National Semiconductor

PRELIMINARY

NS32332-10/NS32332-15 32-Bit Advanced Microprocessors

General Description

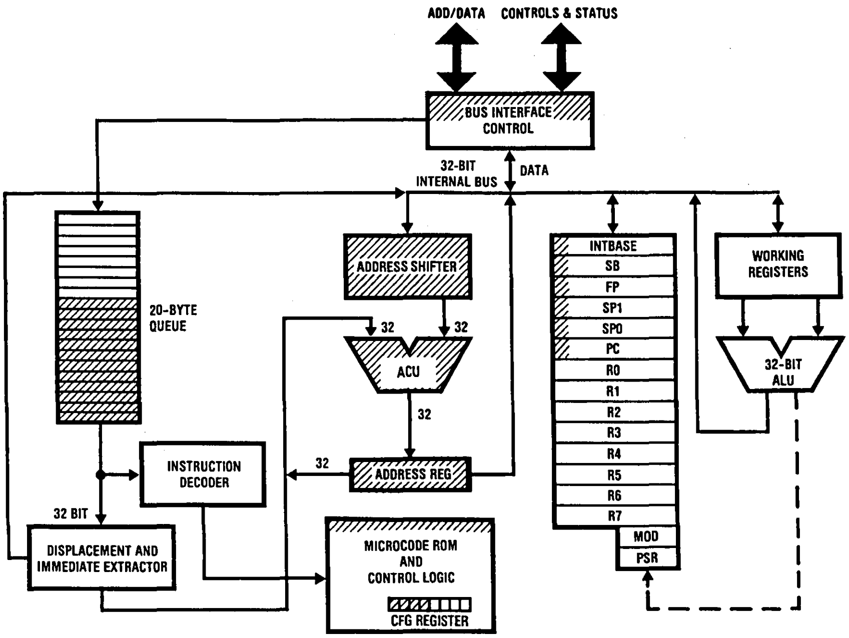
The NS32332 is a 32-bit, virtual memory microprocessor with 4 GByte addressing and an enhanced internal implementation. It is fully object code compatible with other Series 32000® microprocessors, and it has the added features of 32-bit addressing, higher instruction execution throughput, cache support, and expanded bus handling capabilities. The new bus features include bus error and retry support, dynamic bus sizing, burst mode memory accessing, and enhanced slave processor communication protocol. The higher clock frequency and added features of the NS32332 enable it to deliver 2 to 3 times the performance of the NS32032.

The NS32332 microprocessor is designed to work with both the 16- and 32-bit slave processors of the Series 32000 family.

Features

- 32-bit architecture and implementation
- 4 Gbyte uniform addressing space
- Software compatible with the Series 32000 Family
- Powerful instruction set
  - — General 2-address capability
  - — Very high degree of symmetry
  - — Address modes optimized for high level languages
- Supports both 16- and 32-bit Slave Processor Protocol
  - — Memory management support via NS32082 or NS32382
  - — Floating point support via NS32081 or NS32381
- Extensive bus feature
  - — Burst mode memory accessing
  - — Cache memory support
  - — Dynamic bus configuration (8-, 16-, 32-bits)
  - — Fast bus protocol
- High speed XMOS™ technology
- 84 Pin grid array package

Block Diagram



TL/EE/8673-1

FIGURE 1

*Shaded areas indicate enhancements from the NS32032.

Table of Contents

List of Illustrations

List of Illustrations (Continued)

1.0 Product Introduction

The Series 32000 Microprocessor family is a new generation of devices using National's XMOS and CMOS technologies. By combining state-of-the-art MOS technology with a very advanced architectural design philosophy, this family brings mainframe computer processing power to VLSI processors.

The Series 32000 family supports a variety of system configurations, extending from a minimum low-cost system to a powerful 4 gigabyte system. The architecture provides complete upward compatibility from one family member to another. The family consists of a selection of CPUs supported by a set of peripherals and slave processors that provide sophisticated interrupt and memory management facilities as well as high-speed floating-point operations. The architectural features of the Series 32000 family are described briefly below:

Powerful Addressing Modes. Nine addressing modes available to all instructions are included to access data structures efficiently.

Data Types. The architecture provides for numerous data types, such as byte, word, doubleword, and BCD, which may be arranged into a wide variety of data structures.

Symmetric Instruction Set. While avoiding special case instructions that compilers can't use, the Series 32000 family incorporates powerful instructions for control operations, such as array indexing and external procedure calls, which save considerable space and time for compiled code.

Memory-to-Memory Operations. The Series 32000 CPUs represent two-address machines. This means that each operand can be referenced by any one of the addressing modes provided. This powerful memory-to-memory architecture permits memory locations to be treated as registers for all useful operations. This is important for temporary operands as well as for context switching.

Memory Management. Either the NS32382 or the NS32082 Memory Management Unit may be added to the system to provide advanced operating system support functions, including dynamic address translation, virtual memory management, and memory protection.

Large, Uniform Addressing. The NS32332 has 32-bit address pointers that can address up to 4 gigabytes without requiring any segmentation; this addressing scheme provides flexible memory management without added-on expense.

Modular Software Support. Any software package for the Series 32000 family can be developed independent of all other packages, without regard to individual addressing. In addition, ROM code is totally relocatable and easy to access, which allows a significant reduction in hardware and software cost.

Software Processor Concept. The Series 32000 architecture allows future expansions of the instruction set that can be executed by special slave processors, acting as extensions to the CPU. This concept of slave processors is unique to the Series 32000 family. It allows software compatibility even for future components because the slave hardware is transparent to the software. With future advances in semiconductor technology, the slaves can be physically integrated on the CPU chip itself.

To summarize, the architectural features cited above provide three primary performance advantages and characteristics:

- High-Level Language Support
- Easy Future Growth Path
- Application Flexibility

1.1 NS32332 KEY FEATURES

The NS32332 is a 32-bit CPU in the Series 32000 family. It is totally software compatible with the NS32032, NS32016, and NS32008 CPUs but with an enhanced internal implementation.

The NS32332 design goals were to achieve two to three times the throughput of the NS32032 and to provide the full 32-bit addressing inherent in the architecture.

The basic approaches to higher throughput were: fewer clock cycles per instruction, better bus use, and higher clock frequency.

An examination of the block diagram of the NS32332 shows it to be identical to that of the NS32032, except for enhanced bus interface control, a 20-byte (rather than 8-byte) instruction prefetch queue, and special hardware in the address unit. The new addressing hardware consists of a high-speed ALU, a barrel shifter on one of its inputs, and an address register. Of the throughput improvement not due to increased clock frequency, about 15% is derived from the new address unit hardware, 15% from the bus enhancements, 10% from the larger prefetch queue, and 60% from microcode improvements.

Other important aspects of the enhanced bus interface circuitry of the NS32332 are a burst access mode, designed to work with nibble and static column RAMs, read and write timing designed to support caches, and support for bus error processing.

An enhanced slave processor communication protocol is designed to achieve improved performance with the NS32382 MMU and NS32381 FPU, while still working directly with the previous NS32082 MMU and NS32081 FPU.

2.0 Architectural Description

2.1 PROGRAMMING MODEL

The Series 32000 architecture has 8 general purpose and 8 dedicated registers. All registers are 32 bits wide except the STATUS and MODULE register. These two registers are each 16 bits wide.

2.1.1 General Purpose Registers

There are eight registers for meeting high speed general storage requirements, such as holding temporary variables and addresses. The general purpose registers are free for any use by the programmer. They are thirty-two bits in length. If a general register is specified for an operand that is eight or sixteen bits long, only the low part of the register is used; the high part is not referenced or modified.

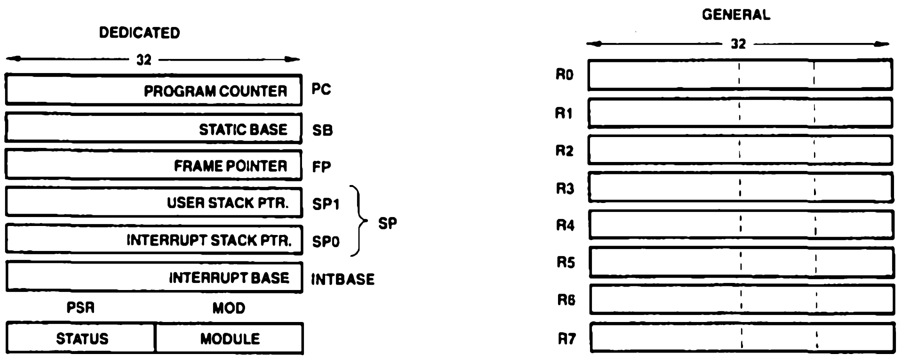
2.1.2 Dedicated Registers

The eight dedicated registers of the processor are assigned specific functions.

PC: The PROGRAM COUNTER register is a pointer to the first byte of the instruction currently being executed. The PC is used to reference memory in the program section.

SP0, SP1: The SP0 register points to the lowest address of the last item stored on the INTERRUPT STACK. This stack is normally used only by the operating system. It is used

2.0 Architectural Description (Continued)



TL/EE/8673-2

FIGURE 2-1. The General and Dedicated Registers

primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and trap service routines. The SP1 register points to the lowest address of the last item stored on the USER STACK. This stack is used by normal user programs to hold temporary data and subroutine return information.

In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0 the SP refers to SP0. If the S bit in the PSR is 1 then SP refers to SP1.

