the i48 MHz clock, or by an oscillator (see Section 5.11. ‘System Clocking’ on page 441).

4.11.2.3. Timing Generation

The timing generation function divides the 32.768 KHz clock by 215 to derive a 1 Hz signal, which serves as the input for the seconds counter. This is performed by a divider chain composed of 15 divide-by-two latches, as shown in Figure 4-9.

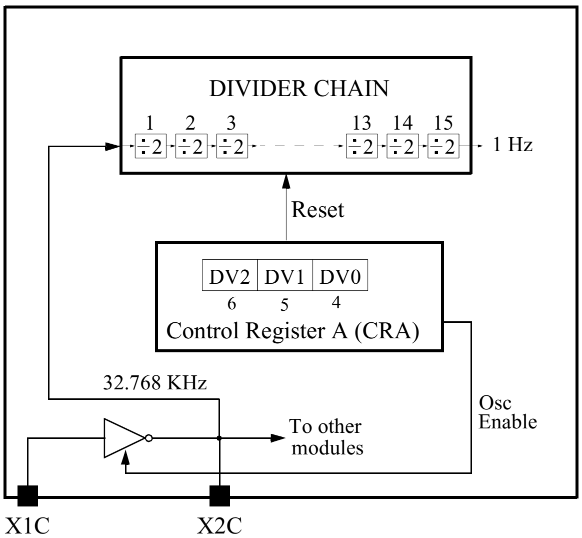


Figure 4-9 Divider Chain Control

Bits 6-4 (DV2-0) of the CRA Register control the following functions:

- Normal operation of the divider chain (counting)
- Divider chain reset to 0
- Oscillator activity when only V_{BAT} power is present (backup state).

The divider chain can be activated by setting normal operational mode (bits 6-4 of CRA). The first update occurs 500 ms after divider chain activation.

Bits 3-0 of the CRA Register select one the of fifteen taps from the divider chain to be used as a periodic interrupt. The periodic flag becomes active after half of the programmed period has elapsed, following divider chain activation.

See Table 4.126 on page 309 for more details.

4.11.2.4. Timekeeping

Data Format

Time is kept in BCD or binary format, as determined by bit 2 (DM) of Control Register B (CRB), and in either 12 or 24-hour format, as determined by bit 1 of this register.

Note: When changing the above formats, re-initialize all the time registers.

Daylight Saving

Daylight saving time exceptions are handled automatically, as described in Table 4.129, "RTC Control Register B (CRB) - 0BH," on page 311.

Leap Years

Leap year exceptions are handled automatically by the internal calendar function. Every four years, February is extended to 29 days. Year 2000 is a leap year.

4.11.2.5. Updating

The time and calendar registers are updated once per second regardless of bit 7 (SET) of the CRB Register. Since the time and calendar registers are updated serially, unpredictable results may occur if they are accessed during the update. Therefore, you must ensure that reading or writing to the time storage locations does not coincide with a system update of these locations. There are several methods to avoid this contention.

Method 1

1. Set bit 7 of the CRB Register to 1. This takes a "snapshot" of the internal time registers and loads them into the user copy registers. The user copy registers are seen when accessing the RTC from outside, and

are part of the double buffering mechanism. You may keep this bit set for up to 1 second, since the time/calendar chain continue to be updated once per second.

2. Read or write the required registers (since bit 1 is set, you access the user copy registers). If you perform a read operation, the information you read is correct from the time when bit 1 was set. If you perform a write operation, you write only to the user copy registers.
3. Reset bit 1 to 0. During the transition, the user copy registers update the internal registers, using the double buffering mechanism to ensure that the update is performed between two time updates. This mechanism enables new time parameters to be loaded in the RTC.

Method 2

1. Access the RTC registers after detection of an Update Ended interrupt. This implies that an update has just been completed and 999 ms remain until the next update.
2. To detect an Update Ended interrupt, you may either:
   a. Poll bit 4 of the CRC Register.
   b. Use the following interrupt routine:
      — Set bit 4 of the CRB Register.
      — Wait for an interrupt from interrupt pin.
      — Clear the IRQF flag of the CRC Register before exiting the interrupt routine.

Method 3

Poll bit 7 of the CRA Register. The update occurs 244 ms after this bit goes high. Therefore, if a 0 is read, the time registers remain stable for at least 244 ms. See Table 4.126 on page 309.

Method 4

Use a periodic interrupt routine to determine if an update cycle is in progress, as follows:

1. Set the periodic interrupt to the desired period.
2. Set bit 6 of the CRB Register to enable the interrupt from periodic interrupt.
3. Wait for the periodic interrupt appearance. This indicates that the period represented by the following expression remains until another update occurs: [(Period of periodic interrupt / 2) + 244 ms]

4.11.2.6. Alarms

The timekeeping function can be set to generate an alarm when the current time reaches a stored alarm time. After each RTC time update (every 1 second), the seconds, minutes, hours, date of month and month counters are compared with their corresponding registers in the alarm settings. If equal, bit 5 of the CRC Register is set. If the Alarm Interrupt Enable bit was previously set (bit 5 of the CRB Register), interrupt request pin will also be active.

Any alarm register may be set to “Unconditional Match” by setting bits 7 and 6 to 11. This combination, not used by any BCD or binary time codes, results in a periodic alarm. The rate of this periodic alarm is determined by the registers that were set to “Unconditional Match”.

For example, if all but the seconds and minutes alarm registers are set to “Unconditional Match”, an interrupt generates every hour at the specified minute and second. If all but the seconds, minutes and hours alarm registers are set to “Unconditional Match”, an interrupt generates every day at the specified hour, minute and second.

4.11.2.7. Power Supply

The device is supplied from three supply voltages.

- System power supply voltage, $V_{CC.}$
- System power supply voltage, V_{CC-IO}
- Backup voltage, from low capacity Lithium battery, V_{BAT}

Note: The ZFx86 contains no reverse polarity protection.

The RTC is supplied from one of two power supplies, V_{CC-IO} or V_{BAT}, according to their levels. An internal voltage comparator delivers the control signals to a pair of switches. Battery backup voltage V_{BAT} maintains the correct time and saves the CMOS memory when the external voltage is absent, due to power failure or disconnection of the external AC/DC input power supply.

To assure that the module uses power from the external source and not from V_{BAT}, the voltage should be maintained above its minimum.

The actual voltage point where the module switches from V_{BAT} to V_{CC-IO} is lower than the minimum workable battery voltage, but high enough to guarantee the correct functionality of the oscillator and the CMOS RAM.

4.11.2.8. System Bus Lockout

During power on or power off, spurious bus transactions from the host may occur. To protect the RTC internal registers from corruption, all inputs are automatically locked out. The lockout condition is asserted when V_{CC-IO} is lower than V_{BAT}.

4.11.2.9. Power-Up Detection

When system power is restored after a power failure or power off state, the lockout condition continues for a delay of 62 ms (minimum) to 125 ms (maximum) after the RTC switches from battery to system power.

The lockout condition is switched off immediately in the following situations:

- If the Divider Chain Control bits, DV0-2, (bits 6-4 in the CRA Register) specify a normal operation mode (01X or 100), all input signals are enabled immediately upon detection of system voltage above that of the battery voltage. See Table 4.126 on page 309
- When battery voltage is below 1 Volt and HMR is 1, all input signals are enabled immediately upon detection of system voltage above that of battery voltage. This also initializes registers at offsets 00h through 0Dh in the ZF-Logic See Table 5.2 on page 400.
- If bit 7 (VRT) of the CRD Register is 0, all input signals are enabled immediately upon detection of system voltage above V_{BAT}.

4.11.2.10. Oscillator Activity

The RTC oscillator is active if:

- V_{CC_IO} power supply is higher than its minimum specified at the DC spec, independent of the battery voltage, V_{BAT}
- V_{BAT} power supply is higher than V_{BATMIN}, regardless if V_{CC_IO} is present or not.

The RTC oscillator is disabled if:

- During power-down (V_{BAT} only), the battery voltage drops below V_{BATMIN}. When this occurs, the oscillator may be disabled and its functionality cannot be guaranteed.
- Software wrote 00X to DV2-0 bits of the CRA Register (see Table 4.126 on page 309) and VCC_IO is removed. This disables the oscillator and decreases the power consumption from the battery connected to the V_{BAT} pin. When disabling

the oscillator, the CMOS RAM is not affected as long as the battery is present at a correct voltage level.

If the RTC oscillator becomes inactive, the following features will be dysfunctional/disabled:

- Timekeeping
- Periodic interrupt
- Alarm

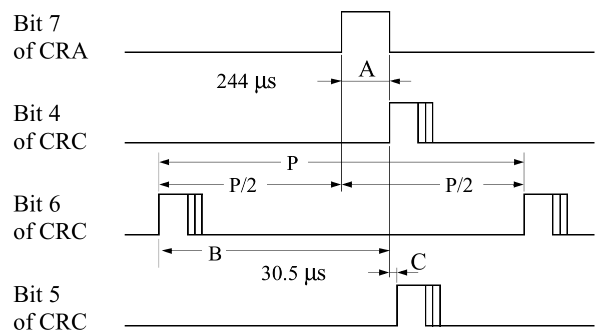
4.11.2.11. Interrupt Handling

The RTC has a single Interrupt Request line which handles the following three interrupt conditions:

- Periodic interrupt
- Alarm interrupt
- Update end interrupt.

The interrupts are generated if the respective enable bits in the CRB Register are set prior to an interrupt event occurrence. Reading the CRC Register clears all interrupt flags. Thus, when multiple interrupts are enabled, the interrupt service routine should first read and store the CRC Register, and then deal with all pending interrupts by referring to this stored status.

If an interrupt is not serviced before a second occurrence of the same interrupt condition, the second interrupt event is lost. Figure 4-10 illustrates the interrupt timing in the RTC.



Flags (and IRQ) are reset at the conclusion of CRC read or by reset.

A = Update In Progress bit high before update occurs = 244 µs
B = Periodic interrupt to update
= Period (periodic int) / 2 + 244 µs
C = Update to Alarm Interrupt = 30.5 µs
P = Period is programmed by RS3-0 of CRA

Figure 4-10 Interrupt/Status Timing

4.11.2.12. Battery-Backed RAMs and Registers

The RTC has two battery-backed RAMs and 17 registers, used by the logical units themselves. Battery-backup power enables information retention during system power down.

The RAMs are:

- Standard RAM
- Extended RAM

Note: The ZFx86 contains no reverse polarity protection.

The memory maps and register content of the RAMs is illustrated in Table 4.136, Table 4.137, and Table 4.138.

The first 14 bytes and 3 programmable bytes of the Standard RAM are overlaid by time, alarm data and control registers. The rest 111 bytes are general-purpose memory.

Registers with reserved bits should be written in “Read-Modify-Write” method.

All register locations within the device are accessed by the RTC Index and Data registers (at base address and base address+1).

The Index register points to the register location being accessed, and the Data register contains the data to be transferred to or from the location. An additional 128 bytes of battery-backed RAM (also called Extended RAM) may be accessed via a second pair of Index and Data registers.

Access to the two RAMs may be locked. For details see Table 4.105 on page 295.

4.11.3. RTC Configuration Registers

Access the RTC configuration registers at any time during normal operation mode; for example, when VDD and VCC_IO are within the recommended operation range. This access is disabled during battery-backed operation.

Table 4.109 RTC Configuration Register Map

Location	Mnemonic	Name	Type	Reset	Reference
Device specific	RLR	RAM Lock Register	R/W	HW	Table 4.110
Device specific	DOMAO	Date of Month Alarm Register Offset	R/W	HW or SW	Table 4.111
Device specific	MONAO	Month Alarm Register Offset	R/W	HW or SW	Table 4.112
Device specific	CENO	Century Register Offset	R/W	HW or SW	Table 4.113

Table 4.110 RAM Lock Register (RLR)

Bit	7	6	5	4	3	2	1	0
Name	**Block Standard RAM**	**Block RAM Write**	**Block Extended RAM Write**	**Block Extended RAM Read**	**Block Extended RAM**	**Reserved**		
Reset	0	0	0	0	0	0		
R/W. The location is device specific. When a non-reserved bit is set to 1, it can be cleared only by hardware reset.								

Bit	Description
7	**Block Standard RAM** 0 = No effect on Standard RAM access (default) 1 = Read and write to locations 38h-3Fh of the Standard RAM are blocked, writes ignored, and reads return FHA
6	**Block RAM Write** 0 = No effect on RAM access (default) 1 = Writes to RAM (Standard and Extended) are ignored
5	**Block Extended RAM Write** This bit controls writes to bytes 00h-1Fh of the Extended RAM. 0 = No effect on the Extended RAM access (default) 1 = Writes to bytes 00h-1Fh of the Extended RAM are ignored

Bit	Description
4	**Block Extended RAM Read** This bit controls read from bytes 00h-1Fh of the Extended RAM. 0 = No effect on Extended RAM access (default) 1 = Reads to bytes 00h-1Fh of the Extended RAM are ignored
3	Block Extended RAM This bit controls access to the Extended RAM 128 bytes. 0 = No effect on Extended RAM access (default) 1 = Read and write to the Extended RAM are blocked: writes are ignored and reads return FFh
2-0	**Reserved**

Table 4.111 Date Of Month Alarm Register Offset (DOMAO)

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**	**Date of Month Alarm Register Offset Value**						
Reset	0	0	0	0	0	0	0	0
R/W. The location is device specific.								

Bit	Description
7	**Reserved**
6-0	Date of Month Alarm Register Offset Value

Table 4.112 Month Alarm Register Offset (DOMAO)

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**	**Month Alarm Register Offset Value**						
Reset	0	0						
R/W. The location is device specific.								

Bit	Description
7	**Reserved**
6-0	Month Alarm Register Offset Value

Table 4.113 Century Register Offset (CENO)

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**	**Century Register Offset Value**						
Reset	0	0						
R/W. The location is device specific.								

Bit	Description
7	**Reserved**
6-0	Century Register Offset Value

Table 4.114 RTC Configuration Register Bitmap

Register		Bits							
Location	Mnemonic	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Device specific	RLR	RAM Lock	RAM Mask Write	RAM Block Write	RAM Block Read	Upper RAM Block	Reserved		
Device specific	DOMAO	Reserved	Date of Month Alarm Register Offset Value						
Device specific	MONAO	Reserved	Month Alarm Register Offset Value						
Device specific	CENO	Reserved	Century Register Offset Value						

4.11.4. RTC Registers

The RTC registers can be accessed at any time during normal operation mode; that is when VCC_IO is within the recommended operation range. This access is disabled during battery-backed operation. The write operation to these registers is also disabled if bit 7 of the CRD Register is 0 (see Table 4.131 on page 314).

Note: Before attempting to perform any start-up procedures, make sure to read about bit 7 (VRT) of the CRD Register.

This section describes the RTC Timing and Control Registers that control basic RTC functionality.

Table 4.115 RTC Register Map

Index	Mnemonic	Name	Type	Reset	Reference
00h	SEC	Seconds Register	R/W	V_{PP} PUR	Table 4.116
01h	SECA	Seconds Alarm Register	R/W	V_{PP} PUR	Table 4.117
02h	MIN	Minutes Register	R/W	V_{PP} PUR	Table 4.118
03h	MINA	Minutes Alarm Register	R/W	V_{PP} PUR	Table 4.119
04h	HOR	Hours Register	R/W	V_{PP} PUR	Table 4.120
05h	HORA	Hours Alarm Register	R/W	V_{PP} PUR	Table 4.121
06h	DOW	Day Of Week Register	R/W	V_{PP} PUR	Table 4.122
07h	DOM	Date Of Month Register	R/W	V_{PP} PUR	Table 4.123
08h	MON	Month Register	R/W	V_{PP} PUR	Table 4.124
09h	YER	Year Register	R/W	V_{PP} PUR	Table 4.125
0Ah	CRA	RTC Control Register A	R/W	Bit specific	Table 4.126
0Bh	CRB	RTC Control Register B	R/W	Bit specific	Table 4.129
0Ch	CRC	RTC Control Register C	R/O	Bit specific	Table 4.130
0Dh	CRD	RTC Control Register D	R/O	V_{PP} PUR	Table 4.131
Programmable[a]	DOMA	Date of Month Alarm Register	R/W	V_{PP} PUR	Table 4.132
Programmable[a]	MONA	Month Alarm Register	R/W	V_{PP} PUR	Table 4.133
Programmable[a]	CEN	Century Register	R/W	V_{PP} PUR	Table 4.134

a. Overlaid on RAM bytes in range 0Eh-7Fh.

Table 4.116 Seconds Register (SEC)) – Index 00H

Bit	7	6	5	4	3	2	1	0
Name	**Seconds Data**							
Reset	0	0	0	0	0	0	0	0
R/W.								

Bit	Description
7-0	**Seconds Data** Values may be 00 to 59 in BCD format or 00 to 3B in Binary format.

Table 4.117 Seconds Alarm Register (SECA)) – 01H

Bit	7	6	5	4	3	2	1	0
Name	**Seconds Alarm Data**							
Reset	0	0	0	0	0	0	0	0
R/W.								

Bit	Description
7-0	**Seconds Alarm Data** Values may be 00 to 59 in BCD format or 00 to 3B in Binary format. When bits 7 and 6 are both set to one (“11”), unconditional match is selected.

Table 4.118 Minutes Register (MIN)) – 02H

Bit	7	6	5	4	3	2	1	0
Name	**Minutes Data**							
Reset	0	0	0	0	0	0	0	0
R/W.								

Bit	Description
7-0	**Minutes Data** Values may be 00 to 59 in BCD format or 00 to 3B in Binary format.

Table 4.119 Minutes Alarm Register (MINA) – 03H

Bit	7	6	5	4	3	2	1	0
Name	**Minutes Alarm Data**							
Reset	0	0	0	0	0	0	0	0
R/W.								

Bit	Description
7-0	**Minutes Alarm Data** Values may be 00 to 59 in BCD format or 00 to 3B in Binary format. When bits 7 and 6 are both set to one (“11”), unconditional match is selected.

Table 4.120 Hours Register (HOR) – 04H

Bit	7	6	5	4	3	2	1	0
Name	**Hours Data**							
Reset	0	0	0	0	0	0	0	0
R/W.								

Bit	Description
7-0	**Hours Data** For 12-hour mode, values may be 01 to 12 (AM) and 81 to 92 (PM) in BCD format or 01 to 0C (AM) and 81 to 8C (PM) in Binary format. For 24-hour mode, values may be 0- to 23 in BCD format or 00 to 17 in Binary format.

Table 4.121 Hours Alarm Register (HORA) – 05H

Bit	7	6	5	4	3	2	1	0
Name	**Hours Alarm Data**							
Reset	0	0	0	0	0	0	0	0
R/W.								

Bit	Description
7-0	**Hours Alarm Data** For 12-hour mode, values may be 01 to 12 (AM) and 81 to 92 (PM) in BCD format or 01 to 0C (AM) and 81 to 8C (PM) in Binary format. For 24-hour mode, values may be 0- to 23 in BCD format or 00 to 17 in Binary format. When bits 7 and 6 are both set to one (“11”), unconditional match is selected.

Table 4.122 Day Of Week Register (DOW) – 06H

Bit	7	6	5	4	3	2	1	0
Name	**Day Of Week Data**							
Reset	0	0	0	0	0	0	0	0
R/W.								

Bit	Description
7-0	**Day Of Week Data**. Values may be 01 to 07 in BCD format or 01 to 07 in Binary format.

Table 4.123 Date Of Month Register (DOM) – 07H

Bit	7	6	5	4	3	2	1	0
Name	**Date Of Month Data**							
Reset	0	0	0	0	0	0	0	0
R/W.								

Bit	Description
7-0	**Date Of Month Data** Values may be 01 to 31 in BCD format or 01 to 1F in Binary format.

Table 4.124 Month Register (MON) - 08H

Bit	7	6	5	4	3	2	1	0
Name	**Month Data**							
Reset	0	0	0	0	0	0	0	0
R/W.								

Bit	Description
7-0	**Month Data** Values may be 01 to 12 in BCD format or 01 to 0C in Binary format.

Table 4.125 Year Register (YER) - 09H

Bit	7	6	5	4	3	2	1	0
Name	**Year Data**							
Reset	0							
R/W.								

Bit	Description
7-0	**Year Data** Values may be 00 to 99 in BCD format or 00 to 63 in Binary format.

Table 4.126 RTC Control Register A (CRA) – 0AH

Bit	7	6	5	4	3	2	1	0
Name	**Update in Progress**	**Divider Chain Control 2-0**			**Periodic Interrupt Rate Select 3-0**			
Reset	0	0	1	0	0	0	0	0
R/W. This register controls test selection, among other functions. This register cannot be written before reading bit 7 of the CRD Register.								

Bit	Description
7	**Update in Progress** This RO bit is not affected by reset. This bit reads 0 when bit 7 of the CRB Register is 1. 0 =: Timing registers not updated within 244 ms 1 = Timing registers updated within 244 ms
6-4	**Divider Chain Control** These R/W bits control the configuration of the divider chain for timing generation and register bank selection. See Table 4.127. They are cleared to 000 as long as bit 7 of the CRD Register is reads 0.
3-0	**Periodic Interrupt Rate Select** These R/W bits select one of fifteen output taps from the clock divider chain to control the rate of the periodic interrupt. See Table 4.128 and Figure 4-9. They are cleared to 000 as long as bit 7 of the CRD Register is reads 0.

Table 4.127 Divider Chain Control and Test Selection

DV2	DV1	DV0	Configuration
CRA 6	**CRA 5**	**CRA 4**	
0	0	X	Oscillator Disabled
0	1	0	Normal Operation
0	1	1	Test
1	0	X	
1	1	X	Divider Chain Reset

Table 4.128 Periodic Interrupt Rate Encoding

Rate Select 3 2 1 0	Periodic Interrupt Rate (ms)	Divider Chain Output
0 0 0 0	No interrupts	
0 0 0 1	3.906250	7
0 0 1 0	7.812500	8
0 0 1 1	0.122070	2
0 1 0 0	0.244141	3
0 1 0 1	0.488281	4
0 1 1 0	0.976562	5
0 1 1 1	1.953125	6
1 0 0 0	3.906250	7
1 0 0 1	7.812500	8
1 0 1 0	15.625000	9
1 0 1 1	31.250000	10
1 1 0 0	62.500000	11
1 1 0 1	125.000000	12
1 1 1 0	250.000000	13
1 1 1 1	500.000000	14

Table 4.129 RTC Control Register B (CRB) - 0BH

Bit	7	6	5	4	3	2	1	0
Name	**Set Mode**	**Periodic Interrupt Enable**	**Alarm Interrupt Enable**	**Update Ended Interrupt Enable**	**Reserved**	**Data Mode**	**Hour Mode**	**Daylight Saving**
Reset	0	0	0	0	0	0	0	0
R/W.								

Bit	Description
7	**Set Mode.**This bit is reset at VPP power-up reset only. 0 = Timing updates occur normally 1 = User copy of time is “frozen”, allowing the time registers to be accessed whether or not an update occurs
6	**Periodic Interrupt Enable** Bits 3-0 of the CRA Register page 309 determine the rate at which this interrupt is generated. It is cleared to 0 on RTC reset (that is, hardware or software reset) or when RTC is disabled. 0 = Disabled 1 = Enabled
5	**Alarm Interrupt Enable** This interrupt is generated immediately after a time update in which the seconds, minutes, hours, date and month time equal their respective alarm counterparts. It is cleared to 0 as long as bit 7 of the CRD Register is reads 0. 0 = Disabled 1 = Enabled
4	**Update Ended Interrupt Enable** This interrupt is generated when an update occurs. It is cleared to 0 on RTC reset (for example, hardware or software reset) or when the RTC is disable. 0 = Disabled 1 = Enabled
3	Reserved Bit always reads 0.
2	**Data Mode** This bit is reset at V_{PP} power-up reset only. 0 = BCD format enabled 1 = Binary format enabled
1	**Hour Mode** This bit is reset at V_{PP} power-up reset only. 0 = 12-hour format enabled 1 = 24-hour format enabled

Bit	Description
0	**Daylight Saving** This bit is reset at V_{PP} power-up reset only. 0 = Disabled 1 = Enabled In the spring, time advances from1:59:59 AM to 3:00:00 AM on the first Sunday in April. In the fall, time returns from 1:59:59 AM to 1:00:00 AM on the last Sunday in October.

Table 4.130 RTC Control Register C (CRC) - 0CH

Bit	7	6	5	4	3	2	1	0
Name	**IRQ Flag**	**Periodic Interrupt Flag**	**Alarm Interrupt Flag**	**Update Ended Interrupt Flag**	**Reserved**			
Reset	0	0	0	0	0	0	0	0
R/O.								

Bit	Description
7	**IRQ Flag** This RO bit mirrors the value on the interrupt output signal. When interrupt is active, IRQF is 1. To clear this bit (and deactivate the interrupt pin), read the CRC Register as the flag bits UF, AF and PF are cleared after reading this register. 0 = IRQ inactive 1 = Logic equation is true: (UIE and UF) or (AIE and AF) or (PIE and PF).
6	**Periodic Interrupt Flag** This RO bit is cleared to 0 on RTC reset (that is, hardware or software reset) or the RTC disabled. In addition, this bit is cleared to 0 when this register is read. 0 = No transition occurred on the selected tap since the last read 1 = Transition occurred on the selected tap of the divider chain
5	**Alarm Interrupt Flag** This RO bit is cleared to 0 as long as bit 7 of the CRD Register is reads 0. In addition, this bit is cleared to 0 when this register is read. 0 = No alarm detected since the last read 1 = Alarm condition detected
4	**Update Ended Interrupt Flag** This RO bit is cleared to 0 on RTC reset (for example, hardware or software reset) or the RTC disabled. In addition, this bit is cleared to 0 when this register is read. 0 = No update occurred since the last read 1 = Time registers updated
3-0	**Reserved**

Table 4.131 RTC Control Register D (CRD) - 0DH

Bit	7	6	5	4	3	2	1	0
Name	**Valid RAM and Time**	**Reserved**						
Reset	0	0						
R/O.								

Bit	Description
7	**Valid RAM and Time**. This bit senses the voltage that feeds the RTC (VCC_IO or) and indicates whether or not it was too low since the last time this bit was read. If it was too low, the RTC contents (time/calendar registers and CMOS RAM) is not valid. 0 = The voltage that feeds the RTC was too low. 1 = RTC contents (time/calendar registers and CMOS RAM) valid
6-0	**Reserved**

Table 4.132 Date of Month Alarm Register (DOMA)

Bit	7	6	5	4	3	2	1	0
Name	**Date of Month Alarm Data**							
Reset	1	1	0	0	0	0	0	0
R/W.Location is a Programmable Index								

Bit	Description
7-0	**Date of Month Alarm Data** Values may be 01 to 31 in BCD format or 01 to 1F in Binary format. When bits 7 and 6 are both set to one (“11”), unconditional match is selected (default).

Table 4.133 Month Alarm Register (MONA)

Bit	7	6	5	4	3	2	1	0
Name	**Month Alarm Data**							
Reset	1	1	0	0	0	0	0	0
R/W.Location is a Programmable Index								

Bit	Description
7-0	**Month Alarm Data** Values may be 01 to 12 in BCD format or 01 to 0C in Binary format. When bits 7 and 6 are both set to one (“11”), unconditional match is selected (default).

Table 4.134 Century Register (CEN)

Bit	7	6	5	4	3	2	1	0
Name	**Century Data**							
Reset	**0**							
R/W.Location is a Programmable Index								

Bit	Description
7-0	**Century Data** Values may be 00 to 99 in BCD format or 00 to 63 in Binary format.

Table 4.135 BCD and Binary Formats

Parameter	BCD Format	Binary Format
Seconds	00 to 59	00 to 3B
Minutes	00 to 59	00 to 3B
Hours	12-hour mode: 01 to 12 (AM) 81 to 92 (PM) 24-hour mode: 00 to 23	12-hour mode: 01 to 0C (AM) 81 to 8C (PM) 24-hour mode: 00 to 17
Day	01 to 07 (Sunday = 01)	01 to 07
Date	01 to 31	01 to 1F
Month	01 to 12 (January = 01)	01 to 0C
Year	00 to 99	00 to 63
Century	00 to 99	00 to 63

4.11.4.1. Usage Hints

1. Read bit 7 of the CRD Register at each system power-up to validate the contents of the RTC registers and the CMOS RAM. When this bit is 0, the contents of these registers and the CMOS RAM are questionable. This bit is reset when the backup battery voltage is too low. The voltage level at which this bit is reset is below the minimum recommended battery voltage, 2.4 V. Although the RTC oscillator may function properly and the register contents may be correct at lower than 2.4 V, this bit is reset since correct functionality cannot be guaranteed. System BIOS may use a checksum method to revalidate the contents of the CMOS-RAM. The checksum byte should be stored in the same CMOS RAM.
2. Change the backup battery while normal operating power is present, and not in backup mode, to maintain valid time and register information. If a low leakage capacitor is connected to V_{BAT}, the battery may be changed in backup mode.
3. A rechargeable NiCd battery may be used instead of a non-rechargeable Lithium battery. This is a preferred solution for portable systems, where small size components is essential.
4. A supercap capacitor may be used instead of the normal Lithium battery. In a portable system usually the VCC_IO voltage is always present since the power management stops the system before its voltage falls to low. The supercap capacitor

in the range of 0.047-0.47 F should supply the power during the battery replacement.

Table 4.136 RTC Register Bitmap

Register		Bits							
Index	Mnemonic	7	6	5	4	3	2	1	0
00h	**SEC**	**Seconds Data**							
01h	**SECA**	**Seconds Alarm Data**							
02h	**MIN**	**Minutes Data**							
02h	**MINA**	**Minutes Alarm Data**							
04h	**HOR**	**Hours Data**							
05h	**HORA**	**Hours Alarm Data**							
06h	**DOW**	**Day of Week Data**							
07h	**DOM**	**Date of Month Data**							
08h	**MON**	**Month Data**							
09h	**YER**	**Year Data**							
0Ah	**CRA**	**Update in Progress**	**Divider Chain Control 2-0**			**Periodic Interrupt Rate Select 3-0**			
0Bh	**CRB**	**Set Mode**	**Periodic Interrupt**	**Alarm Interrupt**	**Update Ended Interrupt**	**Reserved**	**Data Mode**	**Hour Mode**	**Daylight Saving**
0Ch	**CRC**	**IRQ Flag**	**Periodic Interrupt Flag**	**Alarm Interrupt Flag**	**Update Ended Interrupt Flag**	**Reserved**			
0Dh	**CRD**	**Valid RAM and Time**	**Reserved**						
Prog.	**DOMA**	**Date of Month Alarm Data**							
Prog.	**MONA**	**Month Alarm Data**							
Prog.	**CEN**	**Century Data**							

4.11.5. RTC General-purpose RAM Map

Table 4.137 Standard RAM Map

Index	Description
0Eh - 7Fh[a]	Battery-backed General-purpose 111-byte RAM

a. Battery-backed 111-byte RAM (114 - 3 overlaid registers).

Table 4.138 Extended RAM Map

Index	Description
00h - 7Fh	Battery-backed General-purpose 128-byte RAM.

4.12. ACCESS.bus Interface (ACB)

The ACB is a two-wire synchronous serial interface compatible with the ACCESS.bus physical layer. The ACB is also compatible with Intel's SMBus and Philips' I2C. Configure the ACB as a bus master or slave, and maintain bi-directional communication with both multiple master and slave devices. As a slave device, the ACB may issue a request to become the bus master.

The ACB allows easy interfacing to a wide range of low-cost memories and I/O devices, including: EEPROMs, SRAMs, timers, ADC, DAC, clock chips and peripheral drivers.

This text describes the general ACB functional block. A device may include a different implementation. For device specific implementation, see Section 4.5.4. 'Device Architecture and Configuration' on page 256.

4.12.1. Functional Description

The ACCESS.bus protocol uses a two-wire interface for bi-directional communication between the ICs connected to the bus. The two interface lines are the Serial Data Line (SDL) and the Serial Clock Line (SCL). These lines should be connected to a positive supply via an internal or external pull-up resistor, and remain high even when the bus is idle.

Each IC has a unique address and can operate as a transmitter or a receiver (though some peripherals are only receivers).

During data transactions, the master device initiates the transaction, generates the clock signal and terminates the transaction. For example, when the ACB initiates a data transaction with an attached ACCESS.bus compliant peripheral, the ACB becomes the master. When the peripheral responds and transmits data to the ACB, their master/slave (data transaction initiator and clock generator) relationship is unchanged, even though their transmitter/receiver functions are reversed.

4.12.1.1. Data Transactions

One data bit is transferred during each clock pulse. Data is sampled during the high state of the serial clock (SCL). Consequently, throughout the clock's high period, the data should remain stable (see Figure 4-11). Any changes on the SDA line during the high state of the SCL and in the middle of a transaction aborts the current transaction. New data should be sent during the low SCL state. This protocol permits a single data line to transfer both command/control information and data, using the synchronous serial clock.

Each data transaction is composed of a Start Condition, a number of byte transfers (set by the software) and a Stop Condition to terminate the transaction. Each byte is transferred with the most significant bit first, and after each byte (8 bits), an Acknowledge signal must follow. The following sections provide further details of this process.

During each clock cycle, the slave can stall the master while it handles the previous data or prepares new data. This can be done for each bit transferred, or on a byte boundary, by the slave holding SCL low to extend the clock-low period. Typically, slaves extend the first clock cycle of a transfer if a byte read has not yet been stored, or if the next byte to be transmitted is not yet ready. Some microcontrollers, with limited hardware support for ACCESS.bus, extend the access after each bit, thus allowing the software to handle this bit.

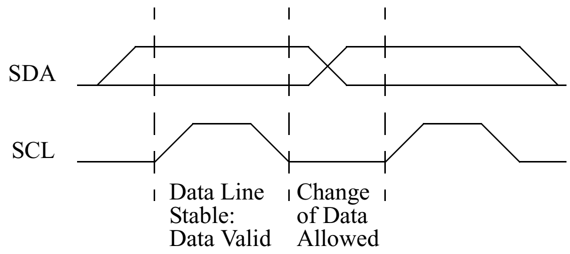


Figure 4-11 Bit Transfer

4.12.1.2. Start and Stop Conditions

The ACCESS.bus master generates Start and Stop Conditions (control codes). After a Start Condition is generated, the bus is considered busy and retains this status for a certain time after a Stop Condition is generated. A high to low transition of the data line (SDA) while the clock (SCL) is high indicates a Start Condition. A low to high transition of the SDA line while the SCL is high indicates a Stop Condition (Figure 4-12).

In addition to the first Start Condition, a repeated Start Condition can be generated in the middle of a transaction. This allows another device to be accessed, or a change in the direction of data transfer.

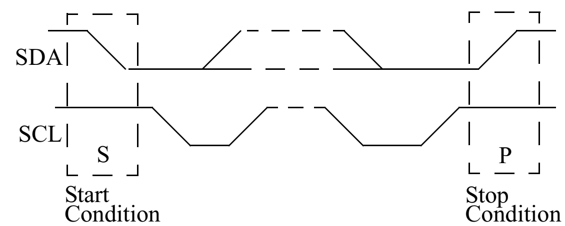


Figure 4-12 Start and Stop Conditions

4.12.1.3. Acknowledge (ACK) Cycle

The ACK cycle consists of two signals: the ACK clock pulse sent by the master with each byte transferred, and the ACK signal sent by the receiving device (see Figure 4-13).

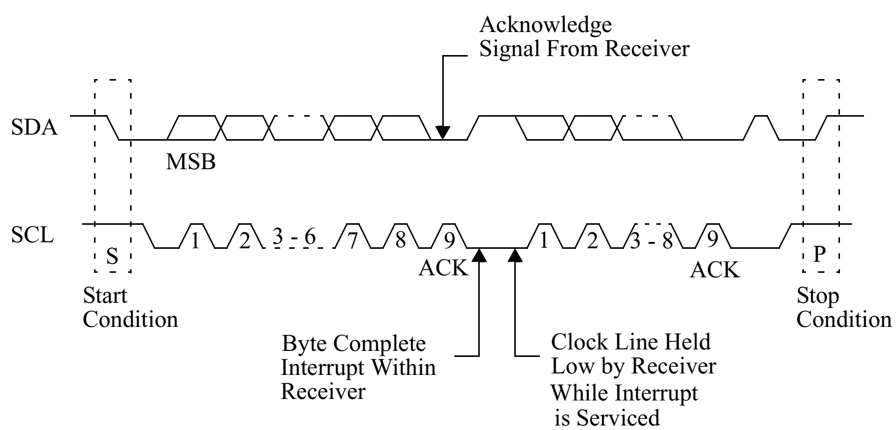


Figure 4-13 ACCESS.bus Data Transaction

The master generates the ACK clock pulse on the ninth clock pulse of the byte transfer. The transmitter releases the SDA line (permits it to go high) to allow the receiver to send the ACK signal. The receiver must pull down the SDA line during the ACK clock pulse, signalling that it has correctly received the last data byte and is ready to receive the next byte. Figure 4-14 illustrates the ACK cycle.

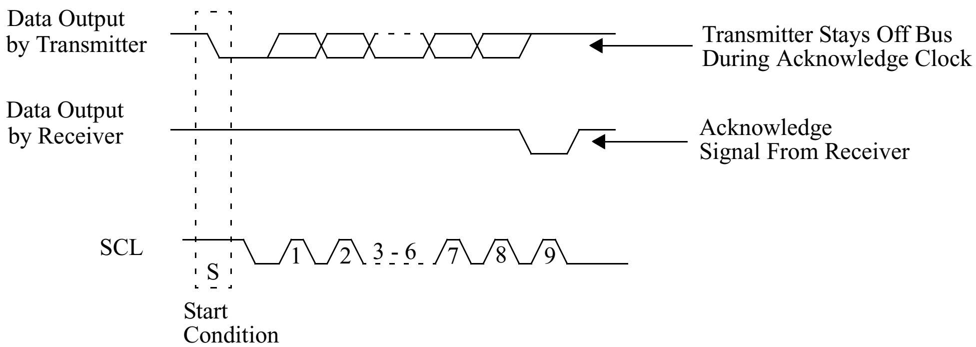


Figure 4-14 ACCESS.bus Acknowledge Cycle

4.12.1.4. Acknowledge after Every Byte Rule

According to this rule, the master generates an acknowledge clock pulse after each byte transfer, and the receiver sends an acknowledge signal after every byte received. There are two exceptions to this rule:

1. When the master is the receiver, it must indicate to the transmitter the end of data by not acknowledging (negative acknowledge) the last byte clocked out of the slave. This negative acknowledge still includes the acknowledge clock pulse (generated by the master), but the SDA line is not pulled down.
2. When the receiver is full, otherwise occupied, or a problem has occurred, it sends a negative acknowledge to indicate that it cannot accept additional data bytes.

4.12.1.5. Addressing Transfer Formats

Each device on the bus has a unique address. Before any data is transmitted, the master transmits the address of the slave being addressed. The slave device should send an acknowledge signal on the SDA line, once it recognizes its address.

The address consists of the first 7 bits after a Start Condition. The direction of the data transfer (R/W) depends on the bit sent after the address, the eighth bit. A low to high transition during a SCL high period indicates the Stop Condition, and ends the transaction of SDA (see Figure 4-15).

When the address is sent, each device in the system compares this address with its own. If there is a match, the device considers itself addressed and sends an acknowledge signal. Depending on the state of the R/W bit (1=read, 0=write), the device acts either as a transmitter or a receiver.

The I2C bus protocol allows a general call address to be sent to all slaves connected to the bus. The first byte sent specifies the general call address (00h) and the second byte specifies the meaning of the general call (for example, write slave address by software only). Those slaves that require data acknowledge the call, and become slave receivers; other slaves ignore the call.

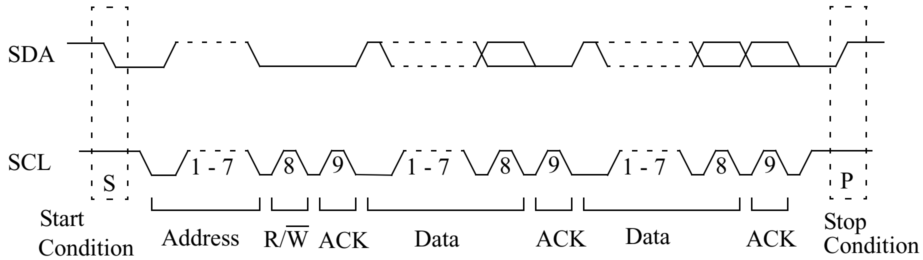


Figure 4-15 A Complete ACCESS.bus Data Transaction

4.12.1.6. Arbitration on the Bus

Multiple master devices on the bus require arbitration between their conflicting bus access demands. Bus control is initially determined according to address bits and clock cycle. If the masters are trying to address the same slave, data comparisons determine the outcome of this arbitration. In master mode, the device immediately aborts a transaction if the value sampled on the SDA line differs from the value driven by the device. (An exception to this rule is SDA while receiving data. The lines may be driven low by the slave without causing an abort.)

The SCL signal is monitored for clock synchronization and to allow the slave to stall the bus. The actual clock period is set by the master with the longest clock period, or by the slave stall period. The clock high period is determined by the master with the shortest clock high period.

When an abort occurs during the address transmission, a master that identifies the conflict should give up the bus, switch to slave mode and continue to sample SDA to check if it is being addressed by the winning master on the bus.

4.12.1.7. Master Mode

Requesting Bus Mastership

An ACCESS.bus transaction starts with a master device requesting bus mastership. It asserts a Start Condition, followed by the address of the device it wants to access. If this transaction is successfully completed, the software may assume that the device has become the bus master.

For the device to become the bus master, the software should perform the following steps:

1. Configure the INTEN bit of the ACBCTL1 Register to the desired operation mode (Polling or Interrupt) and set the START bit of this register. This causes the ACB to issue a Start Condition on the ACCESS.bus when the ACCESS.bus becomes free (BB bit of the ACBCST Register is cleared, or other conditions that can delay start). It then stalls the bus by holding SCL low.
2. If a bus conflict is detected (that is, another device pulls down the SCL signal), the BER bit of the ACBST Register set.
3. If there is no bus conflict, the MASTER bit of the ACBST Register and the SCAST of the ACBST Register sets.
4. If the INTEN bit of the ACBCTL1 Register is set and either the BER or SDAST bit of the ACBST Register sets, then an interrupt is issued.

Sending the Address Byte

When the device is the active master of the ACCESS.bus (the MASTER bit of the ACBST Register sets), it can send the address on the bus.

The address sent should not be the device's own address, as defined by the ADDR bit of the ACBADDR Register if the SAEN bit of this register is set, nor should it be the global call address if the GCMTCH bit of the ACBCST Register is set.

To send the address byte, use the following sequence:

1. For a receive transaction where the software wants only one byte of data, it should set the ACB bit of the ACBCTL1 Register. If only an address needs to be sent or if the device requires stall for some other reason, set the STASTRE bit of the ACBCTL1 Register.
2. Write the address byte (7-bit target device address) and the direction bit to the ACBSDA Register. This causes the ACB to generate a transaction. At the end of this transaction, the acknowledge bit received is copied to the NEGACK bit of the ACBST Register. During the transaction, the SDA and SCL lines are continuously checked for conflict with other devices. If a conflict is detected, the transaction is aborted, the BER bit of the ACBST Register is set and the MASTER bit of this register is cleared.
3. If the STASTRE bit of the ACBCTL1 Register is set and the transaction was successfully completed (that is, both the BER and NEGACK bits of the ACBST Register are cleared), the STASTR bit is set. In this case, the ACB stalls any further ACCESS.bus operations (that is, holds SCL low). If the INTEN bit of the ACBCTL1 Register is set, it also sends an interrupt request to the host.
4. If the requested direction is transmit and the start transaction was completed successfully (that is, neither the NEGACK nor the BER bit of the ACBST Register is set, and no other master has accessed the device), the SDAST bit of the ACBST Register is set to indicate that the ACB awaits attention.
5. If the requested direction is receive, the start transaction was completed successfully and the STASTRE bit of the ACBCTL1 Register is cleared, the ACB starts receiving the first byte automatically.
6. Check that both the BER and NEGACK bits of the ACBST Register are cleared. If the INTEN bit of the ACBCTL1 Register is set, an interrupt is generated when either the BER or NEGACK bit of the ACBST Register is set.

Master Transmit

After becoming the bus master, the device can start transmitting data on the ACCESS.bus.

To transmit a byte in an interrupt or polling controlled operation, the software should do the following:

1. Check that both the BER and NEGACK bits of the ACBST Register clears, and that the SDAST bit of the ACBST Register sets. If the STASTRE bit of the ACBCTL1 Register sets, also check that the STASTR bit of the ACBST Register clears (and clear it if required).
2. Write the data byte to be transmitted to the ACBSDA Register.

When either the NEGACK or BER bit of the ACBST Register sets, an interrupt is generated. When the slave responds with a negative acknowledge, the NEGACK bit of the ACBST Register sets and the SDAST bit of the ACBST Register remains cleared. In this case, if the INTEN bit of the ACBCTL1 Register sets, an interrupt issues.

Master Receive

After becoming the bus master, the device starts receiving data on the ACCESS.bus.

To receive a byte in an interrupt or polling operation, the software should:

1. Check that the SDAST bit of the ACBST

Register is set and that the BER bit is cleared. If the STASTRE bit of the ACBCTL1 Register is set, also check that the STASTRE bit of the ACBST Register is cleared (and clear it if required).
2. Set the ACK bit of the ACBCTL1 Register to 1, if the next byte is the last byte that should be read. This causes a negative acknowledge to be sent.
3. Read the data byte from the ACBSDA Register.

Before receiving the last byte of data, set the ACK bit of the ACBCTL1 Register.

Master Stop

To end a transaction, set the STOP bit of the ACBCTL1 Register before clearing the current stall flag (that is, the SDAST, NEGACK or STASTR bit of the ACBST Register). This causes the ACB to send a Stop Condition immediately, and to clear the STOP bit of the ACBCTL1 Register. A Stop Condition may be issued only when the device is the active bus master (the MASTER bit of the ACBST Register is set).

Master Bus Stall

The ACB can stall the ACCESS.bus between transfers while waiting for the host response. The ACCESS.bus is stalled by holding the SCL signal low after the acknowledge cycle. Note that this is interpreted as the beginning of the following bus operation. The user must make sure that the next operation is prepared before the flag that causes the bus stall is cleared.

The flags that can cause a bus stall in master mode are:

- Negative acknowledge after sending a byte (ACBST.NEGACK=1).
- ACBST.SDAST bit is set.
- ACBCTL1.STASTRE=1, after a successful start (ACBST.STASTR=1).

Repeated Start

A repeated start is performed when the device is already the bus master (ACBST.MASTER is set). In this case, the ACCESS.bus is stalled and the ACB awaits host handling due to: negative acknowledge (ACBST.NEGACK=1), empty buffer (ACBST.SDAST=1) and/or a stall after start (ACBST.STASTR=1).

For a repeated start:

1. Set (1) ACBCTL1.START.
2. In master receive mode, read the last data item from ACBSDA.
3. Follow the address send sequence, as described in ‘Sending the Address Byte’ on 321.
4. If the ACB was awaiting handling due to ACBST.STASTR=1, clear it only after writing the requested address and direction to ACBSDA.

Master Error Detection

The ACB detects illegal Start or Stop Conditions (that is, a Start or Stop Condition within the data transfer, or the acknowledge cycle) and a conflict on the data lines of the ACCESS.bus. If an illegal condition is detected, BER is set, and master mode is exited (ACBST.MASTER is cleared).

Bus Idle Error Recovery

When a request to become the active bus master or a restart operation fails, the ACBST.BER bit is set to indicate the error. In some cases, both the device and the other device may identify the failure and leave the bus idle. In this case, the start sequence may be incomplete and the ACCESS.bus may remain deadlocked.

To recover from deadlock, use the following sequence:

1. Clear ACBST.BER bit and the ACBCST.BB bit.
2. Wait for a time-out period to check that

there is no other active master on the bus (that is, ACBCST.BB remains cleared).

3. Disable, and re-enable the ACB to put it in the non-addressed slave mode. This completely resets the functional block.

At this point, some of the slaves may not identify the bus error. To recover, the ACB becomes the bus master: it asserts a Start Condition, sends an address byte, then asserts a Stop Condition which synchronizes all the slaves.

4.12.1.8. Slave Mode

A slave device waits in idle mode for a master to initiate a bus transaction. Whenever the ACB is enabled and it is not acting as a master (that is, ACBST.MASTER is cleared), it acts as a slave device.

Once a Start Condition on the bus is detected, the device checks whether the address sent by the current master matches either:

- The ACBADDR.ADDR value if ACBADDR.SAEN=1, or
- The general call address if ACBCTL1.GCMEN=1.

This match is checked even when ACBST.MASTER is set. If a bus conflict (on SDA or SCL) is detected, ACBST.BER is set, ACBST.MASTER is cleared and the device continues to search the received message for a match.

If an address match or a global match is detected:

1. The device asserts its SDA pin during the acknowledge cycle
2. The ACBCST.MATCH and ACBST.NMATCH bits are set. If ACBST.XMIT=1 (that is, slave transmit mode) ACBST.SDAST is set to indicate that the buffer is empty.
3. If ACBCTL1.INTEN is set, an interrupt is generated if both the ACBCTL1.INTEN and ACBCTL1.NMINTE bits are set.
4. The software then reads the ACBST.XMIT bit to identify the direction requested by the master device. It clears the ACBST.NMATCH bit so future byte transfers are identified as data bytes.

Slave Receive and Transmit

Perform slave receive and transmit after a match is detected and the data transfer direction is identified. After a byte transfer, the ACB extends the acknowledge clock until the software reads or writes the ACBSDA Register. The receive and transmit sequences are identical to those used in the master routine.

Slave Bus Stall

When operating as a slave, the device stalls the ACCESS.bus by extending the first clock cycle of a transaction in the following cases:

- ACBST.SDAST is set.
- ACBST.NMATCH and ACBCTL1.NMINTE are set.

Slave Error Detection

The ACB detects illegal Start and Stop Conditions on the ACCESS.bus (that is, a Start or Stop Condition within the data transfer or the acknowledge cycle). When this occurs, the BER bit is set and MATCH and GMATCH are cleared, setting the ACB as an unaddressed slave.

4.12.1.9. Configuration

SDA and SCL Signals

The SDA and SCL are open-drain signals. The device permits the user to define whether to enable or disable the internal pull-up of each of these signals.

ACB Clock Frequency

The ACB permits the user to set the clock frequency for the ACCESS.bus clock. The clock is set by the ACBCTL2.SCLFRQ field, which determines the SCL clock period used by the device. This clock low period may be extended by stall periods initiated by the ACB or by another ACCESS.bus device. In case of a conflict with another bus master, a shorter clock high period may be forced by the other bus master until the conflict is resolved.

4.12.2. ACB Registers

The register maps in this chapter use abbreviations for Type. See Table 4.55 "Register Type Abbreviations" on page 264

Table 4.139 ACB Register Map

Offset	Mnemonic	Register Name	Type	Reference
00h	ACBSDA	ACB Serial Data	R/W	Table 4.140
01h	ACBST	ACB Status	Varies per bit	Table 4.141
02h	ACBCST	ACB Control Status	Varies per bit	Table 4.142
03h	ACBCTL1	ACB Control 1	R/W	Table 4.143
04h	ACBADDR	ACB Own Address	R/W	Table 4.144
05h	ACBCTL2	ACB Control 2	R/W	Table 4.145

Table 4.140 ACB Serial Data Register (ACBSDA) - 00H

Bit	7	6	5	4	3	2	1	0
Name	**ACB Serial Data**							
Reset								
R/W: This shift register is used to transmit and receive data. The most significant bit is transmitted (received) first, and the least significant bit is transmitted last. Reading or writing to the ACBSDA Register is allowed only when the SDAST bit of the ACBST Register is set, or for repeated starts after setting the START bit. An attempt to access the register in other cases may produce unpredictable results.								

Table 4.141 ACB Status Register (ACBST) - 01H

Bit	7	6	5	4	3	2	1	0
Name	**SLVSTP**	**SDAST**	**BER**	**NEGACK**	**STASTR**	**NMATCH**	**MASTER**	**XMIT**
Reset	0	0	0	0	0	0	0	0

Type (R/W, etc.) varies per bit. This is a read register with a special clear. Some of its bits may be cleared by software, as described in the table below. This register maintains the current ACB status. On reset, and when the ACB is disabled, ACBST is cleared (00h).

Bit	Type	Description
7	R/W1C	**Slave Stop – SLVSTP** Writing 0 to SLVSTP is ignored. 0 = Writing 1 or ACB disabled 1 = Stop Condition detected after a slave transfer in which MATCH or GCMATCH was set
6	RO	**SDA Status – SDAST** 0 = Reading from the ACBSDA Register during a receive, or when writing to it during a transmit. When ACBCTL1.START is set, reading the ACBSDA Register does not clear SDAST. This enables ACB to send a repeated start in master receive mode. 1 = SDA Data Register awaiting data (transmit - master or slave) or holds data that should be read (receive - master or slave).
5	R/W1C	**Bus Error – BER** Writing 0 to BER is ignored. 0 = Writing 1 or ACB disabled 1 = Start or Stop Condition detected during data transfer (that is, Start or Stop Condition during the transfer of bits 2 through 8 and acknowledge cycle), or when an arbitration problem detected.
4	R/W1C	**Negative Acknowledge – NEGACK** Writing 0 to NEGACK is ignored. 0 = Writing 1 or ACB disabled 1 = Transmission not acknowledged on the ninth clock (In this case, SDAST is not set)
3	R/W1C	**Stall After Start – STASTR** Writing 0 to STASTR is ignored. 0 = Writing 1 or ACB disabled 1 = Address sent successfully (that is, a Start Condition sent without a bus error, or Negative Acknowledge), if ACBCTL1.STASTRE is set. This bit is ignored in slave mode. When STASTR is set, it stalls the ACCESS.bus by pulling down the SCL line, and suspends any further action on the bus (e.g., receive of first byte in master receive mode). In addition, if ACBCTL1.INTEN is set, it also causes the ACB to send an interrupt.
2	R/W1C	**New Match – NMATCH** Writing 0 to NMATCH is ignored. If ACBCTL1.INTEN is set, an interrupt is sent when this bit is set. 0 = Software writes 1 to this bit 1 = Address byte follows a Start Condition or a repeated start, causing a match or a global-call match.

Bit	Type	Description
1	RO	**Master** 0 = Arbitration loss (BER is set) or recognition of a Stop Condition 1 = Bus master request succeeded and master mode active
0	RO	**Transmit – XMIT** Direction bit. 0 = Master/slave transmit mode not active 1 = Master/slave transmit mode active

Table 4.142 ACB Control Status Register (ACBCST) - 02H

Bit	7	6	5	4	3	2	1	0
Name	**Reserved**		**TGSCL**	**TSDA**	**GCMTCH**	**MATCH**	**BB**	**BUSY**
Reset	0	0	0	X	0	0	0	0
This register configures and controls the ACB functional block. It maintains the current ACB status and controls several ACB functions. On reset and when the ACB is disabled, the non-reserved bits of ACBCST are cleared.								

Bit	Type	Description
7-6		**Reserved**
5	R/W	**Toggle SCL Line – TGSCL** Enables toggling the SCL line during error recovery. 0 = Clock toggle completed 1 = When the SDA line is low, writing 1 to this bit toggles the SCL line for one cycle. Writing 1 to TGSCL while SDA is high is ignored.
4	RO	**Test SDA Line – TSDA** This bit reads the current value of the SDA line. It can be used while recovering from an error condition in which the SDA line is constantly pulled low by an out-of-sync slave. Data written to this bit is ignored.
3	RO	**Global Call Match – GCMTCH** 0 = Start Condition or repeated Start and a Stop Condition (including illegal Start or Stop Condition) 1 = In slave mode, ACBCTL1.GCMEN is set and the address byte (the first byte transferred after a Start Condition) is 00h.
2	RO	**Address Match – MATCH** 0 = Start Condition or repeated Start and a Stop Condition (including illegal Start or Stop Condition) 1 = ACBADDR.SAEN is set and the first 7 bits of the address byte (the first byte transferred after a Start Condition) match the 7-bit address in the ACBADDR Register.
1	R/W1C	**Bus Busy – BB** 0 = Writing 1, ACB disabled, or Stop Condition detected 1 = Bus active (a low level on either SDA or SCL), or Start Condition

Bit	Type	Description
0	RO	**Busy** This bit should always be written 0. This bit indicates the period between detecting a Start Condition and completing receipt of the address byte. After this, the ACB is either free or enters slave mode. 0 = Completion of any state below or ACB disabled 1 = ACB is in one of the following states: Generating a Start Condition Master mode (ACBST.MASTER is set) Slave mode (ACBCST.MATCH or ACBCST.GCMTCH set).

Table 4.143 ACB Control Register 1 (ACBCTL1) - 03H

Bit	7	6	5	4	3	2	1	0
Name	**STASTRE**	**NMINTE**	**GCMEN**	**ACK**	**Reserved**	**INTEN**	**STOP**	**START**
Reset	0	0	0	0	0	0	0	0
R/W								

Bit	Description
7	**Stall After Start Enable – STASTRE** 0 = When cleared, ACBST.STASTR can not be set. However, if ACBST.STASTR is set, clearing STASTRE will not clear ACBST.STASTR. 1 = Stall after start mechanism enabled, and ACB stalls the bus after the address byte
6	**New Match Interrupt Enable – NMINTE** 0: = No interrupt issued on a new match 1 = Interrupt issued on a new match only if ACBCTL1.INTEN set
5	**Global Call Match Enable – GCMEN** 0 = ACB not responding to global call 1 = Global call match enabled
4	**Receive Acknowledge** This bit is ignored in transmit mode. When the device acts as a receiver (slave or master), this bit holds the stop transmitting instruction that is transmitted during the next acknowledge cycle. 0 = Cleared after acknowledge cycle 1 = Negative acknowledge issued on next received byte
3	**Reserved**
2	**Interrupt Enable** 0 = ACB interrupt disabled 1 = ACB interrupt enabled. An interrupt is generated in response to one of the following events: — Detection of an address match (ACBST.NMATCH=1) and NMINTE=1 — Receipt of Bus Error (ACBST.BER=1) — Receipt of Negative Acknowledge after sending a byte (ACBST.NEGACK=1) — Acknowledge of each transaction (same as the hardware set of the ACBST.SDAST bit) — In master mode if ACBCTL1.STASTRE=1, after a successful start (ACBST.STASTR=1) — Detection of a Stop Condition while in slave mode (ACBST.SLVSTP=1).

Bit	Description
1	**Stop** 0 = Automatically cleared after STOP issued 1 = Setting this bit in master mode generates a Stop Condition to complete or abort current message transfer
0	**Start** Set this bit only when in master mode or when requesting master mode. 0 = Cleared after Start Condition sent or Bus Error (ACBST.BER=1) detected 1 = Single or repeated Start Condition generated on the ACCESS.bus. If the device is not the active master of the bus (ACBST.MASTER=0), setting START generates a Start Condition when the ACCESS.bus becomes free (ACBCST.BB=0). An address transmission sequence should then be performed. If the device is the active master of the bus (ACBST.MASTER=1), setting START and then writing to the ACB-SDA Register generates a Start Condition. If a transmission is already in progress, a repeated Start Condition is generated. Use this condition to switch the direction of the data flow between the master and the slave, or to choose another slave device without separating them with a Stop Condition.

Table 4.144 ACB Own Address Register (ACBADDR) - 04H

Bit	7	6	5	4	3	2	1	0
Name	**SAEN**	**ADDR**						
Reset								
R/W: This is a byte-wide register that holds the ACB ACCESS.bus address. The reset value of this register is undefined.								

Bit	Description
7	**Slave Address Enable – SAEN** 0 = ACB does not check for an address match with ADDR field 1 = ADDR field holds a valid address and enables the match of ADDR to an incoming address byte
6-0	**Own Address** These bits hold the 7-bit device address. When in slave mode, the first 7 bits received after a Start Condition are compared with this field (first bit received is compared with bit 6, and the last bit with bit 0). If the address field matches the received data and ACBADDR.SAEN is 1, a match is declared.

Table 4.145 ACB Control Register 2 (ACBCTL2) - 05H

Bit	7	6	5	4	3	2	1	0
Name	**SCLFRQ**							**ENABLE**
Reset	0							0
R/W: This register enables/disables the functional block and determines the ACB clock rate.								

Bit	Description
7-1	**SCL Frequency – SCLFRQ** This field defines the SCL period (low and high time) when the device serves as a bus master. The clock low and high times are defined as follows: $t_{SCLl} = t_{SCLh} = 2 * SCLFRQ * t_{CLK}$ where t_{CLK} is the module input clock cycle,. Program SCLFRQ to values in the range of 0001000_2 (8_{10}) through 1111111_2 (127_{10}). Using any other value has unpredictable results.
0	**Enable** 0 = ACB disabled, ACBCTL1, ACBST and ACBCST cleared, and clocks halted 1 = ACB enabled

Table 4.146 ACB Register Bitmap

Register		Bits							
Offset	Mnemonic	7	6	5	4	3	2	1	0
00h	ACBSDA	**ACB Serial Data**							
01h	ACBST	**SLVSTP**	**SDAST**	**BER**	**NEGACK**	**STASTR**	**NMATCH**	**MASTER**	**XMIT**
02h	ACBCST	**Reserved**		**TGSCL**	**TSDA**	**GCMTCH**	**MATCH**	**BB**	**BUSY**
03h	ACBCTL1	**STASTRE**	**NMINTE**	**GCMEN**	**ACK**	**Reserved**	**INTEN**	**STOP**	**START**
04h	ACBADDR	**SAEN**	**ADDR**						
05h	ACBCTL2	**SCLFRQ**							**ENABLE**

4.13. Legacy Functional Blocks

This chapter briefly describes the following blocks that provide legacy device functions:

- Keyboard and Mouse Controller (KBC)
- Floppy Disk Controller (FDC)
- Parallel Port
- Serial Port 1 and Serial Port 2(SP1 and SP2), UART Functionality for both Serial Port 1 and Serial Port 2
- Infrared Communication Port Functionality

The description of each Legacy block includes the sections listed below.

- General Description
- Register Map table(s)
- Bitmap table(s)

The register maps in this chapter use abbreviations for Type. See Table 4.55 "Register Type Abbreviations" on page 264.

4.13.1. Keyboard and Mouse Controller (KBC)

The KBC is implemented physically as a single hardware module and houses two separate logical devices: a Mouse controller and a Keyboard controller.

The KBC is functionally equivalent to the industry standard 8042A Keyboard controller, which may serve as a detailed technical reference for the KBC.

Table 4.147 KBC Register Map

Offset	Mnemonic	Register Name	Type
00h	DBBOUT	Read KBC Data	R
	DBBIN	Write KBC Data	W
04h	STATUS	Read Status	R
	DBBIN	Write KBC Command	W

Table 4.148 KBC Bitmap Summary

Register		Bits							
Offset	**Mnemonic**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
00h	DBBOUT	**KBC Data Bits (For Read cycles)**							
	DBBIN	**KBC Data Bits (For Write cycles)**							
04h	STATUS	**General Purpose Flags**				**F1**	**F0**	**IBF**	**OBF**
	DBBIN	**KBC Command Bits (For Write cycles)**							

4.13.2. Floppy Disk Controller (FDC)

The generic FDC is a standard FDC with a digital data separator, and is DP8473 and N82077 software compatible.

The FDC is implemented in this device as follows:

- FM and MFM modes are supported. To select either mode, set bit 6 of the first command byte when writing to/reading from a diskette, where:
  0 = FM mode
  1 = MFM mode
- Automatic media sense is supported on the Parallel Port pins.
- A logic 1 is returned for all floating (TRI-STATE) FDC register bits upon LPC I/O read cycles.

Table 4.149 FDC Register Map

Offset	Mnemonic	Register Name	Type
00h	SRA	Status A	RO
01h	SRB	Status B	RO
02h	DOR	Digital Output	R/W
03h	TDR	Tape Drive	R/W
04h	MSR	Main Status	R
	DSR	Data Rate Select	W
05h	FIFO	Data (FIFO)	R/W
06h	**Reserved**		
07h	DIR	Digital Input	R
	CCR	Configuration Control	W

4.13.2.1. FDC Bitmap Summary

The FDC supports two system operation modes: PC-AT mode and PS/2 mode (Micro-Channel systems). Unless specifically indicated otherwise, all fields in all registers are valid in both drive modes.

Table 4.150 FDC Bitmap Summary

<table>
<tr><th colspan="2">Register</th><th colspan="8">Bits</th></tr>
<tr><td>Offset</td><td>Mnemonic</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td>00h</td><td>SRA[a]</td><td>IRQ Pending</td><td>Reserved</td><td>Step</td><td>$\overline{\text{TRK0}}$</td><td>Head Select</td><td>$\overline{\text{INDEX}}$</td><td>$\overline{\text{WP}}$</td><td>Head Direction</td></tr>
<tr><td>01h</td><td>SRB[a]</td><td colspan="2">Reserved</td><td>Drive Select 0 Status</td><td>$\overline{\text{WDATA}}$</td><td>$\overline{\text{RDATA}}$</td><td>$\overline{\text{WGATE}}$</td><td>Reserved</td><td>$\overline{\text{MTR0}}$</td></tr>
<tr><td>02h</td><td>DOR</td><td colspan="3">Reserved</td><td>Motor Enable 0</td><td>DMAEN</td><td>Reset Controller</td><td colspan="2">Drive Select</td></tr>
<tr><td rowspan="2">03h</td><td>TDR</td><td colspan="6">Reserved</td><td colspan="2">Tape Drive Select 1,0</td></tr>
<tr><td>TDR[b]</td><td colspan="2">Reserved</td><td colspan="2">Drive ID Information</td><td colspan="2">Logical Drive Exchange</td><td colspan="2">Tape Drive Select 1,0</td></tr>
<tr><td rowspan="2">04h</td><td>MSR</td><td>RQM</td><td>Data I/O Direction</td><td>Non-DMA Execution</td><td>Command in Progress</td><td>Drive 3 Busy</td><td>Drive 2 Busy</td><td>Drive 1 Busy</td><td>Drive 0 Busy</td></tr>
<tr><td>DSR</td><td>Software Reset</td><td>Low Power</td><td>Reserved</td><td colspan="3">Precompensation Delay Select</td><td colspan="2">Data Transfer Rate Select</td></tr>
<tr><td>05h</td><td>FIFO</td><td colspan="8">Data Bits</td></tr>
<tr><td rowspan="2">07h</td><td>DIR[c]</td><td>$\overline{\text{DSKCHG}}$</td><td colspan="7">Reserved</td></tr>
<tr><td>DIR[a]</td><td>$\overline{\text{DSKCHG}}$</td><td colspan="6">Reserved</td><td>High Density</td></tr>
<tr><td>07h</td><td>CCR</td><td colspan="8">Reserved</td></tr>
</table>

a. Applicable only in PS/2 Mode
b. Applicable only in Enhanced TDR Mode
c. Applicable only in PC-AT Compatible Mode

4.13.3. Parallel Port

The Parallel Port supports all IEEE1284 standard communication modes:

- Compatibility (known also as Standard or SPP)
- Bidirectional (known also as PS/2)
- FIFO, EPP (known also as Mode 4)
- ECP (with an optional Extended ECP mode)

4.13.3.1. Parallel Port Register Map

The Parallel Port functional block register maps are grouped according to first and second level offsets. EPP and second level offset registers are available only when base address is 8-byte aligned.

Table 4.151 Parallel Port Register Map for First Level Offset

First Level Offset	Mnemonic	Register Name	Modes (ECR Bits) 7 6 5	Type
000h	DATAR	PP Data	0 0 0 0 0 1	R/W
000h	AFIFO	ECP Address FIFO	0 1 1	W
001h	DSR	Status	All Modes	RO
002h	DCR	Control	All Modes	R/W
003h	ADDR	EPP Address	1 0 0	R/W
004h	DATA0	EPP Data Port 0	1 0 0	R/W
005h	DATA1	EPP Data Port 1	1 0 0	R/W
006h	DATA2	EPP Data Port 2	1 0 0	R/W
007h	DATA3	EPP Data Port 3	1 0 0	R/W
400h	CFIFO	PP Data FIFO	0 1 0	W
400h	DFIFO	ECP Data FIFO	0 1 1	R/W
400h	TFIFO	Test FIFO	1 1 0	R/W
400h	CNFGA	Configuration A	1 1 1	RO
401h	CNFGB	Configuration B	1 1 1	RO
402h	ECR	Extended Control	All Modes	R/W
403h	EIR	Extended Index	All Modes	R/W
404h	EDR	Extended Data	All Modes	R/W
405h	EAR	Extended Auxiliary Status	All Modes	R/W

Table 4.152 Parallel Port Register Map for Second Level Offset

Second Level Offset	Register Name	Type
00h	**Control0**	**R/W**
02h	**Control2**	**R/W**
04h	**Control4**	**R/W**
05h	**PP Confg0**	**R/W**

4.13.3.2. Parallel Port Bitmap Summary

The Parallel Port functional block bitmaps are grouped according to first and second level offsets.

Table 4.153 Parallel Port Bitmap Summary for First Level Offset

Register		Bits							
Offset	Mnemonic	7	6	5	4	3	2	1	0
000h	DATAR	Data Bits							
	AFIFO	Address Bits							
001h	DSR	Printer Status	$\overline{\text{ACK}}$ Status	PE Status	SLCT Status	$\overline{\text{ERR}}$ Status	Reserved		EPP Time-out Status
002h	DCR	Reserved		Direction Control	Interrupt Enable	PP Input Control	Printer Initialization Control	Automatic Line Feed Control	Data Strobe Control
003h	ADDR	EPP Device or Register Selection Address Bits							
004h	DATA0	EPP Device or R/W Data							
005h	DATA1	EPP Device or R/W Data							
006h	DATA2	EPP Device or R/W Data							
007h	DATA3	EPP Device or R/W Data							
400h	CFIFO	Data Bits							
400h	DFIFO	Data Bits							
400h	TFIFO	Data Bits							
400h	CNFGA	Reserved				Bit 7 of PP Confg0	Reserved		
401h	CNFGB	Reserved	Interrupt Request Value	Interrupt Select			Reserved	DMA Channel Select	
402h	ECR	ECP Mode Control			ECP Interrupt Mask	ECP DMA Enable	ECP Interrupt Service	FIFO Full	FIFO Empty
403h	EIR	Reserved					Second Level Offset		
404h	EDR	Data Bits							
405h	EAR	FIFO Tag	Reserved						

Table 4.154 Parallel Port Bitmap Summary for Second Level Offset

Register		Bits							
Second Level Offset	Mnemonic	7	6	5	4	3	2	1	0
00h	Control0	Reserved		DCR Register Live	Freeze Bit	Reserved			EPP Time-out Interrupt Mask
02h	Control2	SPP Compatibility	Channel Address Enable	Reserved	Revision 1.7 or 1.9 Select	Reserved			
04h	Control4	Reserved	PP DMA Request Inactive Time			Reserved	PP DMA Request Active Time		
05h	PP Confg0	Bit 3 of CNFGA	Demand DMA Enable	ECP IRQ Channel Number			PE Internal Pull-up or Pull-down	ECP DMA Channel Number	

4.13.4. UART Functionality (SP1/SP2)

Both Serial Port 1 and 2 provide UART functionality with remote peripheral devices or a modem using a wired interface. The UART functions as a standard 16450, 16550, or as an Extended UART.

4.13.4.1. UART Mode Register Bank Overview

Four register banks, each containing eight registers, control UART operation. All registers use the same 8-byte address space to indicate offsets 00h through 07h. The BSR register selects the active bank and is common to all banks. See Figure 4-16.

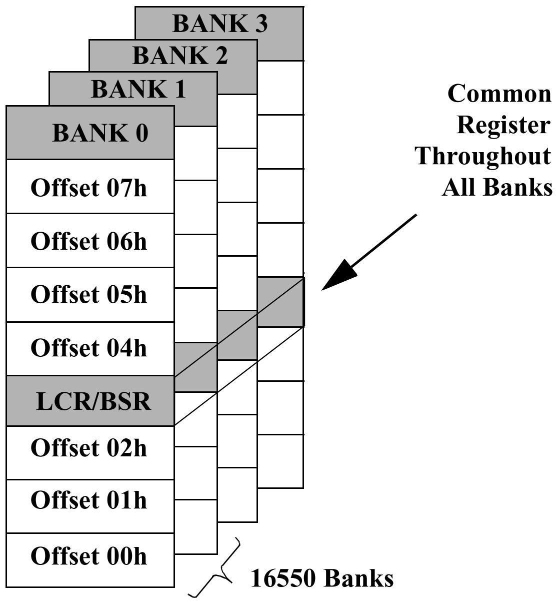


Figure 4-16 UART Mode Register Bank Architecture

4.13.4.2. SP1 and SP2 Register Maps for UART Functionality

Table 4.155 Bank 0 Register Map

Offset	Mnemonic	Register Name	Type
00h	RXD	Receiver Data Port	RO
00h	TXD	Transmitter Data Port	W
01h	IER	Interrupt Enable	R/W
02h	EIR	Event Identification (Read Cycles)	RO
	FCR	FIFO Control (Write Cycles)	W
03h	LCR[a]	Line Control	R/W
	BSR[a]	Bank Select	
04h	MCR	Modem/Mode Control	R/W
05h	LSR	Link Status	RO
06h	MSR	Modem Status	RO
07h	SPR/ASCR	Scratch pad/Auxiliary Status and Control	R/W

a. When bit 7 of this Register is set to 1, bits 6-0 of BSR select the bank, as shown in Table 4.156.

Table 4.156 Bank Selection Encoding

BSR Bits								Bank Selected
7	6	5	4	3	2	1	0	
0	x	x	x	x	x	x	x	0
1	0	x	x	x	x	x	x	1
1	1	x	x	x	x	1	x	1
1	1	x	x	x	x	x	1	1
1	1	1	0	0	0	0	0	2
1	1	1	0	0	1	0	0	3

Table 4.157 Bank 1 Register Map

Offset	Mnemonic	Register Name	Type
00h	LBGD(L)	Legacy Baud Generator Divisor Port (Low Byte)	R/W
01h	LBGD(H)	Legacy Baud Generator Divisor Port (High Byte)	R/W
02h	**Reserved**		
03h	LCR/BSR	Line Control/Bank Select	R/W
04h - 07h	**Reserved**		

Table 4.158 Bank 2 Register Map

Offset	Mnemonic	Register Name	Type
00h	BGD(L)	Baud Generator Divisor Port (Low Byte)	R/W
01h	BGD(H)	Baud Generator Divisor Port (High Byte)	R/W
02h	EXCR1	Extended Control1	R/W
03h	LCR/BSR	Line Control/Bank Select	R/W
04h	EXCR2	Extended Control 2	R/W
05h	**Reserved**		
06h	TXFLV	TX_FIFO Level	R/W
07h	RXFLV	RX_FIFO Level	R/W

Table 4.159 Bank 3 Register Map

Offset	Mnemonic	Register Name	Type
00h	MRID	Module Revision ID	RO
01h	SH_LCR	Shadow of LCR (Read Only)	RO
02h	SH_FCR	Shadow of FIFO Control (Read Only)	RO
03h	LCR/BSR	Line Control/Bank Select	R/W
04h-07h	**Reserved**		

4.13.4.3. SP1 and SP2 Bitmap Summary for UART Functionality

Table 4.160 Bank 0 Bitmap

Register		Bits							
Offset	Mnemonic	7	6	5	4	3	2	1	0
00h	RXD	Receiver Data Bits							
	TXD	Transmitter Data Bits							
01h	IER[a]	Reserved				MS_IE	LS_IE	TXLDL_IE	RXHDL_IE
	IER[b]	Reserved		TXEMP_IE	Reserved[c] or DMA_IE[d]	MS_IE	LS_IE	TXLDL_IE	RXHDL_IE
02h	EIR[a]	FEN1	FEN0	Reserved		RXFT	IPR1	IPR0	IPF
	EIR[b]	Reserved		TXEMP_EV	Reserved[c] or DMA_EV[d]	MS_EV	LS_EV or TXHLT_EV	TXLDL_EV	RXHDL_EV
	FCR	RXFTH1	RXFTH0	TXFTH1	TXFTH0	Reserved	TXSR	RXSR	FIFO_EN
03h	LCR[e]	BKSE	SBRK	STKP	EPS	PEN	STB	WLS1	WLS0
	BSR[e]	BKSE	Bank Select						
04h	MCR[a]	Reserved			LOOP	ISEN or DCDLP	RILP	RTS	DTR
	MCR[b]	Reserved				TX_DFR	Reserved	RTS	DTR
05h	LSR	ER_INF	TXEMP	TXRDY	BRK	FE	PE	OE	RXDA
06h	MSR	DCD	RI	DSR	CTS	DDCD	TERI	DDSR	DCTS
07h	SPR[a]	Scratch Data							
	ASCR[b]	Reserved	TXUR[d]	RXACT[d]	RXWDG[d]	Reserved	S_OET[d]	Reserved	RXF_TOUT

a. Non-Extended Mode
b. Extended Mode
c. In SP2 only.
d. In SP1 only
e. When bit 7 of this register is set to 1, bits 6-0 of BSR select the bank, as shown in Table 4.156.

Table 4.161 Bank 1 Bitmap

Register		Bits							
Offset	**Mnemonic**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
00h	LBGD(L)	**Legacy Baud Generator Divisor (Least Significant Bits)**							
01h	LBGD(H)	**Legacy Baud Generator Divisor (Most Significant Bits)**							
02h	**Reserved**								
03h	LCR/BSR	**Same as Bank 0**							
04h-07h	**Reserved**								

Table 4.162 Bank 2 Bitmap

Register		Bits							
Offset	**Mnemonic**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
00h	BGD(L)	**Baud Generator Divisor Low (Least Significant Bits)**							
01h	BGD(H)	**Baud Generator Divisor High (Most Significant Bits)**							
02h	EXCR1	**BTEST**	**Reserved**	**ETDLBK**	**LOOP**	**Reserved**			**EXT_SL**
03h	LCR/BSR	**Same as Bank 0**							
04h	EXCR2	**LOCK**	**Reserved**	**PRESL1**	**PRESL0**	**Reserved**			
05h	**Reserved**								
06h	TXFLV	**Reserved**			**TFL4**	**TFL3**	**TFL2**	**TFL1**	**TFL0**
07h	RXFLV	**Reserved**			**RFL4**	**RFL3**	**RFL2**	**RFL1**	**RFL0**

Table 4.163 Bank 3 Bitmap

Register		Bits							
Offset	**Mnemonic**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
00h	MRID	**Module ID (MID 7-4)**				**Revision ID(RID 3-0)**			
01h	SH_LCR	**BKSE**	**SBRK**	**STKP**	**EPS**	**PEN**	**STB**	**WLS1**	**WLS0**
02h	SH_FCR	**RXFTH1**	**RXFTH0**	**TXFHT1**	**TXFTH0**	**Reserved**	**TXSR**	**RXSR**	**FIFO_EN**
03h	LCR/BSR	**Same as Bank 0**							
04h-07h	**Reserved**								

4.13.5. IR Communication Port (IRCP) Functionality

This section describes the IR support registers. The IR functional block provides advanced, versatile serial communications features with IR capabilities.

The IRCP also supports two DMA channels; the functional block uses either one or both of them. One channel required for IR-based applications, since IR communication works in half duplex fashion. Two channels would normally be needed to handle high-speed full duplex IR based applications.

4.13.5.1. Functional Overview Of IR Modes

This section describes all operation modes. Although each mode is unique, certain system resources and features are common.

UART Mode

UART mode supports serial data communication with a remote peripheral device or modem using a wired interface. The functional block provides receive and transmit channels that operate concurrently in full-duplex mode. This functional block performs all functions required to conduct parallel data interchange with the system and composite serial data exchange with the external data channel.

Use this mode to support serial data communications with a remote peripheral module or modem using a wired interface. It provides transmit and receive channels that operate concurrently to handle full-duplex operation. They perform parallel-to-serial conversion on data characters received from the CPU or a DMA controller, and serial-to-parallel conversion on data characters received from the serial interface. The format of the serial data stream is shown in Figure 4-17. A data character contains 5 to 8 data bits, preceded by a start bit and followed by an optional parity bit and a stop bit. Data is transferred in Little Endian order (least significant bit first).

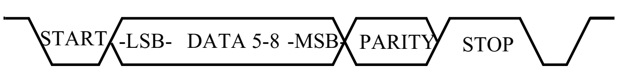


Figure 4-17 Composite Serial Data

Implement UART mode in standard 16450 and 16550 compatibility (Non-Extended) and Extended mode. UART 16450 compatibility mode is the default after power-up or reset. When you select Extended mode, the functional block architecture changes slightly and a variety of additional features are available. The interrupt sources are no longer prioritized, and an auxiliary status and control register replaces the scratch pad register. The additional features include: transmitter FIFO (TX_FIFO) thresholding, DMA capability, and interrupts on transmitter empty and DMA event.

The clock for both transmit and receive channels is provided by an internal baud generator that divides its input clock by any divisor value from 1 to 2^{16} -1. This output clock frequency must be programmed to be sixteen times the baud rate value. The baud generator input clock is derived from a 24 MHz clock through a programmable prescaler. The prescaler value is determined by the PRESL bits in the EXCR2 register. Its default value is 13. This allows all the standard baud rates, up to 115.2 Kbaud, to be obtained. Smaller prescaler values allow baud rates up to 921.6 Kbaud (standard) and 1.5 Mbaud (non-standard).

Before operation begins, both the communications format and baud rate must be programmed by the software. Program the communications format by loading a control byte into the LCR register, select the baud rate by loading an appropriate value into the baud generator divisor register. The software reads the status of the functional block at any time during operation. The status information includes FULL/EMPTY state for both transmit and receive channels, and any other condition detected on the received data stream, such as parity error, framing error, data overrun, or break event.

SHARP-IR Mode

This mode supports bidirectional data communication with a remote device using IR radiation as the transmission medium. Sharp-IR uses Digital Amplitude Shift Keying (DASK) and allows serial communication at baud rates up to 38.4 Kbaud. The format of the serial data is similar to the UART data format. Each data word is sent serially beginning with a 0 value start bit, followed by up to eight data bits (LSB first), an optional parity bit, and ending with at least one stop bit with a binary value of one. A logical 0 is signalled by sending a 500 KHz continuous pulse train of IR radiation. A logical 1 is signalled by the absence of any IR signal. This functional block performs the modulation and demodulation operations internally, or relies on the external optical module to perform them.

Sharp-IR device operation is similar to operation in UART mode; the main difference being that data transfer operations normally performed in half duplex fashion, and the modem control and status signals are not used. Selection of the Sharp-IR mode is controlled by the Mode Select (MDSL) bits in the MCR register when the functional block is in Extended mode, or by the IR_SL bits in the IRCR1 register when the functional block is in Non-Extended mode. This prevents legacy software, running in Non-Extended mode, from spuriously switching the functional block to UART mode when the software writes to the MCR register.

SIR Mode

SIR mode supports bidirectional data communication with a remote device using IR radiation as the transmit medium. SIR allows serial communication at baud rates up to 115.2 Kbaud. The serial data format is similar to the UART data format. Each data word is sent serially beginning with a 0 value start bit, followed by eight data bits (LSB first), an optional parity bit, and ending with at least one stop bit with a binary value of 1. A 0 value is signalled by sending a single IR pulse. A 1 value is signalled by not sending any pulse. The width of each pulse can be either 1.6 msec or 3/16 of the time required to transmit a single bit. (1.6 msec equals 3/16 of the time required to transmit a single bit at 115.2 Kbps.) This way, each word begins with a pulse for the start bit.

Operation in SIR is similar to the operation in UART mode, the main difference being that data transfer operations are normally performed in half duplex fashion. Selection of the IrDA 1.0 SIR mode is controlled by the MDSL bits in the MCR register when the UART is in Extended mode, or by the IR_SL bits in the IRCR1 register when the UART is in Non-Extended mode. This prevents legacy software, running in Non-Extended mode, from spuriously switching the functional block to UART mode when the software writes to the MCR register.

Consumer Electronic IR (CEIR) Mode

The CEIR circuitry is designed to optimally support all major protocols presently used in remote controlled home entertainment equipment: RC-5, RC-6, RECS 80, NEC and RCA. This module, in conjunction with an external optical device, provides the physical layer functions necessary to support these protocols. These functions include: modulation, demodulation, serialization, deserialization, data buffering, status reporting, interrupt generation, and so on. The software is responsible for the generation of the IR code to be transmitted, and for the interpretation of the received code.

CEIR Transmit Operation

The code to be transmitted consists of a sequence of bytes that represent either a bit string or a set of run-length codes. The number of bits or run-length codes usually needed to represent each IR code bit depends on the IR protocol to be used. The RC-5 protocol, for example, needs two bits or between one and two run-length codes to represent each IR code bit.

Transmission is initiated when the CPU or DMA module writes code bytes into the empty TX_FIFO. Transmission is normally completed when the CPU sets the S_EOT bit of the ASCR register (see ‘ASCR Register, Extended Mode’ on 366), before writing the last byte, or when the DMA controller activates the terminal count (TC). Transmission also terminates if the CPU simply stops transferring data and the transmitter becomes empty. However, in this case, a transmitter-underrun condition generates, which must be cleared in order to begin the next transmission.

The transmission bytes are either de-serialized or run-length encoded, and the resulting bit string modulates a carrier signal and is sent to the transmitter LED. The transfer rate of this bit string, like in UART mode, is determined by the value programmed in the Baud Generator Divisor registers. Unlike a UART transmission, start, stop and parity bits are not included in the transmitted data stream. A logic 1 in the bit string keeps the LED off, so no IR signal is transmitted. A logic 0 generates a sequence of modulating pulses which lights the transmitter LED. Frequency and pulse width of the modulating pulses are programmed by the MCFR and MCPW fields in the IRTXMC register, as well as the TXHSC bit of the RCCFG register.

The RC_MMD field selects the transmitter modulation mode. If the C_PLS mode is selected, modulating pulses generate continuously for the entire logic 0 bit time. If 6_PLS or 8_PLS mode is selected, six or eight pulses generate each time a logic 0 bit is transmitted following a logic 1 bit.

C_PLS modulation mode is used for RC-5, RC-6, NEC and RCA protocols. 8_PLS or 6_PLS modulation mode is used for the RECS 80 protocol. The 8_PLS or 6_PLS mode allows minimization of the number of bits needed to represent the RECS 80 IR code sequence. The current transmitter implementation supports only the modulated modes of the RECS 80 protocol. It does not support the Flash mode.

Note: The total transmission time for the logic 0 bits must be equal to or greater than 6 or 8 times the period of the modulation subcarrier, otherwise fewer pulses will be transmitted.

CEIR Receive Operation

The CEIR receiver is significantly different from a UART receiver in two ways. Firstly, the incoming IR signals are DASK modulated, therefore, demodulation may be necessary. Secondly, there are no start bits in the incoming data stream.

The operations performed by the receiver, whenever an IR signal is detected, are slightly different, depending on whether or not receiver demodulation is enabled. If demodulation is disabled, the receiver immediately becomes active. If demodulation is enabled, the receiver checks the carrier frequency of the incoming signal and becomes active only if the frequency is within the programmed range. Otherwise, the signal is ignored and no other action is taken.

When the receiver enters the active state, the RXACT bit of the ASCR register is set to 1. Once in the active state, the receiver keeps sampling the IR input signal and generates a bit string, where a logic 1 indicates an idle condition and a logic 0 indicates the presence of IR energy. The IR input is sampled regardless of the presence of IR pulses at a rate determined by the value loaded into the Baud Generator Divisor registers. The received bit string is either de-serialized and assembled into 8 bit characters, or it is converted to run-length encoded values. The resulting data bytes are then transferred into the receiver FIFO (RX_FIFO).

The receiver also sets the RXWDG bit of the ASCR register each time an IR pulse signal is detected. This bit is automatically cleared when the ASCR register is read. It is intended to assist the software in determining when the IR link has been idle for a certain time. The software can then stop data from being received by writing a 1 into the RXACT bit to clear it and return the receiver to the inactive state.

The frequency bandwidth for the incoming modulated IR signal is selected by the DFR and DBW fields in the IRRXDC register. There are two CEIR receive data modes: Oversampled and Programmed T Period. For either mode, the sampling rate is determined by the setting of the Baud Generator Divisor registers.

Oversampled mode can be used with the receiver demodulator either enabled or disabled. It should be used with the demodulator disabled when a detailed snapshot of the incoming signal is needed; for example, to determine the period of the carrier signal. If the demodulator is enabled, the stream of samples can be used to reconstruct the incoming bit string. To obtain good resolution, a fairly high sampling rate should be selected.

Programmed T Period mode should be used with the receiver demodulator enabled. The T Period represents one-half bit time for protocols using biphase encoding, or the basic unit of pulse distance for protocols using pulse distance encoding. The baud is usually programmed to match the T Period. For long periods of logic low or high, the receiver samples the demodulated signal at the programmed sampling rate.

Whenever a new IR energy pulse is detected, the receiver synchronizes the sampling process to the incoming signal timing. This reduces timing related errors and eliminates the possibility of missing short IR pulse sequences, especially with the RECS 80 protocol. In addition, the Programmed T Period sampling minimizes the amount of data used to represent the incoming IR signal, therefore reducing the processing overhead in the host CPU.

IrDA 1.1 MIR and FIR Modes

The functional block supports both IrDA 1.1 MIR and FIR modes, with data rates of 576 kbps, 1.152 Mbps and 4.0 Mbps. Details on the frame format, encoding schemes, CRC sequences, etc. are provided in the appropriate IrDA documents. The MIR transmitter front end section performs bit stuffing on the outbound data stream and places the Start and Stop flags at the beginning and end of MIR frames. The MIR receiver front end section removes flags and “de-stuffs” the inbound bit stream, and checks for abort conditions.

The FIR transmitter front end section adds the Preamble as well as Start and Stop flags to each frame, and encodes the transmit data into a 4 Pulse Position Modulation (PPM) data stream. The FIR receiver front end section strips the Preamble and flags from the inbound data stream and decodes the 4 PPM data while also checking for coding violations.

Both MIR and FIR front ends also automatically append CRC sequences to transmitted frames and check for CRC errors on received frames.

High Speed Infrared Transmit Operation

When the transmitter is empty, if either the CPU or the DMA controller writes data into the TX_FIFO, transmission of a frame begins. Frame transmission can be normally completed by using one of the following methods:

- **S_EOT bit (Set End of Transmission).** This method is used when data transfers are performed in Programmed Input/Output (PIO) mode. When the CPU sets the S_EOT bit before writing the last byte into the TX_FIFO, the byte will be tagged with an EOF indication. When this byte reaches the TX_FIFO bottom and is read by the transmitter front end, a CRC is appended to the transmitted DATA and the frame is normally terminated.
- **DMA TC Signal (DMA Terminal Count).** This method is used when data transfers are performed in DMA mode. It works similarly to the previous method except that the tagging of the last byte of a frame occurs when the DMA controller asserts the TC signal during the write of the last byte to the TX_FIFO.

Frame Length Counter. This method can be used when data transfers are performed in either PIO or DMA mode. The value of the FEND_MD bit of the IRCR2 register determines whether the Frame Length Counter is effective in the PIO or DMA mode. The counter is loaded from the Frame Length register (TFRL) at the beginning of each frame, and it is decremented as each byte is transmitted. An EOF is generated when the counter reaches 0. When used in DMA mode with an 8237 type DMA controller, this method allows a large data block to be automatically split into equal-size back-to-back frames, plus a shorter frame that is terminated by the DMA TC signal, if the block size is not an exact multiple of the frame size.

An option is also provided to stop transmission at the end of each frame. This happens when the transmitter Frame End stop mode is enabled (TX_MS bit of the IRCR2 register set to 1). By using this option, the software can send frames of different sizes without re-initializing the DMA controller for each frame. After transmission of each frame, the transmitter stops and generates an interrupt. The software loads the length of the next frame into the TFRL register and restarts the transmitter by clearing the TXHFE bit of the ASCR register.

Note: PIO or DMA mode is only controlled by setting the DMA_EN bit of the Extended mode MCR register. The functional block treats CPU and DMA access cycles the same except that DMA cycles always access the TX or RX_FIFO, regardless of the selected bank. When DMA_EN is set to 1, the CPU can still access the TX_FIFO and RX_FIFO. The CPU accesses will, however, be treated as DMA accesses as far as the function of the FEND_MD bit is concerned.

While a frame is being transmitted, data must be written to the TX_FIFO at a rate dictated by the transmission speed. If the CPU or DMA controller fails to meet this requirement, a transmitter underrun occurs, an inverted CRC is appended to the frame being transmitted, and the frame is terminated with a Stop flag. Data transmission then stops. Transmission of the inverted CRC guarantees that the remote receiving module receives the frame with a CRC error and discards it.

Following an underrun condition, data transmission always stops at the next frame boundary. The frame bytes from the point where the underrun occurred to the end of the frame are not sent out to the external infrared interface. Nonetheless, they are removed from the TX_FIFO by the transmitter and discarded. The underrun indication is reported only when the transmitter detects the end of frame via one of the methods described above. The software can do various things to recover from an underrun condition. For example, it can simply clear the underrun condition by writing a 1 into bit 6 of ASCR and retransmit the underrun frame later, or it can retransmit it immediately, before transmitting other frames.

If it chooses to retransmit the frame immediately, it needs to perform the following steps:

1. Disable DMA controller, if DMA mode was selected.
2. Read the TXFLV register to determine the number of bytes in the TX_FIFO. (This is needed to determine the exact point where the underrun occurred, and whether or not the first byte of a new frame is in the TX_FIFO.)
3. Reset TX_FIFO.
4. Backup DMA controller registers.
5. Clear Transmitter underrun bit.
6. Re-enable DMA controller.

High Speed Infrared Receive Operation

When the receiver front end detects an incoming frame, it starts de-serializing the infrared bit stream and loading the resulting data bytes into the RX_FIFO. When the EOF is detected, two or four CRC bytes are appended to the received data, and an EOF flag is written into the tag section of the RX_FIFO, along with the last byte. In the present implementation, the CRC bytes are always transferred to the RX_FIFO following the data. Additional status information, related to the received frame, is also written into the RX_FIFO tag section at this time. The status information is loaded into the LSR register when the last frame byte reaches the RX_FIFO bottom.

The receiver keeps track of the number of received bytes from the beginning of the current frame. It only transfers to the RX_FIFO a number of bytes not exceeding the maximum frame length value which is programmed via the RFRML register in bank Any additional frame bytes are discarded. When the maximum frame length value is exceeded, the MAX_LEN error flag is set.

Although data transfers from the RX_FIFO to memory can be performed either in PIO or DMA mode, DMA mode should be used due to the high data rates.

In order to handle back-to-back incoming frames, when DMA mode is selected and an 8237 type DMA controller is used, an 8-level Status FIFO (ST_FIFO) is provided. When an EOF is detected, in 8237 DMA mode, the status and byte count information for the frame is written into the ST_FIFO. An interrupt is generated when the ST_FIFO level reaches a programmed threshold or an ST_FIFO timeout occurs.

The CPU uses this information to locate the frame boundaries in the memory buffer where the data, belonging to the received frames, has been transferred by the 8237 type DMA controller.

During reception of multiple frames, if the RX_FIFO and/or the ST_FIFO fills up due to the DMA controller or CPU not serving them in time, one or more frames can be crushed and lost. This means that no bytes belonging to these frames were written to the RX_FIFO. In fact, a frame is lost in 8237 mode when the ST_FIFO is full for the entire time during which the frame is being received, even though there were empty locations in the RX_FIFO. This is because no data bytes can be loaded into the RX_FIFO and then transferred to memory by the DMA controller, unless there is at least one available entry in the ST_FIFO to store the number of received bytes. This information, as mentioned before, is needed by the software to locate the frame boundaries in the DMA memory buffer.

In the event that a number of frames is lost for any of the reasons mentioned above, one or more lost-frame indications including the number of lost frames, are loaded into the ST_FIFO.

Frames can also be lost in PIO mode, but only when the RX_FIFO is full, since in these cases, the ST_FIFO is only used to store lost-frame indications. It does not store frame status and byte count.