Stacks in the Series 32000 family grow downward in memory. A Push operation pre-decrements the Stack Pointer by the operand length. A Pop operation post-increments the Stack Pointer by the operand length.

FP: The FRAME POINTER register is used by a procedure to access parameters and local variables on the stack. The FP register is set up on procedure entry with the ENTER instruction and restored on procedure termination with the EXIT instruction.

The frame pointer holds the address in memory occupied by the old contents of the frame pointer.

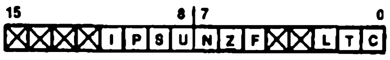
SB: The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules. The SB register holds the lowest address in memory occupied by the global variables of a module.

INTBASE: The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Sec. 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table.

MOD: The MODULE register holds the address of the module descriptor of the currently executing software module. The MOD register is sixteen bits long, therefore the module table must be contained within the first 64K bytes of memory.

PSR: The PROCESSOR STATUS REGISTER (PSR) holds the status codes for the microprocessor.

The PSR is sixteen bits long, divided into two eight-bit halves. The low order eight bits are accessible to all programs, but the high order eight bits are accessible only to programs executing in Supervisor Mode.



TL/EE/8673-3

FIGURE 2-2. Processor Status Register

C: The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It can be used with the ADDC and SUBC instructions to perform multiple-precision integer arithmetic calculations. It may have a setting of 0 (no carry or borrow) or 1 (carry or borrow).

T: The T bit causes program tracing. If this bit is a 1, a TRC trap is executed after every instruction (Sec. 3.8.5).

L: The L bit is altered by comparison instructions. In a comparison instruction the L bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as unsigned integers. Otherwise, it is set to "0". In Floating Point comparisons, this bit is always cleared.

F: The F bit is a general condition flag, which is altered by many instructions (e.g., integer arithmetic instructions use it to indicate overflow).

Z: The Z bit is altered by comparison instructions. In a comparison instruction the Z bit is set to "1" if the second operand is equal to the first operand; otherwise it is set to "0".

N: The N bit is altered by comparison instructions. In a comparison instruction the N bit is set to "1" if the second operand is less than the first operand, when both operands are interpreted as signed integers. Otherwise, it is set to "0".

U: If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When U = 0 the processor is said to be in Supervisor Mode; when U = 1 the processor is said to be in User Mode. A User Mode program is restricted from executing certain instructions and accessing certain registers which could interfere with the operating system. For example, a User Mode program is prevented from changing the setting of the flag used to indicate its own privilege mode. A Supervisor Mode program is assumed to be a trusted part of the operating system, hence it has no such restrictions.

S: The S bit specifies whether the SP0 register or SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

2.0 Architectural Description (Continued)

P: The P bit prevents a TRC trap from occurring more than once for an instruction (Sec. 3.8.5.). It may have a setting of 0 (no trace pending) or 1 (trace pending).

I: If I = 1, then all interrupts will be accepted (Sec. 3.8.). If I = 0, only the NMI interrupt is accepted. Trap enables are not affected by this bit.

2.1.3 The Configuration Register (CFG)*

Within the Control section of the CPU is the CFG Register, which declares the presence and type of external devices. It is referenced by only one instruction, SETCFG, which is intended to be executed only as part of system initialization after reset. The format of the CFG Register is shown in *Figure 2-3*.

*The NS32332 CPU has four new bits in the CFG Register, namely P, FC, FM and FF.

7							0
P	FC	FM	FF	C	M	F	I

FIGURE 2-3. CFG Register

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the $\overline{\text{INT}}$ pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Sec. 3.8.

The F, M and C bits declare the presence of the FPU, MMU and Custom Slave Processors. If these bits are not set, the corresponding instructions are trapped as being undefined.

The FF, FM, FC bits define the Slave Communication Protocol to be used in FPU, MMU and Custom Slave instructions (Sec. 3.4.9). If these bits are not set, the corresponding instructions will use the 16-bit protocol (32032 compatible). If these bits are set, the corresponding instructions will use the new (fast) 32-bit protocol.

The P bit improves the efficiency of the Write Validation Buffer in the CPU. It is set if the Virtual Memory has page size(s) larger than or equal to 4 Kbytes. It is reset otherwise. In Systems where the MMU is not present, the P bit is not used.

2.1.4 Memory Organization

The main memory is a uniform linear address space. Memory locations are numbered sequentially starting at zero and ending at $2^{32} - 1$. The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits. Unless otherwise noted, diagrams in this document show data stored in memory with the lowest address on the right and the highest address on the left. Also, when data is shown vertically, the lowest address is at the top of a diagram and the highest address at the bottom of the diagram. When bits are numbered in a diagram, the least significant bit is given the number zero, and is shown at the right of the diagram. Bits are numbered in increasing significance and toward the left.

7 0
A

Byte at Address A

Two contiguous bytes are called a word. Except where noted (Sec. 2.2.1), the least significant byte of a word is stored at the lower address, and the most significant byte of the word is stored at the next higher address. In memory, the address of a word is the address of its least significant byte, and a word may start at any address.

15 MSB's 8	7 LSB's 0
A+1	A

Word at Address A

Two contiguous words are called a double word. Except where noted (Sec. 2.2.1), the least significant word of a double word is stored at the lowest address and the most significant word of the double word is stored at the address two greater. In memory, the address of a double word is the address of its least significant byte, and a double word may start at any address.

31 MSB's 24	23 16	15 8	7 LSB's 0
A + 3	A + 2	A + 1	A

Double Word at Address A

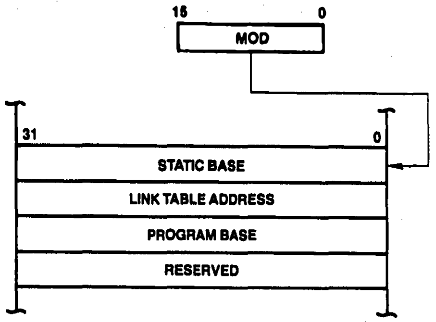
Although memory is addressed as bytes, it is actually organized as double-words. Note that access time to a word or a double-word depends upon its address, e.g. double-words that are aligned to start at addresses that are multiples of four will be accessed more quickly than those not so aligned. This also applies to words that cross a double-word boundary.

2.1.5 Dedicated Tables

Two of the dedicated registers (MOD and INTBASE) serve as pointers to dedicated tables in memory.

The INTBASE register points to the Interrupt Dispatch and Cascade tables.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically up-dated by the Call External Procedure instructions (CXP and CXPD).



TL/EE/8673-4

FIGURE 2-4. Module Descriptor Format

The format of a Module Descriptor is shown in *Figure 2-4*. The Static Base entry contains the address of static data assigned to the running module. It is loaded into the CPU Static Base register by the CXP and CXPD instructions. The Program Base entry contains the address of the first byte of instruction code in the module. Since a module may have multiple entry points, the Program Base pointer serves only as a reference to find them.

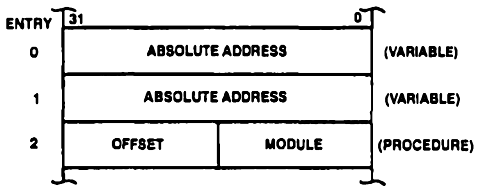
2.0 Architectural Description (Continued)

The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information needed for:

1) Sharing variables between modules. Such variables are accessed through the Link Table via the External addressing mode.
2) Transferring control from one module to another. This is done via the Call External Procedure (CXP) instruction.

The format of a Link Table is given in *Figure 2-5.* A Link Table Entry for an external variable contains the 32-bit address of that variable. An entry for an external procedure contains two 16-bit fields: Module and Offset. The Module field contains the new MOD register contents for the module being entered. The Offset field is an unsigned number giving the position of the entry point relative to the new module's Program Base pointer.

For further details of the functions of these tables, see the Series 32000 Instruction Set Reference Manual.



TL/EE/8673-5

FIGURE 2-5. A Sample Link Table

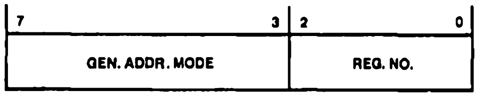
2.2 INSTRUCTION SET

2.2.1 General Instruction Format

Figure 2-6 shows the general format of a Series 32000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See *Figure 2-7.*

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected address modes. Each Disp/Imm field may contain one or two displacements, or one immediate value. The size of a Displacement field is encoded with the top bits of that field, as shown in *Figure 2-8*, with the remaining bits interpreted as a signed (two's complement) value. The size of an immediate value is determined from the Opcode field. Both Displacement and Immediate fields are stored most significant byte first. Note that this is different from the memory representation of data (Sec. 2.1.4).



TL/EE/8673-7

FIGURE 2-7. Index Byte Format

Some instructions require additional, "implied" immediates and/or displacements, apart from those associated with addressing modes. Any such extensions appear at the end of the instruction, in the order that they appear within the list of operands in the instruction definition (Sec. 2.2.3).

2.2.2 Addressing Modes

The CPU generally accesses an operand by calculating its Effective Address based on information available when the operand is to be accessed. The method to be used in performing this calculation is specified by the programmer as an "addressing mode."

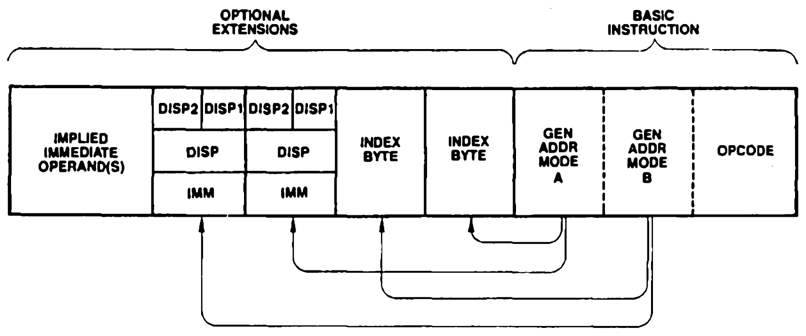
Addressing modes are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

Addressing Modes fall into nine basic types:

Register: The operand is available in one of the eight General Purpose Registers. In certain Slave Processor instructions, an auxiliary set of eight registers may be referenced instead.

Register Relative: A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.

Memory Space. Identical to Register Relative above, except that the register used is one of the dedicated registers PC, SP, SB or FP. These registers point to data areas generally needed by high-level languages.

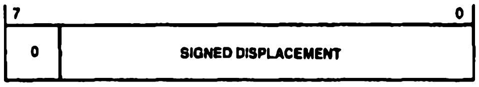


TL/EE/8673-6

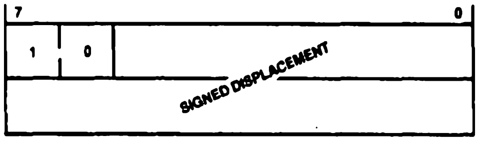
FIGURE 2-6. General Instruction Format

2.0 Architectural Description (Continued)

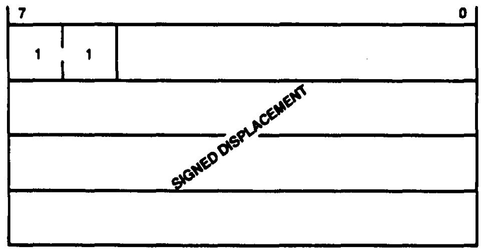
BYTE DISPLACEMENT: RANGE −64 TO +63



WORD DISPLACEMENT: RANGE −8192 TO +8191



DOUBLE WORD DISPLACEMENT: RANGE $-(2^{29}-2^{24})$ to $+(2^{29}-1)$*



TL/EE/8673-8

FIGURE 2-8. Displacement Encodings

*Note: The pattern "11100000" for the most significant byte of the displacement is reserved by National for future enhancements. Therefore, it should never be used by the user program. This causes the lower limit of the displacement range to be $-(2^{29}-2^{24})$ instead of -2^{29}.

Memory Relative: A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

Immediate: The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written.

Absolute: The address of the operand is specified by a displacement field in the instruction.

External: A pointer value is read from a specified entry of the current Link Table. To this pointer value is added a displacement, yielding the Effective Address of the operand.

Top of Stack: The currently-selected Stack Pointer (SP0 or SP1) specifies the location of the operand. The operand is pushed or popped, depending on whether it is written or read.

Scaled Index: Although encoded as an addressing mode. Scaled Indexing is an option on any addressing mode except Immediate or another Scaled Index. It has the effect of calculating an Effective Address, then multiplying any General Purpose Register by 1, 2, 4 or 8 and adding it into the total, yielding the final Effective Address of the operand.

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Instruction Set Reference Manual.

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the Series 32000 instruction set. The Format column refers to the Instruction Format tables (Appendix A). The Instruction column gives the instruction as coded in assembly language, and the Description column provides a short description of the function provided by that instruction. Further details of the exact operations performed by each instruction may be found in the Instruction Set Reference Manual.

Notations:

i = Integer length suffix: B = Byte
W = Word
D = Double Word

f = Floating Point length suffix: F = Standard Floating
L = Long Floating

gen = General operand. Any addressing mode can be specified.

short = A 4-bit value encoded within the Basic Instruction (see Appendix A for encodings).

imm = Implied immediate operand. An 8-bit value appended after any addressing extensions.

disp = Displacement (addressing constant): 8, 16 or 32 bits. All three lengths legal.

reg = Any General Purpose Register: R0–R7.

areg = Any Dedicated/Address Register: SP, SB, FP, MOD, INTBASE, PSR, US (bottom 8 PSR bits).

mreg = Any Memory Management Status/Control Register.

creg = A Custom Slave Processor Register (Implementation Dependent).

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction (see Appendix A for encodings).

2.0 Architectural Description (Continued)

TABLE 2-1
NS32332 Addressing Modes

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
Register			
00000	Register 0	R0 or F0	None: Operand is in the specified register
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
Register Relative			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
Memory Relative			
10000	Frame memory relative	disp2(disp1(FP))	Disp2 + Pointer; Pointer found at address Disp1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1(SP))	
10010	Static memory relative	disp2(disp1(SB))	
Reserved			
10011	(Reserved for Future Use)		
Immediate			
10100	Immediate	value	None: Operand is input from instruction queue.
Absolute			
10101	Absolute	@disp	Disp.
External			
10110	External	EXT (disp1) + disp2	Disp2 + Pointer; Pointer is found at Link Table Entry number Disp1.
Top of Stack			
10111	Top of stack	TOS	Top of current stack, using either User or Interrupt Stack Pointer, as selected in PSR. Automatic Push/Pop included.
Memory Space			
11000	Frame memory	disp(FP)	Disp + Register; "SP" is either SP0 or SP1, as selected in PSR.
11001	Stack memory	disp(SP)	
11010	Static memory	disp(SB)	
11011	Program memory	* + disp	
Scaled Index			
11100	Index, bytes	mode[Rn:B]	EA (mode) + Rn.
11101	Index, words	mode[Rn:W]	EA (mode) + 2× Rn.
11110	Index, double words	mode[Rn:D]	EA (mode) + 4× Rn.
11111	Index, quad words	mode[Rn:Q]	EA (mode) + 8 × Rn. 'Mode' and 'n' are contained within the Index Byte. EA (mode) denotes the effective address generated using mode.

2.0 Architectural Description (Continued)

TABLE 2-2
Series 32000 Instruction Set Summary

MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a signed 4-bit constant.
7	MOVMi	gen,gen,disp	Move Multiple: disp bytes (1 to 16).
7	MOVZBW	gen,gen	Move with zero extension.
7	MOVZiD	gen,gen	Move with zero extension.
7	MOVXBW	gen,gen	Move with sign extension.
7	MOVXiD	gen,gen	Move with sign extension.
4	ADDR	gen,gen	Move Effective Address.

INTEGER ARITHMETIC

Format	Operation	Operands	Description
4	ADDI	gen,gen	Add.
2	ADDQi	short,gen	Add signed 4-bit constant.
4	ADDCi	gen,gen	Add with carry.
4	SUBi	gen,gen	Subtract.
4	SUBCi	gen,gen	Subtract with carry (borrow).
6	NEGi	gen,gen	Negate (2's complement).
6	ABSi	gen,gen	Take absolute value.
7	MULi	gen,gen	Multiply
7	QUOi	gen,gen	Divide, rounding toward zero.
7	REMI	gen,gen	Remainder from QUO.
7	DIVi	gen,gen	Divide, rounding down.
7	MODi	gen,gen	Remainder from DIV (Modulus).
7	MEIi	gen,gen	Multiply to Extended Integer.
7	DEIi	gen,gen	Divide Extended Integer.

PACKED DECIMAL (BCD) ARITHMETIC

Format	Operation	Operands	Description
6	ADDPi	gen,gen	Add Packed.
6	SUBPi	gen,gen	Subtract Packed.

INTEGER COMPARISON

Format	Operation	Operands	Description
4	CMPi	gen,gen	Compare.
2	CMPQi	short,gen	Compare to signed 4-bit constant.
7	CMPMi	gen,gen,disp	Compare Multiple: disp bytes (1 to 16).

LOGICAL AND BOOLEAN

Format	Operation	Operands	Description
4	ANDi	gen,gen	Logical AND.
4	ORi	gen,gen	Logical OR.
4	BICi	gen,gen	Clear selected bits.
4	XORi	gen,gen	Logical Exclusive OR.
6	COMi	gen,gen	Complement all bits.
6	NOTi	gen,gen	Boolean complement: LSB only.
2	Scondi	gen	Save condition code (cond) as a Boolean variable of size i.

2.0 Architectural Description (Continued)

SHIFTS

Format	Operation	Operands	Description
6	LSHi	gen,gen	Logical Shift, left or right.
6	ASHi	gen,gen	Arithmetic Shift, left or right.
6	ROTi	gen,gen	Rotate, left or right.

BITS

Format	Operation	Operands	Description
4	TBITi	gen,gen	Test bit.
6	SBITi	gen,gen	Test and set bit.
6	SBITIi	gen,gen	Test and set bit, interlocked
6	CBITi	gen,gen	Test and clear bit.
6	CBITIi	gen,gen	Test and clear bit, interlocked.
6	IBITi	gen,gen	Test and invert bit.
8	FFSi	gen,gen	Find first set bit

BIT FIELDS

Bit fields are values in memory that are not aligned to byte boundaries. Examples are PACKED arrays and records used in Pascal. "Extract" instructions read and align a bit field. "Insert" instructions write a bit field from an aligned source.

Format	Operation	Operands	Description
8	EXTi	reg,gen,gen,disp	Extract bit field (array oriented).
8	INSi	reg,gen,gen,disp	Insert bit field (array oriented).
7	EXTSi	gen,gen,imm,imm	Extract bit field (short form).
7	INSSi	gen,gen,imm,imm	Insert bit field (short form).
8	CVTP	reg,gen,gen	Convert to Bit Field Pointer.