4.13.5.2. Special Features Description

FIFO Timeouts

Timeout mechanisms are provided to prevent received data from remaining in the RX_FIFO indefinitely in case the programmed interrupt or DMA thresholds are not reached.

An RX_FIFO timeout generates a Receiver Data Ready interrupt and/or a receiver DMA request if bit 0 of the IER register and/or bit 2 of the MCR register (in Extended mode) are set to 1 respectively. An RX_FIFO timeout also sets bit 0 of the ASCR register to 1 if the RX_FIFO is below the threshold. When a Receiver Data Ready interrupt occurs, this bit is tested by the software to determine whether a number of bytes indicated by the RX_FIFO threshold can be read without checking bit 0 of the LSR register. A Status FIFO (ST_FIFO) timeout is enabled only in MIR and FIR modes, and generates an interrupt if bit 6 of IER is set to 1.

The conditions that must exist for a timeout to occur in the various modes of operation are described below. When a timeout has occurred, it can only be reset when the FIFO is read by the CPU or DMA controller.

Timeout Conditions for MIR and FIR Modes

RX_FIFO Timeout Conditions:

1. At least one byte is in the RX_FIFO, and
2. More than 64 ms have elapsed since the last byte was loaded into the RX_FIFO from the receiver logic, and
3. More than 64 ms have elapsed since the last byte was read from the RX_FIFO by the CPU or DMA controller.

ST_FIFO Timeout Conditions:

1. At least one entry is in the ST_FIFO, and
2. More than 1 ms has elapsed since the last byte was loaded into the RX_FIFO by the receiver logic, and
3. More than 1 ms has elapsed since the last entry was read from the ST_FIFO by the CPU.

RX_FIFO Timeout Conditions for UART, SIR and Sharp-IR Modes

1. At least one byte is in the RX_FIFO, and
2. More than four character times have elapsed since the last byte was loaded into the RX_FIFO from the receiver logic, and
3. More than four character times have elapsed since the last byte was read from the RX_FIFO by the CPU or DMA controller.

Timeout Conditions for CEIR Mode

The RX_FIFO Timeout, in CEIR mode, is disabled while the receiver is active. The conditions for this timeout to occur are as follows:

1. At least one byte has been in the RX_FIFO for 64 ms or more, and
2. The receiver has been inactive (RXACT=0) for 64 ms or more, and
3. More than 64 ms have elapsed since the last byte was read from the RX_FIFO by the CPU or DMA controller.

Transmit Deferral

This feature allows software to send short high-speed data frames in PIO mode without the risk of generating a transmitter underrun. Even though this feature is available and works the same way in all modes, it is most likely to be used in MIR and FIR modes to support high-speed negotiations. This is because in other modes, either the transmit data rate is relatively low and thus the CPU can keep up with it without letting an underrun occur, as in the case of CEIR mode, or transmit underruns are allowed and are not considered to be error conditions.

Transmit deferral is available only in Extended mode and when the TX_FIFO is enabled. When transmit deferral is enabled (TX_DFR bit of the MCR register set to 1) and the transmitter becomes empty, an internal flag is set and locks the transmitter. If the CPU now writes data into the TX_FIFO, the transmitter does not start sending the data until the TX_FIFO level reaches 14, at which time the internal flag is cleared. The internal flag is also cleared and the transmitter starts transmitting when a timeout condition is reached. This prevents some bytes from being in the TX_FIFO indefinitely if the threshold is not reached.

The timeout mechanism is implemented by a timer that is enabled when the internal flag is set and there is at least one byte in the TX_FIFO. Whenever a byte is loaded into the TX_FIFO, the timer is reloaded with the initial value. If no byte is loaded for a 64-msec time, the timer times out and the internal flag is cleared, thus enabling the transmitter.

Automatic Fallback to 16550 Compatibility Mode

This feature is designed to support existing legacy software packages using the 16550 Serial Port. For proper operation, many of these software packages require that the module look identical to a plain 16550, since they access the Serial Port registers directly. Due to the fact that several extended features, as well as new operational modes are provided, make sure that the module is in the proper state before executing a legacy program.

The fallback mechanism eliminates the need to change the state when a legacy program is executed following completion of a program that used extended features. It automatically switches the module to 16550 compatibility mode and turns off any extended features, whenever the Baud Generator Divisor Register is accessed through the LBGD(L) or LBGD(H) ports in register bank 1.

In order to avoid spurious fallbacks, baud generator divisor ports are provided in bank 2. Baud generator divisor access through these ports changes the baud rate setting but does not cause fallback.

New programs designed to take advantage of the extended features should not use LBGD(L) and LBGD(H) to change the baud rate. Instead, they should use BGD(L) and BGD(H).

A fallback can occur from either Extended or Non-Extended modes. If Extended mode is selected, fallback is always enabled. In this case, when a fallback occurs, the following happens:

1. TX_FIFO and RX_FIFO switch to 16 levels.
2. A value of 13 is selected for the baud generator prescaler.
3. ETDLBK and BTEST of the EXCR1 register are cleared.
4. UART mode is selected.
5. The functional block switches to Non-Extended mode.

When fallback occurs from Non-Extended mode, only the first three of the above actions occur. No switching to UART mode occurs if either Sharp_IR or SIR infrared modes were selected. This prevents spurious switching to UART mode when a legacy program, running in Infrared mode, accesses the baud rate generator divisor register from bank 1.

Fallback from Non-Extended mode can be disabled by setting LOCK in the EXCR2 register to 1. When LOCK is set and the functional block is in Non-Extended mode, two scratchpad registers overlaid with LBGD(L) and LBGD(H) are enabled. Any attempted CPU access of the baud generator divisor register through LBGD(L) and LBGD(H) accesses the scratchpad registers, without affecting the baud rate setting. This feature allows existing legacy programs to run faster than 115.2 Kbaud, without realizing they are running at this speed.

Optical Transceiver Interface

This module implements a very flexible interface for the external infrared transceiver. Several signals are provided for this purpose. A transceiver module with one or two receive signals can be directly interfaced without any additional logic. Since various operational modes are supported, the transmitter power as well as the receiver filter in the transceiver module must be configured according to the selected mode.

Four special interface pins (ID/IRSL2-0 and ID3) are used to control the operational mode of the infrared transceiver. The logic levels of the ID/IRSL2-0 pins are directly controlled by the software (through the setting of bits 2-0 in the IRCFG1 register).

The ID/IRSL2-0 pins power up as inputs and can be driven by an external source. When in input mode, they can be used to read the identification data of infrared adapters. The ID3 pin is input only and is also used for this purpose. The ID0/IRSL0/IRRX2 pin can also function as an input to support an additional infrared receive signal. In this case, however, only one configuration pin will be available. The IRSL0_DS and IRSL1_DS bits in the IRCFG4 register determine the direction of the ID/IRSL2-0 pins.

4.13.5.3. IR Mode Register Bank Overview

Eight register banks, each containing eight registers, control UART operation. All registers use the same 8-byte address space to indicate offsets 00h through 07h. The active bank must be selected by the software.

The register bank organization enables access to the banks as required for activation of all module modes, while maintaining transparent compatibility with 16450 or 16550 software, which activates only the registers and specific bits used in those devices. For details, see “UART Mode” on page 341.

The Bank Selection register (BSR) selects the active bank and is common to all banks. See Figure 4-18. Therefore, each bank defines seven new registers.

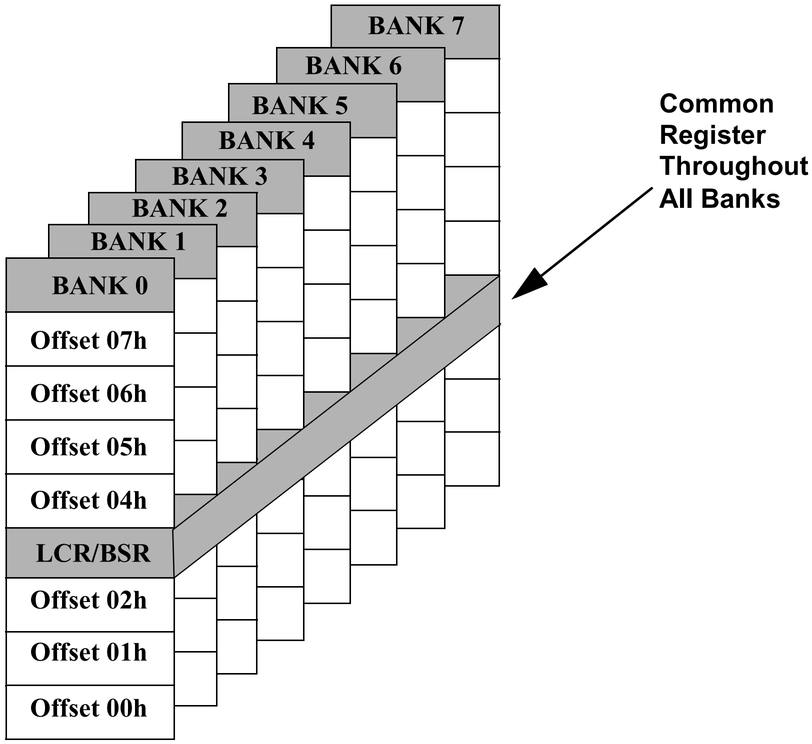


Figure 4-18 IRCP Register Bank Architecture

The default bank selection after system reset is 0, which places the module in UART 16550 mode. Additionally, setting the baud in bank 1 (as required to initialize the 16550 UART) switches the module to Non-Extended UART mode. This ensures that running existing 16550 software switches the system to the 16550 configuration without software modification.

Table 4.164 shows the main functions of the registers in each bank. Banks 0-3 control both UART and IR modes of operation; banks 4-7 control and configure the IR modes only.

Table 4.164 Register Bank Summary

Bank	UART	IR Mode	Main Functions	Reference
0	3	3	Global Control and Status	"BANK 0" on page 351
1	3	3	Legacy Bank	"BANK 1" on page 368
2	3	3	Alternative Baud Generator Divisor, Extended Control and Status	"BANK 2" on page 370
3	3	3	Module Revision ID and Shadow registers	"BANK 3" on page 375
4		3	IR mode setup	"BANK 4" on page 377
5		3	IR Control and Status FIFO	"BANK 5" on page 380
6		3	IR Physical Layer Configuration	"BANK 6" on page 383
7		3	CEIR and Optical Transceiver Configuration	"BANK 7" on page 388

The register maps in this section use the following abbreviations for Type:

- R/W = Read/Write
- R = Read from a specific address returns the value of a specific register. Write to the same address is to a different register.
- W = Write
- RO = Read Only
- R/W1C = Read/Write 1 to Clear. Writing 1 to a bit clears it to 0. Writing 0 has no effect.

4.13.5.4. IRCP Register Map

BANK 0

In Non-Extended modes of operation, bank 0 is compatible with both the 16450 and the 16550. Upon reset, this functional block defaults to the 16450 mode. In Extended mode, all the registers (except RXD/ TXD) offer additional features.

Table 4.165 Bank 0 Register Map

Offset	Mnemonic	Register Name	Type	Section
00h	RXD	Receiver Data Port	RO	"Receiver Data Port (RXD) or the Transmitter Data Port (TXD)" on page 351
	TXD	Transmitter Data Port	W	
01h	IER	Interrupt Enable	R/W	"Interrupt Enable Register (IER)" on page 351
02h	EIR	Event Identification	RO	"Event Identification Register (EIR) and FIFO Control Register (FCR)" on page 354
	FCR	FIFO Control	R/W	
03h	LCR	Link Control	W	"Link Control Register (LCR) and Bank Select Register (BSR)" on page 358
	BSR	Bank Select	R/W	
04h	MCR	Modem/Mode Control	R/W	"Modem/Mode Control Register (MCR)" on page 361
05h	LSR	Link Status	R/W	"Link Status Register (LSR)" on page 363
06h	MSR	Modem Status	R/W	"Modem Status Register (MSR)" on page 365
07h	SPR	Scratchpad	R/W	"SPR Register, Non-Extended Mode" on page 366
	ASCR	Auxiliary Status and Control	Varies per bit	"ASCR Register, Extended Mode" on page 366

Receiver Data Port (RXD) or the Transmitter Data Port (TXD)

These ports share the same address.

RXD is accessed during CPU read cycles. It is used to read data from the Receiver Holding register when the FIFOs are disabled, or from the bottom of the RX_FIFO when the FIFOs are enabled.

TXD is accessed during CPU write cycles. It is used to write data to the Transmitter Holding register when the FIFOs are disabled, or to the TX_FIFO when the FIFOs are enabled.

DMA cycles always access the TXD and RXD ports, regardless of the selected bank.

Interrupt Enable Register (IER)

This register controls the enabling of various interrupts. Some interrupts are common to all operating modes of the functional block, while others are mode specific. Bits 4 to 7 can be set in Extended mode only. They are cleared in Non-Extended mode. When a bit is set to 1, an interrupt is generated when the corresponding event occurs. In Non-Extended mode, most events can be identified by reading the LSR and MSR registers. The receiver high-data-level event can only be identified by reading the EIR register after the corresponding interrupt has been generated. In Extended mode, events are identified by event flags in the EIR register.

The bitmap of this register is defined differently, depending on the operating mode of the functional block.

The different modes can be divided into the following groups:

- UART, Sharp-IR and SIR in Non-Extended mode (EXT_SL of the EXCR1 register is set to 0)
- UART, Sharp-IR, SIR and CEIR in Extended mode (EXT_SL of the EXCR1 register is set to 1)
- MIR and FIR modes

The following sections describe the bits in this register for each of these modes.

Note 1: If the interrupt signal drives an edge-sensitive interrupt controller input, it is advisable to disable all interrupts by clearing all the IER bits upon entering the interrupt routine, and re-enable them just before exiting it. This guarantees proper interrupt triggering in the interrupt controller in case one or more interrupt events occurs during execution of the interrupt routine.

Note 2: If an interrupt source must be disabled, the CPU can do so by clearing the corresponding bit of the IER register. However, if an interrupt event occurs just before the corresponding enable bit of the IER register is cleared, a spurious interrupt may be generated. To avoid this problem, clearing any IER bit should be done during execution of the interrupt service routine. If the interrupt controller is programmed for level-sensitive interrupts, clearing IER bits can also be performed outside the interrupt service routine, but with the CPU interrupt disabled.

Note 3: If the LSR, MSR or EIR registers are to be polled, the interrupt sources which are identified via self-clearing bits should have their corresponding IER bits set to 0. This prevents spurious pulses on the interrupt output pin.

IER, Non-Extended Mode (UART, SIR or Sharp-IR in Non-Extended Mode)

Upon reset, the IER supports UART, SIR and Sharp-IR in Non-Extended modes.

Location: Offset 01h

Type: R/W

Table 4.166 Interrupt Enable Register (IER, Non-Extended Mode)

Bit	7	6	5	4	3	2	1	0
Mnemonic	**Reserved**				**MS_IE**	**LS_IE**	**TXLDL_IE**	**RXHDL_IE**
Reset	0	0	0	0	0	0	0	0

Bit	Description
7-4	**Reserved**
3	**Modem Status Interrupt Enable – MS_IE** Setting this bit to 1 enables the interrupts on Modem Status events. (EIR bits 3-0 are 0000. See Table 4.167.)
2	**Link Status Interrupt Enable – LS_IE** Setting this bit to 1 enables interrupts on Link Status events. (EIR bits 3-0 are 0110. See Table 4.167.)
1	**Transmitter Low Data Level Interrupt Enable – TXLDL_IE** Setting this bit to 1 enables interrupts on Transmitter Low Data Level events (EIR bits 3-0 are 0010. See Table 4.167.)
0	**Receiver High Data Level Interrupt Enable – RXHDL_IE** Setting this bit to 1 enables interrupts on Receiver High Data Level, or RX_FIFO Timeout events (EIR bits 3-0 are 0100 or 1100. See Table 4.167.)

IER, Extended Mode (UART, SIR, Sharp-IR and CEIR in Extended Mode)

Location: Offset 01h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	**TMR_IE**	**SFIF_IE**	**TXEMP_IE/ PLD_IE**	**DMA_IE**	**MS_IE**	**LS_IE/ TXHLT_IE**	**TXLDL_IE**	**RXHDL_IE**
Reset	0	0	0	0	0	0	0	0

Bit	Description
7	**Timer Interrupt Enable– TMR_IE**
6	**SFIF_IE (ST_FIFO Interrupt Enable)**
5	**TXEMP_IE/PLD_IE (Transmitter Empty Interrupt Enable/Pipeline Load Interrupt Enable).** Setting this bit to 1 enables Transmitter Empty interrupts (in all modes) and Pipeline Load Interrupts (in SIR, MIR and FIR).
4	**DMA_IE (DMA Interrupt Enable).** Setting this bit to 1 enables the interrupt on terminal count when the DMA is enabled.

Bit	Description
3	**Modem Status Interrupt Enable – MS_IE** Setting this bit to 1 enables the interrupts on Modem Status events.
2	**Link Status Interrupt Enable/Transmitter Halted Interrupt Enable – LS_IE/TXHLT_IE** Setting this bit enables Link Status Interrupts, TX_FIFO underrun interrupts in MIR and FIR and Transmitter Halted interrupts in CEIR.
1	**Transmitter Low-Data-Level Interrupt Enable – TXLDL_IE** Setting this bit to 1 enables interrupts when the TX_FIFO is below the threshold level or the Transmitter Holding register is empty.
0	**Receiver High-Data-Level Interrupt Enable – RXHDL_IE** Setting this bit to 1 enables interrupts when the RX_FIFO is equal to or above the RX_FIFO threshold level, or an RX_FIFO timeout occurs.

Event Identification Register (EIR) and FIFO Control Register (FCR)

The EIR register, a read only register, shares the same address as the FCR register, which is a write only register. The EIR indicates the interrupt source, and operates in two modes: Non-Extended Mode (EXT_SL of the EXCR1 register is set to 0), and Extended (EXT_SL of the EXCR1 register is set to 1).

When in Non-Extended mode (default), this register functions the same as in the 16550 mode.

EIR, Non-Extended Mode

In Non-Extended UART mode, the functional block prioritizes interrupts into four levels. The EIR indicates the highest level of interrupt that is pending. See Table 4.167 for the encoding of these interrupts

Location: Offset 02h
Type: RO

Bit	7	6	5	4	3	2	1	0
Mnemonic	FEN1, 0		Reserved		RXFT	IPR1, 0		IPF
Reset	0	0	0	0	0	0	0	1

Bit	Description
7-6	**FEN1, 0 (FIFOs Enabled)** 0 = No FIFO enabled (default) 1 = FIFOs enabled (bit 0 of FCR is set to 1)
5-4	**Reserved** – Set to 0.
3	**RX_FIFO Timeout) – RXFT** In the 16450 mode, this bit is always 0. In the 16550 mode (FIFOs enabled), this bit is set to 1 when an RX_FIFO read timeout occurred and the associated interrupt is currently the highest priority pending interrupt.
2-1	**Interrupt Priority – IPR1, 0** When bit 0 (IPF) is 0, these bits indicate the pending interrupt with the highest priority. See Table 4.167. Default value is 0.

Bit	Description
0	**Interrupt Pending Flag – IPF** 0 = Interrupt pending 1 = No interrupt pending (default)

Table 4.167 Non-Extended Mode Interrupt Priorities

EIR Bits 3 2 1 0	Priority Level	Interrupt Type	Interrupt Source	Interrupt Reset Control
0 0 0 1	N/A	None	None	N/A
0 1 1 0	Highest	Link Status	Parity error, framing error, data overrun or break event	Read Link Status Register (LSR)
0 1 0 0	Second	Receiver High Data Level Event	Receiver Holding Register (RXD) full, or RX_FIFO level equal to or above threshold	Reading the RXD or RX_FIFO level drops below threshold
1 1 0 0	Second	RX_FIFO Timeout	At least one character is in the RX_FIFO, and no character has been input to or read from the RX_FIFO for 4 character times.	Reading the RXD port.
0 0 1 0	Third	Transmitter Low Data Level Event	Transmitter Holding register or TX_FIFO empty	Reading the EIR register if this interrupt is currently the highest priority pending interrupt, or writing into the TXD port
0 0 0 0	Fourth	Modem Status	Any transition on $\overline{CTS}$, $\overline{DSR}$ or $\overline{DCD}$ or a low to high transition on $\overline{RI}$	Reading the Modem Status register (MSR)

EIR, Extended Mode

In Extended mode, each of the previously prioritized and encoded interrupt sources is broken down into individual bits. Each bit in this register acts as an interrupt pending flag, and is set to 1 when the corresponding event occurred or is pending, regardless of the IER register bit setting. When this register is read the DMA event (bit 4) is cleared if an 8237 type DMA is used. All other bits are cleared when the corresponding interrupts are acknowledged by reading the relevant register (e.g. reading the MSR register clears MS_EV).

Location: Offset 02h
Type: RO

Bit	7	6	5	4	3	2	1	0
Mnemonic	TMR_EV	SFIF_EV	TXEMP_EV	DMA_EV	MS_EV	LS_EV/ TXHLT_EV	TXLDL_EV	RXHDL_EV
Reset	0	0	0	0	0	0	0	1

Bit	Description
7	**Timer Event – TMR_EV** Set to 1 when the timer reaches 0. Cleared by writing 1 into bit 7 of the ASCR register.

Bit	Description
6	**ST_FIFO Event – SFIF_EV** Set to 1 when the ST_FIFO level is equal to or above the threshold, or an ST_FIFO timeout occurs. This bit is cleared when the CPU reads the ST_FIFO and its level drops below the threshold.
5	**Transmitter Empty Interrupt Enable – TXEMP_EV** This bit is the same as bit 6 of the LSR register. It is set to 1 when the transmitter is empty.
4	**DMA Event – DMA_EV** This bit is set to 1 when a DMA terminal count (TC) is activated. It is cleared upon read.
3	**Modem Status Event – MS_EV** UART mode: This bit is set to 1 when any of the 0 to 3 bits in the MSR register is set to 1. Any IR mode: The function of this bit depends on the setting of IRMSSL of the IRCR2 register (see ‘IR Control Register 2 (IRCR2)’ on 380). When IRMSSL is 0, the bit functions as Modern Status Interrupt event; when IRMSSL is set to 1, the bit is forced to 0.
2	**Link Status Event – LS_EV (** In UART, Sharp-IR and SIR: Set to 1 when a receiver error or break condition is reported. When FIFOs are enabled, the Parity Error), Frame Error and Break conditions are only reported when the associated character reaches the bottom of the RX_FIFO. An Overrun Error is reported as soon as it occurs. LS_EV/TXHLT_EV (Link Status Event or Transmitter Halted Event) - In MIR and FIR: Set to 1 when any of the following conditions occur: 1. Last byte of received frame reaches the bottom of the RX_FIFO 2. Receiver overrun 3. Transmitter underrun 4. Transmitter halted on frame end in CEIR mode **Link Status Event or Transmitter Halted Event – LS_EV/TXHLT_EV** In CEIR: Set to 1 when the receiver is overrun or the transmitter underrun. **Note:** A high speed CPU can service the interrupt generated by the last frame byte reaching the RX_FIFO bottom before that byte is transferred to memory by the DMA controller. This can happen when the CPU interrupt latency is shorter than the RX_FIFO Timeout (Refer to the “FIFO Timeouts” section). A DMA request is generated only when the RX_FIFO level reaches the DMA threshold or when a FIFO timeout occurs, in order to minimize the performance degradation due to DMA signal handshake sequences. If the DMA controller must be set up before receiving each frame, the software in the interrupt routine should make sure that the last byte of the frame just received has been transferred to memory before re-initializing the DMA controller, otherwise that byte could appear as the first byte of the next received frame.
1	**Transmitter Low-Data-Level Event – TXLDL_EV** FIFOs disabled: Set to 1 when the Transmitter Holding register is empty. FIFOs enabled: Set to 1 when the TX_FIFO level is below the threshold level.
0	**Receiver High-Data-Level Event – RXHDL_EV** FIFOs disabled: Set to 1 when a character is in the Receiver Holding register. FIFOs enabled: Set to 1 when the RX_FIFO is above threshold or an RX_FIFO timeout has occurred.

FIFO Control Register (FCR)

The FCR register is a write only register which shares the same address as the EIR (see "Event Identification Register (EIR) and FIFO Control Register (FCR)" on page 354) which is a read only register. FCR is used to enable the FIFOs, clear the FIFOs and set the interrupt thresholds levels for the RX_FIFO and TX_FIFO. FCR may be read through SH_FCR register in bank 3 (see 'Shadow of FIFO Control Register (SH_FCR)' on 376).

Location: Offset 02h
Type: W

Bit	7	6	5	4	3	2	1	0
Mnemonic	RXFTH1, 0		TXFTH1, 0		Reserved	TXSR	RXSR	FIFO_EN
Reset	0	0	0	0	0	0	0	0

Bit	Description
7-6	**RX_FIFO Interrupt Threshold Level – RXFTH1, 0** These bits select the RX_FIFO interrupt threshold level. An interrupt is generated when the level of the data in the RX_FIFO is equal to or above the encoded threshold. **Bits RX_FIF0 Interrupt Threshold Level** **7 6 (16-Level FIFO) (32-Level FIFO)** 0 0 1(Default) 1(Default) 0 1 4 8 1 0 8 16 1 1 14 26
5-4	**Transmitter Empty – TXFTH1, 0** In Non-Extended modes, these bits have no effect. In Extended modes, these bits select the TX_FIFO interrupt threshold level. An interrupt is generated when the level of the data in the TX_FIFO drops below the encoded threshold. **Bits TX_FIF0 Interrupt Threshold Level** **5 4 (16-Level FIFO) (32-Level FIFO)** 0 0 1(Default) 1(Default) 0 1 3 7 1 0 9 17 1 1 13 25
3	**Reserved.** Write 0.
2	**Transmitter Soft Reset – TXSR** Writing a 1 to this bit generates a transmitter soft reset, which clears the TX_FIFO and the transmitter logic. This bit is automatically cleared by the hardware.
1	**Receiver Soft Reset – RXSR** Writing a 1 to this bit generates a receiver soft reset, which clears the RX_FIFO and the receiver logic. This bit is automatically cleared by the hardware.
0	**FIFO Enable – FIFO_EN** When set to 1, enables both the TX_FIFO and RX_FIFOs. Resetting this bit clears both FIFOs. In MIR, FIR and CEIR modes, the FIFOs are always enabled and the setting of this bit is ignored.

Link Control Register (LCR) and Bank Select Register (BSR)

These registers share the same address.

The Link Control register (LCR) selects the communications format for data transfers in UART, SIR and Sharp-IR modes.

The Bank select register (BSR) is used to select the register bank to be accessed next.

Reading the register at this address location returns the content of the BSR. The content of LCR may be read from the Shadow of Link Control register (SH_LCR) in bank 3 (see 'Shadow of Link Control Register (SH_LCR)' on 376). During a write operation to this register at this address location, setting of Bank Select Enable (BKSE, bit 7) determines the register to be accessed, as follows:

- If bit 7 is 0, both LCR and BSR are written into.
- If bit 7 is 1, only BSR is written into, and LCR remains unchanged. This prevents the communications format from being spuriously affected when a bank other than bank 0 or bank 1 is accessed.

LCR Register

Bits 6-0 are only effective in UART, Sharp-IR and SIR modes. They are ignored in CEIR, MIR and FIR modes.

Location: Offset 03h
Type: W

Bit	7	6	5	4	3	2	1	0
Mnemonic	**BKSE**	**SBRK**	**STKP**	**EPS**	**PEN**	**STB**	**WLS1, 0**	
Reset	0	0	0	0	0	0	0	0

Bit	Description
7	**Bank Select Enable – BKSE** 0: Register functions as the Link Control register (LCR). 1: Register functions as the Bank Select register (BSR).
6	**Set Break – SBRK** Enables or disables a break. During the break, the transmitter can be used as a character timer to accurately establish the break duration. This bit acts only on the transmitter front end and has no effect on the rest of the transmitter logic.When set to 1 the following occurs: If a UART mode is selected, the SOUT pin is forced to a logic 0 state. If SIR mode is selected, pulses are issued continuously on the IRTX pin. If Sharp-IR mode is selected and internal modulation is enabled, pulses are issued continuously on the IRTX pin. If Sharp-IR mode is selected and internal modulation is disabled, the IRTX pin is forced to a logic 1 state. To avoid transmission of erroneous characters as a result of the break, use the following procedure to set SBRK: • Wait for the transmitter to be empty. (TXEMP = 1). • Set SBRK to 1. • Wait for the transmitter to be empty, and clear SBRK when normal transmission must be restored.
5	**Stick Parity – STKP** When parity is enabled (PEN is 1), this bit and EPS (bit 4) control the parity bit as shown in Table 4.168

Bit	Description
4	**Even Parity Select – EPS** When parity is enabled (PEN is 1), this bit and STKP (bit 5) control the parity bit, as shown in Table 4.168. 0: If parity enabled, an odd number of logic 1's is transmitted or checked in the data word bits and parity bit. (default) 1: If parity enabled, an even number of logic 1's is transmitted or checked
3	**Parity Enable – PEN** This bit enables the parity bit. See Table 4.168. The parity enable bit is used to produce an even or odd number of 1's when the data bits and parity bit are summed, as an error detection device. 0: No parity bit used (default) 1: A parity bit is generated by the transmitter and checked by the receiver
2	**Stop Bits – STB** This bit specifies the number of stop bits transmitted with each serial character. 0: 1 stop bit generated (default) 1: If the data length is set to 5 bits via bits 1,0 (WLS1,0), 1.5 stop bits are generated. For 6, 7 or 8 bit word lengths, 2 stop bits are transmitted. The receiver checks for 1 stop bit only, regardless of the number of stop bits selected.
1-0	**Character Length Select – WLS1, 0** These bits specify the number of data bits in each transmitted or received serial character and are encoded as follows: **Bits** **1 0 Character Length** 0 0 5 (Default) 0 1 6 1 0 7 1 1 8

Table 4.168 Bit Settings for Parity Control

PEN	EPS	STKP	Selected Parity Bit
0	x	x	None
1	0	0	Odd
1	1	0	Even
1	0	1	Logic 1
1	1	1	Logic 0

BSR Register, All Banks

The BSR register selects which register bank is to be accessed next. For details on how to access this register, see the description of BKSE (bit 7) of the LCR register.

Location: Offset 03h
Type: R/W

Bits	7	6	5	4	3	2	1	0
Mnemonic	**BKSE**	**BSR6-0**						
Reset	0	0	0	0	0	0	0	0

Bit	Description
7	**Bank Select Enable – BKSE** 0 = Bank 0 selected 1 = Bits 6-0 specify the selected bank
6-0	**Bank Select** When BKSE (bit 7) is set to 1, these bits select the bank, as shown in Table 4.169.

Table 4.169 Bank Selection Encoding

BSR Bits								Bank Selected
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**	
0	x	x	x	x	x	x	x	0
1	0	x	x	x	x	x	x	1
1	1	x	x	x	x	1	x	1
1	1	x	x	x	x	x	1	1
1	1	1	0	0	0	0	0	2
1	1	1	0	0	1	0	0	3
1	1	1	0	1	0	0	0	4
1	1	1	0	1	1	0	0	5
1	1	1	1	0	0	0	0	6
1	1	1	1	0	1	0	0	7
1	1	1	1	1	x	0	0	Reserved
1	1	0	x	x	x	0	0	Reserved

Modem/Mode Control Register (MCR)

This register controls the interface with the modem or data communications set, and the device operational mode when the device is in the Extended mode. The register function differs for Extended and Non-Extended modes.

MCR, Non-Extended Mode

Location: Offset 04h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	**Reserved**			**LOOP**	**ISEN or DCDLP**	**RILP**	**RTS**	**DTR**
Reset	0	0	0	0	0	0	0	0
Required	0	0	0					

Bit	Description
7-5	**Reserved**
4	**Loopback Enable – LOOP** This bit accesses the same internal register as LOOP (bit 4) of the EXCR1 register (See 'Extended Control Register 1 (EXCR1)' on 370 for more information on Loopback mode). 0 = Loopback disabled (default) 1 = Loopback enabled
3	**Interrupt Signal Enable or DCD Loopback – ISEN or DCDLP** In normal operation (standard 16450 or 16550) mode, this bit controls the interrupt signal and must be set to 1 in order to enable the interrupt request signal. When loopback is enabled, the interrupt output signal is always enabled, and this bit internally drives $\overline{\text{DCD}}$. New programs should always keep this bit set to 1 during normal operation. The interrupt signal should be controlled through the Plug and Play logic.
2	**Request Interrupt in Loopback – RILP** When loopback is enabled, this bit internally drives $\overline{\text{RI}}$. Otherwise, it is unused.
1	**Request To Send – RTS** This bit controls the $\overline{\text{RTS}}$ signal output. When set to 1, $\overline{\text{RTS}}$ is driven low. When loopback is enabled, this bit drives $\overline{\text{CTS}}$ internally.
0	**Data Terminal Ready – DTR** Controls the $\overline{\text{DTR}}$ signal output. When set to 1, $\overline{\text{DTR}}$ is driven low. When loopback is enabled, this bit internally drives $\overline{\text{DSR}}$.

MCR, Extended Mode

In Extended mode, this register is used to select the operation mode (IrDA, Sharp, etc.) of the device and to enable the DMA interface. In these modes, the interrupt output signal is always enabled, and loopback can be enabled by setting bit 4 of the EXCR1 register.

Location: Offset 04h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	MDSL2-0			IR_PLS	TX_DFR	DMA_EN	RTS	DTR
Reset	0	0	0	0	0	0	0	0

Bit	Description
7-5	**Mode Select – MDSL2-0** These bits select the operation mode of the functional block when in Extended mode. When the mode is changed, the TX_FIFO and RX_FIFOs are flushed, Link Status and Modem Status Interrupts are cleared, and all of the bits in the auxiliary status and control register are cleared. **Bits** **7 6 5 Operation Mode** 0 0 0 UART (Default) 0 0 1 Reserved 0 1 0 Sharp-IR 0 1 1 SIR 1 0 0 MIR 1 0 1 FIR 1 1 0 CEIR 1 1 1 Reserved
4	**Infrared Interaction Pulse – IR_PLS** This bit is effective only in MIR and FIR modes. It is set to 1 by writing 1 into it. Writing 0 into it has no effect. When set to 1, a 2 ms infrared interaction pulse is transmitted at the end of the frame and the bit is automatically cleared by the hardware.This bit is also cleared when the transmitter is soft reset. The interaction pulse must be emitted at least once every 500 ms, as long as the high-speed connection lasts, in order to quiet slower (115.2 kbps or below) systems that might otherwise interfere with the link.
3	**Transmit Deferral – TX_DFR** For a detailed description of the transmit deferral see ‘Transmit Deferral’ on 347. This bit is effective only if the TX_FIFOs is enabled. 0: No transmit deferral enabled (default) 1: Transmit deferral enabled.
2	**DMA Enable – DMA_EN** When set to1, DMA mode of operation is enabled. When DMA is selected, transmit and/or receive interrupts should be disabled to avoid spurious interrupts. DMA cycles always address the Data Holding registers or FIFOs, regardless of the selected bank. 0: DMA mode disabled (default) 1: DMA mode enabled
1	**Request To Send – RTS** This bit controls the $\overline{RTS}$ signal output. When set to1, $\overline{RTS}$ is driven low. When loopback is enabled, this bit internally drives both CTS and DCD
0	**Data Terminal Ready – DTR** This bit controls the $\overline{DTR}$ signal output. When set to 1, $\overline{DTR}$ is driven low. When loopback is enabled, this bit internally drives both DSR and RI.

Link Status Register (LSR)

This register provides status information concerning data transfer.Upon reset, this register assumes the value of 0x60h. The bit definitions change depending upon the operation mode of the functional block.

Bits 1 through 4 of the LSR indicate link status events. These bits are sticky (accumulate any conditions occurred since the last time the register was read). They are cleared when one of the following events occur:

- Hardware reset
- Receiver soft reset
- LSR register read

The LSR is intended for read operations only. Writing to the LSR is not permitted.

Location: Offset 05h
Type: RO

Bit	7	6	5	4	3	2	1	0
Mnemonic	ER_INF/ FR_END	TXEMP	TXRDY	BRK/ MAX_LEN	FE/ PHY_ERR	PE/ BAD_CRC	OE	RXDA
Reset	0	1	1	0	0	0	0	0

Bit	Description
7	**Error in RX_FIFO – ER_INF/ FR_END** In UART, Sharp-IR and SIR modes, this bit is set to 1 if there is at least one framing error, parity error or break indication in the RX_FIFO. This bit is always 0 in 16450 mode. It is cleared upon read or upon reset, if there is no faulted byte in RX_FIFO. **Frame End – FR_END** In MIR and FIR modes, set to 1 when the last byte (Frame End Byte) of a received frame reaches the bottom of the RX_FIFO. Cleared upon read.
6	**Transmitter Empty – TXEMP** This bit is set to 1 when the Transmitter Holding register or the TX_FIFO is empty, and the transmitter front end is idle.
5	**Transmitter Ready – TXRDY** This bit is set to 1 when the Transmitter Holding register or the TX_FIFO is empty. It is cleared when a data character is written to the TXD register.
4	**Break Event Detected – BRK** In UART, Sharp-IR and SIR modes, this bit is set to 1 when a break event is detected (that is, when a sequence of logic 0 bits, equal or longer than a full character transmission, is received). If the FIFOs are enabled, the break condition is associated with the particular character in the RX_FIFO to which it applies. In this case, the BRK bit is set when the character reaches the bottom of the RX_FIFO. When a break event occurs, only one 0 character is transferred to the Receiver Holding register or to the RX_FIFO. The next character transfer takes place after at least one logic 1 bit is received followed by a valid start bit. This bit is cleared upon read. **Maximum Length – MAX_LEN** In MIR and FIR modes, set to 1 when a frame exceeding the maximum length is received, and the last byte of the frame has reached the bottom of the RX_FIFO. Cleared upon read.

Bit	Description
3	**Framing Error – FE** In UART, Sharp-IR and SIR modes, this bit is set to 1 when the received data character does not have a valid stop bit (that is, the stop bit following the last data bit or parity bit is a 0). If the FIFOs are enabled, this Framing Error is associated with the particular character in the FIFO that it applies to. This error is revealed to the CPU when its associated character is at the bottom of the RX_FIFO. After a framing error is detected, the receiver will try to resynchronize. If the bit following the erroneous stop bit is 0, the receiver assumes it to be a valid start bit and shifts in the new character. If that bit is a 1, the receiver enters the idle state and awaits the next start bit. This bit is cleared upon read. **Physical Layer Error – PHY_ERR** In MIR and FIR modes, set to 1 when an abort condition is detected during the reception of a frame, and the last byte of the frame has reached the bottom of the RX_FIFO. Cleared upon read.
2	**Parity Error – PE** In UART, Sharp-IR and SIR modes, this bit is set to 1 if the received data character does not have the correct parity, even or odd as selected by the parity control bits of the LCR register. If the FIFOs are enabled, this error is associated with the particular character in the FIFO that it applies to. This error is revealed to the CPU when its associated character is at the bottom of the RX_FIFO. This bit is cleared upon read. **CRC Error – BAD_CRC** In MIR and FIR modes, set to 1 when a mismatch between the received CRC and the receiver-generated CRC is detected, and the last byte of the received frame has reached the bottom of the RX_FIFO. Cleared upon read.
1	**Overrun Error – OE** In UART, Sharp-IR and SIR modes, set to 1 as soon as an overrun condition is detected by the receiver. Cleared upon read. **FIFOs Disabled:** An overrun occurs when a new character is completely received into the receiver front end section and the CPU has not yet read the previous character in the receiver holding register. The new character is discarded, and the receiver holding register is not affected. **FIFOs Enabled:** An overrun occurs when a new character is completely received into the receiver front end section and the RX_FIFO is full. The new character is discarded, and the RX_FIFO is not affected. **Overrun Error – OE** In MIR and FIR modes, an overrun occurs when a new character is completely received into the receiver front end section and the RX_FIFO or the ST_FIFO is full. The new character is discarded, and the RX_FIFO is not affected. Cleared upon read.
0	**Receiver Data Available – RXDA** Set to 1 when the Receiver Holding register is full. If the FIFOs are enabled, this bit is set when at least one character is in the RX_FIFO. It is cleared when the CPU reads all the data in the Holding register or in the RX_FIFO.

Modem Status Register (MSR)

The function of this register depends on the selected operational mode. When a UART mode is selected, this register provides the current-state as well as state-change information of the status lines from the modem or data transmission module.

When any of the IR modes is selected, the register function is controlled by the setting of the IRMSSL bit of the IRCR2 (‘IR Control Register 2 (IRCR2)’ on 380). If IRMSSL is 0, the MSR register works as in UART mode. If IRMSSL is 1, the MSR register returns the value 30 hex, regardless of the state of the modem input lines.

When loopback is enabled, the MSR register works similarly except that its status input signals are internally driven by appropriate bits in the MCR register since the modem input lines are internally disconnected. Refer to bits 3-0 in MCR (‘Modem/Mode Control Register (MCR)’ on 361) and to the LOOP & ETDLBK bits at the EXCR1 (‘Extended Control Register 1 (EXCR1)’ on 370) for more information.

A Modem Status Event (MS_EV) is generated if the MS_IE bit in IER is enabled and any of the bits 0, 1, 2 or 3 in this register is set to 1.

Bits 0 to 3 are set to 0 as a result of any of the following events:

- A hardware reset occurs.
- The operational mode is changed and the IRMSSL bit is 0.
- The MSR register is read.

In the reset state, bits 4 through 7 are indeterminate as they reflect their corresponding input signals.

Note: The modem status lines can be used as general purpose inputs. They have no effect on the transmitter or receiver operation

Location: Offset 06h

Type: RO

Bit	7	6	5	4	3	2	1	0
Mnemonic	**DCD**	**RI**	**DSR**	**CTS**	**DDCD**	**TERI**	**DDSR**	**DCTS**
Reset	X	X	X	X	0	0	0	0

Bit	Description
7	**Data Carrier Detect – DCD** This bit returns the inverse of the $\overline{\text{DCD}}$ input signal.
6	**Ring Indicate – RI** This bit returns the inverse of the $\overline{\text{RI}}$ input signal.
5	**Data Set Ready – DSR** This bit returns the inverse of the $\overline{\text{DSR}}$ input signal.
4	**Clear To Send – CTS** This bit returns the inverse of the $\overline{\text{CTS}}$ input signal.
3	**Delta Data Carrier Detect – DDCD** Set to 1, when the $\overline{\text{DCD}}$ input signal changes state. 1: $\overline{\text{DCD}}$ signal state changed.
2	**Trailing Edge Ring Indicate – TERI** Set to 1, when the $\overline{\text{RI}}$ input signal changes state from low to high. This bit is cleared upon read.
1	**Delta Data Set Ready – DDSR** Set to 1, when the $\overline{\text{DSR}}$ input signal changes state. This bit is cleared upon read.

Bit	Description
0	**Delta Clear to Send – DCTS** Set to 1, when the $\overline{CTS}$ input signal changes state. This bit is cleared upon read.

SPR Register, Non-Extended Mode

The SPR shares the same address as ASCR (see ‘ASCR Register, Extended Mode’ on 366).

In Non-Extended mode, this is a Scratchpad register (as in the 16550) for temporary data storage.

ASCR Register, Extended Mode

The ASCR shares the same address as SPR (see ‘SPR Register, Non-Extended Mode’ on 366).

This register is accessed when the Extended mode of operation is selected. The definition of the bits in this case is dependent upon the mode selected in the MCR register, bits 7 through 5. This register is cleared upon hardware reset Bit 2 is cleared also when the transmitter is “soft reset” or after S-EOT byte is transmitted. Bit 6 is cleared also when the transmitter is “soft reset” or by writing 1 into it. Bits 0,1,4 and 5 are cleared also when the receiver is “soft reset”.

Location: Offset 07h
Type: Varies per bit

Bit	7	6	5	4	3	2	1	0
Mnemonic	CTE	TXUR	RXACT/ RXBSY	RXWDG/ LOST_FR	TXHFE	S_OET	FEND_INF	RXF_TOUT
Reset	0	0	0	0	0	0	0	0

Bit	Description
7	**Clear Timer Event – CTE** In SIR, MIR and FIR modes, writing 1 to this bit position clears the TMR_EV bit of the EIR register. Writing 0 has no effect.
6	**IR Transmitter Underrun – TXUR** In MIR, FIR and CEIR, this is the Transmitter Underrun flag. This bit is set to 1 when a transmitter underrun occurs. It is always cleared when a mode other than CEIR is selected. This bit must be cleared, by writing 1 into it, to re-enable transmission.
5	**Receiver Active – RXACT** In CEIR, set to 1 when an IR pulse or pulse-train is received. If a 1 is written into this bit position, the bit is cleared and the receiver is deactivated. When this bit is set, the receiver samples the IR input continuously at the programmed baud and transfers the data to the RX_FIFO. **Receiver Busy – RXBSY** In MIR and FIR, this bit is read only, and returns a 1 when reception of a frame is in progress.
4	**Reception WATCHDOG – RXWDG** In CEIR, set to 1 each time a pulse or pulse-train (modulated pulse) is detected by the receiver. It can be used by the software to detect a receiver idle condition. Cleared upon read. **Lost Frame Flag – LOST_FR** In MIR and FIR, this bit is read only, and reflects the setting of the lost-frame indicator flag at the bottom of the ST_FIFO.

Bit	Description
3	**Transmitter Halted on Frame End – TXHFE** In MIR and FIR, this bit is used only when the transmitter frame-end stop mode is selected (TX_MS bit in IRCR2 set to 1). It is set to 1 by the hardware when transmission of a frame is complete and the end-of-frame condition was generated by the TFRCC counter reaching 0. This bit must be cleared, by writing 1 into it, to re-enable transmission.
2	**Set End of Transmission – S_OET** In CEIR, when a 1 is written into this bit position before writing the last character into the TX_FIFO, data transmission is properly completed. In this mode, if the CPU simply stops writing data into the TX_FIFO at the end of the data stream, a transmitter underrun is generated and the transmitter stops. In this case this is not an error, but the software must clear the underrun before the next transmission can occur. This bit is automatically cleared by hardware when a character is written to the TX_FIFO. **Set End of Frame – S_OET** In MIR and FIR, when a 1 is written into this bit position before writing the last character into the TX_FIFO, frame transmission is completed and a CRC + EOF is sent. This bit can be used as an alternative to the Transmitter Frame Length register. If this method is to be used, the FEND_MD bit of the IRCR2 register should be set to 1, or the Transmitter Frame Length register should be set to maximum count.
1	**Transmitter Halted on Frame End – FEND_INF** In MIR and FIR, this bit is read only, and is set to 1 when one or more Frame End bytes are in the RX_FIFO. Cleared when no Frame End byte is in the RX_FIFO.
0	**RX_FIFO Timeout – RXF_TOUT** This bit is read only and set to 1 when an RX_FIFO timeout occurs. It is cleared when a character is read from the RX_FIFO.

Table 4.170 Bank 0 Bitmap

Register		Bits							
Offset	Mnemonic	7	6	5	4	3	2	1	0
00h	RXD	RXD7	RXD6	RXD5	RXD4	RXD3	RXD2	RXD1	RXD0
00h	TXD	TXD7	TXD6	TXD5	TXD4	TXD3	TXD2	TXD1	TXD0
01h	IER[a]	Reserved				MS_IE	LS_IE	TXLDL_IE	RXHDL_IE
	IER[b]	TMR_IE	SFIF_IE	TXEMP_IE/ PLD_IE	DMA_IE	MS_IE	LS_IE	TXLDL_IE	RXHDL_IE
02h	EIR[a]	FEN1, 0		Reserved		RXFT	IPR1, 0		IPF
	EIR[b]	TMR_EV	SFIF_EV	TXEMP_EV/ PLD_EV	DMA_EV	MS_EV	LS_EV/ TXHLT_EV	TXLDL_EV	RXHDL_IE
	FCR	RXFTH1, 0		TXFTH1, 0		Reserved	TXSR	RXSR	FIFO_EN
03h	LCR	BKSE	SBRK	STKP	EPS	PEN	STB	WLS1	WLS0
	BSR	BKSE	BSR6-0						
04h	MCR[a]	Reserved			LOOP	ISEN or DCDLP	RILP	RTS	DTR
	MCR[b]	MDSL2-0			IR_PLS	TX_DFR	DMA_EN	RTS	DTR
05h	LSR	ER_INF/ FR_END	TXEMP	TXRDY	BRK/ MAX_LEN	FE/ PHY_ERR	PE/ BAD_CRC	OE	RXDA
06h	MSR	DCD	RI	DSR	CTS	DDCD	TERI	DDSR	DCTS
07h	SPR[a]	Scratch Data							
	ASCR[b]	CTE/PLD	TXUR	RXACT/ RXBSY	RXWDG/ LOST_FR	TXHFE	S_OET	FEND_INF	RXF_TOUT

a. Non-Extended mode
b. Extended mode

BANK 1

Table 4.171 Bank 1 Register Map

Offset	Mnemonic	Register Name	Type	Reference
00h	LBGD(L)	Legacy Baud Generator Divisor Port (Low Byte)	R/W	page 368
01h	LBGD(H)	Legacy Baud Generator Divisor Port (High Byte)	R/W	
02h	Reserved			
03h	LCR/BSR	Link Control/Bank Select	R/W	
04h - 07h	Reserved			

Legacy Baud Generator Divisor Ports LSB (LBGD(L)) and MSB (LBGD(H))

The Legacy Baud Generator Divisor (LBGD) port provides an alternate data path to the Baud Divisor Generator register. This port is implemented to maintain compatibility with 16550 standard and to support existing legacy software packages. In case of using legacy software, the addresses 0 and 1 are shared with the data ports RXD/TXD (see ‘Receiver Data Port (RXD) or the Transmitter Data Port (TXD)’ on 351). The selection between them is controlled by the value of the BKSE bit (see ‘Link Control Register (LCR) and Bank Select Register (BSR)’ on 358, LCR bit 7). See ‘Automatic Fallback to 16550 Compatibility Mode’ on 348 for more information regarding the fallback mechanism.

The programmable baud rates in the Non-Extended mode are achieved by dividing a 24 MHz clock by a prescale value of 13, 1.625 or 1. This prescale value is selected by the PRESL field of EXCR2 (‘Extended Control and Status Register 2 (EXCR2)’ on 373). This clock is subdivided by the two Baud Generator Divisor buffers, which output a clock at 16 times the desired baud (this clock is the BOUT clock). This clock is used by I/O circuitry, and after a last division by 16 produces the output baud.

Divisor values between 1 and 2^{16}-1 can be used (zero is forbidden). The Baud Generator Divisor must be loaded during initialization to ensure proper operation of the Baud Generator. Upon loading either part of it, the Baud Generator counter is immediately loaded. Table 4.173 shows typical baud divisors. After reset the divisor register contents are indeterminate.

Any access to the LBGD(L) or LBGD(H) ports causes a reset to the default Non-Extended mode, that is, 16550 mode (‘Automatic Fallback to 16550 Compatibility Mode’ on 348). To access a Baud Generator Divisor when in Extended mode, use the port pair in bank 2 (see ‘Baud Generator Divisor Ports, LSB (BGD(L)) and MSB (BGD(H))’ on 370).

Table 4.172 shows the bits which are cleared when fallback occurs during Extended or Non-Extended modes. If the UART is in Non-Extended mode and the LOCK bit is 1, the content of the divisor (BGD) ports will not be affected and no other action is taken. When programming the baud, the new divisor is loaded upon writing into LBGD(L) and LBGD(H). After reset, the contents of these registers are indeterminate. Divisor values between 1 and 2^{16}-1 can be used (zero is forbidden). Table 4.173 shows typical baud divisors.

Table 4.172 Bits Cleared on Fallback

Register Mnemonic	UART Mode and LOCK Bit before Fallback		
	Extended Mode LOCK = x	**Non-Extended Mode LOCK = 0**	**Non-Extended Mode LOCK = 1**
MCR	2 to 7	None	None
EXCR1	0, 5 and 7	5 and 7	None
EXCR2	0 to 5	0 to 5	None

Table 4.173 Baud Generator Divisor Settings

Prescaler Value	13		1.625		1	
Baud	**Divisor**	**% Error**	**Divisor**	**% Error**	**Divisor**	**% Error**
50	2304	0.16%	18461	0.00%	30000	0.00%
75	1536	0.16%	12307	0.01%	20000	0.00%
110	1047	0.19%	8391	0.01%	13636	0.00%
134.5	857	0.10%	6863	0.00%	11150	0.02%
150	768	0.16%	6153	0.01%	10000	0.00%
300	384	0.16%	3076	0.03%	5000	0.00%
600	192	0.16%	1538	0.03%	2500	0.00%
1200	96	0.16%	769	0.03%	1250	0.00%
1800	64	0.16%	512	0.16%	833	0.04%
2000	58	0.53%	461	0.12%	750	0.00%
2400	48	0.16%	384	0.16%	625	0.00%
3600	32	0.16%	256	0.16%	416	0.16%
4800	24	0.16%	192	0.16%	312	0.16%
7200	16	0.16%	128	0.16%	208	0.16%
9600	12	0.16%	96	0.16%	156	0.16%
14400	8	0.16%	64	0.16%	104	0.16%
19200	6	0.16%	48	0.16%	78	0.16%
28800	4	0.16%	32	0.16%	52	0.16%
38400	3	0.16%	24	0.16%	39	0.16%
57600	2	0.16%	16	0.16%	26	0.16%
115200	1	0.16%	8	0.16%	13	0.16%
230400	---	---	4	0.16%	---	---
460800	---	---	2	0.16%	---	---
750000	---	---	---	---	2	0.00%
921600	---	---	1	0.16%	---	---
1500000	---	---	---	---	1	0.00%

Link Control Register (LCR) and Bank Select Register (BSR)

These registers are the same as the registers at offset 03h in bank 0.

Table 4.174 Bank 1 Bitmap

Register		Bits							
Offset	Mnemonic	7	6	5	4	3	2	1	0
00h	LBGD(L)	**LBGD7-0**							
01h	LBGD(H)	**LBGD15-8**							
02h	**Reserved**								
03h	LCR	**BKSE**	**SBRK**	**STKP**	**EPS**	**PEN**	**STB**	**WLS1**	**WLS0**
	BSR	**BKSE**	**BSR6-0**						
04-07h	**Reserved**								

BANK 2

Table 4.175 Bank 2 Register Map

Offset	Mnemonic	Register Name	Type	Section
00h	BGD(L)	Baud Generator Divisor Port (Low Byte)	R/W	
01h	BGD(H)	Baud Generator Divisor Port (High Byte)	R/W	
02h	EXCR1	Extended Control1	R/W	
03h	BSR	Bank Select	R/W	
04h	EXCR2	Extended Control 2	R/W	
05h	Reserved			
06h	TXFLV	TX_FIFO Level	RO	
07h	RXFLV	RX_FIFO Level	RO	

Baud Generator Divisor Ports, LSB (BGD(L)) and MSB (BGD(H))

These ports perform the same function as the Legacy Baud Divisor Ports in Bank 1 and are accessed identically, but do not change the operation mode of the functional block when accessed. See 'Legacy Baud Generator Divisor Ports LSB (LBGD(L)) and MSB (LBGD(H))' on 368 for more details.

Use these ports to set the baud when operating in Extended mode to avoid fallback to a Non-Extended operation mode, that is, 16550 compatible. When programming the baud, writing to BGDH causes the baud to change immediately.

Extended Control Register 1 (EXCR1)

Use this register to control operation in the Extended mode. Upon reset all bits are set to 0

Location: Offset 02h
Type: R/W

Bits	7	6	5	4	3	2	1	0
Mnemonic	**BTEST**	**Reserved**	**EDTLBK**	**LOOP**	**DMASWP**	**DMATH**	**DMANF**	**EXT_SL**
Reset	0	1	0	0	0	0	0	0
Required		1			0	0	0	

Bit	Description
7	**Baud Generator Test – BTEST** When set, this bit routes the Baud Generator output to the $\overline{\text{DTR}}$ pin for testing purposes.
6	**Reserved.** Write 1.
5	**Enable Transmitter During Loopback – EDTLBK** When this bit is set to 1, the transmitter serial output is enabled and functions normally when loopback is enabled
4	**Loopback Enable – LOOP** During loopback, the transmitter output is connected internally to the receiver input, to enable system self-test of serial communications. In addition to the data signal, all additional signals within the UART are interconnected to enable real transmission and reception using the UART mechanisms. When this bit is set to 1, loopback is selected. This bit accesses the same internal register as bit 4 in the MCR register, when the UART is in a Non-Extended mode. Loopback behaves similarly in both Non-Extended and Extended modes. When Extended mode is selected, the $\overline{\text{DTR}}$ bit of the MCR register internally drives both $\overline{\text{DSR}}$ and $\overline{\text{RI}}$, and the $\overline{\text{RTS}}$ bit drives $\overline{\text{CTS}}$ and $\overline{\text{DCD}}$. During loopback, the following actions occur: — The transmitter and receiver interrupts are fully operational. The Modem Status Interrupts are also fully operational, but the interrupt sources are now the lower bits of the MCR register. Modem interrupts in IR modes are disabled unless the IRMSSL bit of the IRCR2 register is 0. Individual interrupts are still controlled by the IER register bits. — The DMA control signals are fully operational. — UART and IR receiver serial input signals are disconnected. The internal receiver input signals are connected to the corresponding internal transmitter output signals. — The UART transmitter serial output is forced high and the IR transmitter serial output is forced low, unless the ETDLBK bit is set to 1 in which case they function normally. — The modem status input pins ($\overline{\text{DSR}}$, $\overline{\text{CTS}}$, $\overline{\text{RI}}$ and $\overline{\text{DCD}}$) are disconnected. The internal modem status signals are driven by the lower bits of the MCR register.
3	**DMA Swap – DMASWP** This bit selects the routing of the DMA control signals between the internal DMA logic and the configuration module of the chip. When this bit is 0, the transmitter and receiver DMA control signals are not swapped. When it is 1, they are swapped. A block diagram illustrating the control signals routing is given in Figure 4-19. The swap feature is particularly useful when only one 8237 DMA channel is used to serve both transmitter and receiver. In this case only one external DMA Request/DMA Acknowledge pair will be interconnected to the swap logic by the configuration module. Routing the external DMA channel to either the transmitter or the receiver DMA logic is then simply controlled by the DMASWP bit. This way, the IR device drivers do not need to know the details of the configuration module.

Bit	Description
2	**DMA FIFO Threshold – DMATH** This bit selects the TX_FIFO and RX_FIFO threshold levels used by the DMA request logic to support demand transfer mode. A transmission DMA request is generated when the TX_FIFO level is below the threshold. A reception DMA request is generated when the RX_FIFO level reaches the threshold or when a DMA timeout occurs. Table 4.176 lists the threshold levels for each FIFO.
1	**DMA Fairness Control – DMANF** This bit controls the maximum duration of DMA burst transfers. 0 = DMA requests forced inactive after approximately 10.5 msec of continuous transmitter and/or receiver DMA operation (default) 1 = TX-DMA request deactivated when the TX_FIFO is full. An RX DMA request is deactivated when the RX_FIFO is empty.
0	**Extended Mode Select – EXT_SL** When set to 1, Extended mode is selected.

Table 4.176 DMA Threshold Levels

Bit Value	DMA Threshold for FIFO Type		
	RX_FIFO	TX_FIFO	
		16 Levels	32 Levels
0	4	13	29
1	10	7	23

.

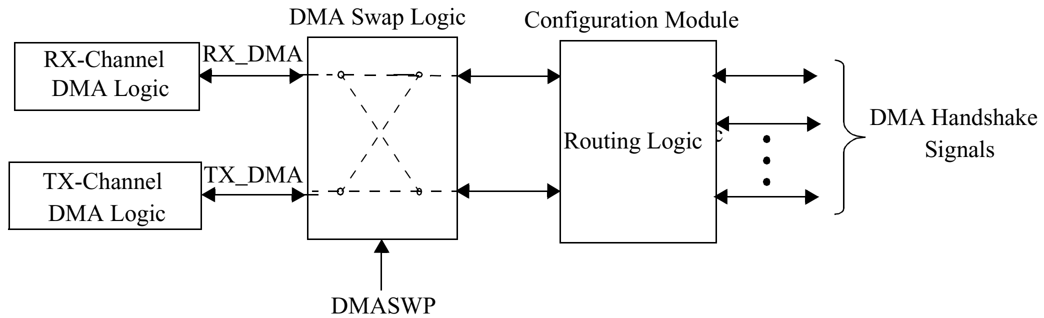


Figure 4-19 DMA Control Signals Routing

Bank Select Register (BSR)

These register is the same as the BSR register at offset 03h in bank 0.

Extended Control and Status Register 2 (EXCR2)

This register configures the RX_FIFO and TX_FIFO sizes and the value of the Prescaler and controls the Baud Divisor register Lock. Upon reset all bits are set to 0.

Location: Offset 04h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	LOCK	Reserved	PRESL1, 0		RF_SIZ1, 0		TF_SIZ1, 0	
Reset	0	0	0	0	0	0	0	0

Bit	Description
7	**Baud Divisor Register Lock – LOCK** When set to 1, accesses to the Baud Generator Divisor register through LBGD(L) and LBGD(H) as well as fallback are disabled from non-extended mode. In this case two scratchpad registers overlaid with LBGD(L) and LBGD(H) are enabled, and any attempted CPU access of the Baud Generator Divisor register through LBGD(L) and LBGD(H) will access the ScratchPad registers instead. This bit must be set to 0 when extended mode is selected.
6	**Reserved**
5-4	**Prescaler Select – PRESL1, 0** The prescaler divides the 24 MHz input clock frequency to provide the clock for the Baud Generator. **Bits** **5 4 Prescaler Value** 0 0 13 (Default) 0 1 1.625 1 0 Reserved 1 1 1.0
3-2	**RX_FIFO Levels Select – RF_SIZ 1, 0** These bits select the number of levels for the RX_FIFO. They are effective only when the FIFOs are enabled. **Bits** **3 2 RX_FIFO Levels** 0 0 16 (Default) 0 1 32 1 X Reserved
1-0	**TX_FIFO Levels Select – TF_SIZ1, 0** These bits select the number of levels for the TX_FIFO. They are effective only when the FIFOs are enabled. **Bits** **3 2 TX_FIFO Levels** 0 0 16 (Default) 0 1 32 1 X Reserved

Reserved Register

Upon reset, all bits in bank 2 register with offset 05h are set to 0.

TX_FIFO Current Level Register (TXFLV)

This register returns the number of bytes in the TX_FIFO. It can be used to facilitate programmed I/O modes during recovery from transmitter underrun in one of the fast IR modes.

Location: Offset 06h
Type: RO

Bit	7	6	5	4	3	2	1	0
Mnemonic	Reserved		TFL5-0					
Reset	0	0	0	0	0	0	0	0

Bit	Description
7-6	Reserved
5-0	**Number of Bytes in TX_FIFO – TFL5-0** These bits specify the number of bytes in the TX_FIFO. Note: The contents of TXFLV and RXFLV are not frozen during CPU reads. Therefore, invalid data could be returned if the CPU reads these registers during normal transmitter and receiver operation. To obtain correct data, the software should perform three consecutive reads and then take the data from the second read, if first and second read yield the same result, or from the third read, if first and second read yield different results.

RX_FIFO Current Level Register (RXFLV)

This register returns the number of bytes in the RX_FIFO. It can be used for software debugging.

Location: Offset 07
Type: RO

Bit	7	6	5	4	3	2	1	0
Mnemonic	Reserved		RFL5-0					
Reset	0	0	0	0	0	0	0	0

Bit	Description
7-6	**Reserved**
5-0	**Number of Bytes in RX_FIFO – RFL5-0** These bits specify the number of bytes in the RX_FIFO. Note: The contents of TXFLV and RXFLV are not frozen during CPU reads. Therefore, invalid data could be returned if the CPU reads these registers during normal transmitter and receiver operation. To obtain correct data, the software should perform three consecutive reads and then take the data from the second read, if first and second read yield the same result, or from the third read, if first and second read yield different results.

Table 4.177 Bank 2 Bitmap

Register		Bits							
Offset	Mnemonic	7	6	5	4	3	2	1	0
00h	BGD(L)	BGD7-0							
01h	BGD(H)	BGD15-8							
02h	EXCR1	BTEST	Reserved	EDTLBK	LOOP	DMASWP	DMATH	DMANF	EXT_SL
03h	BSR	BKSE	BSR6-0						
04h	EXCR2	LOCK	Reserved	PRESL1, 0		RF_SIZ1, 0		TF_SIZ1, 0	
05h	Reserved								
06h	TXFLV	Reserved		TFL5-0					
07h	RXFLV	Reserved		RFL5-0					

BANK 3

Table 4.178 Bank 3 Register Map

Offset	Mnemonic	Register Name	Type	Section
00h	MRID	Module Identification and Revision ID	RO	
01h	SH_LCR	Shadow of LCR	RO	
02h	SH_FCR	Shadow of FIFO Control	RO	
03h	BSR	Bank Select	R/W	
04h-07h	Reserved			

Module Identification and Revision ID Register (MRID)

This register identifies the revision of the module. When read, it returns the module ID and revision level in the format 2xh, where x indicates the revision number.

Location: Offset 00h
Type: RO

Bit	7	6	5	4	3	2	1	0
Mnemonic	MID3-0				RID3-0			
Reset	0	0	1	0	X	X	X	X

Bit	Description
7-4	**Module ID – MID3-0** The value in these bits identifies the module type.
3-0	**Revision ID – RID3-0** The value in these bits identifies the revision level.

Shadow of Link Control Register (SH_LCR)

This register returns the value of the LCR register. The LCR register is written into when a byte value, according to bits 1,0 of the LCR register in 'Link Control Register (LCR) and Bank Select Register (BSR)' on 358, is written to the LCR/BSR registers location (at offset 03h) from any bank.

Location: Offset 01h
Type: RO

Bit	7	6	5	4	3	2	1	0
Mnemonic	**Reserved**	**SBRK**	**STKP**	**EPS**	**PEN**	**STB**	**WLS1**	**WLS0**
Reset	0	0	0	0	0	0	0	0

See 'Link Control Register (LCR) and Bank Select Register (BSR)' on 358 for bit descriptions.

Shadow of FIFO Control Register (SH_FCR)

This register returns the contents of the FCR register in bank 0

Location: Offset 02h

Type: RO

Bit	7	6	5	4	3	2	1	0
Mnemonic	**RXFTH1, 0**		**TXFTH1, 0**		**Reserved**	**TXSR**	**RXSR**	**FIFO_EN**
Reset	0	0	0	0	0	0	0	0

Bank Select Register (BSR)

This register is the same as the BSR register at offset 03h in bank 0.

Reserved Registers

Bank 3 registers with offsets 04h to 07h are reserved.

Table 4.179 Bank 3 Bitmap

Register		Bits							
Offset	Mnemonic	7	6	5	4	3	2	1	0
00h	MRID	**MID3-0**				**RID3-0**			
01h	SH_LCR	**Reserved**	**SBRK**	**STKP**	**EPS**	**PEN**	**STB**	**WLS1**	**WLS0**
02h	SH_FCR	**RXFTH1, 0**		**TXFTH1, 0**		**Reserved**	**TXSR**	**RXSR**	**FIFO_EN**
03h	BSR	**BKSE**	**BSR6-0**						
04h-07h	**Reserved**								

BANK 4

Table 4.180 Bank 4 Register Map

Offset	Mnemonic	Register Name	Type	Page
00h	TMR(L)	Timer (Low Byte)	RO	page 377
01h	TMR(H)	Timer (High Byte)	RO	
02h	IRCR1	IR Control 1		page 378
03h	BSR	Bank Select		page 378
04h	TFRL(L)/ TFRCC(L)	Transmitter Frame Length (Low Byte)/ Transmitter Frame Current Count (Low Byte)	R/W	page 378
05h	TFRL(H)/ TFRCC(H)	Transmitter Frame Length (High Byte)/ Transmitter Frame Current Count (High Byte)	R/W	
06h	RFRML(L)/ RFRCC(L)	Receiver Frame Maximum Length (Low Byte)/ Receiver Frame Current Count (Low Byte)	R/W	page 379
07h	RFRML(H)/ RFRCC(H)	Receiver Frame Maximum Length (High Byte)/ Receiver Frame Current Count (High Byte)	R/W	

Timer Registers (TMR)

The Timer registers at offsets 00h, 01h are used to program the reload value for the internal down-counter as well as to read the current counter value. TMR is 12 bits wide and is split into two independently accessible parts occupying consecutive address locations. TMR(L) is located at the lower address and accesses the least significant 8 bits, whereas TMR(H) is located at the higher address and accesses the most significant 4 bits. Values from 1 to $2^{12}-1$ can be used. The 0value is reserved and must not be used. The upper 4 bits of TMR(H) are reserved and must be written with 0's.

The timer resolution is 125 ms, providing a maximum timeout interval of approximately 0.5 seconds. To properly program the timer, the CPU must always write the lower value into TMR(L) first, and then the upper value into TMR(H). Writing into TMR(H) causes the counter to be loaded. A read of TMR returns the current counter value if the CTEST bit is 0, or the programmed reload value if CTEST is 1. In order for a read access to return an accurate value, the CPU should always read TMR(L) first, and then TMR(H). This is because a read of TMR(H) returns the content of an internal latch that is loaded with the 4 most significant bits of the current counter value when TMR(L) is read. After reset, the content of this register is indeterminate.

IR Control Register 1 (IRCR1)

Enables the Sharp-IR or SIR IR mode in Non-Extended mode of operation.

Upon reset, all bits are set to 0.

Location: Offset 02h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	Reserved				IR_SL1, 0		CTEST	TMR_EN
Reset	0	0	0	0	0	0	0	0

Bit	Description
7-4	**Reserved**
3-2	**Sharp-IR or SIR Mode Select – IR_SL1, 0** Non-Extended mode only. These bits enable Sharp-IR and SIR modes in Non-Extended mode. They allow selection of the appropriate IR interface when Extended mode is not selected. These bits are ignored when Extended mode is selected. **Bits** **1 0 Selected Mode** 0 0 UART (Default) 0 1 Reserved 1 0 Sharp-IR 1 1 SIR
1	**Counters Test – CTEST** When this bit is set to 1, the TMR register reload value, as well as the TFRL and RFRML register contents, are returned during CPU reads.
0	**Timer Enable – TMR_EN** Extended mode only. When this bit is 1, the timer is enabled. When it is 0, the timer is frozen.

Bank Select Register (BSR)

These register is the same as the BSR register at offset 03h in bank 0.

Transmitter Frame Length/Current Count (TFRL/TFRCC)

These registers share the same addresses. TFRL is always accessed during write cycles and is used to program the frame length, in bytes, for the frames to be transmitted. The frame length value does not include any appended CRC bytes. TFRL is accessed during read cycles if the CTEST bit is set to 1, and returns the previously programmed value. Values from 1 to 2^{13} –1 can be used. The 0value is reserved and must not be used.

TFRCC is loaded with the content of TFRL when transmission of a frame begins, and decrements after each byte is transmitted. It is read only and is accessed during CPU read cycles when the CTEST bit is 0. It returns the number of currently remaining bytes of the frame being transmitted. These registers are 13 bits wide and are split into two independently accessible parts occupying consecutive address locations. TFRL(L) and TFRCC(L) are located at the lower address and access the least significant 8 bits, whereas TFRL(H) and TFRCC(H) are located at the higher address and access the most significant 5 bits. To properly program TFRL, the CPU must always write the lower value into TFRL(L) first, and then the upper value into TFRL (H). The upper 3 bits of TFRL(H) are reserved and must be written with 0's. In order for a read access of TFRCC to return an accurate value, the CPU should always read TFRCC(L) first, and then TFRCC(H). After reset, the content of the TFRL register is 800h.

Receiver Frame Maximum Length/Current Count (RFRML/RFRCC)

These registers share the same addresses. RFRML is always accessed during write cycles and is used to program the maximum frame length, in bytes, for the frames to be received. The maximum frame length value includes the CRC bytes. RFRML is accessed during read cycles if the CTEST bit is set to 1, and returns the previously programmed value.

Values from 4 to $2^{13} - 1$ can be used. The values from 0 to 3 are reserved and must not be used. RFRCC holds the current byte count of the incoming frame, and increments after each byte is received. It is read only and is accessed during CPU read cycles when the CTEST bit is 0. These registers are 13 bits wide and are split into two independently accessible parts occupying consecutive address locations. RFRML(L) and RFRCC(L) are located at the lower address and access the least significant 8 bits, whereas RFRML(H) and RFRCC(H) are located at the higher address and access the most significant 5 bits. To properly program RFRML, the CPU must always write the lower value into RFRML(L) first, and then the upper value into RFRML(H). The upper 3 bits of RFRML(H) are reserved and must be written with 0's. In order for a read access of RFRCC to return an accurate value, the CPU should always read RFRCC(L) first, and then RFRCC(H). After reset, the content of the RFRML register is 800h.

Note: TFRCC and RFRCC are intended for testing purposes only. Use of these registers for any other purpose is not recommended.

Table 4.181 Bank 4 Bitmap

Register		**Bits**							
Offset	Mnemonic	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
00h	TMR(L)	**TMR7-0**							
01h	TMR(H)	**Reserved**				**TMR11-8**			
02h	IRCR1	**Reserved**				**IR_SL1,0**		CTEST	TMR_EN
03h	BSR	**BKSE**	**BSR6-0**						
04h	TFRL(L)/ TFRCC(L)	**TFRL7-0/TFRCC7-0**							
05h	TFRL(H)/ TFRCC(H)	**Reserved**			**TFRL12-8/TFRCC12-8**				
06h	RFRML(L)/ RFRCC(L)	**RFRML7-0/RFRCC7-0**							
07h	RFRML(H)/ RFRCC(H)	**Reserved**			**RFRML12-8/RFRCC12-8**				

BANK 5

Table 4.182 Bank 5 Register Map

Offset	Mnemonic	Register Name	Type	Reference
00h	SPR2	Scratchpad 2	R/W	Page 380
01h	SPR3	Scratchpad 3	R/W	
02h	Reserved			
03h	BSR	Bank Select	R/W	Page 380
04h	IRCR2	IR Control 2	R/W	Page 380
05h	FRM_ST	Frame Status	RO	
06h	RFRL(L)/LSTFRC	Received Frame Length (Low Byte) / Lost Frame Count	RO	
07h	RFRL(H)	Received Frame Length (High Byte)	RO	

Scratchpad Registers

These registers are to be used by the programmer to contain data temporarily. They have no control over the device operation. The reset value of SPR2 is 01h; the reset value of SPR3 is 00h.

Reserved Register

Bank 5 register with offset 02h is reserved.

Bank Select Register (BSR)

This register is the same as the BSR register at offset 03h in bank 0.

IR Control Register 2 (IRCR2)

This register controls the basic settings of the IR modes. Upon reset, the content of this register is 02h

Location: Offset 04h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	**Reserved**	**SFTSL**	**FEND_MD**	**AUX_IRRX**	**TX_MS**	**MDRS**	**IRMSSL**	**IR_FDPLX**
Reset	0	0	0	0	0	0	1	0

Bit	Description
7	**Reserved**
6	**ST_FIFO Threshold Select – SFTSL** An interrupt request is generated when the ST_FIFO level reaches the threshold or when an ST_FIFO timeout occurs. 0: Threshold level 2 (default) 1: Threshold level 4
5	**Frame End Control – FEND_MD** This bit selects whether a terminal-count condition from the TFRCC register generates an EOF in PIO mode or DMA mode. 0: TFRCC terminal count effective in PIO mode (default) 1: TFRCC terminal count effective in DMA mode

Bit	Description
4	**Auxiliary IR Input Select – AUX_IRRX** When set to 1, the IR signal is received from the auxiliary input. (Separate input signals may be desired for different front end circuits). See Table 4.195.
3	**Transmitter Mode Select – TX_MS** This bit is used in MIR and FIR modes only. When it is set to 1, transmitter frame-end stop mode is selected. In this case the transmitter stops after transmission of a frame is complete, if the end-of-frame condition was generated by the TFRCC counter reaching 0. The transmitter can be restarted by clearing the TXHFE bit of the ASCR register.
2	**MIR Data Rate Select – MDRS** This bit determines the data rate in MIR mode. 0 = 1.152 Mbps 1 = 0.576 Mbps
1	**MSR Register Function Select in IR Mode – IRMSSL** This bit selects the behavior of the Modem Status register (MSR) and the Modem Status Interrupt (MS_EV) when any IR mode is selected. When a UART mode is selected, the Modem Status register and the Modem Status Interrupt function normally, and this bit is ignored. 0 = MSR register and modem status interrupt work in the IR modes as in the UART mode (Enables external circuitry to perform carrier detection and provide wake-up events). 1 = MSR register returns 30h, and Modem Status Interrupt disabled (default)
0	**Enable IR Full Duplex Mode – IR_FDPLX** When set to 1, the IR receiver is not masked during transmission.

The Status FIFO

The ST_FIFO is used in MIR and FIR Modes.

It is an 8-level FIFO and is intended to support back-to-back incoming frames in DMA mode, when an 8237-type DMA controller is used. Each ST_FIFO entry contains either status information and frame length for a single frame, or the number of lost frames.

The bottom entry spans three address locations, and is accessed via the FRMST, RFRL(L)/LSTFRC and RFRL(H) registers. The ST_FIFO is flushed when a hardware reset occurs or when the receiver is soft reset.

Note: The status and length information of received frames is loaded into the ST_FIFO whenever the DMA_EN bit of the extended-mode MCR register is set to 1 and an 8237 type DMA controller is used, regardless of whether the CPU or the DMA controller is transferring the data from the RX_FIFO to memory. This implies that, during testing, if full duplex is enabled and a DMA channel is servicing the transmitter while the CPU is servicing the receiver, the CPU must still read the ST_FIFO. Otherwise, it fills up and incoming frames will be rejected.

Frame Status (FRM_ST)

This register returns the status byte at the bottom of the ST_FIFO. If the LOSTFR bit is 0, bits 0 to 4 indicate if any error condition occurred during reception of the corresponding frame. Error conditions also affect the error flags in the LSR register.

Location: Offset 05h
Type: RO

Bit	7	6	5	4	3	2	1	0
Mnemonic	VLD	LOST_FR	Reserved	MAX_LEN	PHY_ERR	BAD_CRC	OVR1	OVR2
Reset	0	0	0	0	0	0	1	0

Bit	Description
7	**ST_FIFO Entry Valid – VLD** When set to 1, the bottom ST_FIFO entry contains valid data.
6	**Lost Frame Flag – LOST_FR** Indicates the type of information provided by this ST_FIFO entry. 0 = Entry provides status information and length for a received frame 1 = Entry provides overrun indications and number of lost frames
5	**Reserved**
4	**Maximum Frame Length Exceeded – MAX_LEN** Set to 1 when a frame exceeding the maximum length has been received.
3	**Physical Layer Error – PHY_ERR** Set to 1 when an encoding error or the sequence BOF-data-BOF is detected in FIR mode, or an abort condition is detected in MIR mode.
2	**CRC Error – BAD_CRC** Set to 1 when a mismatch between the received CRC and the receiver-generated CRC is detected.
1	**Overrun Error 1 – OVR1** This bit is set to 1 when incoming characters or entire frames have been discarded due to the RX_FIFO being full.
0	**Overrun Error 2 – OVR2** This bit is set to 1 when incoming characters or entire frames have been discarded due to the ST_FIFO being full.

Received Frame Length Low Byte (RFRL(L)) / Lost Frame Count (LSTFRC)

This read only register should be read only when the VLD bit in FRM_ST is 1. The information returned depends on the setting of the LOST_FR bit. Upon reset, all bits are set to 0.

LOST_FR = 0: Least significant 8 bits of the received frame length.
LOST_FR = 1: Number of lost frames

Received Frame Length High Byte (RFRL(H))

This read only register should be read only when the VLD bit in FRM_ST is 1. The information returned depends on the setting of the LOST_FR bit. Upon reset, all bits are set to 0.

LOST_FR = 0: Most significant 5 bits of the received frame length.
LOST_FR = 1: Reading this register removes the bottom ST_FIFO entry.

Table 4.183 Bank 5 Bitmap

Register		Bits							
Offset	Mnemonic	7	6	5	4	3	2	1	0
00h	SPR2	**Scratchpad 2**							
01h	SPR3	**Scratchpad 3**							
02h	**Reserved**								
03h	BSR	**BKSE**	**BSR6-0**						
04h	IRCR2	**Reserved**	**SFTSL**	**FEND_MD**	**AUX_IRRX**	**TX_MS**	**MDRS**	**IRMSSL**	**IR_FDPLX**
05h	FRM_ST	**VLD**	**LOST_FR**	**Reserved**	**MAX_LEN**	**PHY_ERR**	**BAD_CRC**	**OVR1**	**OVR2**
06h	RFRL(L)/ LSTFRC	**RFRL7-0/LSTFRC7-0**							
07h	RFRL(H)	**RFRL15-8**							

BANK 6

Table 4.184 Bank 6 Register Map

Offset	Mnemonic	Register Name	Type	Page
00h	IRCR3	IR Control 3	R/W	Page 384
01h	MIR_PW	MIR Pulse Width Control	R/W	Page 386
02h	SIR_PW	SIR Pulse Width Control	R/W	Page 386
03h	BSR	Bank Select	R/W	Page 386
04h	BFPL	Beginning Flags/ Preamble Length	R/W	Page 386
05h-07h	Reserved			

IR Control Register 3 (IRCR3)

This register is used to select the operating mode of the infrared interface.

Location: Offset 00h
Type: R/W.

Bit	7	6	5	4	3	2	1	0
Mnemonic	**SHDM_DS**	**SHMD_DS**	**FIR_CRC**	**MIR_CRC**	**Reserved**	**TXCRC_INV**	**TXCRC_DS**	**Reserved**
Reset	0	0	1	0	0	0	0	0

Bit	Description
7	**Sharp-IR Demodulation Disable – SHDM_DS** 0 = Internal 500 KHz receiver demodulation enabled (default) 1 = Internal demodulation disabledþ
6	**Sharp-IR Modulation Disable – SHMD_DS** 0 = Internal 500 KHz transmitter modulation enabled (default) 1 = Internal modulation disabled
5	**FIR Mode CRC Select – FIR_CRC** Determines the length of the CRC in FIR mode. 0 = 16-bit CRC 1 = 32-bit CRC
4	**MIR Mode CRC Select – MIR_CRC** Determines the length of the CRC in MIR mode. 0 = 16-bit CRC 1 = 32-bit CRC
3	**Reserved**
2	**Invert Transmitter CRC – TXCRC_INV** When set to 1, an inverted CRC is transmitted.This bit can be used to force a bad CRC for testing purposes.
1	**Disable Transmitter CRC – TXCRC_DS** When set to 1, a CRC is not transmitted.
0	**Reserved**

MIR Pulse Width Register (MIR_PW)

This register is used to program the width of the transmitted MIR infrared pulses in increments of either 20.833 ns or 41.666 ns depending on the setting of the MDSR bit of the IRCR2 register. The programmed value has no effect on the MIR receiver. After reset, the content of this register is 0Ah.

Location: Offset 01h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	Reserved				MPW3-0			
Reset	0	0	0	0	1	0	1	0

Bit	Description
7-4	**Reserved**
3-0	**MIR Pulse Width – MPW 3-0** The settings of the pulse width are given in Table 4.185.

Table 4.185 IMIR Pulse Width Settings

Encoding	Pulse Width MDRS = 0	Pulse Width MDRS = 1
00XX	Reserved	Reserved
0100	83.33 ns	166.66 ns
0101	104.16 ns	208.33 ns
0110	125 ns	250 ns
0111	145.83 ns	291.66 ns
1000	166.66 ns	333.33 ns
1001	187.50 ns	374.99 ns
1010	208.33 ns	416.66 ns
1011	229.16 ns	458.33
1100	250 ns	500 ns
1101	270.83 ns	541.66 ns
1110	291.66 ns	583.32 ns
1111	312.5 ns	625 ns

SIR Pulse Width Register (SIR_PW)

This register sets the pulse width for transmitted pulses in SIR operation mode. These settings do not affect the receiver. Upon reset, the content of this register is 00h, which defaults to a pulse width of 3/16 of the baud

Location: Offset 02h
Type: R/W.

Bit	7	6	5	4	3	2	1	0
Mnemonic	Reserved				SPW3-0			
Reset	0	0	0	0	0	0	0	0

Bit	Description
7-4	**Reserved**
3-0	**SIR Pulse Width – SPW 3-0** Two codes for setting the pulse width are available. All other values for this field are reserved. 0000: Pulse width = 3/16 of bit period (default) 1101: Pulse width = 1.6 msec

Bank Select Register (BSR)

These register is the same as the BSR register at offset 03h in bank 0.

Beginning Flags/Preamble Length Register (BFPL)

Used to program the number of beginning flags and preamble symbols for MIR and FIR modes respectively.

After reset, the content of this register is 2Ah, selecting 2 beginning flags and 16 preamble symbols.

Location: Offset 04h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	MBF7-4				FPL3-0			
Reset	0	0	1	0	1	0	1	0

Bit	Description
7-4	**MIR Beginning Flags– MBF7-4** Selects the number of beginning flags for MIR frames.The settings of the are given in Table 4.186.
3-0	**FIR Preamble Length – FPL3-0** Selects the number of preamble symbols for FIR frames.The settings of the are given in Table 4.187.

Table 4.186 MIR Beginning Flags

Encoding	Beginning Flags
0000	Reserved
0001	1
0010	2 (Default)
0011	3
0100	4
0101	5
0110	6
0111	8
1000	10
1001	12
1010	16
1011	20
1100	24
1101	28
1110	32
1111	Reserved

Table 4.187 FIR Preamble Length

Encoding	Preamble Length
0000	Reserved
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	8
1000	10
1001	12
1010	16 (Default)
1011	20
1100	24
1101	28
1110	32
1111	Reserved

Reserved Registers

Bank 6 registers with offsets 05h-07h are reserved.

Table 4.188 Bank 6 Bitmap

Register		Bits							
Offset	Mnemonic	7	6	5	4	3	2	1	0
00h	IRCR3	SHDM_DS	SHDM_DS	FIR_CRC	MIR_CRC	Reserved	TXCRC_INV	TXCRC_DS	Reserved
01h	MIR_PW	Reserved				MPW3	MPW2	MPW1	MPW0
02h	SIR_PW	Reserved				SPW3	SPW2	SPW1	SPW0
03h	BSR	BKSE	BSR6-0						
04h	BFPL	MBF7-4				FPL3-0			
05h-07h	Reserved								

BANK 7

Table 4.189 Bank 7 Register Map

Offset	Mnemonic	Register Name	Type	Section
00h	IRRXDC	IR Receiver Demodulator Control	R/W	
01h	IRTXMC	IR Transmitter Modulator Control	R/W	
02h	RCCFG	CEIR Configuration	R/W	
03h	BSR	Bank Select	R/W	
04h	IRCFG1	IR Interface Configuration 1	Varies per bit	
05h	Reserved			
06h	Reserved			
07h	IRCFG4	IR Interface Configuration 4	R/W	

The CEIR utilizes two carrier frequency ranges (see also Table 4.194):

- Low range which spans from 30 KHz to 56 KHz, in 1 KHz increments, and
- High range which includes three frequencies: 400 KHz, 450 KHz or 480 KHz.

High and low frequencies are specified independently to allow separate transmission and reception modulation settings. The transmitter uses the carrier frequency settings in Table 4.194.

The four registers at offsets 04h through 07h (the IR transceiver configuration registers) are provided to configure the IR Interface (the transceiver). The transceiver mode is selected by up to three special output signals (IRSL2-0). When programmed as outputs these signals are forced to low when a UART mode is selected.

IR Receiver Demodulator Control Register (IRRXDC)

This register controls settings for Sharp-IR and CEIR reception. After reset, the content of this register is 29h. This setting selects a subcarrier frequency in a range between 34.61 KHz and 38.26 KHz for the CEIR mode, and from 480.0 to 533.3 KHz for the Sharp-IR mode. The value of this register is ignored in both modes if the receiver demodulator is disabled (see bit 4 at 'CEIR Configuration Register (RCCFG)' on 392). The available frequency ranges for CEIR and Sharp-IR modes are given in Tables 4.190 through 4.192.

Location: Offset 00h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	DBW2-0			DFR4-0				
Reset	0	0	1	0	1	0	0	1

Bit	Description
7-5	**Demodulator Bandwidth – DBW2-0** These bits set the demodulator bandwidth for the selected frequency range. The subcarrier signal frequency must fall within the specified frequency range in order to be accepted. Used for both Sharp-IR and CEIR modes.
4-0	**Demodulator Frequency – DFR4-0** These bits select the subcarrier's center frequency for the CEIR mode.

Table 4.190 CEIR, Low Speed Demodulator (RXHSC = 0)

DFR Bits 43210	DBW2-0 Bits (Bits 7-5 of IRRXDC)												Freq.
	001		010		011		100		101		110		
	min	max	min	max	min	max	min	max	min	max	min	max	
00011	28.6	31.6	27.3	33.3	26.1	35.3	25.0	37.5	24.0	40.0	23.1	42.9	KHz
00100	29.3	32.4	28.0	34.2	26.7	36.2	25.6	38.4	24.6	41.0	23.7	43.9	KHz
00101	30.1	33.2	28.7	35.1	27.4	37.1	26.3	39.4	25.2	42.1	24.3	45.1	KHz
00110	31.7	35.1	30.3	37.0	29.0	39.2	27.8	41.7	26.7	44.4	25.6	47.6	KHz
00111	32.6	36.0	31.1	38.1	29.8	40.3	28.5	42.8	27.4	45.7	26.3	48.9	KHz
01000	33.6	37.1	32.0	39.2	30.7	41.5	29.4	44.1	28.2	47.0	27.1	50.4	KHz
01001	34.6	38.3	33.0	40.4	31.6	42.8	30.3	45.4	29.1	48.5	28.0	51.9	KHz
01011	35.7	39.5	34.1	41.7	32.6	44.1	31.3	46.9	30.0	50.0	28.8	53.6	KHz
01100	36.9	40.7	35.2	43.0	33.7	45.5	32.3	48.4	31.0	51.6	29.8	55.3	KHz
01101	38.1	42.1	36.4	44.4	34.8	47.1	33.3	50.0	32.0	53.3	30.8	57.1	KHz
01111	39.4	43.6	37.6	45.9	36.0	48.6	34.5	51.7	33.1	55.1	31.8	59.1	KHz
10000	40.8	45.1	39.0	47.6	37.3	50.4	35.7	53.6	34.3	57.1	33.0	61.2	KHz
10010	42.3	46.8	40.4	49.4	38.6	52.3	37.0	55.6	35.6	59.3	34.2	63.5	KHz
10011	44.0	48.6	42.0	51.3	40.1	54.3	38.5	57.7	36.9	61.5	35.5	65.9	KHz

Table 4.190 CEIR, Low Speed Demodulator (RXHSC = 0)

DFR Bits 43210	DBW2-0 Bits (Bits 7-5 of IRRXDC)												
	001		010		011		100		101		110		Freq.
	min	max	min	max	min	max	min	max	min	max	min	max	
10101	45.7	50.5	43.6	53.3	41.7	56.5	40.0	60.0	38.4	64.0	36.9	68.6	KHz
10111	47.6	52.6	45.5	55.6	43.5	58.8	41.7	62.5	40.0	66.7	38.5	71.4	KHz
11010	49.7	54.9	47.4	57.9	45.3	61.4	43.5	65.2	41.7	69.5	40.1	74.5	KHz
11011	51.9	57.4	49.5	60.6	47.4	64.1	45.4	68.1	43.6	72.7	41.9	77.9	KHz
11101	54.4	60.1	51.9	63.4	49.7	67.2	47.6	71.4	45.7	76.1	43.9	81.6	KHz

Table 4.191 Consumer IR High Speed Demodulator (RXHSC = 1)

DFR Bits 43210	DBW2-0 Bits (Bits 7-5 of IRRXDC)												
	001		010		011		100		101		110		Freq.
	min	max	min	max	min	max	min	max	min	max	min	max	
00011	381.0	421.1	363.6	444.4	347.8	470.6	333.3	500.0	320.0	533.3	307.7	571.4	KHz
01000	436.4	480.0	417.4	505.3	400.0	533.3	384.0	564.7	369.2	600.0	355.6	640.0	KHz
01011	457.7	505.3	436.4	533.3	417.4	564.7	400.0	600.0	384.0	640.0	369.9	685.6	KHz

Table 4.192 Sharp-IR Demodulator

DFR Bits 4321 0	DBW2-0 Bits (Bits 7-5 of IRRXDC)												
	001		010		011		100		101		110		Freq.
	min	max	min	max	min	max	min	max	min	max	min	max	
xxxxx	480.0	533.3	457.1	564.7	436.4	600.0	417.4	640.0	400.0	685.6	384.0	738.5	KHz

IR Transmitter Modulator Control Register (IRTXMC)

This register selects the modulation subcarrier parameters for CEIR and Sharp-IR mode transmission. For Sharp-IR, only the subcarrier pulse width is controlled by this register - the subcarrier frequency is fixed at 500 KHz.

After reset, the value of this register is 69h, selecting a carrier frequency of 36 KHz and an IR pulse width of 7 msec for CEIR, or a pulse width of 0.8 msec for Sharp-IR.

Type: R/W
Location: Offset 01

Bit	7	6	5	4	3	2	1	0
Mnemonic		**MCPW2-0**				**MCFR4-0**		
Reset	0	1	1	0	1	0	0	1

Bit	Description
7-5	**Modulation Subcarrier Pulse Width – MCPW2-0** Specify the pulse width of the subcarrier clock as shown in Table 4.193.
4-0	**Modulation Subcarrier Frequency – MCFR4-0** These bits set the frequency for the CEIR modulation subcarrier. The encoding is defined in Table 4.194.

Table 4.193 Carrier Clock Pulse Width Options (Frequency Ranges in KHz)

Encoding MCPW Bits 7 6 5	Low Frequency (TXHSC = 0)	High Frequency (TXHSC = 1)	Encoding MCPW Bits 7 6 5	Low Frequency (TXHSC = 0)	High Frequency (TXHSC = 1)
0 0 0	Reserved	Reserved	1 0 0	9 msec	0.9 msec
0 0 1	Reserved	Reserved	1 0 1	10.6 msec	Reserved
0 1 0	6 msec	0.7 msec	1 1 0	Reserved	Reserved
0 1 1	7 msec	0.8 msec	1 1 1	Reserved	Reserved

Table 4.194 CEIR Carrier Frequency Encoding (Frequency Ranges in KHz)

Encoding MCFR Bits 43210	Low Frequency (TXHSC = 0)	High Frequency (TXHSC = 1)	Encoding MCFR Bits 43210	Low Frequency (TXHSC = 0)	High Frequency (TXHSC = 1)
00000	Reserved	Reserved	01011	38 KHz	480 KHz
00001	Reserved	Reserved	01100	39 KHz	Reserved
00010	Reserved	Reserved	01101	40 KHz	Reserved
00011	30 KHz	400 KHz	01110	41 KHz	Reserved
00100	31 KHz	Reserved	---	---	---
00101	32 KHz	Reserved	11010	53 KHz	Reserved
00110	33 KHz	Reserved	11011	54 KHz	Reserved
00111	34 KHz	Reserved	11100	55 KHz	Reserved

Table 4.194 CEIR Carrier Frequency Encoding (Frequency Ranges in KHz) (cont.)

Encoding MCFR Bits 43210	Low Frequency (TXHSC = 0)	High Frequency (TXHSC = 1)	Encoding MCFR Bits 43210	Low Frequency (TXHSC = 0)	High Frequency (TXHSC = 1)
01000	35 KHz	450 KHz	11101	56 KHz	Reserved
01001	36 KHz	Reserved	11110	56.9 KHz	Reserved
01010	37 KHz	Reserved	11111	Reserved	Reserved

CEIR Configuration Register (RCCFG)

This register control the basic operation of the CEIR mode.