ARRAYS

Format	Operation	Operands	Description
8	CHECKi	reg,gen,gen	Index bounds check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.

STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 - Comparison Value

R3 - Translation Table Pointer

R2 - String 2 Pointer

R1 - String 1 Pointer

R0 - Limit Count

Options on all string instructions are:

B (Backward):	Decrement string pointers after each step rather than incrementing.
U (Until match):	End instruction if String 1 entry matches R4.
W (While match):	End instruction if String 1 entry does not match R4.

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	Descriptions
5	MOVSi	options	Move String 1 to String 2.
	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare String 1 to String 2.
	CMPST	options	Compare translating, String 1 bytes.
5	SKPSi	options	Skip over String 1 entries
	SKPST	options	Skip, translating bytes for Until/While.

2.0 Architectural Description (Continued)

JUMPS AND LINKAGE

Format	Operation	Operands	Description
3	JUMP	gen	Jump.
0	BR	disp	Branch (PC Relative).
0	Bcond	disp	Conditional branch.
3	CASEi	gen	Multiway branch.
2	ACBi	short,gen,disp	Add 4-bit constant and branch if non-zero.
3	JSR	gen	Jump to subroutine.
1	BSR	disp	Branch to subroutine.
1	CXP	disp	Call external procedure.
3	CXPD	gen	Call external procedure using descriptor.
1	SVC		Supervisor Call.
1	FLAG		Flag Trap.
1	BPT		Breakpoint Trap.
1	ENTER	[reg list],disp	Save registers and allocate stack frame (Enter Procedure).
1	EXIT	[reg list]	Restore registers and reclaim stack frame (Exit Procedure).
1	RET	disp	Return from subroutine.
1	RXP	disp	Return from external procedure call.
1	RETT	disp	Return from trap. (Privileged)
1	RETI		Return from interrupt. (Privileged)

CPU REGISTER MANIPULATION

Format	Operation	Operands	Description
1	SAVE	[reg list]	Save General Purpose Registers.
1	RESTORE	[reg list]	Restore General Purpose Registers.
2	LPRi	areg,gen	Load Dedicated Register. (Privileged if PSR or INTBASE)
2	SPRi	areg,gen	Store Dedicated Register. (Privileged if PSR or INTBASE)
3	ADJSPi	gen	Adjust Stack Pointer.
3	BISPSRi	gen	Set selected bits in PSR. (Privileged if not Byte length)
3	BICPSRi	gen	Clear selected bits in PSR. (Privileged if not Byte length)
5	SETCFG	[option list]	Set Configuration Register. (Privileged)

FLOATING POINT

Format	Operation	Operands	Description
11	MOVf	gen,gen	Move a Floating Point value.
9	MOVLF	gen,gen	Move and shorten a Long value to Standard.
9	MOVFL	gen,gen	Move and lengthen a Standard value to Long.
9	MOVif	gen,gen	Convert any integer to Standard or Long Floating.
9	ROUNDfi	gen,gen	Convert to integer by rounding.
9	TRUNCfi	gen,gen	Convert to integer by truncating, toward zero.
9	FLOORfi	gen,gen	Convert to largest integer less than or equal to value.
11	ADDf	gen,gen	Add.
11	SUBf	gen,gen	Subtract.
11	MULf	gen,gen	Multiply.
11	DIVf	gen,gen	Divide.
11	CMPf	gen,gen	Compare.
11	NEGf	gen,gen	Negate.
11	ABSf	gen,gen	Take absolute value.
12	POLYf	gen,gen	Polynomial Step.
12	DOTf	gen,gen	Dot Product.
12	SCALBf	gen,gen	Binary Scale.
12	LOGBf	gen,gen	Binary Log.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

2.0 Architectural Description (Continued)

MEMORY MANAGEMENT

Format	Operation	Operands	Description
14	LMR	mreg,gen	Load Memory Management Register. (Privileged)
14	SMR	mreg,gen	Store Memory Management Register. (Privileged)
14	RDVAL	gen	Validate address for reading. (Privileged)
14	WRVAL	gen	Validate address for writing. (Privileged)
8	MOVSUi	gen,gen	Move a value from Supervisor Space to User Space. (Privileged)
8	MOVUSi	gen,gen	Move a value from User Space to Supervisor Space. (Privileged)

MISCELLANEOUS

Format	Operation	Operands	Description
1	NOP		No Operation.
1	WAIT		Wait for interrupt.
1	DIA		Diagnose. Single-byte "Branch to Self" for hardware breakpointing. Not for use in programming.

CUSTOM SLAVE

Format	Operation	Operands	Description
15.5	CCAL0c	gen,gen	Custom Calculate.
15.5	CCAL1c	gen,gen	
15.5	CCAL2c	gen,gen	
15.5	CCAL3c	gen,gen	
15.5	CMOV0c	gen,gen	Custom Move.
15.5	CMOV1c	gen,gen	
15.5	CMOV2c	gen,gen	
15.5	CMOV3c	gen,gen	
15.5	CCMP0c	gen,gen	Custom Compare.
15.5	CCMP1c	gen,gen	
15.1	CCV0ci	gen,gen	Custom Convert.
15.1	CCV1ci	gen,gen	
15.1	CCV2ci	gen,gen	
15.1	CCV3ic	gen,gen	
15.1	CCV4DQ	gen,gen	
15.1	CCV5QD	gen,gen	
15.1	LCSR	gen	Load Custom Status Register.
15.1	SCSR	gen	Store Custom Status Register.
15.0	CATST0	gen	Custom Address/Test. (Privileged)
15.0	CATST1	gen	(Privileged)
15.0	LCR	creg,gen	Load Custom Register. (Privileged)
15.0	SCR	creg,gen	Store Custom Register. (Privileged)

3.0 Functional Description

The following is a functional description of the NS32332 CPU.

3.1 POWER AND GROUNDING

The NS32332 requires a single 5-volt power supply, applied on 7 pins. The Logic Voltage pins ($V_{CC}L1$ and $V_{CC}L2$) supply the power to the on-chip logic. The Buffer Voltage pins (V_{CCB1} to V_{CCB5}) supply the power to the output drivers of the chip. The Logic Voltage pins and the Buffer Voltage pins should be connected together by a power (V_{CC}) plane on the printed circuit board.

The NS32332 grounding connections are made on 8 pins. The Logic Ground pins (GNDL1 and GNDL2) are the ground pins for the on-chip logic. The Buffer Ground pins (GNDB1 to GNDB6) are the ground pins for the output drivers of the chip. The Logic Ground pins and the Buffer Ground pins should be connected together by a ground plane on the printed circuit board.

In addition to V_{CC} and Ground, the NS32332 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (*Figure 3.1*) from the BBG pin to Ground.

Recommended values for these are:

C1: 1 μF, Tantalum

C2: 1000 pF, Low inductance. This should be either a disc or monolithic capacitor.

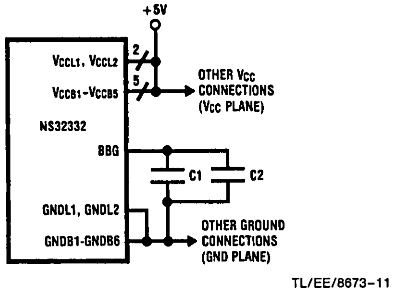


FIGURE 3-1. Recommended Supply Connections

3.2 CLOCKING

The NS32332 inputs clocking signals from the Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin A7) and PHI2 (pin B8). Their relationship to each other is shown in *Figure 3-2*.

Each rising edge of PHI1 defines a transition in the timing state ("T-State") of the CPU. One T-State represents the execution of one microinstruction within the CPU, and/or one step of an external bus transfer. See Sec. 4 for complete specifications of PHI1 and PHI2.

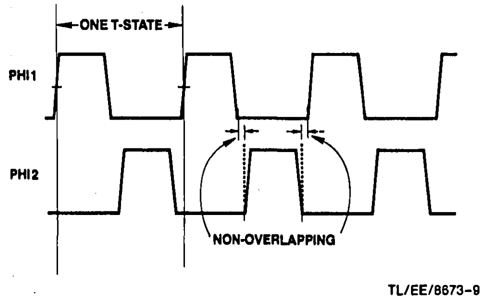


FIGURE 3-2. Clock Timing Relationships

As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected anywhere except from the TCU to the CPU and, if present, the MMU. A TTL Clock signal (CTTL) is provided by the TCU for all other clocking.

3.3 RESETTING

The $\overline{\text{RST}}/\overline{\text{ABT}}$ pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Sec. 3.5.2.

The $\overline{\text{DT}}/\overline{\text{SDONE}}$ pin is sampled on the rising edge of PHI1, one cycle before the reset signal is deasserted to select the data timing during write cycles. If $\overline{\text{DT}}/\overline{\text{SDONE}}$ is sampled high, AD0–AD31 are floated during state T2 and the data is output during state T3. This mode must be selected if an MMU is used (Section 3.5). If $\overline{\text{DT}}/\overline{\text{SDONE}}$ is sampled low, the data is output during state T2. See *Figure 3-7*.

The CPU may be reset at any time by pulling the $\overline{\text{RST}}/\overline{\text{ABT}}$ pin low for at least 64 clock cycles. Upon detecting a reset, the CPU terminates instruction processing, resets its internal logic, and clears the Program Counter (PC) and Processor Status Register (PSR) to all zeroes.

On application of power, $\overline{\text{RST}}/\overline{\text{ABT}}$ must be held low for at least 50 μsec after V_{CC} is stable. This is to ensure that all

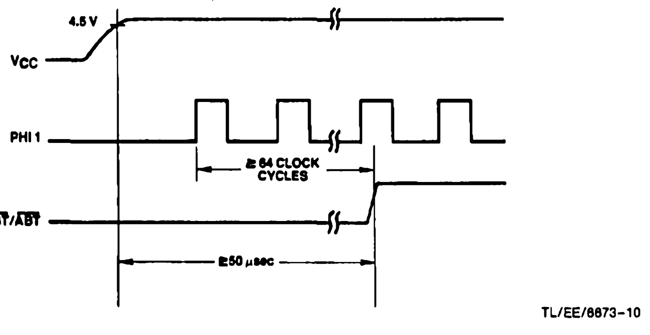
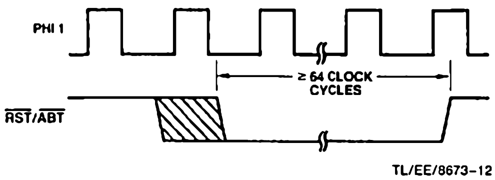


FIGURE 3-3. Power-on Reset Requirements

3.0 Functional Description (Continued)

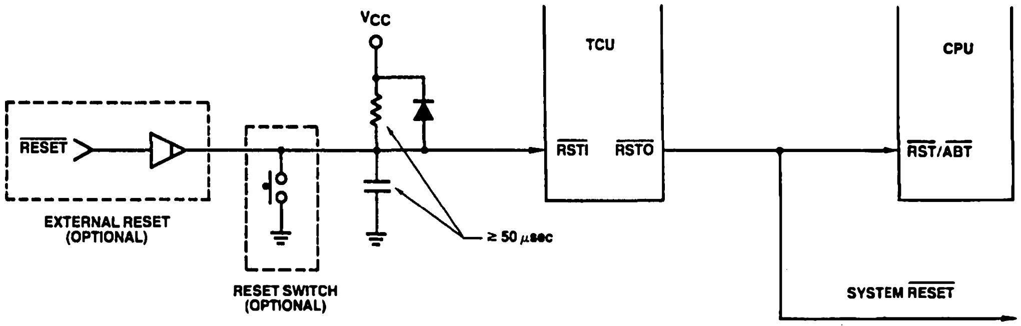
on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain active for not less than 64 clock cycles. See *Figures 3-3* and *3-4*.

The Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32332 CPU. *Figure 3-5a* shows the recommended connections for a non-Memory-Managed system. *Figure 3-5b* shows the connections for a Memory-Managed system.



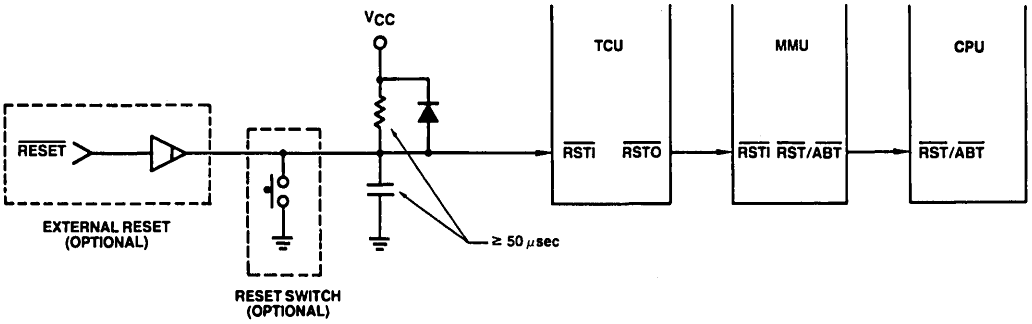
TL/EE/8673-12

FIGURE 3-4. General Reset Timing



TL/EE/8673-13

FIGURE 3-5a. Recommended Reset Connections, Non-Memory-Managed System



TL/EE/8673-14

FIGURE 3-5b. Recommended Reset Connections, Memory-Managed System

3.4 BUS CYCLES

The NS32332 CPU will perform Bus cycles for one of the following reasons:

1) To write or read data to or from memory or peripheral interface device. Peripheral input and output are memory mapped in the Series 32000 family.
2) To fetch instructions into the 20-byte instruction queue. This happens whenever the bus would otherwise be idle and the queue is not already full.
3) To acknowledge an interrupt and allow external circuitry to provide a vector number, or to acknowledge completion of an interrupt service routine.
4) To transfer information to or from a Slave Processor.

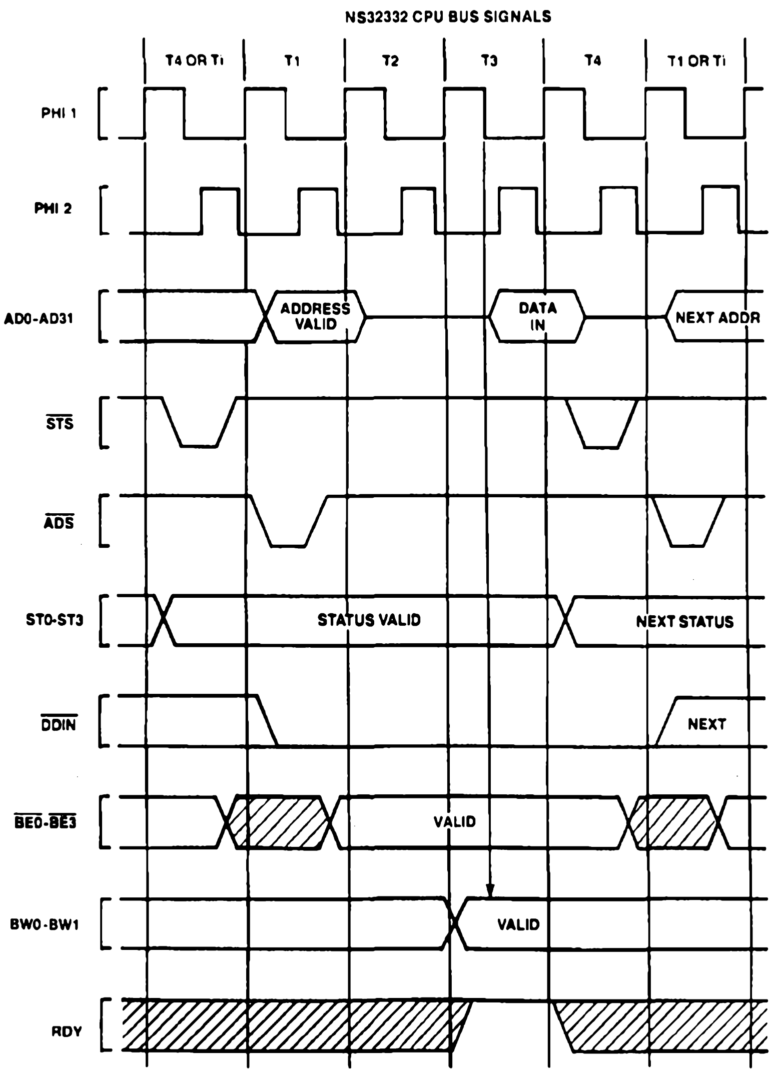
In terms of bus timing, cases 1 through 3 above are identical. For timing specifications, see Sec. 4. The only external difference between them is the 4-bit code placed on the Bus Status pins (ST0–ST3). Slave Processor cycles differ in that separate control signals are applied (Sec. 3.4.6).

For case 1 (only Read) and case 2, the NS32332 supports Burst cycles which are suitable for memories that can handle "nibble mode" accesses. (Sec. 3.4.2).

The sequence of events in a non-Slave, non-Burst Bus cycle is shown in *Figure 3-6* for a Read cycle, and *Figure 3-7* for a Write cycle. The cases shown assume that the selected memory or interface device is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Sec. 3.4.1).

A full speed Bus cycle is performed in four cycles of the PHI1 clock, labeled T1 through T4. Clock cycles not associated with a Bus cycle are designated Ti (for idle).