Location: Offset 02h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	R_LEN	T_OV	RXHSC	RCDM_DS	Reserved	TXHSC	RC_MMD1, 0	
Reset	0	0	0	0	0	0	0	0

Bit	Description
7	**Run Length Control – R_LEN** When set to 1 enables or disables run length encoding/decoding. The format of a run length code is: YXXXXXXX where Y is the bit value and XXXXXXX is the number of bits minus 1 (Selects from 1 to 128 bits).
6	**Receiver Sampling Mode – T_OV** 0 = Programmed T period sampling 1 = Oversampling mode
5	**Receiver Carrier Frequency Select – RXHSC** This bit selects the receiver demodulator frequency range. 0 = Low frequency: 30-56.9 KHz 1 = High frequency: 400-480 KHz
4	**Receiver Demodulation Disable – RCDM_DS** When this bit is 1, the internal demodulator is disabled. The internal demodulator, when enabled, performs carrier frequency checking and envelope detection. This bit must be set to 1 (disabled), when the demodulation is performed externally, or when oversampling mode is selected to determine the carrier frequency.
3	**Reserved**
2	**Transmitter Subcarrier Frequency Select – TXHSC** This bit selects the modulation carrier frequency range. 0 = Low frequency: 30-56.9 KHz 1 = High frequency: 400-480 KHz

Bit	Description
1-0	**Transmitter Modulator Mode – RC_MMD1,0** Determine how IR pulses are generated from the transmitted bit string. Bits Mode Description 00 C_PLS Modulation Mode. Pulses are generated continuously for the entire logic 0 bit time. 01 8_PLS Modulation Mode. 8 pulses are generated each time one or more logic 0 bits are transmitted following a logic 1 bit. 10 6_PLS Modulation Mode. 6 pulses are generated each time one or more logic 0 bits are transmitted following a logic 1 bit. 11 Reserved. Result is indeterminate.

Bank Select Register (BSR)

This register is the same as the BSR register at offset 03h in bank 0.

IR Interface Configuration Register 1 (IRCFG1)

This register holds the transceiver configuration data for Sharp-IR and SIR modes. It is also used to directly control the transceiver operation mode when automatic configuration is not enabled. The four least significant bits are also used to read the identification data of a Plug and Play IR interface adapter.

Location: Offset 04h
Type: Varies per bit

Bit	7	6	5	4	3	2	1	0
Mnemonic	**STRV_MS**	**Reserved**	**Set IRTX**[a]	**IRRX1 Level**[a]	**IRID3**	**IRIC2-0**		
Reset	0	0	0	0	0	0	0	X

a. **The values shown for bits 5-4 are valid when the value of the MRID register, bank 3, is 24h or higher. If the MRID register value is lower than 24h, bits 5-4 are reserved.**

Bit	Description
7	**Special Transceiver Mode Selection – STRV_MS** When this R/W bit is set to 1, the IRTX output signal is forced to active high and a timer is started. The timer times out after 64 msec, at which time the bit is reset and the IRTX output signal becomes low again. The timer is restarted every time a 1 is written to this bit. Although it is possible to extend the period during which IRTX remains high beyond 64 msec, this should be avoided to prevent damage to the transmitter LED. Writing a 0 to this bit has no effect.
6	**Reserved**
5	**Set IRTX** When this R/W bit is set to 1, it forces the IRTX signal high.
4	**IRRX1 Level** This RO bit reflects the value of the IRRX1 input signal, either 0 or 1.
3	**Transceiver Identification – IRID3 (** Upon read, this RO bit returns the logic level of the IRSL3 pin. Data written to this bit is ignored.

Bit	Description
2-1	**Transceiver Identification/Control – IRIC2-1** The function of these R/W bits depends on whether IRSL2-1 pins are programmed as inputs or outputs. If IRSL(2-1) are programmed as input (IRSL21_DS = 0) then: • Upon read, these bits return the logic level of the pins (allowing external devices to identify themselves). • Data written to these bit positions is ignored. If IRSL(2-1) are programmed as output (IRSL21_DS = 1) then: • If AMCFG (bit 7 of IRCFG4) is set to 1, these bits drive the IRSL(2-1) pins when Sharp-IR mode is selected. • If AMCFG is 0, these bits drive the IRSL(2-1) pins, regardless of the selected mode. Upon read, these bits return the values previously written.
0	**IRIC0 (Transceiver Identification/Control).** The function of this R/W bit depends on whether IRSL0/IRRX2 pin is programmed as an input or an output. If IRSL0/IRRX2 is programmed as an input (IRSL0_DS = 0) then: • Upon read, this bit returns the logic level of the pin (allowing external devices to identify themselves). • Data written to this bit position is ignored. If IRSL0/IRRX2 is programmed as an output (IRSL0_DS = 1), then: • If AMCFG (bit 7 of IRCFG4) is set to 1, this bit drives the IRSL0/IRRX2 pin when Sharp-IR mode is selected. • If AMCFG is 0, this bit drive the IRSL0/IRRX2 pin, regardless of the selected mode. Upon read, this bit returns the value previously written.

Reserved Registers

Bank 7 registers with offset 05h and 06h are reserved.

IR Interface Configuration Register 4 (IRCFG4)

This register configures the receiver data path and enables the automatic selection of the configuration pins.

After reset, this register contains 00h.

Location: Offset 07h
Type: R/W

Bit	7	6	5	4	3	2	1	0
Mnemonic	**Reserved**	**IRRX_MD**	**IRSL0_DS**	**RXINV**	**IRSL21_DS**	**Reserved**		
Reset	0	0	0	0	0	0	0	0

Bit	Description
7	**Reserved.** This bit must be written with 0.
6	**IRRX Mode Select – IRRX_MD** Determines whether a single input or two separate inputs are used for Low-Speed and High-Speed IrDA modes. Table 4.195 shows the IRRXn pins used for the low-speed and high-speed infrared modes, and for the various combinations of IRSL0_DS, IRRX_MD and AUX_IRRX. 0 = One input is used for both SIR and MIR/FIR. 1 = Separate inputs are used for SIR and MIR/FIR.
5	**IRSL0/IRRX2 Pin Direction Select – IRSL0_DS** This bit determines the direction of the IRSL0/IRRX2 pin. See Table 4.195. 0 = Pin direction input 1 = Pin direction output
4	**IRRX Signal Invert – RXINV** This bit supports optical transceivers with receive signals of opposite polarity (active high instead of active low). When set to 1, an inverter is placed on the receiver input signal path.
3	**IRSL2-1 Pin Direction Select – IRSL21_DS** This bit determines the direction of the IRSL2 and IRSL1 pins. 0 = Pin direction input 1 = Pin direction output
2-0	**Reserved**

Table 4.195 Infrared Receiver Input Selection

IRSL0_DS[a]	IRRX_MD[b]	AUX_IRRX[c]	HIS_IR[d]	IRRXn
0	0	0	x	IRRX1
0	0	1	x	IRRX2
0	1	x	0	IRRX1
0	1	x	1	IRRX2
1	0	0	x	IRRX1
1	0	1	x	Reserved
1	1	x	0	IRRX1
1	1	x	1	Reserved

a. IRSL0_DS (bit 5 of IRCF4) in ‘IR Interface Configuration Register 4 (IRCFG4)’ on 395.
b. IRRX_MD (bit 6 of IRCF4) in ‘IR Interface Configuration Register 4 (IRCFG4)’ on 395.
c. AUX_IRRX (bit 4 of IRCR2) in ‘IR Control Register 2 (IRCR2)’ on 380.
d. HIS_IR = 1 When the selected mode is MIR or FIR.

Table 4.196 Bank 7 Bitmap

<table>
<tr><th colspan="2">Register</th><th colspan="8">Bits</th></tr>
<tr><th>Offset</th><th>Mnemonic</th><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td>00h</td><td>IRRXDC</td><td colspan="3">DBW2-0</td><td colspan="5">DFR4-0</td></tr>
<tr><td>01h</td><td>IRTXMC</td><td colspan="3">MCPW2-0</td><td colspan="5">MCFR4-0</td></tr>
<tr><td>02h</td><td>RCCFG</td><td>R_LEN</td><td>T_OV</td><td>RXHSC</td><td>RCDM_DS</td><td>Reserved</td><td>TXHSC</td><td colspan="2">RC_MMD1, 0</td></tr>
<tr><td>03h</td><td>BSR</td><td>BKSE</td><td colspan="7">BSR6-0</td></tr>
<tr><td>04h</td><td>IRCFG1</td><td>STRV_MS</td><td colspan="3">SIRC2-0</td><td>IRID3</td><td colspan="3">IRIC2-0</td></tr>
<tr><td>05h</td><td colspan="9">Reserved</td></tr>
<tr><td>06h</td><td colspan="9">Reserved</td></tr>
<tr><td>07h</td><td>IRCFG4</td><td>Reserved</td><td>IRRX_MD</td><td>IRSL0_DS</td><td>RXINV</td><td>IRSL12_DS</td><td colspan="3">Reserved</td></tr>
</table>

5. ZF-Logic and Clocking

The ZF-Logic module (ZFL) is a collection of additional functions for the ZFx86. ZFL is connected internally to ISA bus. ZFL uses dedicated IO pads on ZFx86 to control external devices. There is interconnect between the bootstrap register bits and the system clocking setup.

Chapter Quick Reference

5.1. Features

The features of the ZFL is shown in Figure 5-1 ZF-Logic Features. These features include:

- ZFL Register Set in ISA I/O Space
- Programmable PWM generator
- Programmable Watchdog timer
- ISA Memory Mapper for Flash/SRAM
- ISA I/O Mapper General Purpose Chip Select (GPCS)
- Programmable Z-tag Interface
- Bootstrap Register (DIP switches/Pull-Ups) External Control of Boot Process
- User and BIOS Scratch Registers

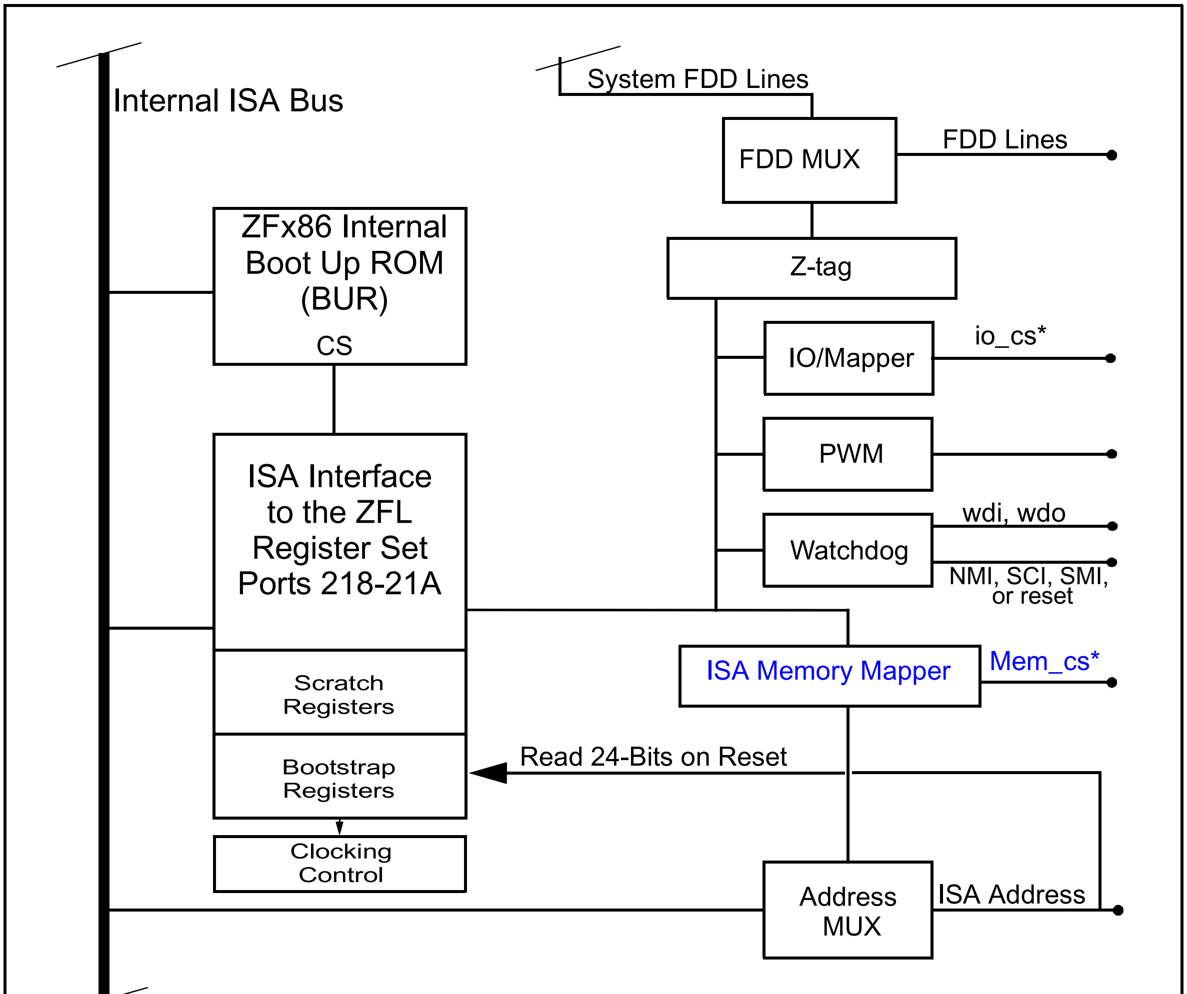


Figure 5-1 ZF-Logic Features

5.2. ZFL Register Space Summary

The ZFL register space is accessed from a 8-bit index register and 8, 16, or 32-bit data pathway located in ISA IO space. The addresses are:

Table 5.1 Access to ZFL

Item	I/O Address
8-bit Index	218h
8-bit Data at index	219h
16-bit Data at index	21Ah
32-bit Data at index	21Ah

ZFL data registers are accessed in one of the the following ways:

A. 8-bit data transfer: Works with all indexes. Indexes single 8-bit register at index.

1. Write register number to port 218H.

2. Write the register value using data register 219H.

B. 16 and 32 bit access. Works only with even indexes. Indexes two or four consecutive data registers with single IO instruction.

1. Write register number to port 218H.

2. Read or write the register value using data register 21AH.

Example: Read the 16-Bit Data at Index 02 to pick up the ZF-Logic Revision:

The current revision, and thus the revision of this specification, is 1234H.

```
mov   al,02h     ; Index
mov   dx,218h    ; Index Address
out   dx,al      ; Set Index
mov   dx,21Ah    ; read value:
in    ax,dx      ; AX=1234H
```

Programming Caution: 16 bit access has two opportunities for the programmer to get wrong results: make attempt to use 16 bit access for odd addresses and use wrong data IO address. In the example above, we used an even (word aligned) address to do a 16-bit transfer (into AX), and we used index port 21A. This is the correct way to do it

Accessing the ZF-Logic Registers

A complete index of the ZF-Logic registers is shown in the tables below. While all registers can be accessed as single bytes, sometimes it is convenient to access them as 16-bit words (see the example on the previous page).

You may also access them as 32-bit words when appropriate. See the example in 'Fields in 32-bit memory settings register' on page 410.

Table 5.2 ZF-Logic Complete Index

Index	8-Bit Data at Index	8-Bit Data at Index + 1	
02	ZF-Logic Revision (LSB) -- (02H)	ZF- Logic Revision (MSB) -- (03H)	
04	PWM Prescaler Low Byte -- (04H)	PWM Prescaler High Byte - (05H)	PWM
06	PWM duty cycle -- (06H)		
08	PWM I/O Control -- (08H)		
0A	PWM Read Output -- (0AH)		
0C	Watchdog 1 Count Low Byte -- (0CH)	Watchdog 1 Count High Byte -- (0DH)	W/D
0E	Watchdog 2 Count Value -- (0EH)	Watchdog Reset Pulse Length -- (0FH)	
10	Watchdog Control Low -- (10H)	Watchdog Control High -- (11H)	
12	Watchdog Status -- (12H)		
Index	8-Bit Data at Index	8-Bit Data at Index + 1	
14	I/O Window 0 Base Low (14H)	I/O Window 0 Base High (15H)	I/O Window
16	I/O Window 0 Control (16H)		
18	I/O Window 1 Base Low (18H)	I/O Window 1 Base High (19H)	
1A	I/O Window 1 Control (1AH)		
1C	I/O Window 2 Base Low (1CH)	I/O Window 2 Base High (1DH)	
1E	I/O Window 2 Control (1EH)		
20	I/O Window 3 Base Low (20H)	I/O Window 3 Base High (21H)	
22	I/O Window 3 Control (22EH)		
24			

Table 5.2 ZF-Logic Complete Index (cont.)

Index	8-Bit Data at Index	8-Bit Data at Index + 1		
26	Memory Window 0 Base Bits 7-0	MW0 Base 15-12	MW0 Base 11-8	Memory Window
28	Memory Window 0 Base Bits 23-16	Memory Window 0 Base Bits 31-24		
2A	Memory Window 0 Size Bits 7-0	MW0 Size 15-12	MW0 Size 11-8	
2C	Memory Window 0 Size Bits 23-16	Memory Window 0 Size Bits 31-24		
2E	Memory Window 0 Page Bits 7-0	MW0 Page 15-12	MW0 Page 11-8	
30	Memory WIndow 0 Page Bits 23-16	Memory Window 0 Page Bits 31-24		
32	Memory Window 1 Base Bits 7-0	MW1 Base 15-12	MW1 Base 11-8	
34	Memory Window 1 Base Bits 23-16	Memory Window 1 Base Bits 31-24		
36	Memory Window 1 Size Bits 7-0	MW1 Size 15-12	MW1 Size 11-8	
38	Memory Window 1 Size Bits 23-16	Memory Window 1 Size Bits 31-24		
3A	Memory Window 1 Page Bits 7-0	MW1 Page 15-12	MW1 Page 11-8	
3C	Memory Window 1 Page Bits 23-16	Memory Window 1 Page Bits 31-24		
3E	Memory Window 2 Base Bits 7-0	MW2 Base 15-12	MW2 Base 11-8	
40	Memory Window 2 Base Bits 23-16	Memory Window 2 Base Bits 31-24		
42	Memory Window 2 Size Bits 7-0	MW2 Size 15-12	MW2 Size 11-8	
44	Memory Window 2 Size Bits 23-16	Memory Window 2 Size Bits 31-24		
46	Memory Window 2 Page Bits 7-0	MW2 Page 15-12	MW2 Page 11-8	
48	Memory Window 2 Page Bits 23-16	Memory Window 2 Page Bits 31-24		
4A	Memory Window 3 Base Bits 7-0	MW3 Base 15-12	MW3 Base 11-8	
4C	Memory Window 3 Base Bits 23-16	Memory Window 3 Base Bits 31-24		
4E	Memory Window 3 Size Bits 7-0	MW3 Size 15-12	MW3 Size 11-8	
50	Memory Window 3 Size Bits 23-16	Memory Window 3 Size Bits 31-24		
52	Memory Window 3 Page Bits 7-0	MW3 Page 15-12	MW3 Page 11-8	
54	Memory Window 3 Page Bits 23-16	Memory Window 3 Page Bits 31-24		

Table 5.2 ZF-Logic Complete Index (cont.)

Index	8-Bit Data at Index	8-Bit Data at Index + 1	
56		BUR Base Low (57H)	
58	BUR Base High (58H)		
5A	Memory Control Low (5AH)	Memory Control High (5BH)	
5C			
5E	Z-tag Data Write Register (5EH)		
60	Z-tag Data Read Register (60H)		
62	Bootstrap Bits 7-0 (62H)	Bootstrap Bits 15-8 (63H)	
64	Bootstrap Bits 23-16 (64H)		
66	I/O+Memory Window Map Events 66H		
68	Scratch Register 0 Low (68H)	Scratch Register 0 High (69H)	Scratch Registers
6A	Scratch Register 1 Low (6AH)	Scratch Register 1 High (6BH)	
6C	Scratch Register 2 Low (6CH)	Scratch Register 2 High (6CH)	
6E	Scratch Register 3 Low (6EH)	Scratch Register 3 High (6FH)	
70	Scratch Register 4 Low (70H)	Scratch Register 4 High (70H)	
72	Scratch Register 5 Low (72H)	Scratch Register 5 High (73H)	
74	Scratch Register 6 Low (74H)	Scratch Register 6 High (75H)	
76	Scratch Register 7 Low (76H)	Scratch Register 7 High (77H)	
78	Scratch Register 8 Low (78H)	Scratch Register 8 High (79H)	
7A	Scratch Register 9 Low (7AH)	Scratch Register 9 High (7BH)	
7C	Z-tag control register (7CH)	Z-tag Sequencer Divisor Register (7DH)	
7E	Z-tag Sequencer Waveform (7EH)	Z-tag Sequencer Strobe Points (7FH)	
80	Z-tag Sequencer Data (80H)	Z-tag Sequencer Status (81H)	

5.2.1. Pins Associated with ZF-Logic

Listed below (for a complete reference) are all of the electrical pins associated with the ZF-Logic.

You may cross refererence to ‘Pin Descriptions Sorted by Pin’ on page 537.

Table 5.3 ZF-Logic Pin List

B03	io_cs0	ZF-Logic I/O Mapper GPCS 0
A02	io_cs1	ZF-Logic I/O Mapper GPCS 1
A01	io_cs2	ZF-Logic I/O Mapper GPCS 2
C03	io_cs3	ZF-Logic I/O Mapper GPCS 3
B04	Mem_cs0	ZF-Logic Memory Mapper CS 0[a]
D05	Mem_cs1	ZF-Logic Memory Mapper CS 1[a.]
A03	Mem_cs2	ZF-Logic Memory Mapper CS 2[a.]
C04	Mem_cs3	ZF-Logic Memory Mapper CS 3[a.]
B05	Pwm	ZF-Logic PWM Output
AC01	sa[00]	ISA Address/Bootstrap Register In
AB02	sa[01]	ISA Address/Bootstrap Register In
AB01	sa[02]	ISA Address/Bootstrap Register In
AA03	sa[03]	ISA Address/Bootstrap Register In
AA02	sa[04]	ISA Address/Bootstrap Register In
Y04	sa[05]	ISA Address/Bootstrap Register In
AA01	sa[06]	ISA Address/Bootstrap Register In
Y02	sa[07]	ISA Address/Bootstrap Register In
Y03	sa[08]	ISA Address/Bootstrap Register In
Y01	sa[09]	ISA Address/Bootstrap Register In
W03	sa[10]	ISA Address/Bootstrap Register In
W02	sa[11]	ISA Address/Bootstrap Register In
W01	sa[12]	ISA Address/Bootstrap Register In
V03	sa[13]	ISA Address/Bootstrap Register In
V04	sa[14]	ISA Address/Bootstrap Register In
V02	sa[15]	ISA Address/Bootstrap Register In
V01	sa[16]	ISA Address/Bootstrap Register In
U02	sa[17]	ISA Address/Bootstrap Register In
U03	sa[18]	ISA Address/Bootstrap Register In
U01	sa[19]	ISA Address/Bootstrap Register In

Table 5.3 ZF-Logic Pin List

T04	sa[20]	ISA Address/Bootstrap Register In
T03	sa[21]	ISA Address/Bootstrap Register In
T02	sa[22]	ISA Address/Bootstrap Register In
R03	sa[23]	ISA Address/Bootstrap Register In
A04	Wdi	ZF Logic - Watch Dog Timer
C05	Wdo	ZF Logic - Watch Dog Timer

a. When the ZF-Logic is disabled, the MEM_CS* pins remain active. However, their functions changes. Generally there is never a reason to disable the ZF-Lgoic. See 'Composite BootStrap Register Map' on page 432.

5.3. ISA Memory Mapper for Flash/SRAM

The ZFL allows the ZFx86 to control up to four external memory devices on the ISA bus. These devices can be mapped into the system memory address space. Typically, this feature is used to map external Flash memory into the ISA address space without external address decoding logic.

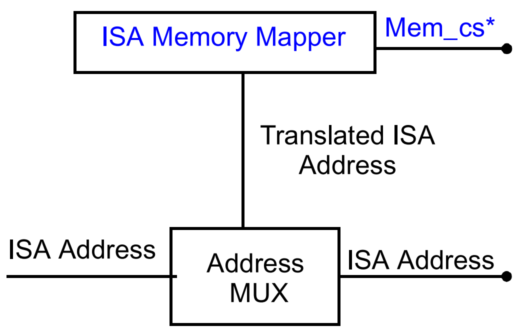


Table 5.4 Memory Mapper Pins

PKG	Name	Description
B04	Mem_cs0	ZF-Logic Memory Mapper CS 0
D05	Mem_cs1	ZF-Logic Memory Mapper CS 1
A03	Mem_cs2	ZF-Logic Memory Mapper CS 2
C04	Mem_cs3	ZF-Logic Memory Mapper CS 3

These devices are connected externally to ISA BUS *address*, *data* and *memr*/*memw* signals. Each device has a dedicated chip select signal generated by ZFL. Each device can occupy up to 16 Megabytes (occupying all 24 ISA address lines). The minimum window size is 8Kbyte.

These devices are mapped into the system memory space and accessed through windows (memory view ports) in the memory space. In DOS mode these windows can be up to 256K bytes and reside only in upper 1 Mbyte DOS ROM area (C0000-FFFFF). In protected mode the windows can occupy all 24 ISA address lines (000000 - FFFFFF). This area is accessed in protected mode through memory space above system SDRAM. If the address is not in the system memory and no PCI device claims it then it is forwarded to the ISA bus. This makes the ISA bus appear multiple times in the upper memory area.

A separate window (memory viewport) can be defined for each device with the following parameters:

- Window Size
- Base address
- Page

All parameters are aligned with 4 KB increments in system memory.Access mode (rd_only / wr) can be changed from the window control register (see Table 5.12, 'Memory Control Low -- Index 5AH,' on page 407). The Memory Chip Select lines (Mem_cs*) are always active low.

The BASE and SIZE are the window in the ZFx86 ISA address space, and the PAGE is a relative value (+ or - from BASE) used to calculate the target address in the flash.

Memory page registers are translated on-the-fly using ISA bus address lines. If an active mapping window is accessed then the FLASH page register is added to ISA upper address lines. The ISA address lines in ZFx86 are stable during entire memory cycle and thus FLASH can be directly connected to ISA bus signals.

A memory windows **overlap** event is generated if two or more mem_cs* signals are active at the same time, i.e. when memory windows do overlap in memory space. This event can be routed to NMI, SCI or SMI. Access to overlapping area will not cause any of the mem_cs* to became active.

Window 0 has special power up initialization.

Table 5.5 Indices For Memory Windows

Function	0	1	2	3	Reference	Description
Base Low	27H	33H	3FH	4BH	Table 5.6 on p. 405	base bits 15:12 (nibble 3)
Base High	28H	34H	40H	4CH	Table 5.7 on p. 406	base bits 23:16 (nibbles 5-4)
Size Low	2BH	37H	43H	4FH	Table 5.8 on p. 406	page size nibble 3
Size High	2CH	38H	44H	50H	Table 5.9 on p. 406	page size nibbles 5 - 4
Page Low	2FH	3BH	47H	53H	Table 5.10 on p. 406	page nibble 3
Page High	30H	3CH	48H	54H	Table 5.11 on p. 406	page nibbles 5 - 4
Control Low	5AH				Table 5.12 on p. 407	read/write control
Control High	5BH				Table 5.13 on p. 407	address decoding for boot
Status	66H				Table 5.14 on p. 408	Memory (and I/O) Status

Table 5.6 Memory Window "N" Base Low - Bits 15:12 (nibble 3)

Bit	7	6	5	4	3	2	1	0
Function	d15	d14	d13	d12	reserved			
Default	0	0	0	0	0			
R/W	R/W	R/W	R/W	R/W	R/O			

Table 5.7 Memory Window "N" Base High - Bits 23:16 (nibbles 5-4)

Bit	7	6	5	4	3	2	1	0
Function	d23	d22	d21	d20	d19	d18	d17	d16
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 5.8 Memory Window "N" Size Low - (nibble 3)

Bit	7	6	5	4	3	2	1	0
Function	d15	d14	d13	d12	reserved	reserved	reserved	reserved
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/O	R/O	R/O	R/O
Programming Notes for Memory Window Size: Setting the memory window size to zero disables the memory window.								

Table 5.9 Memory Window "N" Size High - (nibbles 5-4)

Bit	7	6	5	4	3	2	1	0
Function	d23	d22	d21	d20	d19	d18	d17	d16
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 5.10 Memory Window "N' Page Low - (nibble 3)

Bit	7	6	5	4	3	2	1	0
Function	d15	d14	d13	d12	reserved	reserved	reserved	reserved
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/O	R/O	R/O	R/O

Table 5.11 Memory Window "N" Page High - (nibbles 5-4)

Bit	7	6	5	4	3	2	1	0
Function	d23	d22	d21	d20	d19	d18	d17	d16
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Table 5.12 Memory Control Low -- Index 5AH

Bit	7	6	5	4	3	2	1	0
Function	w3_ro	w2_ro	w1_ro	w0_ro	w3_8	w2_8	w1_8	w0_8
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Note: The default value for this register bit 0 is dependent on bootstrap jumper setting, connected to the ZFx86 SA12 line. This can be overridden writing new value to the memory window control register. Setting the memory window to read-only mode disables both, the MEMW_N and SMEMW_N signals on ISA bus for the according memory window region.								

Bit	Name	Function
7:4	wn_ro	Window n Read-write Control 0: Window N Is Read-write 1: Window N Is Read-only
3:0	wn_8	Window n Data Bus Width 0 = Window N Uses 16-bit Data Access 1 = Window N Uses 8-bit Data Access

Table 5.13 Memory Control High -- Index 5BH

Bit	7	6	5	4	3	2	1	0
Function	reserved							full ISA
Default	0							0
R/W	R/O							R/W

Bit	Name	Function
7:1	Reserved	
0	full ISA	Masks address bits 23:20 out of comparison. This is used for boot to fetch data from ROM 0 = Enable only bits 19:0 in address calculations 1 = Enable full ISA 23:0 in address calculations

Table 5.14 I/O and Memory Window Mapper Events -- Index 66H

Bit	7	6	5	4	3	2	1	0
Function	Reserved		Event Type		Memory Overlap	I/O Overlap	memory window change	I/O window change
Default	0		0		0	0	0	0
R/W	R/O		R/W		R/W	R/W	R/W	R/W

Bit	Name	Function
7:6	Reserved	
5:4	Event Type	Generated event type 00 = No event 01 = SCI 10 = NMI 11 = SMI
3	Memory Overlap	Enable resolve event on memory overlap 0 = Disable event on memory overlap 1 = Enable event on memory overlap
2	I/O Overlap	Enable event on I/O window overlap 0 = Disable event on I/O window overlap 1 = Enable event on I/O window overlap
1	Memory Access	Enable event on memory window change 0= Disable event 1 = Enable event
0	I/O Access	Enable event on I/O window change 0 = Disable event 1 = Enable event

It is only possible to boot from these sources:

- External 8 bit ROM/FLASH , mem_cs0
- External 16-bit ROM/FLASH, mem_cs0, bootstrap 12 = 0
- External BUR, always 8-bit, gpio0, bootstrap 11 = 0. See BS11 in Table 5.42.
- Internal BUR, always 8-bit

The selection between BUR and ROM/FLASH is done according to bootstrap bit 23:

0 = boot from ROM/FLASH

1 = boot from BUR

The boot starts from the first code fetch at address FFFFFFF0 which is translated to ISA space FFFFF0 (upper 8 bits truncated).

The internal decoder sees only the first megabyte of it (the FULL_ISA bit in ZF reg space) -FFFF0. The chip select is programmed to respond to the addesses in the range F0000 – FFFFF or the last 64 Kbytes.

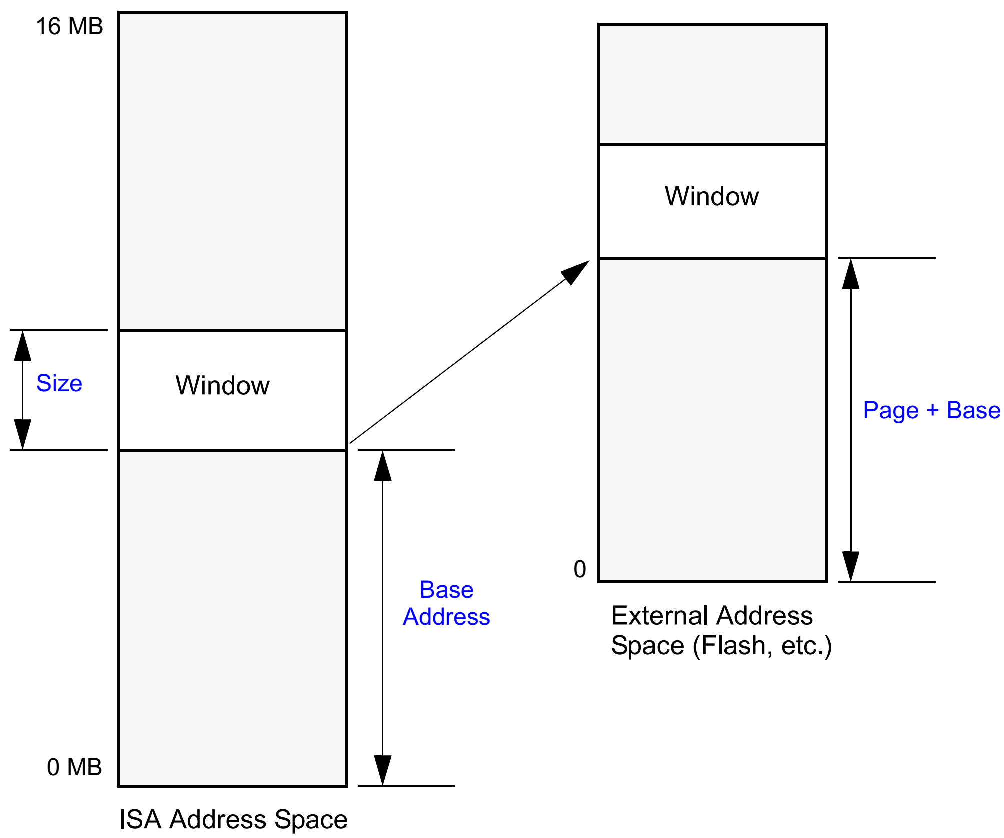


Figure 5-2 Memory Window Mapping

5.3.1. Window settings registers

The *starting address*, *window size* and *page* define the mapping for each of the four external memory devices. The mapping is disabled when *window size* is zero.

The register map of Memory decode area is shown in Table 5.5, 'Indices For Memory Windows,' on page 405 Window 0 has special power up initialization. See "Initialization of mem_cs0" on page 412.,

The starting address, window size and page registers are each 32-bit registers. Each of these consumes four 8-bit registers in ZFL register space. Window size, location and FLASH page can be set in 4KB increments (as the last 12 bits are implicitly 0). ISA BUS limits the address space to 24 bits, so the top 8 bits are implicitly 0. The layout of the memory settings register is shown below. The lower 12 bits must be zero to comply with 4KB increments and the upper 8 bits are 0 because the ISA bus has only 24 address lines.

When determining whether or not an addresses emitted by the processor should generate a mem_cs*, the upper 8-bits of the 32-bit memory address are ignored. Thus a memory page can appear (alias) many places in the (CPU's) 4 GB address space.

5.3.1.1. Page Register Calculations

Note, that page addresses are always ADDED to the actual address, so in order to access flash chip at address smaller than memory window location, the page register value must cause roll-over at 16Mbyte boundary. The simple rule of the thumb for calculating desired page register value is:

Page_register = (16M-memory_window_base) + desired_flash_offset

Therefore, in order to map flash offset 0000h for memory window, located at E0000h, the correct formula is:

Page_register = (1000000h-E0000h)+0 = F20000h

For another example, see 5.3.5. "Sample Code for Memory Window Calculation" on page 415.

5.3.1.2. Size Register

The right-most 12 bits of the Size Register are implicitly 1's. So F000H is really FFFFH. The size actually represents the maximum address from memory window base that will be decoded as window, so the window size 0FFFFH (actually 0FxxxH) will result 64 Kbyte window, and 10000H (actually 10xxxH) will result in a 68 Kbyte window.

5.3.1.3. Base Register

The memory window content is visible using two different access methods:

a) Memory window can be located in upper portion of the first 1MB of the address space at addresses (C0000-FFFFF). This window is directly visible as far as shadow memory is not mapped to that area. (See Memory Holes following).

b) Because of the ISA-PCI address mapping, the same memory window will roll over at every 16Mbyte on entire 4 Gbyte PCI memory space as far as there is no SDRAM at this area and no PCI device claims it. For example, if 5 Mbyte memory window is created in memory area 100000h-600000h, the same memory window content can be seen and accessed at regions 1100000h-1600000h, 2100000h-2600000h etc. This allows the access of the memory window content despite of the SDRAM presence, since memory window accesses can always happen above first 256 Mbyte.

There are two 8-bit registers each for Window Size, Base and Page. In each case, the two 8-bit registers supply bits 15-12 and bits 23-16 respectively. You may access these registers as a 32-bit operand. That is, you do a 32-bit in or out to the lowest of the 4 addresses. See the programming example following.

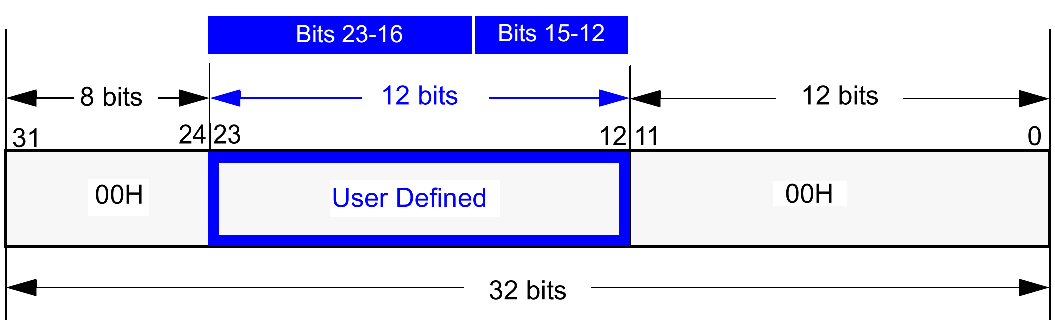


Figure 5-3 Fields in 32-bit memory settings register

5.3.1.4. Viewing Memory Holes

The DRAM address space is potentially the first 265 MB of the address space (depending upon the size of the DRAM). Table 3.20, 'Shadow RAM Read Enable Control Register (SHADRC),' on page 144, and Table 3.21, 'Shadow RAM Write Enable Control Register,' on page 145, describe the allocation of DRAM memory between C0000 and F0000. DRAM which not enabled for write and not enabled for read represent a memory hole. DRAM addresses in that range propagate to the ISA address bus and can be "intercepted" by memory windows.The printout below is from a DOS application program run on ZFx86 Integrated Development System after boot using the Phoenix BIOS. This tells you which addresses in the range C0000 to FFFFF are available for windows. In this case addresses from C0000 to C7FFF (32K) are taken up by the video BIOS on the Video Card.

```
#define NBIndex      0x24     // North Bridge Configuration Index Register
#define NBData       0x26     // North Bridge Configuration Data Register
#define RID          0x100    // North Bridge RevisionID Register
#define SHADRC       0x200    // Shadow RAM Read Enable Control Register
#define SHADWC       0X201    // Shadow RAM Write Enable Control Register

void displayNBConfiguration (void)
{
unsigned long uii;
printf ("North Bridge RevisionID Register = %04XH\r\n", uiGetNB (RID));
printf ("North Bridge Shadow RAM Read Enable Control Register = %04XH\r\n", uiGetNB (SHADRC));
printf ("North Bridge Shadow RAM Write Enable Control Register = %04XH\r\n", uiGetNB (SHADWC));
// figure out which pages are available for memory mapping  we need 0 in both --
uii = uiGetNB (SHADRC) |  uiGetNB (SHADWC);
uii =  uii ^ 0xFFFF;  // toggle all the bits - 1 means that both RD and WR disabled

if ( (uii & BIT0  ) == BIT0  ) printf ("Window OK at C0000-C3FFFH \r\n");
if ( (uii & BIT1  ) == BIT1  ) printf ("Window OK at C4000-C7FFFH \r\n");
if ( (uii & BIT2  ) == BIT2  ) printf ("Window OK at C8000-CBFFFH \r\n");
if ( (uii & BIT3  ) == BIT3  ) printf ("Window OK at CC000-CFFFFH \r\n");
if ( (uii & BIT4  ) == BIT4  ) printf ("Window OK at D0000-D3FFFH \r\n");
if ( (uii & BIT5  ) == BIT5  ) printf ("Window OK at D4000-D7FFFH \r\n");
if ( (uii & BIT6  ) == BIT6  ) printf ("Window OK at D8000-DBFFFH \r\n");
if ( (uii & BIT7  ) == BIT7  ) printf ("Window OK at DC000-DFFFFH \r\n");
if ( (uii & BIT8  ) == BIT8  ) printf ("Window OK at E0000-E3FFFH \r\n");
if ( (uii & BIT9  ) == BIT9  ) printf ("Window OK at E4000-E7FFFH \r\n");
if ( (uii & BIT10 ) == BIT10 ) printf ("Window OK at E8000-EBFF
if ( (uii & BIT11 ) == BIT11 ) printf ("Window OK at EC000-EFFF
if ( (uii & BIT12 ) == BIT12 ) printf ("Window OK at F0000-F3FF
if ( (uii & BIT13 ) == BIT13 ) printf ("Window OK at F4000-F7FF
if ( (uii & BIT14 ) == BIT14 ) printf ("Window OK at F8000-FBFF
if ( (uii & BIT15 ) == BIT15 ) printf ("Window OK at FC000-FFFF
}
unsigned int uiGetNB (unsigned int uiNBItem) {
   outpw (NBIndex, uiNBItem);
   return inpw (NBData); }
```

```
Window OK at C8000-CBFFFH
Window OK at CC000-CFFFFH
Window OK at D0000-D3FFFH
Window OK at D4000-D7FFFH
Window OK at D8000-DBFFFH
Window OK at DC000-DFFFFH
```

5.3.2. Control (R/W, 8/16)

The control register is common for all four memory devices. The first four bits control the data width of external memory device. The lower four bits set the write enable mask to write protect memory. See Table 5.13, 'Memory Control High -- Index 5BH,' on page 407.

5.3.3. Events (SMI, etc.)

The mem_cs* is asserted when it does not overlap with others in system RAM. This run-time test is executed every time the memory mapping registers are accessed. The events register (Table 5.14, 'I/O and Memory Window Mapper Events -- Index 66H,' on page 408) allows control of the generated event (SCI, NMI, etc.) based on Window Change or Window Overlap.

5.3.4. Initialization of mem_cs0

The ZFx86 is designed to boot from an external ROM/FLASH chip using mem_cs0. In order to do this, the ZFx86 pre-loads the Base, Size and Page registers for mem_cs0 on power up. It also sets the width bit based on your setting of SA12 (see 'Boot Parameters Register' on page 431). The mem_cs0 registers are set up such that the top 64K of the external Flash Chip will be accessed.

The exact boot up sequence of an X86 processor is well known -- but the addition of the ZFx86 ISA Memory Mapper makes the process a bit tricky.

The default initialization provides a 64K window starting at F000H in the ZFx86.

Default Power-Up Initialization of mem_cs0 (window 0) is done by the ZFx86 hardware to boot from flash using window 0:

- Base address = F0000H
- Window size = 64K-1 (F000H)
- Width = dependent on bootstrap at SA12
- Page = 00000h

It's essential to set the mem_cs0 page to 00F00000h before the first FAR jump in BIOS. Otherwise the boot fails with flash chips bigger than 1MB, since upper addresses are cleared from the ISA bus after a far jump and the next memory reads (instruction fetches) go to the last 64K of the first 1M of the flash chip.

Mem_cs0 Programming Example

NOTE: It's essential to set the mem_cs0 page address to 00F00000h before the first FAR jump in BIOS for flash chips larger than 1 Mbyte. Otherwise the POST fails, since upper addresses are cleared from the ISA bus after the first far jump and subsequent memory reads appear at the first megabyte of the chip, instead of the last megabyte.

```
; set page address to access last MB inside flash

mov al, 2Eh ; Memwindow 0 page register 32-bit access
mov dx, 218h ; ZFL IDX register
out dx, al ; Select memwin 0 base
add dx, 2 ; seek to 32-bit data register
mov eax, 00F00000h ; set page to be 16Mbyte-1Mbyte
out dx, eax ; set page (not needed see below)

; Following near jump is mandatory
jmp $+2  ; refill CPU pipeline
comment * Thanks to code already fetched into CPU pipeline we  do not crash after next OUT instruc-
tion. However, after  that we can not make another code fetch from ISA device  until we make the
first FAR jump that clears upper address bits bus. *
out dx,eax

; Nothing must come between OUT and FAR JMP or we don’t fit
; inside pipeline (16 bytes) any more.

db  0EAh  ; JMP FAR absolute
dw  offset FirstFarJmp
dw  0F000h
FirstFarJmp:

comment * We are all right now for whatever chip size, because we always see the last megabyte of
the chip from now on.  If we need a bigger window to flash (because of the 256K BIOS image per-
haps), we can enable it here and now, before proceeding with actual POST *

IF (WINDOW_256K) ; set window size to 256K now

mov al, 2Ah
dec dx
dec dx
out dx, al
inc dx
inc dx
mov eax, 40000h-1 ; window size to 256K
out dx, eax

; drop window base address to C0000h

mov al, 26h
dec dx
dec dx
out dx, al
inc dx
inc dx
mov eax, 0C0000h
out dx, eax

ENDIF ;(WINDOW_256K)

; Jump now whenever it was necessary in a first place

    db 0EAh     ; JMP FAR absolute
FJum_Offset dw 0h ; offset into BIOS
```

```
FJump_Segment dw 0h ; segment into BIOS
```

5.3.5. Sample Code for Memory Window Calculation

Here we will create a **32K** window starting at **D0000** in the ZFx86 address space. The window initially point to **offset 2000H** within the flash chip.

Example: You would have to set the Page register to 00F32000H (or F32 in the ZFx86 Phoenix BIOS Memory Window Setup Screen).

Phoenix ZFx86 BIOS Memory Window Setup Calculator

Helper: Enter Window Size between 8 and 16384K: 32

Recommended Value for Size = 007000H

Enter Desired Window Base in Hex (example DC000) **D0000**

Enter Desired Window Size in Hex (example 1000 = 8K, FFF000 = 16 MB **7000**

```
unsigned long ulBase, ulSize, ulTarget, ulPage, ulDesiredK;
    printf ("\r\n\n\nPhoenix ZFx86 BIOS Memory Window Setup Calculator \r\n", uiWorkingCS);

    printf ("\r\nHelper: Enter Window Size between 8 and 16384K: ");
    ulDesiredK = uiScanInDecimal(0);

    if ((ulDesiredK >= 8) && (ulDesiredK <= 16384))
         printf ("\r\n Recommended Value for ""Size"" = %06lXH\r\n", (ulDesiredK-4) * 1024 );

    printf ("\r\nEnter Desired Window Base in Hex (example DC000) ");
            ulBase = ulScanInHex ();
    printf ("\r\nEnter Desired Window Size in Hex (example 1000 = 8K, FFF000 = 16 MB ");
            ulSize = ulScanInHex ();
            printf ("\r\nEnter Flash Target Address in Hex (example 2000 = 8K) ");
            ulTarget = ulScanInHex ();
             ulBase = ulBase & 0xFFF000;
            ulSize = ulSize & 0xFFF000 ;
            ulTarget = ulTarget & 0xFFF000;
            printf ("\r\nBase = %08lXH or %03lXH or %04ld decimal", ulBase, ulBase >> 12, ulBase
>>12);
            if (ulSize == 0) printf ("\r\nWindow Disabled Size == 0");
            else
            printf ("\r\nSize = %08lXH or %03lXH or %04ld decimal for Size = %dK", ulSize, ulSize
>> 12, ulSize >>12, ((ulSize >> 12) + 1) * 4);
            ulPage = (0x1000000 - ulBase + ulTarget) & 0xFFF000;
            printf ("\r\nPage = %08lXH or %03lXH or %04ld decimal for Flash Off set =
%08lXH\r\n\n",
                        ulPage, ulPage >> 12, ulPage >>12, ulTarget);
```

5.4. GPCS I/O mapper

ZFx86 has four GPCS (General Purpose Chip Select) signals mapped to io_cs* pins. GPCS signals can be used to connect external devices to ISA I/O space without external address decode logic. Each io_cs* signal is assigned an address (or a set of consecutive addresses) in the ISA I/O space. This address, or set of consecutive addresses, is called the “window”. The window can cover 16 byte ports or 8 word ports.

For example, if the chip you wish to connect to the ZFx86 using one of the io_cs pins has four ports (such as the old 8255 chip), you would want the chip select to be asserted for four consecutive addresses. The chip itself would differentiate between the addresses by looking at the low two bits of the ISA address bus.

.

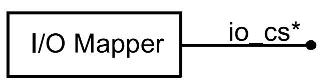


Table 5.15 GPCS Pins

PKG	Name	Description
B03	io_cs0	ZF-Logic I/O Mapper GPCS 0
A02	io_cs1	ZF-Logic I/O Mapper GPCS 1
A01	io_cs2	ZF-Logic I/O Mapper GPCS 2
C03	io_cs3	ZF-Logic I/O Mapper GPCS 3

I/O ranges of different GPCS signals can not overlap. An internal check is done to assure that this condition is satisfied before enabling the io_cs* lines.

The io_cs* signal can be programmed for either 8-bit or 16-bit wide bus access: on an aligned 16 bit I/O transfer, a 16-bit wide bus would generate one I/O cycle, and an 8-bit wide bus would generate two I/O cycles.

The base address register is 16-bits wide. This allows the window to be defined anywhere in the 16-bit IO space.

The GPCS register set is shown below, and also in Table 5.16.

1. GPCS 0 settings
   - 15H-14H:io_cs0 base address
   - 16H: io_cs0 control
2. GPCS 1 settings
   - 19H-18H:io_cs0 base address
   - 1AH: io_cs0 control
3. GPCS 2 settings
   - 1DH-1CH:io_cs0 base address
   - 1EH: io_cs0 control
4. GPCS 3 settings
   - 21H-20H:io_cs0 base address
   - 22H: io_cs0 control
5. Events Register (66H) (NMI, etc.)

Table 5.16 ZF-Logic Indices For I/O Windows

Function	0	1	2	3	Reference	Description
Base Low	14H	18H	1CH	20H	Table 5.18 on page 417	base bits 7-0
Base High	15H	19H	1DH	21H	Table 5.19 on page 418	base bits 15-8
Control	16H	1AH	1EH	22H	Table 5.20 on page 420	

Table 5.17 ZF-Logic Index for I/O Windows

14	I/O Window 0 Base Low (14H)	I/O Window 0 Base High (15H)	I/O Window
16	I/O Window 0 Control (16H)		
18	I/O Window 1 Base Low (18H)	I/O Window 1 Base High (19H)	
1A	I/O Window 1 Control (1AH)		
1C	I/O Window 2 Base Low (1CH)	I/O Window 2 Base High (1DH)	
1E	I/O Window 2 Control (1EH)		
20	I/O Window 3 Base Low (20H)	I/O Window 3 Base High (21H)	
22	I/O Window 3 Control (22H)		
... **Note**: see Table 5.14, 'I/O and Memory Window Mapper Events -- Index 66H,' on page 408			
66	I/O+Memory Window Map Events (66H) Events Register		

Table 5.18 I/O Window "N" Base Low Format

Bit	7	6	5	4	3	2	1	0
Function	I/O Window 0 Base Low							
Default	0							
R/W	R/W							

Table 5.19 I/O Window "N" Base High Format

Bit	7	6	5	4	3	2	1	0
Function	I/O Window 0 Base High							
Default	0							
R/W	R/W							

Bit	Name	Function
7	win_ro	I/O window read/write control 0: = Access is read-write 1: = Access is read-only Setting window to read-only mode disables IOW_N signal on ISA bus for IO window address range.
6	16_bit	I/O window datapath width 0 = 8-bit wide access 1 = 16-bit wide access
5	act_lvl	io_cs active level 0 = io_cs is active low 1 = io_cs is active high
4	win_en	I/O window enable in I/O space 0 = I/O window is disabled 1 = I/O window is enabled
3:0	win_siz	Number of consecutive 8-bit I/O addresses to decode starting from I/O window base. The number of consecutive addresses decoded is win_siz + 1. For example, setting the window size to 0 enables one I/O address at I/O window base. Setting size to 0Fh will enable I/O window of 16 addresses starting from I/O window base. You can access a maximum of 16 byte ports or 8 word ports.

5.4.1. GPCS control

The *Active Level* (act_lvl) selects the active level of io_cs* line. The *Window Enable* (win_en) enables the decoder. The *Write Enable (win_ro)* masks the ISA I/O write signal making the window read-only. The *Window Size* (win_siz) determines the window size in the ISA IO space.

5.4.2. GPCS base low byte

This byte is located at *GPCS base* in ZFL register map. It determines the low byte of the window start address in the ISA IO space.

5.4.3. GPCS base high byte

This byte is located at *GPCS base* + 1 in ZFL register map. It determines the high byte of window start address in ISA IO space.

5.4.4. GPCS Events

See Table 5.14, 'I/O and Memory Window Mapper Events -- Index 66H,' on page 408.

5.5. Watchdog Timer

The watchdog timer is used to stabilize and recover the system in case possible failures and bugs in a program make the ZFx86 uncontrollable. Whenever the watchdog timer is not reloaded during a pre programmed interval it generates an event to notify the system of an error condition.

It works like this: The first watchdog timer is initialized to a 16-bit time-out value through registers 0Ch and 0Dh. After enabling through control register (10h) it starts the countdown to zero. The first watchdog timer can be reloaded to an initial value by writing into control register (10h) or asserting the watchdog external control pin on ZFx86 (WDI).

Whenever the first watchdog is not reloaded during the time-out value it generates an event to notify the system of an error condition and outputs the logical “1” to a watchdog output pin on ZFx86 (WDO). The notification event can be routed to NMI, SMI, SCI or it can reset the system immediately. The watchdog in ZFL consists of two timers - WD#1 and WD#2. Both timers are driven by 32 KHz clock. The maximum period for WD#1 is 2 seconds and for WD#2 7.2 ms. After WD#1 has expired it can generate NMI, SMI, SCI or RESET and start timer WD#2.

The second watchdog timer 8-bit time-out value is initialized through register 0Eh and starts counting down after WD#1 time-out. When the WD#2 counter reaches zero, it will unconditionally cause system reset.

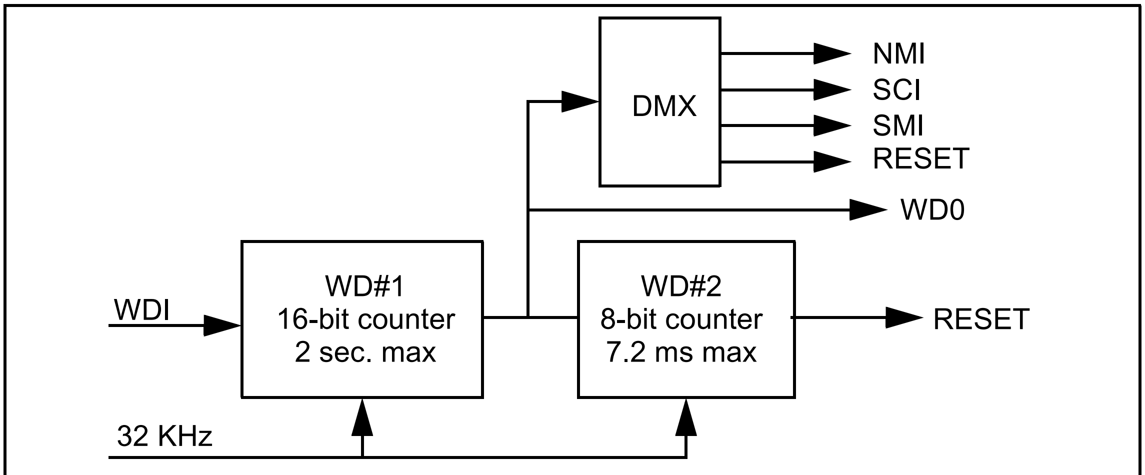


Figure 5-4 Watchdog Block Diagram

There is a WDO and WDI pin -- the WDI can be programmed to reload WD#1. If you toggle WDI (falling or rising) you can prevent the WD from ever expiring. The benefit of this is that so long as an external square wave is coming in, the WD never expires. You are thus using an EXTERNAL way of keeping the ZFx86 from resetting -- you are watching for an external “dead man” switch.

To do this, you need to have an outside event generator. Let’s assume that all you have is a logic signal which shows you if the external system is working or not. You can then connect WDO to WDI with an OR gate or AND gate to that external signal.

If you set it up this way, then you set wdo-1 to generate event 1 pulse before expiring. If you did not do this, it would expire. See wdo-1 in Table 5.6 "Memory Window “N” Base Low - Bits 15:12 (nibble 3)" on page 405

5.5.1. Watchdog Registers

Table 5.20 ZF-Logic Index for the Watchdog Timers

0C	Watchdog 1 Count Low Byte -- (0CH)	Watchdog 1 Count High Byte -- (0DH)	W/D
0E	Watchdog 2 Count Value -- (0EH)	Watchdog Reset Pulse Length -- (0FH)	
10	Watchdog Control Low -- (10H)	Watchdog Control High -- (11H)	
12	Watchdog Status -- (12H)		

Table 5.21 Watchdog 1 Count Low Byte -- Index 0CH

Bit	7	6	5	4	3	2	1	0
Function	Watchdog 1 count low byte							
Default	0							
R/W	R/W							

Table 5.22 Watchdog 1 Count High Byte -- Index 0DH

Bit	7	6	5	4	3	2	1	0
Function	Watchdog 1 Count High Byte							
Default	0							
R/W	R/W							

Example: This example illustrates reading the counter as a single 16-bit register, and writing back an incremented value:

```
MOV  AL,  0CH
MOV  DX,  218H
OUT  DX,  AL

MOV  DX,  21AH
IN   AX,  DX   ; Get Counter Value
INC  DX        ; Increment Data
OUT  DX,  AX   ; Write Back
```

Table 5.23 Watchdog Generated Reset Pulse Length -- Index 0FH

Bit	7	6	5	4	3	2	1	0
Function	Reset Pulse Length							
Default	0							
R/W	R/W							
Programming Notes: When WD#1 is programmed to cause system reset or when WD#2 resets the system, this register is used to determine the number of 32kHz ticks to hold the system reset signal low.								

Table 5.24 Watchdog Control Low -- Index 10H

Bit	7	6	5	4	3	2	1	0
Function	reserved		wd2 load	wd1 load	reserved		wd2 enable	wd1 enable
Default	0		0	0	0		0	0
R/W	R/O		R/W	R/W	R/O		R/W	R/W

Bit	Name	Function
7:6	Reserved	
5	wd2 load	Reload WD#2 counter. Active event for this bit is transition from 0 to 1
4	wd1 load	Reload WD#1 counter. Active event for this bit is transition from 0 to 1
3:2	Reserved	
1	wd2 enable	Enable WD#2 0 = WD#2 is disabled 1 = WD#2 is enabled
0	wd1 enable	Enable WD#1 0 = WD#1 is disabled 1= WD#1 is enabled

Table 5.25 Watchdog Control High -- Index 11H

Bit	7	6	5	4	3	2	1	0
Function	reserved	wdi_en	wdo_-1	wdi edge	wd1 reset	wd1 SMI	wd1 NMI	wd1 SCI
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7	Reserved	
6	wdi_en	Enable the assertion of WDI input pin on ZFx86 to reload the watchdog 1 counter 0 = WDI input ignored 1= WDI assertion reloads watchdog 1 counter
5	wdo_-1	Create output on WDO output pin on ZFx86 at WD#1 time-out or one 32kHz clock tick before 0 = WDO signal will be set high on WD#1 expiration 1 = WDO signal is set high one clock tick before WD#1 expires. WD#1 events will always occur at WD#1 time-out and are not affected of wdo_-1 bit setting. This feature allows automatic reload of WD#1 when WDO is wired to WDI.
4	wdi edge	Active front of WDI input 0 = WDI is asserted on 0->1 transition 1 = WDI is asserted on 1->0 transition
3	wd1 reset	WD#1 generates System Reset on time-out 0 = WD#1 will not generate system reset 1 = WD#1 will generate system reset on time-out
2	wd1 SMI	WD#1 generates SMI on time-out 0= WD#1 will not generate SMI 1 = WD#1 generates SMI on time-out
1	wd1 NMI	WD#1 generates NMI on time-out 0 = WD#1 will not generate NMI on time-out 1 = WD#1 generates NMI on time-out
0	wd1 SCI	WD#1 generates SCI on time-out 0 = WD#1 will not generate SCI on time-out 1 = WD#1 generates SCI on time-out

Table 5.26 Watchdog Status -- Index 12H

Bit	7	6	5	4	3	2	1	0
Function	wd1_gn	wd1_gc	wd1_gs	wd1 reset	wd2 reset	wd1_ev	reserved	reserved
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/O	R/O

Bit	Name	Function
7	wd1_gn	WD#1 generated NMI 0: WD#1 did not generate NMI 1: WD#1 did generate NMI
6	wd1_gc	WD#1 generated SCI 0: WD#1 did not generate SCI 1: WD#1 did generate SCI
5	wd1_gs	WD#1 generated SMI 0: WD#1 did not generate SMI 1: WD#1 did generate SMI
4	wd1 reset	WD#1 caused system reset 0: WD#1 did not reset the system 1: WD#1 did reset the system
3	wd2 reset	WD#2 caused system reset 0: WD#2 did not reset the system 1: WD#2 did reset the system
2	wd1_ev	WDI input pin asserted 0: WDI input has not been asserted 1: WDI input has been asserted
1:0	Reserved	
Programming Notes: This register is set each time any of the watchdog events occurs. The register is NOT reset by POR (Power On Reset) signal of ZFx86. The register can be cleared by writing anything into it. Any write to watchdog status register (12h) will reset all the register bits to default values (00h).		

5.6. PWM generator

The PWM (Pulse Width Modulation) output may be used to create DC control voltage for an LCD backlight or any other device that requires this feature. The conversion is done by integrating variable duty cycle signal externally. At higher frequencies it may be used to control external transformer for DC/DC conversion.

The clock signal derives the first timer. This 16-bit timer (prescaler) sets PWM period. The input clock can be set to the (generally) 8MHz ISA bus clock (see "ISACLK_N" on page 178) or 32kHz clock. The prescaler resets the PWM output to low and clocks second 8-bit timer what starts counting from zero. When it reaches the reference count given in 8-bit PWM duty cycle register (06h), the comparator sets PWM output value to high. This process repeats forever. By changing the comparator reference value it is possible to generate different duty cycles. From the control register it is also possible to control the signal value on PWM output pin directly.

The precision is 1/(2^8)=1/256 or approximately 0.5%

The pulse width generator (PWM) generator is controlled using four 8-bit registers in ZFL.

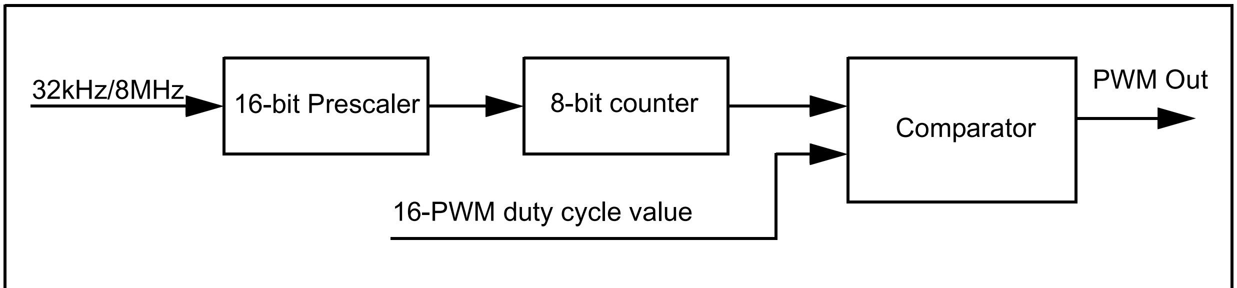


Figure 5-5 PWM Control Unit

Table 5.27 ZF-Logic Index for the PWM Generator

04	PWM Prescaler Low Byte -- (04H)	PWM Prescaler High Byte - (05H)	PWM
06	PWM duty cycle -- (06H)		
08	PWM I/O Control -- (08H)		
0A	PWM Read Output -- (0AH)		

Table 5.28 PWM Prescaler Low Byte - Index 04H

Bit	7	6	5	4	3	2	1	0
Function	PWM Master Prescaler Low Byte							
Default	0							
R/W	R/W							
16-bit PWM prescaler divisor word low byte. Divides ISACLK_N or 32kHz input clock selected at PWM control register. Actual divisor is 16-bit PWM divisor word (combined of registers 04h and 05h) + 1								

Table 5.29 PWM Prescaler High Byte - Index 05h

Bit	7	6	5	4	3	2	1	0
Function	PWM Master Prescaler High Byte							
Default	0							
R/W	R/W							
16-bit PWM prescaler divisor word high byte. Divides ISACLK_N or 32kHz input clock selected at PWM control register. Actual divisor is 16-bit PWM divisor word (combined of registers 04h and 05h) + 1								

Table 5.30 PWM duty cycle - Index 06h

Bit	7	6	5	4	3	2	1	0
Function	This is the duty cycle of the PWM							
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Programming Notes: This sets the% of the cycle to be low. 0 = 100% 255 = 0%								

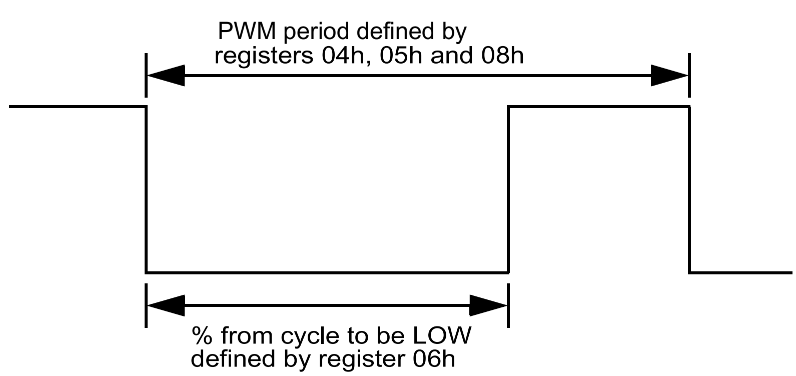


Figure 5-6 PWM Period and Duty Cycle

Table 5.31 PWM I/O Control -- Index 08H

Bit	7	6	5	4	3	2	1	0
Function	reserved		enable direct	direct out-put	reserved		slow-fast clksrc	Enable PWM
Default	0		0	0	0		0	0
R/W	R/O		R/W	R/W	R/O		R/W	R/W

Bit	Name	Function
7:6	Reserved	
5	enable direct	Enables direct control of PWM output by bit 4 0: PWM drives the output 1: Bit 4 of register 08H drives the PWM output pin
4	direct output	The value of PWM output when bit of register 08h is set to 1
3:2	Reserved	
1	slow-fast (clksrc)	Selects the PWM prescaler input clock 1: PWM is clocked by 32kHz clock 0: PWM is clocked by 8 MHz ISA clock (ISACLK_N)
0	Enable/Disable PWM	Enable/Disable PWM output 0: PWM is disabled 1: PWM is enabled

Table 5.32 PWM Read Output -- Index 0AH

Bit	7	6	5	4	3	2	1	0
Function	reserved							pwmval
Default	0							0
R/W	R/O							R/O

Bit	Name	Function
7:1	Reserved	
0	pwmval	Value at PWM output pin

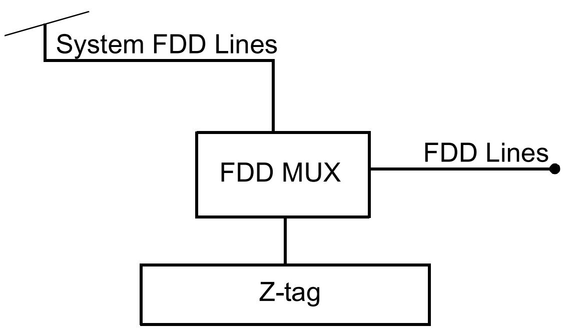
Example:

```
; initialize PWM as:
         ; prescaler = 0
         ; duty cycle = 50%
         ; ISA clocked, enabled
         ;
         ; This setup will cause PWM generator to produce output frequency
         ; 4.078kHz
         ;

                       ZFLWritew  04h,0
 0124    B0 04         mov        al,04h
 0126    BA 0218       mov        dx,ZFLINDEX
 0129    EE            out        dx,al
 012A    BA 021A       mov        dx,ZFLDATA16
 012D    B8 0000       mov        ax,0
 0130    EF            out        dx,ax

                       ZFLWriteb  06h,80h
 0131    B0 06         mov        al,06h
 0133    BA 0218       mov        dx,ZFLINDEX
 0136    EE            out        dx,al
 0137    BA 0219       mov        dx,ZFLDATA8
 013A    B0 80         mov        al,80h
 013C    EE            out        dx,al

                       ZFLWriteb  08h,03
 013D    B0 08         mov        al,08h
 013F    BA 0218       mov        dx,ZFLINDEX
 0142    EE            out        dx,al
 0143    BA 0219       mov        dx,ZFLDATA8
 0146    B0 03         mov        al,03
 0148    EE            out        dx,al

         ; Now change duty cycle to 90%. As result of this PWM output
         ; waveform will be "mostly" HIGH

                       ZFLWriteb  06h,0E8h
 0149    B0 06         mov        al,06h
 014B    BA 0218       mov        dx,ZFLINDEX
 014E    EE            out        dx,al
 014F    BA 0219       mov        dx,ZFLDATA8
 0152    B0 E8         mov        al,0E8h
 0154    EE            out        dx,al
```

5.7. Z-tag Overview

The Z-tag interface is used to read data for programming FLASH devices in system maintenance mode. Maintenance mode is entered from the internal boot ROM.

- Improves speed over using serial interface
- Frees legacy ports from system FLASH update function
- Creates a dedicated and simple interface for system upgrading



The Z-tag interface shares pins with FDD bus. The Floppy device will ignore Z-tag data when MTR0 and SEL0 signals are inactive. These lines are probed by the Z-tag dongle to enable the output buffers. Shared floppy lines in Z-tag mode are multiplexed to ZFL block. Four inputs and four outputs are used to connect external flash updating device (Z-tag dongle or other source) to the ZFx86. These lines can be accessed through dedicated register in ZFL.

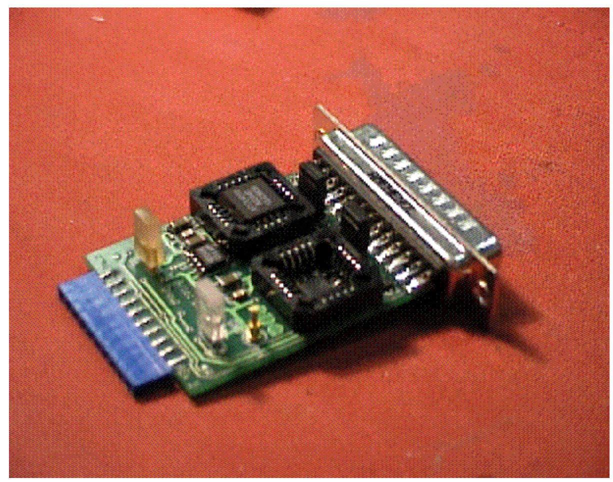


Figure 5-7 Dongle (w/o Cover)

.

Table 5.33 ZF-Logic Index for the Z-tag

5E	Z-tag Data Write Register (5EH)		Z-tag
60	Z-tag Data Read Register (60H)		
...			
7C	Z-tag control register (7CH)	Z-tag Sequencer Divisor Register (7DH)	Z-tag
7E	Z-tag Sequencer Waveform (7EH)	Z-tag Sequencer Strobe Points (7FH)	
80	Z-tag Sequencer Data (80H)	Z-tag Sequencer Status (81H)	

Table 5.34 Z-tag Data Write Register -- Index 5EH

Bit	7	6	5	4	3	2	1	0
Function	reserved				write hdsel	write data	write step	write dir
Default	0				0	0	0	0
R/W	R/O				R/W	R/W	R/W	R/W

Bit	Name	Function
7:4	Reserved	
3	write hdsel	controls the HDSEL line on FDD interface
2	write data	controls the DATA line on FDD interface
1	write step	controls the STEP line on FDD interface
0	write dir	controls the DIR line on FDD interface
Programming Notes: These bits control FDD lines if ZT_CTRL (7CH) reg bit 4 is 1		

Table 5.35 Z-tag Data Read Register -- Index 60H

Bit	7	6	5	4	3	2	1	0
Function	reserved		read drv0	read mtr0	read disc	read rdata	read wrprt	read trk0
Default	0							
R/W	R/O		R/O	R/O	R/O	R/O	R/O	R/O

Bit	Name	Function
7:6	Reserved	
5	read drv0	Monitors DRV0 line on FDD interface
4	read mtr0	Monitors MTR0 line on FDD interface
3	read dskch	Monitors DSKCH line on FDD interface
2	read rdata	Monitors RDATA line on FDD interface
1	read wrprt	Monitors WRPRT line on FDD interface
0	read trk0	Monitors TRK0 line on FDD interface

Table 5.36 Z-tag Control Register -- Index 7CH

Bit	7	6	5	4	3	2	1	0
Function	reserved				Z-tag enable	accel enable	snoop ahead	reg select
Default	0				0	0	0	0
R/W	R/O				R/W	R/W	R/W	R/W

Bit	Name	Function
7:4	Reserved	
3	Z-tag enable	Muxes the FDD lines 0: normal operation 1: Z-tag signals on FDD lines
2	Accel enable	0: Z-tag normal operation 1: Z-tag accelerator active
1	Snoop ahead	0: Clear ready flag on accel reg read 1: Do not change the ready flag on accel read
0	Reg select	0: Select the output buffer for read 1: Select the accumulator register for read

Table 5.37 Z-tag Sequencer Divisor -- Index 7DH

Bit	7	6	5	4	3	2	1	0
Function	ISA Divider							
Default	0							
R/W	R/W							
Programming Notes: ISA Divider Divides the ISA clock for sequencer input								

Table 5.38 Z-tag Sequencer Waveform -- Index 7EH

Bit	7	6	5	4	3	2	1	0
Function	Waveform programming register							
Default	0							
R/W	R/W							
Programming Notes: This register is clocked with ISA clock divided by DIVIDER register. The cyclic sequencer outputs the bits as the clock signal to Z-tag interface starting from lower bit.								

Table 5.39 Z-tag Sequencer Strobe Points -- Index 7FH

Bit	7	6	5	4	3	2	1	0
Function	Z-tag strobe							
Default	0							
R/W	R/W							
Programming Notes: 0->1 transition marks the data strobe point in Z-tag accelerator								

Table 5.40 Z-tag Sequencer Data -- Index 80H

Bit	7	6	5	4	3	2	1	0
Function	sequencer data							
Default	0							
R/W	R/O							
Programming Notes: The sequencer data is the data byte from Z-tag sequencer.								

The Z-tag interface has two operating modes selected from control register:

1. Direct software control of IO pins.

The CPU is responsible for checking the status and data bits and controlling the clock. Slow because single bit input demands at least 4 ISA IO instructions.

2. Internal FSM

Serial input device with bit-level handshaking and waveform control. This device collects the serial data from the Dongle and stores it into read buffer. It waits for acknowledge from Dongle and when the read buffer is full. With standard 8 MHz input clock it can read as much as 4 Mbit/sec.

For additional information on Z-tag, see Chapter 6. Z-tag, BUR, and The ZFiX Console.

5.8. Boot Parameters Register

When power-on reset is asserted 24 signals are read into the Boot Parameters Register (configuration register) from the ISA Address Bus. External hardware is responsible to place data on the external ISA Address bus during the time that the power on reset line is asserted. Typically the user will provide this data via DIP switches or pull-up/pull-downs. These 24 inputs are stored into the internal power-up Boot Parameters Register (configuration register).

This 24 bit register is loaded off the ISA address pins when the ZFx86 is reset. During normal operation, these pins are output only but during reset, they are tri-state and allow an external device to drive them. During reset, these pins are also driven by weak internal pull-ups or pull-downs. If a pin is not driven externally during reset, then these pull the pin to the indicated default. In a typical design, the DIP switches will be connected such that an ON is a pull-up if the default is low, and that an OFF is a pull-down if the default is high.

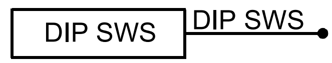
When the ZFx86 receives the reset pulse, it samples the ISA address bus. The 24 bits read in are stored in the boot parameters register.

The ISA address bus (pins SA0-SA23) is tri-stated during the reset pulse. It contains on-chip weak (about 20K) pull-ups and pull-downs to set the default state of the bootstrap register. To override this, we use a 2.2K pull-up or pull down and a DIP switch or jumper. Once the reset pulse is done, the ISA bus has sufficient drive to overcome the effect of these 2.2K resistors.

Thus, conceptually the ISA address bus has three “modes”: (1) the weak on-chip pull-ups/pull-downs which are operative during the tri-state; (2) the 2.2K pull-ups/pull-downs which may be activated via DIP switches; and (3) the normal execution time mode where the drive of the ISA address bus will override these resistors.

Since the Boot Parameters Register is read only, the values sampled on the ISA bus on the trailing edge of reset are “permanent” until the next hardware reset. Software can read the data which is latched, but cannot change the data in the bootstrap registers.

In a typical design, DIP switches or jumpers are used (with appropriate resistors) set to the bits in the BootStrap Register.



These bits are latched on reset and are read only.

Example:

; read bootstrap registers as 32-bit value into EAX

```
0102    B0 62   mov     al,     62h
0104    BA 0218mov      dx,     ZFLINDEX
0107    EE      out     dx,     al
0108    BA 021Ain       eax,    dx
010D    66| 25 00FFFFFFandeax,0FFFFFFh
```

Table 5.41 ZF-Logic Index for the Boot Parameters Register

62	Bootstrap Bits 7-0 (62H)	Bootstrap Bits 15-8 (63H)	
64	Bootstrap Bits 23-16 (64H)		

Table 5.42 Composite BootStrap Register Map

ISA BIT	ZFL Index	BS Bit	Name	Default	Function	BIOS / BUR Mapping to ZFx86 I/O Control Register (or notes)
0-3	62H	0-3	User Defined	0	User Defined	
4	62H	4	Reserved	0	South Bridge scan enabled[a.] 1 = enable scan in operational mode.	

Table 5.42 Composite BootStrap Register Map (cont.)

ISA BIT	ZFL Index	BS Bit	Name	Default	Function	BIOS / BUR Mapping to ZFx86 I/O Control Register (or notes)
5	62H	5	14 Mhz clock source	0	14MHz Clock Source If 1, derive from 48Mhz. If 0, use mhz14_c pin. [AF16]	See "IO_EXT_CLK_14M" on page 240
6	62H	6	32 KHz	0	32KHz Clock Source If 1, derive from 48MHz. If 0, use 32KHZC [AF01]	See "IO_RTC_32K" on page 239
7	62H	7	Reserved	1	486 DLL mode[a] 1 = DLL mode enabled	
8	63H	0	Reserved	1	486 raw clock mode[a.] 1 = raw clock disable.	
9	63H	1	3rd PCI Request	0	Third PCI Request/Grant 1 = drq1 = req2_n and dack1_n = gnt2_n	Shared I/O Pins switched to DMA or PCI. See 5.8.1.1. "Multiplexing of Pins A14 - B14" on page 434.
10	63H	2	Reserved	0	SIO Test Mode [a.]	
11	63H	3	Reserved	1	Internal / External BUR Source[a.] 0 = External BUR 1 = Internal BUR	External BUR uses GPIO0 as chip select.[b]
12	63H	4	ISA Boot ROM Width	1	ISA Boot ROM Width 0 = 16 bit 1 = 8 bit	This is ignored for External BUR (Bit 11).
13 14 15	63H	5 6 7	Reserved	0 1 0	CPU Delay[a.]	
16 17	64H	0 1	CLK MODE	11	00 - Sys Clk * 1 01 - Sys Clk * 2 11 - Sys Clk * 3 (default) 10 - Sys Clk * 4	
18	64H	2	FPCI divide	0	Frontside PCI Clock Divide. 0- SysClk 1 - SysClk / 2. Note: SYSCLK_C is pin A20	**Note**: Frontside is the on chip PCI devices such as the IDE controller and the USB controller. See ZFx86 Fail-Safe PC-on-a-Chip Block Diagram.

Table 5.42 Composite BootStrap Register Map (cont.)

ISA BIT	ZFL Index	BS Bit	Name	Default	Function	BIOS / BUR Mapping to ZFx86 I/O Control Register (or notes)
19	64H	3	BPCI divide	0	Backside PCI Clock Divide. 0- SysClk 1 - SysClk / 2. Note: No effect if bit 20 is 0	**Note**: Backside PCI is the slots or off-chip PCI devices. See ZFx86 Fail-Safe PC-on-a-Chip Block Diagram.
20	64H	4	BPCI Select	1	Backside PCI Clock Select. 0 - External clock. 1- Internal clock.	
21	64H	5	Reserved	0	0 = USB normal operation[a.] 1 = USB test mode	
22	64H	6	Reserved	1	ZF-Logic Enable. [a.] Disables all[c] ZF-Logic if low.	See "IO_ZT_EN" on page 240
23	64H	7	Z-tag enable	0	Causes BUR Boot. Enables the Z-tag Interface and BUR if high. See 5.10. "BUR Base Register" on page 439	See "IO_ZFL_EN" on page 240
0-3	62H	0-3	User Defined	0	User Defined	
4	62H	4	Reserved	0	South Bridge scan enabled[a.] 1 = enable scan in operational mode.	
5	62H	5	14 Mhz clock source	0	14MHz Clock Source If 1, derive from 48Mhz. If 0, use mhz14_c pin. [AF16]	See "IO_EXT_CLK_14M" on page 240
6	62H	6	32 KHz	0	32KHz Clock Source If 1, derive for 48MHz. If 0, use 32KHZC [AF01]	See "IO_RTC_32K" on page 239
7	62H	7	Reserved	1	486 DLL mode[a.] 1 = DLL mode enabled	

a. This is a **reserved** bit used in testing. It should be left in its default state in user-designed systems.
b. If you select BUR boot (BS23) and external BUR (BS11) - in that case GPIO0 becomes the chip select for the external BUR. BS12 is ignored in this case, and the width of the external ROM/FLASH is forced to 8-bits. External BUR allows ZF to produce custom BUR chips for special applications.
c. When the ZF Logic is disabled, the MEM_CS* pins remain active.

5.8.1. Special Notes of Interest

The bootstrap registers or boot parameters register, as it is sometimes called, highlights a number of options and points of interest.

5.8.1.1. Multiplexing of Pins A14 - B14

While designing the ZFx86 "System on a Chip", every attempt was made to include all of the standard features and services used by current PCs, plus as many extras as we could squeeze in. In order to accomplish this, a few trade-offs had to be made. The number of available I/O pins on the ZFx86 do not allow all functions to operate simultaneously. We assumed that designers would tend to user either PCI or ISA devices, but not a large number of both types on the same board. The decision was made to share one set of I/O pins (A14 and B14 - see 8.2. "Pin Descriptions (Sorted by Pin)" on page 537) between the following functions:

1) DMA Request/Acknowledge #1

These signals are used (primarily) by ISA cards. Specifically, ISA Sound Boards tend to use these as one of the default DMA channels. It is often possible to select a different DMA channel, and avoid using these signals.

2) PCI Request/Grant on Slot #3

These signals are needed for any card installed into the third PCI slot. The ZFx86 Integrated Development System, and any other design built around the ZFx86 Chip, can use either of these sets of signals, but only ONE of the two functions can be used on a given board.

A more detailed disussion of this occurs in the Quick Start Guide for the Integrated Development System, where BS9 is changed in conjunciton with another jumper to route pins A14-B14 to their appropriate ISA or PCI destination.

5.8.2. Design Example

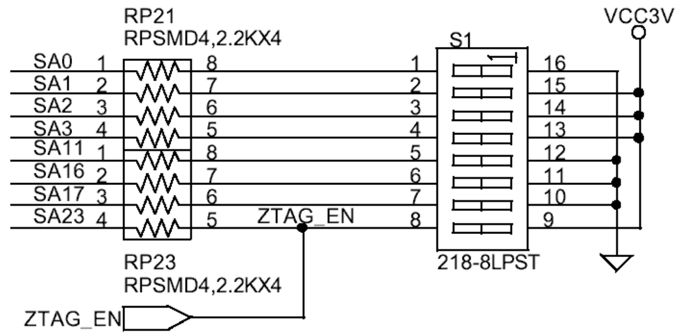
Table 5.43 Sample DIP Switch Settings

SW	Function
1	ON - USER DEFINED
2	ON - USER DEFINED
3	ON - USER DEFINED
4	ON - USER DEFINED
5	ON - EXT ROM when Z-tag enabled[a]
6	System Clock Speed
7	System Clock Speed
8	ON - BUR Boot (and thus Z-tag enabled)

a. This bit should NOT be used in real designs. It is for testing only.

A sample hardware design is shown below. Note that DIP Switch 8 controls SA23 (Boot Parameters Register Bit 23). When the dongle is plugged in to the board, the dongle connects CLK to ACK and also pulls up SA23. See 6.3.2. "Z-tag Data Transfer Protocol" on page 454.

Compare this table (and the drawing below) with the Boot Parameters Register. For example, note that SW 6-7 go to SA16-17.



Note: This schematic is a sample. Refer to ZF's reference designs for a "real world" DIP switch application.

Figure 5-8 Sample DIP Switch Schematic

ISA BIT	ZFL Index	BS Bit	Name	Default	Function
23	64H	7	Boot from BUR (sometimes called Z-tag_EN)	0	Boot from BUR 1 = Boot from BUR 0 = Boot from Flash

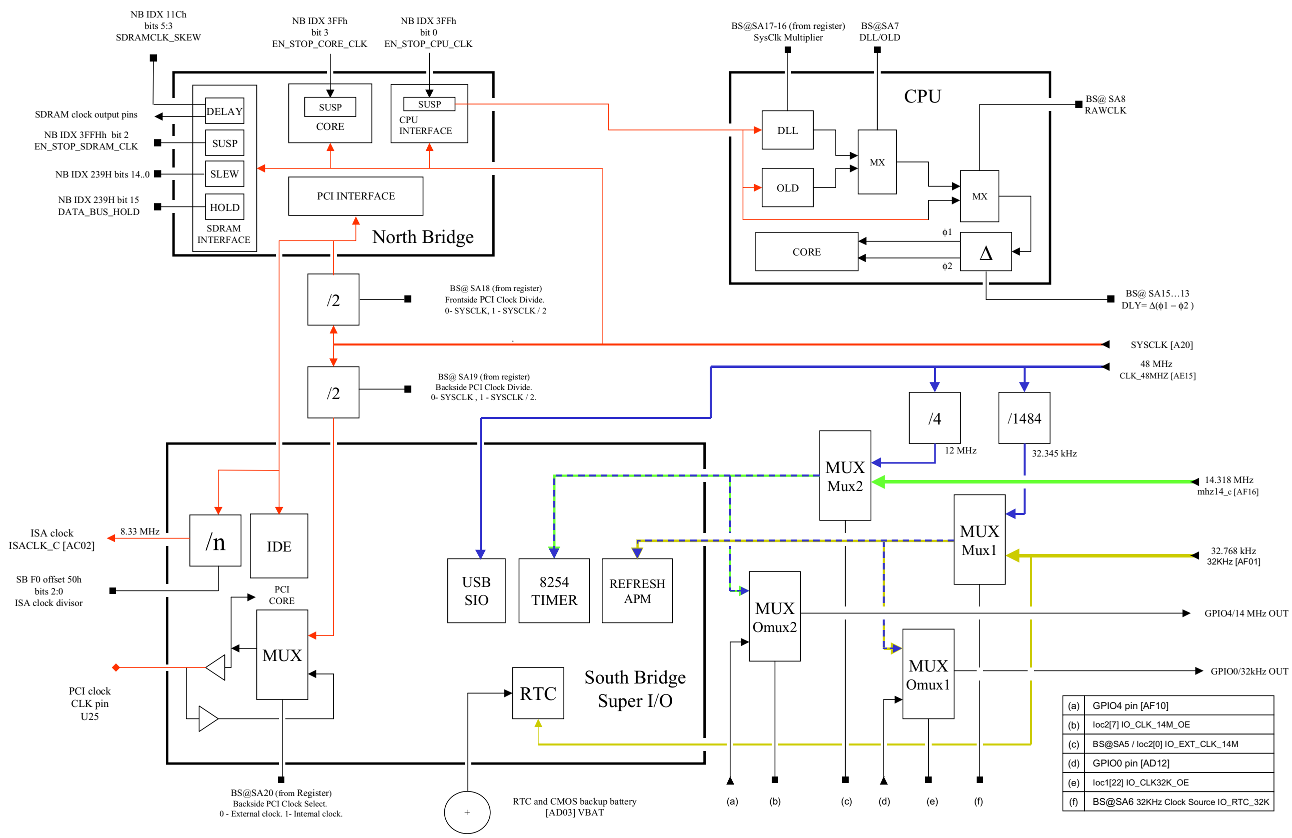


Figure 5-9 System Clocking and Control

5.8.3. Clocking and Control Overview

ZF Micro Devices Inc. has not attempted to hide some of the complexity of the chip; this manual instead illustrates internal functionality which is controlled by the bootstrap bits (even though many of these bits are reserved for testing).These bits rely on jumpers and/or DIP switches which appear on the demonstration and evaluation boards available from ZF. See Chapter 9. Design Example - Evaluation 1 Board and the "Evaluation 1 Board Quick Start Guide".

In general, the chart above illustrates clocking control bits, APM/suspend control bits, and other control bits related to the SDRAM and CPU core options.

The four clocks which may all be present are:

- CLK_48MHZ - USB Clock
- 32KHz - Refresh Generator / APM / RTC
- SYSCLK (typically 66 MHz)
- mhz14_c - Used for the 8254 PIT

5.8.3.1. Clocking Options

The ZFx86 has various clocking options. These options represent different trade-offs that the designer must investigate to come to the best solution for the application being considered. Essentially, the chip can be clocked using as many as four clocks or as few as one.

Various combinations of bootstrap registers allow the ZFx86 to be used with fewer clocks in the system. Examples are shown in "System Clocking" on page 441.

5.9. Data registers (F0H to FEH)

These 20 bytes of register space are used to store the ZFx86 BIOS specific data.

- The first 5 registers are reset during cold boot (registers 68H to 71H).
- The second 5 keep their values (registers 72H to 7BH).

These registers do not interfere with other device functions and are used by BIOS as a scratch area.

Table 5.44 ZF-Logic Index for the Scratch Register

68	Scratch Register 0 Low (68H)	Scratch Register 0 High (69H)	Scratch Registers
6A	Scratch Register 1 Low (6AH)	Scratch Register 1 High (6BH)	
6C	Scratch Register 2 Low (6CH)	Scratch Register 2 High (6DH)	
6E	Scratch Register 3 Low (6EH)	Scratch Register 3 High (6FH)	
70	Scratch Register 4 Low (70H)	Scratch Register 4 High (71H)	
72	Scratch Register 5 Low (72H)	Scratch Register 5 High (73H)	
74	Scratch Register 6 Low (74H)	Scratch Register 6 High (75H)	
76	Scratch Register 7 Low (76H)	Scratch Register 7 High (77H)	
78	Scratch Register 8 Low (78H)	Scratch Register 8 High (79H)	
7A	Scratch Register 9 Low (7AH)	Scratch Register 9 High (7BH)	

Table 5.45 Indices for Scratch Registers

Function	0	1	2	3	4	5	6	7	8	9	Description
Low	68H	6AH	6CH	6EH	70H	72H	74H	76H	78H	7AH	Scratch Register “N” Low
High	69H	6BH	6DH	6FH	71H	73H	75H	77H	79H	7BH	Scratch Register “N” High
	Temporary Scratch Registers					Persistent Scratch Registers					

Table 5.46 Scratch Register “N” High or Low

Bit	7	6	5	4	3	2	1	0
Function	Temporary Data Register							
Default	0							
R/W	R/W							

5.10. BUR Base Register

Table 5.47 ZF-Logic Index for BUR Base

Index	8-Bit Data at Index	8-Bit Data at Index + 1	
56		BUR Base Low (57H)	
58	BUR Base High (58H)		

Table 5.48 BUR Base Bits 15-12

Bit	7	6	5	4	3	2	1	0
Function	d15	d14	d13	d12	fixed	fixed	fixed	fixed
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	N/A	N/A	N/A	N/A

Bit	Name	Function
7:4	d15..d12	BUR base address bits 15..12
3-0	fixed	Fixed in Hardware

Programming Notes: The BUR base register defines the beginning address of the on-chip BUR software in system memory space. Normally this register is initialized to zero and BUR does not appear anywhere in memory. However, if BUR is requested by bootstrap at SA23 on system reset, the BUR base register is initialized to 000FC000H and BUR becomes visible at BIOS ROM area.

Writing zero to that register instantly disables the BUR window.

The BUR is actually 12Kbyte software inside the chip, however the base register enables a 16K byte window for it. Therefore, the actual data begins at BASE+4Kbyte.

The BUR base register is 32-bit register, accessible through 218h/21Ah. The middle two bytes can be accessed through 8-bit accesses at 218h/219h

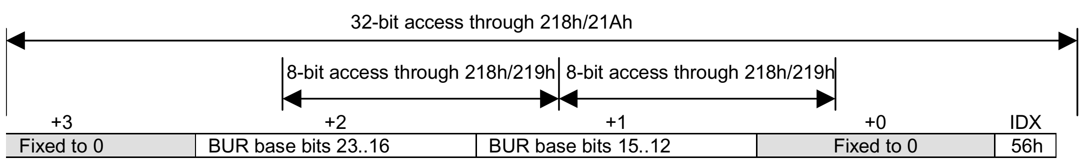
Table 5.49 BUR Base Bits 23-16

Bit	7	6	5	4	3	2	1	0
Function	d23	d22	d21	d20	d19	d18	d17	d16
Default	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit	Name	Function
7:0	d23-16	BUR base address bits 23..16

Programming Notes: .BUR base register are internally 32-bit register, that can be accessed through 218h/21Ah. The bits 32..24 and 7..0 of the BUR base address register are fixed and can not be modified. The register is also accessible as 8-bit registers through 218h/219h.

Using the 32-bit access at 218h/21Ah allows to write the direct 32-bit value to a register BUR base address.



5.11. System Clocking

Table 5.50 System Clocking

Functional block	Type	Pin	Name	Frequencies allowed
System clock	Input	A20	SYSCLK_C	1 to 66MHz [a]
Real-time clock	Crystal	AE01	32KHZ_C	32768Hz
Real-time clock	Crystal	AF01	32KHZC_C	32768Hz
8254 timer	Input	AC14	14MHz_C	14.318MHz [b]
Super-IO	Input	AE15	USB_48MHz_C	48MHz [c]
ISA bus	Output	AC02	ISACLK_C	Derived from the Backside PCI Clock
Backside PCI	Output	U25	PCICLK_C	SYSCLK or SYSCLK/2
SDRAM	Output	B22	SDRAM_CLK[0]	SYSCLK
SDRAM	Output	A22	SDRAM_CLK[1]	SYSCLK
SDRAM	Output	B21	SDRAM_CLK[2]	SYSCLK
SDRAM	Output	A21	SDRAM_CLK[3]	SYSCLK
Optional 32 KHz	Output	AD12	GPIO[0][d]	Program GPIO[0] pin to output this frequency.
Optional 14 MHz	Output	AF10	GPIO[4][d.]	Program GPIO[4] pin to output this frequency.

a. The System clock drives the North bridge (including the SDRAM interface), the CPU and the South bridge PCI interface. It is possible to use any frequency up to 66 MHz. The CPU also has a multiplier that allows it to be clocked at 1, 2, 3 or 4 times the system clock.

b. 8254 timer (PIT, Programmable Interval Timer) clock drives the 8254 timer circuit. It provides the timer interrupt and Legacy system time-base. If this clock is any other value than 14.318 MHz (ISA bus OSC signal) it is necessary to reprogram PIT to create the correct time-base.

c. Super-IO clock synchronizes all Super-IO devices. It must be EXACTLY 48 MHz to create the correct USB and RS232 timings. It is possible to run ZFx86 from only this source.

d. See Table 4.8 "GPIO Interface Signals" on page 177, and see Figure 5-9 "System Clocking and Control" on page 437 (and the figures following in this section).

Various combinations of bootstrap registers allow the ZFx86 to be used with fewer clocks in the system.

Mode 1:

All clock sources present. System, Real Time, USB and 8254 Timer clocks.

- Compatible with legacy PC systems
- Most flexible in both speed, features, and software
- Most expensive HW cost
- No change to SW/HW

Mode 2:

Two clock sources: USB and Real Time clocks.

- 14 MHz ISA clock derived from the USB clock, this will generate a clocking error. It also requires the Programmable Interval Timer (PIT divider) to be changed.
- PCI clocks need to be either 24MHz (SYSCLK divided by two) or 48MHz.
- Less expensive HW.

Mode 3:

One clock source: USB clock only with GPIO[0] pin fed back to RTC. Connect the USB clock input to the SYSCLK input.,

- Incompatible with legacy PC systems.
- Will lose the clock (Date and Time) if the power is removed from the system.
- Legacy SW/HW may need to be changed if dependent on legacy timers.
- Least expensive HW solution.

Table 5.51 Formal Clock Names and Clocking Modes

Name	Reference	Nominal	Mode 1[a]	Mode 2	Mode 3
System Clock (generates PCI and CPU Clock)	"SYSCLK_C [A20]" on page 444	33-66 MHz	yes	b	b.
Real Time Clock	'32KHZC_C [AF01]' on page 443	32KHz	yes	yes	c
USB Clock	'USB_48MHz_C [AE15]' on page 447	48 MHz	yes	yes	yes
8254 Timer	'mhz_14c [AF16]' on page 442	14 MHz	yes	d	d.

a. In mode 1 all clocks are present.
b. Drive both the USB Clock and the SYSCLK with a 48 MHz signal.,
c. Drive the 32KHz Real Time Clock with GPIO[0] output which is USB_48MHz/1484. See Figure 5-11 "32KHZC_C [AF01] Clocking Control Circuitry" on page 444.
d. Get the 8254 Timer the USB Clock/4 or 12 MHz. See Figure 5-13 "USB_48MHz_C [AE15] Clocking Control Circuitry" on page 447.

5.11.1. mhz_14c [AF16]

This clock generates the timer interrupt from the 8254 PIT. The sources for this signal are from mux2. The default is to use the 14.318 MHz input pin, but you can use an internally generated 12 MHz clock by asserting ***either*** IO_EXT_CLK_14M or SA5.The 12 MHz clock comes from CLK_48MHz. Z which is a reguired clock. See IO_EXT_CLK_14M on page 240).

The output of Mux2 (14.318 or 12 MHz) may be driven out on GPIO[4] . See IO_CLK_14M_OE on page 240. [1]

If you use the 12 MHz source, the input frequency to the 8254 PIT is lower, but it is possible to compensate for it by using different divisors in the PIT.

The clock input to the 8254 PIT drives all 3 PIT channels. The clock input is divided by 12 (not shown in the schematic).

The legacy PIT Channel 0 is used to create refresh DMA cycles. As the memory controller has moved to North Bridge this is obsolete.

The legacy PIT Channel 1 creates PC speaker tones. The higher tones (10 KHz) differ by (12/14) / 1000 = 0.8 mHz. The lower tones differ in µHz areas. This change is not noticeable in a PC-Speaker.

1. In order to GPIO[4] to work as a clock output, you must program a "0" (or at least not program a "1" to the GPIO[4] output pin. See Table 4.33 on page 228.

The legacy PIT Channel 3 creates a timer interrupt. by dividing the input signal by 0xFFFF. The legacy result is 18.206268 Hz or 54.926138 ms. In our case the timer runs at 1 µs therefore, the divisor must be 54926 (decimal) resulting in a frequency of 18.206313 Hz (54.926002 ms). This creates an error of 2.47x10-4 % or one timer tick less in 398014 ticks. If not corrected this produces one second delay every four days.

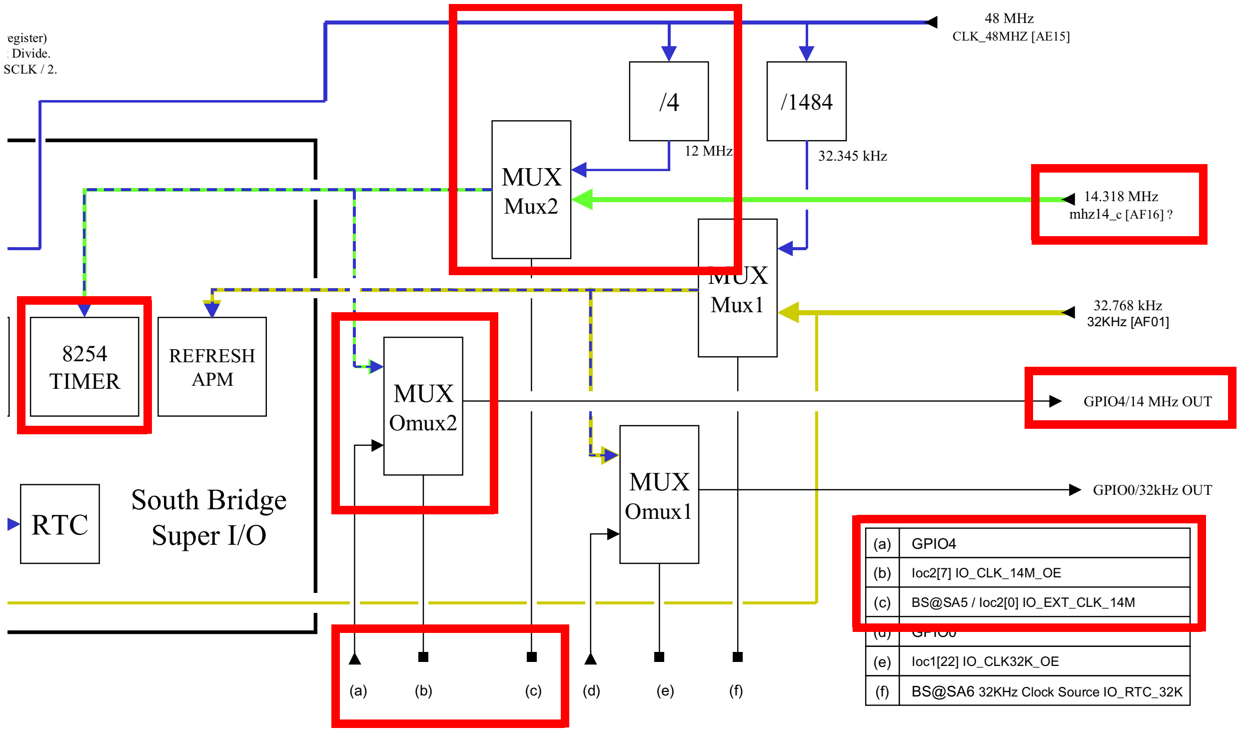


Figure 5-10 mhz_14c[AF16] Clocking Control Circuitry

5.11.2. 32KHZC_C [AF01]

This signal generates the refresh for SDRAM and drives power management logic. It also synchronizes the Watchdog timers and PWM in ZF-logic. The two sources are from MUX1. MUX1 selects between the 32.768 kHz input in pin AF01, or the 48 MHz clock divided by 1484.

If you do not have a 32.768 kHz input, you may use the 48 MHz clock. To use the 48MHz clock to emulate the 32 KHz clock, assert SA6 using a DIP switch or resistor pack (see Table 5.42) **or** use software to assert **IO_RTC_32K** (see page 239).

The real time clock 32 Khz crystal is required to keep the clock running from battery. If time loss is not important, then the RTC can be driven from the internal 32 KHz by first outputting it to GPIO0 and then connecting it externally to crystal input pin.

If you do not have a 32.768 kHz clock, and use the 48 MHz SIO clock divided by 1484, you will get an RTC error of approximately 1.29% (32345 vs. 32768).

This 32 kHz (actual or derived) clock may be driven out on GPIO[0] by setting **IO_CLK32K_OE** (see page 239).

If you are using the derived clock (the 48 MHz clock /1484 option), please note that the 32 kHZ clock does not go to the RTC as there is no interconnect along that pathway. In that case, you can use GPIO0 in its 32kHz output mode and run that output back around into the 32KHz input. That is somewhat roundabout, but it works.

Compare 9.2.5. "Schematic Page 2 JP9 - Real Time Clock (32KHz)" on page 595.

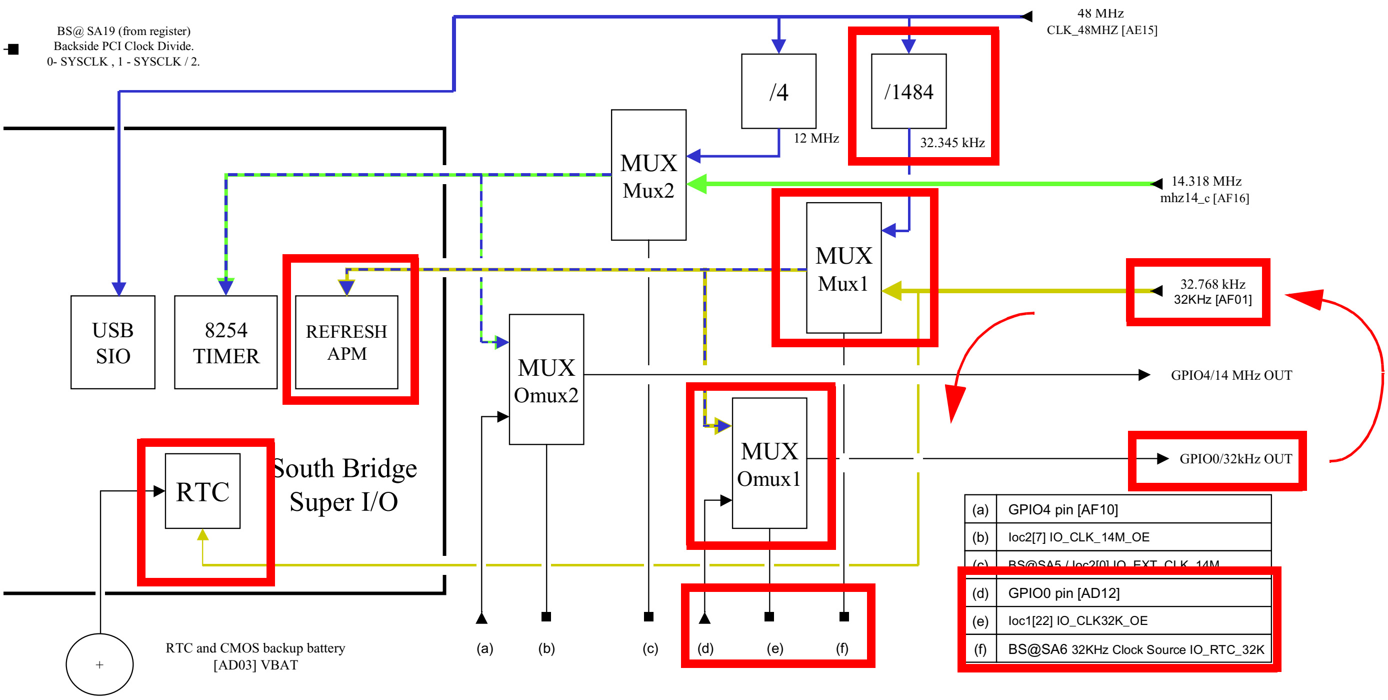


Figure 5-11 32KHZC_C [AF01] Clocking Control Circuitry

5.11.3. SYSCLK_C [A20]

This clock drives the Processor, the North Bridge and both South Bridge PCI interfaces. The signal has a single source - the SYSCLK input pin. This clock is limited by following considerations:

1. The backside PCI bus implies a limit of 33 MHz for external devices. Although internally it can be higher each external device has to be checked separately. It is possible to divide the SB PCI frequency separately by two and run the SDRAM and CPU interfaces at a higher frequency. A very key component of the performance of the system is the clock speed at which the SDRAM gets accessed. Therefore, pushing the system clock up significantly enhances the system performance.

2. Front side PCI core frequency must always be greater than or equal to the backside PCI bus. It's recommended not to exceed 33 MHz. This clock can also be divided by two.

3. The memory controller can run at a maximum speed of 66 MHz.

These conditions are summarized in Table 5.52.

Table 5.52 CORE frequencies (MHz).

Functional Block	Min. Freq.	Typical Freq.	Max. Freq.
Backside PCI	1	33	33 (dependent on device)
Frontside PCI	1	33	33 (greater than or equal to the backside PCI)
SDRAM controller and CPU interface	1	33	66
CPU clock (multiplier)	4 (4x)	99 (3x)	132 (2x)

CPU clock

The CPU clock can be programmed to be 1, 2, 3 or 4 times the SYSCLK. As the CPU core will work only up to 133 MHz it is necessary to select the proper multiplier based on the selected SYSCLK frequency. Suspend logic is present in the device to allow for stopping the clock to the CPU if the North bridge receives a suspend request from the South bridge.

SDRAM clock

SDRAM clock is exactly the same as SYSCLK input. It has a skew control register, however, all measurements taken during the development cycle indicate that the skew register should be left at zero.

Front and Back side PCI clock

These clocks can be selected to run at the system clock frequency or half the system clock frequency.

ISA Clock

See ISACLK in Table 4.9 "Full ISA Interface" on page 178. ISACLK is derived from PCICLK and is typically programmed for 8.33MHz.

F0 Index 50h[2:0] is used to program the ISA clock divisor. These bits determine the divisor of the PCI clock used to make the 8.33MHz ISA bus clock. See "F0 Index xxh: PCI Header and Bridge Configuration Registers" on page 202.

Suspend logic

The suspend logic allows the clocks to be turned off when a suspend request is active. The system suspend has the following hierarchy:

1. South bridge suspend request -> North bridge
2. North bridge suspend request -> CPU
3. CPU suspended
4. CPU suspend acknowledge -> North bridge
5. North bridge suspended
6. North bridge suspend acknowledge -> South bridge
7. South bridge suspended

The wakeup follows the opposite scheme: SB -> NB -> CPU. It is possible to switch off the SDRAM, CPU and NB core clocks when the system is suspended. The PCI and ISA clocks will always run and the system will keep the last value on the output pins when suspended. The South bridge is responsible for generating suspend and wakeup events.

Refer to the South Bridge spec for a detailed description of all possible wakeup events. The CPU can suspend itself until the next interrupt following HLT command. This must be enabled in the CPU control register.

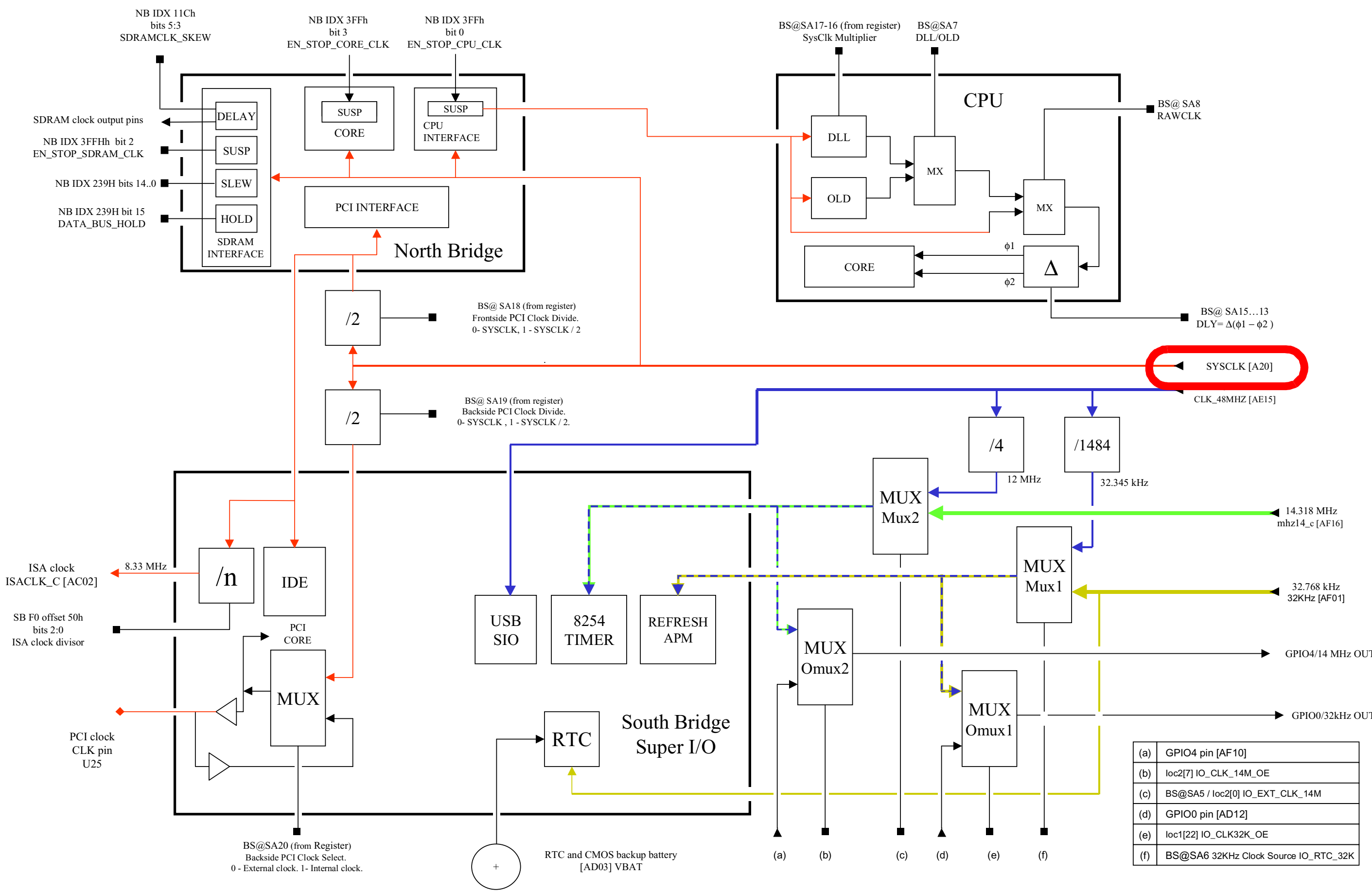
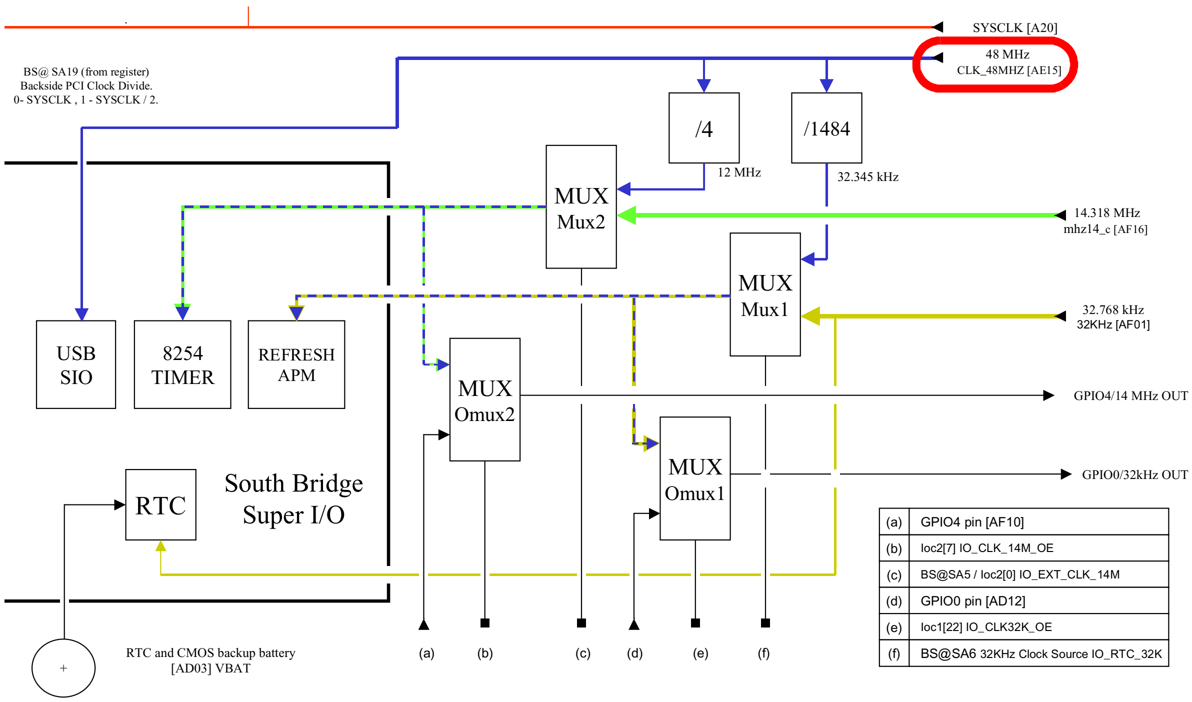


Figure 5-12 SYSCLK_C [A20] Clocking Control Circuitry

5.11.4. USB_48MHz_C [AE15]

Note that the USB and SIO (Super I/O) get their timing from a fixed 48 MHz clock and do not depend from other clocks. However the USB is a BUS master device and requires certain bandwith to operate at high speeds. It is therefore not reasonable to set the system clock below 33 MHz when using it.

Note that when using only one clock for the entire system, you need to provide this clock. See 'Mode 3:' on page 442.



(a)	GPIO4 pin [AF10]
(b)	Ioc2[7] IO_CLK_14M_OE
(c)	BS@SA5 / Ioc2[0] IO_EXT_CLK_14M
(d)	GPIO0 pin [AD12]
(e)	Ioc1[22] IO_CLK32K_OE
(f)	BS@SA6 32KHz Clock Source IO_RTC_32K

Figure 5-13 USB_48MHz_C [AE15] Clocking Control Circuitry

5.11.5. PCI Clocking

As an example, you could set SYSCLK to 66 MHz. Then use a clock multiplier of 2 to run the CPU at 132 MHz. You can use SA18 to control the frequency of the Frontside PCI Bus. The frontside bus should be running at a clock greater than or equal to the backside PCI Bus. The frontside PCI clock controls the internal IDE devices.

If you use the internal IDE controllers, they are validated at 33 MHz and thus the FrontSide PCI bus should not be set to a speed higher than that. In that case, you would enable both /2 divisors and use 33 MHz for the FrontSide and BackSide PCI Clocks. The PCI Core circuitry inside of the South Bridge drives the external PCI buses.

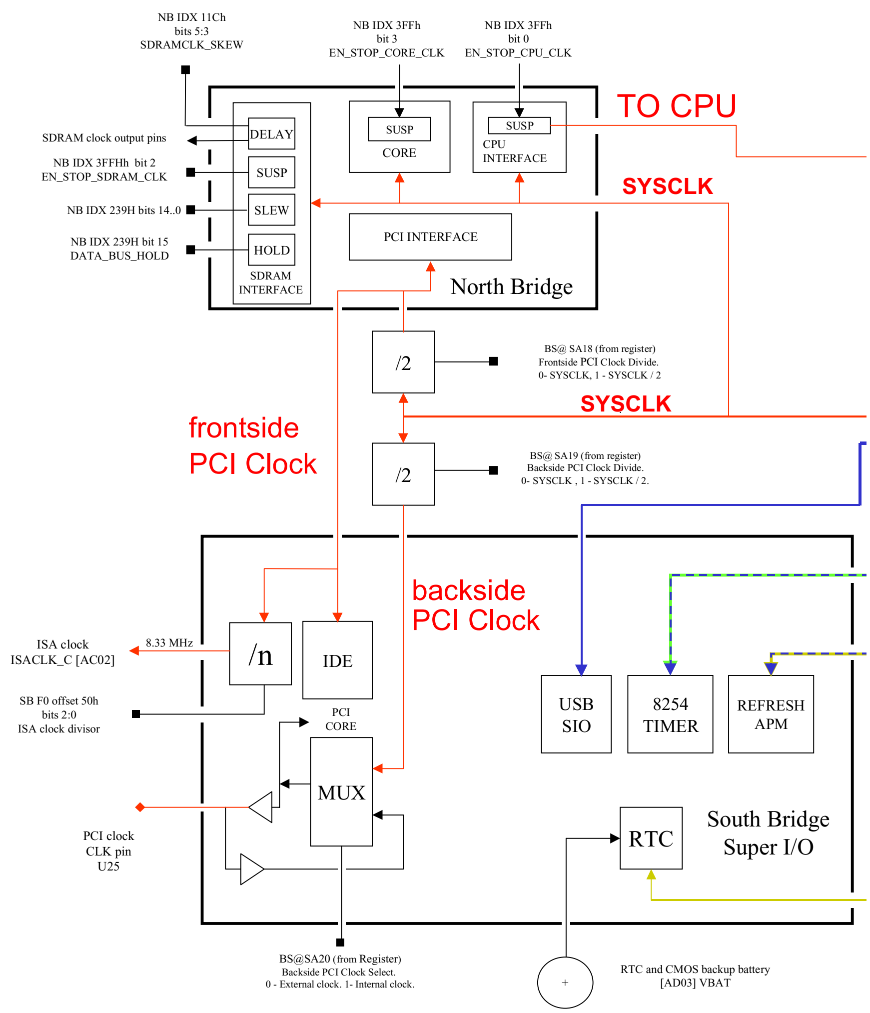


Figure 5-14 PCI Clocking Control Circuitry

6. Z-tag, BUR, and The ZFiX Console

The ZFx86 introduces a solution to resolve the situation created by having a high integration CPU connected to flash device that has not been initialized. During the manufacturing process the on board flash device must be initialized with the system startup code (BIOS) and/or application software. This implies that the ZFx86 chip is in a situation where there is no boot code that can be read from the flash device to allow the update. For that purpose, the ZFx86 contains a special **B**IOS **U**pdate **R**OM (BUR), that when executed updates the flash memory. The BUR contains the minimal code necessary to read data into the ZFx86 chip. In order to update the flash, flash component specific software must also be downloaded and executed. The connection between the ZFx86 chip and the flash data source can be made in two ways: by a ZF proprietary Z-tag Dongle interface or by using a serial port.

6.1. Serial Port Connection

The serial connection utilizes the ZFx86 embedded standard COM1 (base address 0x3f8), allowing a remote PC with special host software or a terminal to access the ZFx86 flash device and use the update process. This solution can be used in ZFx86 applications where the serial port is not hardwired to an external device and access to the serial port is physically possible.

6.2. Z-Tag Dongles

ZF provides two dongles types: PassThrough and Memory

6.2.1. PassThrough Dongle

Use the ZFx86 PassThrough Dongle, supplied with the ZFx86 IDS system, to make a physical connection from the PC running Z-tag Manager to the target system.

The PassThrough Dongle contains three LEDs (two green and one yellow) that indicate power, busy, and status.

6.2.2. Memory Dongle

An optional ZFx86 “Memory" Dongle contains 256Kbytes of memory, two LEDs (green and red) that indicate status, and two configuration jumpers.

- Jumper JP1 sets write protection for the Dongle’s onboard SEEPROM(s), when set in position 2-3.
- Jumper JP2 enables PassThrough or Normal operation mode. Jumpering 2-and-3 enables the Z-tag software PassThrough mode, while jumpering 1-and-2 selects Normal Mode.

On hardware reset the ZFx96 samples the ISA address bit 32 (bootstrap bit 32). If it is pulled high, the BUR is executed. If it is pulled low, the boot is attempted from the external Flash memory.

Note that the PassThrough dongle requires a Printer cable for proper operation.

6.2.3. Using the Dongle

Primarilary, you use the dongle and the Z-tag Manager interface to download programs (like BIOS) into your designs external Flash memory. You need not power down the design system to do this. You simply plug in the Dongle, press the system's reset button, and wait for the BIOS to download. When it completes, the Status LED appears solid yellow.

Since you can replace the entire Flash memory "almost instantly", you can also put manufacturing test programs into the dongle, load them, run them, and when finished simply replace the memory. How is this done? The on-chip BUR code has the capability to read commands and data buckets from the dongle – thus the intelligence to do the transfer (similar to the ZFx86 FailSafe Mechanism) is already on chip.

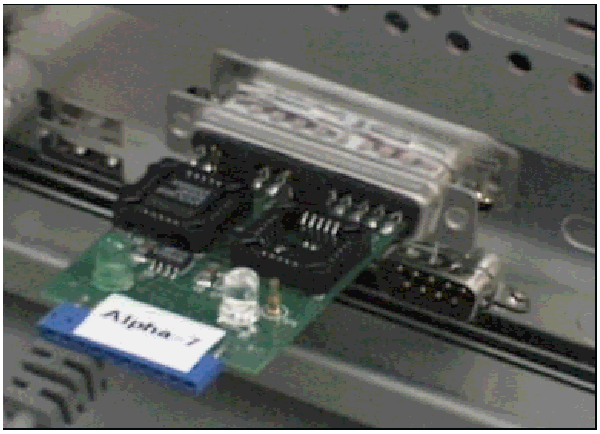
Create the command set in a host PC running 32-bit Windows using a ZF Windows application called the Z-tag Manager. This application allows you to drag and drop commands into a command list. One of the commands contains an embedded (attached) ROM image. The ROM image is contained within a "basket" or as an attachment.

6.2.3.1. Using the PassThrough Dongle

The PassThrough Dongle serves as an adapter that connects the Host PC to your target board. The Z-tag Manager software manages the transfer of data directly to the target system, emulating the behavior of the SEEPROM(s) on the "Memory" Dongle but from within the software. An advantage of PassThrough dongle is that it handles larger-sized transfer images. Because the data is not written into the Dongle's SEEPROM(s), it instead transfers directly to the target system's memory chip(s). See the Z-tag Manager User's Manual for more information.

6.2.3.2. Using the Memory Dongle

In the figure following, the Memory dongle is being programmed while attached to the Printer port of the host system. Then the dongle is plugged into the target system, and the program download into the ZFx86. See 6.2.3.3. "Carrying the Command Set in The Memory Dongle".



It is also possible to bypass the SEEPROMS on the Memory dongle, and use it in "PassThrough" mode where a cable from the host system is connected to the dongle, and the dongle is left plugged into the target board Enable the PassThrough mode in software using the Z-tag Manager.

- In "normal mode" you load the software and command set directly into the dongle's SEEPROM memory chips.
- In "passthrough mode" you directly connect the ZFx86 board (through the dongle) to the host PC.

In both cases, you use the Z-tag manager to transfer the information into the dongle or throught the dongle directly to the target board. The Z-tag manager is available at no cost on the ZF web site and on the ZF CD ROM.

Note: Additional Flash chip-specific software is required to program a specific flash device. See 6.7. "Flash Programming Example" on page 464.

6.2.3.3. Carrying the Command Set in The Memory Dongle

Connect the Memory dongle directly to the printer port of a personal computer running Windows. Then, using the Z-tag manager, create a command set which includes carrying a ROM image in a "basket". This command sequence, which includes a file image as part of a transfer command, is stored in the SEEPROM in the dongle. There are two SEEPROMs on the Memory dongle, so you can load a program with a total size of 256K Bytes.

Note: If your program exceeds 256KBytes, use the ZF designed PassThrough dongle.

After using the Z-tag manager to load the Memory dongle, unplug the dongle from your PC printer port, and plug it into the ZFx86 board using the Z-tag header. Use a a 2514-6002UB 3M DIP-14 Header, or equivalent, to create the Z-tag header on your board.

If you have the dongle connected, the dongle jumpers CLK to ACK and pulls up SA23.

When there is a reset-pulse with the Memory dongle plugged in, the BUR starts [SA23] and reads from the dongle. If the BUR finds no Z-tag commands, it switchs to COM1 and looks for Z-tag commands. If none are found there, then the BUR starts the ZFix Console on COM1.

Note: To use the BUR with no dongle connected, use the DIP switch to pull up SA23.

The speed of the transfer from the dongle into the ZFx86 is limited by the software (that is, the clock frequency) of the BUR. Typically we would expect about 1.5 Mbps. The Z-tag interface is designed (from electrical waveform point of view) to match the signals set of an SEEPROM. That is, the BUR puts out a CLK signal and reads in data. The basic protocol is that when CLK goes high the SEEPROM (or data source) puts new data on the data pin. The data is read when CLK goes low. When CLK goes high again the data source puts fresh data on the data pin.

This transfer method works just fine if the Z-tag interface is directly connected to the dongle or to an SEEPROM. However, if the data source is a host computer and the dongle is being used in PassThrough mode, then the host computer (which is generating waveforms on its printer port) may not be able to send the data out quick enough. Therefore, another signal was added called acknowledge (ACK). When the dongle is plugged into the ZFx86 (in the dongle "normal" mode of operation), the dongle connects ACK to CLK.

6.2.3.4. Memory Dongle Jumper Settings

The Memory dongle contains two jumpers (see Table 6.1). The Mode jumper sets the dongle to "PassThrough" mode or "normal" mode.

- In "PassThrough" mode, the Z-tag interface simply passes through the dongle to the printer connection; and you must run a cable from the printer port on your personal computer to the dongle (which then passes through to the Z-tag interface). In this case, CLK is not hardwired to ACK, allowing the host information-provider to provide the ACK signal after it places the data bit on the data pin. Thus, in "PassThrough" mode, the host computer watchs the CLK signal. When CLK goes high, the host computer places data on the data pin and raises the ACK signal. When the BUR sees the ACK signal go high, it drops CLK low and reads the data. See 6.3.2. "Z-tag Data Transfer Protocol" on page 454.
- In "normal" mode, the dongle directly connects the CLK output to the ACK output, allowing a maximum speed transfer. Note that this is totally transparent to the BUR, as BUR simply watches for ACK and then drops CLK.

The second jumper on the Memory dongle simply write protects the SEEPROMs on the dongle.

Table 6.1 Memory Dongle Jumper Settings

Name	Location	Function	Comments
JP2	Dongle	PassThrough or Normal	Normal Connects CLK to ACK when Dongle Plugged In
JP1	Dongle	Protect SEEPROM's memory	
Bootstrap SA23	Host Board	Cause BUR Boot	Automatically set by Dongle — add DIP switch to provide BUR boot w/o Dongle
CLK to ACK	Host Board	Reads Host Board SEEPROM	Not Needed if SEEPROM in the Dongle is providing Data

6.2.3.5. Using the Memory Dongle in "PassThrough" Mode

If you connect the host computer printer port through a cable to the dongle which in turn is plugged into the ZFx86 board, the operation is similar in that you use the software Z-tag manager to create your command set. When you have created the command set, you can then push the reset button on the ZFx86 board and the BUR will read from the Z-tag interface and get the same command set that you would have loaded into the dongle. The benefit of this mode is that you do not have to program the SEEPROM(s) in the dongle, but you can go directly to the desired operation. See Table 6.2.

Table 6.2 Z-tag and ZFiX Summary

Hardware	Mode	Software	Function
Dongle + Host		Z-tag Manager Windows Application	Programs the Dongle
Dongle + Board	Normal	BUR	Read/execute commands from the dongle.

Table 6.2 Z-tag and ZFiX Summary (cont.)

Hardware	Mode	Software	Function
Host + Dongle + Board	PassThrough	Z-tag Manager Windows Application and BUR	Read/execute commands from the host.
Host + COM1		Z-tag Manager Windows Application and BUR ZFix	Read and Execute Commands from the Host.
Host + COM1		Windows Terminal Emulator and BUR ZFix	Console Commands

6.3. Z-tag Manager Software

The Z-tag Manager program is a Windows Application provided by ZF Linux Devices. The program is used to create Z-tag command sequences and transfer them to the dongle or directly to the ZFx86 PC board. See the Z-tag Manager Manual and also see Chapter 4 in the ZFx86 Integrated Development System Quick Start Guide.

6.3.1. Z-tag Summary

Z-tag is ZF proprietary connection interface that does not interfere with any PC legacy devices and can be accessed even if all of the ZFx86 external peripheral interfaces are allocated. Physically, the Z-tag signals share the pins with the FDD interface and appear when the FDD MTR0 and DRV0 signals are not active. The communication protocol is compatible with standard serial EEPROM devices, thus allowing a direct connection of the serial EEPROM to the Z-tag interface as a source media. A second possibility for a data source is a remote PC emulating the Z-tag protocol with the parallel port. An adapter, called “the dongle”, converts the FDD pins (generally a header on the ZFx86 circuit board) to a Parallel Printer port interface. Z-tag Hardware Interface.

The Z-tag manufacturing connector layout has the following pinout.

- **CLK, RST** and **DATA** are asynchronous serial interface signals used on serial EEPROM’s. The use of these signals can be determined from the timing diagram of the Z-tag communication protocol.
- **ACK** input to ZFx86 is a loopback from **CLK** signal and is useful in situations where the source media response speed is unknown (i.e. emulation through PC parallel interface). The ACK signal can be either the CLK if the source media speed is considered to be faster than the ZFx86 clocking (i.e. the data bit will be set fast enough after the CLK rising edge appears) or specially created after the data bit is set by the source media.
- **LED1** and **LED2** are general purpose output bits, that are used to provide progress or status information when BIOS Update ROM is updating the flash.

Table 6.3 Pins for the FLOPPY / Z-tag Logic

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name
G03	DIR _N, Z-tagOut3	FLOPPY, Z-tag	m_fdc_p	DIR
F01	DR0_N	FLOPPY	m_fdc_p	DR0
J02	DSKCHG_N, Z-tagIn0	FLOPPY, Z-tag	m_fdc_p	DSKCHG
H01	HDSEL_N, Z-tagOut0	FLOPPY, Z-tag	m_fdc_p	HDSEL
F03	INDEX_N	FLOPPY	m_fdc_p	INDEX
F02	MTR0_N	FLOPPY	m_fdc_p	MTR0
J04	RDATA_N, Z-tagIn1	FLOPPY, Z-tag	m_fdc_p	RDATA
G04	STEP_N, Z-tagOut2	FLOPPY, Z-tag	m_fdc_p	STEP
H03	TRK0_N, Z-tagin3	FLOPPY, Z-tag	m_fdc_p	TRK0
G02	WDATA_N, Z-tagOut1	FLOPPY, Z-tag	m_fdc_p	WDATA
G01	WGATE_N	FLOPPY	m_fdc_p	WGATE
H02	WRPRT_N, Z-tagin2	FLOPPY, Z-tag	m_fdc_p	WRPRT

6.3.2. Z-tag Data Transfer Protocol

The hardware component of the data transfer is illustrated by following the timing diagram shown in Figure 6-1 "Data Transfer Protocol". The data bit latching (see blue arrows) must happen at a rising edge of the **CLK** signal. In those cases when the source device speed is unknown, the ACK signal is used to determine when the bit is ready for reading at DATA input (**ACK**=High) and when source device is ready to accept the new rising edge of CLK signal (**ACK**=Low).

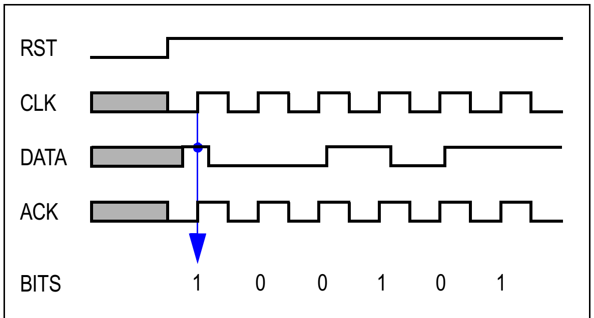


Figure 6-1 Data Transfer Protocol

DATA input shown in figure Figure 6-2 "Data Input".

In this case, the source (host) device sets the ACK signal when it has received the **CLK** edge and latched the data to the DATA input. The target (ZFx86) then reads a bit and resets the CLK signal. After that, the target/ZFx86 starts reading the **ACK** signal again, waiting for it to become low. When **ACK** goes low, the cycle will be repeated for next data bit. The bit stream following the reset deactivation is a continuous 8-bit character bit stream with no control bits, byte separators or addressing information. The software must implement a special packet header for determining the addressing and other control information.

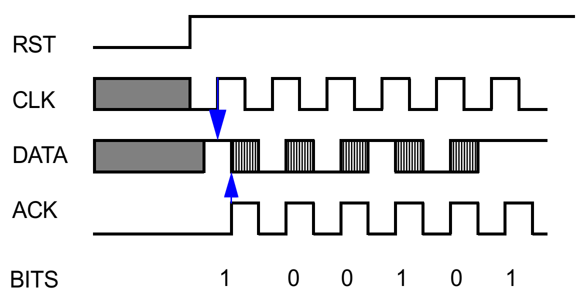


Figure 6-2 Data Input

6.3.3. Z-tag Port Interface

The Z-tag interface access is done via 8-bit ZF registers Z-tag Data Write Register (5EH), and Z-tag Data Read Register (60H). See ‘ZF-Logic Index for the Z-tag” on page 455.

6.4. Z-tag Register Descriptions

6.4.1. Z-tag data (D1h)

Z-tag interface is used to update ROM devices in BIOS upgrade mode. FDD interface signals are connected to the Z-tag data register when *MTR0* and *DRV0* are not asserted. FDD MUX is controlled using a bit in Z-tag control register. This process is transparent to the user.

The Z-tag has four inputs and four outputs. Each Z-tag line is connected to FDD line. In Table 6.4 "Z-tag Data Lines" the FDD lines are shown in parenthesis.

Table 6.4 Z-tag Data Lines

BIT	Name	Description
7	Z-tagIn3	Z-tag in 3, (TRK0)
6	Z-tagIn2	Z-tag in 2, (WRPRT)
5	Z-tagIn1	Z-tag in 1, (RDATA)
4	Z-tagIn0	Z-tag in 0, (DSKCHG)
3	Z-tagOut3	Z-tag out 3, (DIR)
2	Z-tagOut2	Z-tag out 2, (STEP)
1	Z-tagOut1	Z-tag out 1, (WDATA)
0	Z-tagOut0	Z-tag out 0, (HDSEL)

Table 6.5 ZF-Logic Index for the Z-tag

5E	Z-tag Data Write Register (5EH)		Z-tag
60	Z-tag Data Read Register (60H)		
7C	Z-tag control register (7CH)	Z-tag Sequencer Divisor Register (7DH)	Z-tag
7E	Z-tag Sequencer Waveform (7EH)	Z-tag Sequencer Strobe Points (7FH)	
80	Z-tag Sequencer Data (80H)	Z-tag Sequencer Status (81H)	

6.4.2. Z-tag control (7Ch)

The Z-tag enable bit of the Z-tag Control Register switches the FDD pins between FDD and Z-tag. These pins are listed in Table 6.4 "Z-tag Data Lines" on page 455. The four FDD output signals are taken from Z-tag data write register. The FDD *WGATE*, *MTR0* and *DRV0* signal are put to logical '1'. This ensures that the FDD is not interfering with the Z-tag data.

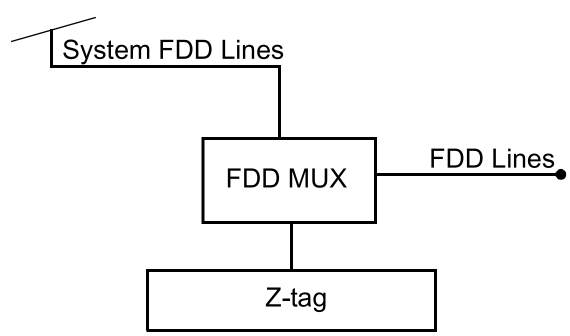


Table 6.6 Z-tag Control Register -- Index 7CH

Bit	7	6	5	4	3	2	1	0
Function	Reserved				Z-tag enable	accel enable	snoop ahead	reg select
Default	0				0	0	0	0
R/W	R/O				R/W	R/W	R/W	R/W

Bit	Name	Function
7:4	Reserved	
3	Z-tag enable	Muxes the FDD lines 0: normal operation 1: Z-tag signals on FDD lines
2	Accel enable	0: Z-tag normal operation 1: Z-tag accelerator active
1	Snoop ahead	0: Clear ready flag on accel reg read 1: Do not change the ready flag on accel read
0	Reg select	0: Select the output buffer for read 1: Select the accumulator register for read

6.5. BUR (BIOS Update ROM)

BUR is a ZFx86 built-in software that serves as a prototype debug tool and Flash update utility. BUR is a 12K binary image residing inside ZFx86 as ROM memory. On startup, the BUR performs basic component initialization functions and tests the ZF internal Static RAM. The Static RAM is used as an 8K scratch pad Area for performing BUR tasks. See ‘On-Chip RAM Assignment in BUR” on page 464.

After initialization, following system components are active:

- North Bridge
- South Bridge
- ISA Bus
- Internal Static RAM
- IRQ controller
- Timer (8259)
- COM1
- Z-tag interface

The 32-entry interrupt table is built and vectors are assigned for IRQ0-15. And INT0-8. (Additional 8 remaining interrupt table entries are used for internal service vectors by BUR code).

No DRAM is initialized or used by BUR code.

The functionality of BUR code can be divided into four categories:

- Basic component initialization
- Elementary debugger console functionality through COM1
- Data fetch and execution through Z-tag interface
- Basic OS functionality for user code

When BUR is executed after system reset, the component initialization occurs first. After component initialization, BUR will probe the Z-tag interface for dongle present (see Figure 5-7 "Dongle (w/o Cover)" on page 428). If the dongle is found, the BUR will start fetching command records from the dongle and perform accordingly. When the dongle cannot be found using the Z-tag interface, BUR will initialize COM1 and start ZFiX debugger console on COM1. An example of downloading a small test program via the COM1 port into the scratch pad RAM is shown in 'BUR COM1 Download Examples" on page 460.

6.5.1. ZFiX Console Functions

The ZFiX console is an ordinary TTY command-line interpreter that can receive typed commands and performs tasks requested by the operator. See Table 6.7 for the supported commands.

Table 6.7 ZFix Console Commands

Command	Action
i[n[b]]/inw/ind <port>	read 8/16/32-bit value from port
o[ut[b]]/outw/outd <port> <value>	write 8/16/32-bit value to port
zfr <register>	read 8-bit value from ZFLogic register
zfw <register> <value>	write 8-bit value to ZFLogic register
db/dw/dd <address>	display memory in byte/word/dword mode
d	display next memory page in previous mode poke[b]/pokew/poked <address> <value(s)> -
linear	use linear mode addressing
real	use real mode addressing
h[elp]/?	show help
ver	display verson information
speed <96/19/38/56/115> <hs>	serial speed. Set hs to 1 for RTS/CTS[a]
ztreset	reset Z-tag device
ztdir <size>	show Z-tag device content within given size
ztload <address> <count>	load bytes to location from Z-tag
ztexec <record> [range]	fetch and exec command from Z-tag if in range
ztseek <record> [range]	seek Z-tag device to record number if in range
ztseekl <offset>	seek Z-tag device to byte offset
ztremote [record]	start automatic fetch-exec procedure
yload <address>	load data through YModem to address
ysend <address> <length> [filename]	send data through YModem from address

Table 6.7 ZFix Console Commands

Command	Action
g[o] <address>	start executing from address
dls	Display available download segment address

a. The default speed on power up is 9600. The <hs> handshake bit is currently not working. You may try higher speeds, but you may lose data.

The information in the table above is from the actual ZFiX console help screen (displayed in response to the "help" command).

Most of help line descriptions are self-explanatory, however, some of these do need a second look:

"real/linear" - allows to use 16-bit or 32-bit memory addressing for db, dw, dd and poke* commands. If "linear" is selected, the whole 4Gb memory range can be examined and addressed. "real" command will drop back into real mode, when SEG:OFFSET notation should be used and only first megabyte of memory is accessible.

"zt*" - These commands are used for operating the Z-tag dongle as a block device. They give you access like to normal data storage device and allow executing and loading data from dongle manually or request automatic fetch and execution procedure from dongle. The optional range parameter on some commands is used to prevent infinite seek when requested command record is not found, since there is no way for determining serial data stream length and without specified range BUR tries to load more and more bytes from the dongle until the specified record number is found.

"ysend/yload" - these are YModem data transfer functions. Data transfers are protected by 16-bit CRC and are compatible with popular terminals like Term95, Hyperterm, Telix etc.

The console command line parser handles all entered numeric values as hexadecimal and uses extensive data type checking to catch user errors like entering a double word in place of a word. The command line is also cleared up before parsing, i.e. all lowercase is converted to uppercase and extra spaces removed between tokens.

6.5.2. Z-tag Functionality

If Z-tag dongle is detected at BUR startup, the command records will be fetched from the dongle and executed. The Z-tag dongle is serial stream device, so the execution will be done on record-by-record basis starting from offset0 and new records are fetched and executed until STOP record (type 05) is reached.

The Z-tag dongle data is divided into records. Each record has its own header and CRC. The header structure for Z-tag dongle data record is seen in Figure 6-3.

There are 6 different type of commands which the BUR understands and executes:

00 - Start/Resume BUR console. After this command BUR will drop into console mode and no more data from Z-tag dongle is fetched nor processed. Note, that by default there is no Serial output defined for BUR, so command 02 (Serial Console Mode) must be executed before or after command 00.

;db	07Fh,0F0h,055h	; command start ID
;db	001h	; command code
;db	019h,099h,010h,021h	; BCD date = 1999.10.21
;db	023h,059h	; BCD time = 23:59
;db	006h	; description string length
;db	'Sample'	; description string (23 bytes max!)
;db	001h,030h	; BCD version = 1.30
;dd	00000100	; body length
;db	body_length dup (?)	; body
;dw	E2FC	; 16-bit CRC of body

Figure 6-3 Dongle Data Record

01 - Upload and execute code. This function finds "best fit" available memory location and uploads code specified in command body from Z-tag dongle there. It executes the data after loading and when the executable returns with RETF instruction, resumes data fetch from Z-tag.

02 - Serial console mode. When fetching data from Z-tag, there can or can not be data display to serial port. By default, the output is disabled. This command allows to enable the data outputting. The console setting remains selected until next execution of this command, so this command should be executed only for changing/disabling the output device.

Note, that there is a special serial console mode, what enables data redirection to Z-tag LED pins. This way the terminal console can be used without COM1 port usage.

03 - Execute console command line. This may be useful, if some kind of command scripting is used at board debugging. You can specify the command line and BUR goes and executes it on ZFiX console. If command results should be displayed through serial port, the command 02 must be executed first.

04 - Add command to a console. This function creates the internal command for BUR console, so users can specify new commands they need at the debugging process and upload them using Z-tag dongle. If you type "help", the new command definition can be seen as well (the help line is defined at command definition data).

05 - Stop. This will just light up the GREEN LED on Z-tag dongle and freeze BUR. May be useful to place after everything else to notify operator, that everything is done and prevent infinite execution of data fetch/exec procedure.

FF - Basket. The command code 0FFh is reserved by developers and dedicated for generic payload data, if needed (like BIOS images etc.).

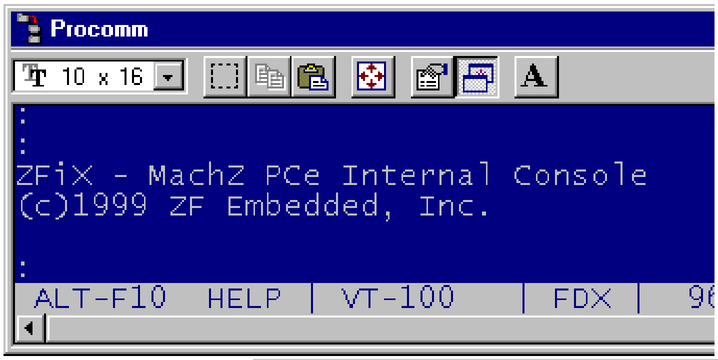
Any other command code is ignored and BUR execution continues without interruption.

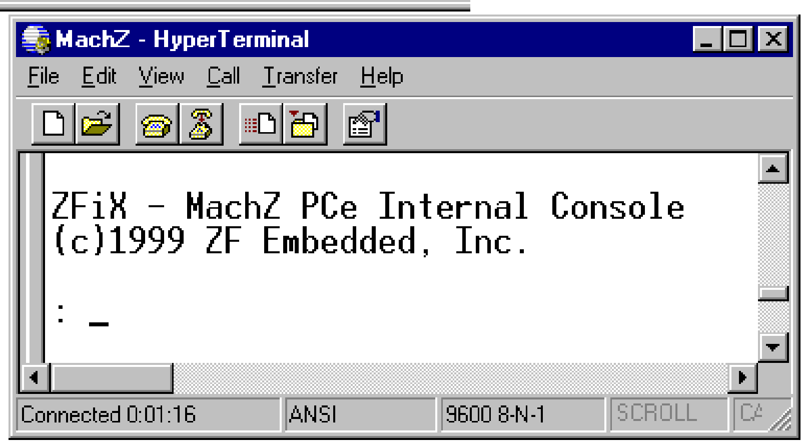
6.5.3. Internal Functionality

There is some support for programmers, who implement the BUR Extensions (for example the flash programming routines. At the end of BUR, there is 80-byte area with the pointers to useful functions and tables inside BUR what can be re-used by BUR Extension code to perform checksum calculations, serial I/O, timer and various other tasks. Between all BUR versions, this table location and structure remains same to maintain future compatibility. The detailed descriptions about this functionality can be found in 'BUR API' on page 578

6.6. BUR COM1 Download Examples

The two examples following show how to download a test program using Procomm or HyperTerminal via COM1. These little test programs which run entirely on the ZFx86 space are called the BUR Extensions.

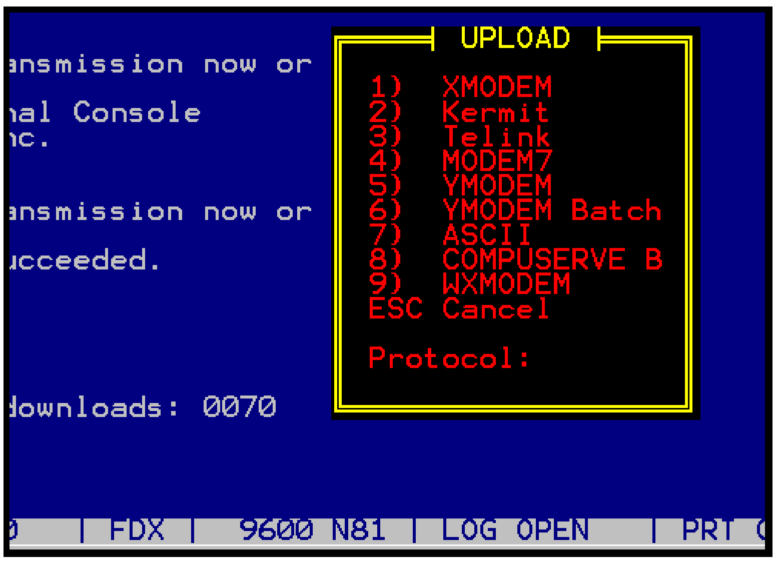




6.6.1. Procomm: Download a Test Program

1. Connect Null Modem Cable
2. Set **Procomm** 2.4.2 to the following:
   - 9600 baud
   - 8 bits
   - no parity
   - 1 stop bit
3. Press the reset push button on the board.
4. Use **DLS** to find available memory. Reference Table 6.7 "ZFix Console Commands". See also Table 6.8 "On-Chip RAM Assignment in BUR"
5. Use the **yload** command to download. In order to do the file transfer, see Figure 6-4 "Using Procomm YMODEM Batch"
6. Run the program using the “g” command.

```
ZFiX - ZFx86 PCe Internal Console
(c)1999 ZF Micro Devices, Inc.
: DLS                                               // Display available segment for
 0070                                                  downloads
: yload 70:0
Please start YModem transmission now or press <ESC>   // see Figure 6-4 "Using Procomm
CCCCCCCCCCCC ←waiting for xfr                          YMODEM Batch" on page 461
YModem data transfer succeeded.
: g 70:0
```



Press PgUp in Procomm to bring up the "UPLOAD" dialog. Select YMODEM Batch.

Then enter the name of the .com file to download:

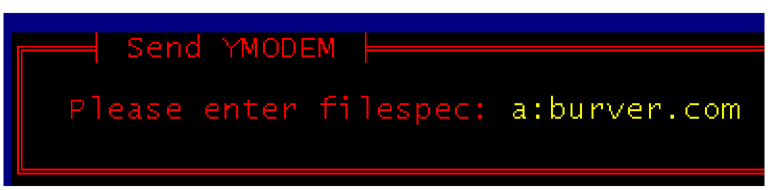


Figure 6-4 Using Procomm YMODEM Batch

6.6.2. HyperTerminal: Download a Test Program

1. Connect Null Modem Cable

2. Set **HyperTerminal** to the following:

   - 9600 baud
   - 8 bits
   - no parity
   - 1 stop bit

3. Press the reset push button on the board

4. Use **DLS** to find available memory. Reference Table 6.7 "ZFix Console Commands".

5. Use the **yload** command to download. In order to do the file transfer, see Figure 6-5 "Using HyperTerminal - Send File Ymodem"

6. Run the program using the “g” command.

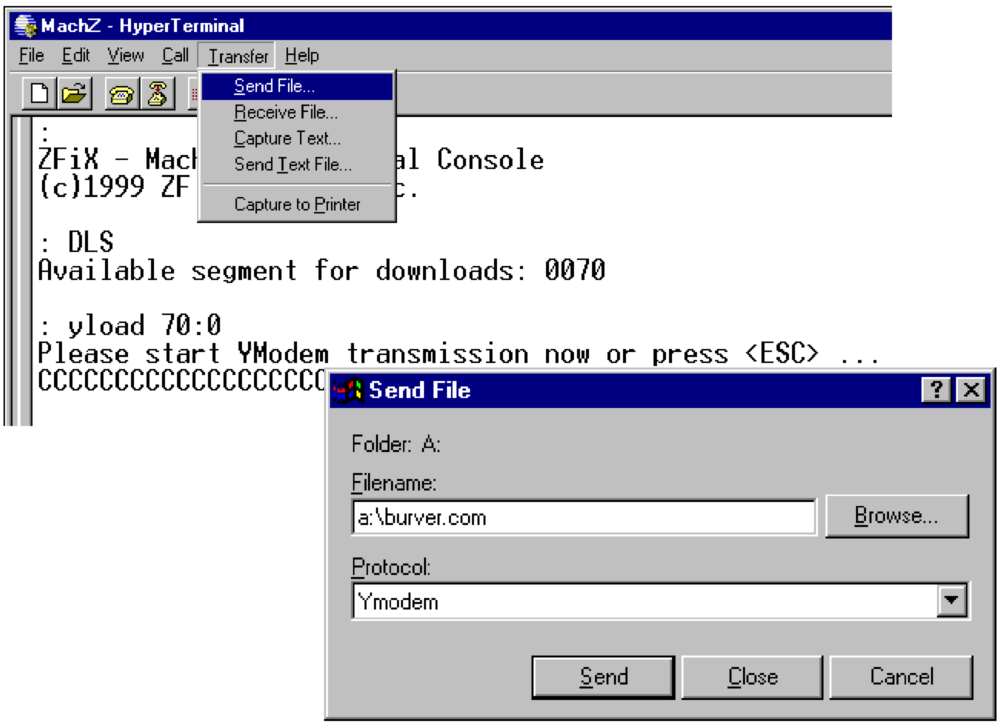


Figure 6-5 Using HyperTerminal - Send File Ymodem

6.6.3. BUR Version Test Program Source Code

```
;; Copyright 2000 ZF Micro Devices, Inc. All rights reserved.
title ZFx86 Bur Extension Code sample Obtains BUR Version Number

.486
burrom          segment USE16 at 0f000h
; Services table. These are the function pointers for
; uploaded code use.
                org     0ff00h  ; f000:ff00 in BUR ROM
Bur_Version     db      ?

                org     0ff0ah
CRLF            label   far     ; call ff00:ff0a --> CR/LF to COM1
                org     0ff22h
SerOut16        label   far     ; call ff00:ff22 --> AX to COM1 as decimal
                org     0ff3ah
SerSend         label   far     ; call ff00:ff3a --> Charout to COM1
burrom          ends

CODE            segment USE16 'CODE'
                assume cs:code

START:
                push    cs
                pop     es
                mov     di,offset VerText       ; ES:DI - text to show
                xor     cx,cx                   ; display until 0 reached
                call    SerSend

                les     bx,psBur_Version        ; es:bx --> Bur_Version String
                mov     ax,es:[bx]

                call    SerOut16                ; display AX to COM1 as decimal
                call    CRLF                    ; CR/LF to COM1

                retf                            ; resume with BUR

psBur_Version   dd      Bur_Version             ; define string pointer
VerText         db      'BUR Version: ',0

CODE            ends
                end     START
```

6.6.4. BUR/BET Memory Map

Table 6.8 On-Chip RAM Assignment in BUR

Address Range	Assignment
00000h-0007Fh (000h:0000h-0000h:007Fh)	IRQ vector space (128 bytes)
00080h-002FFh (000h:0080h-0000h:02FFh)	System variables (640 bytes) -- Reserved for BUR
00300h-006FFh (030h:0000h-0030h:03FFh)	stack (1024 bytes)
00700h-01FFFh (070h:0000h-0070h:18FFh)	generic code/data space (6400 bytes)

6.7. Flash Programming Example

```
;
; On-board flash programmer on ZFx86 BUR environment for AT49F516 chip
;
; (c)2000 ZF Micro Devices, Inc.
;
; Target Chip:Atmel AT49F516
; Size:                    64K
; Chipselect: ms_cs0
; Mode:                    16-bit
; Chip page:  00000000h
; Window base:E0000h
; Window size:10000h (64K)
;
; This code executes as BUR "Load and execute" function. It will fetch
; payload code following to the executable code in dongle.
;

.MODEL TINY
.CODE
.486p

                           org          0
START:
                           push         cs
                           pop          es

                           mov          di, offset HelloText; ES:DI - text to show
                           xor          cx, cx       ; display until 0 reached
                           call         SerSend
                           call         CRLF
                           call         ZTPrepareRead

                           ; Fetch header now
```

```
                    mov         cx, 4
GetSig:
                    shl         eax, 8
                    call        ZTRead
                    loop        GetSig

                    cmp         eax, 7FF055FFh          ; is that a basket?
                    jz          @f
                    mov         di, offset NoPayloadText
                    xor         cx, cx
                    call        SerSend
                    call        CRLF
                    jmp         ExitPgmrFail
@@:
                    ; Loose date and time

                    mov         cx, 6
DiscardTime:
                    call        ZTRead
                    loop        DiscardTime

                    ; Output basket name

                    mov         di, offset WritingText
                    xor         cx, cx
                    call        SerSend

                    call        ZTRead
                    xor         cx, cx
                    mov         cl, al                          ;
string length
ShowDesc:
                    call        ZTRead
                    call        SerSend2
                    loop        ShowDesc

                    ; Show version

                    mov         al, ' '
                    call        SerSend2
                    mov         al, 'V'
                    call        SerSend2
                    call        ZTRead
                    call        SerOut8
                    mov         al, '.'
                    call        SerSend2
                    call        ZTRead
                    call        SerOut8

                    ; Show size

                    mov         di, offset SizeText
                    xor         cx, cx
                    call        SerSend
```

```
                mov             cx, 4
ReadSize:
                shl             eax, 8
                call            ZTRead
                loop            ReadSize

                bswap           eax             ; convert bytes to double
                push            eax             ; save basket byte count

                call            SerOut32

                mov             di, offset BytesText
                xor             cx, cx
                call            SerSend
                call            CRLF

                ; OK, bells and whistles are there. Do programming now.

                ; Create Memory window 0 from E0000 to FFFFF

                mov             dx, 218h
                mov             al, 26h
                out             dx, al
                mov             dx, 21Ah
                mov             eax, 0E0000h ; set window base to E0000h
                out             dx, eax

                ; Set memwin0 size to 64K

                mov             dx, 218h
                mov             al, 2Ah
                out             dx, al
                mov             dx, 21Ah
                mov             eax, 10000h-1             ; create 64K window
                out             dx, eax

                ; Set chip page to 0

                mov             dx, 218h
                mov             al, 2Eh
                out             dx, al
                mov             dx, 21Ah
                mov             eax, 0h         ; set page to the beginning of
                out             dx, eax         ;  the chip

                ; Set chip width to 16-bit and read-write mode

                mov             dx, 218h
                mov             al, 5Ah
                out             dx, al
                mov             dx, 219h
                in              al, dx
                and             al, 11101110b
                out             dx, al
```

```
; Turn ISA bus speed as slow as possible, since we do not know what the
; SYSCLK is and we can not program device well if we do it too fast

                          pushf
                          cli
                          mov          eax, 80009050h; PIT Control/ISA Clock divider
                          mov          dx, 0CF8h
                          out          dx, eax
                          mov          dx, 0CFCh
                          in           al, dx
                          or           al, 00000111b ; divide by 8
                          out          dx, al
                          popf

                          ; Now we have chip layed out from E000:0 to E000:FFFF
                          ; Time to program some flash.

                          ; Check, whenever or not we have AT49F516 part on-board

                          mov          al, 90h                    ; get ID
                          call         FlashCMD16

                          push         0E000h
                          pop          ds
                          xor          si, si
                          mov          eax, ds:[si]               ; get chip ID to EAX

                          mov          ebx, eax                   ; save ID

                          mov          di, offset MfgText
                          xor          cx, cx
                          call         SerSend
                          call         SerOut8
                          mov          di, offset DevIDText
                          xor          cx, cx
                          call         SerSend
                          shr          eax, 16
                          call         SerOut8
                          call         CRLF

                          mov          al, 0F0h
                          call         FlashCMD16   ; end identification mode

                          pop          ecx          ; restore bytes count (was EAX
originally)

                          and          ebx, 0FFFCFFFFh ; strip ID two low bits, since
this may be the minor version code
                          cmp          ebx, 00084001Fh ; check, if it's AT49F516
                          jz           @f

                          mov          di, offset WrongDevText
                          xor          cx, cx
                          call         SerSend
                          call         CRLF
```

```
                    jmp         ExitPgmrFail

@@:
                    ; We have right chip. Erase device now

                    push        ecx

                    mov         di, offset ErasingText
                    xor         cx, cx
                    call        SerSend

                    mov         al, 80h      ; chip erase command part 1
                    call        FlashCmd16
                    mov         al, 10h      ; chip erase command part 2
                    call        FlashCmd16

                    ; Erase is now in progress. Check, when we are ready.

                    push        0E000h
                    pop         fs
                    xor         si, si
                    call        DSBX2Var
                    mov         ax, 182
                    mov         ds:[bx.CountDown], ax; gives 10sec. timeout

                    mov         dx, 218h
                    mov         al, 05Eh ;ZT_SIG_OUT
                    out         dx, al
                    inc         dx
                    in          al, dx
                    and         al, 11110101b ; disable LED's
                    out         dx, al
                    call        ZTPrepareRead ; ZFLogic back to track
@@:
                    cmp         word ptr ds:[bx.CountDown], 0
                    jz          @f
                    cmp         dword ptr fs:[si], 0FFFFFFFFh
                    jnz         @b ; loop until erase completes
@@:
; reset timer, so we will not loose our blinking LED's

                    mov         word ptr ds:[bx.CountDown], 0
                    cmp         dword ptr fs:[si], 0FFFFFFFFh
                    jz          @f

                    mov         di, offset FailedText
                    xor         cx, cx
                    call        SerSend
                    jmp         ExitPgmrFail
@@:
                    mov         di, offset OkText
                    xor         cx, cx
                    call        SerSend

                    ; All set. Programming ...
```

```
                call        ResetCRC

                mov         di, offset PgmText
                xor         cx, cx
                call        SerSend

                pop         ecx         ; restore byte count in basket
                cmp         ecx, 0      ; don't do anything
                jz          PgmDone     ;  if basket size is 0

                shr         ecx, 1      ; divide by two to get word count

                push        0E000h
                pop         ds
                xor         si, si

ProgramLoop:
                mov         al, 0A0h    ; "program word" command
                call        FlashCmd16
                call        ZTRead
                shl         ax, 8
                call        ZTRead
                xchg        al, ah      ; byte order for 16-bit writing
to memory
                mov         ds:[si], ax
                mov         bx, ax

; in the extreme case we can have PCI backside clock 80Mhz. ISA
; divider is 8, so we have 10M ISA bus clock. Programming cycle can be max.
; 50us, so we need to wait here about 500 ISA cycles to kill that time.

                push        cx
                mov         cx, 500
WaitWriting:
                in          al, 80h     ; create one ISA cycle
                cmp         ds:[si], bx ; see, if we have byte ready
                loopnz      WaitWriting
                pop         cx

                mov         ax, ds:[si] ; read back the word we
                int         17h         ;  programmed
                                        ; update CRC
                shr         ax, 8
                int         17h
                add         si, 2
                cmp         ds:[si-2], bx
                loopz       ProgramLoop
PgmDone:
                cmp         ds:[si-2], bx             ; see, why we exited
                jz          @f                                     ; Last
byte is OK, so it's because all is done

                mov         di, offset FailedText
                xor         cx, cx
```

```
                    call        SerSend
                    jmp         ExitPgmrFail
@@:
                    mov         di, offset OkText
                    xor         cx, cx
                    call        SerSend

                    ; Get original checksum

                    call        ZTRead
                    shl         ax, 8
                    call        ZTRead
                    xchg        al, ah

                    call        DSBX2Var     ; get variables block to DS:BX
                    cmp         word ptr ds:[bx.YModemCRChi_C], ax
                    jz          CRCOK

                    mov         di, offset CRCFailedText
                    xor         cx, cx
                    call        SerSend

                    call        SerOut16
                    mov         al, ' '
                    call        SerSend2
                    mov         ax, word ptr ds:[bx.YModemCRChi_C]
                    call        SerOut16
                    call        CRLF
                    jmp         ExitPgmrFail
CRCOK:
                    mov         di, offset CRCOkText
                    xor         cx, cx
                    call        SerSend
                    call        CRLF

                    jmp         ExitPgmrOk
ExitPgmrFail:
                    ; Lit up RED LED and do not do anything else!

                    mov         dx, 218h
                    mov         al, 05Eh ;ZT_SIG_OUT
                    out         dx, al
                    inc         dx
                    in          al, dx
                    and         al, 11110101b
                    or          al, ZT_LED_RED            ;00001000b
                    out         dx, al

                    jmp         $
ExitPgmrOk:
                    ; Lit up GREEN LED and continue with BUR
                    mov         dx, 218h
                    mov         al, 05Eh ;ZT_SIG_OUT
                    out         dx, al
                    inc         dx
```

```
                            in          al, dx
                            and         al, 11110101b
                            or          al, ZT_LED_GREEN;00000010b
                            out         dx, al
ExitPgmr:
                            call        CRLF

                            ; Set timer to maximum value. This is useful to prevent
                            ; BUR from blinking with LED's when loading next commands.
                            ; This way we maintain our RED or GREEN LED setting

                            call        DSBX2Var
                            mov         word ptr ds:[bx.CountDown], 0FFFFh

; Always exit with ZFL registers prepared for accelerated read!

                            call        ZTPrepareRead

                            retf                    ; resume with BUR

                            include ..\BURAPI.ASM

HelloText     db            0Dh, 0Ah
                            db          'PGM - ZFx86 AT49F516 Flash Programmer V0.80',
0Dh, 0Ah
                            db          '----------------------------------------------
', 0
NoPayloadText db            'No Payload. Exiting.', 0
WritingText   db            'Source: ', 0
SizeText      db            ' (0x', 0
BytesText     db            ' bytes)', 0
MfgText                     db          'Device: Mfg=',0
DevIDText     db            ' DevID=',0
WrongDevText  db            'This is not Atmel AT49F516', 0
OkText                      db          'OK!', 0Dh, 0Ah, 0
FailedText    db            'FAILED!', 0Dh, 0Ah, 0
ErasingText   db            'Erasing .. ', 0
PgmText                     db          'Programming .. ', 0
CRCFailedText db            'Data CRC failure: ', 0
CRCOkText     db            'Data CRC was OK!', 0


;
; Execute flash device command
;
; Entry: AL - command code
;
; Exit: none
;
; Uses: none
;

FlashCMD16    proc
                            push        ds
                            push        si
```

```
                push        ax

                push        0e000h
                pop         ds

        ; we have address bits shifted by one in 16-bit socket. It means,
        ; that control bytes should actually go one bit shifted to left, in
        ; order to get A0 right

                mov         al, 0AAh     ; CMD sequence part 1
                mov         si, 05555h
                shl         si, 1
                mov         ds:[si], al

                mov         al, 055h     ; CMD sequence part 2
                mov         si, 0AAAAh
                shl         si, 1
                mov         ds:[si], al

                pop         ax           ; restore command code

                mov         si, 05555h
                shl         si, 1
                mov         ds:[si], al

                pop         si
                pop         di
                ret
FlashCMD16    endp

END START
ENDS
```

7

7. Electrical Specifications

This chapter provides information about:

- General electrical specifications
- DC characteristics
- AC characteristics

All voltage values in this chapter are with respect to V_{SS} unless otherwise noted.

7.1. General Specifications

7.1.1. MTTF and FIT Specifications

The ZFx86's Mean Time To Failure (MTTF) and Failures In Time (FIT – Failures per billion operating device-hours) specifications are as follows.

Parameter	ZFx86
Technology	CMOS 8
Dynamic Op Life Temperature	125°C
Dynamic Op Life Voltage	3.2V
Typical Use Temperature and Voltage	60°C; 2.5V
Mean Time To Failure	21,129,173.19 hours

7.1.2. Power/Ground Connections and Decoupling

When testing and operating the ZFx86 device, use standard high frequency techniques to reduce parasitic effects. For example:

- Filter the DC power leads with low-inductance decoupling capacitors.
- Use low-impedance wiring.
- Utilize the POWER and GND pins.

Absolute Maximum Ratings

Stresses beyond those indicated in the following table may cause permanent damage to the ZFx86 device, reduce device reliability and result in premature failure, even when there is no immediately apparent sign of failure. Prolonged exposure to conditions at or near the absolute maximum ratings may also result in reduced device life span and reduced reliability.

Note: The values in the following table are stress ratings only. They do not imply that operation under other conditions is impossible.

Table 7.1 Absolute Maximum Ratings

Parameter	Min	Max
Operating case temperature[1, 2]	-40	85°C
Storage temperature[3]	-45	125°C
Supply voltage		Maximum supply voltage is as indicated. See Table 7.2.
Voltage on: -5V tolerance pins -others	 -0.5 -0.5	 6.0V 4.2V
Input clamp current, I_{IK}[1]		10mA
Output clamp current, I_{OK}[1]		25mA

1. Power applied
2. Temperature range industrial
3. No bias

Table 7.2 Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Typ	Max
T_C	Operating Case Temperature				
	Extended Temperature Range ZFx86	33/66/100 MHz[1]	–40°C	—	85°C
	Standard ZFx86	128 MHz[2]	–0°C	—	70°C
V_{BAT}	Battery Supply Voltage. Powers RTC	—	2.85V	3.0V	3.15V
$V_{dd-Core}$	Core Processor (CPU) Power Supply	—			
	Extended Temperature Range ZFx86		2.14V	2.25V	2.36V
	Standard ZFx86		2.65V	2.70V	2.84V
$V_{dd-I/O}$	I/O Buffer Power Supply	—	3.14V	3.30V	3.46V

1. Temperature range industrial
2. Temperature range commercial

Table 7.3 Current Consumption

Symbol	Parameter	Conditions	Max
$I_{dd-Core}$	Core Supply Current	33 MHz sysclk with 3X CPU	380mA
$I_{dd-Core}$	Core Supply Current	Suspend mode with external clocks running	120mA
$I_{dd-Core}$	Core Supply Current	Suspend mode with external clocks off	45mA
I_{BAT}	V_{BAT} Battery Supply Current	V_{BAT} = 3V other supplies at 0V	2.0µA
$I_{dd-I/O}$	$V_{dd-I/O}$ Power Supply Current	Suspend mode	TBD
$I_{dd-I/O}$	$V_{dd-I/O}$ Power Supply Current	Generic OS	TBD

Table 7.4 Pin Capacitance and Inductance

Symbol	Parameter	Min	Typ	Max
C_{IN}[1]	Input Pin Capacitance	TBD pF	TBD pF	TBD pF
C_{IN}[1]	Clock Input Capacitance	TBD pF	TBD pF	TBD pF
$C_{I/O}$[1]	I/O Pin Capacitance	TBD pF	TBD pF	TBD pF
C_O[1]	Output Pin Capacitance	TBD pF	TBD pF	TBD pF
L_{PIN}[2]	Pin Inductance	TBD nH	TBD nH	TBD nH

1. T_A = 25°C, f = 1 MHz. All capacitances not 100% tested.
2. Not 100% tested.

7.2. Signal I/O Buffer Type Directory

The table shown below describes the various buffer types which are used for signals of the ZFx86 device. Immediately following this table is another table which lists all the signals according to logical group, and the buffer types which are relevant for each signal.

Table 7.5 I/O Cell Characteristics

TYPE	IOH	IOL	AC L=>H	AC H=>L	COMMENT
Generic2 I/O, OE, IE	-8ma	10ma	10ns buf delay .4/2.4 V 50pf	10ns buf delay 2.4/.4 V 50pf	RENU (160mic amps) REND (036mic amps) input hysteresis 250mv. For ISA Bus.
MIDE I/O, OE, IE	-3ma	5ma	15/28ns buf delay .4/2.4 V 75pf / 150pf	15/28ns buf delay 2./.4 V 75pf / 150pf	No RENU or REND Input hysteresis 200mv For IDE interface
MAC97 I/O, OE, IE	-2ma	8ma	10ns buf delay .4/2.4 V 50pf	10ns buf delay 2.4/.4 V 50pf	RENU only input hysteresis For AC97 and Open Drain
MPCI I/O, OE, IE	-1ma	1.5ma	5ns buf delay 50pf 32ma peak	5ns buf delay 50pf 38ma peak	RENU only Designed to meet PCI specification
MPCI_CLK I/O, OE, IE	—	—	5ns	5ns	RENU only For PCI CLK
M_FDC_PP I/O, OE, IE	-14ma	14ma	10ns buf delay .4/2.4 V 50pf	10ns buf delay 2.4/.4 V 50pf	RENU & REND. input hysteresis. floppy, parallel, and open drain
MVBAT ~ a wire	—	—	—	—	For vbat. Wire with poly resis
MUSB I/O, OE, IE	—	—	—	—	No RENU or REND USB power enable and over current
MWUSB just a wire	—	—	—	—	Connects pad to USB hard Macro
MMC_D I/O, OE, IE Slew cntrl JTAG Bound support	2ma	5ma	2v/ns 60pf load	2v/ns 60pf load	No RENU or REND No hysteresis For SDRAM data and address

Table 7.5 I/O Cell Characteristics (cont.)

TYPE	IOH	IOL	AC L=>H	AC H=>L	COMMENT
MMC_SDCLK Out & OE No slew. JTAG Bound support	2ma	5ma	1.7v/ns 60pf load	1.7v/ns 60pf load	No RENU or REND For SDRAM output clocks
MMC_SDCLKIN Input only. No IE. JTAG Bound support	—	—	—	—	No RENU or REND No hysteresis SDRAM or other high speed clock input

Note: See further information in the Cell Types in 7.3. "Detailed DC Characteristics of Cells" on page 476.

7.3. Detailed DC Characteristics of Cells

Table 7.6 Input, MPCi

Symbol	Parameter	Conditions	Min	Max
V_{IH}	Input High Voltage	—	$0.5V_{IO}$	V_{IO}+0.5V[1]
V_{IL}	Input Low Voltage	—	-0.5V[1]	$0.3V_{IO}$
V_{IPU}	Input Pull-up Voltage[2]	—	$0.7V_{IO}$	—
I_{IL}	Input Leakage Current [2, 3]	$0 < V_{IN} < V_{IO}$	—	+/-10µA

1. Not 100% tested.
2. Not 100% tested. This parameter indicates the minimum voltage to which we calculate the pull-up resistors in order to pull a floated network.
3. Input leakage currents include hi-Z output leakage for all bidirectional buffers with TRI-STATE outputs.

Table 7.7 Input, Generic2

Symbol	Parameter	Conditions	Min	Max
V_{IH}	Input High Voltage	—	2.0V	V_{IO}+0.5V
V_{IL}	Input Low Voltage	—	-0.5V	0.8V
I_{IL}	Input Leakage Current	$V_{IN} = V_{IO}$	—	10µA
		$V_{IN} = V_{SS}$	—	-10µA
V_H	Input Hysteresis	—	250mV	—

Table 7.8 Input, MUSB

Symbol	Parameter	Conditions	Min	Max
V_{IH}	Input High Voltage	—	$0.5V_{IO}$	$V_{IO}+0.5V$
V_{IL}	Input Low Voltage	—	–0.5V	$0.3V_{IO}$
I_{IL}	Input Leakage Current	$V_{IN} = V_{I/O}$	—	10µA
		$V_{IN} = V_{SS}$	—	–10µA
V_{HIS}	Input Hysteresis		200mV	

Table 7.9 Input, MIDE

Symbol	Parameter	Conditions	Min	Max
V_{IH}	Input High Voltage	—	$0.5V_{I/O}$	$V_{IO}+0.5V$
V_{IL}	Input Low Voltage	—	–0.5V	$0.3V_{I/O}$
I_{IL}	Input Leakage Current	$V_{IN} = V_{I/O}$	—	10µA
		$V_{IN} = V_{SS}$	—	–10µA
V_{HIS}	Input Hysteresis	—	200mV	—

Table 7.10 Input, M-FDCP

Symbol	Parameter	Conditions	Min	Max
V_{IH}	Input High Voltage	—	$0.5V_{I/O}$	$V_{I/O}+0.5V$
V_{IL}	Input Low Voltage	—	–0.5V	$0.3V_{I/O}$
I_{IL}	Input Leakage Current	$V_{IN} = V_{I/O}$	—	10µA
		$V_{IN} = V_{SS}$	—	–10µA
V_{HIS}	Input Hysteresis	—	200mV	—

Table 7.11 Input, MMC-D

Symbol	Parameter	Conditions	Min	Max
V_{IH}	Input High Voltage	—	$0.5V_{I/O}$	$V_{I/O}+0.5V$
V_{IL}	Input Low Voltage	—	–0.5V	$0.3V_{I/O}$
I_{IL}	Input Leakage Current	$V_{IN} = V_{I/O}$	—	10µA
		$V_{IN} = V_{SS}$	—	–10µA
V_{HIS}	Input Hysteresis	—	200mV	—

Table 7.12 Input, MWUSB

Symbol	Parameter	Conditions	Min	Max
V_{IH}	Input High Voltage	—	$0.5V_{I/O}$	$V_{I/O}+0.5V$[1]
V_{IL}	Input Low Voltage	—	$-0.5V$[1]	$0.3V_{I/O}$
I_{IL}	Input Leakage Current	$V_{IN} = V_{I/O}$	—	10µA
		$V_{IN} = V_{SS}$	—	-10µA
V_{HIS}	Input Hysteresis[1]	—	200mV	—
V_{DI}	Differential Input Sensitivity	|(D+)-(D-)| and Figure 7-1	0.2V	—
V_{CM}	Differential Common Mode Range	Includes V_{DI} Range	0.8V	2.5V
V_{SE}	Single Ended Receiver Threshold	—	0.8V	2.0V

1. Not 100% tested.

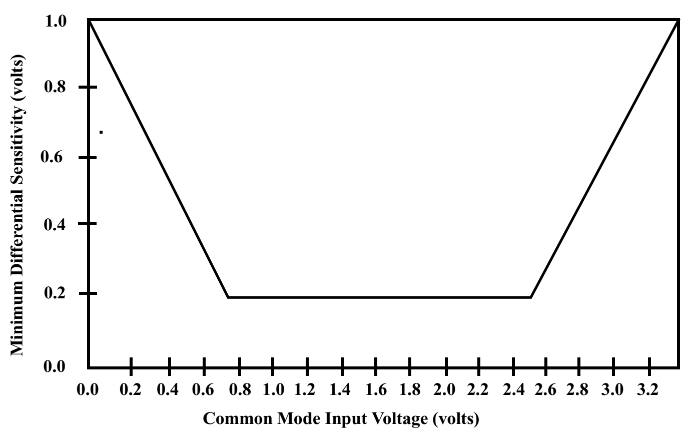


Figure 7-1 Differential Input Sensitivity for Common Mode Range

Table 7.13 Input, MAC97[1]

Symbol	Parameter	Conditions	Min	Max
V_{IH}	Input High Voltage	—	1.4V	—
V_{IL}	Input Low Voltage	—	$-0.5V$[2]	0.8V
I_{IL}	Input Leakage Current	$V_{IN}=V_{I/O}$	—	10µA
		$V_{IN}=V_{SS}$	—	-10µA
V_{HIS}	Input hysteresis	—	200mV	—

1. Buffer Type: IN_{AB}
2. Not 100% tested.

Table 7.14 Output, PCI TRI-STATE Buffer

Symbol	Parameter	Conditions	Min	Max
V_{OH}	Output High Voltage	I_{OH} = -500 µA	$0.9V_{I/O}$	—
V_{OL}	Output Low Voltage	I_{OL} =1500 µA	—	$0.1V_{IO}$
I_{PU}	Pull-up current	Output short to ground	mA	mA

Table 7.15 Output, GENERIC 2[1]

Symbol	Parameter	Conditions	Min	Max
V_{OH}	Output High Voltage	I_{OH} = *-8* mA	2.4V	—
V_{OL}	Output Low Voltage	I_{OL} = *10* mA	—	0.4V
I_{PU}	Pull-up current	Output short to ground	mA	mA
I_{PD}	Pull-down current	Output short to $V_{I/O}$	mA	mA

1. Output, TRI-STATE buffer capable of sourcing I/OH mA and sinking I/OL mA

Table 7.16 Output, MIDE

Symbol	Parameter	Conditions	Min	Max
V_{OH}	Output High Voltage	I_{OH} = -3 mA	2.4V	—
V_{OL}	Output Low Voltage	I_{OL} = 5 mA	—	0.4V

Table 7.17 Output, MUSB

Symbol	Parameter	Conditions	Min	Max
V_{OH}	Output High Voltage	I_{OH} = -*?* mA	2.4V	—
V_{OL}	Output Low Voltage	I_{OL} = *?* mA	—	0.4V

Table 7.18 Output, M-FDC_PP

Symbol	Parameter	Conditions	Min	Max
V_{OH}	Output High Voltage	I_{OH} = -14 mA	2.4V	—
V_{OL}	Output Low Voltage	I_{OL} = 14 mA	—	0.4V
I_{PU}	Pull-up current	Output short to ground	mA	mA
I_{PD}	Pull-down current	Output short to V I/O	mA	mA

Table 7.19 Output, MMC_D

Symbol	Parameter	Conditions	Min	Max
V_{OH}	Output High Voltage	I_{OH} = -2 mA	2.4V	—
V_{OL}	Output Low Voltage	I_{OL} = 5 mA	—	0.4V

.

Table 7.20 Output, MWUSB[1]

Symbol	Parameter	Conditions	Min	Max
V_{USB_OH}	High-level output voltage	I_{OH} = -0.25 mA R_L = 15 KΩ to GND	2.8V	3.6V[2]
V_{USB_OL}	Low-level output voltage	I_{OL} = 2.5 mA R_L = 1.5 KΩ to 3.6V	—	0.3V
t_{USB_CRS}	Output signal crossover voltage	—	1.3V	2.0V

1. Buffer Type: O_{USB}
2. Tested by characterization.

Table 7.21 Output, MAC97[1]

Symbol	Parameter	Conditions	Min	Max
V_{OH}	Output High Voltage	I_{OH} = - 2 mA	$0.9V_{I/O}$	—
V_{OL}	Output Low Voltage	I_{OL} = 8 mA	—	$0.1V_{I/O}$
I_{PU}	Pull-up current	Output short to ground	mA	mA

1. Buffer Type: O_{MAC97I}

7.4. AC Characteristics

The tables in this section list the following AC characteristics:

- Output delays
- Input setup requirements
- Input hold requirements
- Output float delays

The default levels for measurement of the rising clock edge reference voltage (V_{REF}), and other voltages are shown in the Table below. Input or output signals must cross these levels during testing. Unless otherwise specified, all measurement points in this section conform to these default levels.

Note: The following naming conventions are used in this section:
name1,2 = name1 or name2
name1/name2 = name1 or name2
namex1,2/namey1,2 = namex1, namex2, namey1, or namey2

Table 7.22 Default Levels for Measurement of Switching Parameters

Symbol	Parameter	Value	Reference
V_{REF}	Reference voltage	1.5V	
V_{IHD}	Input High Drive voltage	2.4V	Figure 7-2
V_{ILD}	Input Low Drive voltage	0.4V	Figure 7-2
V_{OHD}	Output High Drive voltage	2.4V	
V_{OLD}	Output Low Drive voltage	0.4V	

7.4.1. System Interface

Table 7.23 sysclk_c Clock Parameters

Symbol	Parameter	Min	Max
t CYC	Clock Cycle Time	500nS	15nS
t HIGH	Clock High Time	14nS	7nS
t LOW	Clock Low Time	14nS	7nS
	Clock Slew Rate	1V/nS	4V/ns

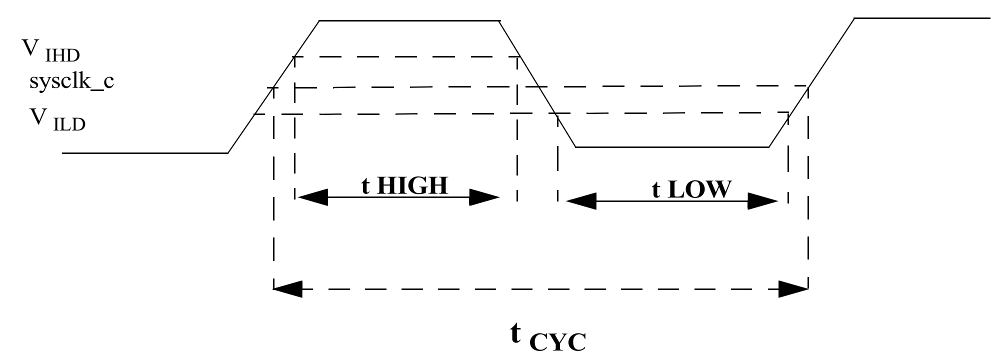
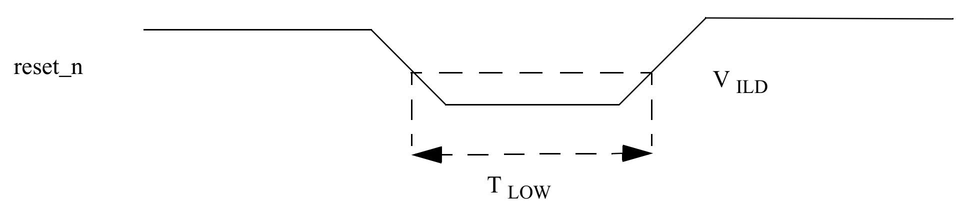
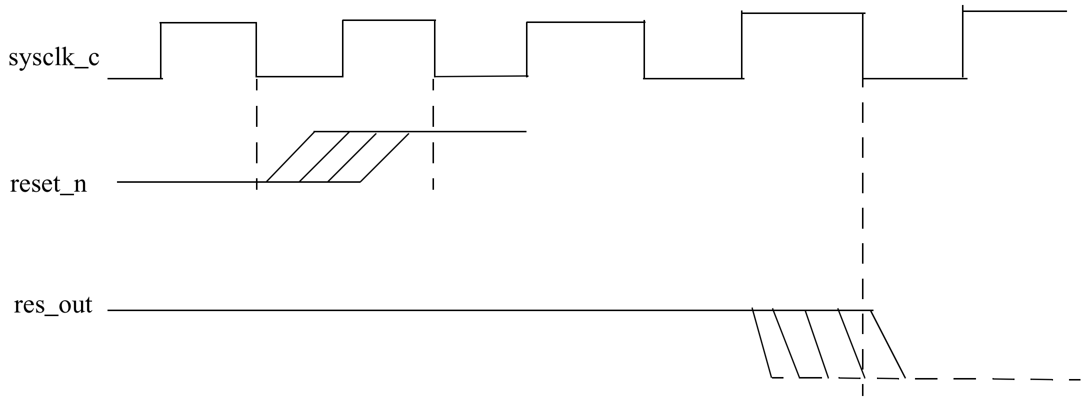


Figure 7-2 sysclk_c Timing and Measurement Points



reset_n may be asserted and de-asserted asynchronous to clock. T_{LOW} must be at least 15 nano seconds.

Figure 7-3 reset_n timing



pcirtst_n has the same timing as res_out but opposite in sign. Both res_out and pcirst_n are asserted when reset_n is asserted without regards to clock.Both de-assert on the high to low edge of clock.

Figure 7-4 res_out timing

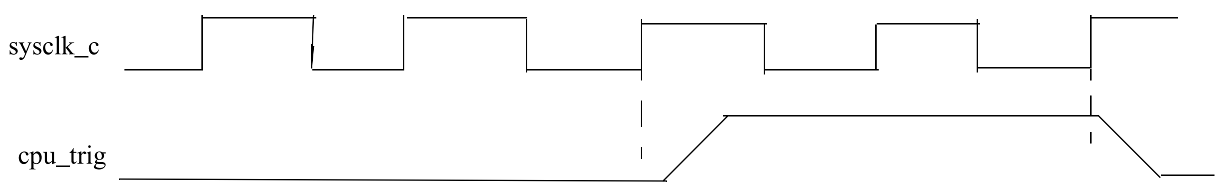


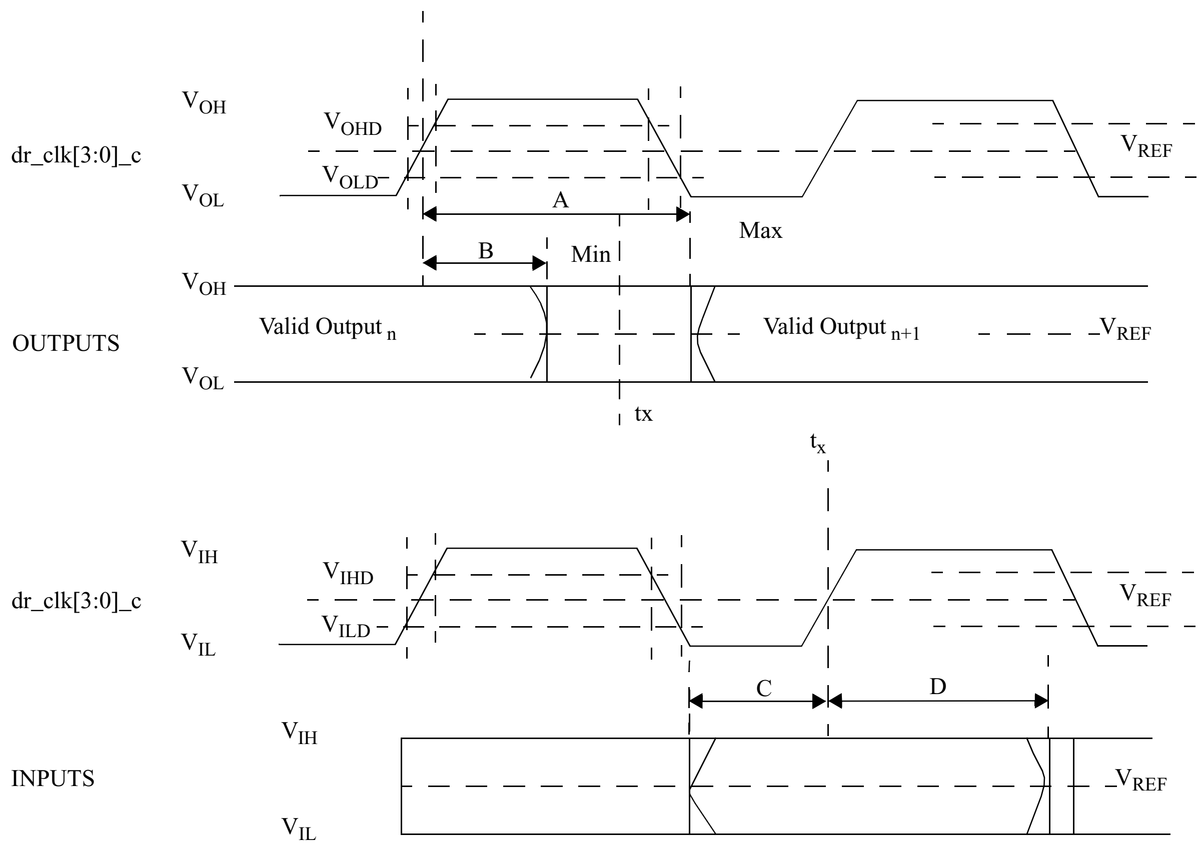
Figure 7-5 CPU_trig timing

7.4.2. Memory Interface

The Minimum Input setup and hold times described in Figure 7-6 (legend C and D) define the smallest acceptable sampling window during which a synchronous input signal must be stable to ensure correct operation.