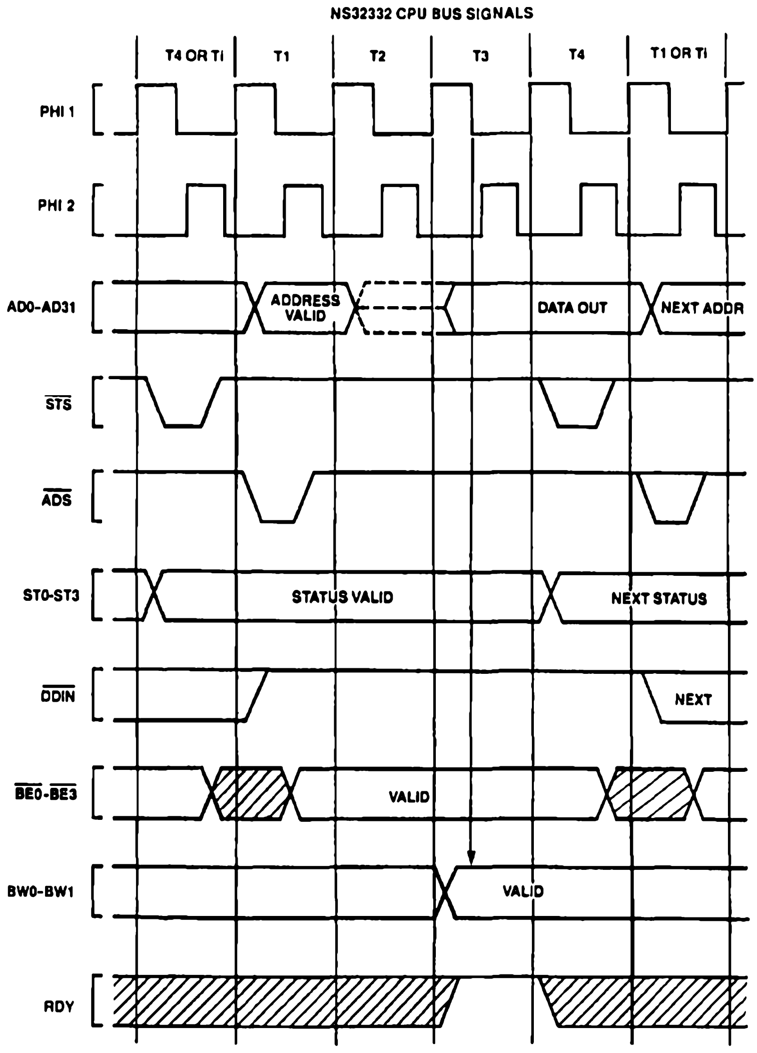
3.0 Functional Description (Continued)



TL/EE/8673-15

FIGURE 3-6. Read Cycle Timing

3.0 Functional Description (Continued)



TL/EE/8673-16

FIGURE 3-7. Write Cycle Timing

3.0 Functional Description (Continued)

During T4 or Ti which preceed T1 of the current Bus cycle, the CPU applies a Status Code on pins ST0-ST3. It also provides a low-going pulse on the $\overline{\text{STS}}$ pin to indicate that the status code is valid.

The $\overline{\text{ADS}}$ signal has the dual purpose of informing the external circuitry that a Bus cycle is starting and of providing control to an external latch for demultiplexing address bits 0–31 from AD0–AD31 pins. (See *Figure 3-8.*)

During this time, the control signal $\overline{\text{DDIN}}$, which indicates the direction of the transfer, and $\overline{\text{BE0}}$–$\overline{\text{BE3}}$ which indicate which of the four bus bytes to be referenced, become valid. Note that during Instruction Fetch cycles $\overline{\text{BE0}}$–$\overline{\text{BE3}}$ are all active, but in operand Read or Write cycles they indicate the byte(s) to be referenced.

Note: If a burst cycle occurs during an operand read, all the memory banks should be enabled, during the burst cycle, regardless of $\overline{\text{BE}}$n. The CPU $\overline{\text{BE}}$n lines, in this case, are valid in the middle of T3 of the burst cycle—thus, there may not be enough time to selectively enable the different memory banks, unless a WAIT state is added. See *Figure 4-6.*

During T2 the CPU floats AD0–AD31 lines unless $\overline{\text{DT}}/\overline{\text{SDONE}}$ is sampled low on the rising edge of reset and the bus cycle is a write cycle. T2 is a time window to be used for virtual to physical address translation by the Memory Management Unit, if virtual memory is used in the system.

The T3 state provides for access time requirements and it occurs at least once in a bus cycle. In the middle of T3 on the falling edge of PHI1, the RDY line is sampled to determine whether the bus cycle will be extended (Sec. 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0–AD31) is sampled on the falling edge of PHI2 of the last T3 state. See Sec. 4. Data must, however, be held at least until the beginning of T4. The T4 state finishes the Bus cycle. Data from the CPU during Write cycles remains valid throughout T4. Note that the Bus Status lines (ST0–ST3) change at the beginning of T4, anticipating the following bus cycle (if any).

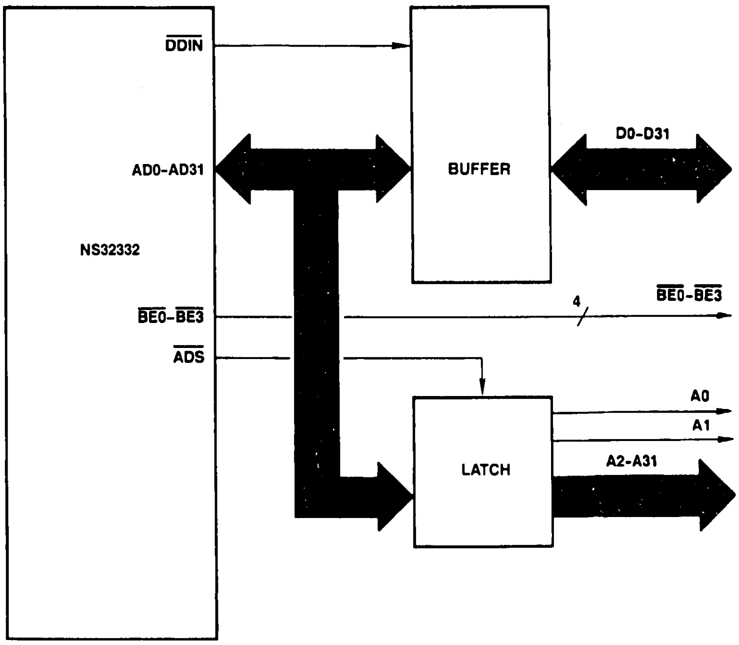
3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS32332 provides for extension of a bus cycle. Any type of bus cycle except a Slave Processor cycle can be extended.

In *Figures 3-7* and *3-8*, note that during T3 all bus control signals from the CPU and TCU are flat. Therefore, a bus cycle can be cleanly extended by causing the T3 state to be repeated. This is the purpose of the RDY (Ready) pin.

In the middle of T3 on the falling edge of PHI1, the RDY line is sampled by the CPU. If RDY is high, the next T-state will be T4, ending the bus cycle. If RDY is low, then another T3 state will be inserted and the RDY line will again be sampled on the falling edge of PHI1. Each additional T3 state after the first is referred to as a "WAIT STATE". See *Figure 3-9.*

Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the RDY pin.

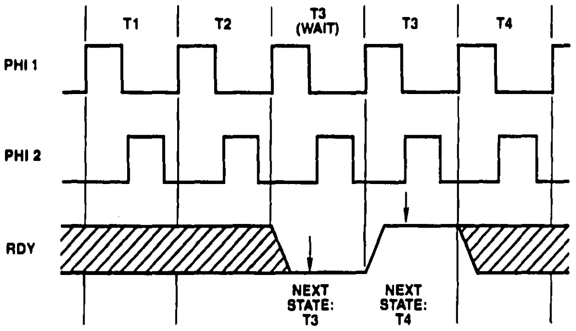


$\overline{\text{DDIN}}$
AD0-AD31
NS32332
$\overline{\text{BE0}}$-$\overline{\text{BE3}}$
$\overline{\text{ADS}}$
BUFFER
D0-D31
4
$\overline{\text{BE0}}$-$\overline{\text{BE3}}$
LATCH
A0
A1
A2-A31

TL/EE/8673-17

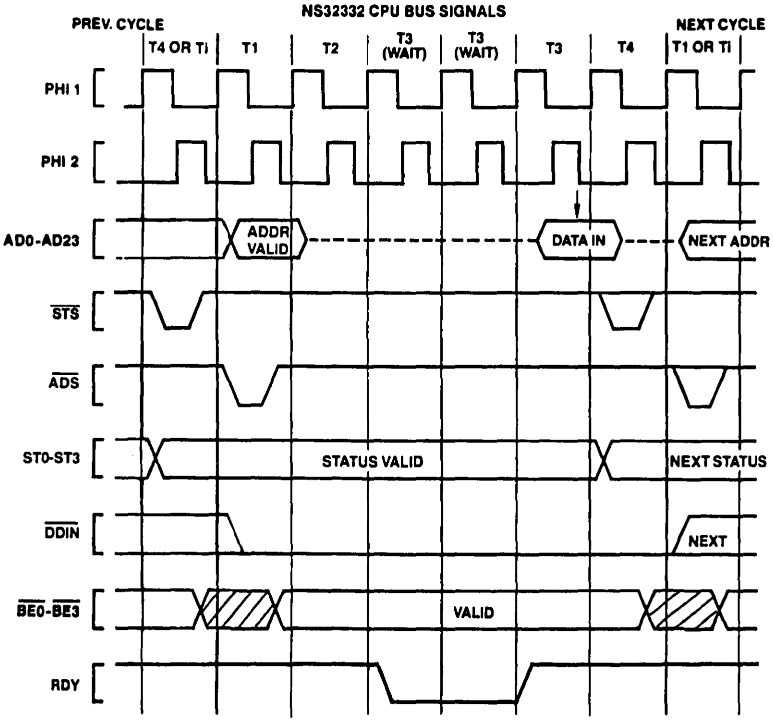
FIGURE 3-8. Bus Connections

3.0 Functional Description (Continued)



TL/EE/8673-18

FIGURE 3-9. RDY Pin Timing



TL/EE/8673-19

FIGURE 3-10. Extended Cycle Example

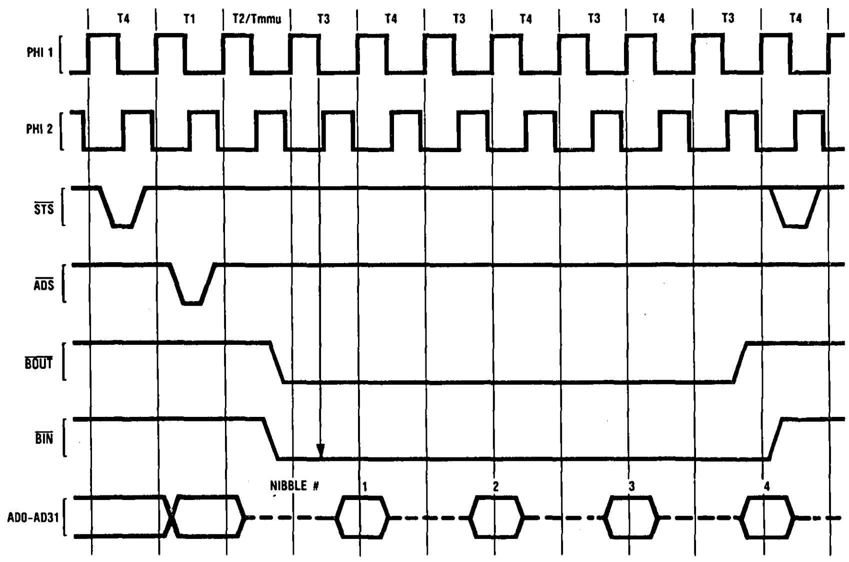
3.0 Functional Description (Continued)