All AC tests are as follows unless otherwise specified:

- $V_{I/O}$ = 3.0V to 3.6V (3.3V nominal)
- TC = 0 ^{o}C to 70 ^{o}C
- C_L = 50 pF



Legend: A = Maximum Output Delay
B = Minimum Output Delay
C = Minimum Input Setup
D = Minimum Input Hold

Figure 7-6 Drive Level and Measurement Points for Switching Characters

Table 7.24 SDRAM Interface Signals

Symbol	Parameter	Min	Max
t_1	Control Output [1] Valid from dr_clk	2.3nS	5.5nS
t_2	ma[3:0], Output Valid from dr_clk	2.0nS	4.4nS
t_3	d[31:0] Output Valid from dr_clk	2.4nS	5.9nS
t_4	d[31:0] Read Data in Setup to dr_clk	3.1nS	6.6nS
t_5	d[31:0] Read Data Hold to dr_clk	2.0nS	3.0nS
t_6	dr_clk cycle time	Same as sysclk_c	
t_7	dr_clk fall/rise time between (V_{OLD}-V_{OHD})	2.5nS	2.5nS

1. Control output includes all the following signals: dr_cs[3:0]_n, dr_msk[3:0]_n, dr_we0_n, dr_ras0_n,dr_cas0_n Load = 50pF, V_{CORE} = 2.5, $V_{I/O}$ = 3.3V, @25°C.

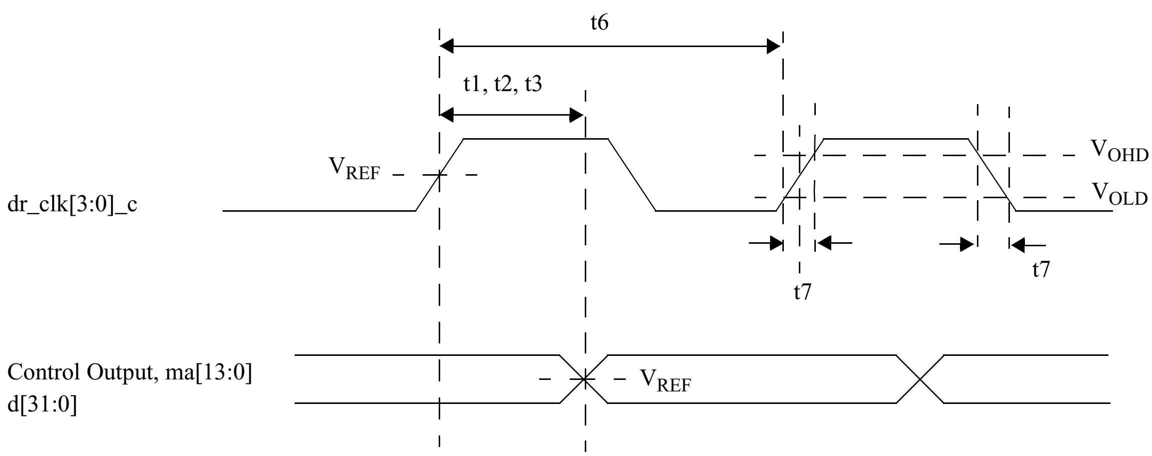


Figure 7-7 Output Valid Timing

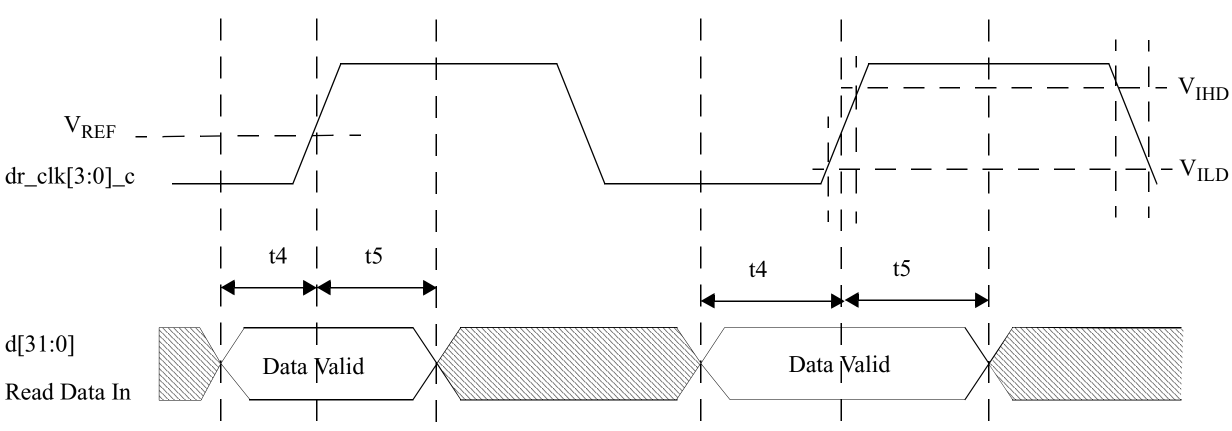


Figure 7-8 Setup and Hold Timing - Read Data In

7.4.3. ACCESS.bus Interface

1. All ACCESS.bus timing is not 100% tested. Timing is guaranteed by design.

Table 7.25 ACCESS.bus Interface

Symbol	Parameter	Min	Max	Reference
t_{SCLfi}	SCLK signal fall time		300nS	Figure 7-9
t_{SCLri}	SCLK signal rise time		1µs	Figure 7-9
t_{SDAfi}	SDAT signal fall time		300nS	Figure 7-9
t_{SDAri}	SDAT signal rise time		1µs	Figure 7-9

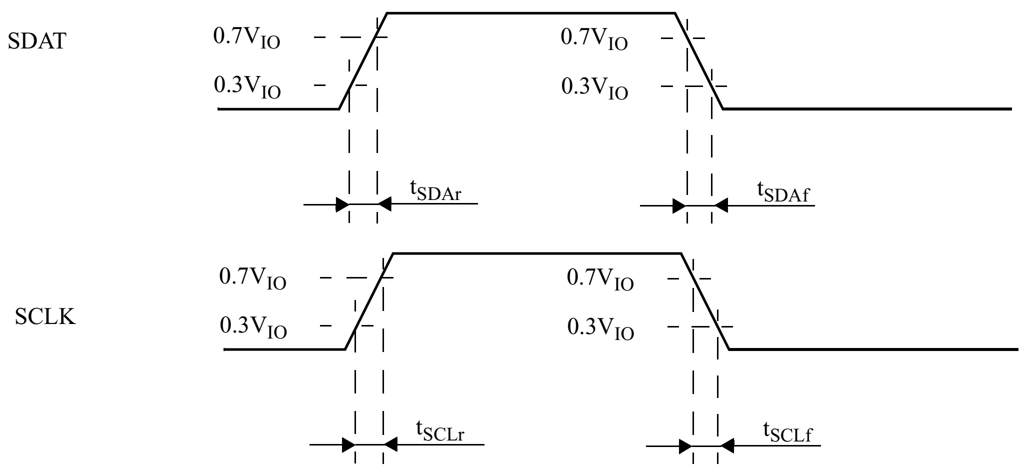


Figure 7-9 ACB Signals (SDAT AND SCLK) Rising and Falling times

7.4.4. PCI Bus

The SC1400B device is compliant with PCI Bus Rev. 2.1 specifications. Relevant information from the PCI Bus specifications is provided below.

The $\overline{PME}$ signal is compliant with PCI Bus Revision 2.2.

All parameters in the following table are not 100% tested.

Table 7.26 PCI Bus - AC Specifications

Symbol	Parameter	Condition	Min	Max	Unit
$I_{OH}(AC)$[1, 2]	Switching Current High	$0<V_{OUT}\leq 0.3V_{I/O}$	$-12V_{I/O}$	—	mA
		$0.3V_{I/O}<V_{OUT}<0.9V_{I/O}$	$-17.1(V_{I/O}-V_{OUT})$	—	mA
		$0.7V_{I/O}<V_{OUT}<V_{I/O}$	—	Equation A	
	Test Point[2]	$V_{OUT}=0.7V_{I/O}$	—	$-32V_{I/O}$	mA
$I_{OL}(AC)$[1]	Switching Current Low	$V_{I/O}>V_{OUT}$Š$\geq 0.6V_{I/O}$	$16V_{I/O}$	—	mA
		$0.6V_{I/O}>V_{OUT}>0.1V_{I/O}$[1]	$26.7V_{OUT}$	—	mA
		$0.18V_{I/O}>V_{OUT}>0$[1, 2]	—	Equation B	
	Test Point[2]	$V_{OUT}=0.18V_{I/O}$	—	$38V_{I/O}$	mA
I_{CL}	Low Clamp Current	$-3<V_{IN}\leq -1$	$-25+(V_{IN}+1)/0.015$	—	mA
I_{CH}	High Clamp Current	$V_{I/O}+4>V_{IN}>V_{I/O}+1$	$25+(V_{IN}-V_{I/O}-1)/0.015$	—	mA
$SLEW_R$[3]	Output Rise Slew Rate	$0.2V_{I/O}$ - 0.6 $V_{I/O}$ Load	1V/nS	4V/nS	V/nS
$SLEW_F$[3]	Output Fall Slew Rate	$0.6V_{I/O}$ - 0.2 $V_{I/O}$ Load	1V/nS	4V/nS	V/nS

1. Refer to the V/I curves in Figure 7-11 This specification does not apply to PCICLK0, PCICLK1, and $\overline{PCIRST}$ which are system outputs.
2. Maximum current requirements are met when drivers pull beyond the first step voltage. Equations which define these maximum values (A and B) are provided with relevant diagrams in Figure 7-11. These maximum values are guaranteed by design.
3. Rise slew rate does not apply to open-drain outputs. This parameter is interpreted as the cumulative edge rate across the specified range. According to the test circuit Figure 7-10.

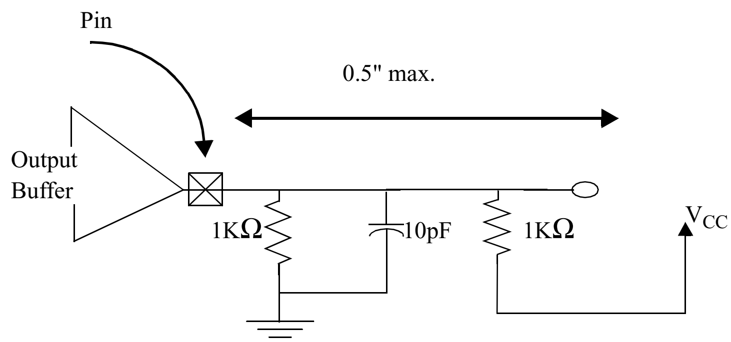
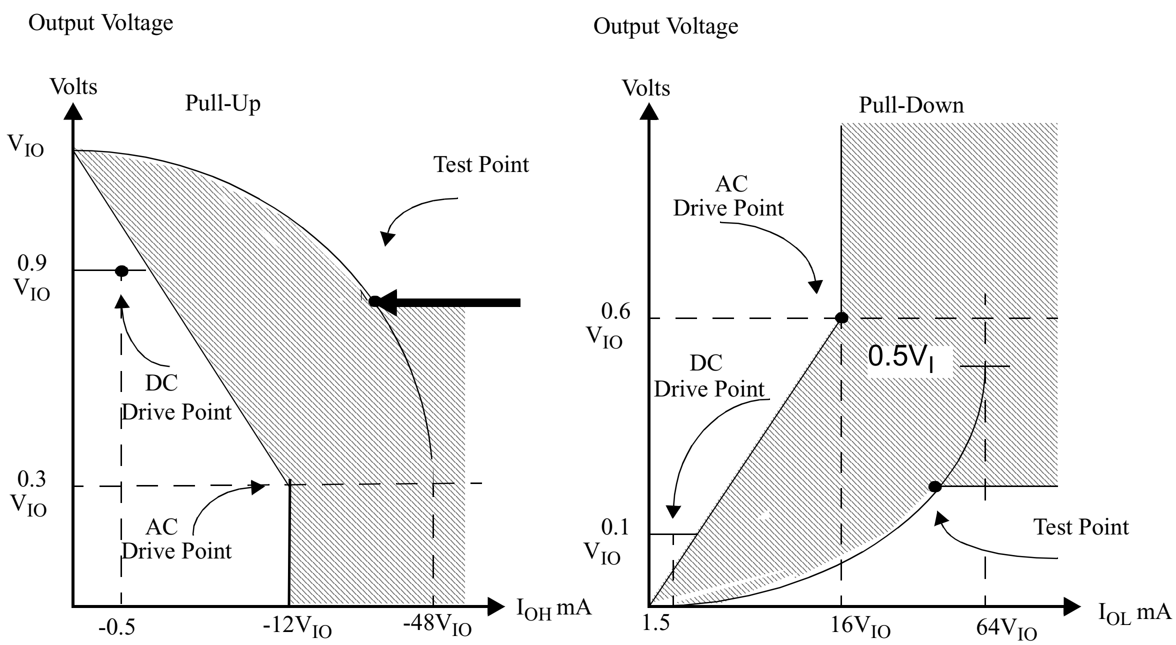


Figure 7-10 Testing Setup for Slew Rate and Minimum Timing



Equation A

$I_{oh}=(98.0/V_{IO})*(V_{OUT}-V_{IO})*(V_{OUT}+0.4V_{IO})$

for $V_{IO}>V_{OUT}>0.7V_{IO}$

Equation B

$I_{OL}=(256/V_{IO})*V_{OUT}*(V_{IO}-V_{OUT})$

for $0V<V_{OUT}<0.18V_{IO}$

Figure 7-11 V/I Curves for PCI Output Signals

Table 7.27 PCI Clock Parameters

Symbol	Parameter	Min	Max
t_{CYC}	PCICLK Cycle time[1]	30nS	—
t_{HIGH}	PCICLK High time[2]	11nS	—
t_{LOW}	PCICLK Low time[2]	11nS	—
$PCICLK_{sr}$	PCICLK Slew Rate[3]	1V/nS	4
$PCIRST_{sr}$	$\overline{\text{PCIRST}}$ Slew Rate[4]	50mV/nS	—

1. Clock frequency is between nominal DC and 33 MHz. Device operational parameters at frequencies under 16 MHz are not 100% tested. The clock can only be stopped in a low state.
2. Guaranteed by characterization.
3. Slew rate must be met across the minimum peak-to-peak portion of the clock waveform (see Figure 7-12).
4. The minimum $\overline{\text{PCIRST}}$ slew rate applies only to the rising (deassertion) edge of the reset signal. See Figure 7-12 for $\overline{\text{PCIRST}}$ timing.

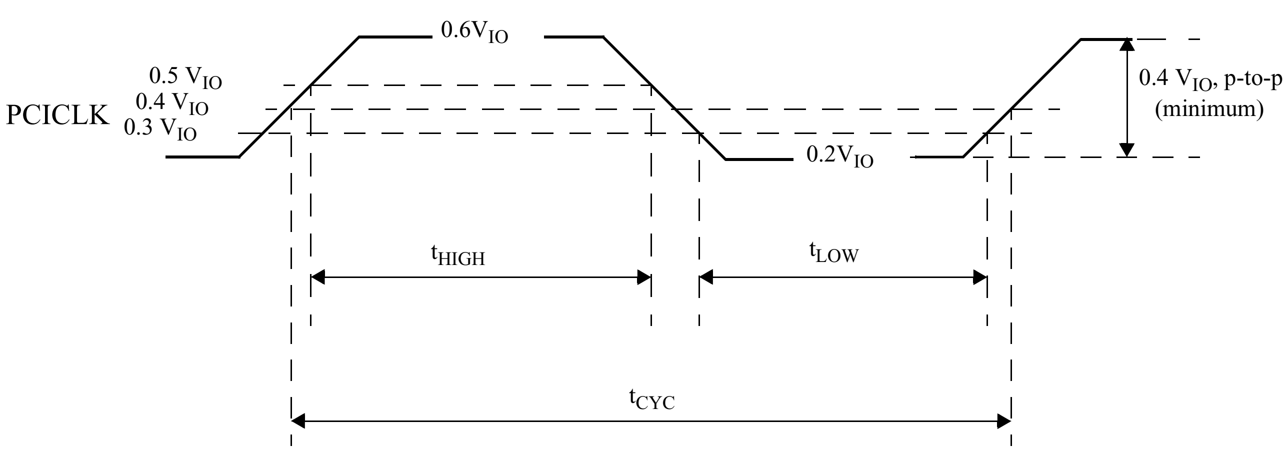


Figure 7-12 PCICLK Timing and Measurement Points

Table 7.28 PCI Bus Timing Parameters

Symbol	Parameter	Min	Max
t_{VAL}	PCICLK to Signal Valid Delay[1,2,4] (on the bus)	2nS	11nS
t_{VAL}(ptp)	PCICLK to Signal Valid Delay[1,2,4] (point-to-point)	2nS	12nS
t_{ON}	Float to Active Delay[1,3]	2nS	—
t_{OFF}	Active to Float Delay[1,3]	—	28nS
t_{SU}	Input Set up Time to PCICLK[4,5] (on the bus)	7nS	—
t_{SU}(ptp)	Input Set up Time to PCICLK[4,5] (point-to-point)	10nS, 12nS	—
t_H	Input Hold Time from PCICLK[5]	0nS	—
t_{RST}	$\overline{\text{PCIRST}}$ Active Time After Power Stable[6,3]	1mS	—
$t_{RST\text{-}CLK}$	$\overline{\text{PCIRST}}$ Active Time After PCICLK Stable[6, 3]	100µS	—
$t_{RST\text{-}OFF}$	$\overline{\text{PCIRST}}$ Active to Output Float Delay[3, 6, 7]	—	40nS

1. See the timing measurement conditions in Figure 7-10.
2. Minimum times are evaluated with same load used for slew rate measurement (as shown in Figure 7-10); maximum times are evaluated with the load circuits shown in Figure 7-13, for high-going and low-going edges respectively.
3. Not 100% tested.
4. $\overline{\text{REQ}}$ and $\overline{\text{GNT}}$ are point-to-point signals, and have different output valid delay and input setup times than do signals on the bus. $\overline{\text{GNT}}$ has a setup of 10; $\overline{\text{REQ}}$ has a setup of 12. All other signals are sent via the bus.
5. See the timing measurement conditions in Figure 7-15.
6. $\overline{\text{PCIRST}}$ is asserted and deasserted asynchronously with respect to PCICLK (see Figure 7-16).
7. All output drivers are asynchronously floated when $\overline{\text{PCIRST}}$ is active.

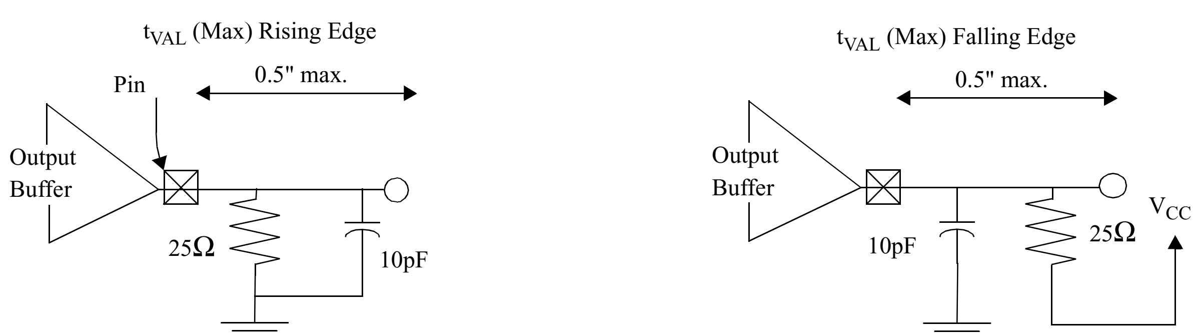


Figure 7-13 Load Circuits for Maximum Time Measurements

Measurement and Test Conditions

Table 7.29 Measurement Condition Parameters

Symbol	Value
V_{TH}[1]	0.6 $V_{I/O}$
V_{TL}[1]	0.2 $V_{I/O}$
V_{TEST}	0.4 $V_{I/O}$
V_{STEP} (rising edge)	0.285 $V_{I/O}$
V_{STEP} (falling edge)	0.615 $V_{I/O}$
V_{MAX}[2]	0.4 $V_{I/O}$
Input Signal Edge Rate	1V/nS

1. The input test is performed with 0.1 $V_{I/O}$ of overdrive. Timing parameters must not exceed this overdrive.
2. V_{MAX} specifies the maximum peak-to-peak waveform allowed for measuring input timing.

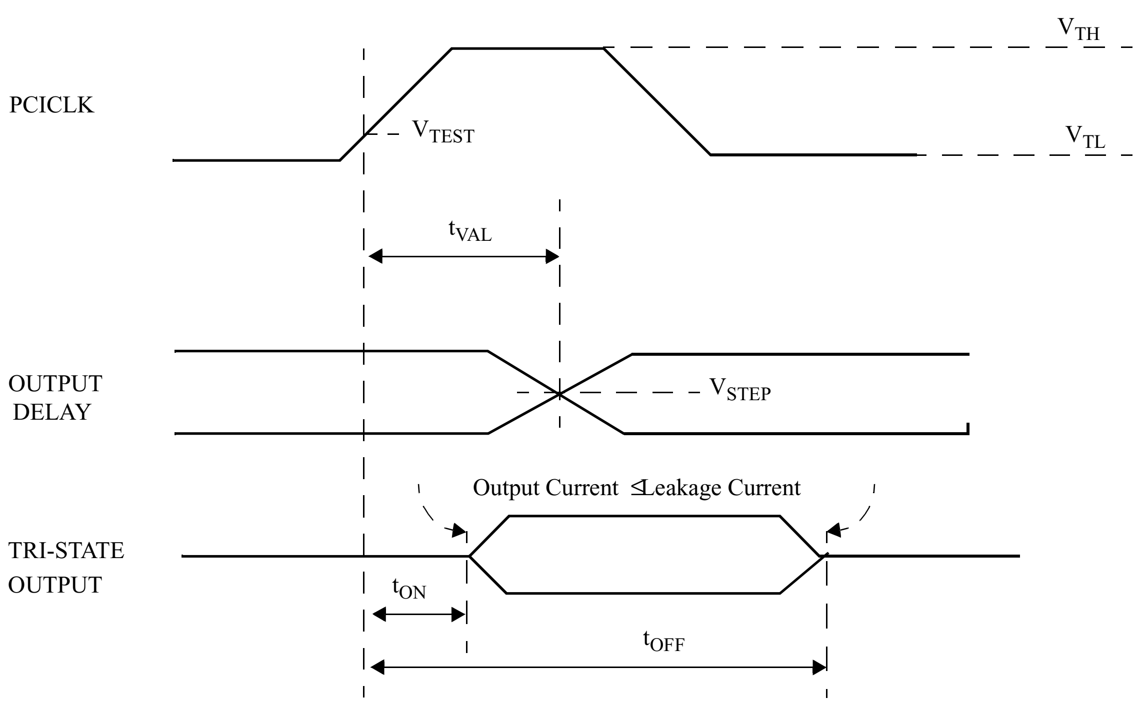


Figure 7-14 Output Timing Measurement Conditions

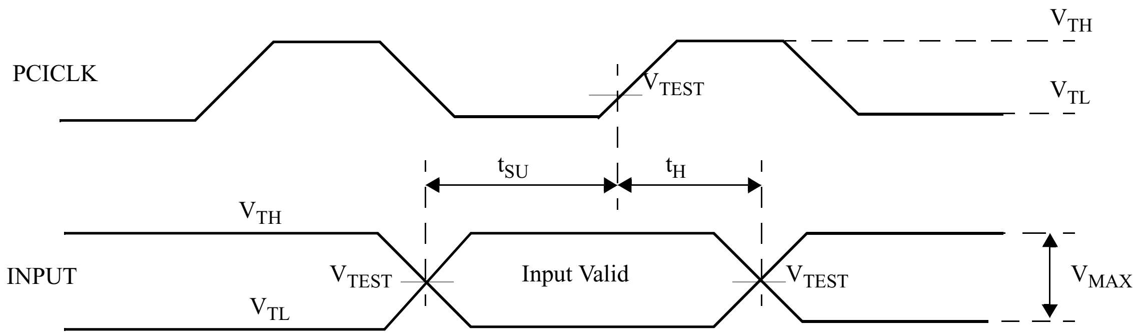


Figure 7-15 Input Timing Measurement Conditions

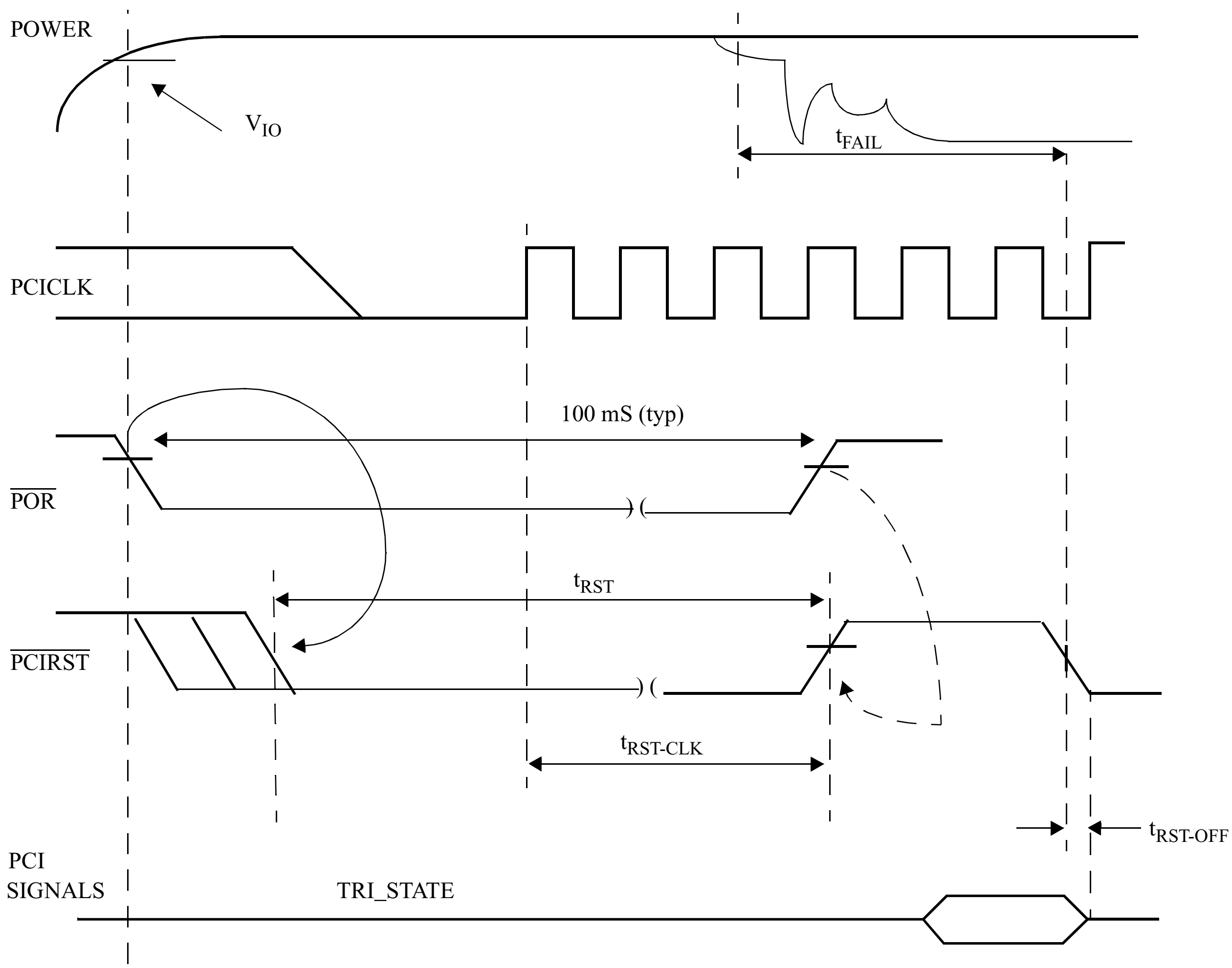


Figure 7-16 Reset Timing

Note: The value of t_{FAIL} is 500 nS (maximum) from the power rail which exceeds specified tolerance by more than 500 mV.

7.4.5. ISA Interface

All output timing is guaranteed for 50 pF load, unless otherwise specified.

The ISA Clock divisor (defined in bits[2:0] of Core Logic Function 0, index 50h) is 011.

Table 7.30 ISA Output Signals

Symbol	Parameter	Conditions	Bus Width	Type	Min (nS)	Max (nS)	Reference
t_{RD1}	$\overline{\text{MEMR}}$ Read Active pulse width FE to RE	Standard	16	M	225nS	—	Figure 7-17
t_{RD2}	$\overline{\text{MEMR}}$ Read Active pulse width FE to RE	Zero wait state	16	M	105nS	—	Figure 7-17
t_{RD3}	$\overline{\text{IOR}}$ Read Active pulse width FE to RE	Standard	16	I/O	160nS	—	Figure 7-17
t_{RD4}	$\overline{\text{IOR/MEMR}}$ Read Active pulse width FE to RE	Standard	8	M, I/O	520nS	—	Figure 7-17
t_{RD5}	$\overline{\text{IOR}}$/$\overline{\text{MEMR}}$ Read Active pulse width FE to RE	Zero wait state	8	M, I/O	160nS	—	Figure 7-17
t_{RCU1}	$\overline{\text{MEMR}}$ Inactive pulse width		16	M	103nS	—	Figure 7-17
t_{RCU2}	$\overline{\text{MEMR}}$ Inactive pulse width		8	M	163nS	—	Figure 7-17
t_{RCU3}	$\overline{\text{IOR}}$ Inactive pulse width		8, 16	I/O	163nS	—	Figure 7-17
t_{WR1}	$\overline{\text{MEMW}}$ Write Active pulse width FE to RE	Standard	16	M	225nS	—	Figure 7-18
t_{WR2}	$\overline{\text{MEMW}}$ Write Active pulse width FE to RE	Zero wait state	16	M	105nS	—	Figure 7-18
t_{WR3}	$\overline{\text{IOW}}$ Write Active pulse width FE to RE	Standard	16	I/O	160nS	—	Figure 7-18
t_{WR4}	$\overline{\text{IOW/MEMW}}$ Write Active pulse width FE to RE	Standard	8	M, I/O	520nS	—	Figure 7-18
t_{WR5}	$\overline{\text{IOW/MEMW}}$ Write Active pulse width FE to RE	Zero wait state	8	M, I/O	160nS	—	Figure 7-18
t_{WCU1}	$\overline{\text{MEMW}}$ Inactive pulse width		16	M	103nS	—	Figure 7-18
t_{WCU2}	$\overline{\text{MEMW}}$ Inactive pulse width		8	M	163nS	—	Figure 7-18
t_{WCU3}	$\overline{\text{IOW}}$ Inactive pulse width		8, 16	I/O	163nS	—	Figure 7-18
t_{RDYH}	$\overline{\text{IOR}}$/$\overline{\text{MEMR}}$/$\overline{\text{IOW}}$/$\overline{\text{MEMW}}$ Hold after IOCHRDY RE		8, 16	M, I/O	120nS	—	Figure 7-18
t_{RDYA1}	IOCHRDY valid after $\overline{\text{IOR}}$/$\overline{\text{MEMR}}$/$\overline{\text{IOW}}$/$\overline{\text{MEMW}}$ FE		16	M, I/O	—	78nS	Figure 7-18
t_{RDYA2}	IOCHRDY valid after $\overline{\text{IOR}}$/$\overline{\text{MEMR}}$/$\overline{\text{IOW}}$/$\overline{\text{MEMW}}$ FE		8	M, I/O	—	366nS	Figure 7-18
t_{IOCSA}	$\overline{\text{IOCS[1:0]}}$/$\overline{\text{DOCS}}$/$\overline{\text{ROMCS}}$ Driven active from A[23:0] valid		8, 16	M, I/O	—	TBD	Figure 7-17, Figure 7-18
t_{IOCSH}	$\overline{\text{IOCS[1:0]}}$/$\overline{\text{DOCS}}$/$\overline{\text{ROMCS}}$ Valid hold after A[23:0] invalid		8, 16	M, I/O	0nS	—	Figure 7-17, Figure 7-18
t_{AR1}	A[23:0]/$\overline{\text{BHE}}$ valid before $\overline{\text{MEMR}}$ active		16	M	34nS	—	Figure 7-17
t_{AR2}	A[23:0]/$\overline{\text{BHE}}$ valid before $\overline{\text{IOR}}$ active		16	I/O	100nS	—	Figure 7-17
t_{AR3}	A[23:0]/$\overline{\text{BHE}}$ valid before $\overline{\text{MEMR}}$/$\overline{\text{IOR}}$ active		8	M, I/O	100nS	—	Figure 7-17
t_{RA}	A[23:0]/BHE valid hold after $\overline{\text{MEMR}}$/$\overline{\text{IOR}}$ inactive		8, 16	M, I/O	41nS	—	Figure 7-17

Table 7.30 ISA Output Signals (cont.)

Symbol	Parameter	Conditions	Bus Width	Type	Min (nS)	Max (nS)	Reference
t_{RVDS}	Read data D[15:0] valid setup before $\overline{\text{MEMR}}$/$\overline{\text{IOR}}$ inactive		8, 16	M, I/O	24nS	—	Figure 7-17
t_{RDH}	Read data D[15:0] valid holdafter $\overline{\text{MEMR}}$/$\overline{\text{IOR}}$ inactive		8, 16	M, I/O	0nS	—	Figure 7-17
t_{HZ}	Read data floating after $\overline{\text{MEMR}}$/$\overline{\text{IOR}}$ inactive		8, 16	M, I/O	—	41nS	Figure 7-17
t_{AW1}	A[23:0]/$\overline{\text{BHE}}$ valid before $\overline{\text{MEMW}}$ active		16	M	34nS	—	Figure 7-18
t_{AW2}	A[23:0]/$\overline{\text{BHE}}$ valid before $\overline{\text{IOW}}$ active		16	I/O	100nS	—	Figure 7-18
t_{AW3}	A[23:0]/$\overline{\text{BHE}}$ valid before $\overline{\text{MEMW}}$/$\overline{\text{IOW}}$ active		8	M, I/O	100nS	—	Figure 7-18
t_{WA}	A[23:0]/$\overline{\text{BHE}}$ valid hold after $\overline{\text{MEMW}}$/$\overline{\text{IOW}}$ invalid		8, 16	M, I/O	41nS	—	Figure 7-18
t_{DV1}	Write data D[15:0] valid after $\overline{\text{MEMW}}$ active		8, 16	M	40nS	—	Figure 7-18
t_{DV2}	Write data D[15:0] valid after $\overline{\text{IOW}}$ active		8	I/O	40nS	—	Figure 7-18
t_{DV3}	Write data D[15:0] valid after $\overline{\text{IOW}}$ active		16	I/O	–23nS	—	Figure 7-18
t_{DH}	Write data D[15:0] after $\overline{\text{MEMW}}$/$\overline{\text{IOW}}$ inactive		8, 16	M, I/O	45nS	—	Figure 7-18
t_{DF}	Write data D[15:0] tristated after $\overline{\text{MEMW}}$/$\overline{\text{IOW}}$ inactive		8, 16	M, I/O	—	105nS	Figure 7-18
t_{WDAR}	Write data D[15:0] after read $\overline{\text{MEMR}}$/$\overline{\text{IOR}}$		8, 16	M, I/O	41nS	—	Figure 7-18

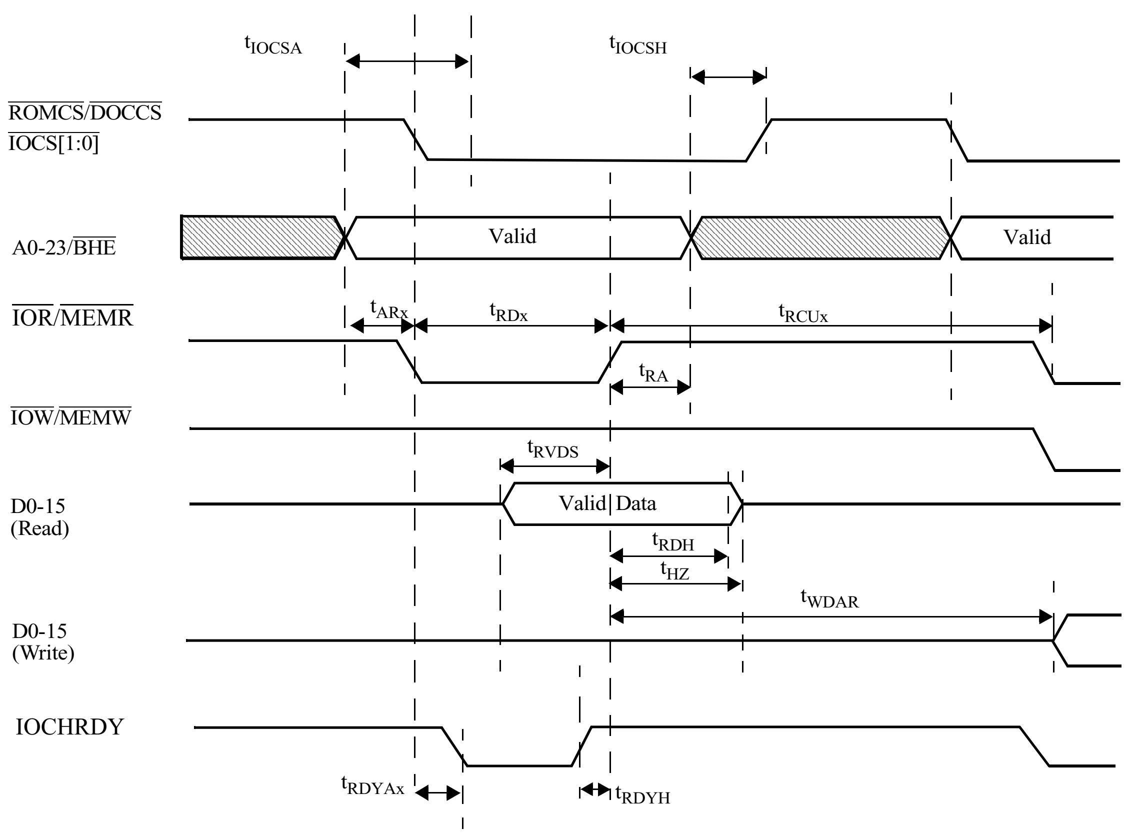


Figure 7-17 ISA Read Operation

Note: x indicates a numeric index for the relevant symbol.

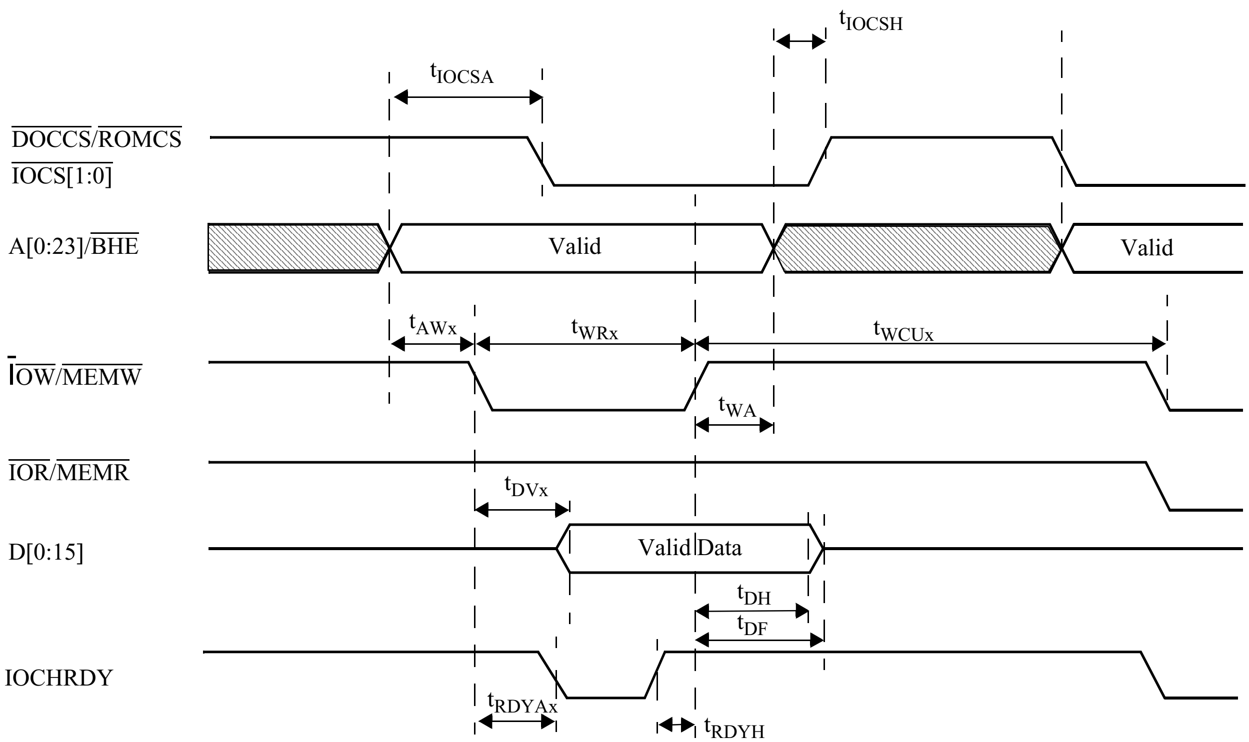


Figure 7-18 ISA Write Operation

Note: x indicates a numeric index for the relevant symbol.

7.4.6. IDE Interface Timing

Capacitance load is 150 pF for signals IDE_RST, IDE_CS0,1, IDE_DATA[15:0], and IDE_ADDR[2:0].

To measure IDE channel 1, register F3 offset 4 bit 1 must be set and GPIOs programmed in the propr direction.

For all other signals of the IDE interface, capacitance load is 75pF.

Table 7.31 General Timing of the IDE Interface

Symbol	Parameter	Conditions	Min	Max
t_{IDE_FALL}	Fall time of all IDE signals. From $0.9V_{I/O}$ to $0.1V_{I/O}$	C_L = 40pF	5nS	
t_{IDE_RISE}	Rise time of all IDE signals. From $0.1V_{I/O}$ to $0.9V_{I/O}$	C_L = 40pF	5nS	
$t_{IDE\_RST\_PW}$	$\overline{\text{IDE\_RST}}$ pulse width	—	25µS	

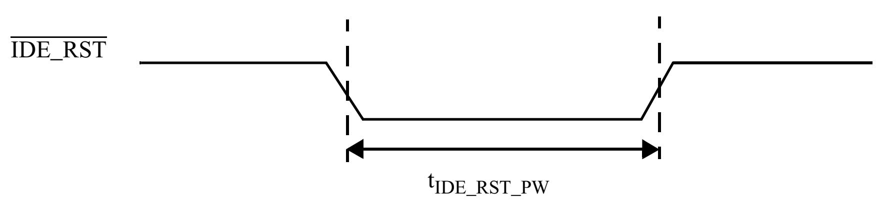


Figure 7-19 IDE Reset Timing

Table 7.32 IDE Register Transfer To/From Device

Symbol	Description	Mode 0 ns	Mode 1 ns	Mode 2 ns	Mode 3 ns	Mode 4 ns
t_0	Cycle time[1] (min)	600	383	240	180	120
t_1	Address valid to $\overline{\text{IDE\_IOR0,1}}$/ $\overline{\text{IDE\_IOW0,1}}$ setup (min)	70	50	30	30	25
t_2	$\overline{\text{IDE\_IOR0,1}}$/ $\overline{\text{IDE\_IOW0,1}}$ pulse width 8-bit[1] (min)	290	290	290	80	70
t_{2i}	$\overline{\text{IDE\_IOR0,1}}$/ $\overline{\text{IDE\_IOW0,1}}$ recovery time[1] (min)	—	—	—	70	25
t_3	$\overline{\text{IDE\_IOW0,1}}$ data setup (min)	60	45	30	30	20
t_4	$\overline{\text{IDE\_IOW0,1}}$ data hold (min)	30	20	15	10	10
t_5	$\overline{\text{IDE\_IOR0,1}}$ data setup (min)	50	35	20	20	20
t_6	$\overline{\text{IDE\_IOR0,1}}$ data hold (min)	5	5	5	5	5
t_{6Z}	$\overline{\text{IDE\_IOR0,1}}$ data tristate[2] (max)	30	30	30	30	30
t_9	$\overline{\text{IDE\_IOR0,1}}$ / $\overline{\text{IDE\_IOW0,1}}$ to address valid hold (min)	20	15	10	10	10
t_{RD}	Read Data Valid to IDE_IORDY0,1 active (if IDE_IORDY0,1 initially low after t_A) (min)	0	0	0	0	0
t_A	IDE_IORDY0,1 Setup time[3]	35	35	35	35	35
t_B	IDE_IORDY0,1 Pulse Width (max)	1250	1250	1250	1250	1250
t_C	IDE_IORDY0,1 assertion to release (max)	5	5	5	5	5

1. t_0 is the minimum total cycle time, t_2 is the minimum command active time, and t_{2i} is the minimum command recovery time or command inactive time. The actual cycle time equals the sum of the command active time and the command inactive time. The three timing requirements of t_0, t_2, and t_{2i} are met. The minimum total cycle time requirements is greater than the sum of t_2 and t_{2i}. (This means that a host implementation can lengthen t_2 and/or t_{2i} to ensure that t_0 is equal to or greater than the value reported in the device's IDENTIFY DEVICE data.)
2. This parameter specifies the time from the rising edge of $\overline{\text{IDE\_IOR0,1}}$ to the time that the data bus is no longer driven by the device (tristate).
3. The delay from the activation of $\overline{\text{IDE\_IOR0,1}}$ or $\overline{\text{IDE\_IOW0,1}}$ until the state of IDE_IORDY0,1 is first sampled. If IDE_IORDY0,1 is inactive, then the host waits until IDE_IORDY0,1 is active before the PIO cycle is completed. If the device is not driving IDE_IORDY0,1 negated after activation (t_A) of $\overline{\text{IDE\_IOR0,1}}$ or $\overline{\text{IDE\_IOW0,1}}$, then t_5 is met and t_{RD} is not applicable. If the device is driving IDE_IORDY0,1 negated after activation (t_A) of $\overline{\text{IDE\_IOR0,1}}$ or $\overline{\text{IDE\_IOW0,1}}$, then t_{RD} is met and t_5 is not applicable.

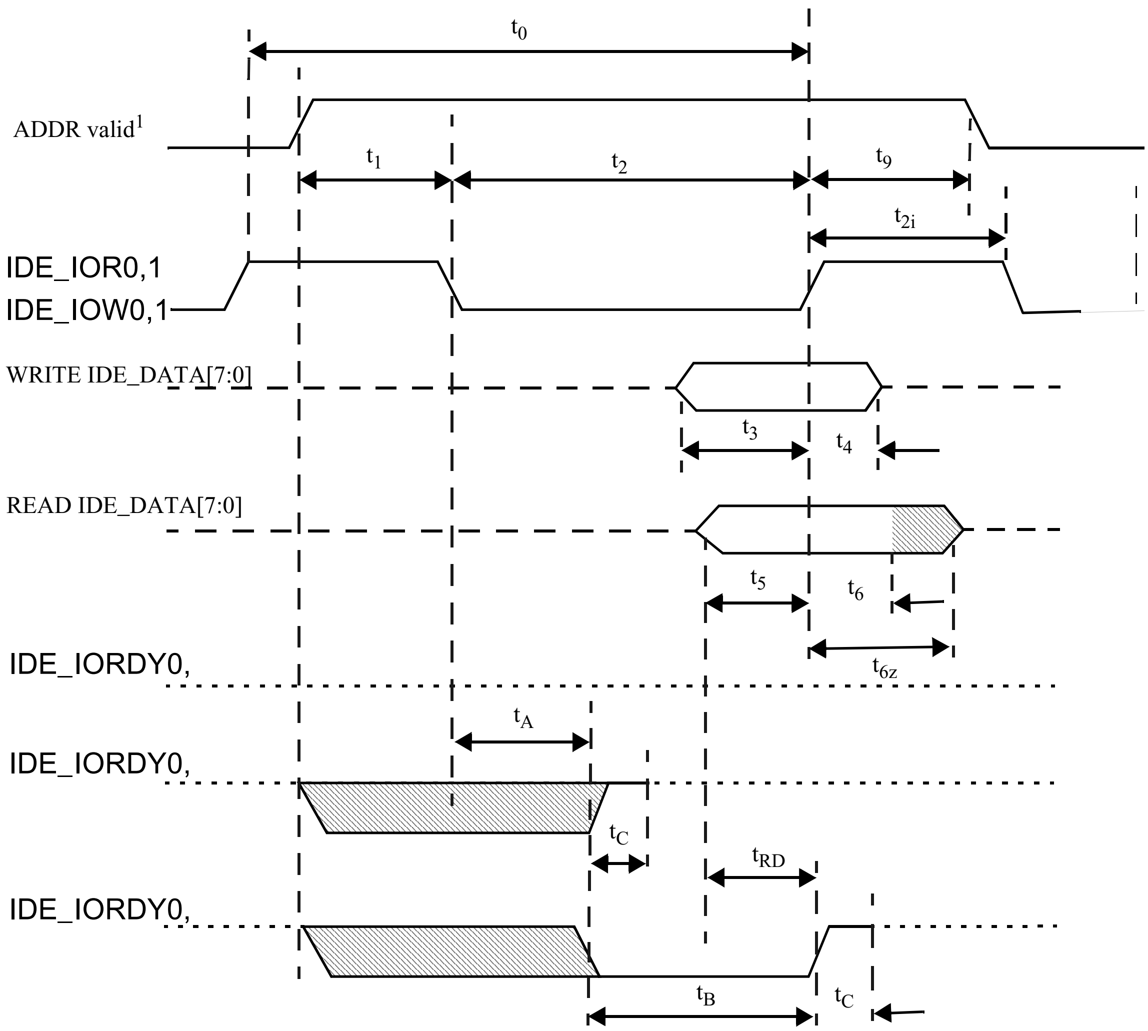


Figure 7-20 IDE Register Transfer To/From Device

Notes:

1. Device address consists of signals IDE_CS0, IDE_CS1 and IDE_ADDR[2:0].
2. Negation of IDE_IORDY0,1 is used to extend the PIO cycle. The determination of whether or not the cycle is to be extended is made by the host after t_A from the assertion of IDE_IOR0,1 or IDE_IOW0,1.
3. Device never negates IDE_IORDY0,1. Devices keep IDE_IORDY0,1 released, and no wait is generated.
4. Device negates IDE_IORDY0,1 before t_A but causes IDE_IORDY0,1 to be asserted before t_A. IDE_IORDY0,1 is released, and no wait is generated.
5. Device negates IDE_IORDY0,1 before t_A. IDE_IORDY0,1 is released prior to negation and may be asserted for no more than 5 ns before release. A wait is generated.
6. The cycle completes after IDE_IORDY0,1 is reasserted. For cycles where a wait is generated and IDE_IOR0,1 is asserted, the device places read data on IDE_DATA[15:0] for t_{RD} before asserting IDE_IORDY0,1.

Table 7.33 IDE PIO Data Transfer To/From Device

Symbol	Description	Mode 0 ns	Mode 1 ns	Mode 2 ns	Mode 3 ns	Mode 4 ns
t_0	Cycle time[1] (min)	600	383	240	180	120
t_1	Address valid to IDE_IOR0,1/ IDE_IOW0,1 setup (min)	70	50	30	30	25
t_2	IDE_IOR0,1/ IDE_IOW0,1 16-bit[1] (min)	165	125	100	80	70
t_{2i}	IDE_IOR0,1/ IDE_IOW0,1 recovery time[1] (min)	-	-	-	70	25
t_3	IDE_IOW0,1 data setup (min)	60	45	30	30	20
t_4	IDE_IOW0,1 data hold (min)	30	20	15	10	10
t_5	IDE_IOR0,1 data setup (min)	50	35	20	20	20
t_6	IDE_IOR0,1 data hold (min)	5	5	5	5	5
t_{6Z}	IDE_IOR0,1 data tristate[2] (max)	30	30	30	30	30
t_9	IDE_IOR0,1 / IDE_IOW0,1 to address valid hold (min)	20	15	10	10	10
t_{RD}	Read Data Valid to IDE_IORDY0,1 active (if IDE_IORDY0,1 initially low after t_A) (min)	0	0	0	0	0
t_A	IDE_IORDY0,1 Setup time[3]	35	35	35	35	35
t_B	IDE_IORDY0,1 Pulse Width (max)	1250	1250	1250	1250	1250
t_C	IDE_IORDY0,1 assertion to release (max)	5	5	5	5	5

1. t_0 is the minimum total cycle time, t_2 is the minimum command active time, and t_{2i} is the minimum command recovery time or command inactive time. The actual cycle time equals the sum of the command active time and the command inactive time. The three timing requirements of t_0, t_2, and t_{2i} are met. The minimum total cycle time requirement is greater than the sum of t_2 and t_{2i}. (This means that a host implementation may lengthen t_2 and/or t_{2i} to ensure that t_0 is equal to or greater than the value reported in the device's IDENTIFY DEVICE data.)
2. This parameter specifies the time from the rising edge of IDE_IOR0,1 to the time that the data bus is no longer driven by the device (tristate).
3. The delay from the activation of IDE_IOR0,1 or IDE_IOW0,1 until the state of IDE_IORDY0,1 is first sampled. If IDE_IORDY0,1 is inactive, then the host waits until IDE_IORDY0,1 is active before the PIO cycle is completed. If the device is not driving IDE_IORDY0,1 negated after the activation (t_A) of IDE_IOR0,1 or IDE_IOW0,1, then t_5 is met and t_{RD} is not applicable. If the device is driving IDE_IORDY0,1 negated after the activation (t_A) of IDE_IOR0,1 or IDE_IOW0,1, then t_{RD} is met and t_5 is not applicable.

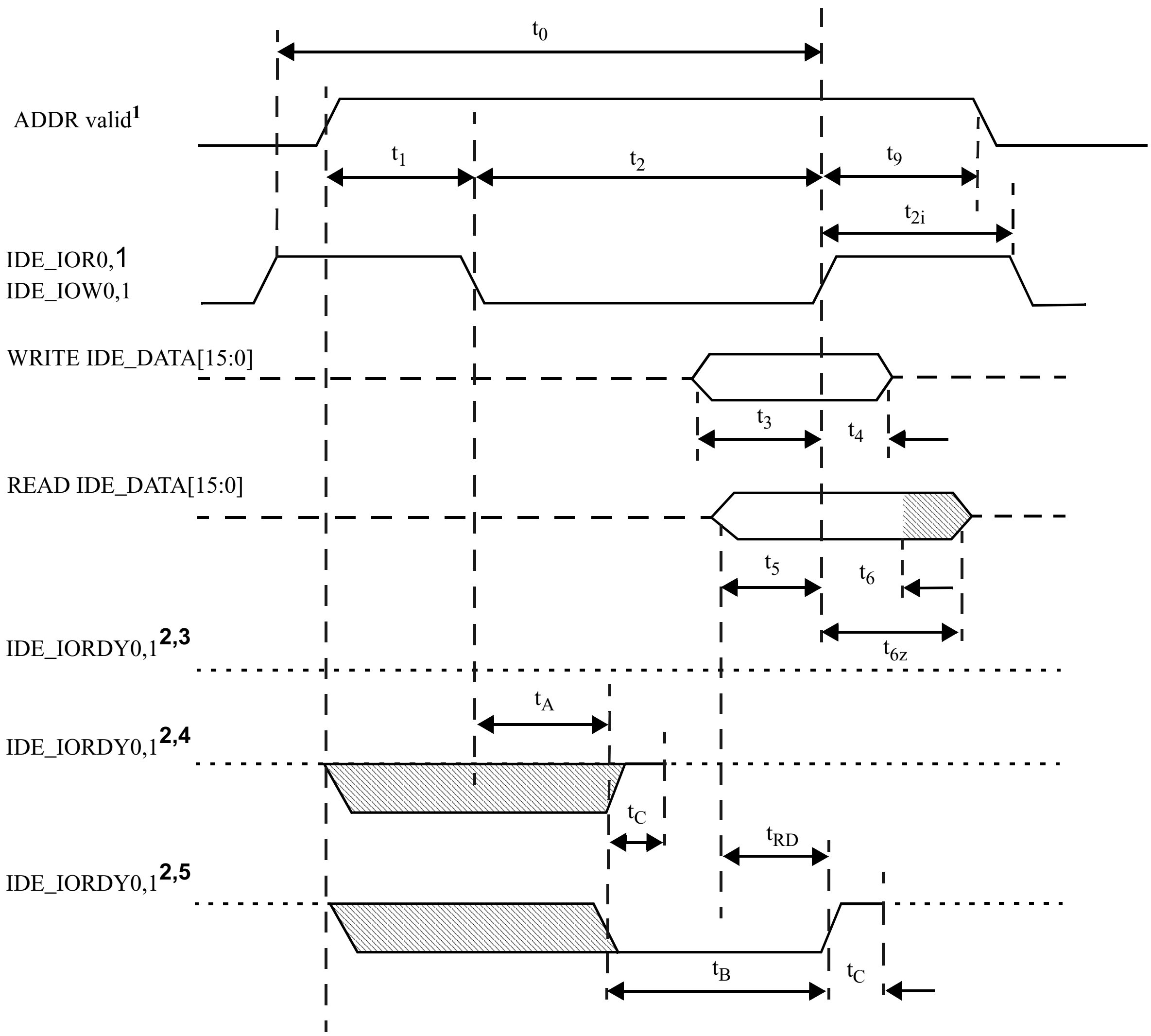


Figure 7-21 IDE PIO Data Transfer To/From Device

Notes:

1. Device address consists of signals $\overline{\text{IDE\_CS0}}$, $\overline{\text{IDE\_CS1}}$ and IDE_ADDR[2:0].
2. Negation of IDE_IORDY0,1 is used to extend the PIO cycle. The determination of whether or not the cycle is to be extended is made by the host after t_A from the assertion of $\overline{\text{IDE\_IOR0,1}}$ or $\overline{\text{IDE\_IOW0,1}}$.
3. Device never negates IDE_IORDY0,1. Devices keep IDE_IORDY0,1 released, and no wait is generated.
4. Device negates IDE_IORDY0,1 before t_A but causes IDE_IORDY0,1 to be asserted before t_A. IDE_IORDY0,1 is released, and no wait is generated.
5. Device negates IDE_IORDY0,1 before t_A. IDE_IORDY0,1 is released prior to negation and may be asserted for no more than 5 ns before release. A wait is generated.
6. The cycle completes after IDE_IORDY0,1 is reasserted. For cycles where a wait is generated and $\overline{\text{IDE\_IOR0,1}}$ is asserted, the device places read data on IDE_DATA[15:0] for t_{RD} before asserting IDE_IORDY0,1.

Table 7.34 IDE Multiword DMA Data Transfer

Symbol	Description	Mode 0 ns	Mode 1 ns	Mode 2 ns
t_0	Cycle time[1] (min)	480	150	120
t_D	$\overline{\text{IDE\_IOR0,1}}$/ $\overline{\text{IDE\_IOW0,1}}$ (min)	215	80	70
t_E	$\overline{\text{IDE\_IOR0,1}}$ data access (max)	150	60	50
t_F	$\overline{\text{IDE\_IOR0,1}}$ data hold (min)	5	5	5
t_G	$\overline{\text{IDE\_IOW0,1}}$ / $\overline{\text{IDE\_IOW0,1}}$ data setup (min)	100	30	20
t_H	$\overline{\text{IDE\_IOW0,1}}$ data hold (min)	20	15	10
t_I	$\overline{\text{IDE\_DACK0,1}}$ to $\overline{\text{IDE\_IOR0,1/IDE\_IOW0,1}}$ setup (min)	0	0	0
t_J	$\overline{\text{IDE\_IOR0,1}}$ / $\overline{\text{IDE\_IOW0,1}}$ to $\overline{\text{IDE\_DACK0,1}}$ hold (min)	20	5	5
t_{KR}	$\overline{\text{IDE\_IOR0,1}}$ negated pulse width (min)	50	50	25
t_{KW}	$\overline{\text{IDE\_IOW0,1}}$ negated pulse width (min)	215	50	25
t_{LR}	$\overline{\text{IDE\_IOR0,1}}$ to IDE_DREQ0,1 delay (max)	120	40	35
t_{LW}	$\overline{\text{IDE\_IOW0,1}}$ to IDE_DREQ0,1 delay (max)	40	40	35
t_M	$\overline{\text{IDE\_CS0}}$ / $\overline{\text{IDE\_CS1}}$ valid to $\overline{\text{IDE\_IOR0,1}}$ / $\overline{\text{IDE\_IOW0,1}}$ (min)	50	30	25
t_N	$\overline{\text{IDE\_CS0}}$ / $\overline{\text{IDE\_CS1}}$ hold	15	10	10
t_Z	$\overline{\text{IDE\_DACK0,1}}$ to tristate	20	25	25

1. t_0 is the minimum total cycle time, t_D is the minimum command active time, and t_{KR} or t_{KW} is the minimum command recovery time or command inactive time. The actual cycle time equals the sum of the command active time and the command inactive time. The three timing requirements of t_0, t_D and $t_{KR/KW}$, are met. The minimum total cycle time requirement t_0 is greater than the sum of t_D and $t_{KR/KW}$. (This means that a host implementation can lengthen t_D and/or $t_{KR/KW}$ to ensure that t_0 is equal to or greater than the value reported in the device's IDENTIFY DEVICE data.)

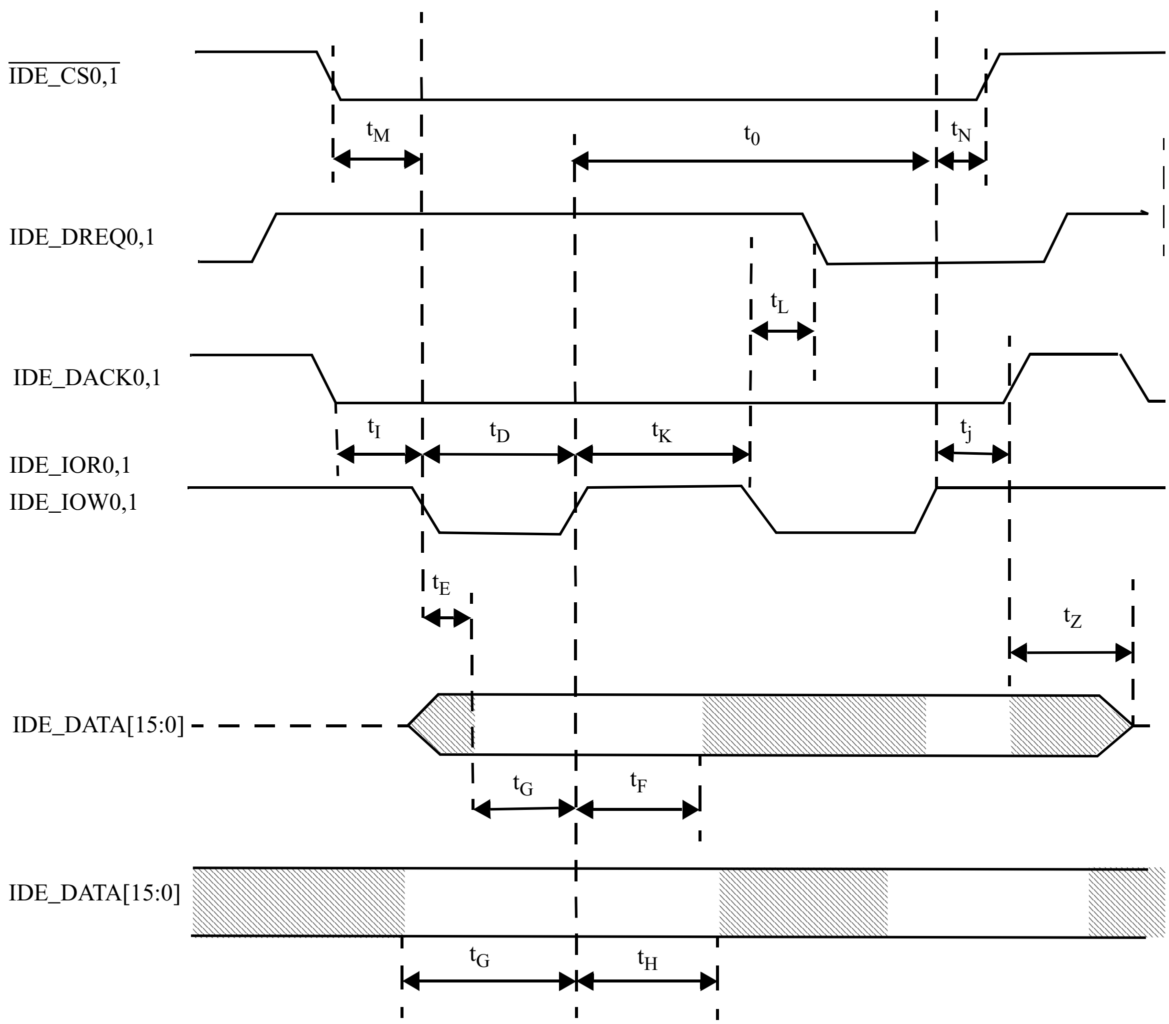


Figure 7-22 Multiword Data Transfer

Notes:

1. For Multi-Word DMA transfers, the Device may negate IDE_DREQ0,1 within the t_L specified time once $\overline{\text{IDE\_DACK0,1}}$ is asserted, and reassert it again at a later time to resume the DMA operation. Alternatively, if the device is able to continue the transfer of data, the device may leave IDE_DREQ0,1 asserted and wait for the host to reassert $\overline{\text{IDE\_DACK0,1}}$.
2. This signal can be negated by the host to suspend the DMA transfer in process.

Table 7.35 Ultra DMA Data Burst Timing Requirements

Symbol	Mode 0 ns		Mode 1 ns		Mode 2 ns		Comment
	MIN	MAX	MIN	MAX	MIN	MAX	
t_{2CYC}	240	—	160	—	120	—	Typical sustained average two cycle time
	235	—	156	—	117	—	Two cycle time allowing for clock variations (from rising edge to next rising edge or from falling edge to next falling edge of STROBE)
t_{CYC}	114	—	75	—	55	—	Cycle time allowing for asymmetry and clock variations (from STROBE edge to STROBE edge)
t_{DS}	15	—	10	—	7	—	Data setup time (at recipient)
t_{DH}	5	—	5	—	5	—	Data hold time (at recipient)
t_{DVS}	70	—	48	—	34	—	Data valid setup time at sender (from data bus being valid until STROBE edge)
t_{DVH}	6	—	6	—	6	—	Data valid hold time at sender (from STROBE edge until data may become invalid)
t_{FS}	0	230	0	200	0	170	First STROBE time (for device to first negate IDE_IRDY0,1(DSTROBE0,1) from IDE_IOW0,1(STOP0,1) during a data in burst)
t_{LI}	0	150	0	150	0	150	Limited interlock time[1]
t_{MLI}	20	—	20	—	20	—	Interlock time with minimum[1]
t_{UI}	0	—	0	—	0	—	Unlimited interlock time[1]
t_{AZ}		10	—	10	—	10	Maximum time allowed for output drivers to release (from being asserted or negated)
t_{ZAH}	20	—	20	—	20	—	Minimum delay time required for output drivers to assert or negate (from released state)
t_{ZAD}	0	—	0	—	0	—	
t_{ENV}	20	70	20	70	20	70	Envelope time (from IDE_DACK0,1 to IDE_IOW0,1(STOP0,1) and IDE_IOR0,1(HDMARDY0,1) during data out burst initiation)
t_{SR}	—	50	—	30	—	20	STROBE to DMARDY time (if DMARDY- is negated before this long after STROBE edge, the recipient shall receive no more than one additional data word)
t_{RFS}	—	75	—	60	—	50	Ready-to-final-STROBE time (no STROBE edges shall be sent this long after negation of DMARDY-)
t_{RP}	160		125	—	100	—	Ready-to-pause time (time that recipient shall wait to initiate pause after negating DMARDY-)
t_{IORDYZ}	—	20	—	20	—	20	Pull-up time before allowing IDE_IORDY0,1 to be released

Table 7.35 Ultra DMA Data Burst Timing Requirements (cont.)

Symbol	Mode 0 ns		Mode 1 ns		Mode 2 ns		Comment
	MIN	MAX	MIN	MAX	MIN	MAX	
t_{ZIORDY}	0	—	0	—	0	—	Minimum time device shall wait before driving IDE_IORDY0,1
T_{ACK}	20	—	20	—	20	—	Setup and hold times for $\overline{\text{IDE\_DACK0,1}}$ (before assertion or negation)
T_{SS}	50	—	50	—	50	—	Time from STROBE edge to negation of $\overline{\text{IDE\_DREQ0}}$,1 or assertion of $\overline{\text{IDE\_IOW0,1}}$(STOP0,1) (when sender terminates a burst)

1. t_{UI} , t_{MLI}, and t_{LI} indicate sender-to-recipient or recipient-to-sender interlocks, that is, one agent (either sender or recipient) is waiting for the other agent to respond with a signal before proceeding. t_{UI} is an unlimited interlock with no maximum time value. t_{MLI} is a limited time-out with a defined minimum. t_{LI} is a limited time-out with a defined maximum.

All timing parameters are measured at the connector of the device to which the parameter applies. For example, the sender stops generating STROBE edges t_{RFS} after the negation of DMARDY. Both STROBE and DMARDY timing measurements are taken at the connector of the sender.

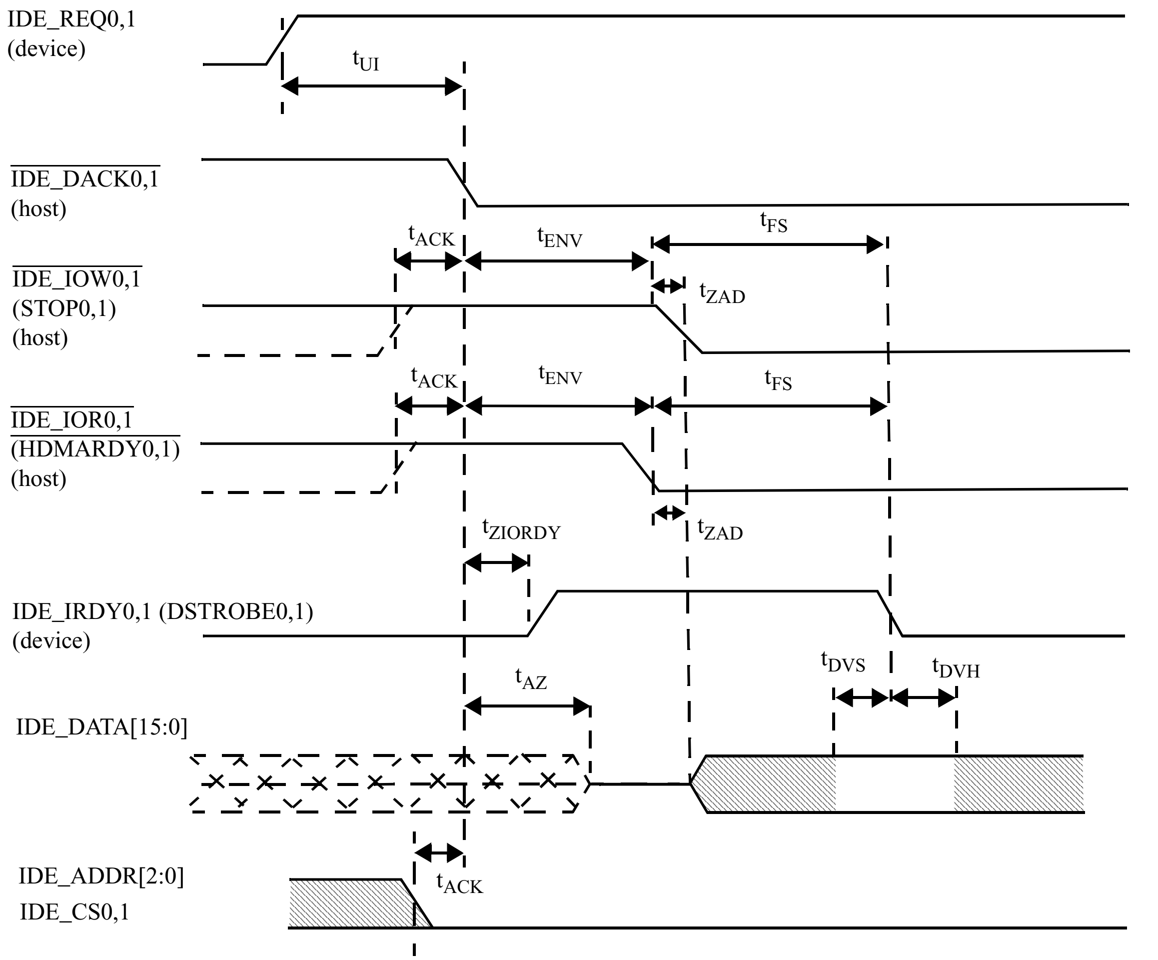


Figure 7-23 Initiating an Ultra DMA Data in Burst

Note: The definitions for the IDE_IOW0,1(STOP0,1), IDE_IOR0,1(HDMARDY0,1) and IDE_IRDY0,1(DSTROBE0,1) signal lines are not in effect until IDE_REQ0,1 and IDE_DACK0,1 are asserted.

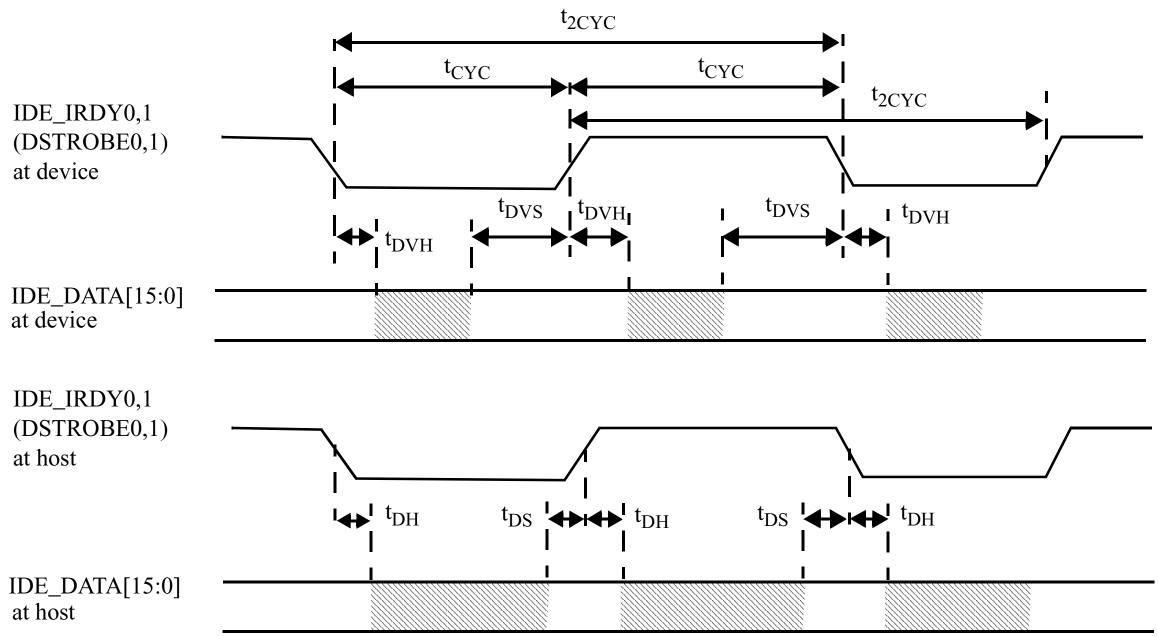


Figure 7-24 Sustained Ultra DMA Data In Burst

Note: IDE_DATA[15:0] and IDE_IRDY0,1(DSTROBE0,1) signals are shown at both the host and the device to emphasize that cable settling time and cable propagation delay do not allow the data signals to be considered stable at the host until a certain amount of time after they are driven by the device.

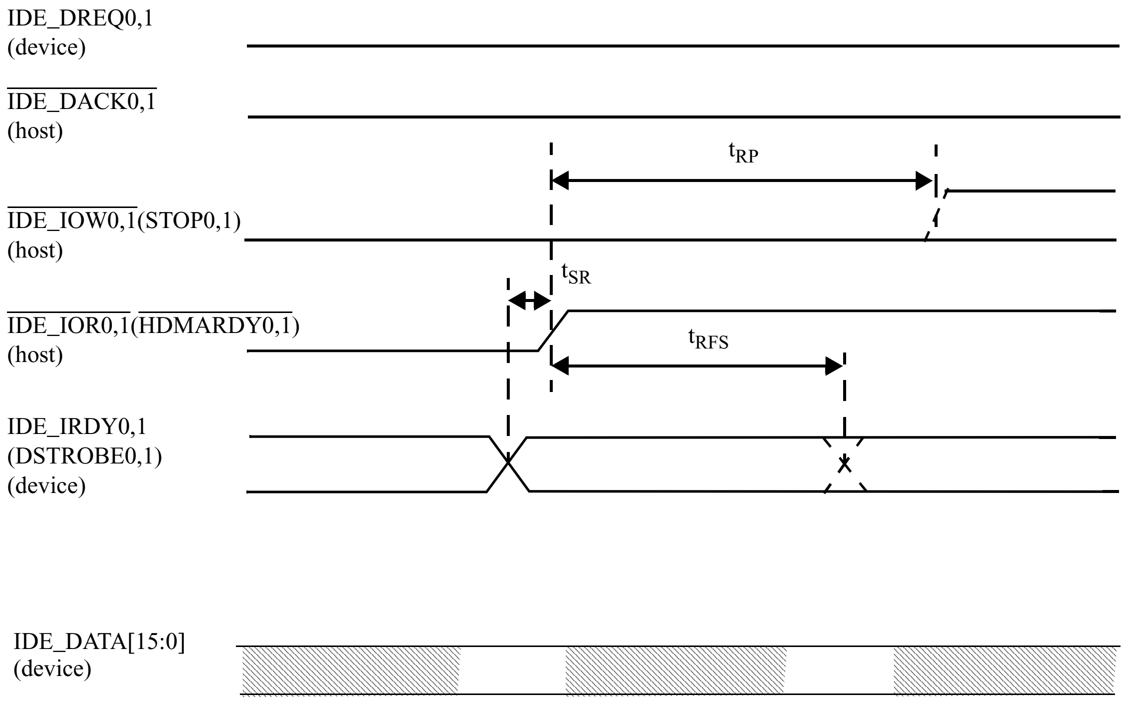


Figure 7-25 Host Pausing an Ultra DMA Data In Burst

Notes:

1. The host can assert IDE_IOW0,1(STOP0,1) to request termination of the Ultra DMA burst no sooner than t_{RP} after IDE_IOR0,1(HDMARDY0,1) is deasserted.
2. If the t_{SR} timing is not satisfied, the host may receive up to two additional datawords from the device.

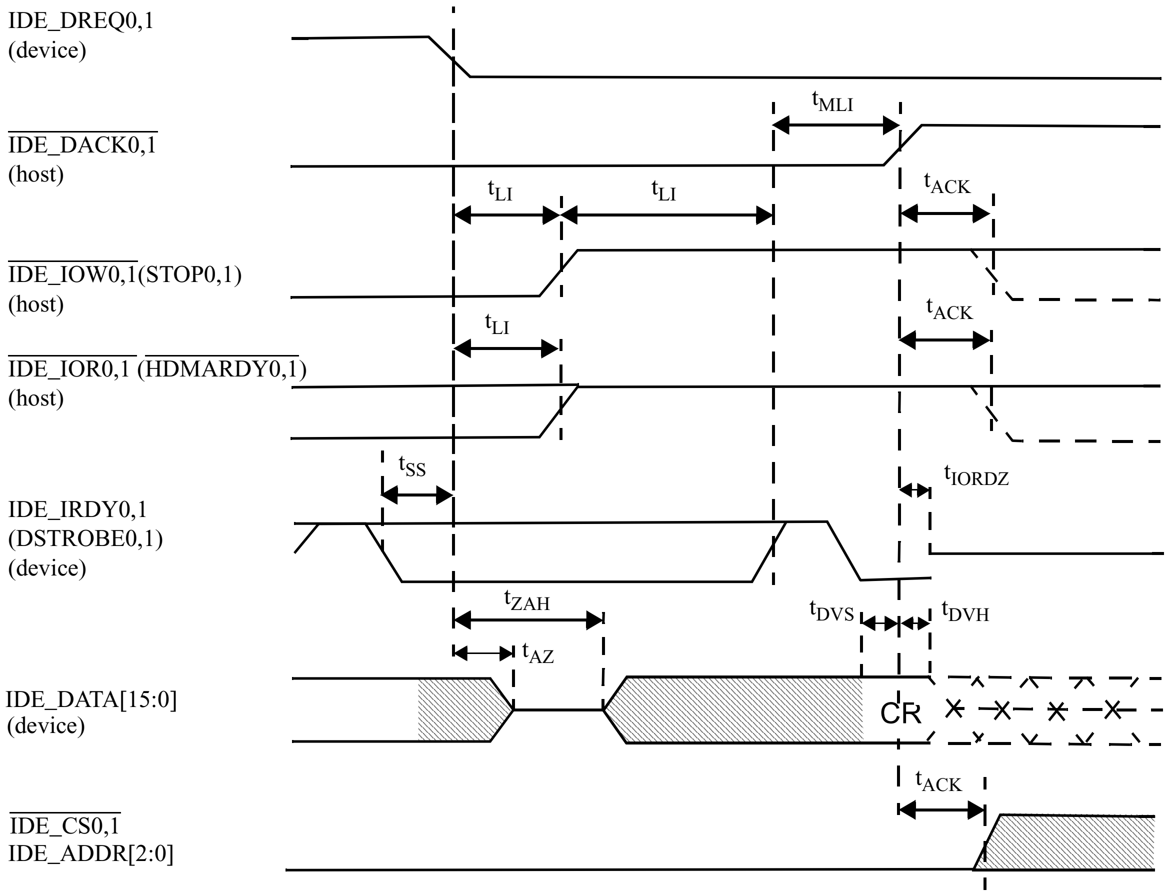


Figure 7-26 Device Terminating an Ultra DMA Data In Burst

Note: The definitions for the IDE_IOW0,1(STOP0,1), IDE_IOR0,1(HDMARDY0,1) and IDE_IRDY0,1(DSTROBE0,1) signal lines are no longer in effect after IDE_DREQ0,1 and IDE_DACK0,1 are deasserted.

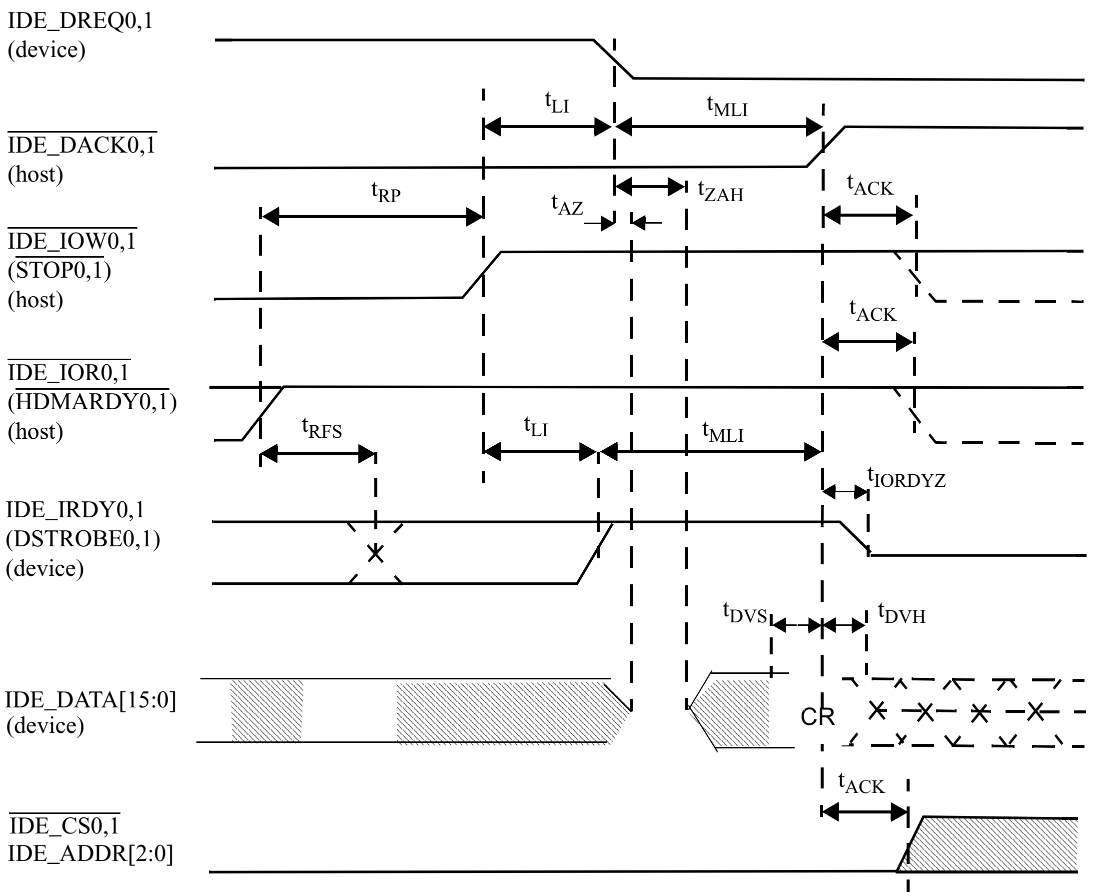


Figure 7-27 Host Terminating an Ultra DMA Data In Burst

Note: The definitions for the IDE_IOW0,1(STOP0,1), IDE_IOR0,1(HDMARDY0,1) and IDE_IRDY0,1(DSTROBE0,1) signal lines are no longer in effect after IDE_DREQ0,1 and IDE_DACK0,1 are deasserted.

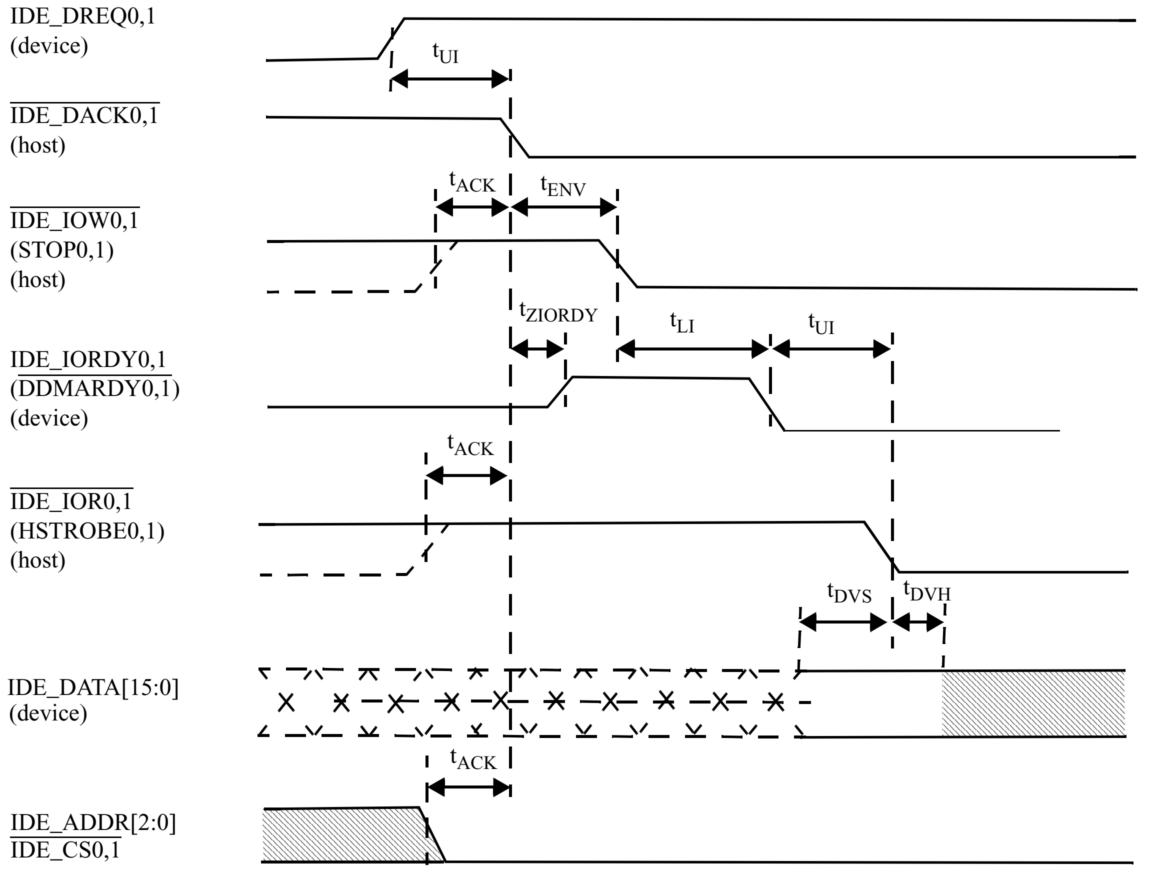


Figure 7-28 Initiating an Ultra DMA Data Out Burst

Note: The definitions for the IDE_IOW0,1(STOP0,1), IDE_IORDY0,1(DDMARDY0,1) and IDE_IOR0,1(HSTROBE0,1) signal lines are not in effect until IDE_DREQ0,1 and IDE_DACK0,1 are asserted.

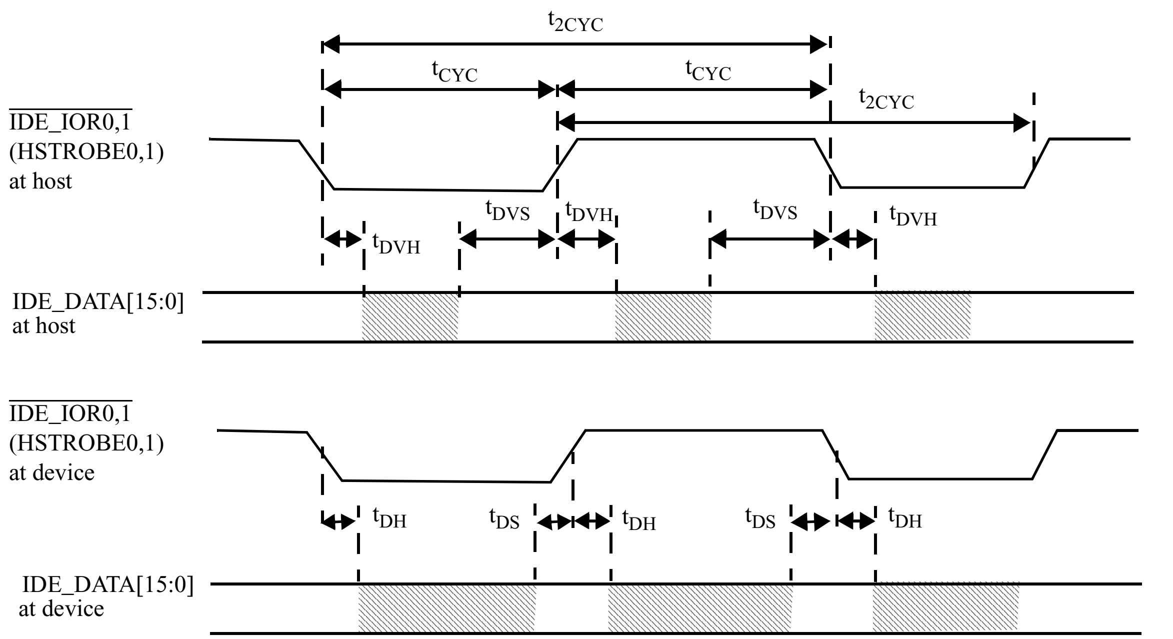


Figure 7-29 Sustained Ultra DMA Data Out Burst

Note: IDE_DATA[15:0] and $\overline{\text{IDE\_IOR0,1}}$(HSTROBE0,1) signals are shown at both the device and the host to emphasize that cable settling time and cable propagation delay do not allow the data signals to be considered stable at the device until a certain amount of time after they are driven by the device.

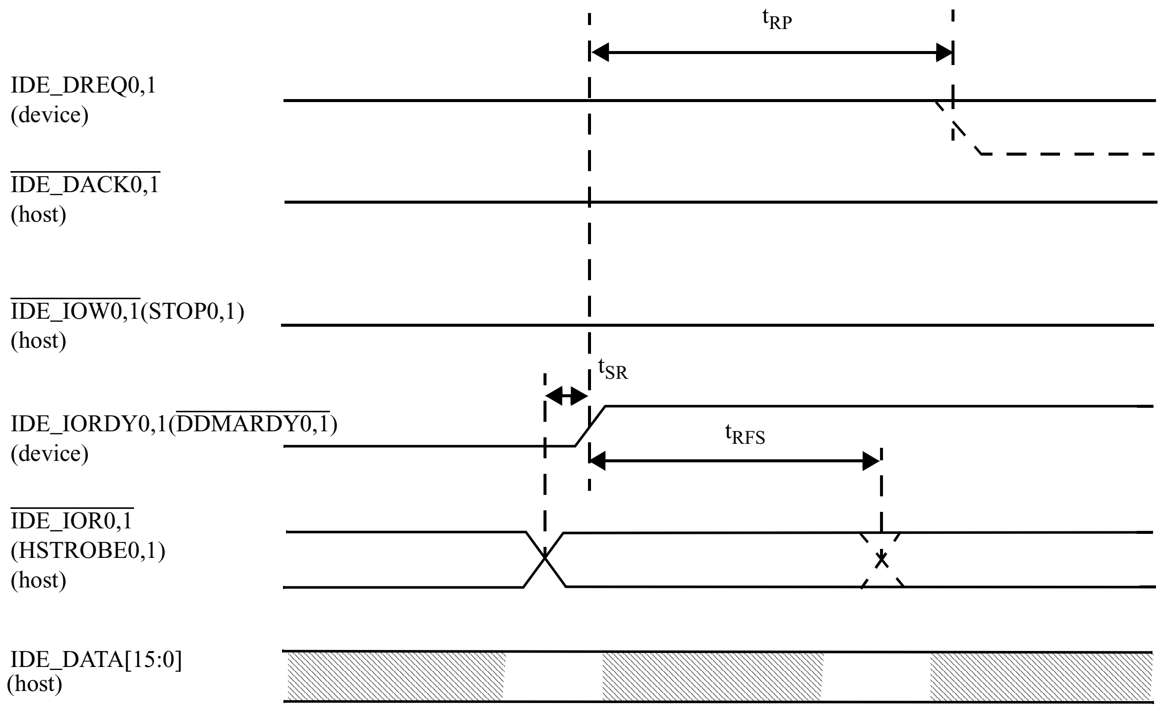


Figure 7-30 Device Pausing an Ultra DMA Data Out Burst

Note:

1. The device can deassert IDE_DREQ0,1 to request termination of the Ultra DMA burst no sooner than t_{RP} after IDE_IORDY0,1(DDMARDY0,1) is deasserted.
2. If the t_{SR} timing is not satisfied, the device may receive up to two additional datawords from the host.

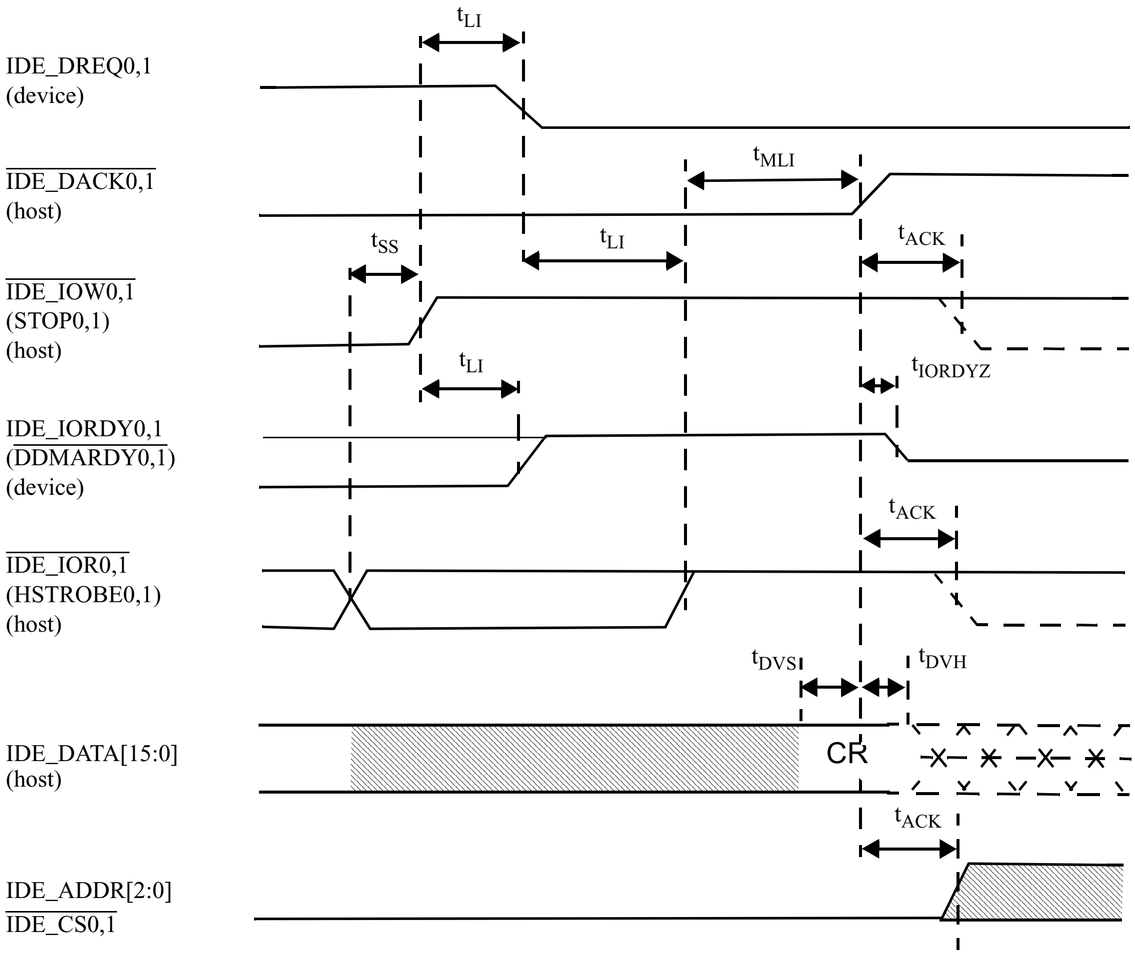


Figure 7-31 Host Terminating an Ultra DMA Data Out Burst

Note: The definitions for the IDE_IOW0,1(STOP0,1), IDE_IORDY0,1(DDMARDY0,1) and IDE_IOR0,1(HSTROBE0,1) signal lines are no longer in effect after IDE_DREQ0,1 and IDE_DACK0,1 are deasserted.

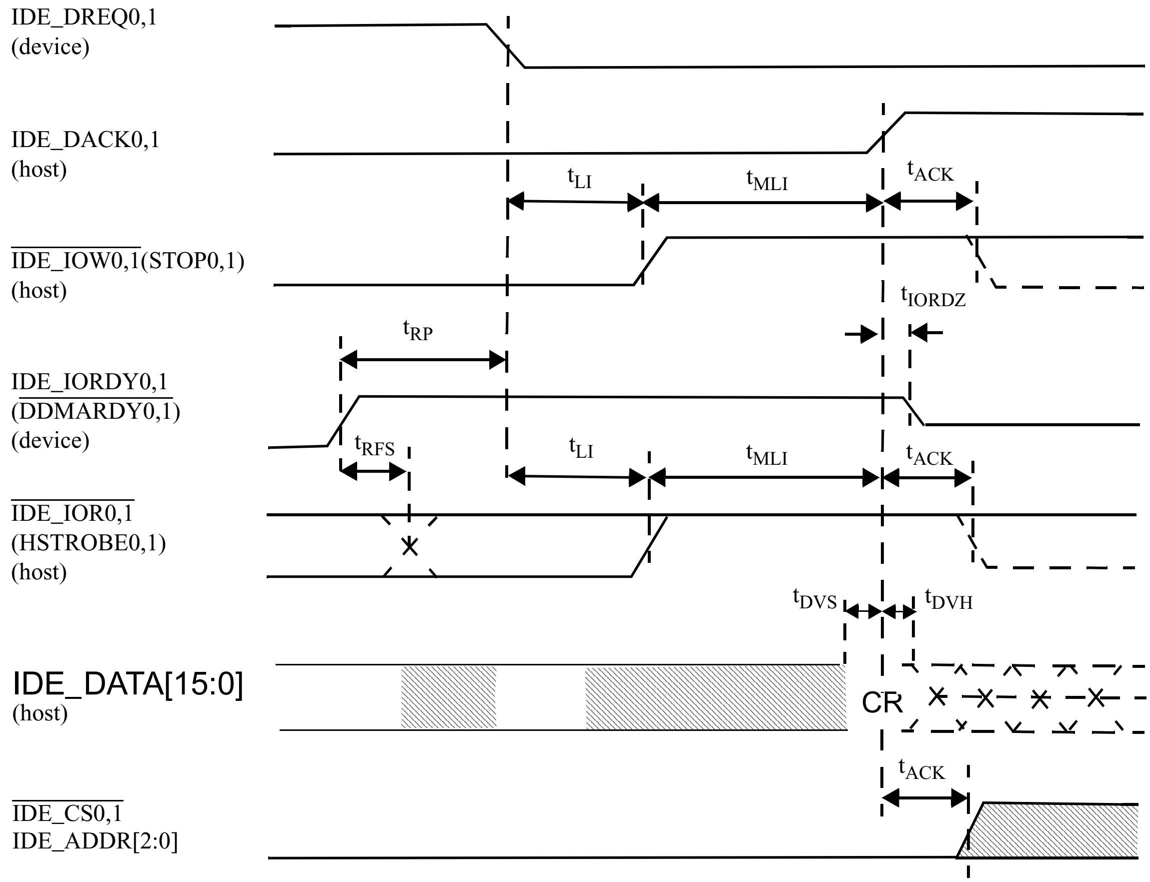


Figure 7-32 Device Terminating an Ultra DMA Data Out Burst

Note: The definitions for the IDE_IOW0,1(STOP0,1), IDE_IORDY0,1(DDMARDY0,1) and IDE_IOR0,1(HSTROBE0,1) signal lines are no longer in effect after IDE_DREQ0,1 and IDE_DACK0,1 are deasserted.

7.4.7. Universal Serial Bus (USB)

Table 7.36 Universal Serial Bus (USB)

Symbol	Parameter	Conditions	Min	Max	Reference
Full Speed Source[1, 2]					
t_{USB_R1}	D+_Port1,2, D–_Port1,2 Driver Rise Time	(Monotonic) from 10% to 90% of the D_Port lines	4nS	20nS	Figure 7-33
t_{USB_F1}	D+_Port1,2, D–_Port1,2 Driver Fall Time	(Monotonic) from 90% to 10% of the D_Port lines	4nS	20nS	Figure 7-33
t_{USB_FRFM}	Rise/Fall time matching		90%	110%	
t_{USB_FSDR}	Full-speed data rate	Average bit rate 12 Mbps ± 0.25%	11.97M bps	12.03M bps	
t_{USB_FSF}	Full-speed frame interval	1.0 mS ±0.05%	0.9995 mS	1.0005 mS	
t_{period_F}	Full-speed period between data bits	Average bit rate 12 Mbps	83.1nS	83.5nS	
t_{USB_DOR}	Driver-output resistance	Steady-state drive	28Ω	43Ω	
t_{USB_DJ11}	Source differential driver jitter[3, 4] for consecutive transition	—	–3.5nS	3.5nS	Figure 7-34
t_{USB_DJ12}	Source differential driver jitter[3, 4] for paired transitions	—	–4.0nS	4.0nS	Figure 7-34
t_{USB_SE1}	Source EOP width[4, 5]	—	160nS	175nS	Figure 7-35
t_{USB_DE1}	Differential to EOP transition skew[4, 5]	—	–2nS	5nS	Figure 7-35
t_{USB_RJ11}	Receiver data jitter tolerance[4] for consecutive transition	—	–18.5nS	18.5nS	Figure 7-36
t_{USB_RJ12}	Receiver data jitter tolerance[4] for paired transitions	—	–9nS	9nS	Figure 7-36
Full Speed Receiver EOP Width[4]					
t_{USB_RE11}	Must reject as EOP[5]	—	—	40nS	Figure 7-35
t_{USB_RE12}	Must accept as EOP[5]	—	82nS	—	Figure 7-35

Table 7.36 Universal Serial Bus (USB) (cont.)

Symbol	Parameter	Conditions	Min	Max	Reference
Low Speed Source[1, 6]					
t_{USB_R2}	D+_Port1,2, D–_Port1,2 Driver Rise Time	(Monotonic) from 10% to 90% of the D_Port lines	75nS	300[6]nS	Figure 7-33
t_{USB_F2}	D+_Port1,2, D–_Port1,2 Driver Fall Time	(Monotonic) from 90% to 10% of the D_Port lines	75nS	300[6]nS	Figure 7-33
t_{USB_LRFM}	Low-speed Rise/Fall time matching		80%	120%	
t_{USB_LSDR}	Low-speed data rate	Average bit rate 1.5 Mbps ± 1.5%	1.4775 Mbps	1.5225 Mbps	
t_{PERIOD_L}	Low-speed period	at 1.5 Mbps	0.657µS	0.677µS	
t_{USB_DJD21}	Source differential driver jitter[4] for consecutive transactions	Host (downstream)	–75nS	75nS	
t_{USB_DJD22}	Source differential driver jitter[4] for paired transactions	Host (downstream)	–45nS	45nS	Figure 7-34
t_{USB_DJU21}	Source differential driver jitter[4] for consecutive transaction	Function (downstream)	–95nS	95nS	Figure 7-34
t_{USB_DJU22}	Source differential driver jitter[4] for paired transactions	Function (downstream)	–150nS	150nS	Figure 7-34
t_{USB_SE2}	Source EOP width[4, 5]		1.25µS	1.5µS	Figure 7-35
t_{USB_DE2}	Differential to EOP [5] transition skew		–40nS	100nS	Figure 7-35
t_{USB_RJD21}	Receiver Data Jitter Tolerance[4] for consecutive transactions	Host (upstream)	–152nS	152nS	Figure 7-36
t_{USB_RJD22}	Receiver Data Jitter Tolerance[4] for paired transactions	Host (upstream)	–200nS	200nS	Figure 7-36
t_{USB_RJU21}	Receiver Data Jitter Tolerance[4] for consecutive transactions	Function (downstream)	–75nS	75nS	Figure 7-36
t_{USB_RJU22}	Receiver Data Jitter Tolerance[4] for paired transactions	Function (downstream)	–45nS	45nS	Figure 7-36
Low Speed Receiver EOP Width[5]					
t_{USB_RE21}	Must reject as EOP	—	—	330nS	Figure 7-35
t_{USB_RE22}	Must accept as EOP	—	675nS	—	Figure 7-35

1. Unless otherwise specified, all timings use a 50 pF capacitive load (C_L) to ground.
2. Full-speed timing has a 1.5 KΩ pull-up to 2.8 V on the D+_Port1,2 lines.
3. Timing difference between the differential data signals (D+_PORT1,2 and D–_PORT1,2).
4. Measured at the crossover point of differential data signals (D+_PORT1,2 and D–_PORT1,2).
5. EOP is the End of Packet where D+_PORT[t] = D–_PORT = SE0. SE0 occurs when output level voltage ≤V_{SE} (Min).
6. C_L = 350 pF.

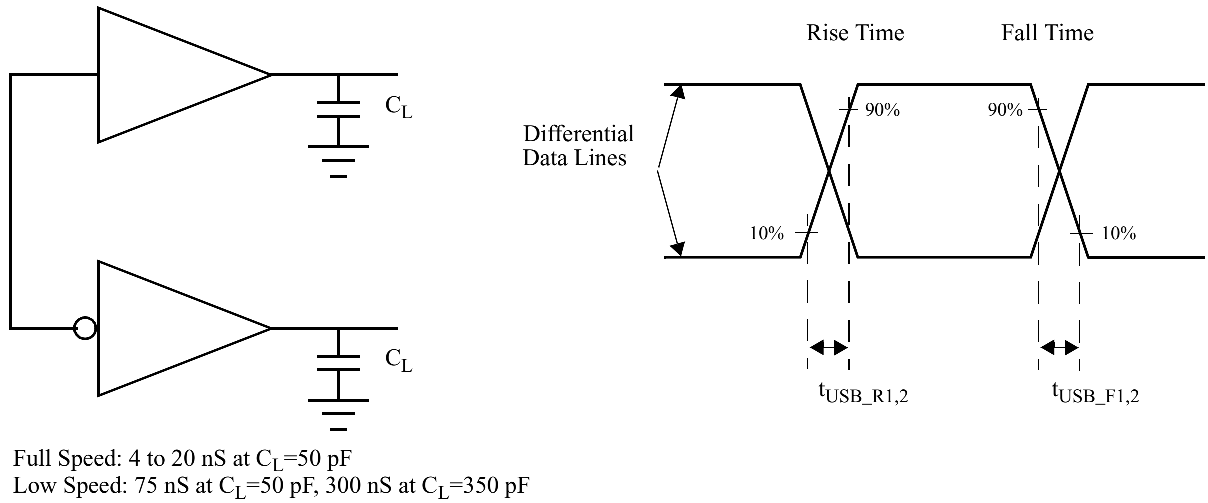


Figure 7-33 Data Signal Rise and Fall Time

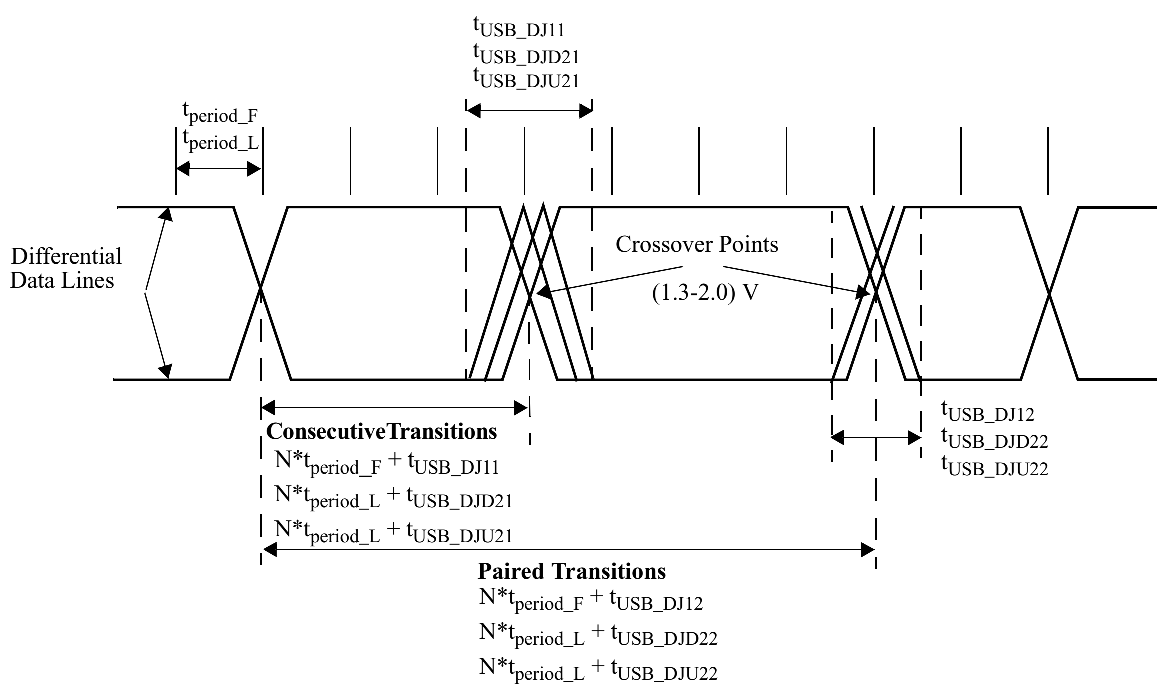


Figure 7-34 Source Differential Data Jitter

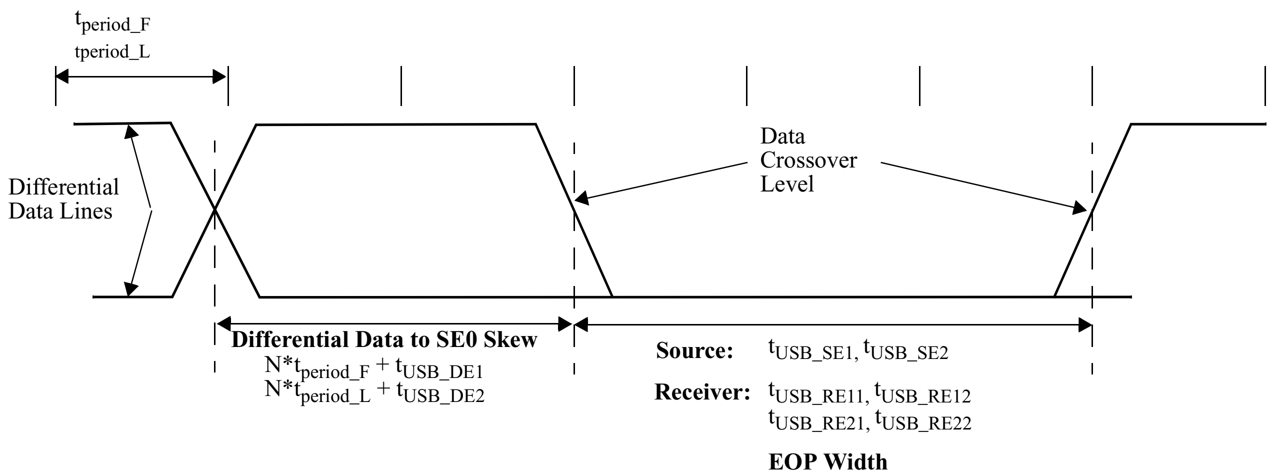


Figure 7-35 EOP Width Timing

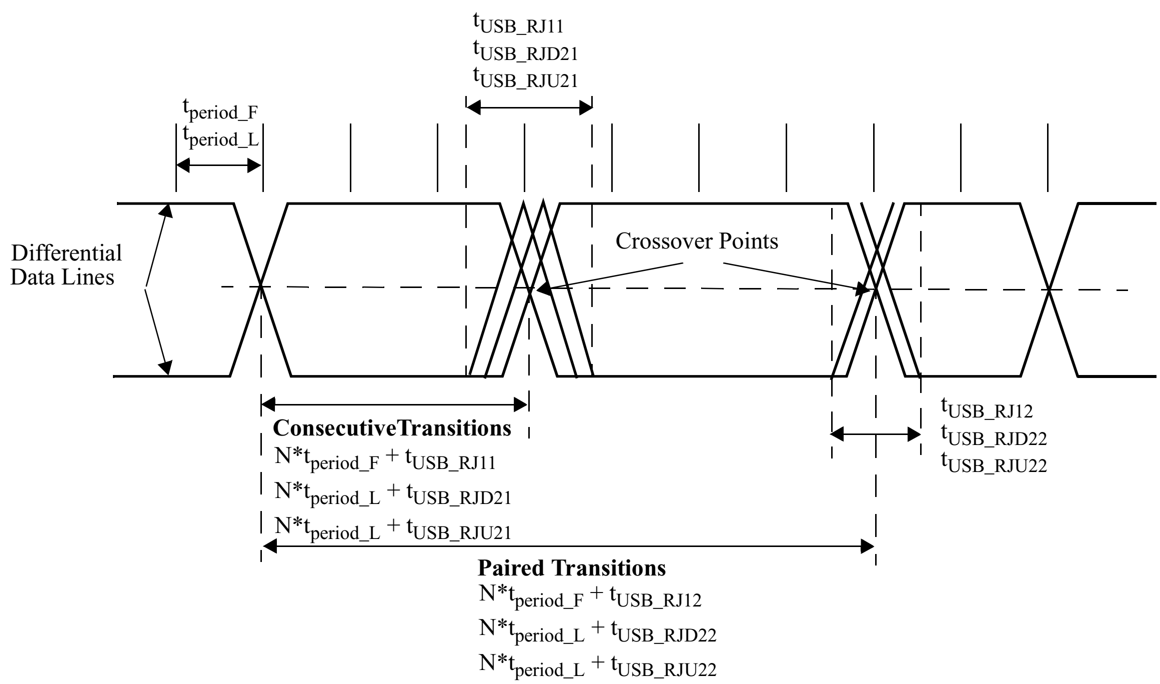


Figure 7-36 Receiver Jitter Tolerance

7.4.8. Serial Port (UART)

Table 7.37 UART, Sharp-IR, SIR, and Consumer Remote Control Parameters

Symbol	Parameter	Conditions	Min	Max	Figure
t_{BT}	Single Bit Time in UART and Sharp-IR	Transmitter	t_{BTN} –25nS[1]	t_{BTN} + 25nS	Figure 7-37
		Receiver	t_{BTN} –2%	t_{BTN} + 2%	
t_{CMW}	Modulation Signal Pulse Width in Sharp-IR and Consumer Remote Control	Transmitter	t_{CWN} –25nS[2]	t_{CWN} + 25nS	Figure 7-37
		Receiver	500nS		
t_{CMP}	Modulation Signal Period in Sharp-IR and Consumer Remote Control	Transmitter	t_{CPN} –25nS[3]	t_{CPN} + 25nS	Figure 7-37
		Receiver	t_{MMIN}[4]	t_{MMAX}[4]	
t_{SPW}	SIR Signal Pulse Width	Transmitter, Variable	$(^3/_{16}) \times t_{BTN}$ –15 [1]	$(^3/_{16}) \times t_{BTN}$ +15 [1]	Figure 7-37
		Transmitter, Fixed	1.48µS	1.78µS	
		Receiver	1µS	—	
S_{DRT}	SIR Data Rate Tolerance % of Nominal Data Rate	Transmitter	—	±0.87%	
		Receiver	—	±2.0%	
t_{SJT}	SIR Leading Edge Jitter % of Nominal Bit Duration	Transmitter	—	±2.5%	
		Receiver	—	±6.5%	

1. t_{BTN} is the nominal bit time in UART, Sharp-IR, SIR and Consumer Remote Control modes. It is determined by the setting of the Baud Generator Divisor registers.
2. t_{CWN} is the nominal pulse width of the modulation signal for Sharp-IR and Consumer Remote Control modes. It is determined by the MCPW field (bits 7-5) of the IRTXMC register and the TXHSC bit (bit 2) of the RCCFG register.
3. t_{CPN} is the nominal period of the modulation signal for Sharp-IR and Consumer Remote Control modes. It is determined by the MCFR field (bits 4-0) of the IRTXMC register and the TXHSC bit (bit 2) of the RCCFG register.
4. t_{MMIN} and t_{MMAX} define the time range within which the period of the incoming subcarrier signal has to fall in order for the signal to be accepted by the receiver. These time values are determined by the contents of register IRRXDC and the setting of the RXHSC bit (bit 5) of the RCCFG register.

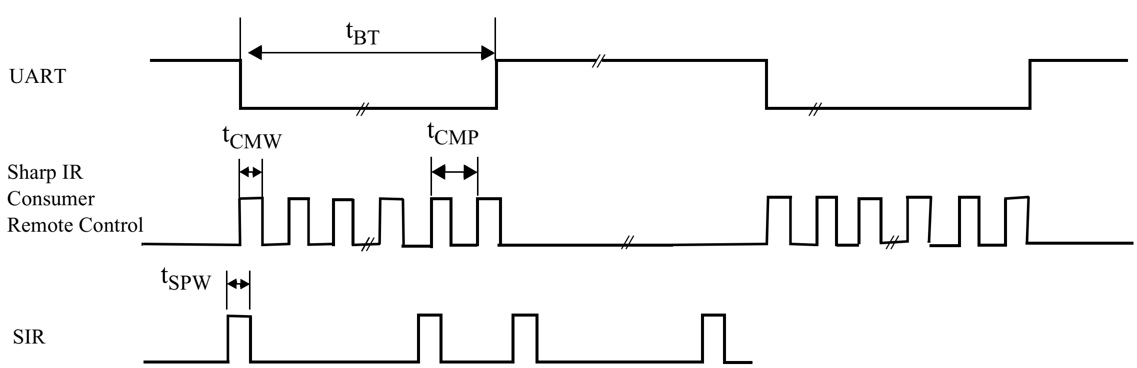


Figure 7-37 UART, Sharp-IR, SIR, and Consumer Remote Control Timing

7.4.9. Fast IR Port Timing

Table 7.38 Fast IR Port Timing Parameters

Symbol	Parameter	Conditions	Min	Max
t_{MPW}	MIR Signal Pulse Width	Transmitter	t_{MWN}-25nS[1]	t_{MWN}+25nS
		Receiver	60nS	—
M_{DRT}	MIR Transmitter Data Rate Tolerance			±0.1%
t_{MJT}	MIR Receiver Edge Jitter, % of Nominal Bit Duration			±2.9%
t_{FPW}	FIR Signal Pulse Width	Transmitter	120nS	130nS
		Receiver	90nS	160nS
t_{FDPW}	FIR Signal Double Pulse Width	Transmitter	245nS	255nS
		Receiver	215nS	285nS
F_{DRT}	FIR Transmitter Data Rate Tolerance			±0.01%
t_{FJT}	FIR Receiver Edge Jitter, % of Nominal Bit Duration			±4.0%

1. t_{MWN} is the nominal pulse width for MIR mode. It is determined by the M_PWID field (bits 4-0) in the MIR_PW register at offset 01h in bank 6 of logical device 5.

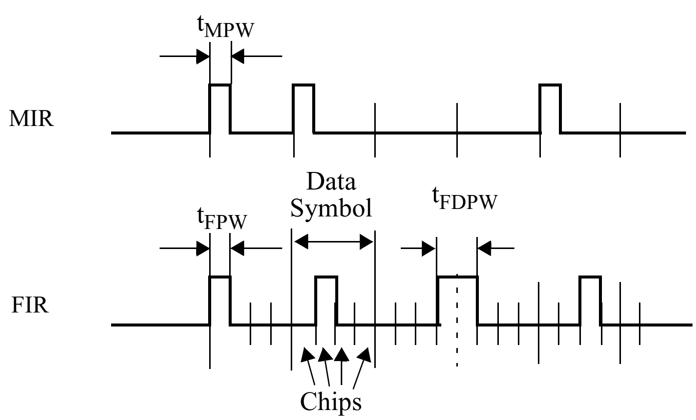


Figure 7-38 Fast IR Timing (MIR and FIR)

7.4.10. JTAG Timing

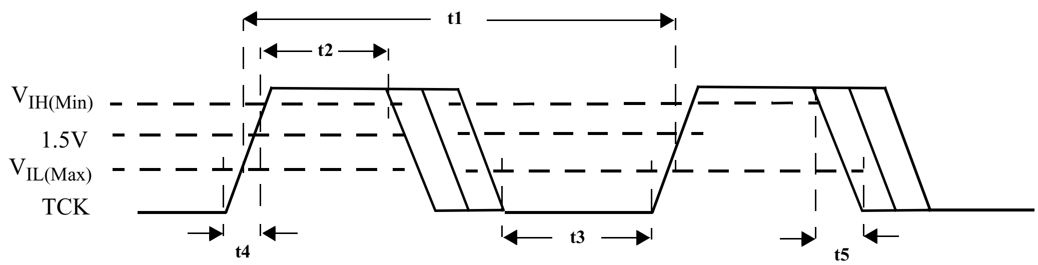


Figure 7-39 TCK Timing and Measurement Points

Table 7.39 JTAG Timing

Symbol	Parameter	Min	Max	Reference
	TCK Frequency (MHz)	—	25MHz	Figure 7-39
t_1	TCK Period	40ns	—	Figure 7-39
t_2	TCK High Time	10ns	—	Figure 7-39
t_3	TCK :Low Time	10ns	—	Figure 7-39
t_4	TCK Rise Time	—	4ns	Figure 7-39
t_5	TCK Fall Time	—	4ns	Figure 7-39
t_6	TDO Valid Delay	3ns	25ns	Figure 7-39
t_7	Non-test Outputs Valid Delay	3ns	25ns	Figure 7-39
t_8	TDO Float Delay	—	30ns	
t_9	Non-test Outputs Float Delay	—	36ns	
t_{10}	TDI, TMS Setup Time	8ns	—	
t_{11}	Non-test Inputs Setup Time	8ns	—	
t_{12}	TDI, TMS Hold Time	7ns	—	
t_{13}	Non-test Inputs Hold Time	7ns	—	

7.4.11. GPIO Timing

Table 7.40 GPIO Timing

Symbol	Parameter	Min	Max	Unit
t_1	pciclk to GPIO output			nS
t_2	GPIO input set up to pciclk			nS
t_3	GPIO input hold from to pciclk			nS

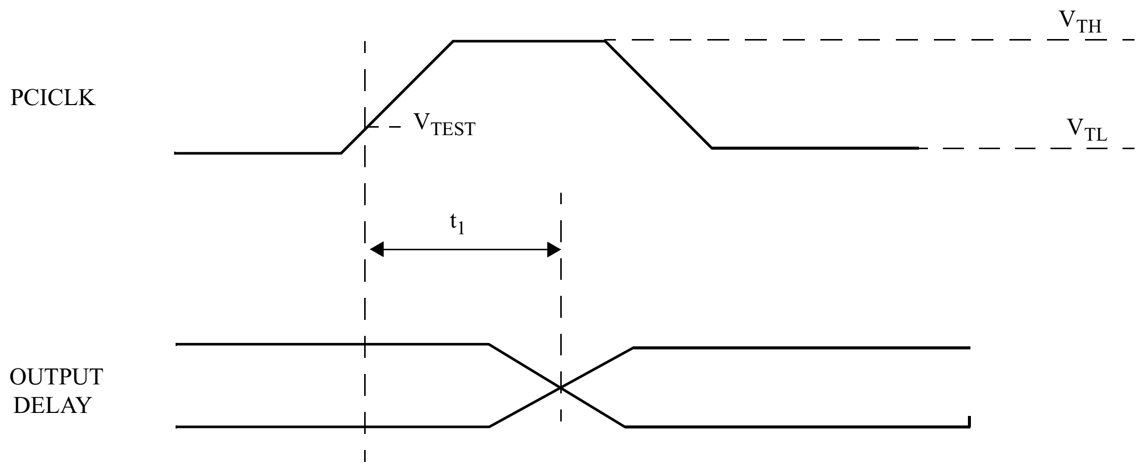


Figure 7-40 GPIO Output Timing Measurement Conditions

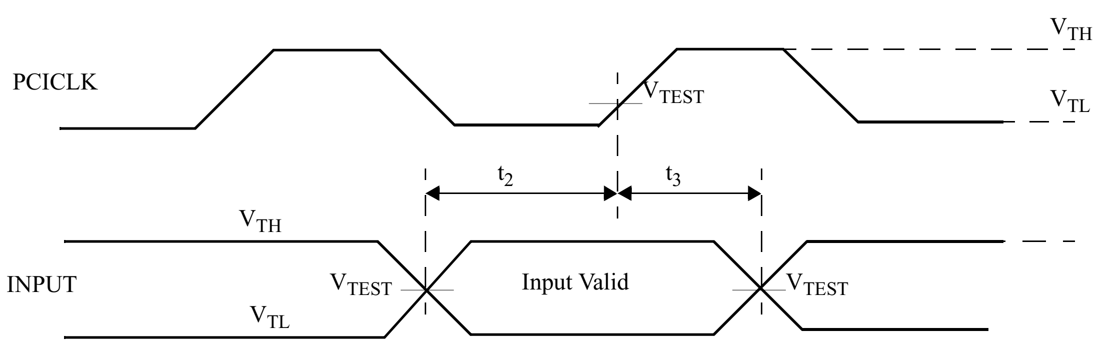


Figure 7-41 GPIO Input Timing Measurement Conditions

7.4.12. Floppy Disk Interface

Reset Timing

Table 7.41 Floppy Disk Reset Timing

Symbol	Parameter	Min	Max
t_{RW}	Reset Width [1]	100µS	—
t_{SRC}	Reset to Control Inactive [2]	—	300nS

1. The software reset pulse width is 100 nsec.
2. Not tested. Guaranteed by design.

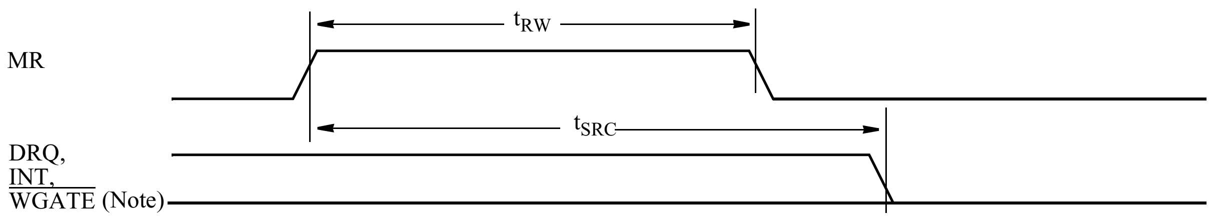


Figure 7-42 Floppy Disk Reset Timing

Note: In PC-AT mode, the DRQ and IRQ signals of the FDC are in TRI-STATE after time t_{SRC}.

Write Data Timing

Table 7.42 Floppy Disk Write Data Timing

Symbol	Parameter	Min	Max
t_{HDH}	$\overline{\text{HDSEL}}$ Hold from $\overline{\text{WGATE}}$ Inactive[1]	750µS	—
t_{HDS}	$\overline{\text{HDSEL}}$ Setup to $\overline{\text{WGATE}}$ Active[a]	100µS	—
t_{WDW}	Write Data Pulse Width	See Table 7.43	—

1. Not tested. Guaranteed by design.

Table 7.43 Write Data Timing – Minimum t_{WDW} Values

Data Rate	t_{DRP}	t_{WDW}	t_{WDW} Value
1 Mbps	1000	2 x t_{ICP} [1]	250nS
500 Kbps	2000	2 x t_{ICP} [1]	250nS
300 Kbps	3333	2 x t_{ICP} [1]	375nS
250 Kbps	4000	2 x t_{ICP} [1]	500nS

1. t_{ICP} is the internal clock period defined in TABLE TBD.

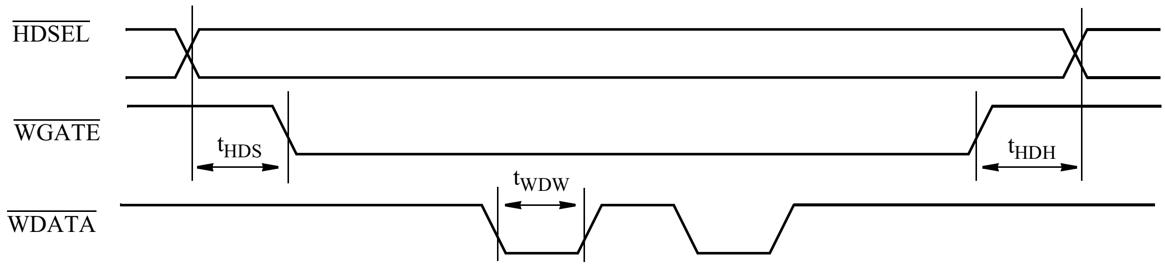


Figure 7-43 Write Data Timing

Drive Control Timing

Table 7.44 Drive Control Timing

Symbol	Parameter	Min	Max
t_{DRV}	$\overline{\text{DR1,0}}$ and $\overline{\text{MTR1,0}}$ from End of $\overline{\text{WR}}$	—	110nS
t_{DST}	$\overline{\text{DIR}}$ Setup to $\overline{\text{STEP}}$ Active [1]	6µS	—
t_{IW}	Index Pulse Width	100nS	—
t_{STD}	$\overline{\text{DIR}}$ Hold from $\overline{\text{STEP}}$ Inactive	t_{STR}mS	—
t_{STP}	$\overline{\text{STEP}}$ Active High Pulse Width	8µS	—
t_{STR}	$\overline{\text{STEP}}$ Rate Time (See TABLE TBD.)	1mS	—

1. Not tested. Guaranteed by design.

Figure 7-44 Drive Control Timing

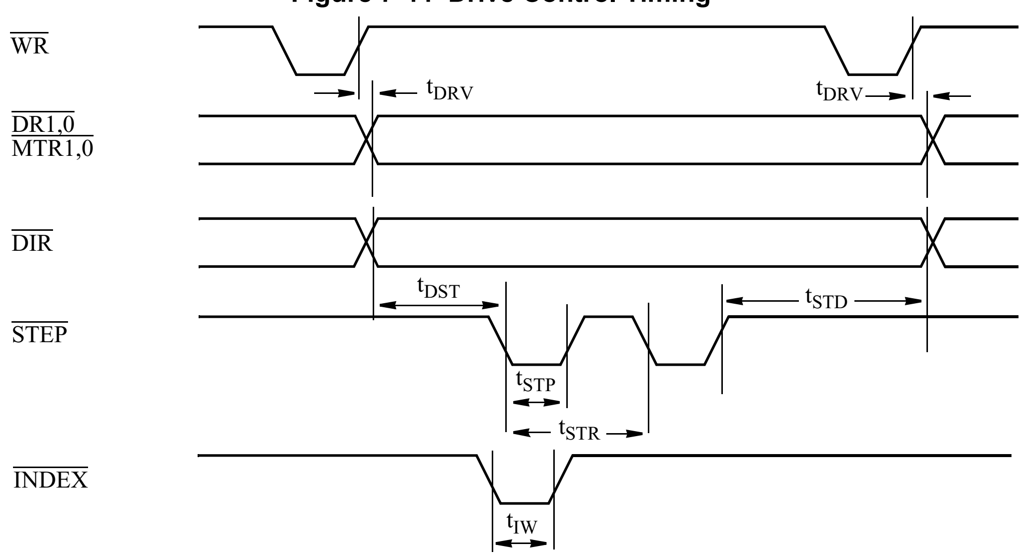


Table 7.45 Read Data Timing

Parameter	Symbol	Min	Max
Read Data Pulse Width	t_{RDW}	50nS	

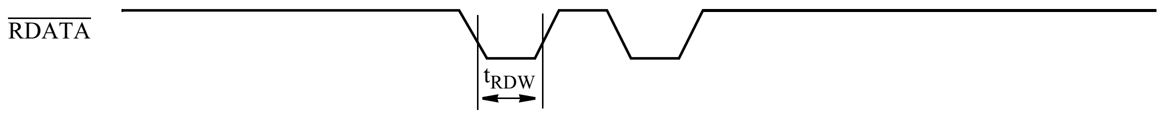


Figure 7-45 Read Data Timing

7.4.13. Keyboard and Mouse Interface

All Keyboard and Mouse timing is not 100% tested. Timing is guaranteed by design.

Relates to KCLK, KDAT, KBLOCK, MCLK, MDAT

Table 7.46 KBC Signals Rising and Falling

Symbol	Parameter	Condition	Min	Max	Reference
t_f	signal fall time			100nS	Figure 7-46
t_r	signal rise time			100nS	Figure 7-46

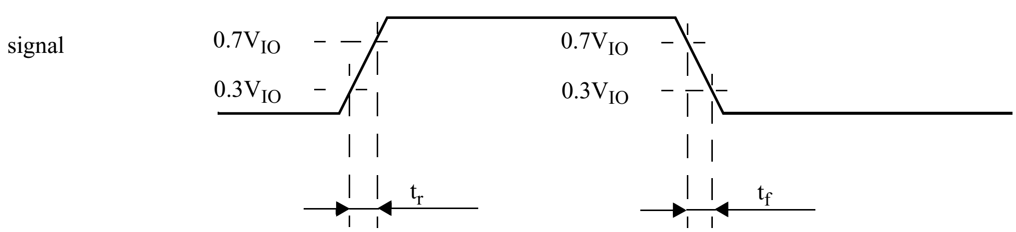


Figure 7-46 KBC Signals Rising and Falling

7.4.14. Parallel Port

Parallel Port Timing

Table 7.47 Standard Parallel Port Timing

Symbol	Parameter	Conditions	Typ	Max
t_{PDH}	Port Data Hold	These times are system dependent and are therefore not tested.	500nS	—
t_{PDS}	Port Data Setup	These times are system dependent and are therefore not tested.	500nS	—
t_{PILa}	Port Active Low Interrupt, Active	—	—	33nS
t_{PILia}	Port Active Low Interrupt, Inactive	—	—	33nS
t_{PIHa}	Port Active High Interrupt, Active	—	—	33nS
t_{PIHia}	Port Active High Interrupt, Inactive	—	—	33nS
t_{PIz}	Port Active High Interrupt, TRISTATE	—	—	33nS
t_{SW}	Strobe Width	These times are system dependent and are therefore not tested.	500nS	—

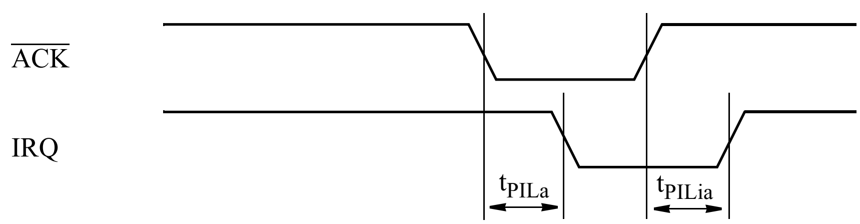


Figure 7-47 Parallel Port Interrupt Timing (Compatible Mode)

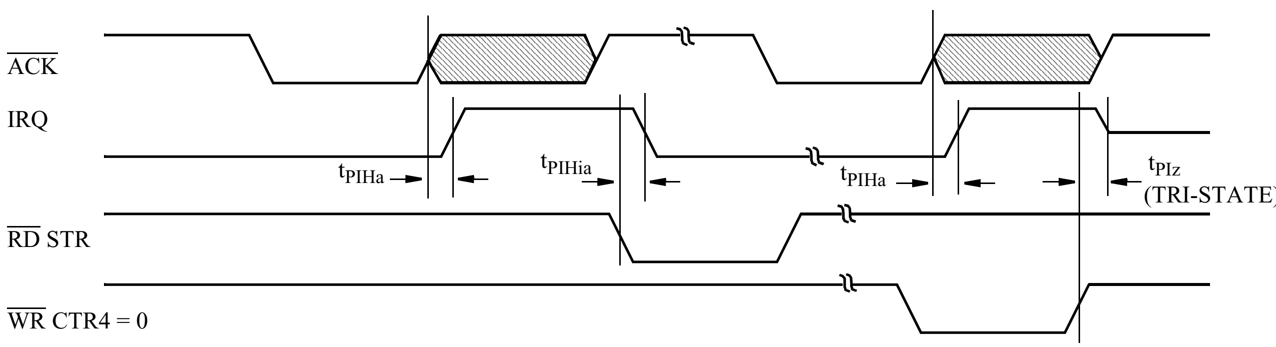


Figure 7-48 Parallel Port Interrupt Timing (Extended Mode)

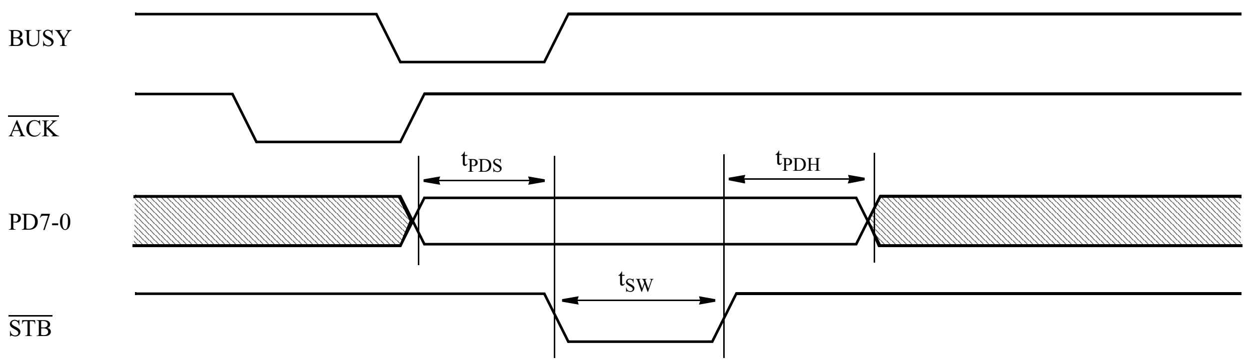


Figure 7-49 Typical Parallel Port Data Exchange

Enhanced Parallel Port 1.7 Timing

Table 7.48 Enhanced Parallel Port 1.7 Timing Parameters

Symbol	Parameter	Min	Max
t_{WW17}	$\overline{\text{WRITE}}$ Active or Inactive from $\overline{\text{WR}}$ Active or Inactive	—	45nS
t_{WST17}	$\overline{\text{DSTRB}}$ or $\overline{\text{ASTRB}}$ Active or Inactive from $\overline{\text{WR}}$ or $\overline{\text{RD}}$ Active or Inactive[1]	—	45nS
t_{WEST17}	$\overline{\text{DSTRB}}$ or $\overline{\text{ASTRB}}$ Active after $\overline{\text{WRITE}}$ Becomes Active	0nS	—
t_{WPD17h}	PD7-0 Hold after $\overline{\text{WRITE}}$ Becomes Inactive	50nS	—
t_{HRW17}	IOCHRDY Active or Inactive after $\overline{\text{WAIT}}$ Becomes Active or Inactive	—	40nS
t_{WPDS17}	PD7-0 Valid after $\overline{\text{WRITE}}$ Becomes Active[2]	—	15nS
t_{EPDW17}	PD7-0 Valid Width	80nS	—
t_{EPD17h}	PD7-0 Hold after $\overline{\text{DSTRB}}$ or $\overline{\text{ASTRB}}$ Becomes Inactive	0nS	—
t_{ZWS17a}	$\overline{\text{ZWS}}$ Valid after $\overline{\text{WR}}$ or $\overline{\text{RD}}$ Active	—	45nS
t_{ZWS17h}	$\overline{\text{ZWS}}$ Hold after $\overline{\text{WR}}$ or $\overline{\text{RD}}$ Inactive	0nS	—

1. The design guarantees that $\overline{\text{WRITE}}$ will not change from low to high before $\overline{\text{DSTRB}}$, or $\overline{\text{ASTRB}}$, goes from low to high.
2. D7-0 is stable 15 nsec before $\overline{\text{WR}}$ becomes active.

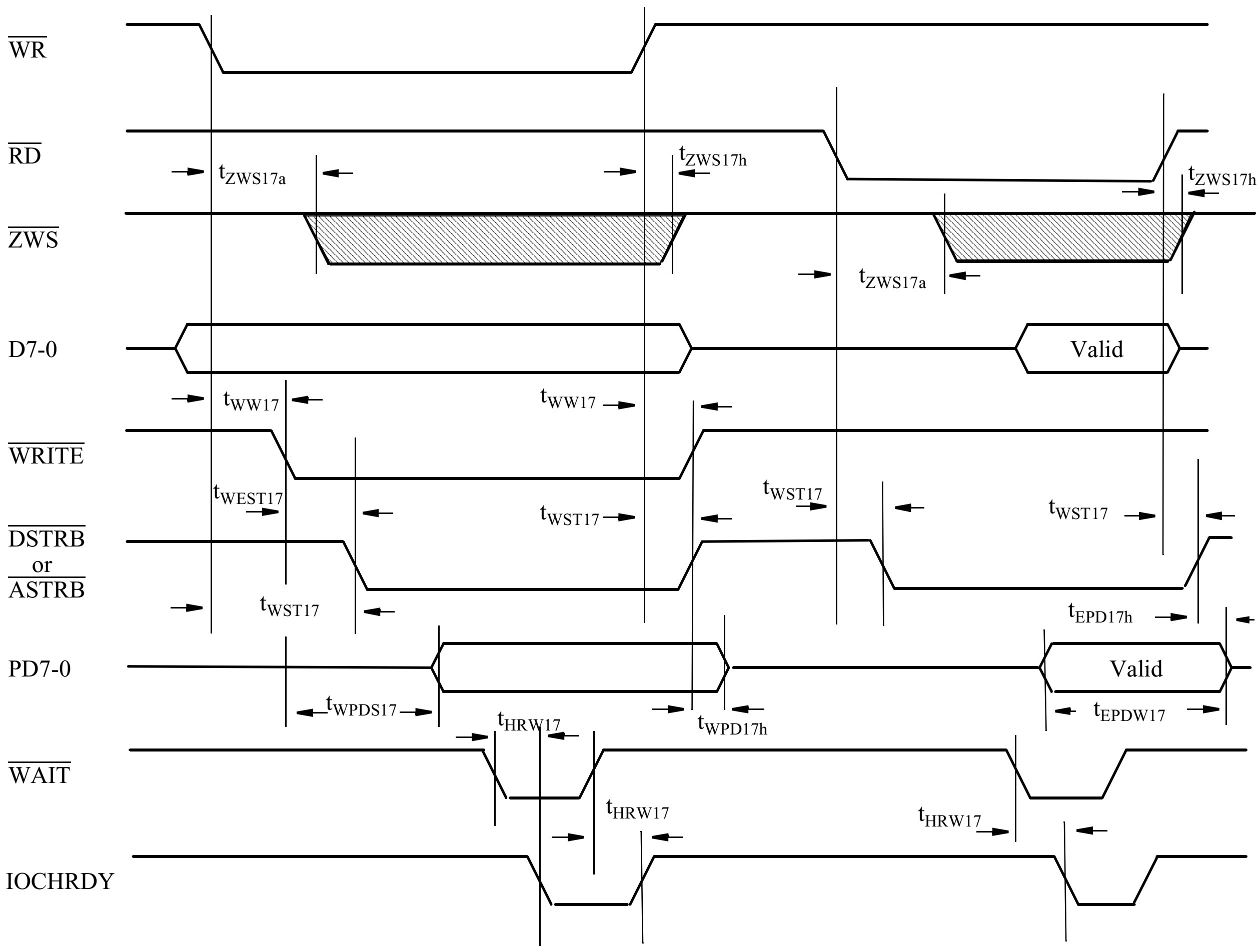


Figure 7-50 Enhanced Parallel Port 1.7 Timing

Enhanced Parallel Port 1.9 Timing

Table 7.49 Enhanced Parallel Port 1.9 Timing Parameters

Symbol	Parameter	Min	Max
t_{WW119a}	$\overline{\text{WRITE}}$ Active from $\overline{\text{WR}}$ Active or $\overline{\text{WAIT}}$ Low [1]	—	45nS
t_{WW19ia}	$\overline{\text{WRITE}}$ Inactive from $\overline{\text{WAIT}}$ Low	—	45nS
t_{WST19a}	$\overline{\text{DSTRB}}$ or $\overline{\text{ASTRB}}$ Active from $\overline{\text{WR}}$ or $\overline{\text{RD}}$ Active or $\overline{\text{WAIT}}$ Low1 [2]	—	65nS
$t_{WST19ia}$	$\overline{\text{DSTRB}}$ or $\overline{\text{ASTRB}}$ Inactive from $\overline{\text{WR}}$ or $\overline{\text{RD}}$ High	—	45nS
t_{WEST19}	$\overline{\text{DSTRB}}$ or $\overline{\text{ASTRB}}$ Active after $\overline{\text{WRITE}}$ Active	10nS	—

Table 7.49 Enhanced Parallel Port 1.9 Timing Parameters

Symbol	Parameter	Min	Max
t_{WPD19h}	PD7-0 Hold after $\overline{WRITE}$ Inactive	0nS	—
t_{HRW19}	IOCHRDY Active after $\overline{WR}$ or $\overline{RD}$ Active or Inactive after $\overline{WAIT}$ High	—	40nS
t_{WPDS19}	PD7-0 Valid after $\overline{WRITE}$ Active [3]	—	15nS
t_{EPDW19}	PD7-0 Valid Width	80nS	—
t_{EPD19h}	PD7-0 Hold after $\overline{DSTRB}$ or $\overline{ASTRB}$ Inactive	0nS	—
t_{ZWS19a}	$\overline{ZWS}$ Valid after $\overline{WR}$ or $\overline{RD}$ Active	—	45nS
t_{ZWS19h}	$\overline{ZWS}$ Hold after $\overline{WR}$ or $\overline{RD}$ Inactive	0nS	—

1. When $\overline{WAIT}$ is low, t_{WST19a} and t_{WW19a} are measured after $\overline{WR}$ or $\overline{RD}$ becomes active; else t_{WST19a} and t_{WW19a} are measured after $\overline{WAIT}$ becomes low.
2. The PC87307VUL design guarantees that $\overline{WRITE}$ will not change from low to high before $\overline{DSTRB}$, or $\overline{ASTRB}$, goes from low to high.
3. D7-0 is stable 15 nsec before $\overline{WR}$ becomes active.

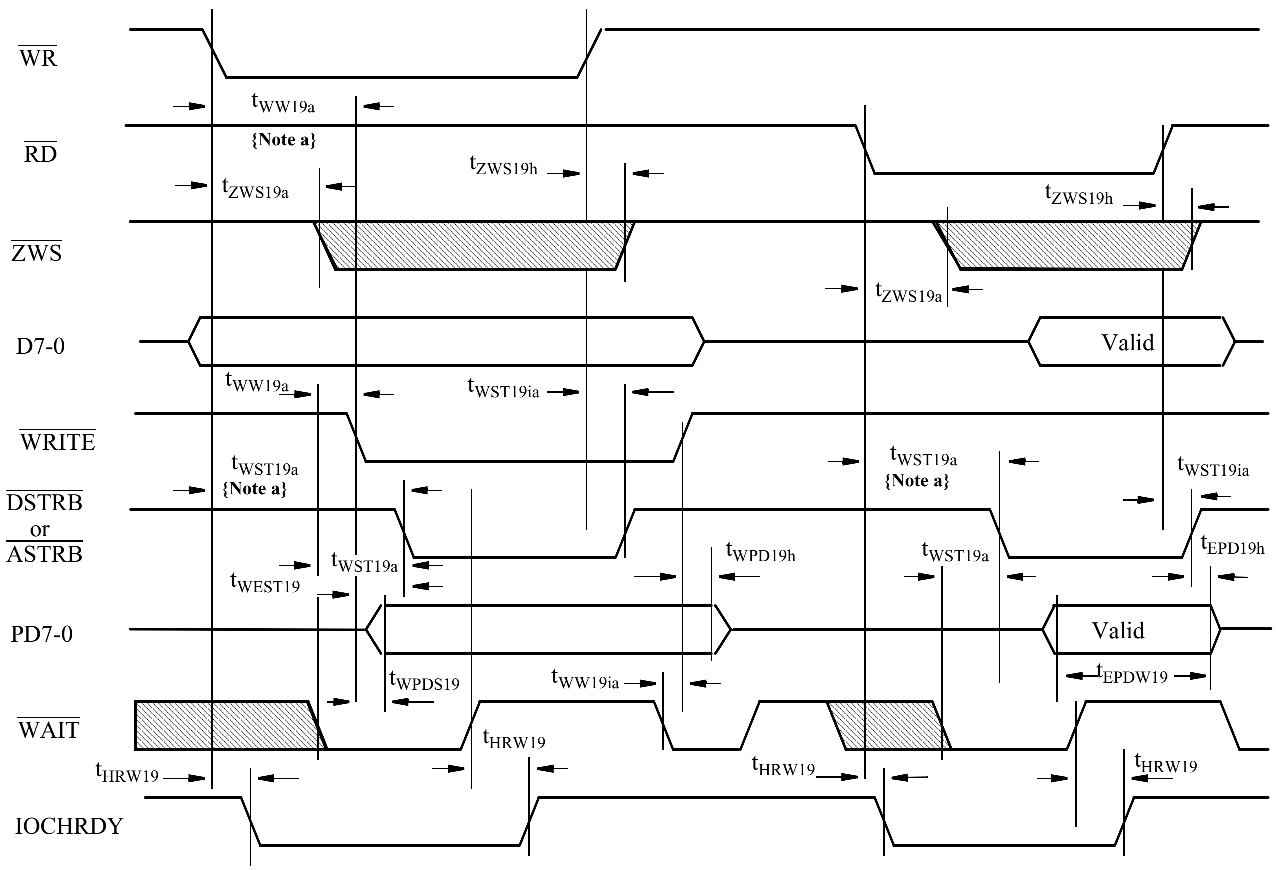


Figure 7-51 Enhanced Parallel Port 1.9 Timing

Extended Capabilities Port (ECP) Timing

Table 7.50 Extended Capabilities Port (ECP) Timing – Forward

Symbol	Parameter	Min	Max
t_{ECDSF}	Data Setup before $\overline{STB}$ Active	0nS	—
t_{ECDHF}	Data Hold after BUSY Inactive	0nS	—
t_{ECLHF}	BUSY Setup after Strobe Active	75nS	—
t_{ECHHF}	$\overline{STB}$ Inactive after BUSY Active	0S	1S
t_{ECHLF}	BUSY Setup after $\overline{STB}$ Active	0mS	35mS
t_{ECLLF}	Strobe Active after BUSY Inactive	0nS	—

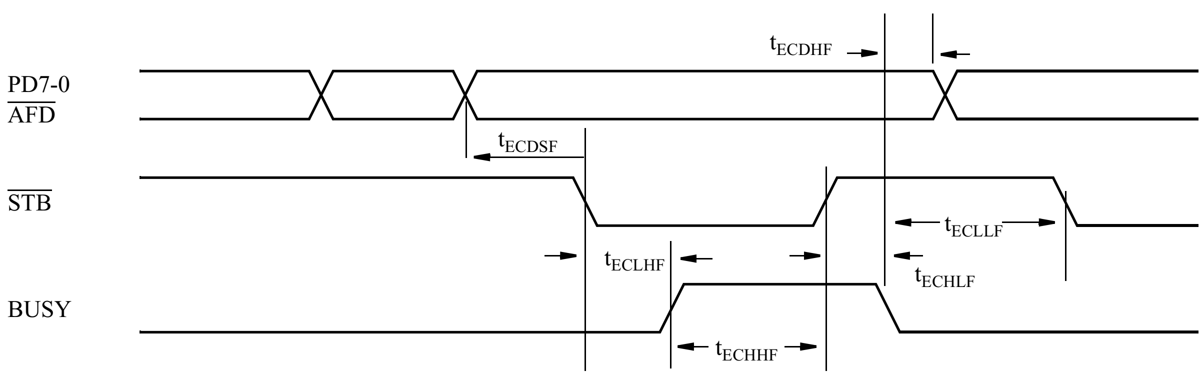


Figure 7-52 ECP Parallel Port Forward Timing Diagram

Table 7.51 Extended Capabilities Port (ECP) Timing – Backward

Symbol	Parameter	Min	Max
t_{ECDSB}	Data Setup before $\overline{ACK}$ Active	0nS	—
t_{ECDHB}	Data Hold after $\overline{AFD}$ Active	0nS	—
t_{ECLHB}	BUSY Setup after $\overline{ACK}$ Active	75nS	—
t_{ECHHB}	Strobe Inactive after $\overline{AFD}$ Inactive	0Sec	1Sec
t_{ECHLB}	BUSY Setup after $\overline{ACK}$ Active	0mS	35mS
t_{ECLLB}	Strobe Active after $\overline{AFD}$ Active	0nS	—

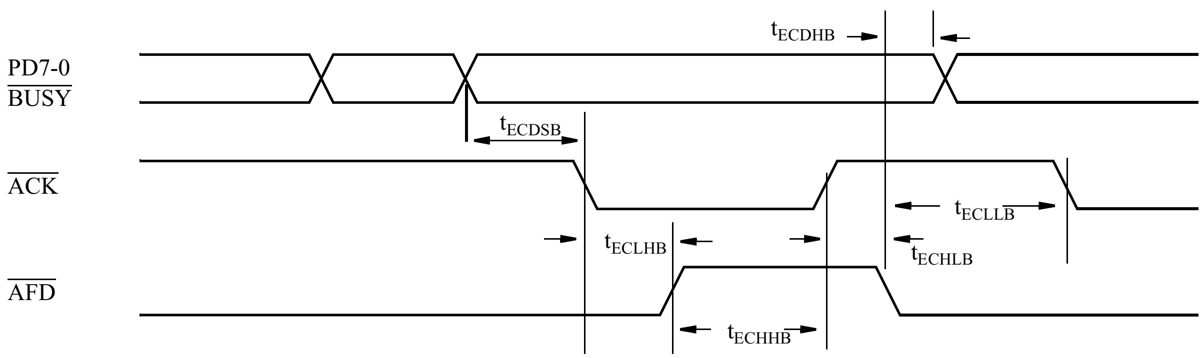


Figure 7-53 ECP Parallel Port Backward Timing Diagram

7.4.15. ZF-Logic

MEM_CS[3:0], IO_CS[3:0], PWM, WDO, WDI.

Table 7.52 ZF-Logic Signals

Symbol	Parameter	Min	Max	Unit
t_1	pciclk to ZF-Logic output			nS
t_2	ZF-Logic input set up to pciclk			nS
t_3	ZF- Logic input hold from to pciclk			nS

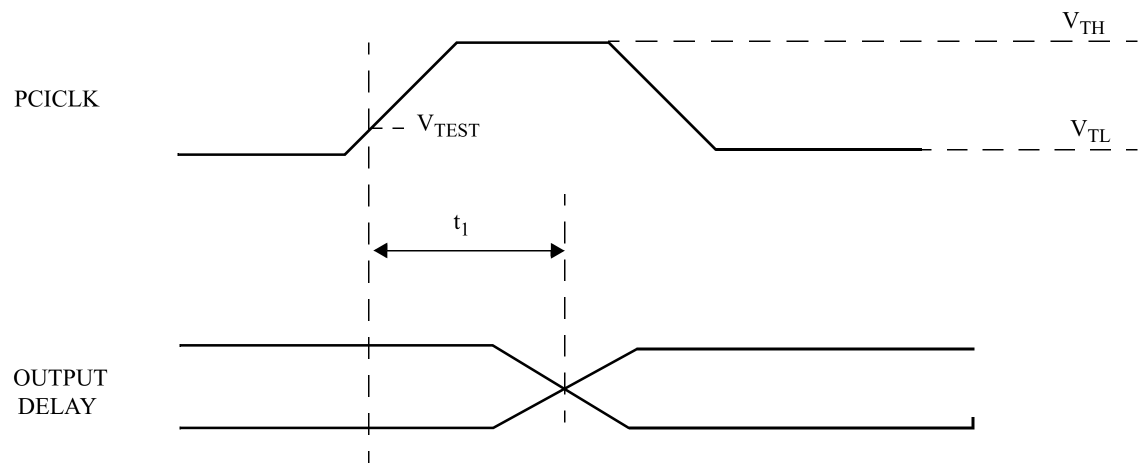


Figure 7-54 ZF-Logic Output Timing Measurement Conditions

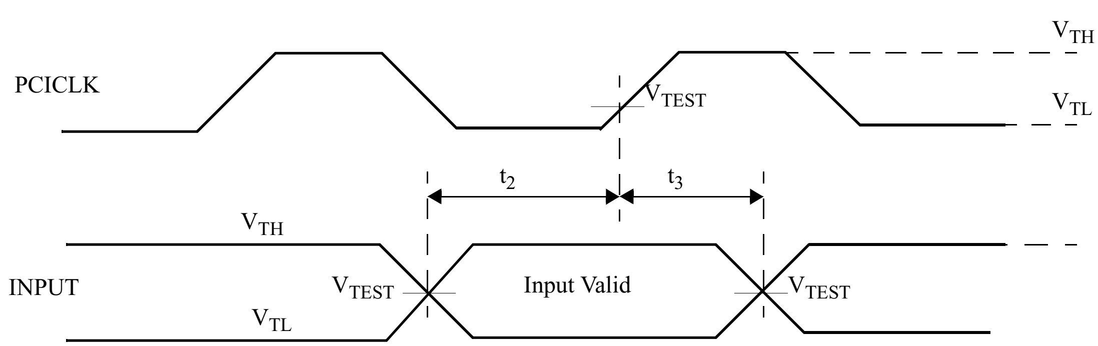


Figure 7-55 ZF-Logic Input Timing Measurement Conditions

8

8. Pinout Summary

ZFx86 has 308 functional IO pads including clocks, reset, JTAG, spares, and the following specialized power pins:

- Real-time Clock Battery
- USB Power
- USB Ground

The following tables are located in this chapter:

At the package level, the utilization is as follows:

Table 8.1 Pin Utilization

Use of Pins	Number of Pins
IO 3.3 volt power	16
Core 2.5 volt power	20
IO and Core ground	44
Functional IO	308
Total	**388**

Naming Conventions

*_n indicates active low
*_c indicates a clock

8.1. Pad Assignments

This chapter details all ZFx86 pad assignments including power and ground. Each pad assignment consists of the following items:

Package Ref: The package ball assigned to the pad as illustrated in 'ZFx86 Package - Solder Balls' on page 534.

Name: Pad name as it appears in the net list.

Comment: As appropriate.

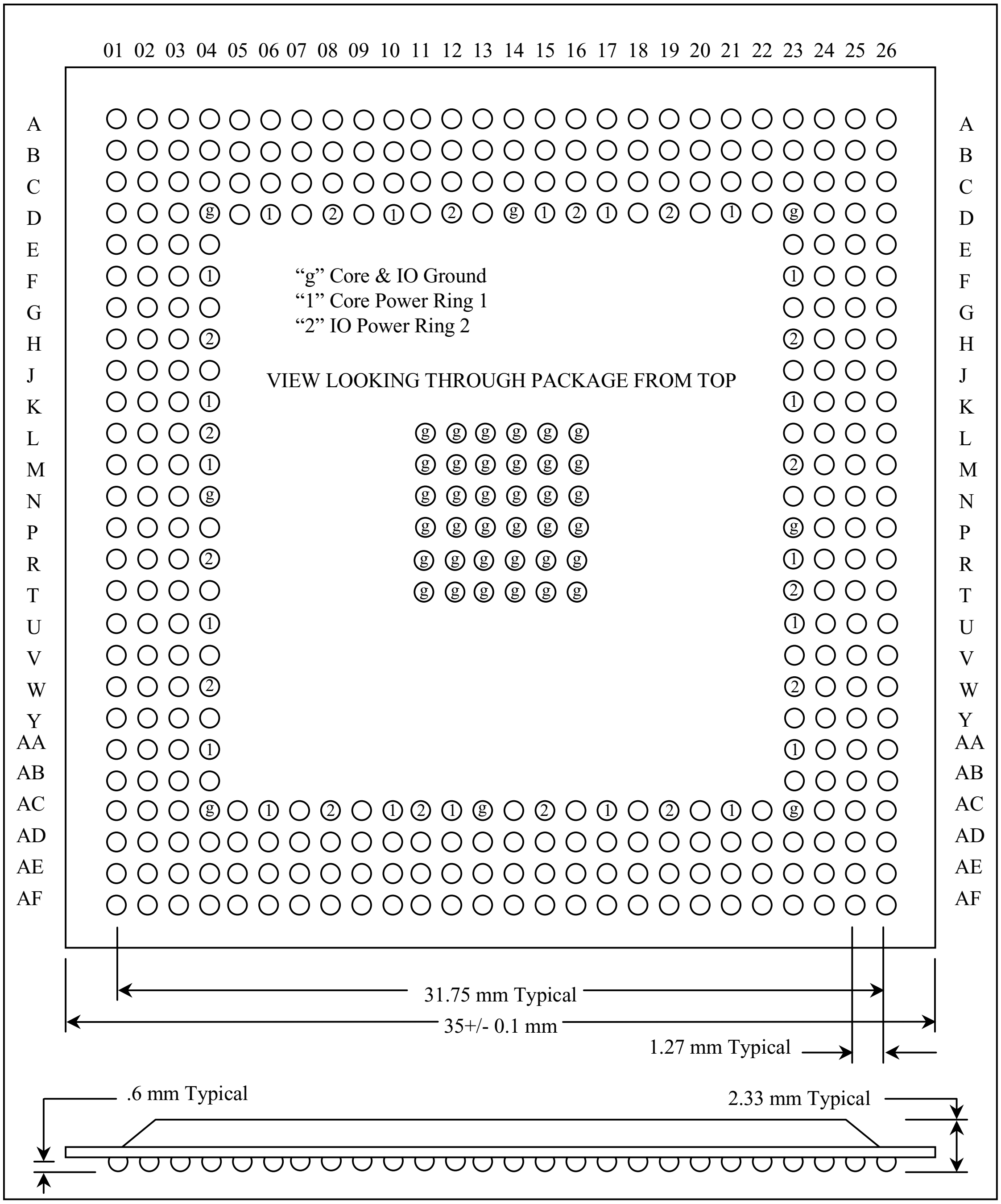


Figure 8-1 ZFx86 Package - Solder Balls

Note: Solder ball diameter is 0.75 millimeter.

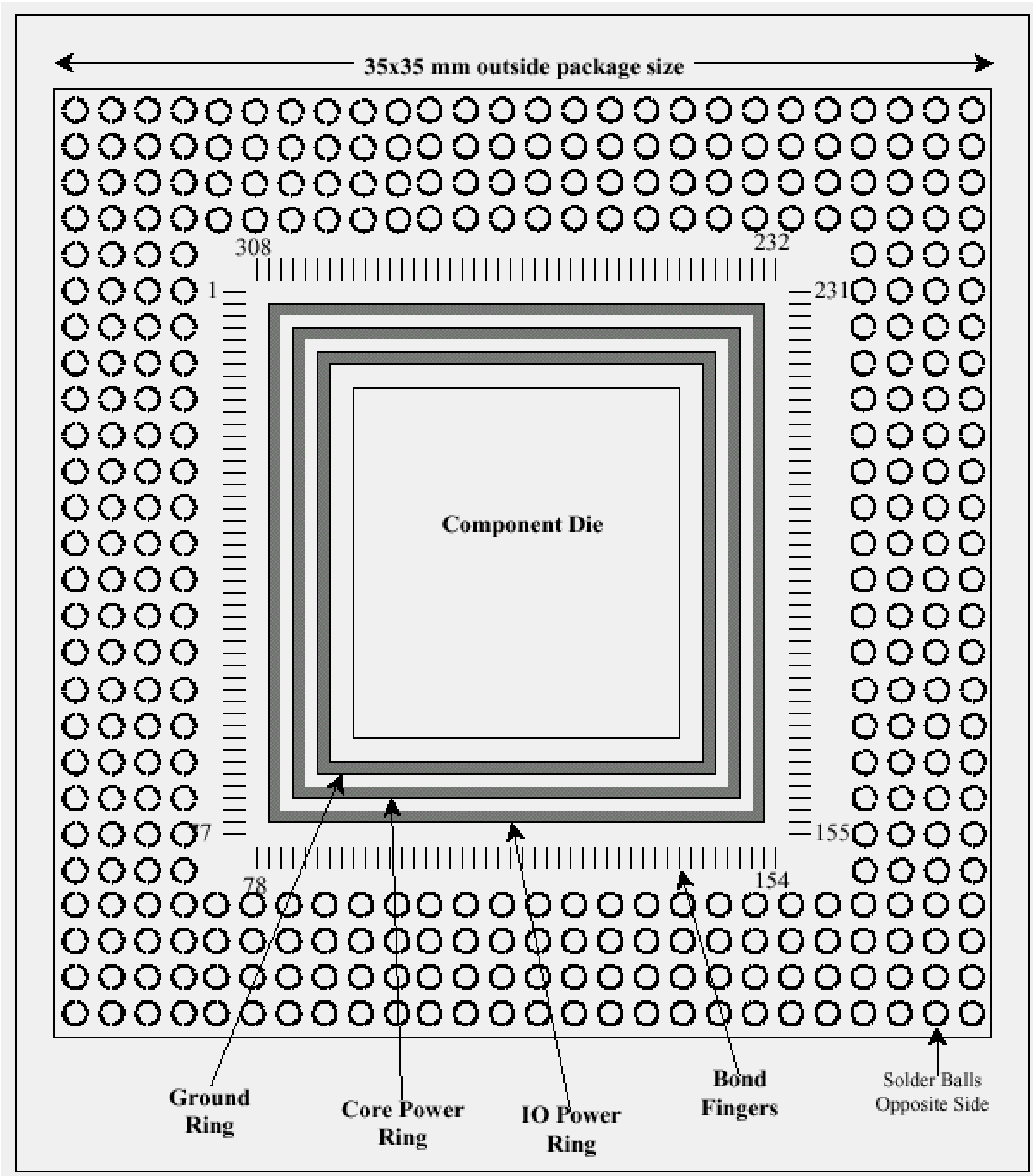


Figure 8-2   388 BGA Internal

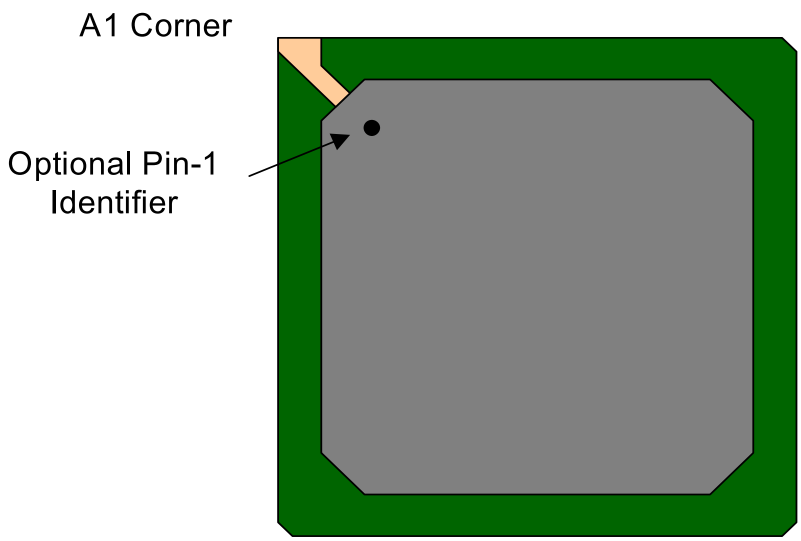


Figure 8-1 ZFx86 Orientation

When you place the ZFx86 on the board, the stripe and Optional Pin 1 Identifier will ensure that you orient the chip properly.

8.2. Pin Descriptions (Sorted by Pin)

See Table 7.5 "I/O Cell Characteristics" on page 475 for a description of Cell Type. Pins whose name ends in _N are active low.

Table 8.2 Pin Descriptions Sorted by Pin

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
A01	I0_CS[2]	ZF-Logic I/O Mapper GPCS 2	Generic2	I0_CS2	ZFLogic
A02	I0_CS[1]	ZF-Logic I/O Mapper GPCS 1	Generic2	I0_CS1	ZFLogic
A03	MEM_CS[2]	ZF-Logic Memory Mapper CS 2	Generic2	MEM_CS2	ZFLogic
A04	WDI	ZF Logic - Watch Dog Timer	Generic2	WDI	ZFLogic
A05	IRTX	UART & IR	Generic2	IR_TX	Super IO
A06	CTS2_N, IRSL3	UART & IR	Generic2	CTS2_N/IR_SL3	Super IO
A07	RX2	UART & IR	Generic2	RXD2	Super IO
A08	RI1_N	UART & IR	Generic2	RI1_N	Super IO
A09	RTS1_N	UART & IR	Generic2	RTS1_N	Super IO
A10	DCD1_N	UART & IR	Generic2	DCD1_N	Super IO
A11	KBDAT	KEYBOARD & MOUSE	Generic2	KBDATA	Super IO
A12	TC	ISA DMA	Generic2	TC	ISA
A13	DACK5_N	ISA DMA	Generic2	DACK5_5	ISA
A14	DACK1_N	ISA DMA (optional PCI Master gnt2_n)	Generic2	DACK1_N	ISA
A15	MA[00]	SDRAM ADDRESS	MMC_D	A0	North Bridge
A16	MA[05]	SDRAM ADDRESS	MMC_D	A5	North Bridge
A17	CPU_TRIG	CPU Trigger	Generic2	CPU_TRIG	
A18	MA[09]	SDRAM ADDRESS	MMC_D	A9	North Bridge
A19	MA[12]	SDRAM ADDRESS	MMC_D	A12	North Bridge
A20	SYSCLK_C	System CLOCK	MMC_SDCLKIN	CLK33MHZ (SYS_CLK)	Processor
A21	SDRAM_CLK[3]_N	SDRAM CLOCK	MMC_SDCLK	CLK3	North Bridge
A22	SDRAM_CLK[1]_N	SDRAM CLOCK	MMC_SDCLK	CLK1	North Bridge
A23	SDRAM_DQM[3]_N	SDRAM Mask / Command	MMC_D	DQM3_N	North Bridge
A24	SDRAM_CS[2]_N	SDRAM Chip Select	MMC_D	CS2_N	North Bridge
A25	SDRAM_CS[1]_N	SDRAM Chip Select	MMC_D	CS1_N	North Bridge
A26	D[01]	SDRAM DATA	MMC_D	D1	North Bridge
AA01	SA[06]	ISA ADDRESS	Generic2	SA6	ISA

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AA02	SA[04]	ISA ADDRESS	Generic2	SA4	ISA
AA03	SA[03]	ISA ADDRESS	Generic2	SA3	ISA
AA04	VDD_CORE			VDD_CORE	
AA23	VDD_CORE			VDD_CORE	
AA24	AD[14]	PCI ADDRESS & DATA	MPCI	AD14	South Bridge
AA25	AD[13]	PCI ADDRESS & DATA	MPCI	AD13	South Bridge
AA26	AD[12]	PCI ADDRESS & DATA	MPCI	AD12	South Bridge
AB01	SA[02]	ISA ADDRESS	Generic2	SA2	ISA
AB02	SA[01]	ISA ADDRESS	Generic2	SA1	ISA
AB03	ISAERR_N	ISA CONTROLS	Generic2	ISA_ERR_N	ISA
AB04	ZWS_N	ISA CONTROLS	Generic2	ZWS_N	ISA
AB23	AD[19]	PCI ADDRESS & DATA	MPCI	AD19	South Bridge
AB24	AD[17]	PCI ADDRESS & DATA	MPCI	AD17	South Bridge
AB25	AD[16]	PCI ADDRESS & DATA	MPCI	AD16	South Bridge
AB26	AD[15]	PCI ADDRESS & DATA	MPCI	AD15	South Bridge
AC01	SA[00]	ISA ADDRESS	Generic2	SA0	ISA
AC02	ISACLK_C	ISA CLOCK	Generic2	ISACLK	ISA
AC03	SBHE_N	ISA CONTROLS	Generic2	SBHE_N	ISA
AC04	GND			GND	
AC05	SD[9]	ISA DATA	Generic2	SD9	ISA
AC06	VDD_CORE			VDD_CORE	
AC07	SD[0]	ISA DATA	Generic2	SD0	ISA
AC08	VDD_IO			VDD_IO	
AC09	MEMW_N	ISA CONTROLS	Generic2	MEMW_N	ISA
AC10	VDD_CORE			VDD_CORE	
AC11	VDD_IO			VDD_IO	
AC12	VDD_CORE			VDD_CORE	
AC13	GND			GND	
AC14	SPARE1	Spare	Generic	SPARE1	
AC15	VDD_IO			VDD_IO	

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AC16	IDE_DATA[12]	IDE DATA	MIDE	IDE_D12	South Bridge
AC17	VDD_CORE			VDD_CORE	
AC18	IDE_DATA[06]	IDE DATA	MIDE	IDE_D6	South Bridge
AC19	VDD_IO			VDD_IO	
AC20	IDE_IOR0_N	IDE CONTROL	MIDE	IDE_IOR0_N	South Bridge
AC21	VDD_CORE			VDD_CORE	
AC22	IDE_DMA_ACK0_N	IDE CONTROL	MIDE	IDE_DACK0_N	South Bridge
AC23	GND			GND	
AC24	AD[21]	PCI ADDRESS & DATA	MPCI	AD21	South Bridge
AC25	AD[20]	PCI ADDRESS & DATA	MPCI	AD20	South Bridge
AC26	AD[18]	PCI ADDRESS & DATA	MPCI	AD18	South Bridge
AD01	IOCHRDY	ISA CONTROLS	Generic2	IOCHRDY	ISA
AD02	BALE	ISA CONTROLS	Generic2	BALE	ISA
AD03	VBAT	Realtime clock battery backup	MVBAT	VBAT	Super IO
AD04	SD[11]	ISA DATA	Generic2	SD11	ISA
AD05	SD[7]	ISA DATA	Generic2	SD7	ISA
AD06	SD[4]	ISA DATA	Generic2	SD4	ISA
AD07	SD[1]	ISA DATA	Generic2	SD1	ISA
AD08	SMEMW_N	ISA CONTROLS	Generic2	SMEMW_N	ISA
AD09	IOR_N	ISA CONTROLS	Generic2	IOR_N	ISA
AD10	GPIO[6], IDE_RDY1	GPIO (optional 2nd IDE diordy)	Generic2	GPIO6/IDE_IORDY1	South Bridge
AD11	GPIO[3], IDE_IOR1_N	GPIO (optional 2nd IDE dior)	Generic2	GPIO3/IDE_DIORD1_N	South Bridge
AD12	GPIO[0], CLK32KHZ_OUT	GPIO (optional 32KHz out)	Generic2	GPIO0/CLK32KHZ_OUT	South Bridge
AD13	OVER_CUR1_N	USB Over Current Sense 1	MUSB	OC_SENS1	South Bridge
AD14	USB_GND	USB circuit ground	MWUSB	GND_USB	South Bridge
AD15	IDE_DATA[15]	IDE DATA	MIDE	IDE_D15	South Bridge
AD16	IDE_DATA[13]	IDE DATA	MIDE	IDE_D13	South Bridge
AD17	IDE_DATA[10]	IDE DATA	MIDE	IDE_D10	South Bridge
AD18	IDE_DATA[05]	IDE DATA	MIDE	IDE_D5	South Bridge

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AD19	IDE_DATA[02]	IDE DATA	MIDE	IDE_D2	South Bridge
AD20	IDE_DATA[00]	IDE DATA	MIDE	IDE_D0	South Bridge
AD21	IDE_ADDR0	IDE CONTROL	MIDE	IDE_ADDR0	South Bridge
AD22	IDE_RDY0	IDE CONTROL	MIDE	IDE_IORDY0_N	South Bridge
AD23	AD[30]	PCI ADDRESS & DATA	MPCI	AD30	South Bridge
AD24	AD[26]	PCI ADDRESS & DATA	MPCI	AD26	South Bridge
AD25	AD[22]	PCI ADDRESS & DATA	MPCI	AD22	South Bridge
AD26	AD[23]	PCI ADDRESS & DATA	MPCI	AD23	South Bridge
AE01	32KHZ_C	Realtime CLOCK	MWUSB	CLK32KHZ	Super IO
AE02	SD[15]	ISA DATA	Generic2	SD15	ISA
AE03	SD[13]	ISA DATA	Generic2	SD13	ISA
AE04	SD[10]	ISA DATA	Generic2	SD10	ISA
AE05	SD[6]	ISA DATA	Generic2	SD6	ISA
AE06	SD[3]	ISA DATA	Generic2	SD3	ISA
AE07	MEMCS16_N	ISA CONTROLS	Generic2	MEMCS16_N	ISA
AE08	SMEMR_N	ISA CONTROLS	Generic2	SMEMR_N	ISA
AE09	IOW_N	ISA CONTROLS	Generic2	IOW_N	ISA
AE10	GPIO[5], IDE_DMA_REQ1_N	GPIO (optional 2nd IDE dreq)	Generic2	GPIO5/ IDE_DREQ1	South Bridge
AE11	GPIO[1], IDE_DMA_ACK1_N	GPIO (optional 2nd IDE dmack)	Generic2	GPIO1/ IDE_DACK1_N	South Bridge
AE12	POWER_EN	USB Power Enable	MUSB	PWR_EN	South Bridge
AE13	PORT1_P	USB Port1 Data Plus	MWUSB	PORT1_P	South Bridge
AE14	PORT2_M	USB Port2 Data Minus	MWUSB	PORT2_M	South Bridge
AE15	USB_48MHZ_C	USB CLOCK	MMC_SDCLKIN	CLK48MHZ (USB_CLK)	South Bridge
AE16	IDE_DATA[14]	IDE DATA	MIDE	IDE_D14	South Bridge
AE17	IDE_DATA[09]	IDE DATA	MIDE	IDE_D9	South Bridge
AE18	IDE_DATA[07]	IDE DATA	MIDE	IDE_D7	South Bridge
AE19	IDE_DATA[03]	IDE DATA	MIDE	IDE_D3	South Bridge
AE20	IDE_CS0_N	IDE CONTROL	MIDE	IDE_CS0_N	South Bridge
AE21	IDE_IOW0_N	IDE CONTROL	MIDE	IDE_IOW0_N	South Bridge

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AE22	IDE_ADDR2	IDE CONTROL	MIDE	IDE_ADDR2	South Bridge
AE23	IDE_DMA_REQ0_N	IDE CONTROL	MIDE	IDE_DREQ0_N	South Bridge
AE24	AD[29]	PCI ADDRESS & DATA	MPCI	AD29	South Bridge
AE25	AD[25]	PCI ADDRESS & DATA	MPCI	AD25	South Bridge
AE26	AD[24]	PCI ADDRESS & DATA	MPCI	AD24	South Bridge
AF01	32KHZC_C	Realtime CLOCK	MWUSB	CLK32KHZC (CLK IN)	Super IO
AF02	SD[14]	ISA DATA	Generic2	SD14	ISA
AF03	SD[12]	ISA DATA	Generic2	SD12	ISA
AF04	SD[8]	ISA DATA	Generic2	SD8	ISA
AF05	SD[5]	ISA DATA	Generic2	SD5	ISA
AF06	SD[2]	ISA DATA	Generic2	SD2	ISA
AF07	IOCS16_N	ISA CONTROLS	Generic2	IOCS16_N	ISA
AF08	MEMR_N	ISA CONTROLS	Generic2	MEMR_N	ISA
AF09	GPIO[7]	GPIO	Generic2	GPIO7	South Bridge
AF10	GPIO[4]	GPIO	Generic2	GPIO4	South Bridge
AF11	GPIO[2], IDE_IOW1_N	GPIO (optional 2nd IDE diow)	Generic2	GPIO2/IDE_IOW1_N	South Bridge
AF12	OVER_CUR2_N	USB Over Current Sense 2	MUSB	OC_SENS2	South Bridge
AF13	PORT1_M	USB Port1 Data Minus	MWUSB	PORT1_M	South Bridge
AF14	PORT2_DP	USB Port2 Data Plus	MWUSB	PORT2_P	South Bridge
AF15	USB_PWR	USB circuit power	MWUSB	VDD_USB	South Bridge
AF16	mhz14_c	14 MHz Clock input	Generic2	CLK14MHZ (TIMER_CLK)	
AF17	IDE_DATA[11]	IDE DATA	MIDE	IDE_D11	South Bridge
AF18	IDE_DATA[08]	IDE DATA	MIDE	IDE_D8	South Bridge
AF19	IDE_DATA[04]	IDE DATA	MIDE	IDE_D4	South Bridge
AF20	IDE_DATA[01]	IDE DATA	MIDE	IDE_D1	South Bridge
AF21	IDE_CS1_N	IDE CONTROL	MIDE	IDE_CS1_N	South Bridge
AF22	IDE_ADDR1	IDE CONTROL	MIDE	IDE_ADDR1	South Bridge
AF23	IDE_RST_N	IDE CONTROL	MIDE	IDE_RST_N	South Bridge
AF24	AD[31]	PCI ADDRESS & DATA	MPCI	AD31	South Bridge
AF25	AD[28]	PCI ADDRESS & DATA	MPCI	AD28	South Bridge

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AF26	AD[27]	PCI ADDRESS & DATA	MPCI	AD27	South Bridge
B01	BEEP_N	PC Speaker	Generic2	BEEP_N	Super IO
B02	IRQ3	INTERRUPT	Generic2	IRQ3	ISA
B03	I0_CS[0]	ZF-Logic I/O Mapper GPCS 0	Generic2	IO_CS0	ZFLogic
B04	MEM_CS[0]	ZF-Logic Memory Mapper CS 0	Generic2	MEM_CS0	ZFLogic
B05	PWM	ZF-Logic PWM Output	Generic2	PWM	ZFLogic
B06	RI2_N	UART & IR	Generic2	RI2_N	Super IO
B07	TX2	UART & IR	Generic2	TXD2	Super IO
B08	DCD2_N, IRSL2	UART & IR	Generic2	DCD2_N/IR_SL2	Super IO
B09	TX1	UART & IR	Generic2	TXD1	Super IO
B10	RX1	UART & IR	Generic2	RXD1	Super IO
B11	KBLOCK	KEYBOARD & MOUSE	Generic2	KBLOCK_N	Super IO
B12	SCL_C	ACCESS BUS	MAC97	SCL_C	Super IO
B13	DRQ5	ISA DMA	Generic2	DRQ5	ISA
B14	DRQ1	ISA DMA (Optional PCI Master req2_n)	Generic2	DRQ1	ISA
B15	MA[02]	SDRAM ADDRESS	MMC_D	A2	North Bridge
B16	MA[04]	SDRAM ADDRESS	MMC_D	A4	North Bridge
B17	MA[07]	SDRAM ADDRESS	MMC_D	A7	North Bridge
B18	MA[11]	SDRAM ADDRESS	MMC_D	A11	North Bridge
B19	PORDIS	Power On Reset Disable	MVBAT	POR_DIS	South Bridge
B20	MR	ISA Reset Drive	Generic2	RESETDRV	ISA
B21	SDRAM_CLK[2]_N	SDRAM CLOCK	MMC_SDCLK	CLK2	North Bridge
B22	SDRAM_CLK[0]_N	SDRAM CLOCK	MMC_SDCLK	CLK0	North Bridge
B23	SDRAM_DQM[1]_N	SDRAM Mask / Command	MMC_D	DQM1_N	North Bridge
B24	SDRAM_CS[3]_N	SDRAM Chip Select	MMC_D	CS3_N	North Bridge
B25	SDRAM_CS[0]_N	SDRAM Chip Select	MMC_D	CS0_N	North Bridge
B26	D[02]	SDRAM DATA	MMC_D	D2	North Bridge
C01	IRQ4	INTERRUPT	Generic2	IRQ4	South Bridge
C02	IRQ5	INTERRUPT	Generic2	IRQ5	ISA
C03	I0_CS[3]	ZF-Logic I/O Mapper GPCS 3	Generic2	IO_CS3	ZFLogic

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
C04	MEM_CS[3]	ZF-Logic Memory Mapper CS 3	Generic2	MEM_CS3	ZFLogic
C05	WDO	ZF Logic - Watch Dog Timer	Generic2	WDO	ZFLogic
C06	IRRX	UART & IR	Generic2	IR_RX	Super IO
C07	RTS2_N, IRSL0	UART & IR	Generic2	RTS2_N/IR_SL0	Super IO
C08	DSR2_N, IRSL1	UART & IR	Generic2	DSR2_N/IR_SL1	Super IO
C09	DTR1_N	UART & IR	Generic2	DTR1_N	Super IO
C10	DSR1_N	UART & IR	Generic2	DSR1_N	Super IO
C11	MDAT	KEYBOARD & MOUSE	Generic2	MDATA	Super IO
C12	KBCLK_C	KEYBOARD & MOUSE	Generic2	KBCLK	Super IO
C13	AEN	ISA DMA	Generic2	AEN	ISA
C14	MA[01]	SDRAM ADDRESS	MMC_D	A1	North Bridge
C15	MA[03]	SDRAM ADDRESS	MMC_D	A3	North Bridge
C16	MA[06]	SDRAM ADDRESS	MMC_D	A6	North Bridge
C17	MA[08]	SDRAM ADDRESS	MMC_D	A8	North Bridge
C18	MA[10]	SDRAM ADDRESS	MMC_D	A10	North Bridge
C19	POR_N	System Reset	Generic2	RESET_N	
C20	SDRAM_RAS_N	SDRAM RAS	MMC_D	RAS_N	North Bridge
C21	SDRAM_CLKE	SDRAM Clock Enable	MMC_D	CLKE	North Bridge
C22	SDRAM_WE_N	SDRAM Write Enable	MMC_D	WE_N	North Bridge
C23	SDRAM_DQM[0]_N	SDRAM Mask / Command	MMC_D	DQM0_N	North Bridge
C24	D[00]	SDRAM DATA	MMC_D	D0	North Bridge
C25	D[03]	SDRAM DATA	MMC_D	D3	North Bridge
C26	D[05]	SDRAM DATA	MMC_D	D5	North Bridge
D01	IRQ11, PCI_INT_C	PCI INTERRUPT C	MPCI	IRQ11/PCI_INT_C	South Bridge
D02	IRQ9, PCI_INT_A	PCI INTERRUPT A	MPCI	IRQ9/PCI_INT_A	South Bridge
D03	IRQ7	INTERRUPT	Generic2	IRQ7	ISA
D04	GND			GND	
D05	MEM_CS[1]	ZF-Logic Memory Mapper CS 1	Generic2	MEM_CS1	ZFLogic
D06	VDD_CORE			VDD_CORE	
D07	DTR2_N	UART & IR	Generic2	DTR2_N	Super IO

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
D08	VDD_IO			VDD_IO	
D09	CTS1_N	UART & IR	Generic2	CTS1_N	Super IO
D10	VDD_CORE			VDD_CORE	
D11	MCLK_C	KEYBOARD & MOUSE	Generic2	MCLK	South Bridge
D12	VDD_IO			VDD_IO	
D13	SDA	ACCESS BUS	MAC97	SDA	Super IO
D14	GND			GND	
D15	VDD_CORE			VDD_CORE	
D16	VDD_IO			VDD_IO	
D17	VDD_CORE			VDD_CORE	
D18	MA[13]	SDRAM ADDRESS	MMC_D	A13	North Bridge
D19	VDD_IO			VDD_IO	
D20	SDRAM_CAS_N	SDRAM CAS	MMC_D	CAS_N	North Bridge
D21	VDD_CORE			VDD_CORE	
D22	SDRAM_DQM[2]_N	SDRAM Mask / Command	MMC_D	DQM2_N	North Bridge
D23	GND			GND	
D24	D[04]	SDRAM DATA	MMC_D	D4	North Bridge
D25	D[07]	SDRAM DATA	MMC_D	D7	North Bridge
D26	D[09]	SDRAM DATA	MMC_D	D9	North Bridge
E01	IRQ15	INTERRUPT	Generic2	IRQ15 (IDES_IRQ)	South Bridge
E02	IRQ14	INTERRUPT	Generic2	IRQ14 (IDEP_IRQ)	ISA
E03	IRQ12, PCI_INT_D	PCI INTERRUPT D	MPCI	IRQ12/PCI_INT_D	South Bridge
E04	IRQ10, PCI_INT_B	PCI INTERRUPT B	MPCI	IRQ10/PCI_INT_B	South Bridge
E23	D[06]	SDRAM DATA	MMC_D	D6	North Bridge
E24	D[08]	SDRAM DATA	MMC_D	D8	North Bridge
E25	D[10]	SDRAM DATA	MMC_D	D10	North Bridge
E26	D[11]	SDRAM DATA	MMC_D	D11	North Bridge
F01	DR0_N	FLOPPY	m_fdc_p	DR0_N	Super IO
F02	MTR0_N	FLOPPY	m_fdc_p	MTR0_N	Super IO
F03	INDEX_N	FLOPPY	m_fdc_p	INDEX_N	Super IO

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
F04	VDD_CORE			VDD_CORE	
F23	VDD_CORE			VDD_CORE	
F24	D[12]	SDRAM DATA	MMC_D	D12	North Bridge
F25	D[13]	SDRAM DATA	MMC_D	D13	North Bridge
F26	D[15]	SDRAM DATA	MMC_D	D15	North Bridge
G01	WGATE_N	FLOPPY	m_fdc_p	WGATE_N	Super IO
G02	WDATA_N	FLOPPY	m_fdc_p	WDATA_N/ZCLK	Super IO
G03	DIR _N	FLOPPY	m_fdc_p	DIR_N/ZRST	Super IO
G04	STEP_N	FLOPPY	m_fdc_p	STEP_N/ZLED2	Super IO
G23	D[14]	SDRAM DATA	MMC_D	D14	North Bridge
G24	D[17]	SDRAM DATA	MMC_D	D17	North Bridge
G25	D[16]	SDRAM DATA	MMC_D	D16	North Bridge
G26	D[18]	SDRAM DATA	MMC_D	D18	North Bridge
H01	HDSEL_N	FLOPPY	m_fdc_p	HDSEL_N/ZLED1	Super IO
H02	WRPRT_N	FLOPPY	m_fdc_p	WRPRT_N/ZACK	Super IO
H03	TRK0_N	FLOPPY	m_fdc_p	TRK0_N/ZGPI0	Super IO
H04	VDD_IO			VDD_IO	
H23	VDD_IO			VDD_IO	
H24	D[19]	SDRAM DATA	MMC_D	D19	North Bridge
H25	D[20]	SDRAM DATA	MMC_D	D20	North Bridge
H26	D[21]	SDRAM DATA	MMC_D	D21	North Bridge
J01	PE	PARALLEL PORT	m_fdc_p	PE	Super IO
J02	DSKCHG_N	FLOPPY	m_fdc_p	DSKCHG_N/ZGPI1	Super IO
J03	SLCT	PARALLEL PORT	m_fdc_p	SLCT	Super IO
J04	RDATA_N	FLOPPY	m_fdc_p	RDATA_N/ZDIN	Super IO
J23	D[23]	SDRAM DATA	MMC_D	D23	North Bridge
J24	D[22]	SDRAM DATA	MMC_D	D22	North Bridge
J25	D[24]	SDRAM DATA	MMC_D	D24	North Bridge
J26	D[25]	SDRAM DATA	MMC_D	D25	North Bridge
K01	SLIN_N	PARALLEL PORT	m_fdc_p	SLIN_N	Super IO

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
K02	ACK_N	PARALLEL PORT	m_fdc_p	ACK_N	Super IO
K03	BUSY	PARALLEL PORT	m_fdc_p	BUSY	Super IO
K04	VDD_CORE			VDD_CORE	
K23	VDD_CORE			VDD_CORE	
K24	D[27]	SDRAM DATA	MMC_D	D27	North Bridge
K25	D[26]	SDRAM DATA	MMC_D	D26	North Bridge
K26	D[28]	SDRAM DATA	MMC_D	D28	North Bridge
L01	ERR_N	PARALLEL PORT	m_fdc_p	ERR_N	Super IO
L02	AFD_N	PARALLEL PORT	m_fdc_p	AFD_N	Super IO
L03	INIT_N	PARALLEL PORT	m_fdc_p	INIT_N	Super IO
L04	VDD_IO			VDD_IO	
L11	GND			GND	
L12	GND			GND	
L13	GND			GND	
L14	GND			GND	
L15	GND			GND	
L16	GND			GND	
L23	D[29]	SDRAM DATA	MMC_D	D29	North Bridge
L24	D[30]	SDRAM DATA	MMC_D	D30	North Bridge
L25	D[31]	SDRAM DATA	MMC_D	D31	North Bridge
L26	GNT0_N	PCI CONTROL	MPCI	GNT0_N	South Bridge
M01	PD[5]	PARALLEL PORT	m_fdc_p	PD5	Super IO
M02	PD[7]	PARALLEL PORT	m_fdc_p	PD7	Super IO
M03	STB_N	PARALLEL PORT	m_fdc_p	STB_N	Super IO
M04	VDD_CORE			VDD_CORE	
M11	GND			GND	
M12	GND			GND	
M13	GND			GND	
M14	GND			GND	
M15	GND			GND	

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
M16	GND			GND	
M23	VDD_IO			VDD_IO	
M24	GNT1_N	PCI CONTROL	MPCI	GNT1_N	South Bridge
M25	REQ0_N	PCI CONTROL	MPCI	REQ0_N	South Bridge
M26	SERR_N	PCI CONTROL	MPCI	SERR_N	South Bridge
N01	PD[3]	PARALLEL PORT	m_fdc_p	PD3	Super IO
N02	PD[4]	PARALLEL PORT	m_fdc_p	PD4	Super IO
N03	PD[6]	PARALLEL PORT	m_fdc_p	PD6	Super IO
N04	GND			GND	
N11	GND			GND	
N12	GND			GND	
N13	GND			GND	
N14	GND			GND	
N15	GND			GND	
N16	GND			GND	
N23	REQ1_N	PCI CONTROL	MPCI	REQ1_N	South Bridge
N24	PERR_N	PCI CONTROL	MPCI	PERR_N	South Bridge
N25	LOCK_N	PCI CONTROL	MPCI	PLOCK_N	South Bridge
N26	PAR	PCI CONTROL	MPCI	PAR	South Bridge
P01	PD[1]	PARALLEL PORT	m_fdc_p	PD1	Super IO
P02	PD[2]	PARALLEL PORT	m_fdc_p	PD2	Super IO
P03	PD[0]	PARALLEL PORT	m_fdc_p	PD0	Super IO
P04	TDI	JTAG (system)	Generic2	TDI	South Bridge
P11	GND			GND	
P12	GND			GND	
P13	GND			GND	
P14	GND			GND	
P15	GND			GND	
P16	GND			GND	
P23	GND			GND	

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
P24	FRAME_N	PCI CONTROL	MPCI	FRAME_N	South Bridge
P25	IRDY_N	PCI CONTROL	MPCI	IRDY_N	South Bridge
P26	STOP_N	PCI CONTROL	MPCI	STOP_N	South Bridge
R01	TCK_C	JTAG (system)	Generic2	TCK	South Bridge
R02	TMS	JTAG (system)	Generic2	TMS	South Bridge
R03	SA[23]	ISA ADDRESS	Generic2	SA23	ISA
R04	VDD_IO			VDD_IO	
R11	GND			GND	
R12	GND			GND	
R13	GND			GND	
R14	GND			GND	
R15	GND			GND	
R16	GND			GND	
R23	VDD_CORE			VDD_CORE	
R24	C/BE[3]_N	PCI COMMAND / BYTE	MPCI	C_BE3_N	South Bridge
R25	DEVSEL_N	PCI CONTROL	MPCI	DEVSEL_N	South Bridge
R26	TRDY_N	PCI CONTROL	MPCI	TRDY_N	South Bridge
T01	TDO	JTAG (system)	Generic2	TDO	South Bridge
T02	SA[22]	ISA ADDRESS	Generic2	SA22	ISA
T03	SA[21]	ISA ADDRESS	Generic2	SA21	ISA
T04	SA[20]	ISA ADDRESS	Generic2	SA20	ISA
T11	GND			GND	
T12	GND			GND	
T13	GND			GND	
T14	GND			GND	
T15	GND			GND	
T16	GND			GND	
T23	VDD_IO			VDD_IO	
T24	C/BE[0]_N	PCI COMMAND / BYTE	MPCI	C_BE0_N	South Bridge
T25	C/BE[2]_N	PCI COMMAND / BYTE	MPCI	C_BE2_N	South Bridge

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
T26	C/BE[1]_N	PCI COMMAND / BYTE	MPCI	C_BE1_N	South Bridge
U01	SA[19]	ISA ADDRESS	Generic2	SA19	ISA
U02	SA[17]	ISA ADDRESS	Generic2	SA17	ISA
U03	SA[18]	ISA ADDRESS	Generic2	SA18	ISA
U04	VDD_CORE			VDD_CORE	
U23	VDD_CORE			VDD_CORE	
U24	AD[00]	PCI ADDRESS & DATA	MPCI	AD0	South Bridge
U25	PCICLK_C	PCI CLOCK	MPCI_CLK	CLK	South Bridge
U26	PCI_RST_N	PCI RESET	MPCI	PRST_N	South Bridge
V01	SA[16]	ISA ADDRESS	Generic2	SA16	ISA
V02	SA[15]	ISA ADDRESS	Generic2	SA15	ISA
V03	SA[13]	ISA ADDRESS	Generic2	SA13	ISA
V04	SA[14]	ISA ADDRESS	Generic2	SA14	ISA
V23	AD[05]	PCI ADDRESS & DATA	MPCI	AD5	South Bridge
V24	AD[02]	PCI ADDRESS & DATA	MPCI	AD2	South Bridge
V25	AD[03]	PCI ADDRESS & DATA	MPCI	AD3	South Bridge
V26	AD[01]	PCI ADDRESS & DATA	MPCI	AD1	South Bridge
W01	SA[12]	ISA ADDRESS	Generic2	SA12	ISA
W02	SA[11]	ISA ADDRESS	Generic2	SA11	ISA
W03	SA[10]	ISA ADDRESS	Generic2	SA10	ISA
W04	VDD_IO			VDD_IO	
W23	VDD_IO			VDD_IO	
W24	AD[07]	PCI ADDRESS & DATA	MPCI	AD7	South Bridge
W25	AD[06]	PCI ADDRESS & DATA	MPCI	AD6	South Bridge
W26	AD[04]	PCI ADDRESS & DATA	MPCI	AD4	South Bridge
Y01	SA[09]	ISA ADDRESS	Generic2	SA9	ISA
Y02	SA[07]	ISA ADDRESS	Generic2	SA7	ISA
Y03	SA[08]	ISA ADDRESS	Generic2	SA8	ISA
Y04	SA[05]	ISA ADDRESS	Generic2	SA5	ISA
Y23	AD[10]	PCI ADDRESS & DATA	MPCI	AD10	South Bridge

Table 8.2 Pin Descriptions Sorted by Pin (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
Y24	AD[11]	PCI ADDRESS & DATA	MPCI	AD11	South Bridge
Y25	AD[09]	PCI ADDRESS & DATA	MPCI	AD9	South Bridge
Y26	AD[08]	PCI ADDRESS & DATA	MPCI	AD8	South Bridge

8.3. Pin Descriptions (Sorted by Pin Name)

See Table 7.5 "I/O Cell Characteristics" on page 475 for a description of Cell Type. Pins whose name ends in _N are active low.