3.4.2 Burst Cycles

The NS32332 is capable of performing Burst cycles in order to increase the bus throughput. Burst is available in instruction Fetch cycles and operand Read cycles only. Burst is not supported in operand Write cycles or Slave cycles.

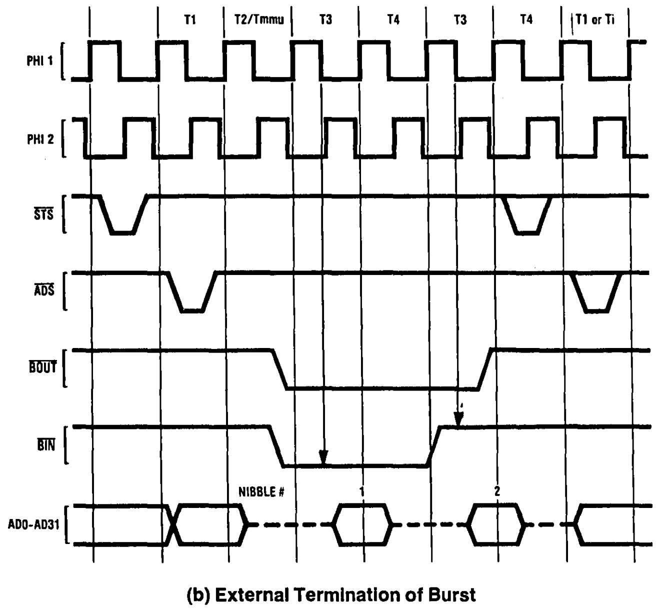
The sequence of events for Burst cycles is shown in *Figure 3-11.* The cases shown assume that the selected memory is capable of communicating with the CPU at full speed. If it is not, then cycle extension may be requested through the RDY line (Sec. 3.4.1).

A Burst cycle is composed of two parts. The first part is a regular cycle (i.e. T1 through T4), in which the CPU outputs the new status and asserts all the other relevant control signals discussed in Sec. 3.4. In addition, the Burst Out Signal ($\overline{BOUT}$) is activated by the CPU indicating that the CPU can perform Burst cycles. If the selected memory allows



TL/EE/8673-20

(a) Normal Termination of Burst



TL/EE/8673-21

(b) External Termination of Burst

FIGURE 3-11. Burst Cycles (For Read Only)

3.0 Functional Description (Continued)

Burst cycles, it will notify the CPU by activating the burst in signal ($\overline{BIN}$). $\overline{BIN}$ is sampled by the CPU in the middle of T3 on the falling edge of PHI1. If the memory does not allow burst ($\overline{BIN}$ high), the cycle will terminate through T4 and $\overline{BOUT}$ will go inactive immediately. If the memory allows burst ($\overline{BIN}$ low), and the CPU has not deasserted $\overline{BOUT}$, the second part of the Burst cycle will be performed (see *Figure 3-11*) and $\overline{BOUT}$ will remain active until termination of the Burst.

The second part consists of up to 3 nibbles. In each nibble, a data item is read by the CPU. The duration of each nibble is 2 clock cycles labeled T3 and T4.

The Burst chain will be terminated in the following cases:

1. The CPU has reached a 16 byte boundary i.e. the byte address of the current nibble is x...x1111 (binary).
2. The CPU detects that the instructions being prefetched (in Burst Mode) are no longer needed due to an alteration of the flow of control. This happens, for example, when a branch instruction is executed or an exception occurs.

Note: In 16-bit bus systems (see Sec. 3.4.7) the Burst chain will be terminated by the CPU on an 8-byte boundary i.e. address x..x111 (binary) and in 8-bit bus system on a 4-byte boundary i.e. address x...x11 (binary).

3. The data operand has been completely read. This applies to burst read cycles for non-aligned operands or when the bus width is either 8 or 16 bits.
4. $\overline{BIN}$, sampled in the current nibble's last T3, is not active any more. (See *Figure 3.11b*).
5. Bus Error or Bus Retry occurs (see Sec. 3.4.8).
6. A $\overline{HOLD}$ Request occurs.

Any nibble's T3 may be extended with WAIT states using the RDY line as described in Sec. 3.4.2.

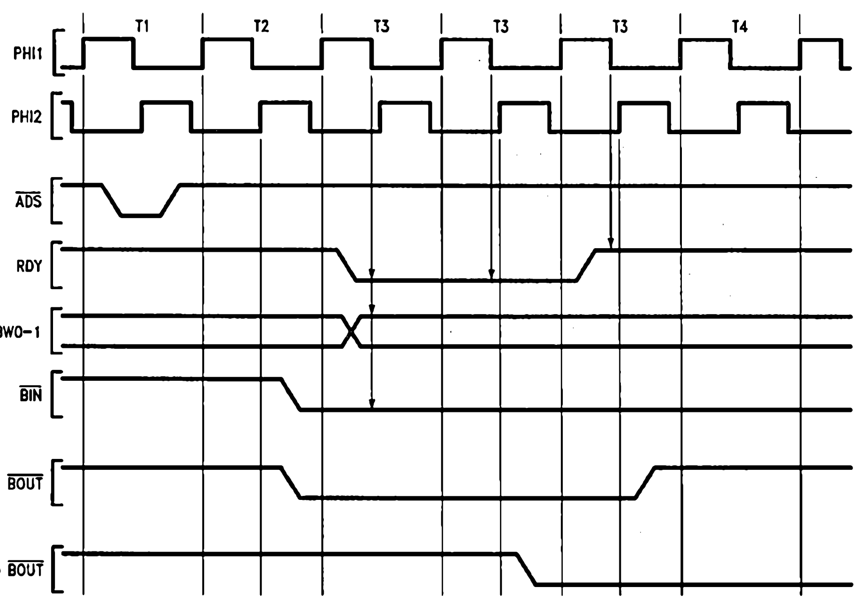
The control signals $\overline{BOUT}$, ST0–ST3, and $\overline{DDIN}$ remain stable during the Burst chain.

$\overline{BE0}$–$\overline{BE3}$ are adjusted for every nibble in operand cycles.

$\overline{BOUT}$ is initially set by the CPU according to the known bus width. Its state may change in a subsequent T3 as a result of a change in the bus width. *Figure 3-12* shows the resulting $\overline{BOUT}$ timing.

Note: If the selected memory is capable of handling burst transfers, it should activate $\overline{BIN}$ regardless of the state of $\overline{BOUT}$.

The reason is that $\overline{BOUT}$ may be activated by the CPU after the $\overline{BIN}$ sampling time. The $\overline{BOUT}$ signal indicates when the CPU is going to burst, and should not be interpreted as a 'Burst Request' signal.



TL/EE/8673-88

Note 1: CPU deasserts $\overline{BOUT}$.

Note 2: CPU asserts $\overline{BOUT}$.

FIGURE 3-12. $\overline{BOUT}$ Timing Resulting from a Bus Width Change

3.0 Functional Description (Continued)

3.4.3 Bus Status

The NS32332 CPU presents four bits of Bus Status information on pins ST0–ST3. The various combinations on these pins indicate why the CPU is performing a bus cycle, or, if it is idle on the bus, then why is it idle.

Referring to *Figures 3-6* and *3-7*, note that Bus Status leads the corresponding Bus Cycle, going valid one clock cycle before T1, and changing to the next state at T4. This allows the system designer to fully decode the Bus Status and, if desired, latch the decoded signals before $\overline{\text{ADS}}$ initiates the Bus Cycle.

The Bus Status pins are interpreted as a four-bit value, with ST0 the least significant bit. Their values decode as follows:

0000 – The bus is idle because the CPU does not yet need to perform a bus access.

0001 – The bus is idle because the CPU is executing the WAIT instruction.

0010 – (Reserved for future use.)

0011 – The bus is idle because the CPU is waiting for a Slave Processor to complete an instruction.

0100 – Interrupt Acknowledge, Master.

The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{\text{NMI}}$) it will read from address FFFFFF00_{16}, but will ignore any data provided.

To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$) it will read from address FFFFFE00_{16}, expecting a vector number to be provided from the Master Interrupt Control Unit. If the vectoring mode selected by the last SETCFG instruction was Non-Vectored, then the CPU will ignore the value it has read and will use a default vector instead. See Sec. 3.4.5.