Table 8.3 Pin Descriptions Sorted by Pin Name

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
K02	ACK_N	PARALLEL PORT	m_fdc_p	ACK_N	Super IO
U24	AD[00]	PCI ADDRESS & DATA	MPCI	AD0	South Bridge
V26	AD[01]	PCI ADDRESS & DATA	MPCI	AD1	South Bridge
V24	AD[02]	PCI ADDRESS & DATA	MPCI	AD2	South Bridge
V25	AD[03]	PCI ADDRESS & DATA	MPCI	AD3	South Bridge
W26	AD[04]	PCI ADDRESS & DATA	MPCI	AD4	South Bridge
V23	AD[05]	PCI ADDRESS & DATA	MPCI	AD5	South Bridge
W25	AD[06]	PCI ADDRESS & DATA	MPCI	AD6	South Bridge
W24	AD[07]	PCI ADDRESS & DATA	MPCI	AD7	South Bridge
Y26	AD[08]	PCI ADDRESS & DATA	MPCI	AD8	South Bridge
Y25	AD[09]	PCI ADDRESS & DATA	MPCI	AD9	South Bridge
Y23	AD[10]	PCI ADDRESS & DATA	MPCI	AD10	South Bridge
Y24	AD[11]	PCI ADDRESS & DATA	MPCI	AD11	South Bridge
AA26	AD[12]	PCI ADDRESS & DATA	MPCI	AD12	South Bridge
AA25	AD[13]	PCI ADDRESS & DATA	MPCI	AD13	South Bridge
AA24	AD[14]	PCI ADDRESS & DATA	MPCI	AD14	South Bridge
AB26	AD[15]	PCI ADDRESS & DATA	MPCI	AD15	South Bridge
AB25	AD[16]	PCI ADDRESS & DATA	MPCI	AD16	South Bridge
AB24	AD[17]	PCI ADDRESS & DATA	MPCI	AD17	South Bridge

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AC26	AD[18]	PCI ADDRESS & DATA	MPCI	AD18	South Bridge
AB23	AD[19]	PCI ADDRESS & DATA	MPCI	AD19	South Bridge
AC25	AD[20]	PCI ADDRESS & DATA	MPCI	AD20	South Bridge
AC24	AD[21]	PCI ADDRESS & DATA	MPCI	AD21	South Bridge
AD25	AD[22]	PCI ADDRESS & DATA	MPCI	AD22	South Bridge
AD26	AD[23]	PCI ADDRESS & DATA	MPCI	AD23	South Bridge
AE26	AD[24]	PCI ADDRESS & DATA	MPCI	AD24	South Bridge
AE25	AD[25]	PCI ADDRESS & DATA	MPCI	AD25	South Bridge
AD24	AD[26]	PCI ADDRESS & DATA	MPCI	AD26	South Bridge
AF26	AD[27]	PCI ADDRESS & DATA	MPCI	AD27	South Bridge
AF25	AD[28]	PCI ADDRESS & DATA	MPCI	AD28	South Bridge
AE24	AD[29]	PCI ADDRESS & DATA	MPCI	AD29	South Bridge
AD23	AD[30]	PCI ADDRESS & DATA	MPCI	AD30	South Bridge
AF24	AD[31]	PCI ADDRESS & DATA	MPCI	AD31	South Bridge
C13	AEN	ISA DMA	Generic2	AEN	ISA
L02	AFD_N	PARALLEL PORT	m_fdc_p	AFD_N	Super IO
AD02	BALE	ISA CONTROLS	Generic2	BALE	ISA
B01	BEEP_N	PC Speaker	Generic2	BEEP_N	Super IO
K03	BUSY	PARALLEL PORT	m_fdc_p	BUSY	Super IO
T24	C/BE[0]_N	PCI COMMAND / BYTE	MPCI	C_BE0_N	South Bridge
T26	C/BE[1]_N	PCI COMMAND / BYTE	MPCI	C_BE1_N	South Bridge
T25	C/BE[2]_N	PCI COMMAND / BYTE	MPCI	C_BE2_N	South Bridge
R24	C/BE[3]_N	PCI COMMAND / BYTE	MPCI	C_BE3_N	South Bridge
A17	CPU_TRIG	CPU Trigger	Generic2	CPU_TRIG	
D09	CTS1_N	UART & IR	Generic2	CTS1_N	Super IO
A06	CTS2_N, IRSL3	UART & IR	Generic2	CTS2_N/IR_SL3	Super IO
C24	D[00]	SDRAM DATA	MMC_D	D0	North Bridge
A26	D[01]	SDRAM DATA	MMC_D	D1	North Bridge
B26	D[02]	SDRAM DATA	MMC_D	D2	North Bridge
C25	D[03]	SDRAM DATA	MMC_D	D3	North Bridge

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
D24	D[04]	SDRAM DATA	MMC_D	D4	North Bridge
C26	D[05]	SDRAM DATA	MMC_D	D5	North Bridge
E23	D[06]	SDRAM DATA	MMC_D	D6	North Bridge
D25	D[07]	SDRAM DATA	MMC_D	D7	North Bridge
E24	D[08]	SDRAM DATA	MMC_D	D8	North Bridge
D26	D[09]	SDRAM DATA	MMC_D	D9	North Bridge
E25	D[10]	SDRAM DATA	MMC_D	D10	North Bridge
E26	D[11]	SDRAM DATA	MMC_D	D11	North Bridge
F24	D[12]	SDRAM DATA	MMC_D	D12	North Bridge
F25	D[13]	SDRAM DATA	MMC_D	D13	North Bridge
G23	D[14]	SDRAM DATA	MMC_D	D14	North Bridge
F26	D[15]	SDRAM DATA	MMC_D	D15	North Bridge
G25	D[16]	SDRAM DATA	MMC_D	D16	North Bridge
G24	D[17]	SDRAM DATA	MMC_D	D17	North Bridge
G26	D[18]	SDRAM DATA	MMC_D	D18	North Bridge
H24	D[19]	SDRAM DATA	MMC_D	D19	North Bridge
H25	D[20]	SDRAM DATA	MMC_D	D20	North Bridge
H26	D[21]	SDRAM DATA	MMC_D	D21	North Bridge
J24	D[22]	SDRAM DATA	MMC_D	D22	North Bridge
J23	D[23]	SDRAM DATA	MMC_D	D23	North Bridge
J25	D[24]	SDRAM DATA	MMC_D	D24	North Bridge
J26	D[25]	SDRAM DATA	MMC_D	D25	North Bridge
K25	D[26]	SDRAM DATA	MMC_D	D26	North Bridge
K24	D[27]	SDRAM DATA	MMC_D	D27	North Bridge
K26	D[28]	SDRAM DATA	MMC_D	D28	North Bridge
L23	D[29]	SDRAM DATA	MMC_D	D29	North Bridge
L24	D[30]	SDRAM DATA	MMC_D	D30	North Bridge
L25	D[31]	SDRAM DATA	MMC_D	D31	North Bridge
A14	DACK1_N	ISA DMA (optional PCI Master gnt2_n)	Generic2	DACK1_N	ISA
A13	DACK5_N	ISA DMA	Generic2	DACK5_5	ISA

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
A10	DCD1_N	UART & IR	Generic2	DCD1_N	Super IO
B08	DCD2_N, IRSL2	UART & IR	Generic2	DCD2_N/IR_SL2	Super IO
R25	DEVSEL_N	PCI CONTROL	MPCI	DEVSEL_N	South Bridge
G03	DIR _N	FLOPPY	m_fdc_p	DIR_N/ZRST	Super IO
F01	DR0_N	FLOPPY	m_fdc_p	DR0_N	Super IO
B14	DRQ1	ISA DMA (Optional PCI Master req2_n)	Generic2	DRQ1	ISA
B13	DRQ5	ISA DMA	Generic2	DRQ5	ISA
J02	DSKCHG_N	FLOPPY	m_fdc_p	DSKCHG_N/ZGPI1	Super IO
C10	DSR1_N	UART & IR	Generic2	DSR1_N	Super IO
C08	DSR2_N, IRSL1	UART & IR	Generic2	DSR2_N/IR_SL1	Super IO
C09	DTR1_N	UART & IR	Generic2	DTR1_N	Super IO
D07	DTR2_N	UART & IR	Generic2	DTR2_N	Super IO
L01	ERR_N	PARALLEL PORT	m_fdc_p	ERR_N	Super IO
P24	FRAME_N	PCI CONTROL	MPCI	FRAME_N	South Bridge
AC04	GND			GND	
AC13	GND			GND	
AC23	GND			GND	
D04	GND			GND	
D14	GND			GND	
D23	GND			GND	
L11	GND			GND	
L12	GND			GND	
L13	GND			GND	
L14	GND			GND	
L15	GND			GND	
L16	GND			GND	
M11	GND			GND	
M12	GND			GND	
M13	GND			GND	
M14	GND			GND	

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
M15	GND			GND	
M16	GND			GND	
N04	GND			GND	
N11	GND			GND	
N12	GND			GND	
N13	GND			GND	
N14	GND			GND	
N15	GND			GND	
N16	GND			GND	
P11	GND			GND	
P12	GND			GND	
P13	GND			GND	
P14	GND			GND	
P15	GND			GND	
P16	GND			GND	
P23	GND			GND	
R11	GND			GND	
R12	GND			GND	
R13	GND			GND	
R14	GND			GND	
R15	GND			GND	
R16	GND			GND	
T11	GND			GND	
T12	GND			GND	
T13	GND			GND	
T14	GND			GND	
T15	GND			GND	
T16	GND			GND	
L26	GNT0_N	PCI CONTROL	MPCI	GNT0_N	South Bridge
M24	GNT1_N	PCI CONTROL	MPCI	GNT1_N	South Bridge

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AD12	GPIO[0], CLK32KHZ_OUT	GPIO (optional 32KHz out)	Generic2	GPIO0/CLK32KHZ_OUT	South Bridge
AE11	GPIO[1], IDE_DMA_ACK1_N	GPIO (optional 2nd IDE dmack)	Generic2	GPIO1/ IDE_DACK1_N	South Bridge
AF11	GPIO[2], IDE_IOW1_N	GPIO (optional 2nd IDE diow)	Generic2	GPIO2/IDE_IOW1_N	South Bridge
AD11	GPIO[3], IDE_IOR1_N	GPIO (optional 2nd IDE dior)	Generic2	GPIO3/IDE_DIORD1_N	South Bridge
AF10	GPIO[4]	GPIO	Generic2	GPIO4	South Bridge
AE10	GPIO[5], IDE_DMA_REQ1_N	GPIO (optional 2nd IDE dreq)	Generic2	GPIO5/ IDE_DREQ1	South Bridge
AD10	GPIO[6], IDE_RDY1	GPIO (optional 2nd IDE diordy)	Generic2	GPIO6/IDE_IORDY1	South Bridge
AF09	GPIO[7]	GPIO	Generic2	GPIO7	South Bridge
H01	HDSEL_N	FLOPPY	m_fdc_p	HDSEL_N/ZLED1	Super IO
B03	IO_CS[0]	ZF-Logic I/O Mapper GPCS 0	Generic2	IO_CS0	ZFLogic
A02	IO_CS[1]	ZF-Logic I/O Mapper GPCS 1	Generic2	I0_CS1	ZFLogic
A01	IO_CS[2]	ZF-Logic I/O Mapper GPCS 2	Generic2	I0_CS2	ZFLogic
C03	IO_CS[3]	ZF-Logic I/O Mapper GPCS 3	Generic2	IO_CS3	ZFLogic
AD21	IDE_ADDR0	IDE CONTROL	MIDE	IDE_ADDR0	South Bridge
AF22	IDE_ADDR1	IDE CONTROL	MIDE	IDE_ADDR1	South Bridge
AE22	IDE_ADDR2	IDE CONTROL	MIDE	IDE_ADDR2	South Bridge
AE20	IDE_CS0_N	IDE CONTROL	MIDE	IDE_CS0_N	South Bridge
AF21	IDE_CS1_N	IDE CONTROL	MIDE	IDE_CS1_N	South Bridge
AD20	IDE_DATA[00]	IDE DATA	MIDE	IDE_D0	South Bridge
AF20	IDE_DATA[01]	IDE DATA	MIDE	IDE_D1	South Bridge
AD19	IDE_DATA[02]	IDE DATA	MIDE	IDE_D2	South Bridge
AE19	IDE_DATA[03]	IDE DATA	MIDE	IDE_D3	South Bridge
AF19	IDE_DATA[04]	IDE DATA	MIDE	IDE_D4	South Bridge
AD18	IDE_DATA[05]	IDE DATA	MIDE	IDE_D5	South Bridge
AC18	IDE_DATA[06]	IDE DATA	MIDE	IDE_D6	South Bridge
AE18	IDE_DATA[07]	IDE DATA	MIDE	IDE_D7	South Bridge
AF18	IDE_DATA[08]	IDE DATA	MIDE	IDE_D8	South Bridge
AE17	IDE_DATA[09]	IDE DATA	MIDE	IDE_D9	South Bridge

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AD17	IDE_DATA[10]	IDE DATA	MIDE	IDE_D10	South Bridge
AF17	IDE_DATA[11]	IDE DATA	MIDE	IDE_D11	South Bridge
AC16	IDE_DATA[12]	IDE DATA	MIDE	IDE_D12	South Bridge
AD16	IDE_DATA[13]	IDE DATA	MIDE	IDE_D13	South Bridge
AE16	IDE_DATA[14]	IDE DATA	MIDE	IDE_D14	South Bridge
AD15	IDE_DATA[15]	IDE DATA	MIDE	IDE_D15	South Bridge
AC22	IDE_DMA_ACK0_N	IDE CONTROL	MIDE	IDE_DACK0_N	South Bridge
AE23	IDE_DMA_REQ0_N	IDE CONTROL	MIDE	IDE_DREQ0_N	South Bridge
AC20	IDE_IOR0_N	IDE CONTROL	MIDE	IDE_IOR0_N	South Bridge
AE21	IDE_IOW0_N	IDE CONTROL	MIDE	IDE_IOW0_N	South Bridge
AD22	IDE_RDY0	IDE CONTROL	MIDE	IDE_IORDY0_N	South Bridge
AF23	IDE_RST_N	IDE CONTROL	MIDE	IDE_RST_N	South Bridge
F03	INDEX_N	FLOPPY	m_fdc_p	INDEX_N	Super IO
L03	INIT_N	PARALLEL PORT	m_fdc_p	INIT_N	Super IO
AD01	IOCHRDY	ISA CONTROLS	Generic2	IOCHRDY	ISA
AF07	IOCS16_N	ISA CONTROLS	Generic2	IOCS16_N	ISA
AD09	IOR_N	ISA CONTROLS	Generic2	IOR_N	ISA
AE09	IOW_N	ISA CONTROLS	Generic2	IOW_N	ISA
P25	IRDY_N	PCI CONTROL	MPCI	IRDY_N	South Bridge
E04	IRQ10, PCI_INT_B	PCI INTERRUPT B	MPCI	IRQ10/PCI_INT_B	South Bridge
D01	IRQ11, PCI_INT_C	PCI INTERRUPT C	MPCI	IRQ11/PCI_INT_C	South Bridge
E03	IRQ12, PCI_INT_D	PCI INTERRUPT D	MPCI	IRQ12/PCI_INT_D	South Bridge
E02	IRQ14	INTERRUPT	Generic2	IRQ14 (IDEP_IRQ)	ISA
E01	IRQ15	INTERRUPT	Generic2	IRQ15 (IDES_IRQ)	South Bridge
B02	IRQ3	INTERRUPT	Generic2	IRQ3	ISA
C01	IRQ4	INTERRUPT	Generic2	IRQ4	South Bridge
C02	IRQ5	INTERRUPT	Generic2	IRQ5	ISA
D03	IRQ7	INTERRUPT	Generic2	IRQ7	ISA
D02	IRQ9, PCI_INT_A	PCI INTERRUPT A	MPCI	IRQ9/PCI_INT_A	South Bridge
C06	IRRX	UART & IR	Generic2	IR_RX	Super IO

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
A05	IRTX	UART & IR	Generic2	IR_TX	Super IO
AC02	ISACLK_C	ISA CLOCK	Generic2	ISACLK	ISA
AB03	ISAERR_N	ISA CONTROLS	Generic2	ISA_ERR_N	ISA
C12	KBCLK_C	KEYBOARD & MOUSE	Generic2	KBCLK	Super IO
A11	KBDAT	KEYBOARD & MOUSE	Generic2	KBDATA	Super IO
B11	KBLOCK	KEYBOARD & MOUSE	Generic2	KBLOCK_N	Super IO
AE01	32KHZ_C	Realtime CLOCK	MWUSB	CLK32KHZ	Super IO
AF01	32KHZC_C	Realtime CLOCK	MWUSB	CLK32KHZC (CLK IN)	Super IO
N25	LOCK_N	PCI CONTROL	MPCI	PLOCK_N	South Bridge
A15	MA[00]	SDRAM ADDRESS	MMC_D	A0	North Bridge
C14	MA[01]	SDRAM ADDRESS	MMC_D	A1	North Bridge
B15	MA[02]	SDRAM ADDRESS	MMC_D	A2	North Bridge
C15	MA[03]	SDRAM ADDRESS	MMC_D	A3	North Bridge
B16	MA[04]	SDRAM ADDRESS	MMC_D	A4	North Bridge
A16	MA[05]	SDRAM ADDRESS	MMC_D	A5	North Bridge
C16	MA[06]	SDRAM ADDRESS	MMC_D	A6	North Bridge
B17	MA[07]	SDRAM ADDRESS	MMC_D	A7	North Bridge
C17	MA[08]	SDRAM ADDRESS	MMC_D	A8	North Bridge
A18	MA[09]	SDRAM ADDRESS	MMC_D	A9	North Bridge
C18	MA[10]	SDRAM ADDRESS	MMC_D	A10	North Bridge
B18	MA[11]	SDRAM ADDRESS	MMC_D	A11	North Bridge
A19	MA[12]	SDRAM ADDRESS	MMC_D	A12	North Bridge
D18	MA[13]	SDRAM ADDRESS	MMC_D	A13	North Bridge
D11	MCLK_C	KEYBOARD & MOUSE	Generic2	MCLK	South Bridge
C11	MDAT	KEYBOARD & MOUSE	Generic2	MDATA	Super IO
B04	MEM_CS[0]	ZF-Logic Memory Mapper CS 0	Generic2	MEM_CS0	ZFLogic
D05	MEM_CS[1]	ZF-Logic Memory Mapper CS 1	Generic2	MEM_CS1	ZFLogic
A03	MEM_CS[2]	ZF-Logic Memory Mapper CS 2	Generic2	MEM_CS2	ZFLogic
C04	MEM_CS[3]	ZF-Logic Memory Mapper CS 3	Generic2	MEM_CS3	ZFLogic
AE07	MEMCS16_N	ISA CONTROLS	Generic2	MEMCS16_N	ISA

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AF08	MEMR_N	ISA CONTROLS	Generic2	MEMR_N	ISA
AC09	MEMW_N	ISA CONTROLS	Generic2	MEMW_N	ISA
AF16	mhz14_c	14 MHz Clock input	Generic2	CLK14MHZ (TIMER_CLK)	
B20	MR	ISA Reset Drive	Generic2	RESETDRV	ISA
F02	MTR0_N	FLOPPY	m_fdc_p	MTR0_N	Super IO
AD13	OVER_CUR1_N	USB Over Current Sense 1	MUSB	OC_SENS1	South Bridge
AF12	OVER_CUR2_N	USB Over Current Sense 2	MUSB	OC_SENS2	South Bridge
N26	PAR	PCI CONTROL	MPCI	PAR	South Bridge
U26	PCI_RST_N	PCI RESET	MPCI	PRST_N	South Bridge
U25	PCICLK_C	PCI CLOCK	MPCI_CLK	CLK	South Bridge
P03	PD[0]	PARALLEL PORT	m_fdc_p	PD0	Super IO
P01	PD[1]	PARALLEL PORT	m_fdc_p	PD1	Super IO
P02	PD[2]	PARALLEL PORT	m_fdc_p	PD2	Super IO
N01	PD[3]	PARALLEL PORT	m_fdc_p	PD3	Super IO
N02	PD[4]	PARALLEL PORT	m_fdc_p	PD4	Super IO
M01	PD[5]	PARALLEL PORT	m_fdc_p	PD5	Super IO
N03	PD[6]	PARALLEL PORT	m_fdc_p	PD6	Super IO
M02	PD[7]	PARALLEL PORT	m_fdc_p	PD7	Super IO
J01	PE	PARALLEL PORT	m_fdc_p	PE	Super IO
N24	PERR_N	PCI CONTROL	MPCI	PERR_N	South Bridge
C19	POR_N	System Reset	Generic2	RESET_N	
B19	PORDIS	Power On Reset Disable	MVBAT	POR_DIS	South Bridge
AF13	PORT1_M	USB Port1 Data Minus	MWUSB	PORT1_M	South Bridge
AE13	PORT1_P	USB Port1 Data Plus	MWUSB	PORT1_P	South Bridge
AF14	PORT2_DP	USB Port2 Data Plus	MWUSB	PORT2_P	South Bridge
AE14	PORT2_M	USB Port2 Data Minus	MWUSB	PORT2_M	South Bridge
AE12	POWER_EN	USB Power Enable	MUSB	PWR_EN	South Bridge
B05	PWM	ZF-Logic PWM Output	Generic2	PWM	ZFLogic
J04	RDATA_N	FLOPPY	m_fdc_p	RDATA_N/ZDIN	Super IO
M25	REQ0_N	PCI CONTROL	MPCI	REQ0_N	South Bridge

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
N23	REQ1_N	PCI CONTROL	MPCI	REQ1_N	South Bridge
A08	RI1_N	UART & IR	Generic2	RI1_N	Super IO
B06	RI2_N	UART & IR	Generic2	RI2_N	Super IO
A09	RTS1_N	UART & IR	Generic2	RTS1_N	Super IO
C07	RTS2_N, IRSL0	UART & IR	Generic2	RTS2_N/IR_SL0	Super IO
B10	RX1	UART & IR	Generic2	RXD1	Super IO
A07	RX2	UART & IR	Generic2	RXD2	Super IO
AC01	SA[00]	ISA ADDRESS	Generic2	SA0	ISA
AB02	SA[01]	ISA ADDRESS	Generic2	SA1	ISA
AB01	SA[02]	ISA ADDRESS	Generic2	SA2	ISA
AA03	SA[03]	ISA ADDRESS	Generic2	SA3	ISA
AA02	SA[04]	ISA ADDRESS	Generic2	SA4	ISA
Y04	SA[05]	ISA ADDRESS	Generic2	SA5	ISA
AA01	SA[06]	ISA ADDRESS	Generic2	SA6	ISA
Y02	SA[07]	ISA ADDRESS	Generic2	SA7	ISA
Y03	SA[08]	ISA ADDRESS	Generic2	SA8	ISA
Y01	SA[09]	ISA ADDRESS	Generic2	SA9	ISA
W03	SA[10]	ISA ADDRESS	Generic2	SA10	ISA
W02	SA[11]	ISA ADDRESS	Generic2	SA11	ISA
W01	SA[12]	ISA ADDRESS	Generic2	SA12	ISA
V03	SA[13]	ISA ADDRESS	Generic2	SA13	ISA
V04	SA[14]	ISA ADDRESS	Generic2	SA14	ISA
V02	SA[15]	ISA ADDRESS	Generic2	SA15	ISA
V01	SA[16]	ISA ADDRESS	Generic2	SA16	ISA
U02	SA[17]	ISA ADDRESS	Generic2	SA17	ISA
U03	SA[18]	ISA ADDRESS	Generic2	SA18	ISA
U01	SA[19]	ISA ADDRESS	Generic2	SA19	ISA
T04	SA[20]	ISA ADDRESS	Generic2	SA20	ISA
T03	SA[21]	ISA ADDRESS	Generic2	SA21	ISA
T02	SA[22]	ISA ADDRESS	Generic2	SA22	ISA

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
R03	SA[23]	ISA ADDRESS	Generic2	SA23	ISA
AC03	SBHE_N	ISA CONTROLS	Generic2	SBHE_N	ISA
B12	SCL_C	ACCESS BUS	MAC97	SCL_C	Super IO
AC07	SD[0]	ISA DATA	Generic2	SD0	ISA
AD07	SD[1]	ISA DATA	Generic2	SD1	ISA
AE04	SD[10]	ISA DATA	Generic2	SD10	ISA
AD04	SD[11]	ISA DATA	Generic2	SD11	ISA
AF03	SD[12]	ISA DATA	Generic2	SD12	ISA
AE03	SD[13]	ISA DATA	Generic2	SD13	ISA
AF02	SD[14]	ISA DATA	Generic2	SD14	ISA
AE02	SD[15]	ISA DATA	Generic2	SD15	ISA
AF06	SD[2]	ISA DATA	Generic2	SD2	ISA
AE06	SD[3]	ISA DATA	Generic2	SD3	ISA
AD06	SD[4]	ISA DATA	Generic2	SD4	ISA
AF05	SD[5]	ISA DATA	Generic2	SD5	ISA
AE05	SD[6]	ISA DATA	Generic2	SD6	ISA
AD05	SD[7]	ISA DATA	Generic2	SD7	ISA
AF04	SD[8]	ISA DATA	Generic2	SD8	ISA
AC05	SD[9]	ISA DATA	Generic2	SD9	ISA
D13	SDA	ACCESS BUS	MAC97	SDA	Super IO
D20	SDRAM_CAS_N	SDRAM CAS	MMC_D	CAS_N	North Bridge
B22	SDRAM_CLK[0]_N	SDRAM CLOCK	MMC_SDCLK	CLK0	North Bridge
A22	SDRAM_CLK[1]_N	SDRAM CLOCK	MMC_SDCLK	CLK1	North Bridge
B21	SDRAM_CLK[2]_N	SDRAM CLOCK	MMC_SDCLK	CLK2	North Bridge
A21	SDRAM_CLK[3]_N	SDRAM CLOCK	MMC_SDCLK	CLK3	North Bridge
C21	SDRAM_CLKE	SDRAM Clock Enable	MMC_D	CLKE	North Bridge
B25	SDRAM_CS[0]_N	SDRAM Chip Select	MMC_D	CS0_N	North Bridge
A25	SDRAM_CS[1]_N	SDRAM Chip Select	MMC_D	CS1_N	North Bridge
A24	SDRAM_CS[2]_N	SDRAM Chip Select	MMC_D	CS2_N	North Bridge
B24	SDRAM_CS[3]_N	SDRAM Chip Select	MMC_D	CS3_N	North Bridge

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
C23	SDRAM_DQM[0]_N	SDRAM Mask / Command	MMC_D	DQM0_N	North Bridge
B23	SDRAM_DQM[1]_N	SDRAM Mask / Command	MMC_D	DQM1_N	North Bridge
D22	SDRAM_DQM[2]_N	SDRAM Mask / Command	MMC_D	DQM2_N	North Bridge
A23	SDRAM_DQM[3]_N	SDRAM Mask / Command	MMC_D	DQM3_N	North Bridge
C20	SDRAM_RAS_N	SDRAM RAS	MMC_D	RAS_N	North Bridge
C22	SDRAM_WE_N	SDRAM Write Enable	MMC_D	WE_N	North Bridge
M26	SERR_N	PCI CONTROL	MPCI	SERR_N	South Bridge
J03	SLCT	PARALLEL PORT	m_fdc_p	SLCT	Super IO
K01	SLIN_N	PARALLEL PORT	m_fdc_p	SLIN_N	Super IO
AE08	SMEMR_N	ISA CONTROLS	Generic2	SMEMR_N	ISA
AD08	SMEMW_N	ISA CONTROLS	Generic2	SMEMW_N	ISA
AC14	SPARE1	Spare	Generic	SPARE1	
M03	STB_N	PARALLEL PORT	m_fdc_p	STB_N	Super IO
G04	STEP_N	FLOPPY	m_fdc_p	STEP_N/ZLED2	Super IO
P26	STOP_N	PCI CONTROL	MPCI	STOP_N	South Bridge
A20	SYSCLK_C	System CLOCK	MMC_SDCLKIN	CLK33MHZ (SYS_CLK)	Processor
A12	TC	ISA DMA	Generic2	TC	ISA
R01	TCK_C	JTAG (system)	Generic2	TCK	South Bridge
P04	TDI	JTAG (system)	Generic2	TDI	South Bridge
T01	TDO	JTAG (system)	Generic2	TDO	South Bridge
R02	TMS	JTAG (system)	Generic2	TMS	South Bridge
R26	TRDY_N	PCI CONTROL	MPCI	TRDY_N	South Bridge
H03	TRK0_N	FLOPPY	m_fdc_p	TRK0_N/ZGPI0	Super IO
B09	TX1	UART & IR	Generic2	TXD1	Super IO
B07	TX2	UART & IR	Generic2	TXD2	Super IO
AE15	USB_48MHZ_C	USB CLOCK	MMC_SDCLKIN	CLK48MHZ (USB_CLK)	South Bridge
AD14	USB_GND	USB circuit ground	MWUSB	GND_USB	South Bridge
AF15	USB_PWR	USB circuit power	MWUSB	VDD_USB	South Bridge
AD03	VBAT	Realtime clock battery backup	MVBAT	VBAT	Super IO
AA04	VDD_CORE			VDD_CORE	

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AA23	VDD_CORE			VDD_CORE	
AC06	VDD_CORE			VDD_CORE	
AC10	VDD_CORE			VDD_CORE	
AC12	VDD_CORE			VDD_CORE	
AC17	VDD_CORE			VDD_CORE	
AC21	VDD_CORE			VDD_CORE	
D06	VDD_CORE			VDD_CORE	
D10	VDD_CORE			VDD_CORE	
D15	VDD_CORE			VDD_CORE	
D17	VDD_CORE			VDD_CORE	
D21	VDD_CORE			VDD_CORE	
F04	VDD_CORE			VDD_CORE	
F23	VDD_CORE			VDD_CORE	
K04	VDD_CORE			VDD_CORE	
K23	VDD_CORE			VDD_CORE	
M04	VDD_CORE			VDD_CORE	
R23	VDD_CORE			VDD_CORE	
U04	VDD_CORE			VDD_CORE	
U23	VDD_CORE			VDD_CORE	
AC08	VDD_IO			VDD_IO	
AC11	VDD_IO			VDD_IO	
AC15	VDD_IO			VDD_IO	
AC19	VDD_IO			VDD_IO	
D08	VDD_IO			VDD_IO	
D12	VDD_IO			VDD_IO	
D16	VDD_IO			VDD_IO	
D19	VDD_IO			VDD_IO	
H04	VDD_IO			VDD_IO	
H23	VDD_IO			VDD_IO	
L04	VDD_IO			VDD_IO	

Table 8.3 Pin Descriptions Sorted by Pin Name (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
M23	VDD_IO			VDD_IO	
R04	VDD_IO			VDD_IO	
T23	VDD_IO			VDD_IO	
W04	VDD_IO			VDD_IO	
W23	VDD_IO			VDD_IO	
G02	WDATA_N	FLOPPY	m_fdc_p	WDATA_N/ZCLK	Super IO
A04	WDI	ZF Logic - Watch Dog Timer	Generic2	WDI	ZFLogic
C05	WDO	ZF Logic - Watch Dog Timer	Generic2	WDO	ZFLogic
G01	WGATE_N	FLOPPY	m_fdc_p	WGATE_N	Super IO
H02	WRPRT_N	FLOPPY	m_fdc_p	WRPRT_N/ZACK	Super IO
AB04	ZWS_N	ISA CONTROLS	Generic2	ZWS_N	ISA

8.4. Pin Descriptions (Sorted by Pin Description)

Table 8.4 Pin Descriptions Sorted by Pin Description

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AC04	GND			GND	
AC13	GND			GND	
AC23	GND			GND	
D04	GND			GND	
D14	GND			GND	
D23	GND			GND	
L11	GND			GND	
L12	GND			GND	
L13	GND			GND	
L14	GND			GND	
L15	GND			GND	
L16	GND			GND	
M11	GND			GND	
M12	GND			GND	
M13	GND			GND	
M14	GND			GND	
M15	GND			GND	
M16	GND			GND	
N04	GND			GND	
N11	GND			GND	
N12	GND			GND	
N13	GND			GND	
N14	GND			GND	
N15	GND			GND	
N16	GND			GND	
P11	GND			GND	
P12	GND			GND	
P13	GND			GND	

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
P14	GND			GND	
P15	GND			GND	
P16	GND			GND	
P23	GND			GND	
R11	GND			GND	
R12	GND			GND	
R13	GND			GND	
R14	GND			GND	
R15	GND			GND	
R16	GND			GND	
T11	GND			GND	
T12	GND			GND	
T13	GND			GND	
T14	GND			GND	
T15	GND			GND	
T16	GND			GND	
AA04	VDD_CORE			VDD_CORE	
AA23	VDD_CORE			VDD_CORE	
AC06	VDD_CORE			VDD_CORE	
AC10	VDD_CORE			VDD_CORE	
AC12	VDD_CORE			VDD_CORE	
AC17	VDD_CORE			VDD_CORE	
AC21	VDD_CORE			VDD_CORE	
D06	VDD_CORE			VDD_CORE	
D10	VDD_CORE			VDD_CORE	
D15	VDD_CORE			VDD_CORE	
D17	VDD_CORE			VDD_CORE	
D21	VDD_CORE			VDD_CORE	
F04	VDD_CORE			VDD_CORE	
F23	VDD_CORE			VDD_CORE	

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
K04	VDD_CORE			VDD_CORE	
K23	VDD_CORE			VDD_CORE	
M04	VDD_CORE			VDD_CORE	
R23	VDD_CORE			VDD_CORE	
U04	VDD_CORE			VDD_CORE	
U23	VDD_CORE			VDD_CORE	
AC08	VDD_IO			VDD_IO	
AC11	VDD_IO			VDD_IO	
AC15	VDD_IO			VDD_IO	
AC19	VDD_IO			VDD_IO	
D08	VDD_IO			VDD_IO	
D12	VDD_IO			VDD_IO	
D16	VDD_IO			VDD_IO	
D19	VDD_IO			VDD_IO	
H04	VDD_IO			VDD_IO	
H23	VDD_IO			VDD_IO	
L04	VDD_IO			VDD_IO	
M23	VDD_IO			VDD_IO	
R04	VDD_IO			VDD_IO	
T23	VDD_IO			VDD_IO	
W04	VDD_IO			VDD_IO	
W23	VDD_IO			VDD_IO	
AF16	mhz14_c	14 MHz Clock input	Generic2	CLK14MHZ (TIMER_CLK)	
B12	SCL_C	ACCESS BUS	MAC97	SCL_C	Super IO
D13	SDA	ACCESS BUS	MAC97	SDA	Super IO
A17	CPU_TRIG	CPU Trigger	Generic2	CPU_TRIG	
G03	DIR _N	FLOPPY	m_fdc_p	DIR_N/ZRST	Super IO
F01	DR0_N	FLOPPY	m_fdc_p	DR0_N	Super IO
J02	DSKCHG_N	FLOPPY	m_fdc_p	DSKCHG_N/ZGPI1	Super IO
H01	HDSEL_N	FLOPPY	m_fdc_p	HDSEL_N/ZLED1	Super IO

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
F03	INDEX_N	FLOPPY	m_fdc_p	INDEX_N	Super IO
F02	MTR0_N	FLOPPY	m_fdc_p	MTR0_N	Super IO
J04	RDATA_N	FLOPPY	m_fdc_p	RDATA_N/ZDIN	Super IO
G04	STEP_N	FLOPPY	m_fdc_p	STEP_N/ZLED2	Super IO
H03	TRK0_N	FLOPPY	m_fdc_p	TRK0_N/ZGPI0	Super IO
G02	WDATA_N	FLOPPY	m_fdc_p	WDATA_N/ZCLK	Super IO
G01	WGATE_N	FLOPPY	m_fdc_p	WGATE_N	Super IO
H02	WRPRT_N	FLOPPY	m_fdc_p	WRPRT_N/ZACK	Super IO
AF10	GPIO[4]	GPIO	Generic2	GPIO4	South Bridge
AF09	GPIO[7]	GPIO	Generic2	GPIO7	South Bridge
AD11	GPIO[3], IDE_IOR1_N	GPIO (optional 2nd IDE dior)	Generic2	GPIO3/IDE_DIORD1_N	South Bridge
AD10	GPIO[6], IDE_RDY1	GPIO (optional 2nd IDE diordy)	Generic2	GPIO6/IDE_IORDY1	South Bridge
AF11	GPIO[2], IDE_IOW1_N	GPIO (optional 2nd IDE diow)	Generic2	GPIO2/IDE_IOW1_N	South Bridge
AE11	GPIO[1], IDE_DMA_ACK1_N	GPIO (optional 2nd IDE dmack)	Generic2	GPIO1/ IDE_DACK1_N	South Bridge
AE10	GPIO[5], IDE_DMA_REQ1_N	GPIO (optional 2nd IDE dreq)	Generic2	GPIO5/ IDE_DREQ1	South Bridge
AD12	GPIO[0], CLK32KHZ_OUT	GPIO (optional 32KHz out)	Generic2	GPIO0/CLK32KHZ_OUT	South Bridge
AD21	IDE_ADDR0	IDE CONTROL	MIDE	IDE_ADDR0	South Bridge
AF22	IDE_ADDR1	IDE CONTROL	MIDE	IDE_ADDR1	South Bridge
AE22	IDE_ADDR2	IDE CONTROL	MIDE	IDE_ADDR2	South Bridge
AE20	IDE_CS0_N	IDE CONTROL	MIDE	IDE_CS0_N	South Bridge
AF21	IDE_CS1_N	IDE CONTROL	MIDE	IDE_CS1_N	South Bridge
AC22	IDE_DMA_ACK0_N	IDE CONTROL	MIDE	IDE_DACK0_N	South Bridge
AE23	IDE_DMA_REQ0_N	IDE CONTROL	MIDE	IDE_DREQ0_N	South Bridge
AC20	IDE_IOR0_N	IDE CONTROL	MIDE	IDE_IOR0_N	South Bridge
AE21	IDE_IOW0_N	IDE CONTROL	MIDE	IDE_IOW0_N	South Bridge
AD22	IDE_RDY0	IDE CONTROL	MIDE	IDE_IORDY0_N	South Bridge
AF23	IDE_RST_N	IDE CONTROL	MIDE	IDE_RST_N	South Bridge

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AD20	IDE_DATA[00]	IDE DATA	MIDE	IDE_D0	South Bridge
AF20	IDE_DATA[01]	IDE DATA	MIDE	IDE_D1	South Bridge
AD19	IDE_DATA[02]	IDE DATA	MIDE	IDE_D2	South Bridge
AE19	IDE_DATA[03]	IDE DATA	MIDE	IDE_D3	South Bridge
AF19	IDE_DATA[04]	IDE DATA	MIDE	IDE_D4	South Bridge
AD18	IDE_DATA[05]	IDE DATA	MIDE	IDE_D5	South Bridge
AC18	IDE_DATA[06]	IDE DATA	MIDE	IDE_D6	South Bridge
AE18	IDE_DATA[07]	IDE DATA	MIDE	IDE_D7	South Bridge
AF18	IDE_DATA[08]	IDE DATA	MIDE	IDE_D8	South Bridge
AE17	IDE_DATA[09]	IDE DATA	MIDE	IDE_D9	South Bridge
AD17	IDE_DATA[10]	IDE DATA	MIDE	IDE_D10	South Bridge
AF17	IDE_DATA[11]	IDE DATA	MIDE	IDE_D11	South Bridge
AC16	IDE_DATA[12]	IDE DATA	MIDE	IDE_D12	South Bridge
AD16	IDE_DATA[13]	IDE DATA	MIDE	IDE_D13	South Bridge
AE16	IDE_DATA[14]	IDE DATA	MIDE	IDE_D14	South Bridge
AD15	IDE_DATA[15]	IDE DATA	MIDE	IDE_D15	South Bridge
E02	IRQ14	INTERRUPT	Generic2	IRQ14 (IDEP_IRQ)	ISA
E01	IRQ15	INTERRUPT	Generic2	IRQ15 (IDES_IRQ)	South Bridge
B02	IRQ3	INTERRUPT	Generic2	IRQ3	ISA
C01	IRQ4	INTERRUPT	Generic2	IRQ4	South Bridge
C02	IRQ5	INTERRUPT	Generic2	IRQ5	ISA
D03	IRQ7	INTERRUPT	Generic2	IRQ7	ISA
AC01	SA[00]	ISA ADDRESS	Generic2	SA0	ISA
AB02	SA[01]	ISA ADDRESS	Generic2	SA1	ISA
AB01	SA[02]	ISA ADDRESS	Generic2	SA2	ISA
AA03	SA[03]	ISA ADDRESS	Generic2	SA3	ISA
AA02	SA[04]	ISA ADDRESS	Generic2	SA4	ISA
Y04	SA[05]	ISA ADDRESS	Generic2	SA5	ISA
AA01	SA[06]	ISA ADDRESS	Generic2	SA6	ISA
Y02	SA[07]	ISA ADDRESS	Generic2	SA7	ISA

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
Y03	SA[08]	ISA ADDRESS	Generic2	SA8	ISA
Y01	SA[09]	ISA ADDRESS	Generic2	SA9	ISA
W03	SA[10]	ISA ADDRESS	Generic2	SA10	ISA
W02	SA[11]	ISA ADDRESS	Generic2	SA11	ISA
W01	SA[12]	ISA ADDRESS	Generic2	SA12	ISA
V03	SA[13]	ISA ADDRESS	Generic2	SA13	ISA
V04	SA[14]	ISA ADDRESS	Generic2	SA14	ISA
V02	SA[15]	ISA ADDRESS	Generic2	SA15	ISA
V01	SA[16]	ISA ADDRESS	Generic2	SA16	ISA
U02	SA[17]	ISA ADDRESS	Generic2	SA17	ISA
U03	SA[18]	ISA ADDRESS	Generic2	SA18	ISA
U01	SA[19]	ISA ADDRESS	Generic2	SA19	ISA
T04	SA[20]	ISA ADDRESS	Generic2	SA20	ISA
T03	SA[21]	ISA ADDRESS	Generic2	SA21	ISA
T02	SA[22]	ISA ADDRESS	Generic2	SA22	ISA
R03	SA[23]	ISA ADDRESS	Generic2	SA23	ISA
AC02	ISACLK_C	ISA CLOCK	Generic2	ISACLK	ISA
AD02	BALE	ISA CONTROLS	Generic2	BALE	ISA
AD01	IOCHRDY	ISA CONTROLS	Generic2	IOCHRDY	ISA
AF07	IOCS16_N	ISA CONTROLS	Generic2	IOCS16_N	ISA
AD09	IOR_N	ISA CONTROLS	Generic2	IOR_N	ISA
AE09	IOW_N	ISA CONTROLS	Generic2	IOW_N	ISA
AB03	ISAERR_N	ISA CONTROLS	Generic2	ISA_ERR_N	ISA
AE07	MEMCS16_N	ISA CONTROLS	Generic2	MEMCS16_N	ISA
AF08	MEMR_N	ISA CONTROLS	Generic2	MEMR_N	ISA
AC09	MEMW_N	ISA CONTROLS	Generic2	MEMW_N	ISA
AC03	SBHE_N	ISA CONTROLS	Generic2	SBHE_N	ISA
AE08	SMEMR_N	ISA CONTROLS	Generic2	SMEMR_N	ISA
AD08	SMEMW_N	ISA CONTROLS	Generic2	SMEMW_N	ISA
AB04	ZWS_N	ISA CONTROLS	Generic2	ZWS_N	ISA

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
AC07	SD[0]	ISA DATA	Generic2	SD0	ISA
AD07	SD[1]	ISA DATA	Generic2	SD1	ISA
AE04	SD[10]	ISA DATA	Generic2	SD10	ISA
AD04	SD[11]	ISA DATA	Generic2	SD11	ISA
AF03	SD[12]	ISA DATA	Generic2	SD12	ISA
AE03	SD[13]	ISA DATA	Generic2	SD13	ISA
AF02	SD[14]	ISA DATA	Generic2	SD14	ISA
AE02	SD[15]	ISA DATA	Generic2	SD15	ISA
AF06	SD[2]	ISA DATA	Generic2	SD2	ISA
AE06	SD[3]	ISA DATA	Generic2	SD3	ISA
AD06	SD[4]	ISA DATA	Generic2	SD4	ISA
AF05	SD[5]	ISA DATA	Generic2	SD5	ISA
AE05	SD[6]	ISA DATA	Generic2	SD6	ISA
AD05	SD[7]	ISA DATA	Generic2	SD7	ISA
AF04	SD[8]	ISA DATA	Generic2	SD8	ISA
AC05	SD[9]	ISA DATA	Generic2	SD9	ISA
C13	AEN	ISA DMA	Generic2	AEN	ISA
A13	DACK5_N	ISA DMA	Generic2	DACK5_5	ISA
B13	DRQ5	ISA DMA	Generic2	DRQ5	ISA
A12	TC	ISA DMA	Generic2	TC	ISA
A14	DACK1_N	ISA DMA (optional PCI Master gnt2_n)	Generic2	DACK1_N	ISA
B14	DRQ1	ISA DMA (Optional PCI Master req2_n)	Generic2	DRQ1	ISA
B20	MR	ISA Reset Drive	Generic2	RESETDRV	ISA
R01	TCK_C	JTAG (system)	Generic2	TCK	South Bridge
P04	TDI	JTAG (system)	Generic2	TDI	South Bridge
T01	TDO	JTAG (system)	Generic2	TDO	South Bridge
R02	TMS	JTAG (system)	Generic2	TMS	South Bridge
C12	KBCLK_C	KEYBOARD & MOUSE	Generic2	KBCLK	Super IO
A11	KBDAT	KEYBOARD & MOUSE	Generic2	KBDATA	Super IO

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
B11	KBLOCK	KEYBOARD & MOUSE	Generic2	KBLOCK_N	Super IO
D11	MCLK_C	KEYBOARD & MOUSE	Generic2	MCLK	South Bridge
C11	MDAT	KEYBOARD & MOUSE	Generic2	MDATA	Super IO
K02	ACK_N	PARALLEL PORT	m_fdc_p	ACK_N	Super IO
L02	AFD_N	PARALLEL PORT	m_fdc_p	AFD_N	Super IO
K03	BUSY	PARALLEL PORT	m_fdc_p	BUSY	Super IO
L01	ERR_N	PARALLEL PORT	m_fdc_p	ERR_N	Super IO
L03	INIT_N	PARALLEL PORT	m_fdc_p	INIT_N	Super IO
P03	PD[0]	PARALLEL PORT	m_fdc_p	PD0	Super IO
P01	PD[1]	PARALLEL PORT	m_fdc_p	PD1	Super IO
P02	PD[2]	PARALLEL PORT	m_fdc_p	PD2	Super IO
N01	PD[3]	PARALLEL PORT	m_fdc_p	PD3	Super IO
N02	PD[4]	PARALLEL PORT	m_fdc_p	PD4	Super IO
M01	PD[5]	PARALLEL PORT	m_fdc_p	PD5	Super IO
N03	PD[6]	PARALLEL PORT	m_fdc_p	PD6	Super IO
M02	PD[7]	PARALLEL PORT	m_fdc_p	PD7	Super IO
J01	PE	PARALLEL PORT	m_fdc_p	PE	Super IO
J03	SLCT	PARALLEL PORT	m_fdc_p	SLCT	Super IO
K01	SLIN_N	PARALLEL PORT	m_fdc_p	SLIN_N	Super IO
M03	STB_N	PARALLEL PORT	m_fdc_p	STB_N	Super IO
B01	BEEP_N	PC Speaker	Generic2	BEEP_N	Super IO
U25	PCICLK_C	PCI CLOCK	MPCI_CLK	CLK	South Bridge
D02	IRQ9, PCI_INT_A	PCI INTERRUPT A	MPCI	IRQ9/PCI_INT_A	South Bridge
E04	IRQ10, PCI_INT_B	PCI INTERRUPT B	MPCI	IRQ10/PCI_INT_B	South Bridge
D01	IRQ11, PCI_INT_C	PCI INTERRUPT C	MPCI	IRQ11/PCI_INT_C	South Bridge
E03	IRQ12, PCI_INT_D	PCI INTERRUPT D	MPCI	IRQ12/PCI_INT_D	South Bridge
U24	AD[00]	PCI ADDRESS & DATA	MPCI	AD0	South Bridge
V26	AD[01]	PCI ADDRESS & DATA	MPCI	AD1	South Bridge
V24	AD[02]	PCI ADDRESS & DATA	MPCI	AD2	South Bridge
V25	AD[03]	PCI ADDRESS & DATA	MPCI	AD3	South Bridge

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
W26	AD[04]	PCI ADDRESS & DATA	MPCI	AD4	South Bridge
V23	AD[05]	PCI ADDRESS & DATA	MPCI	AD5	South Bridge
W25	AD[06]	PCI ADDRESS & DATA	MPCI	AD6	South Bridge
W24	AD[07]	PCI ADDRESS & DATA	MPCI	AD7	South Bridge
Y26	AD[08]	PCI ADDRESS & DATA	MPCI	AD8	South Bridge
Y25	AD[09]	PCI ADDRESS & DATA	MPCI	AD9	South Bridge
Y23	AD[10]	PCI ADDRESS & DATA	MPCI	AD10	South Bridge
Y24	AD[11]	PCI ADDRESS & DATA	MPCI	AD11	South Bridge
AA26	AD[12]	PCI ADDRESS & DATA	MPCI	AD12	South Bridge
AA25	AD[13]	PCI ADDRESS & DATA	MPCI	AD13	South Bridge
AA24	AD[14]	PCI ADDRESS & DATA	MPCI	AD14	South Bridge
AB26	AD[15]	PCI ADDRESS & DATA	MPCI	AD15	South Bridge
AB25	AD[16]	PCI ADDRESS & DATA	MPCI	AD16	South Bridge
AB24	AD[17]	PCI ADDRESS & DATA	MPCI	AD17	South Bridge
AC26	AD[18]	PCI ADDRESS & DATA	MPCI	AD18	South Bridge
AB23	AD[19]	PCI ADDRESS & DATA	MPCI	AD19	South Bridge
AC25	AD[20]	PCI ADDRESS & DATA	MPCI	AD20	South Bridge
AC24	AD[21]	PCI ADDRESS & DATA	MPCI	AD21	South Bridge
AD25	AD[22]	PCI ADDRESS & DATA	MPCI	AD22	South Bridge
AD26	AD[23]	PCI ADDRESS & DATA	MPCI	AD23	South Bridge
AE26	AD[24]	PCI ADDRESS & DATA	MPCI	AD24	South Bridge
AE25	AD[25]	PCI ADDRESS & DATA	MPCI	AD25	South Bridge
AD24	AD[26]	PCI ADDRESS & DATA	MPCI	AD26	South Bridge
AF26	AD[27]	PCI ADDRESS & DATA	MPCI	AD27	South Bridge
AF25	AD[28]	PCI ADDRESS & DATA	MPCI	AD28	South Bridge
AE24	AD[29]	PCI ADDRESS & DATA	MPCI	AD29	South Bridge
AD23	AD[30]	PCI ADDRESS & DATA	MPCI	AD30	South Bridge
AF24	AD[31]	PCI ADDRESS & DATA	MPCI	AD31	South Bridge
T24	C/BE[0]_N	PCI COMMAND / BYTE	MPCI	C_BE0_N	South Bridge
T26	C/BE[1]_N	PCI COMMAND / BYTE	MPCI	C_BE1_N	South Bridge

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
T25	C/BE[2]_N	PCI COMMAND / BYTE	MPCI	C_BE2_N	South Bridge
R24	C/BE[3]_N	PCI COMMAND / BYTE	MPCI	C_BE3_N	South Bridge
R25	DEVSEL_N	PCI CONTROL	MPCI	DEVSEL_N	South Bridge
P24	FRAME_N	PCI CONTROL	MPCI	FRAME_N	South Bridge
L26	GNT0_N	PCI CONTROL	MPCI	GNT0_N	South Bridge
M24	GNT1_N	PCI CONTROL	MPCI	GNT1_N	South Bridge
P25	IRDY_N	PCI CONTROL	MPCI	IRDY_N	South Bridge
N25	LOCK_N	PCI CONTROL	MPCI	PLOCK_N	South Bridge
N26	PAR	PCI CONTROL	MPCI	PAR	South Bridge
N24	PERR_N	PCI CONTROL	MPCI	PERR_N	South Bridge
M25	REQ0_N	PCI CONTROL	MPCI	REQ0_N	South Bridge
N23	REQ1_N	PCI CONTROL	MPCI	REQ1_N	South Bridge
M26	SERR_N	PCI CONTROL	MPCI	SERR_N	South Bridge
P26	STOP_N	PCI CONTROL	MPCI	STOP_N	South Bridge
R26	TRDY_N	PCI CONTROL	MPCI	TRDY_N	South Bridge
U26	PCI_RST_N	PCI RESET	MPCI	PRST_N	South Bridge
B19	PORDIS	Power On Reset Disable	MVBAT	POR_DIS	South Bridge
AE01	32KHZ_C	Realtime CLOCK	MWUSB	CLK32KHZ	Super IO
AF01	32KHZC_C	Realtime CLOCK	MWUSB	CLK32KHZC (CLK IN)	Super IO
AD03	VBAT	Realtime clock battery backup	MVBAT	VBAT	Super IO
A15	MA[00]	SDRAM ADDRESS	MMC_D	A0	North Bridge
C14	MA[01]	SDRAM ADDRESS	MMC_D	A1	North Bridge
B15	MA[02]	SDRAM ADDRESS	MMC_D	A2	North Bridge
C15	MA[03]	SDRAM ADDRESS	MMC_D	A3	North Bridge
B16	MA[04]	SDRAM ADDRESS	MMC_D	A4	North Bridge
A16	MA[05]	SDRAM ADDRESS	MMC_D	A5	North Bridge
C16	MA[06]	SDRAM ADDRESS	MMC_D	A6	North Bridge
B17	MA[07]	SDRAM ADDRESS	MMC_D	A7	North Bridge
C17	MA[08]	SDRAM ADDRESS	MMC_D	A8	North Bridge
A18	MA[09]	SDRAM ADDRESS	MMC_D	A9	North Bridge

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
C18	MA[10]	SDRAM ADDRESS	MMC_D	A10	North Bridge
B18	MA[11]	SDRAM ADDRESS	MMC_D	A11	North Bridge
A19	MA[12]	SDRAM ADDRESS	MMC_D	A12	North Bridge
D18	MA[13]	SDRAM ADDRESS	MMC_D	A13	North Bridge
D20	SDRAM_CAS_N	SDRAM CAS	MMC_D	CAS_N	North Bridge
B25	SDRAM_CS[0]_N	SDRAM Chip Select	MMC_D	CS0_N	North Bridge
A25	SDRAM_CS[1]_N	SDRAM Chip Select	MMC_D	CS1_N	North Bridge
A24	SDRAM_CS[2]_N	SDRAM Chip Select	MMC_D	CS2_N	North Bridge
B24	SDRAM_CS[3]_N	SDRAM Chip Select	MMC_D	CS3_N	North Bridge
B22	SDRAM_CLK[0]_N	SDRAM CLOCK	MMC_SDCLK	CLK0	North Bridge
A22	SDRAM_CLK[1]_N	SDRAM CLOCK	MMC_SDCLK	CLK1	North Bridge
B21	SDRAM_CLK[2]_N	SDRAM CLOCK	MMC_SDCLK	CLK2	North Bridge
A21	SDRAM_CLK[3]_N	SDRAM CLOCK	MMC_SDCLK	CLK3	North Bridge
C21	SDRAM_CLKE	SDRAM Clock Enable	MMC_D	CLKE	North Bridge
C24	D[00]	SDRAM DATA	MMC_D	D0	North Bridge
A26	D[01]	SDRAM DATA	MMC_D	D1	North Bridge
B26	D[02]	SDRAM DATA	MMC_D	D2	North Bridge
C25	D[03]	SDRAM DATA	MMC_D	D3	North Bridge
D24	D[04]	SDRAM DATA	MMC_D	D4	North Bridge
C26	D[05]	SDRAM DATA	MMC_D	D5	North Bridge
E23	D[06]	SDRAM DATA	MMC_D	D6	North Bridge
D25	D[07]	SDRAM DATA	MMC_D	D7	North Bridge
E24	D[08]	SDRAM DATA	MMC_D	D8	North Bridge
D26	D[09]	SDRAM DATA	MMC_D	D9	North Bridge
E25	D[10]	SDRAM DATA	MMC_D	D10	North Bridge
E26	D[11]	SDRAM DATA	MMC_D	D11	North Bridge
F24	D[12]	SDRAM DATA	MMC_D	D12	North Bridge
F25	D[13]	SDRAM DATA	MMC_D	D13	North Bridge
G23	D[14]	SDRAM DATA	MMC_D	D14	North Bridge
F26	D[15]	SDRAM DATA	MMC_D	D15	North Bridge

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
G25	D[16]	SDRAM DATA	MMC_D	D16	North Bridge
G24	D[17]	SDRAM DATA	MMC_D	D17	North Bridge
G26	D[18]	SDRAM DATA	MMC_D	D18	North Bridge
H24	D[19]	SDRAM DATA	MMC_D	D19	North Bridge
H25	D[20]	SDRAM DATA	MMC_D	D20	North Bridge
H26	D[21]	SDRAM DATA	MMC_D	D21	North Bridge
J24	D[22]	SDRAM DATA	MMC_D	D22	North Bridge
J23	D[23]	SDRAM DATA	MMC_D	D23	North Bridge
J25	D[24]	SDRAM DATA	MMC_D	D24	North Bridge
J26	D[25]	SDRAM DATA	MMC_D	D25	North Bridge
K25	D[26]	SDRAM DATA	MMC_D	D26	North Bridge
K24	D[27]	SDRAM DATA	MMC_D	D27	North Bridge
K26	D[28]	SDRAM DATA	MMC_D	D28	North Bridge
L23	D[29]	SDRAM DATA	MMC_D	D29	North Bridge
L24	D[30]	SDRAM DATA	MMC_D	D30	North Bridge
L25	D[31]	SDRAM DATA	MMC_D	D31	North Bridge
C23	SDRAM_DQM[0]_N	SDRAM Mask / Command	MMC_D	DQM0_N	North Bridge
B23	SDRAM_DQM[1]_N	SDRAM Mask / Command	MMC_D	DQM1_N	North Bridge
D22	SDRAM_DQM[2]_N	SDRAM Mask / Command	MMC_D	DQM2_N	North Bridge
A23	SDRAM_DQM[3]_N	SDRAM Mask / Command	MMC_D	DQM3_N	North Bridge
C20	SDRAM_RAS_N	SDRAM RAS	MMC_D	RAS_N	North Bridge
C22	SDRAM_WE_N	SDRAM Write Enable	MMC_D	WE_N	North Bridge
AC14	SPARE1	Spare	Generic	SPARE1	
A20	SYSCLK_C	System CLOCK	MMC_SDCLK IN	CLK33MHZ (SYS_CLK)	Processor
C19	POR_N	System Reset	Generic2	RESET_N	
D09	CTS1_N	UART & IR	Generic2	CTS1_N	Super IO
A06	CTS2_N, IRSL3	UART & IR	Generic2	CTS2_N/IR_SL3	Super IO
A10	DCD1_N	UART & IR	Generic2	DCD1_N	Super IO
B08	DCD2_N, IRSL2	UART & IR	Generic2	DCD2_N/IR_SL2	Super IO
C10	DSR1_N	UART & IR	Generic2	DSR1_N	Super IO

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
C08	DSR2_N, IRSL1	UART & IR	Generic2	DSR2_N/IR_SL1	Super IO
C09	DTR1_N	UART & IR	Generic2	DTR1_N	Super IO
D07	DTR2_N	UART & IR	Generic2	DTR2_N	Super IO
C06	IRRX	UART & IR	Generic2	IR_RX	Super IO
A05	IRTX	UART & IR	Generic2	IR_TX	Super IO
A08	RI1_N	UART & IR	Generic2	RI1_N	Super IO
B06	RI2_N	UART & IR	Generic2	RI2_N	Super IO
A09	RTS1_N	UART & IR	Generic2	RTS1_N	Super IO
C07	RTS2_N, IRSL0	UART & IR	Generic2	RTS2_N/IR_SL0	Super IO
B10	RX1	UART & IR	Generic2	RXD1	Super IO
A07	RX2	UART & IR	Generic2	RXD2	Super IO
B09	TX1	UART & IR	Generic2	TXD1	Super IO
B07	TX2	UART & IR	Generic2	TXD2	Super IO
AE15	USB_48MHZ_C	USB CLOCK	MMC_SDCLK IN	CLK48MHZ (USB_CLK)	South Bridge
AD14	USB_GND	USB circuit ground	MWUSB	GND_USB	South Bridge
AF15	USB_PWR	USB circuit power	MWUSB	VDD_USB	South Bridge
AD13	OVER_CUR1_N	USB Over Current Sense 1	MUSB	OC_SENS1	South Bridge
AF12	OVER_CUR2_N	USB Over Current Sense 2	MUSB	OC_SENS2	South Bridge
AF13	PORT1_M	USB Port1 Data Minus	MWUSB	PORT1_M	South Bridge
AE13	PORT1_P	USB Port1 Data Plus	MWUSB	PORT1_P	South Bridge
AE14	PORT2_M	USB Port2 Data Minus	MWUSB	PORT2_M	South Bridge
AF14	PORT2_DP	USB Port2 Data Plus	MWUSB	PORT2_P	South Bridge
AE12	POWER_EN	USB Power Enable	MUSB	PWR_EN	South Bridge
A04	WDI	ZF Logic - Watchdog Timer	Generic2	WDI	ZFLogic
C05	WDO	ZF Logic - Watchdog Timer	Generic2	WDO	ZFLogic
B03	I0_CS[0]	ZF-Logic I/O Mapper GPCS 0	Generic2	IO_CS0	ZFLogic
A02	I0_CS[1]	ZF-Logic I/O Mapper GPCS 1	Generic2	I0_CS1	ZFLogic

Table 8.4 Pin Descriptions Sorted by Pin Description (cont.)

Pin	Pin Name	Pin Description	Cell Type	ORCAD Name	Used By
A01	I0_CS[2]	ZF-Logic I/O Mapper GPCS 2	Generic2	I0_CS2	ZFLogic
C03	I0_CS[3]	ZF-Logic I/O Mapper GPCS 3	Generic2	IO_CS3	ZFLogic
B04	MEM_CS[0]	ZF-Logic Memory Mapper CS 0	Generic2	MEM_CS0	ZFLogic
D05	MEM_CS[1]	ZF-Logic Memory Mapper CS 1	Generic2	MEM_CS1	ZFLogic
A03	MEM_CS[2]	ZF-Logic Memory Mapper CS 2	Generic2	MEM_CS2	ZFLogic
C04	MEM_CS[3]	ZF-Logic Memory Mapper CS 3	Generic2	MEM_CS3	ZFLogic
B05	PWM	ZF-Logic PWM Output	Generic2	PWM	ZFLogic

9. BUR API

9.1. Using the BUR API

All function calls from BURAPI are performed by ordinary near calls. The BURAPI wrapper will perform the specific far call into the BUR code and return to API. API then returns by near RET into the caller program.

All parameter passing is made through registers, stack is never used for BUR call parameter passing in any direction.

The sample function call would be:

```
          org0
     .
     .
0xxxh  B8 34 12movax, 1234h; set
parameter
0xxxh  E8 xxxxcallSomeBURFunction
     .
     .
          retf
Include BURAPI.ASM
```

It is important to keep in mind that all programs written for BUR environment must start from relative address 0h. This is necessary because all program loaders in BUR are loading the executable data by default to offset 0 of the first available segment address.

BUR programs can use all registers freely, except CS and SS:SP. These registers must be set to the original state before returning the execution to the BUR. All other registers are preserved automatically by BUR upon calling the user software and are restored afterwards as necessary for BUR internal functionality.

All BUR programs run without any CPU protection and can therefore modify the BUR data area, interrupt vector table etc. . Although sometimes useful, this behavior must be used cautiously since modifying these areas may lead to unstable operation of BUR code and cause lot of confusion.

There is, however, no way to cause permanent damage to BUR because all code is 100% inside ROM and next POR will always start up the BUR in it’s original fashion.

BURAPI.ASM is illustrated in Chapter 4 of the ZFx86 Integrated Development System Quick Start Guide. See also the IDS CD directory \BUR Programs\AMD Flash Demonstration Program.

9.2. Function Call Definitions

We are following the simple form for describing all functions presented by BURAPI.

Function Name

Description of the functionality and notes.

Entry:	Register values what must be set before calling
Exit:	Register values returned by function
Uses:	Which registers are getting corrupted during the call

DSBX2Var

Sets DS:BX point to the BUR internal data area in RAM.

Entry:	none
Exit:	DS:BX – Pointer to the data area beginning
Uses:	none

CRLF

Display <CR><LF> on the selected terminal output. This function will output bytes 0Dh 0Ah to the currently selected serial device.

Entry:	none
Exit:	none
Uses:	flags

Delay

Do uncalibrated but sufficiently long delay. The delay is blocking (no execution will return to caller before delay is expired) but does not switch off interrupts.
The actual delay is performed by following code:

```
            mov   cx, 20
DlyLoop:
            in    al, 80h
            loop  DlyLoop
```

The I/O port 80h read is done for wasting reasonably longer amount of time than “NOP” would have been.

Entry:	none
Exit:	none
Uses:	flags

Parm2EDX

This function will parse the data placed at BUR variables area “CmdLine” position (see below) according to the parse type notification supplied as a parameter in AL. This function is used internally by BUR as ZfiX console command line parameters parser and made available for the user application when interaction through the console is performed. This function is also useful when user is implementing the new command for the console, since it helps to gather all necessary parameters without implementing the parser.

The data at CmdLine area must be zero-terminated and parameter parsing starts after first space (CHR(0x20)) encountered in command line string.

The parser is parsing only hexadecimal numeric values, so running the parser for the first parameter in string “command 1234” returns 1234h as a return.

Depending on value in AL the type checking is done for parameter as well. This means that if the byte value is desired (AL = 0) and parser finds larger numeric value than 0FFh at specific parameter position, it will automatically return with the carry flag cleared (NC).

The parser will ignore extra spaces and extra zeros (CHR(0x30)) at the beginning of the numbers.

Entry:	AL – parse type code: 0h – byte (range 0-FFh) 1h – word (range 0-FFFFh) 2h – dword (range 0-FFFFFFFFh) 3h – address (depending on flag “Addressing” at Variables area (see below)): If Addressing = 0 (real mode), then checks for type WORD:WORD (0-FFFFh:0-FFFFh) If Addressing = 1 (linear mode), then checks for doubleword (0-FFFFFFFFh)) CL – parameter number to parse (1 = first parameter etc.)
Exit:	EDX – Parsed value if CF is set to CY CF – CY = value parsed successfully – NC = failed parsing the value
Uses:	EDX, flags

ResetCRC

BUR has internal 16-bit CRC calculation routines. This routine is mapped as interrupt service, so the CRC calculation will be performed by setting the byte value to AL and then calling INT 17h.

This Interrupt will automatically update the CRC value at BUR internal variables area, named “YModemCRChi_C” and “YModemCRClo_C”, for upper and lower byte of 16-bit CRC respectively.

ResetCRC function is used for helping to reset that data area to zeros. This function is equivalent of manually resetting “YModemCRChi_C” and “YModemCRClo_C” to 00h.

Entry: none

Exit: none

Uses: flags

INT 17h (Interrupt Vector)

Although being an interrupt vector, it is a place to describe it's behavior, since this is the only interrupt vector provided by BUR as functional service to user application.

This interrupt is used for CRC updating at BUR variables data area, located at "YModemCRChi_C" and "YModemCRClo_C". For more information see description of BUR API function "ResetCRC".

The reason why CRC calculation algorithm is interrupt vector rather than callable service was the fact that all BUR functions are residing inside ROM area. Code fetch from ROM area is very slow and since the CRC calculation is done for each byte received, it will slow down the entire data processing significally. To avoid that, BUR is caching CRC calculation (and some aggressively used internal functions) from ROM to RAM and mapping them as interrupts.

Entry: AL – byte to include for CRC calculation

Exit: none

Uses: none

SeekParm

It is desired sometimes to use non-numeric parameters for command line parsing. To process those, it is necessary to determine the start offset of string tokens inside the command line.

SeekParm provides that functionality, allowing search within the command line string located at "CmdLine" at BUR internal variables area by parameter number. This function will skip all trailing spaces from command line string tokens, so pointer returned is actually pointing to the first non-space character of the parameter.

The returned value in SI is pointer inside "CmdLine" field at BUR variables area, so the actual string location in memory is:

0000:[Variables.CmdLine + SI]

Entry: CL – parameter number (1 = first parameter etc.)

Exit: SI – offset to parameter within CmdLine when CF is CY
CF – CY = parameter seeked successfully
– NC = parse error

Uses: SI, flags

SerOut8

Displays 8-bit hexadecimal numeric value on active serial console.

The active serial console is selected at BUR variables area at position "SerMode" where
0 = no serial output
1 = COM1
2 = Z-tag LED pins.

The number is displayed in hexadecimal format "as is", so no leading zeros or type identifiers are added. Value 1Ah therefore appears "1A" on serial console.

Entry: AL – value to display

Exit: none

Uses: flags

SerOut16

Displays 16-bit hexadecimal numeric value on active serial console. See comments for the function "SerOut8".

Value 12ABh appears as "12AB" on console.

Entry: AX – value to display

Exit: none

Uses: flags

SerOut32

Displays 32-bit hexadecimal numeric value on active serial console. See comments for the function “SerOut8”.

Value 1234ABCDh appears as “1234ABCD” on console.

Entry:	EAX	– value to display
Exit:	none	
Uses:	flags	

SerOutBits

Displays 8-bit numeric value on active serial console as bit pattern, starting from MSB. See comments for the function “SerOut8”.

Value 5Ah appears as “01011010” on console.

Entry:	AL	– value to display
Exit:	none	
Uses:	flags	

SerSend

Transmits string to the active serial console. See comments for the function “SerOut8”.

The string can be either zero-terminated or with the specified length.

Entry:	ES:DI	– string to display
	CX	– string length (CX = 0 if string is 0-terminated)
Exit:	none	
Uses:	flags	

SerSend2

Transmit character to active serial console. See comments for the function “SerOut8”.

The symbol can be any ASCII character between 0 and 255. No character translation is performed, all data will be transmitted to serial port “as is”.

Entry:	AL	– character to display
Exit:	none	
Uses:	flags	

SerRec

Receive character from active serial console input with no character waiting.

The active serial console is selected at BUR variables area at position “SerMode” where:

0 = no serial output
1 = COM1
2 = Z-tag interface.

The serial communication on BUR is interrupt-driven, so the SerRec function returns byte from the receive buffer located at BUR variables area position “CircBuf”. When buffer is empty, this function will return with ZF set to Z. Otherwise ZF is set to NZ and character is moved from receive buffer to AL.

Receive buffer is 128 bytes long, so host can send 128 symbols without loss before BUR application starts receiving.

Entry:	none
Exit:	AL – symbol received if ZF is NZ
	ZF – NZ = character received
	– Z = no character received
Uses:	AX, flags

SerRecWait

SerRecWait is functionally identical to the SerRec but waits for character to appear into receive buffer for 220ms if receive buffer is empty.

Entry: none

Exit: AL – symbol received if ZF is NZ
ZF – NZ = character received
– Z = no character received

Uses: AX, flags

Addr2Linear

Helper function for converting real-mode address to linear address.

The function behaves differently depending on flag “Addressing” at BUR variables area. When flag indicates 1h (linear addressing), the function will simply pass EDX to EDI. When flag indicates 0h (real mode translation), the address translation is performed.

The actual translation is performed by following routine:

```
mov         edi, edx
      cmp   byte ptr ds:[bx.Address-
ing], 1                      ; EDX
was linear?
      jz    Converted

      xor   di, di
      shr   edi, 12
      shl   edx, 16
      shr   edx, 16                 ;
clear high 16 bits in EDX
      add   edi, edx                ;
edi is now 20-bit address Converted:
```

Entry: EDX – value to convert. Bits 31..16 = real-mode segment address
Bits 15..0 = real-mode offset address

Exit: EDI – linear memory location (20-bit if converted from real mode)

Uses: EDI, flags

YModemGetHeader

BUR has built-in Ymodem transmission protocol for transferring files over serial link. This functionality is also made available as public function for using at BUR applications.

Ymodem support functions are made of two parts: YmodemGetHeader and YmodemGet-Data.

YmodemGetHeader initiates the receive procedure by sending poll characters (‘C’) to the serial line. After each character it will check for response symbol for about 1.5 seconds and re-send the poll character if no response from host is received.

If host sends escape character (0x1B), the YmodemGetHeader function will cancel and exit with error (CF is set to NC).

When host responds to poll character with either STX (0x2) or SOH (0x1), the transmission procedure continues with receiving the packet header information from host.

During the header receive YmodemGetHeader will automatically, according to transmitted header, fill in the values at BUR variables area for fields YmodemHdrByte, YmodemFileSize, YModemCRChi, and YModemCRClo.

Entry: none

Exit: CF – CY = header information received OK
– NC = error receiving header (error or cancelled by user)

Uses: flags

YModemGetData

This function is second of Ymodem data receiveing functions. It pairs with YmodemGetHeader and can be called only after YmodemGetHeader has been executed successfully.

This function will receive all the actual data records of the file transferred by Ymodem. Data is received as “bit-bucket”, so all the fetched data is simply stored in memory starting from specified start address. No file-names are processed and can not be retrieved by BUR application.

After receiving is complete, YmodemGetHeader must be executed again to check for batch transfers. If YmodemGetHeader returns with no-error condition, YModemGetData must be re-executed to receive next transmission in batch.

Note also that YModemGetData takes linear address as pointer to a receive buffer. This is originated from the fact that YModemGetData is able to store data anywhere within 4 Gb memory range because data storing to memory is always performed in linear mode. For using this function with real-mode addressing, addresses can be converted using the function “Addr2Linear” (see definition above).

Entry:	EDI – linear (32-bit) memory address for destination buffer
Exit:	EDI – points to the next byte after received data end CF – CY = data received OK – NC = error during transmission
Uses:	EDI, flags

The example skeleton of the receiving program utilizing the YModemGetXXXX functions is:

```
.
.
      Determine address for store buffer into EDX (See
      definition of “Addr2Linear” function for EDX layout)
.
.

ymodem_tryagain:
      call  YmodemGetHeader   ; initiate transmission
      jnc   yload_err         ; header error, so quit

      cmp   byte ptr ds:[bx.YModemHdrByte], 0
      jz    ymodem_tryagain   ; if empty header, try again

      call  Addr2Linear       ; convert EDX to linear EDI

      xor   cx, cx            ; reset batch transmissions counter

yload_nextfile:

      call  YModemGetData     ; Header was OK, get data
      jc    NextHeader        ; data was ok, go check if
                              ; it is a batch transfer and there
                                    ; is more to come..

      ; if we are here, there was error on receiving
```

```
      cmp   cx, 0             ; anything successfully received?
      jz    yload_err         ; nothing was received
      jmp   yload_suspicious  ; broken batch transmission
NextHeader:
      inc   cx                ; indicate that we had successful
                              ; transmission already
      call  YmodemGetHeader   ; request for next header
      jnc   yload_ok          ; no header found, so we quit
            ; if we are here, then we have batch transmission. However,
            ; some Ymodem terminals like to send empty batch packet
            ; as terminator, so exit batch if received header was empty.

      cmp   byte ptr ds:[bx.YModemHdrByte], 0
      jz    yload_ok

      cmp   dword ptr ds:[bx.YModemFileSize], 0
      jz    yload_ok

      ; it IS a valid batch transmission. Go get next part.

      jmp   yload_nextfile
yload_err:
      .
      .
      Do something for error condition
      .
      .

      jmp   yload_quit
yload_ok:
      .
      .
      Do something for OK condition
      .
      .
      jmp   yload_quit

yload_suspicious:
      .
      .
      Do something for incorrect batch transmission (may be
      protocol incompatibility or something else, the fact is
      that some files (count is in CX) did come through OK).
      .
      .
      jmp   yload_quit
yload_quit:
```

YmodemSendHeader

YmodemSendHeader and YmodemSend-Data are complements to the YModemGetXXXX functions but are used for transmitting data from ZFx86 side to the host.

The usage and operation logic is close to receiving functions but transmitting functionality does not support batch transmissions.

The YmodemSendHeader function can take optional destination filename as parameter. This parameter is fetched from the BUR variables area “CmdLine” entry as third parameter on command line. The first parameters are not important for YmodemSendHeader functionality, so in order to specify filename, CmdLine buffer may look as follows:

```
db    ‘x x x FILENAME.DAT’, 0
```

If the command line buffer does not contain third parameter (terminating zero is reached before third parameter parse), an automatic filename will be created based on the linear address of data source address in form of XXXXXXXX.DAT where XXXXXXXX represents the hexadecimal source address in linear mode, padded with leading zeros (i.e. ‘0001FAE8.DAT’).

A kind of awkward processing of user-defined filename is originated from the fact that YmodemSendHeader function is used internally by BUR as immediate processing function for console command “ysend <address> <length> [filename]” , in which case we have optional destination filename at command line buffer third position.

This function will wait for ‘C’ characters appear from host side and if received, send header packet. After header it will wait for ACK and receive additional 'C' between packet 0 and 1.

Entry: EDI – linear address for data source (see notes on YModemGetData). Used for automatic destination filename creation only.
EAX – length of the data to transmit

Exit: CF – CY = header transmitted OK
– NC = error transmitting header

Uses: flags

YmodemSendData

Actual data transmission function for executing after YmodemSendHeader. This function transmits all the data packets to the host.

Entry: EDI – linear address for data source (see notes on YModemGetData)
EAX – length of the data to transmit

Exit: CF – CY = data transmitted successfully
– NC = transmission failed

Uses: flags

The example skeleton for transmitting data from ZFx86 side to host is very simple:

```
.
.
      Determine address of source buffer into EDX (Seedefinition of
      “Addr2Linear” function for EDX layout)
      and set filename to BUR internal variables CmdLine area
      if specific filename is desired.
.
.

      call  Addr2Linear       ; EDX -> linear EDI

      call  YModemSendHeader
      jnc   ysend_failed      ; header transmission error, so quit
```

```
      call  YModemSendData
      jnc   ysend_failed      ; data tranismission error, so quit

      .
      .
      OK condition handling here
      .
      .

      jmp   ysend_quit
ysend_failed:
      .
      .
      Failure condition handling here
      .
      .
ysend_quit:
```

ZTCMDExec

Z-tag Command processor. This routine fetches and executes single record from Z-Tag dongle. The executable code is always placed starting from DownloadSegment (defined in BUR Variables area) offset 0.

The routine fetches command data out from dongle but assumes that Z-tag dongle is seeked past proper record signature, i.e. next byte fetched will be the command code (means that 7F F0 55 must be read before this function is called).

NOTE: This is a function used internally by BUR and is useful ONLY when there is a need for application program to take over the Z-tag records executing from dongle for some reason. Normally this is done by BUR itself and user programs do not have a need to modify this behavior.

Entry:	none
Exit:	none
Uses:	flags

ZT_Init

Initialize Z-tag interface. This function will perform Z-tag interface init together with checking the CLK to ACK loop and performs reset on Z-tag device. The function will exit with Z-tag device released from Reset condition (i.e. in a condition when reset is not active).

Note that after executing this function the connected Z-tag device will start delivering the data from offset 0, so this function must be used only when it is desired to fetch data from the beginning of the device.

Use ZTPrepareRead when Z-tag interface initialization is needed without resetting the device and loosing the track of the data.

Entry:	none
Exit:	none
Uses:	flags

ZTPrepareRead

Prepare Z-tag read accelerator for data reading.

After Z-tag interface initialization with ZT_Init the Z-tag interface is ready to fetch data. The data reading procedure is performed by accessing the register 80h in ZF-logic address space through index and data I/O-ports 218h/21Ah.

The data read from Z-tag by accelerator is always appearing at the byte read form register 21Ah/80h. It is, however, unnecessary to update the index register each time, so the read can be done in the following way (pseudo-code):

```
out   218h, 80h; select register 80h
in    21Ah  ; get first byte
in    21Ah  ; get second byte

etc…
```

The read process can therefore be possibly speeded up by writing index only once and then performing the multiple reads.

ZTPrepareRead is a function which selects the register 80h from ZF-logic space. The code that does the work for this function is:

```
mov   dx, 218h
mov   al, 80h
      out   dx, al
```

It is necessary to call this function before starting to use ZTRead function (see below) and each time after the ZF-logic registers are accessed between ZTRead calls. This function does not affect the behavior of the Z-tag device nor does it reset the data pointers at Z-Tag device.

Entry:	none
Exit:	none
Uses:	none

ZTRead

Unlike other commands defined in BURAPI, ZTRead is a function implemented inside BUR API itself, not as a pointer into BUR binary code. The ZTRead function allows data fetching from Z-tag interface utilizing ZFx86 internal Z-tag read accelerator. This simple accelerator fetches data bits automatically from Z-tag interface, speeding significally up the data transfer speed between Z-tag device and BUR application.

The read accelerator is essentially a two stage serial to parallel converter with read ahead.

For more information about accelerated Z-tag interface usage refer ZFx86 datasheet(s).

The ZTRead function will blink both LED signals on Z-tag interface (LED1 and LED2) with 1 sec period if continuous receive is performed.

Note that this function waits until the character is received, so it will block the operation of the BUR application when there is no data to receive.

The actual transfer rate is depending on the device connected to a Z-tag interface but up to 1.2 Mbit transfer rates are possible to achieve.

For more information about the behavior of ZTRead function please refer the BURAPI.ASM where the actual function code resides.

Entry:	none
Exit:	AL – character received
Uses:	flags

10. Signal Status After POST

Default control signal setting of various I/O devices are found in the following tables.

10.1. Access Bus

Table 10.1 Access Bus[a] Settings

Signal Name	Pin No.	Status After POST	Comment
scl_c	B12	Input with internal pull-up enabled	ACCESS BUS
sda	D13	Input with internal pull-up enabled*	ACCESS BUS

a. Not all pull ups are implemented.

10.2. Floppy Disk

Tables 10.2 and 10.3 contain Floppy Disk signal settings for the Floppy Disk in FFD active or Z-tag active operation mode.

10.2.1. FDD Active

Table 10.2 Floppy Disk (FDD) Settings

Signal Name	Pin No.	Status After POST	Comment
DSKCHG_N	J02	Input	Floppy Disk change. Door open.
INDEX_N	F03	Input	Floppy disk index pulse
DIR_N	G03	Output high	Floppy Head Direction. Low - in, High - out.
HDSEL_N	H01	Output high	Floppy Head select. Low indicates side zero.
RDATA_N	J04	Input	Floppy Read data.
DRV_N	F01	Output high	Floppy Selects floppy disk 0
MTR_N	F02	Output high	Floppy Selects motor driver 0
STEP_N	G04	Output high	Floppy Step head
TRK0_N	H03	Input	Floppy Track 0 indicator
WDATA_N	G02	Output high	Floppy Write Data (gated internally with wgate_n)
WGATE_N	G01	Output high	Floppy Write Enable Gate
WRPRT_N	H02	Input	Floppy Write protect

10.2.2. Z-tag Active

Table 10.3 Floppy Disk (Z-tag) Settings

Signal Name	Pin No.	Status After POST	Comment
ZPGPI1/CEN2_N	J02	Input	ZTAG general-purpose input. This can also be used as secondary chip enable for daisy-chained on-board FailSafe PROM programming.
INDEX_N	F03	Input	(NOT USED FOR ZTAG) (Floppy disk index pulse)
ZRST	G03	Output high	Must connect to ZTAG dongle through 1K resistor. This pin serves as RESET output for external device connected to a ZTAG.
ZLED1	H01	Output high	The ZTAG interface has two LED outputs capable of driving the status LED's of the device connected.
ZDIN	J04	Input	ZTAG data input (referred in a document mostly as DATA)
DRV_N	F01	Output high	(NOT USED FOR ZTAG) (Floppy Selects floppy disk 0)
MTR_N	F02	Output high	(NOT USED FOR ZTAG) (Floppy Selects motor driver 0)
ZLED2	G04	Output high	The ZTAG interface has two LED outputs capable of driving the status LED's of the device connected.
GPI0	H03	Input	ZTAG general-purpose input
ZCLK	G02	Output high	Must connect to ZTAG dongle through 1K resistor. This pin serves as CLK output when BUR fetches a code through ZTAG interface.
WGATE_N	G01	Output high	(NOT USED FOR ZTAG) (Floppy Write Enable Gate)
ZACK	H2	Input	Acknowledge input

10.3. GPIO

Use the GPIO1, 2, 3, 5, and 6 as the secondary IDE control signals when enabled in BIOS.

Table 10.4 GPIO[a] Settings

Signal Name	Pin No.	Status After POST	Comment
GPIO[0]	AD12	Input with internal pull-up enabled	GPIO (32KHz out)
GPIO[1]	AE11	Input with internal pull-up enabled or 2nd IDE DACK	GPIO / IDE_DACK1_N
GPIO[2]	AF11	Input with internal pull-up enabled or 2nd IDE IOW	GPIO / IDE_DIOW1_N
GPIO[3]	AD11	Input with internal pull-up enabled or 2nd IDE IOR	GPIO / IDE_DIOR1_N
GPIO[4]	AF10	Input with internal pull-up enabled	GPIO
GPIO[5]	AE10	Input with internal pull-up enabled or 2nd IDE DREQ	GPIO / IDE_DREG1
GPIO[6]	AD10	Input with internal pull-up enabled or 2nd IDE IORDY	GPIO / IDE_IORDY1
GPIO[7]	AF09	Input with internal pull-up enabled	GPIO

a. Not all pull-ups are implemented.

10.4. ISA

Table 10.5 ISA Pin Settings

Signal Name	Pin No.	Status After POST	Comment
DACK1_N	A14	output/bi-dir	ISA DACK1_N/ PCI GRNT2_N (BS9)
DRQ1	B14	Input/bi-dir	ISA DRQ1/ PCI REQ2_N (BS9)
ISA_ERR_N	AB3	input	ISA Fatal bus error

10.5. PS/2

Table 10.6 PS/2 Pin Settings

Signal Name	Pin No.	Status After POST	Comment
KCLOCK_C	C12	bi-dir with internal pull-up	Keyboard Clock
KDATA	A11	bi-dir with internal pull-up	Keyboard Data
KBLOCK_N	B11	bi-dir with internal pull-up	Keyboard Lock. Blocks keyboard input.
MCLK_C	D11	bi-dir with internal pull-up	Mouse Clock
MDAT	C11	bi-dir with internal pull-up	Mouse Data

10.6. PCI

Table 10.7 PCI Settings

Signal Name	Pin No.	Status After POST	Comment
PCICLK_C	U25	Input/output	PCI CLOCK (BS20)
IRQ9	D02	Input with internal pull-up	PCI_INT_ A
IRQ10	E04	Input with internal pull-up	PCI_INT_ B
IRQ11	D01	Input with internal pull-up	PCI_INT_ C
IRQ12	E03	Input with internal pull-up	PCI_INT_ D

10.7. LPT

Refer to BIOS settings.

Table 10.8 LPT Settings

Signal Name	Pin No.	Status After POST	Comment
ACK_N	K02	bi-dir-u	Printer acknowledge.
AUTOFD_N	L02	bi-dir-u	Printer auto-linefeed.
BUSY	K03	bi-dir-d	Printer busy.
STRB_N	M03	bi-dir-u	Printer Data valid strobe.
PD[0]	P03	bi-dir	Printer Data.
PD[1]	P01	bi-dir	Printer Data.
PD[2]	P02	bi-dir	Printer Data.
PD[3]	N01	bi-dir	Printer Data.
PD[4]	N02	bi-dir	Printer Data.
PD[5]	M01	bi-dir	Printer Data.
PD[6]	N03	bi-dir	Printer Data.
PD[7]	M02	bi-dir	Printer Data.
ERR_N	L01	bi-dir-u	Printer error.
INIT_N	L03	bi-dir-u	Printer Initialize printer.
SLCK	J03	bi-dir-d	Printer is selected.
PE_N	J01	bi-dir-ud	Printer Port paper end.
SLCTIN_N	K01	bi-dir-u	Printer Port select printer.

10.8. IR Control (COM2)

Table 10.9 IR Control Settings

Signal Name	Pin No.	Status After POST	Comment
CTS2_N	A06	Input	CTS2_N / IR control 3 – input
DCD2_N	B08	Input	DCD2_N / IR control 2 –output
DSR2_N	C08	Input	DSR2_N / IR control 1 – output
RTS2_N	C07	Output high	RTS2_N / IR control 0 – output

10.9. ZF Logic

Table 10.10 ZF Logic Settings

Signal Name	Pin No.	Status After POST	Comment
IO_CS0	B03	Output high	ZFLogic IO chip select.
IO_CS1	A02	Output high	ZFLogic IO chip select.
IO_CS2	A01	Output high	ZFLogic IO chip select.
IO_CS3	C03	Output high	ZFLogic IO chip select.
MEM_CS1	D05	Output high	ZFLogic Memory chip select *
MEM_CS2	A03	Output high	ZFLogic Memory chip select *
MEM_CS3	C04	Output high	ZFLogic Memory chip select *
MEM_CS0	B04	Output	ZFLogic Memory chip select * #
PWM	B05	Output low	ZFLogic Pulse width modulation
WDI	A04	Input with internal pull-up	ZFLogic Watchdog timer input. Re-start timer.
WDO	C05	Output low	ZFLogic Watchdog timer output. Time-out.

11. Phoenix BIOS Register Settings

This chapter contains Phoenix BIOS register settings, brief comments that may apply, and all bits extracted from the registers. It contains an overview of the North Bridge, and South Bridge registers discussed in detail in Chapters 3 and 4 of this manual.

The Phoenix BIOS version 1.03 was booted with default settings (CMOS cleared), except both IDE channels were enabled.

For booting MS Windows 98, default installation was used with BootGUI=0 modification in MSDOS.SYS.

11.1. North Bridge

11.1.1. Reset, Sampling, and Misc North Bridge Registers

Table 12.1 Reset, Sampling, and Misc North Bridge Registers

Register/ Bit	Name	Function	Value
100H	**RID**	**NB Revision ID Register**	**0002h**
3:0	Lambda ID	Lambda version ID.	2H
15:4	Reserved		All 0s
110H	**PR1**	**Programmable Region 1 Register**	**0000h**
2:0	PREG1S<2:0>	Programmable region 1 block size: Bit <2:0> Block size — Bit<2:0> Block size 000 32KB — 100 512KB 001 64KB — 1011 MB 010 128KB — 11X Reserved 011 256KB	0h
15:3	PREG1A<27:15>	Programmable region 1 starting address: The programmable region starting address must be a multiple of the block size.	000h
111H	**PR2**	**Programmable Region 2 Register**	**0000h**
2:0	PREG2S<2:0>	Programmable region 2 block size: Bit <2:0> Block size — Bit<2:0> Block size 000 32KB — 100 512KB 001 64KB — 101 1MB 010 128KB — 11X Reserved 0112 56KB	0h
15:3	PREG2A<27:15>	Programmable region 2 starting address: The programmable region starting address must be a multiple of the block size.	000h
112H	**PR3**	**Programmable Region 3 Register**	**0000h**
2:0	PREG3S<2:0>	Programmable region 3 block size: Bit <2:0> Block size — Bit<2:0> Block size 000 32KB — 100 512KB 001 64KB — 101 1MB 010 128KB — 11X Reserved 011 256KB	0h
15:3	PREG3A<27:15>	Programmable region 3 starting address: The programmable region starting address must be a multiple of the block size.	000h

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
113H	**PR4**	**Programmable Region 4 Register**	**0000**
2:0	PREG4S<2:0>	Programmable region 4 block size: Bit <2:0> Block size / Bit<2:0> Block size 000 32KB / 100 512KB 001 64KB / 1011 MB 010 128KB / 11X Reserved 011 256KB	0h
15:3	PREG4A<27:15>	Programmable region 4 starting address: The programmable region starting address must be a multiple of the block size	000h
114H	**PRC**	**Programmable Region Control Register**	**0000h**
1:0	PRGREG1_SEL<1:0>	Programmable region 1 select<1:0>: Bits<1:0> Function / Bits<1:0> Function 00 Disable / 10 non-cacheable 01 Reserved / 11 Reserved	0h
3:2	PRGREG2_SEL<1:0>	Programmable region 2 select<1:0>: Bits<1:0> Function / Bits<1:0> Function 00 Disable / 10 non-cacheable 01 Reserved / 11 Reserved	00b
5:4	PRGREG3_SEL<1:0>	Programmable region 3 select<1:0>: Bits<1:0> Function / Bits<1:0> Function 00 Disable / 10 non-cacheable 01 Reserved / 11 Reserved	00b
7:6	PRGREG4_SEL<1:0>	Programmable region 4 select<1:0>: Bits<1:0> Function / Bits<1:0> Function 00 Disable / 10 non-cacheable 01 Reserved / 11 Reserved	00b
15:8	Reserved		All “0”s
115H	**COR**	**Cacheability Override Register**	**0000h**
0	CACHE_OVR_A24	Cacheability Override A24: When set, all address with A<24> high is marked non-cacheable.This corresponds to addresses in the range X1000000h–X1FFFFFFh.	0b
1	CACHE_OVR_A25	Cacheability Override A25: When set, all address with A<25> high is marked non-cacheable. This corresponds to addresses in the range X2000000h–X3FFFFFFh.	0b
2	CACHE_OVR_A26	Cacheability Override A26: When set, all address with A<26> high is marked non-cacheable. This corresponds to addresses in the range X4000000h–X7FFFFFFh.	0b
3	CACHE_OVR_A27	Cacheability Override A27: When set, all address with A<27> high is marked non-cacheable. This corresponds to addresses in the range X8000000h–XFFFFFFFh.	0b

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
4	CACHE_OVR_A28	Cacheability Override A28: When set, all address with A<28> high is marked non-cacheable. This corresponds to addresses in the range 10000000h–1FFFFFFFh.	0b
5	CACHE_OVR_A29	Cacheability Override A29: When set, all address with A<29> high is marked non-cacheable. This corresponds to addresses in the range 20000000h–3FFFFFFFh.	0b
6	CACHE_OVR_A30	Cacheability Override A30: When set, all address with A<30> high is marked non-cacheable. This corresponds to addresses in the range 40000000h–7FFFFFFFh.	0b
7	CACHE_OVR_A31	Cacheability Override A31: When set, all address with A<31> high is marked non-cacheable. This corresponds to addresses in the range 80000000h–FFFFFFFFh.	0b
15:8	Reserved		All “0”s
117H	**BCR**	**Back-off Control Register**	**0000h**
1:0	NONPOST_RETRY_CNT<1:0>	Non-posted PCI cycle retry count<1:0> Bits<1:0> count Bits<1:0> count 00 3 10 11 01 7 11 15 These count are effective only when another PCI master is requestor is requesting the bus.Otherwise the cycles will continue to be retried forever	00b
2	NONPOST_RETRY_DIS	Disable PCI retry counter for non-posted cycle: 0 = Enable 1 = Disable	0b
3	Reserved		0b
5:4	POST_RETRY_CNCT<1:0>	Posted PCI cycle retry count<1:0> Bits<1:0> count Bits<1:0> count 00 3 10 11 01 7 11 15 Just like Bit[1:0], these bits are effective only when a PCI request is active.	00b
6	POST_RETRYCNT_DIS	Disable PCI retry counter for posted cycle: 0 = Enable 1 = Disable.	0b
7	Reserved		0b
8	RESET_CNT_ON_GNT RESET	retry counter on any bus master grant: 0 = not reset on grant 1 = reset on grant.	0b
9	HLD_RETRY_ON_REQ	Hold retry on any PCI Bus Master Request: 0 = initiate retry once been backoff 1 = initiate retry only after all pending PCI Bus Master requests have been serviced	0b
15:10	Reserved		All ‘0’s

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
118H	**SMMC**	**SMM Control Register**	**00A6h**
0	Reserved		0b
1	KDISSMMRAM	SMM RAM KEN disable: 1 = KEN# will be inactive(high) during access to SMM RAM 0 = KEN# will function normally within SMM RAM. **Should always be set a ‘1’, to disallow caching.**	1b
2	DIS23RMAP	Disable 20000h-3FFFFh remap to A0000h–BFFFFh physical memory in SMM mode: 0 = Enabled 1 = Disabled. **Note:** This bit can only be used while both L1 and L2 are disabled.	1b
3	FRCREMAP	Enables the SMM remapped address to be used in a non-SMM cycle. This is used during loading of the SMM code to the memory. It works in conjunction with bit 14 and 15 of this register, and they need to be in the correct state to allow the loading.	0b
5:4	SMM_DL_SEL[1:0]	SMM D0000h–D7FFFh select<1:0>: Bits<1:0> Function 00 XXXD0000h–XXXD7FFFh is not used as SMM space. 01 reserved 10 000D0000h–000D7FFFh is used as SMM space. (Remap to 000A000h–-000A7FFFh in physical DRAM space.) 11 1DFD0000h–-1DFD7FFFh is used as SMM space. (Remap to 000A0000h–-000A7FFFh in physical DRAM space.) **Note:** When programmed to 10, 000D0000h–-000D7FFFh will be automatically be set to non-cacheable.	10b
7:6	SMM_DH_SEL[1:0]	SMM D8000h-DFFFFh select<1:0>: Bits<1:0> Function 00 XXXD8000h–-XXXDFFFFh is not used as SMM space. 01 reserved 10 000D8000h–-000DFFFFh is used as SMM space. (remap to 000A8000h-000AFFFFh in physical DRAM space.) 11 1DFD8000h–-1DFDFFFFh is used as SMM space. (remap to 000A8000h–-000AFFFFh in physical DRAM space.) **Note:** When programmed to 10, 000D8000h–-000DFFFFh will be automatically bet set to non-cacheable.	10b

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
9:8	SMM_EL_SEL[1:0]	SMM E0000h—-E7FFFh select<1:0>: Bits<1:0> Function 00 XXXE000h—-XXXE7FFFh is not used as SMM space. 01 reserved 10 000E0000h—-000E7FFFh is used as SMM space. (remap to 000B0000h-000B7FFFh in physical DRAM space.) 11 1DFE0000h—-1DFE7FFFh is used as SMM space. (remap to 000B0000h—-000B7FFFh in physical DRAM space.) **Note:** When programmed to 10, 000E0000h—000E7FFFh will be automatically bet set to non-cacheable	00b
11:10	SMM_EH_SEL[1:0]	SMM E8000h—-EFFFFh select<1:0>: Bits<1:0> Function 00 XXXE8000h—-XXXEFFFFh is not used as SMM space. 01 reserved 10 000E8000h—-000EFFFFh is used as SMM space. (remap to 000B8000h-000BFFFFh in physical DRAM space.) 11 1DFE8000h—-1DFEFFFFh is used as SMM space. (remap to 000B8000h—-000BFFFFh in physical DRAM space.) **Note:** When programmed to 10, 000E8000h—000EFFFFh will be automatically bet set to non-cacheable.	00b
12	SWAP_23_MAP	Swap SMM 2/3 mapping: 0 = 2/3 will be mapped to A/B 1 = 2/3 will be mapped to B/A. Here 2/3 and A/B refer to the address bits 19-16. When 0 = 2XXXX access will be mapped to AXXXX and 3XXXX to BXXXX. When 1 = 2XXXX access will be mapped to BXXXX and 3XXXX to AXXXX	0b
13	SWAP_DE_MAP	Swap SMM D/E mapping: 0 = D/E will be mapped to A/B 1 = D/E will be mapped to B/A. Here again D/E and A/B refer to the address bits 19-16. When 0 = DXXXX access will be mapped to AXXXX and EXXXX to BXXXX. When 1 = DXXXX access will be mapped to BXXXX and EXXXX to AXXXX	0b
14	LDSMIHLDER	Load SMI handler into SMM RAM: 1 = Enable access to SMM RAM during normal cycle 0 = Disable access to SMM RAM during normal cycle.	0b

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
15	SMIHLDERLOCK	SMM RAM access in normal mode lock: This bit provides an option to lock bit 14 in a disabled state, thereby prohibiting any further access to SMM RAM from normal mode. This bit can only be written once. Reading a 0 from this bit indicates that bit 14 above is not locked. Reading a 1 from this bit indicates that bit 14 above is locked to disable state.	
119H	**PROC**	**Processor Control Register**	**006Bh**
0	KENEN	KEN enable: When low, KEN# will be de-asserted for all cycles. When high, KEN# will be asserted for all local memory cycles, except for cycles to local space which has been explicitly marked as non-cacheable via PRC and COR registers, or implied non-cacheable via SMMC register or SHADRC/SHADWC registers	1b
1	L1WBEN	L1 write-back enable: This bit should normally reflect the state of the L1 cache inside the processor. This bit to determine how an access to shadow ROM is handled. If it is a ‘0’, i.e. write-through state, a read to shadow ROM is allowed to be cached (KENEN =1) and a write to shadow ROM causes an invalidation cycle back to the 486. If it is a ‘1’, i.e. write-back state, during a read KENnn is returned HIGH, non-cacheable, and on a write no Invalidation takes place, as data cannot be in the L1 cache.	1b
2	Reserved		0b
3	LINEARBURST	Enable linear burst: 0 = Toggle burst, 1 = Linear burst. This bit should be set to the correct value before the L1 and L2 cache is turn on. This bit determines L2 as well as S/DRAM burst sequencing. **Only Linear will be supported (Fixed to a ‘1’ in hardware)**	1b
4	Reserved		0b
5	WRFIFO_EN	Enable write FIFO: 0 = Disable, 1 = Enable. This bit controls buffer depth of CPU-PCI write buffer and CPU-SDRAM write buffer.When disabled, CPU-PCI depth = 2 CPU-SDRAM depth = 1 When enabled, CPU-PCI depth is controlled by PCIWFIFOC register CPU-SDRAM depth is controlled by WFIFOC registers Note this bit affects the depth of the write buffer only, other characteristics of the write buffers are still controlled by the respective bits in the WFIFOC and PCIWFIFOC.	1b
6	DIS_PSLOCK	Disable PSLOCK – When set to ‘1’, will disable the PSLOCK signal from being used.	1b
7	FLUSH	Setting this bit from 0->1, will cause the core to set FLUSHnn pin to the 486 to go LOW for 1 clock. To do another flush this bit should be reset to ‘0’ and then set to a ‘1’.	0b
8	DIS_FPUCLR_BY_F0	Disable clearing of FPU error by writing to IO port F0h: 0 = Enable clearing 1 = Disable clearing.	0b

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
9	DIS_FPUCLR_BY_F1	Disable clearing of FPU error by writing to IO port F1H: 0 = Enable clearing 1 = Disable clearing.	0b
10	WRM_RST	Warm Reset – When a ‘1’ is written to this bit, a warm reset sequence will be initiated. It works same as FLUSH bit i.e. to do another warm reset, this bit should be cleared to ‘0’ and then set to a ‘1’	0b
11	A20M	Address 20 Mask – Used for DOS compatibility.	0b
15:12	Reserved		All 0’s
11AH	**WFIFOC**	**Write FIFO Control Register**	**0220h**
2:0	FIFOD<2:0>	Write FIFO depth Bits<2:0> FIFO depth / Bits<2:0> FIFO depth 000 8 dwords / 100 4 dwords 001 7 dwords / 101 3 dwords 010 6 dwords / 110 2 dwords 011 5 dwords / 111 1 dword	000b
3	DRMRDREODEREN	DRAM read re-ordering enable: 0 = Disable 1 = Enable. When this bit is set and when there’s pending DRAM write cycle, a DRAM read operation will be performed before a DRAM write operation.	0b
4	Reserved		
5	CPU&EM_DRAM_ARBITRATION	CPU/External Master DRAM Arbitration Priority Scheme: Bits5 Function 0 = CPU has NO Write Buffer access while Ext. Master is accessing DRAM 1 = CPU has Write Buffer access while Ext. Master is writing to DRAM	1b
7:6	Reserved		00b
11:8	RD2WR_LAT<3:0>	Read to write pending latency<3:0>: These bits indicate the number of clocks to delay before switching from a read cycle back to pending cycles in the write buffer.. Bits<3:0> # of CPUCLKs / Bits<3:0> # of CPUCLKs 0H reserved / 8H 8 1H 1 / 9H 9 2H 2 / AH 10 3H 3 / BH 11 4H 4 / CH 12 5H 5 / DH 13 6H 6 / EH 14 7H 7 / FH 15	2h
13:12	Reserved		00b
15:14	WR_LATENCY<1:0>	DRAM write latency<1:0>: These bits indicate the number of processor clocks write are stalled before being issued to DRAM controller. Bits<1:0> number of clocks / Bits<1:0> number of clocks 00 1 / 10 3 01 2 / 11 4	00b

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
11BH	**PCIC**	**PCI Control Register**	**0217h**
0	CPU2PCI_BURST_EN	CPU to PCI burst enable: When 0, Lambda will only do single PCI transfer when CPU is accessing PCI bus. When 1, Lambda will try to burst to PCI when CPU is master.	1b
1	PCIM2DRM_BRST_EN	PCI master to DRAM burst enable: When 0, Lambda will only do single DRAM transfer when PCI master is accessing DRAM. When 1, Lambda will try to do a burst to DRAM when PCI master is accessing.	1b
2	BM_BURSTRD_ALWYS	PCI master read prefetch always: When 0, only PCI read line or PCI read multiple will start a burst read request. For PCI single read, a burst read request will be initiated only after the first data phase is completed and PCI master indicated that it wants a burst access. When 1, any PCI read cycle will initiate a burst read request. **Note:** In order to enable this feature, bit[1] must be enabled.	1b
3	DISC_ON_LN_BOUNDARY	Disconnect from PCI master on CACHE line boundary: 0 = No disconnect 1 = Disconnect.	0b
4	EN_PCI_FAST_DECDE	Enable PCI fast decode when accessing DRAM: 0 = Disable 1 = Enable.	1b
5	EN_ADCBE_FLT_IDLE	Enable AD/CBE/PAR float when PCI is idle and CPU is the bus master: 0 = Disable float 1 = Enable float.	0b
6	DIS_RESOURCE_LOCK	Disable Resource Lock: 0 = Enable 1 = Disable. If set, a LOCK cycle on the PCI bus will not cause LNB to lock itself. **Note:** When EN_BUS_LOCK(bit 7) is set to 1, this bit is ignored.	0b
7	EN_BUS_LOCK	Enable Bus Lock: 0 = Disable 1 = Enable. When enabled, GNT# to a particular PCI master remains asserted until LOCK# is deasserted. **Note:** When this bit is set to 1, DIS_RESOURCE_LOCK(BIT 6) is ignored.	0b

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
8	LCK_RDBURST_EN	Enable the locking of PCI bus during a 64-bit processor read access to the PCI bus. 0 = Disable 1 = Enable. **Note:** Bit not used as CPU cannot do a burst read on PCI. Refer to issue 14 in the Issues section.	0b
9	CNFCY_AD_STEP_DIS	PCI configuration cycle address stepping disable: 0 = Enable, 1 = Disable.	1b
10	BM_DONE_DIS	Disable the waiting of PCI master cycle is done before starting processor initiated PCI cycle. 0 = Enable 1 = Disable. When enabled, Lambda's PCI master controller will not start until, 1) PCI master initiated cycle is done, 2) PCI master write buffer is empty, and 3) PCI master read prefetch is done.	0b
11	Reserved		0b
12	Reserved		0b
15:12	Reserved		All 0's
11CH	**CSA**	**Clock Skew Adjust Register**	**0000h**
2:0	Reserved		000b
5:3	SDRAMCLK_SKEW	There three bits control the skew between the core clock and the SDRAM clock. 000 = Nominal 001 = Minus 1 nS 010 = Minus 2 nS 100 = Plus 1 nS 101 = Plus 2 nS Rest = Default to Nominal	000b
15:6	Reserved		All 0's
11DH	**SNOOPCTRL**	**BUS MASTER And Snooping Control Register**	**23D0h**
0	DIS_SNOOP	Disable Snooping: 0 = Enable snoop 1 = Disable snoop.	0b
1	DIS_CHK_HITM	Disable the check of HITM#: 0 = Enable the checking of HITM# during snooping. 1 = Disable the checking of HITM# during snooping. In either case, the L1 cache may be invalidated with INVAL signal	0b
3:2	CK_HITM_WS<1:0>	Check HITM# wait state: Bits<1:0> Check HITM# wait state 00 2 clock after EADS# is deasserted. Others – Reserved	00b

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
4	ADP_PREF_DIS	Adaptive Prefetch Disable: 0 = Enable 1 = Disable When enabled, Lambda monitors the average burst transfer length of a master access and then controls the number of speculative prefetches accordingly.	1b
5	Reserved		0b
6	DIS_WB_MERGE	0 = Merge CPU/L2 Write-back data with External Master writes. The External Master's valid bytes overwrite the data cast-out from the CPU/L2 and subsequently limit the bandwidth requirements to the s/dram. 1 = Do not merge External Master write data bytes with CPU/L2 write-back cycle.	1b
7	DIS_EM_PREFETCH	0 = Prefetch next "cache" line on EM accesses, and store in prefetch buffer 1= Disable prefetch logic for External Masters (CPU clock based)	1b
8	DIS_CONCURRENCY	CPU/PCI master concurrency disable: 0 = Enable 1 = Disable	1b
9	FAST_TRDY	0 = Normal TRDY# timings 1 = Enable Fast TRDY# timings to EM. Improves path from prefetch data ready (from CPU write-back, yes we snarf-see bit 10, or from DRAM)	1b
10	DIS_BUS_SNARF	0 = Snarf CPU write-back data and return it to the requesting External Master (read), concurrent with it's retirement into DRAM. 1 = Disable bus snarfing and create 2nd cycle to get data after the write-back has retired it to DRAM.	0b
11	FORCE_DRM_PM_PCIM	Force DRAM page miss in bus master cycle: 0 = Disable force page miss mode 1 = Enable force page miss mode.	0b
12	Reserved		0b
13	DISPCIM_ELY_DRM_CY	Speculatively start DRAM cycle for PCI External Master Request and restart it in the event of an L1 or L2 write-back: 0 = Enable 1 = Disable.	1b
14	Reserved		0b
15	ENPCIM_SHADOWRAM	1 = Claim cycle for PCI Master access to 000C0000-000F0000 region. 0 = Do not Claim cycle for PCI Master access to ROM space (shadowed RAM). **Note:** All DRAM Write/Read protect bits are still applicable	0b

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
11EH	**ARBCTRL**	**Arbiter Control Register** **See also PCI register section REG 41H**	**0000h**
0	REQa_slot0	0 = Disable Slot 0 for REQa 1 = Enable Slot 0 for REQa	0b
1	REQa_slot1	0 = Disable Slot 1 for REQa 1 = Enable Slot 1 for REQa	0b
2	REQb_slot2	0 = Disable Slot 2 for REQb 1 = Enable Slot 2 for REQb	0b
3	REQa_slot3	0 = Disable Slot 3 for REQa 1 = Enable Slot 3 for REQa	0b
4	REQpci_slot4	0 = Disable Slot 1 for 2nd Arbitration of PCI 1 = Enable Slot 1 for 2nd Arbitration of PCI	0b
5	PC98_support	0 = V4REQ#/V4GNT# pair treated as such. 1 = The V4REQ#/V4GNT# pair is treated as PHOLD#/PHLDA#	0b
6	SIO_HIPRI	0 = Fair Arbitration between V3 & V4 REQ# pins 1 = Always give priority to V3 REQ#	0b
7	Reserved		0b
15:8	CPU_BUSY_TIMER	Number of PCI clocks that the CPU can *own* the PCI bus before it is preempted by any other active requesters. 00h = Never preempt the CPU 01h = 4 clks 02h = 8 clks ...FFh = 1024 clks	00h
11FH	**DOCKC**	**Docking Control Register**	**0000h**
0	TS_DOCK_SIGS	Tri-state DOCK_PCIRST# and DOCK_PCICLK in normal operating mode. 0 = drive out DOCK_PCIRST# and DOCK_PCICLK. 1 = tri-state DOCK_PCIRST# and DOCK_PCICLK.	0b
1	Reserved		0b
2	DOCK_RST_DASSERT	Deassert DOCK_PCIRST#: 0 = assert DOCK_PCIRST# in a low state. 1 = DOCK_PCIRST# will follow the state of PCIRST#.	0b
3	DOCK_CLK_EN	Enable DOCK_PCICLK to toggle: 0 = force DOCK_PCICLK low. 1 = Enable DOCK_PCICLK to follow the state of PCICLK.	0b
4	DISBSERCK_DOCK_HOLD_ACK	Disable the checking of the bser bit 'DOCK_HOLD_ACK' 0 = Enable checking 1 = Disable.	0b
14:5	Reserved		All 0's

Table 12.1 Reset, Sampling, and Misc North Bridge Registers (cont.)

Register/ Bit	Name	Function	Value
15	DOCKED	System DOCKed indication: This bit is used when the PCI floating method is used. This bit is set to 1 by SMI handler when the system is docked. This bit will be reset to 0 when it's previously set to 1 and DOCK_REQ# is deasserted to indicate the completion of the un-dock procedure.	0b
120H	**PCIWFIFOC**	**PCI Write FIFO Control Register**	**0000h**
2:0	FIFOD[2:0]	PCI Write FIFO depth. This FIFO is for CPU-PCI write transfers. Bits<2:0> FIFO depth Bits<2:0> FIFO depth 000 16 dwords 100 8 dwords 001 14 dwords 101 6 dwords 010 12 dwords 110 4 dwords 011 10 dwords 111 2 dword	000b
3	Reserved		0b
4	Reserved		0b
5	PCI_BM_FREERUNMODE	PCI master write buffer PCI entry count free running mode bit. Transfer loop which copies CPU clocked write buffer entry count to PCI clocked entry count normally operates in an on-demand mode. This forces a free running mode which update the PCI every 6 or 8 CPU clocks (see slow transfer bit below).	0b
6	PCI_BM_SLOWRUNMODE	PCI master write buffer PCI entry count slow transfer mode. Increases transfer loop period from 6 CPU clocks to 8 CPU clocks. Transfer loop period is how often the PCI side entry count is updated from the CPU entry count. 0 = 6 CPU clocks. 1 = 8 CPU clocks.	0b
8:7	Reserved		00b
11:9	Reserved		All 0's
15:12	Reserved		0000b

11.1.2. DRAM Registers

Table 12.2 DRAM Registers

Register/ Bit	Name	Function	Value
200H	**SHADRC**	**Shadow RAM Read Enable Control Register**	**0FF03h**
0	LMEMRDEN0	Local memory C0000h-C3FFFh read enable: 0 = Disable, 1 = Enable.	1b
1	LMEMRDEN1	Local memory C4000h-C7FFFh read enable: 0 = Disable, 1 = Enable.	1b
2	LMEMRDEN2	Local memory C8000h-CBFFFh read enable: 0 = Disable, 1 = Enable.	0b
3	LMEMRDEN3	Local memory CC000h-CFFFFh read enable: 0 = Disable, 1 = Enable.	0b
4	LMEMRDEN4	Local memory D0000h-D3FFFh read enable: 0 = Disable, 1 = Enable.	0b
5	MEMRDEN5	Local memory D4000h-D7FFFh read enable: 0 = Disable, 1 = Enable.	0b
6	LMEMRDEN6	Local memory D8000h-DBFFFh read enable: 0 = Disable, 1 = Enable	0b
7	LMEMRDEN7	Local memory DC000h-DFFFFh read enable: 0 = Disable, 1 = Enable.	0b
8	LMEMRDEN8	Local memory E0000h-E3FFFh read enable: 0 = Disable, 1 = Enable.	1b
9	LMEMRDEN9	Local memory E4000h-E7FFFh read enable: 0 = Disable, 1 = Enable.	1b
10	LMEMRDEN10	Local memory E8000h-EBFFFh read enable: 0 = Disable, 1 = Enable.	1b
11	LMEMRDEN11	Local memory EC000h-EFFFFh read enable: 0 = Disable, 1 = Enable	1b
12	LMEMRDEN12	Local memory F0000h-F3FFFh read enable: 0 = Disable, 1 = Enable.	1b
13	LMEMRDEN13	Local memory F4000h-F7FFFh read enable: 0 = Disable, 1 = Enable.	1b
14	LMEMRDEN14	Local memory F8000h-FBFFFh read enable: 0 = Disable, 1 = Enable.	1b
15	LMEMRDEN15	Local memory FC000h-FFFFFh read enable: 0 = Disable, 1 = Enable.	1b
201H	**SHADWC**	**Shadow RAM Write Enable Control Register**	**0000h**
0	LMEMWREN0	Local memory C0000h-C3FFFh write enable: 0 = Disable, 1 = Enable.	0b
1	LMEMWREN1	Local memory C4000h-C7FFFh write enable: 0 = Disable, 1 = Enable.	0b
2	LMEMWREN2	Local memory C8000h-CBFFFh write enable: 0 = Disable, 1 = Enable.	0b
3	LMEMWREN3	Local memory CC000h-CFFFFh write enable: 0 = Disable, 1 = Enable.	0b
4	LMEMWREN4	Local memory D0000h-D3FFFh write enable: 0 = Disable, 1 = Enable.	0b

Table 12.2 DRAM Registers (cont.)

Register/ Bit	Name	Function	Value
5	LMEMWREN5	Local memory D4000h-D7FFFh write enable: 0 = Disable, 1 = Enable.	0b
6	LMEMWREN6	Local memory D8000h-DBFFFh write enable: 0 = Disable, 1 = Enable.	0b
7	LMEMWREN7	Local memory DC000h-DFFFFh write enable: 0 = Disable, 1 = Enable.	0b
8	LMEMWREN8	Local memory E0000h-E3FFFh write enable: 0 = Disable, 1 = Enable.	0b
9	LMEMWREN9	Local memory E4000h-E7FFFh write enable: 0 = Disable, 1 = Enable.	0b
10	LMEMWREN10	Local memory E8000h-EBFFFh write enable: 0 = Disable, 1 = Enable.	0b
11	LMEMWREN11	Local memory EC000h-EFFFFh write enable: 0 = Disable, 1 = Enable.	0b
12	LMEMWREN12	Local memory F0000h-F3FFFh write enable: 0 = Disable, 1 = Enable.	0b
13	LMEMWREN13	Local memory F4000h-F7FFFh write enable: 0 = Disable, 1 = Enable.	0b
14	LMEMWREN14	Local memory F8000h-FBFFFh write enable: 0 = Disable, 1 = Enable.	0b
15	LMEMWREN15	Local memory FC000h-FFFFFh write enable: 0 = Disable, 1 = Enable.	0b
202H	**N_B0C**	**Bank 0 Control Register**	**0000h**
7:0	B0A<27:20>	Bank 0 starting address <27:20>	00h
8	Reserved		0b
11:9	B0S<2:0>	Bank 0 DRAM size Bits<2:0> DRAM bank size; Bits<2:0> DRAM bank size 000 2MB; 100 32MB 001 4MB; 101 64MB 010 8MB; 110 Reserved 011 16MB; 111 Reserved	000b
14:12	COLADR0<2:0>	Number of column address bits for Bank 0<2:0> Bits<2:0> Column address; Bits<2:0> Column address 000 8 bits; 001 9 bits 0101 0 bits; all others Reserved	000b
15	Reserved		0b
204H	**N_B0TC**	**Bank 0Timing Control Register**	**0FC71h**
1:0	B0_TRP	SDRAM Pre-charge cmd to ACT cmd Bits<1:0> Time; Bits<1:0> Time 00 Reserved; 10 3T 01 2T; 11 4T	01b
3:2	B0_TRC	SDRAM ACT cmd to ACT cmd (same bank) Bits<3:2> Addr. hold time; Bits<3:2> Addr. hold time 00 6T; 10 8T 01 7T; 11 9T This field is actually not used by the hardware. TRC is fixed to 9T.	00b

Table 12.2 DRAM Registers (cont.)

Register/ Bit	Name	Function	Value
6:4	Reserved		111b
7	B0_CAS_LATCY	SDRAM CAS Latency: 0 = 2T, 1 = 3T Should be same as what is programmed in SDRAM via SDRAMMPR.	0b
8	B0_TRCD	SDRAM ACT cmd to R/W cmd delay: 0 = 2T, 1 = 3T	0b
9	B0_TCCD	SDRAM R/W cmd to R/W cmd: 0 = 1T, 1 = 2T	0b
15:10	Reserved		All ‘1’s
205H	**N_B1C**	**Bank 1 Control Register**	**0000h**
7:0	B1A<27:20>	Bank 1 starting address <27:20>	00h
8	Reserved		0b
11:9	B1S<2:0>	Bank 21DRAM size Bits<2:0> DRAM bank size Bits<2:0> DRAM bank size 000 2MB 100 32MB 001 4MB 101 64MB 010 8MB 110 Reserved 011 16MB 111 Reserved	000b
14:12	COLADR1<2:0>	Number of column address bits for Bank 2<2:0> Bits<2:0> Column address Bits<2:0> Column address 000 8 bits 001 9 bits 010 10 bits all others Reserved	000b
15	Reserved		0b
207H	**N_B1TC**	**Bank 1 Timing Control Register**	**0FC71h**
1:0	B1_TRP	SDRAM Pre-charge cmd to ACT cmd Bits<1:0> Time Bits<1:0> Time 00 Reserved 10 3T 01 2T 11 4T	01b
3:2	B1_TRC	SDRAM ACT cmd to ACT cmd (same bank) Bits<3:2> Addr. hold time Bits<3:2> Addr. hold time 00 6T 10 8T 01 7T 11 9T This field is actually not used by the hardware. TRC is fixed to 9T.	00b
6:4	Reserved		111b
7	B1_CAS_LATCY	SDRAM CAS Latency: 0 = 2T, 1 = 3T Should be same as what is programmed in SDRAM via SDRAMMPR.	0b
8	B1_TRCD	SDRAM ACT cmd to R/W cmd delay: 0 = 2T 1 = 3T	0b
9	B1_TCCD	SDRAM R/W cmd to R/W cmd: 0 = 1T 1 = 2T	0b
15:10	Reserved		All ‘1’s
208H	**N_B2C**	**Bank 2 Control Register**	**1800h**
7:0	B2A<27:20>	Bank 2 starting address <27:20>	00h
8	Reserved		0b

Table 12.2 DRAM Registers (cont.)

Register/ Bit	Name	Function	Value
11:9	B2S<2:0>	Bank 2 DRAM size Bits<2:0> DRAM bank size Bits<2:0> DRAM bank size 000 2MB 100 32MB 001 4MB 101 64MB 010 8MB 110 Reserved 011 16MB 111 Reserved	100b
14:12	COLADR2<2:0>	Number of column address bits for Bank 4<2:0> Bits<2:0> Column address Bits<2:0> Column address 000 8 bits 001 9 bits 010 10 bits all others Reserved	001b
15	Reserved		0
20AH	**N_B2TC**	**Bank 2 Timing Control Register**	**0FC71h**
1:0	B2_TRP	SDRAM Pre-charge cmd to ACT cmd Bits<1:0> Time Bits<1:0> Time 00 Reserved 10 3T 01 2T 11 4T	01b
3:2	B2_TRC	SDRAM ACT cmd to ACT cmd (same bank) Bits<3:2> Addr. hold time Bits<3:2> Addr. hold time 00 6T 10 8T 01 7T 11 9T This field is actually not used by the hardware. TRC is fixed to 9T.	00b
6:4	Reserved		111b
7	B2_CAS_LATCY	SDRAM CAS Latency: 0 = 2T, 1 = 3T Should be same as what is programmed in SDRAM via SDRAMMPR.	0b
8	B2_TRCD	SDRAM ACT cmd to R/W cmd delay: 0 = 2T 1 = 3T	0b
9	B2_TCCD	SDRAM R/W cmd to R/W cmd: 0 = 1T 1 = 2T	0b
15:10	Reserved		All ‘1’s
20BH	**N_B3C**	**Bank 3 Control Register**	**1820h**
7:0	B3A<27:20>	Bank 3 starting address <27:20>	20h
8	Reserved		0b
11:9	B3S<2:0>	Bank 3 DRAM size Bits<2:0> DRAM bank size Bits<2:0> DRAM bank size 000 2MB 100 32MB 00 14MB 101 64MB 010 8MB 110 Reserved 01 116MB 111 Reserved	100b
14:12	COLADR3<2:0>	Number of column address bits for Bank 6<2:0> Bits<2:0> Column address Bits<2:0> Column address 000 8 bits 001 9 bits 010 10 bits all others Reserved	001b
15	Reserved		0b

Table 12.2 DRAM Registers (cont.)

Register/ Bit	Name	Function	Value
20DH	**B_B3TC**	**Bank 3 Timing Control Register**	**0FC71h**
1:0	B3_TRP	SDRAM Pre-charge cmd to ACT cmd Bits<1:0> Time / Bits<1:0> Time 00 Reserved / 10 3T 01 2T / 11 4T	01b
3:2	B3_TRC	SDRAM ACT cmd to ACT cmd (same bank) Bits<3:2> Addr. hold time / Bits<3:2> Addr. hold time 00 6T / 10 8T 01 7T / 11 9T This field is actually not used by the hardware. TRC is fixed to 9T.	00b
6:4	Reserved		111b
7	B3_CAS_LATCY	SDRAM CAS Latency: 0 = 2T 1 = 3T Should be same as what is programmed in SDRAM via SDRAMMPR.	0b
8	B3_TRCD	SDRAM ACT cmd to R/W cmd delay: 0 = 2T 1 = 3T	0b
9	B3_TCCD	SDRAM R/W cmd to R/W cmd: 0 = 1T, 1 = 2T	0b
15:10	Reserved		All ‘1’s
20EH	**DCONF1**	**DRAM Configuration Register 1**	**0000h**
2:0	Reserved		000b
5:3	DRAM_INAT_TO	DRAM inactive time-out<2:0> Bits<2:0> Page size / Bits<2:0> Page size 000 never / 100 512T 00 18T / 101 reserved 010 32T / 110 reserved 011 128T / 111 immediate If SDRAM interface is inactive for the set amount of time, a Pre-charge cycle is generated at the end of timeout. Pre-charge cycle would de-activate the DRAM row which may be in “ACTIVE” state. Doing a Pre-charge cycle when SDRAM is in-active for a while will save power. But next memory cycle may be to the row which was just closed, will take a hit of running a RAS cycle causing lower performance.	000b
7:6	Reserved		00b
8	Reserved	Fixed to ‘0’ in hardware.	0b
9	Reserved.		0b
10	Reserved.		0b

Table 12.2 DRAM Registers (cont.)

Register/ Bit	Name	Function	Value
11	SDRAM_CMD_PIPELINE	SDRAM command pipeline enable: 0 = Disable the pipelining of SDRAM command cycle. 1 = Enable the pipelining of SDRAM command cycle. If enabled this bit would allow a new command to be sampled from the Writebuffer as son as the present cycle has been started. Since 486 does not pipeline cycles, there may not be much difference in cycle times whether this bit is a 1 or 0.	0b
12	EN_RELAX_SDRM_CMD_TMING	Enable relax timing for SDRAM command cycle. 0 = Ddisable 1 = Enable relax timing to the SDRAM command cycle. =1 MA, RAS, CAS and WE are asserted 1 clk before CS is asserted. **Note:** setting this bit to 1 will not affect performance but at the same time, allow the potential of not buffering MA, SDRAM_RAS, SDRAM_CAS, and WE# externally.	0b
13	EN_SDRM_PWRDN	Enable SDRAM power-down mode during mix DRAM type configuration: 0 = Disable 1 = Enable SDRAM to get into power-down mode during mix DRAM type configuration and when access is to anywhere other than SDRAM. **FW should always set this bit to ‘0’**	0b
14	FST_SDRM_RD_L2_EN	Enable fast SDRAM read access when L2 is on: 0 = Disable 1 = Enable. **FW should always set this bit to ‘0’**	0b
15	Reserved		0b
20FH	**DCONF2**	**DRAM Configuration Register 2**	**0C00h**
0	BANK0_16EN	Bank 0 enable: 0 = Disable, 1 = Enable. When enabled, bank 0 will operate as a 16bit bank.	0b
1	BANK1_16EN	Bank 1 enable: 0 = Disable, 1 = Enable. When enabled, bank 1 will operate as a 16bit bank.	0b
2	BANK2_16EN	Bank 2 enable: 0 = Disable, 1 = Enable. When enabled, bank 2 will operate as a 16 bit bank.	0b
3	BANK3_16EN	Bank 3 enable: 0 = Disable, 1 = Enable. When enabled, bank 3 will operate as a 32 bit bank.	0b
4	Reserved		0b
5	Reserved		0b
6	Reserved		0b
7	Reserved		0b
8	BANK0_32EN	0 = Bank 0 disabled (bit0 overides this) 1 = Bank 1 enabled as 32 bit bank (this bit overides bit 0)	0b

Table 12.2 DRAM Registers (cont.)

Register/ Bit	Name	Function	Value
9	BANK1_32EN	0 = Bank 0 disabled (bit1 overides this) 1 = Bank 1 enabled as 32 bit bank (this bit overides bit 1)	0b
10	BANK2_32EN	0 = Bank 0 disabled (bit2 overides this) 1 = Bank 1 enabled as 32 bit bank (this bit overides bit 2)	1b
11	BANK3_32EN	0 = Bank 0 disabled (bit3 overides this) 1 = Bank 1 enabled as 32 bit bank (this bit overides bit 3)	1b
15:12	Reserved		0h
211H	**DRFSHC**	**DRAM Refresh Control Register**	**0000h**
0	Reserved		0b
2:1	Reserved		00b
4:3	Reserved		00b
7:5	REFRPRD<2:0>	Refresh period: These bits determine the refresh period for local DRAM. Bits<2:0> Refresh period Bits<2:0> Refresh period 000 15us 001 15us 010 15us all others stopped 011 30us	000b
9:8	Reserved		00b
10	Reserved		0b
11	MANUAL_REFRESH	Manual refresh control: A 1-> 0->1 will generate a refresh cycle after 128 process clocks. Also, this bit will force normal refresh disabled while left at the 1 setting.	0b
13:12	Reserved		00b
15:14	Reserved		00b
213H	**SDRAMMPR**	**SDRAM Mode Program Register**	**0226h**
0	EN_SDRAM_CONFIG	Enable SDRAM MRS configuration cycle: 0 = Disable, 1 = Enable.	0b
2:1	SDRAM_BANK_CONFIG[1:0]	SDRAM bank configuration select<1:0>: SDRAM bank configuration select <1:0> programming options as follows: Bits<1:0> DRAM bank 00 Bank 0 01 Bank 1 10 Bank 2 11 Bank 3	11b
4:3	POWERON_SEQ[1:0]	SDRAM Power-on initialization sequence bits<1:0> Bits<1:0> Function 00 Normal 01 Pre-charge SDRAM bank specified by BANK_CONFIG[1:0] 10 Trigger Mode Program Register Command 11 Trigger CBR refresh cycle	00b

Table 12.2 DRAM Registers (cont.)

Register/ Bit	Name	Function	Value
15:5	WCBR_MA[11:1]	MA[0] comes from the SDRAMMPEX register as it is needed to handle 16 bit banks to do 8 burst cycles [2:0] set to ‘010’ corresponding to burst length of 4 for 32 bit banks set to ‘011’ corresponding to burst length of 8 for 16 bit banks **[3]** **always set to 0 = linear burst type** (Fixed in hardware) [6:4] 010=CAS Latency=2, 011=CAS Latency=3, Others Reserved [11:7]Always leave at ‘00000’ **Note:** MA[13:12] will be forced to 0 always during SDRAM configuration cycle	11h
214H	**SDRAMMPREX**	**SDRAM Mode Program Register**	**0000h**
0	WCBR_MA[0]	SDRAM Mode Register bit 0 used together with bits 11:1 defined earlier.	0b
15:1	Reserved		0h
218H	**Reserved**		
239H	**SDRAMSLEW**	**SDRAM Slew Control Register**	**1249h**
2:0	MD_DQM_SLEW	32 Bit Data and 4 Bit Mask Bus: MD[31:0], DQM[3:0] 000 = Force Tri-State 001 = 2*N-ch + 4*P-ch 010 = 3*N-ch + 6*P-ch 011 = 5*N-ch + 10*P-ch 100 = 4*N-ch + 8*P-ch 101 = 6*N-ch + 12*P-ch 110 = 7*N-ch + 14*P-ch 111 = 8*N-ch + 16*P-ch	001b
5:3	MA_SLEW	14 Bit Address Bus: BA[1:0], MA[11:0] Encoding same as for 2:0 bits	001b
8:6	RAS_CAS_SLEW	RAS and CAS Controls: RASnn and CASnn Encoding same as for 2:0 bits	001b
11:9	WE_SLEW	Write Enable: Wenn Encoding same as for 2:0 bits	001b
14:12	CS_SLEW	4 Chip Select: Csnn[3:0] Encoding same as for 2:0 bits	001b
15	DATA_BUS_HOLD	Data Bus Holder enabled when LOW	0b

11.1.3. Power Management Registers

Table 12.3 Power Management Registers

Register/ Bit	Name	Function	Value
300H	**CC**	**Clock Control Register**	**0000h**
15:0	Reserved		
3FFH	**CC2**	**Clock Control2 Register**	**0000h**
0	EN_STOP_CPU_CLK	Enables stopping of CPU clock during Suspend mode. Stopping the clock conserve much more power than mere putting CPU in Suspend mode. This bit is provided to let the BIOS decide to do that or not. For this bit to function bit1 should also be set.	0b
1	EN_SDRAM_CKE_RST	Enables resetting of SDRAM CKE input during Suspend mode.	0b
2	EN_STOP_SDRAM_CLK	Enables stopping of SDRAM CLK during Suspend mode. This different from SDRAMCLK disable bit in CSA, which disables the clock always. Whereas this bit is used only during Suspend mode. For this bit to function bit1 should also be set.	0b
3	EN_STOP_CORE_CLK	Enable stopping of core clock during Suspend mode. When set to ‘1’, it will stop clocks to most of the cores except clocks needed to detect end of suspend mode. For this bit to function bit1 should also be set.	0b
15:4	Reserved		All "0"s

11.1.4. PCI Configuration Registers

Table 12.4 PCI Configuration Registers

Register/ Bit	Name	Function	Value
00H	**VID**	**Vendor ID Register**	**100Bh**
15:0	VENDOR_ID	Vendor ID number. These bits are hard-wired.	100BH
02h	DID	Device ID Register	0023h
31:16	DEVICE_ID	Device ID number. These bits are hard-wired.	0023H
04H	**COMMD**	**Command Register**	**0006h**
0	Reserved		0
1	MEM_RESPOND	Memory space enable: When 0, PCI master access to main memory is disabled. When 1, PCI master access to main memory is enabled.	1b
5:2	Reserved		0001b
6	PARERR_REP	Parity Error Respond: When 1, Lambda will assert PERR# when a PCI parity error is detected. When 0, Lambda will not assert PERR# when a PCI parity error is detected.	0b
8:7	Reserved		00b
15:9	Reserved		0h
06H	**STAT**	**Status Register**	**2280h**
22:16	Reserved		All ‘0’s
23	FAST_B2B_STAT	Fast Back-to-Back status – This bit is when EN_PCI_FAST_DCD bit in the PCIC register is set. This bit indicates that LNB as a target can accept fast back-to-back cycles from another master.	1b
24	DATA_PAR_DET	Data Parity Detected: This bit will be set when operating as a bus master and either the PERR# output is driven low by Lambda or the target asserts PERR# and bit 6 of the Device Control Register is set. This bit can be reset by writing a 1.	0b
26:25	DEVSEL_TIM	DEVSEL Timing: These bits indicate the slowest time that LNB will return DEVSEL#. 00 = fast, 01 = medium 10 = slow 11 = reserved. These bits are hard-wired to 01	01b
27	Reserved		0b
28	REC_TAG_ABRT	Receive Target Abort: Reading a 1 indicates receiving a target abort condition. This bit can be reset by writing a 1.	0b
29	REC_MST_ABRT	Receive Master Abort: Reading a 1 indicates receiving a master abort condition(not including master abort generated from a special cycle). This bit can be reset by writing a 1.	1b
30	Reserved		0b
31	DET_PAR_ERR	Detect parity error: When Lambda detect a PCI parity error, this bit will be set to 1. This bit can be reset by writing a 1.	0b

Table 12.4 PCI Configuration Registers (cont.)

Register/ Bit	Name	Function	Value
08H	**RID**	**Revision ID Register**	**00h**
7:0	REVISION_ID	Revision ID number. These bits are hard wired.	00h
09h	CLASS	Class Register	060000h
31:8	CLASS_CODE	Class Code. These bits are hard-wired.	060000h
0DH	**LTMR**	**Latency Timer Register**	**0000h**
15:8	LAT_TIM	Latency Timer: Maximum Number of PCI clocks for Lambda initiated Burst cycles.	00h
40H	**ARB_ROUTE**	**PCI Arbiter Routing Register**	**0000h**
2:0	REQa_MAP	000 = REQ0#/GNT0# 001 = REQ1#/GNT1# 010 = REQ2#/GNT2# 011 = REQ3#/GNT3#	000b
3	REQa_PREEMPT	0 = REQa is non-preemtable 1 = REQa is pre-emptable	0b
6:4	REQb_MAP	000 = REQ1#/GNT1# 001 = REQ2#/GNT2# 010 = REQ3#/GNT3# 101 = REQ0#/GNT0#	000b
7	REQb_PREEMPT	0 = REQc is non-preemtable 1 = REQb is preemptable	0b
10:8	REQc_MAP	000 = REQ2#/GNT2# 001 = REQ3#/GNT3# 100 = REQ0#/GNT0# 101 = REQ1#/GNT1#	000b
11	REQc_PREEMPT	0 = REQc is non-preemtable 1 = REQc is preemptable	0b
14:12	REQd_MAP	000 = REQ3#/GNT3# 011 = REQ0#/GNT0# 100 = REQ1#/GNT1# 101 = REQ2#/GNT2#	000b
15	REQd_PREEMPT	0 = REQd is non-preemtable 1 = REQd is preemptable	0b
18:16	Reserved		000b
19	Reserved		0b
22:20	V3_REQ_MAP	000 = SB_REQ#/SB_GNT# 001 = REQ0#/GNT0# 010 = REQ1#/GNT1# 011 = REQ2#/GNT2# 100 = REQ3#/GNT3#	000b
23	V3_PREEMPT	0 = V3 REQ# is non-preemptable 1 = V3 REQ is preemptable	0b
26:24	Reserved		000b
27	Reserved		0b
28	MASKALLREQ	1 = mask all REQ’s (for dram initialization) 0 = unmask all REQ’s	0b
31:29	Reserved		000b

11.2. South Bridge

11.2.1. Floppy Disk Controller

(SIO LDN 00h)

Table 12.5 Floppy Disk Controller Registers

Register/ Bit	Name	Function	Value
30H	**LDN activate**	Bit 0: 1 = device enabled 0 = device disabled	**01h**
60H	**LDN Base addr MSB**	Bits 7-3 (for A15-11) are read only, 00000b.	**03h**
61H	**LDN Base addr LSB**	Bits 2 and 0 (for A2 and A0) are read only, 00b.	**F0h**
7Hh	**LDN IRQ**	Interrupt Number R/W	**06h**
71H	**LDN IRQ type select**	Bit 1 is read/write; other bits are read only. Indicates the type and level of the interrupt request number selected in the previous register. Bit 0 – Type of interrupt request selected in previous register 0 = Edge 1 = Level Bit 1 – Level of interrupt request selected in previous register 0 = Low polarity 1 = High polarity	**03h**
74H	**LDN DMA Select 0**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**02h**
75H	**LDN DMA Select 1**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
F0H	**LDN Config**	FDC Configuration register R/W	**24h**
0	TRI-STATE Control	When enabled and the device is inactive, the logical device output pins are in TRI-STATE. 0 = Disabled (Default) 1 = Enabled	0b
2:1	Reserved	Reset value of bit 2 is 1.	10b

Table 12.5 Floppy Disk Controller Registers (cont.)

Register/ Bit	Name	Function	Value
3	Write Protect	This bit allows forcing of write protect by software. When set, write to the floppy disk drive is disabled. This effect is identical to WP when it is active. 0 = Write protected according to WP signal (Default) 1 = Write protected regardless of value of WP signal	0b
4	Reserved	Must be 0	0b
5	DENSEL Polarity Control	0 = Active low for 500 Kbps or 1 Mbps data rates 1 = Active high for 500 Kbps or 1 Mbps data rates (Default)	1b
6	TDR Register Mode	0 = PC-AT compatible drive mode; i.e., bits 7-2 of the TDR are ignored (Default) 1 = Enhanced drive mode	0b
7	Four Drive Control	0 = Two floppy drives directly controlled by DR1-0 and MTR1-0 (Default) 1 = Four floppy drives controlled with the aid of external logic (One floppy only present in ZFx86)	0b
F1H	**LDN Drive ID**	**Drive ID register**	00h
1:0	Drive 0 ID	When drive 0 is accessed, these bits are reflected on bits 5-4 of the TDR register, respectively.	00b
3:2	Drive 1 ID	When drive 1 is accessed, these bits are reflected on bits 5-4 of the TDR register, respectively.	00b
7:4	Reserved		0000b

12.2.1.1. Floppy Disk Controller Bitmap Summary

Table 12.6 Floppy Disk Controller Bitmap Summary

Register/ Bit	Name	Function	Value
00H[1]	**SRA**	**Status A**	**0FFh**
0		Head Direction	1b
1		WP#	1b
2		INDEX#	1b
3		Head Select	1b
4		TRK0#	1b
5		Step	1b
6		Reserved	1b
7		IRQ pending	1b
01H[1]	**SRB**	**Status B**	**0FFh**
0		MTR0#	1b
1		Reserved	1b
2		WGATE#	1b
3		RDATA#	1b
4		WDATA#	1b
5		Drive Select 0 status	1b
7:6		Reserved	10b

Table 12.6 Floppy Disk Controller Bitmap Summary (cont.)

Register/ Bit	Name	Function	Value
02H	**DOR**	**Digital Output**	**0Ch**
1:0		Drive select	00b
2		Reset controller	1b
3		DMAEN	1b
4		Motor Enable 0	0b
7:5		Reserved	000b
03H	**TDR**	**Tape Drive**	**0FCh**
1:0		Tape Drive select 1,0	00b
3:2[2]		Logical Drive exchange	11b
5:4[2]		Drive ID information	11b
7:6		Reserved	11b
04H	**MSR**	**Main Status**	**80h**
0		Drive 0 busy	0b
1		Drive 1 busy	0b
2		Drive 2 busy	0b
3		Drive 3 busy	0b
4		Command in progress	0b
5		Non-DMA execution	0b
6		Data I/O direction	0b
7		RQM	1b
05H	**FIFO**	**Data (FIFO)**	**02h**
7:0		Data bits	2h
06H	**DIR**	**Digital Input**	**7Fh**
6:0		Reserved	1111111b
7		DSKCHG#	0b
07H	**DIR**	**Digital Input**	**7Fh**
6:0		Reserved	1111111b
7		DSKCHG#	0b

1. Applicable only in PS/2 Mode
2. Applicable only in Enhanced TDR Mode

11.2.2. Parallel Port

SIO LDN 01

Table 12.7 Parallel Port Registers

Register/ Bit	Name	Function	Value
30H	**LDN Activate**	Bit 0: 1 = device enabled 0 = device disabled	**00h**
60H	**LDN Base addr MSB**	Bits 7-3 (for A15-11) are read only, 00000b. Bit 2 (for A10) should be 0b.	**03h**
61H	**LDN Base addr LSB**	Bits 1 and 0 (A1 and A0) are read only, 00b. For ECP Mode 4 (EPP) or when using the Extended registers, bit 2 (A2) should also be 0b.	**78h**
70H	**LDN IRQ**	**Interrupt number**	**00h**
71H	**LDN IRQ Type Select**	Indicates the type and level of the interrupt request number selected in the previous register. Bit 0 = Type of interrupt request selected in previous register 0 = Edge 1 = Level Bit 1 = Level of interrupt request selected in previous register 0 = Low polarity 1 = High polarity	**02h**
74H	**LDN DMA Select 0**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
75H	**LDN DMA Select 1**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5–7 are reserved.	**04h**

Table 12.7 Parallel Port Registers (cont.)

Register/ Bit	Name	Function	Value
0F0H	**LDN Config**	**Parallel port configuration register**	**0E2h**
0	TRI-STATE Control	When enabled and the device is inactive, the logical device output pins are in TRI-STATE. 0 = Disabled (Default) 1 = Enabled	0b
1	Power Mode Control	When the logical device is active: 0 = Parallel port clock disabled. ECP modes and EPP time-out are not functional when the logical device is active. Registers are maintained. 1 = Parallel port clock enabled. All operation modes are functional when the logical device is active (Default).	1b
3:2	Reserved		00b
4	Extended Register Access	0 = Registers at base (address) + 403h, base + 404h and base + 405h are not accessible (reads and writes are ignored). 1 = Registers at base (address) + 403h, base + 404h and base + 405h are accessible. This option supports run-time configuration within the Parallel Port address space.	0b
7:5.	Reserved	Must be 111	111b

11.2.3. Serial Port 1

SIO LDN 02

Table 12.8 Serial Port 1 Registers

Register/ Bit	Name	Function	Value
30H	**LDN Activate**	Bit 0: 1 = device enabled 0 = device disabled	**01h**
60H	**LDN Base addr MSB**	Bits 7-3 (for A15-11) are read only, 00000b.	**02h**
61H	**LDN Base addr LSB**	Bits 2:0 (for A2:0) are read only, 000b.	**0F8h**
70H	**LDN IRQ**	**Interrupt number**	**03h**
71H	**LDN IRQ Type Select**	Indicates the type and level of the interrupt request number selected in the previous register. Bit 0 = Type of interrupt request selected in previous register 0 = Edge 1 = Level Bit 1 = Level of interrupt request selected in previous register 0 = Low polarity 1 = High polarity	**03h**

Table 12.8 Serial Port 1 Registers (cont.)

Register/ Bit	Name	Function	Value
74H	**LDN DMA Select 0**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
75H	**LDN DMA Select 1**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
0F0H	**LDN Config**	**Serial port configuration register**	**82h**
0	TRI-STATE Control	This bit controls the TRI-STATE status of the device output pins when it is inactive (disabled). 0 = Disabled (Default) 1 = Enabled when device inactive	0b
1	Power Mode Control	When the logical device is active in: 0 = Low power mode. Serial Ports 1 and 2 clock disabled. The output signals are set to their default states. The RI input signal can be programmed to generate an interrupt. Registers are maintained. (Unlike Active bit in Index 30 that also prevents access to Serial Ports 1 or 2 registers.) 1 = Normal power mode. Serial Ports 1 and 2 clock enabled. Serial Ports 1 and 2 are functional when the respective logical devices are active (Default).	1b
2	BUSY	This read only bit can be used by power management software to decide when to power-down Serial Ports 1 and 2 logical devices. 0 = No transfer in progress (Default). 1 = Transfer in progress.	0b
6:3	Reserved		0000b
7	Bank select	Enables bank switching for Serial Ports 1 and 2. 0 = Disabled (Default). 1 = Enabled	1b

11.2.4. Serial Port 2

SIO LDN 03

Table 12.9 Serial Port 2 Registers

Register/ Bit	Name	Function	Value
30H	**LDN Activate**	Bit 0: 1 = Device enabled 0 = Device disabled	**00h**
60H	**LDN Base addr MSB**	Bits 7-3 (for A15-11) are read only, 00000b.	**03h**
61H	**LDN Base addr LSB**	Bits 2:0 (for A2:0) are read only, 000b.	**0F8h**
70H	**LDN IRQ**	**Interrupt number**	**04h**
71H	**LDN IRQ Type Select**	Indicates the type and level of the interrupt request number selected in the previous register. Bit 0 = Type of interrupt request selected in previous register 0 = Edge 1 = Level Bit 1 = Level of interrupt request selected in previous register 0 = Low polarity 1 = High polarity	**03h**
74H	**LDN DMA Select 0**	Indicates selected DMA channel for DMA 0 of the logical device (0 – The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
75H	**LDN DMA Select 1**	Indicates selected DMA channel for DMA 0 of the logical device (0 – The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
0F0H	**LDN Config**	**Serial port configuration register**	**02h**
0	TRI-STATE Control	This bit controls the TRI-STATE status of the device output pins when it is inactive (disabled). 0 = Disabled (Default) 1 = Enabled when device inactive	0b

Table 12.9 Serial Port 2 Registers (cont.)

Register/ Bit	Name	Function	Value
1	Power Mode Control	When the logical device is active in: 0 = Low power mode. Serial Ports 1 and 2 clock disabled. The output signals are set to their default states. The RI input signal can be programmed to generate an interrupt. Registers are maintained. (Unlike Active bit in Index 30 that also prevents access to Serial Ports 1 or 2 registers.) 1 = Normal power mode. Serial Ports 1 and 2 clock enabled. Serial Ports 1 and 2 are functional when the respective logical devices are active (Default).	1b
2	BUSY	This read only bit can be used by power management software to decide when to power-down Serial Ports 1 and 2 logical devices. 0 = No transfer in progress (Default). 1 = Transfer in progress.	0b
6:3	Reserved		0000b
7	Bank select	Enables bank switching for Serial Ports 1 and 2. 0 = Disabled (Default). 1 = Enabled	0b

11.2.5. PS/2 Mouse/Keyboard

SIO LDN 05 and 06

Note: The following table contains only LDN 06 register information.

Table 12.10 PS/2 Mouse/Keyboard Registers

Register/ Bit	Name	Function	Value
30H	**LDN Activate**	Bit 0: 1 = Device enabled 0 = Device disabled Activate. See also bit 0 of the SIOCF1. When the Mouse of the KBC is inactive, the IRQ selected by the Mouse Interrupt Number register (index 70h) is not asserted. This register has no effect on host KBC commands handling the PS/2 Mouse.	**01h**
60H	**LDN Base addr MSB**	Bits 7-3 (for A15-11) are read only, 00000b.	**00h**
61H	**LDN Base addr LSB**	Bits 2:0 (for A2:0) are read only, 000b.	**60h**
62H	**LNB Port base MSB**	Bits 7:3 (for A15:11) are read only, 00000b	**00h**
63H	**LNB Port base LSB**	Bits 2:0 rea read only 100b	**64h**
70H	**LDN IRQ**	**KBC Interrupt number**	**01h**

Table 12.10 PS/2 Mouse/Keyboard Registers (cont.)

Register/ Bit	Name	Function	Value
71H	**LDN IRQ Type Select**	KBC interrupt type. Indicates the type and level of the interrupt request number selected in the previous register. Bit 0 = Type of interrupt request selected in previous register 0 = Edge 1 = Level Bit 1 = Level of interrupt request selected in previous register 0 = Low polarity 1 = High polarity	**02h**
74H	**LDN DMA Select 0**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
75H	**LDN DMA Select 1**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
0F0H	**LDN Config**	**KBC configuration register**	**40h**
0	TRI-STATE Control	If KBD is inactive (disabled) when this bit is set, the KBD pins (KBCLK and KBDAT) are in TRI-STATE. If Mouse is inactive (disabled) when this bit is set, the Mouse pins (MCLK and MDAT) are in TRI-STATE. 0 = Disabled (Default) 1 = Enabled	0b
5:1	Reserved	Use read-modify-write to change the value of the register. Do not change the value of these bits. Bit 2 must be 0.	00000b
7:6	Clock Source	The clock source can be changed only when the KBC is inactive (disabled). Bits7 6 Function 0 0 8 MHz 0 1 12 MHz (Default) 1 0 16 MHz 1 1 Reserved	01b

11.2.6. Infrared Communication Port Configuration

(SIO LDN 07)

Table 12.11 Infrared Communication Port Configuration Registers

Register/ Bit	Name	Function	Value
30H	**LDN Activate**	Bit 0: 1 = device enabled 0 = device disabled	**00h**
60H	**LDN Base addr MSB**	Bits 7-3 (for A15-11) are read only, 00000b.	**03h**
61H	**LDN Base addr LSB**	Bits 2:0 (for A2:0) are read only, 000b.	**0E8h**
70H	**LDN IRQ**	**Interrupt number**	**00h**
71H	**LDN IRQ Type Select**	Indicates the type and level of the interrupt request number selected in the previous register. Bit 0 = Type of interrupt request selected in previous register 0 = Edge 1 = Level Bit 1 = Level of interrupt request selected in previous register 0 = Low polarity 1 = High polarity	**03h**
74H	**LDN DMA Select 0**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
75H	**LDN DMA Select 1**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
0F0H	**LDN Config**	**Infrared configuration register**	**02h**
0	TRI-STATE Control	When enabled and the device is inactive, the logical device output pins are in TRI-STATE. One exception is the IRTX pin, which is driven to 0 when Serial Port 2 is inactive and is not affected by this bit. 0 = Disabled (Default) 1 = Enabled	0b

Table 12.11 Infrared Communication Port Configuration Registers (cont.)

Register/ Bit	Name	Function	Value
1	Power Mode Control	When the logical device is active in: 0 = Low power mode Clock disabled. The output signals are set to their default states. The RI input signal can be programmed to generate an interrupt. Registers are maintained. (Unlike Active bit in Index 30 that also prevents access to device registers.) 1 = Normal power mode Clock enabled. The device is functional when the logical device is active (Default).	1b
2	BUSY	This read only bit can be used by power management software to decide when to power-down the device. 0 = No transfer in progress (Default). 1 = Transfer in progress.	0b
6:3	Reserved		0000b
7	Bank select	Enables bank switching. 0 = All attempts to access the extended registers are ignored (Default). 1 = Enables bank switching.	0b

11.2.7. Access Bus

(SIO LDN 08)

Table 12.12 Access Bus Registers

Register/ Bit	Name	Function	Value
30H	**LDN Activate**	Bit 0: 1 = device enabled 0 = device disabled	**00h**
60H	**LDN Base addr_ MSB**	Base address MSB	**00h**
61H	**LDN Base addr LSB**	Base address LSB. Bits 2-0 (for A2-0) are read only, 000b.	**00h**
70H	**LDN IRQ**	Interrupt number	**00h**
71H	**LDN IRQ Type Select**	Indicates the type and level of the interrupt request number selected in the previous register. Bit 0 = Type of interrupt request selected in previous register 0 = Edge 1 = Level Bit 1 = Level of interrupt request selected in previous register 0 = Low polarity 1 = High polarity	**03h**

Table 12.12 Access Bus Registers (cont.)

Register/ Bit	Name	Function	Value
74H	**LDN DMA Select 0**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2–0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
75H	**LDN DMA Select 1**	Indicates selected DMA channel for DMA 0 of the logical device (0 = The first DMA channel when using more than one DMA channel). Bits 2-0 select the DMA channel for DMA 0. The valid choices are 0-3, where a value of 0 selects DMA channel 0, 1 selects channel 1, etc. A value of 4 indicates that no DMA channel is active. The values 5-7 are reserved.	**04h**
0F0H	**LDN ACBCNF**	**ACB configuration register**	**04h**
1:0	Reserved		00b
2	Internal Pullup Enable	0 = No internal pull-up resistors on SCL and SDA (Default) 1 = Internal pull-up resistors on SCL and SDA	1b
7:3	Reserved		00000b

11.2.8. Pin Multiplexor Registers

Table 12.13 Pin Multiplexor Registers

Register/ Bit	Name	Function	Value
F3BAR0+00H	**I/O Control register 1**		**05040000h**
15:0	Reserved		0000h
16	External RTC	Must be left at 0	0b
17	External KBC	Must be left at 0	0b
18	IO_USB_PCI_EN	USB ports 0 = Disable 1 = Enable	1b
19	IO_USB_SMI_PIN_EN	USB SMI I/O Configuration — Route USB-generated SMI directly to the SMI# pin: 0 = Disable 1 = Enable, USB-generated SMI pulls SMI# pin active (low) If bits 19 and 20 are enabled, the SMI generated by the USB is reported through the Top Level SMI Status Register at F1BAR0+I/O Offset 00h[14]. If only bit 19 is enabled, the USB can generate an SMI but there is no status reporting.	0b

Table 12.13 Pin Multiplexor Registers (cont.)

Register/ Bit	Name	Function	Value
20	IO_USB_SMI_PWM_EN	USB Internal SMI — Route USB-generated SMI through the Top Level SMI Status Register at F1BAR0+I/O Offset 00h[14]: 0 = Disable 1 = Enable Bit 19 must be enabled to allow the USB to generate an SMI for status reporting.	0b
21	IO_RTC_32K	This bit selects which 32K clock source is used. Resets to 0. 0 = Use SIO generated 32Khz. Clock is driven by RTC. 1 = Use internally generated 32Khz. Clock is derived by dividing the 48Mhz by 1484. **Note:** Bootstrap[6] = 1 can also be used to select internally generated 32Khz clock.	0b
22	IO_CLK32K_OE	This bit is set to drive 32Khz clock out on to GPIO[0]. Reset to 0.	0b
23	Reserved		0b
24	IO_ENABLE_SIO_DRIVING_ ISA_BUS	Enable Integrated SIO ISA Bus Control — Allow the integrated SIO to drive the internal and external ISA bus: 0 = Disable 1 = Enable (Default).	1b
26:25	IO_SIOCFG_IN	Integrated SIO Input Configuration. These two bits can be used to disable the integrated SIO totally or limit/control the base address: 00 = Integrated SIO disable 01 = Integrated SIO configuration access disable 10 = Integrated SIO base address 02Eh/02Fh enable 11 = Integrated SIO base address 015Ch/015Dh enable	10b
27	IO_ENABLE_SIO_IR	Enable integrated SIO infrared 0 = Disable 1 = Enable	0b
31:28	Reserved		0h
F3BAR0+04H	**I/O Control register 2**		**00000032**

Table 12.13 Pin Multiplexor Registers (cont.)

Register/ Bit	Name	Function	Value
0	IO_EXT_CLK_14M	Select 14.3Mhz clock source. Select either internally or externally generated 14.3Mhz clock source. If internal source is selected the actual clock frequency is 12Mhz (48Mhz / 4). 0 = Select external 14.3Mhz input as source. (Default) 1 = Select internally generated 14.3Mhz clock source. **Note:** Bootstrap[5] = 1 can also be used to select internally generated 14.3Mhz clock	0b
1	IO_IDE_ON_GPIO	Drive IDE channel 2 onto gpio. Must also have gpio conditioned to correct direction corresponding to IDE pin functionality. 0 = do not drive IDE onto gpio. (Default) 1 = drive IDE into gpio. gpio[1] is dmackx, gpio must be configured as output. gpio[2] is diowx, gpio must be configured as output. gpio[3] is diorx, gpio must be configured as output. gpio[5] is dreq, gpio must be configured as input. gpio[6] is iordy, gpio must be configured as input.	1b
2	IO_BUR_ON_SIO	This bit is used in design verification only. Resets to 0.	0b
3	IO_FUNC_ON_SIO	This bit is used in design verification only. Resets to 0.	0b
4	IO_ZFL_EN	This bit is set to enable the ZF-Logic. Resets to 0. **Note:** Bootstrap[22] also used to enable the ZF-Logic.	1b
5	IO_ZT_EN	This bit is set to enable the ZF-Logic ROM interface. Resets to 0. **Note:** Bootstrap[23] also used to enable ZF–Logic ROM interface.	1b
6	Reserved		0b
7	IO_CLK_14M_OE	This bit is set to drive the internally generated 12Mhz clock out on GPIO[4]. Resets to 0.	0b
31:8	Reserved		0000000h

11.2.9. GPIO Configuration Pins

12.2.9.1. GPIO 0

Table 12.14 GPIO0 Registers

Register/ Bit	Name	Function	Value
F0BAR0+24H	**GPIO Pin Configuration Access Register**		**44H**
0	Output Enable	Indicates the GPIO pin output state. It has no effect on input. 0 = TRI-STATE (Default) 1 = Output enabled.	0b
1	Output Type	Controls the output buffer type (open-drain or push-pull) of the corresponding GPIO pin. 0 = Open-drain (Default 1 = Push-pull **Note:** Bit 0 must = 1 for this bit to have effect.	0b
2	Pull-Up Control	Enables/disables the internal pull-up capability of the corresponding GPIO pin. It supports open-drain output signals with internal pull-ups and TTL input signals. 0 = Disable 1 = Enable (Default). **Note:** Bits [1:0] must = 01 for this bit to have effect.	1b
3	Lock	This bit locks the corresponding GPIO pin. Once this bit is set to 1 by software, it can only be cleared to 0 by system reset or power-off. 0 = No effect (Default) 1 = Direction, output type, pull-up and output value locked.	0b
4	PME Edge/Level Select	Selects the type (edge or level) of the signal that issues a PME from the corresponding GPIO pin: 0 = Edge input 1 = Level input. For normal operation always set this bit to 0 (edge input). Erratic system behavior results if this bit is set to 1.	0b
5	PME Polarity	Selects the polarity of the signal that issues a PME from the corresponding GPIO pin (falling/low or rising/high): 0 = Falling edge or low level input. 1 = Rising edge or high level input.	0b
6	PME Debounce Enable	Enables/disables IRQ debounce (debounce period = 16 ms) 0 = Disable 1 = Enable.	1b
31:7	Reserved		0h

12.2.9.2. GPIO 1

Table 12.15 GPIO1 Registers

Register/ Bit	Name	Function	Value
F0BAR0+24H	**GPIO Pin Configuration Access Register**		**03H**
0	Output Enable	Indicates the GPIO pin output state. It has no effect on input. 0 = TRI-STATE (Default) 1 = Output enabled.	1b
1	Output Type	Controls the output buffer type (open-drain or push-pull) of the corresponding GPIO pin. 0 = Open-drain (Default) 1 = Push-pull **Note:** Bit 0 must = 1 for this bit to have effect.	1b
2	Pull-Up Control	Enables/disables the internal pull-up capability of the corresponding GPIO pin. It supports open-drain output signals with internal pull-ups and TTL input signals. 0 = Disable; 1 = Enable (Default). **Note:** Bits [1:0] must = 01 for this bit to have effect.	0b
3	Lock	This bit locks the corresponding GPIO pin. Once this bit is set to 1 by software, it can only be cleared to 0 by system reset or power-off. 0 = No effect (Default); 1 = Direction, output type, pull-up and output value locked.	0b
4	PME Edge/Level Select	Selects the type (edge or level) of the signal that issues a PME from the corresponding GPIO pin: 0 = Edge input; 1 = Level input. **Note:** For normal operation always set this bit to 0 (edge input). Erratic system behavior will result if this bit is set to 1.	0b
5	PME Polarity	Selects the polarity of the signal that issues a PME from the corresponding GPIO pin (falling/low or rising/high). 0 = Falling edge or low level input. 1 = Rising edge or high level input.	0b
6	PME Debounce Enable	Enables/disables IRQ debounce (debounce period = 16 ms): 0 = Disable; 1 = Enable.	0b
31:7	Reserved		0h

12.2.9.3. GPIO 2

Table 12.16 GPIO2 Registers

Register/ Bit	Name	Function	Value
F0BAR0+24H	**GPIO Pin Configuration Access Register**		**03H**
0	Output Enable	Indicates the GPIO pin output state. It has no effect on input. 0 = TRI-STATE (Default) 1 = Output enabled.	1b
1	Output Type	Controls the output buffer type (open-drain or push-pull) of the corresponding GPIO pin. 0 = Open-drain (Default) 1 = Push-pull **Note:** Bit 0 must = 1 for this bit to have effect.	1b
2	Pull-Up Control	Enables/disables the internal pull-up capability of the corresponding GPIO pin. It supports open-drain output signals with internal pull-ups and TTL input signals. 0 = Disable; 1 = Enable (Default). **Note:** Bits [1:0] must = 01 for this bit to have effect.	0b
3	Lock	This bit locks the corresponding GPIO pin. Once this bit is set to 1 by software, it can only be cleared to 0 by system reset or power-off. 0 = No effect (Default); 1 = Direction, output type, pull-up and output value locked.	0b
4	PME Edge/Level Select	Selects the type (edge or level) of the signal that issues a PME from the corresponding GPIO pin: 0 = Edge input; 1 = Level input. **Note:** For normal operation always set this bit to 0 (edge input). Erratic system behavior results if this bit is set to 1.	0b
5	PME Polarity	Selects the polarity of the signal that issues a PME from the corresponding GPIO pin (falling/low or rising/high): 0 = Falling edge or low level input. 1 = Rising edge or high level input.	0b
6	PME Debounce Enable	Enables/disables IRQ debounce (debounce period = 16 ms): 0 = Disable 1 = Enable.	0b
31:7	Reserved		0h

12.2.9.4. GPIO 3

Table 12.17 GPIO3 Registers

Register/ Bit	Name	Function	Value
F0BAR0+24H	**GPIO Pin Configuration Access Register**		**03H**
0	Output Enable	Indicates the GPIO pin output state. It has no effect on input. 0 = TRI-STATE (Default) 1 = Output enabled.	1b
1	Output Type	Controls the output buffer type (open-drain or push-pull) of the corresponding GPIO pin. 0 = Open-drain (Default) 1 = Push-pull **Note:** Bit 0 must = 1 for this bit to have effect.	1b
2	Pull-Up Control	Enables/disables the internal pull-up capability of the corresponding GPIO pin. It supports open-drain output signals with internal pull-ups and TTL input signals. 0 = Disable 1 = Enable (Default). **Note:** Bits [1:0] must = 01 for this bit to have effect.	0b
3	Lock	This bit locks the corresponding GPIO pin. Once this bit is set to 1 by software, it can only be cleared to 0 by system reset or power-off. 0 = No effect (Default); 1 = Direction, output type, pull-up and output value locked.	0b
4	PME Edge/Level Select	Selects the type (edge or level) of the signal that issues a PME from the corresponding GPIO pin: 0 = Edge input; 1 = Level input. **Note:** For normal operation always set this bit to 0 (edge input). Erratic system behavior results if this bit is set to 1.	0b
5	PME Polarity	Selects the polarity of the signal that issues a PME from the corresponding GPIO pin (falling/low or rising/high): 0 = Falling edge or low level input. 1 = Rising edge or high level input.	0b
6	PME Debounce Enable	Enables/disables IRQ debounce (debounce period = 16 ms): 0 = Disable 1 = Enable.	0b
31:7	Reserved		0h

12.2.9.5. GPIO 4

Table 12.18 GPIO4 Registers

Register/ Bit	Name	Function	Value
F0BAR0+24H	**GPIO Pin Configuration Access Register**		**44H**
0	Output Enable	Indicates the GPIO pin output state. It has no effect on input. 0 = TRI-STATE (Default) 1 = Output enabled.	0b
1	Output Type	Controls the output buffer type (open-drain or push-pull) of the corresponding GPIO pin. 0 = Open-drain (Default) 1 = Push-pull **Note:** Bit 0 must = 1 for this bit to have an effect.	0b
2	Pull-Up Control	Enables/disables the internal pull-up capability of the corresponding GPIO pin. It supports open-drain output signals with internal pull-ups and TTL input signals. 0 = Disable 1 = Enable (Default). **Note:** Bits [1:0] must = 01 for this bit to have an effect.	1b
3	Lock	This bit locks the corresponding GPIO pin. Once this bit is set to 1 by software, it can only be cleared to 0 by system reset or power-off. 0 = No effect (Default) 1 = Direction, output type, pull-up and output value locked.	0b
4	PME Edge/Level Select	Selects the type (edge or level) of the signal that issues a PME from the corresponding GPIO pin: 0 = Edge input; 1 = Level input. **Note:** For normal operation always set this bit to 0 (edge input). Erratic system behavior results if this bit is set to 1.	0b
5	PME Polarity	Selects the polarity of the signal that issues a PME from the corresponding GPIO pin (falling/low or rising/high): 0 = Falling edge or low level input. 1 = Rising edge or high level input.	0b
6	PME Debounce Enable	Enables/disables IRQ debounce (debounce period = 16 ms): 0 = Disable 1 = Enable.	1b
31:7	Reserved		0h

12.2.9.6. GPIO 5

Table 12.19 GPIO5 Registers

Register/ Bit	Name	Function	Value
F0BAR0+24H	**GPIO Pin Configuration Access Register**		**00H**
0	Output Enable	Indicates the GPIO pin output state. It has no effect on input. 0 = TRI-STATE (Default) 1 = Output enabled.	0b
1	Output Type	Controls the output buffer type (open-drain or push-pull) of the corresponding GPIO pin. 0 = Open-drain (Default) 1 = Push-pull **Note:** Bit 0 must = 1 for this bit to have effect.	0b
2	Pull-Up Control	Enables/disables the internal pull-up capability of the corresponding GPIO pin. It supports open-drain output signals with internal pull-ups and TTL input signals. 0 = Disable 1 = Enable (Default). **Note:** Bits [1:0] must = 01 for this bit to have effect.	0b
3	Lock	This bit locks the corresponding GPIO pin. Once this bit is set to 1 by software, it can only be cleared to 0 by system reset or power-off. 0 = No effect (Default); 1 = Direction, output type, pull-up and output value locked.	0b
4	PME Edge/Level Select	Selects the type (edge or level) of the signal that issues a PME from the corresponding GPIO pin: 0 = Edge input; 1 = Level input. **Note:** For normal operation, always set this bit to 0 (edge input). Erratic system behavior results if this bit is set to 1.	0b
5	PME Polarity	Selects the polarity of the signal that issues a PME from the corresponding GPIO pin (falling/low or rising/high): 0 = Falling edge or low level input. 1 = Rising edge or high level input.	0b
6	PME Debounce Enable	Enables/disables IRQ debounce (debounce period = 16 ms): 0 = Disable 1 = Enable.	0b
31:7	Reserved		0h

12.2.9.7. GPIO 6

Table 12.20 GPIO6 Registers

Register/ Bit	Name	Function	Value
F0BAR0+24H	**GPIO Pin Configuration Access Register**		**00h**
0	Output Enable	Indicates the GPIO pin output state. It has no effect on input. 0 = TRI-STATE (Default) 1 = Output enabled.	0b
1	Output Type	Controls the output buffer type (open-drain or push-pull) of the corresponding GPIO pin. 0 = Open-drain (Default) 1 = Push-pull **Note:** Bit 0 must = 1 for this bit to have effect.	0b
2	Pull-Up Control	Enables/disables the internal pull-up capability of the corresponding GPIO pin. It supports open-drain output signals with internal pull-ups and TTL input signals. 0 = Disable 1 = Enable (Default). **Note:** Bits [1:0] must = 01 for this bit to have effect.	0b
3	Lock	This bit locks the corresponding GPIO pin. Once this bit is set to 1 by software, it can only be cleared to 0 by system reset or power-off. 0 = No effect (Default) 1 = Direction, output type, pull-up and output value locked.	0b
4	PME Edge/Level Select	Selects the type (edge or level) of the signal that issues a PME from the corresponding GPIO pin: 0 = Edge input; 1 = Level input. **Note:** For normal operation always set this bit to 0 (edge input). Erratic system behavior results if this bit is set to 1.	0b
5	PME Polarity	Selects the polarity of the signal that issues a PME from the corresponding GPIO pin (falling/low or rising/high): 0 = Falling edge or low level input. 1 = Rising edge or high level input.	0b
6	PME Debounce Enable	Enables/disables IRQ debounce (debounce period = 16 ms): 0 = Disable 1 = Enable.	0b
31:7	Reserved		0h

12.2.9.8. GPIO 7

Table 12.21 GPIO7 Registers

Register/ Bit	Name	Function	Value
F0BAR0+24H	**GPIO Pin Configuration Access Register**		**44H**
0	Output Enable	Indicates the GPIO pin output state. It has no effect on input. 0 = TRI-STATE (Default) 1 = Output enabled.	0b
1	Output Type	Controls the output buffer type (open-drain or push-pull) of the corresponding GPIO pin. 0 = Open-drain (Default) 1 = Push-pull **Note:** Bit 0 must = 1 for this bit to have effect.	0b
2	Pull-Up Control	Enables/disables the internal pull-up capability of the corresponding GPIO pin. It supports open-drain output signals with internal pull-ups and TTL input signals. 0 = Disable 1 = Enable (Default). **Note:** Bits [1:0] must = 01 for this bit to have effect.	1b
3	Lock	This bit locks the corresponding GPIO pin. Once this bit is set to 1 by software, it can only be cleared to 0 by system reset or power-off. 0 = No effect (Default); 1 = Direction, output type, pull-up and output value locked.	0b
4	PME Edge/Level Select	Selects the type (edge or level) of the signal that issues a PME from the corresponding GPIO pin: 0 = Edge input; 1 = Level input. **Note:** For normal operation always set this bit to 0 (edge input). Erratic system behavior will result if this bit is set to 1.	0b
5	PME Polarity	Selects the polarity of the signal that issues a PME from the corresponding GPIO pin (falling/low or rising/high): 0 = Falling edge or low level input. 1 = Rising edge or high level input.	0b
6	PME Debounce Enable	Enables/disables IRQ debounce (debounce period = 16 ms): 0 = Disable 1 = Enable.	1b
31:7	Reserved		0h

12.2.9.9. ZF–LOGIC Registers

Table 12.22 ZF–Logic Registers

Register/ Bit	Name	Function	Value
06H	**PWM Duty Cycle**		**00h**
7:0	Duty	00h = 100% low 0FFh = 0% low	00h
08H	**PWM Control**		**00h**
0	PWMEN	Enable/Disable PWM output 0 = PWM is disabled 1 = PWM is enabled	0b
1	Clksrc	Selects the PWM prescaler input clock 0 = PWM is clocked from 32kHz clock 1 = PWM is clocked from 8MHz ISA clock	0b
3:2	Reserved		00b
4	Dlevel	Value to be set at PWM output pin when bit5 of register 08h is set to 1	0b
5	Direct	Enables direct control of PWM output by bit 4 0 = PWM drives the PWM output pin 1 = bit4 of register 08h drives the PWM output pin	0b
7:6	Reserved		00b
10H	**Watchdog control 1**		**00h**
0	wd1_en	Enable watchdog 1 0 = WD1 is disabled 1 = WD1 is enabled	0b
1	wd2_en	Enable watchdog 2 0 = WD2 is disabled 1= WD2 is enabled	0b
3:2	Reserved		00b
4	wd1_ld	Reload WD1 counter Active event for this bit is transition from 0 to 1	0b
5	wd2_ld	Reload WD2 counter. Active event for this bit is transition from 0 to 1	0b
7:6	Reserved		00b
11H	**Watchdog control 2**		**00h**
0	wd1_c	Enable SCI on WD1 expiry 0 = WD1 will not generate SCI 1 = WD1 will generate SCI event on expiry	0b
1	wd1_n	Enable NMI on WD1 expiry 0 = WD1 will not generate NMI 1 = WD1 will generate NMI on expiry	0b
2	wd1_s	Enable SMI event on WD1 expiry 0 = WD1 will not generate SMI 1 = WD1 will generate SMI event on expiry	0b

Table 12.22 ZF–Logic Registers (cont.)

Register/ Bit	Name	Function	Value
3	wd1_rs	Enable RESET on WD1 expiry 0 = WD1 will not generate system reset 1 = WD1 will generate system reset on expiry	0b
4	wdi_ed	Active front of WDI input 0 = WDI is asserted on 0->1 transition 1 - WDI is asserted on 1->0 transition	0b
5	Wdo_-1	Create output on WDO output pin on ZFx86 at WD1 expiry or one 32kHz clock tick before 0 = WDO signal will be set high on WD1 expiery 1 = WDO signal is set high one clock tick before WD1 expires. WD1 events will always occur at WD1 expiry and are not affected of wdo_-1 bit setting. **Note:** This feature gives possibility to cause automatic reload of WD1 when WDO is wired to WDI.	0b
6	wdi_en	Enable the assertion of WDI input pin on ZFx86 to to reload the watchdog 1 counter 0 = WDI input ignored 1 = WDI assertion reloads watchdog 1 counter	0b
7	Reserved		0b
16H	**IO Window 0 control**		**00h**
3:0	Win_siz	Number of consecutive 8-bit I/O addresses to decode starting from I/O window base. The number of addresses decoded is win_siz + 1. For example, setting the window size to 0 enables one I/O address at I/O window base. Setting size to 0Fh will enable I/O window of 16 addresses starting from I/O window base.	0000b
4	win_en	I/O window enable in I/O space 0 = I/O window is disabled 1 = I/O window is enabled	0b
5	act_lvl	io_cs active level 0 = io_cs is active low 1 = io_cs is active high	0b
6	16_bit	I/O window datapath width 0 = 8-bit wide access 1 = 16-bit wide access	0b
7	win_ro	I/O window wead/write control 0 = Access is read-write 1= Access is read-only **Note:** Setting window to read-only mode disables IOW_N signal on ISA bus for IO window address range.	0b

Table 12.22 ZF–Logic Registers (cont.)

Register/ Bit	Name	Function	Value
1AH	**IO Window 1 control**		**00h**
3:0	Win_siz	Number of consecutive 8-bit I/O addresses to decode starting from I/O window base. The number of addresses decoded is win_siz + 1. For example, setting the window size to 0 enables one I/O address at I/O window base. Setting size to 0Fh will enable I/O window of 16 addresses starting from I/O window base.	0000b
4	win_en	I/O window enable in I/O space 0 = I/O window is disabled 1 = I/O window is enabled	0b
5	act_lvl	io_cs active level 0 = io_cs is active low 1 = io_cs is active high	0b
6	16_bit	I/O window datapath width 0 = 8-bit wide access 1 = 16-bit wide access	0b
7	win_ro	I/O window wead/write control 0 = Access is read-write 1= Access is read-only **Note:** Setting window to read-only mode disables IOW_N signal on ISA bus for IO window address range.	0b
1EH	**IO Window 2 control**		**00h**
3:0	Win_siz	Number of consecutive 8-bit I/O addresses to decode starting from I/O window base. The number of addresses decoded is win_siz + 1. For example, setting the window size to 0 enables one I/O address at I/O window base. Setting size to 0Fh will enable I/O window of 16 addresses starting from I/O window base.	0000b
4	win_en	I/O window enable in I/O space 0 = I/O window is disabled 1 = I/O window is enabled	0b
5	act_lvl	io_cs active level 0 = io_cs is active low 1 = io_cs is active high	0b
6	16_bit	I/O window datapath width 0 = 8-bit wide access 1= 16-bit wide access	0b
7	win_ro	I/O window wead/write control 0 = Access is read-write 1 = Access is read-only **Note:** Setting window to read-only mode disables IOW_N signal on ISA bus for IO window address range.	0b

Table 12.22 ZF–Logic Registers (cont.)

Register/ Bit	Name	Function	Value
22H	**IO Window 3 control**		**00h**
3:0	Win_siz	Number of consecutive 8-bit I/O addresses to decode starting from I/O window base. The number of addresses decoded is win_siz + 1. For example, setting the window size to 0 enables one I/O address at I/O window base. Setting size to 0Fh will enable I/O window of 16 addresses starting from I/O window base.	0000b
4	win_en	I/O window enable in I/O space 0 = I/O window is disabled 1 = I/O window is enabled	0b
5	act_lvl	io_cs active level 0 = io_cs is active low 1 = io_cs is active high	0b
6	16_bit	I/O window datapath width 0 = 8-bit wide access 1 = 16-bit wide access	0b
7	win_ro	I/O window wead/write control 0 = Access is read-write 1 = Access is read-only **Note:** Setting window to read-only mode disables IOW_N signal on ISA bus for IO window address range.	0b
27H	**Memory window 0 base 1**	**Memory window base address bits 15..12**	**00h**
28H	**Memory window 0 base 2**	**Memory window base address bits 23..16**	**0F0h**
2BH	**Memory window0 size 1**	**Memory window size bits 15..12**	**0F0h**
2CH	**Memory window 0 size 2**	**Memory window size bits 23..16**	**01h**
2FH	**Memory window 0 page 1**	**Memory window page bits 15..12**	**00h**
30H	**Memory window 0 page 2**	**Memory window page bits 23..16**	**00h**
33H	**Memory window 1 base 1**	**Memory window base address bits 15..12**	**00h**
34H	**Memory window 1 base 2**	**Memory window base address bits 23..16**	**00h**
37H	**Memory window 1 size 1**	**Memory window size bits 15..12**	**00h**
38H	**Memory window 1 size 2**	**Memory window size bits 23..16**	**00h**
3BH	**Memory window 1 page 1**	**Memory window page bits 15..12**	**00h**
3CH	**Memory window 1 page 2**	**Memory window page bits 23..16**	**00h**
3FH	**Memory window 2 base 1**	**Memory window base address bits 15..12**	**00h**
40H	**Memory window 2 base 2**	**Memory window base address bits 23..16**	**00h**
43H	**Memory window 2 size 1**	**Memory window size bits 15..12**	**00h**
44H	**Memory window 2 size 2**	**Memory window size bits 23..16**	**00h**
47H	**Memory window 2 page 1**	**Memory window page bits 15..12**	**00h**
48H	**Memory window 2 page 2**	**Memory window page bits 23..16**	**00h**
4BH	**Memory window 3 base 1**	**Memory window base address bits 15..12**	**00h**
4CH	**Memory window 3 base 2**	**Memory window base address bits 23..16**	**00h**
4FH	**Memory window 3 size 1**	**Memory window size bits 15..12**	**00h**
50H	**Memory window 3 size 2**	**Memory window size bits 23..16**	**00h**
53H	**Memory window 3 page 1**	**Memory window page bits 15..12**	**00h**
54H	**Memory window 3 page 2**	**Memory window page bits 23..16**	**00h**

Numerics

A

B

C

N

R

S

T

X

Y

Z

ZF Micro Devices, Inc.

1052 Elwell Court

Palo Alto, California 94303

(650) 965-3800 · Fax 965-4050

www.zfmicro.com