0101 – Interrupt Acknowledge, Cascaded.

The CPU is reading a vector number from a Cascaded Interrupt Control Unit. The address provided is the address of ICU's Hardware Vector register. See Sec. 3.4.6.

0110 – End of Interrupt, Master.

The CPU is performing a Read cycle to indicate that it is executing a Return from Interrupt (RETI) instruction. See Sec. 3.4.6.

0111 – End of Interrupt, Cascaded.

The CPU is reading from a Cascaded Interrupt Control Unit to indicate that it is returning (through RETI) from an interrupt service routine requested by that unit. See Sec. 3.4.6.

1000 – Sequential Instruction Fetch.

The CPU is reading the next sequential word from the instruction stream into the Instruction Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

1001 – Non-Sequential Instruction Fetch.

The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

1010 – Data Transfer.

The CPU is reading or writing an operand of an instruction.

1011 – Read RMW Operand.

The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.

1100 – Read for Effective Address Calculation.

The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

1101 – Transfer Slave Processor Operand.

The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Sec. 3.9.1.

1110 – Read Slave Processor Status.

The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Sec. 3.9.1.

1111 – Broadcast Slave ID.

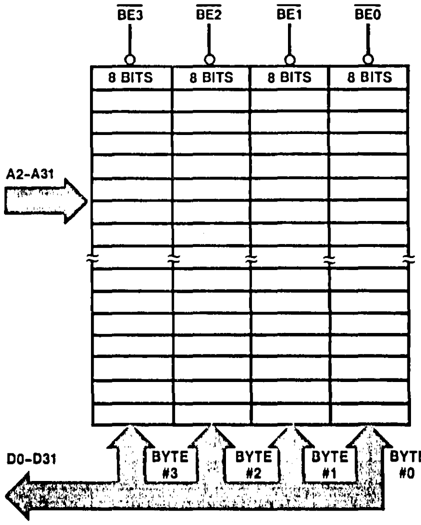
The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Sec. 3.9.1.

3.0 Functional Description (Continued)

3.4.4 Data Access Sequences

The 32-bit address provided by the NS32332 is a byte address; that is, it uniquely identifies one of up to 4 billion eight-bit memory locations. An important feature of the NS32332 is that the presence of a 32-bit data bus imposes no restrictions on data alignment; any data item, regardless of size, may be placed starting at any memory address. The NS32332 provides special control signals. Byte Enable ($\overline{BE0}-\overline{BE3}$) which facilitate individual byte accessing on a 32-bit bus.

Memory is organized as four eight-bit banks, each bank receiving the double-word address (A2–A31) in parallel. One bank, connected to Data Bus pins AD0–AD7 is enabled when $\overline{BE0}$ is low. The second bank, connected to data bus pins AD8–AD15 is enabled when $\overline{BE1}$ is low. The third and fourth banks are enabled by $\overline{BE2}$ and $\overline{BE3}$, respectively. See *Figure 3-13.*



TL/EE/8673-22

FIGURE 3-13. Memory Interface

Since operands do not need to be aligned with respect to the double-word bus access performed by the CPU, a given double-word access can contain one, two, three, or four bytes of the operand being addressed, and these bytes can begin at various positions, as determined by A1, A0. Table 3-1 lists the 10 resulting access types.

TABLE 3-1

Bus Access Types

Type	Bytes Accessed	A1,A0	$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$
1	1	00	1	1	1	0
2	1	01	1	1	0	1
3	1	10	1	0	1	1
4	1	11	0	1	1	1
5	2	00	1	1	0	0
6	2	01	1	0	0	1
7	2	10	0	0	1	1
8	3	00	1	0	0	0
9	3	01	0	0	0	1
10	4	00	0	0	0	0

Accesses of operands requiring more than one bus cycle are performed sequentially, with no idle T-States separating them. The number of bus cycles required to transfer an operand depends on its size and its alignment. Table 3-2 lists the bus cycles performed for each situation.

3.4.4.1 Bit Accesses

The Bit Instructions perform byte accesses to the byte containing the designated bit. The Test and Set Bit instruction (SBIT), for example, reads a byte, alters it, and rewrites it, having changed the contents of one bit.

3.4.4.2 Bit Field Accesses

An access to a Bit Field in memory always generates a Double-Word transfer at the address containing the least significant bit of the field. The Double Word is read by an Extract instruction; an Insert instruction reads a Double Word, modifies it, and rewrites it.

3.4.4.3 Extending Multiple Accesses

The Extending Multiply Instruction (MEI) will return a result which is twice the size in bytes of the operand it reads. If the multiplicand is in memory, the most-significant half of the result is written first (at the higher address), then the least-significant half. This is done in order to support retry if this instruction is aborted.

3.4.5 Instruction Fetches

Instructions for the NS32332 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the twenty-byte Instruction Queue. The CPU performs two types of Instruction Fetch cycles: Sequential and Non-Sequential. These can be distinguished from each other by their differing status combinations on pins ST0–ST3 (Sec. 3.4.3).

A Sequential Fetch will be performed by the CPU whenever the Data Bus would otherwise be idle and the Instruction Queue is not currently full. Sequential Fetches are always type 10 Read cycles (Table 3-1).

A Non-Sequential Fetch occurs as a result of any break in the normally sequential flow of a program. Any jump or branch instruction, a trap or an interrupt will cause the next Instruction Fetch cycle to be Non-Sequential. In addition, certain instructions flush the instruction queue, causing the next instruction fetch to display Non-Sequential status. Only the first bus cycle after a break displays Non-Sequential status, and that cycle depends on the destination address.

If a non-sequential fetch is followed by additional sequential fetches which are burst continuation of the non-sequential fetch, then the Status Bus (ST0–ST3) remains the same.

Note 1: During instruction fetch cycles, $\overline{BE0}-\overline{BE3}$ are all active regardless of the alignment.

Note 2: During Operand Access cycles $\overline{BE0}-\overline{BE3}$ are activated as if the bus is 32 bits wide, regardless of the real width.

3.4.6 Interrupt Control Cycles

Activating the $\overline{INT}$ or $\overline{NMI}$ pin on the CPU will initiate one or more bus cycles whose purpose is interrupt control rather than the transfer of instructions or data. Execution of the Return from Interrupt instruction (RETI) will also cause Interrupt Control bus cycles. These differ from instruction or data transfers only in the status presented on pins ST0–ST3. All Interrupt Control cycles are single-byte Read cycles.

This section describes only the Interrupt Control sequences associated with each interrupt and with the return from its service routine.

3.0 Functional Description (Continued)

TABLE 3-2
Access Sequences

							Data Bus			
Cycle	**Type**	**Address**	$\overline{\text{BE3}}$	$\overline{\text{BE2}}$	$\overline{\text{BE1}}$	$\overline{\text{BE0}}$	**Byte 3**	**Byte 2**	**Byte 1**	**Byte 0**
A. *Word at address ending with 11*									BYTE 1	BYTE 0 ← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	1	A + 1	1	1	1	0	X	X	X	Byte 1
B. *Double word at address ending with 01*							BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	9	A	0	0	0	1	Byte 2	Byte 1	Byte 0	X
2.	1	A + 3	1	1	1	0	X	X	X	Byte 3
C. *Double word at address ending with 10*							BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	7	A	0	0	1	1	Byte 1	Byte 0	X	X
2.	5	A + 2	1	1	0	0	X	X	Byte 3	Byte 2
D. *Double word at address ending with 11*							BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	8	A + 1	1	0	0	0	X	Byte 3	Byte 2	Byte 1
E. *Quad word at address ending with 00*			BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	10	A	0	0	0	0	Byte 3	Byte 2	Byte 1	Byte 0
Other bus cycles (instruction prefetch or slave) can occur here.										
2.	10	A + 4	0	0	0	0	Byte 7	Byte 6	Byte 5	Byte 4
F. *Quad word at address ending with 01*			BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	9	A	0	0	0	1	Byte 2	Byte 1	Byte 0	X
2.	1	A + 3	1	1	1	0	X	X	X	Byte 3
Other bus cycles (instruction prefetch or slave) can occur here.										
3.	9	A + 4	0	0	0	1	Byte 6	Byte 5	Byte 4	X
4.	1	A + 7	1	1	1	0	X	X	X	Byte 7
G. *Quad word at address ending with 10*			BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	7	A	0	0	1	1	Byte 1	Byte 0	X	X
2.	5	A + 2	1	1	0	0	X	X	Byte 3	Byte 2
Other bus cycles (instruction prefetch or slave) can occur here.										
3.	7	A + 4	0	0	1	1	Byte 5	Byte 4	X	X
4.	5	A + 6	1	1	0	0	X	X	Byte 7	Byte 6
H. *Quad word at address ending with 11*			BYTE 7	BYTE 6	BYTE 5	BYTE 4	BYTE 3	BYTE 2	BYTE 1	BYTE 0 ← A
1.	4	A	0	1	1	1	Byte 0	X	X	X
2.	8	A + 1	1	0	0	0	X	Byte 3	Byte 2	Byte 1
Other bus cycles (instruction prefetch or slave) can occur here.										
1.	4	A + 4	0	1	1	1	Byte 4	X	X	X
2.	8	A + 5	1	0	0	0	X	Byte 7	Byte 6	Byte 5

X = Don't Care

3.0 Functional Description (Continued)

TABLE 3-3
Interrupt Sequences

								Data Bus			
Cycle	**Status**	**Address**	**$\overline{DDIN}$**	**$\overline{BE3}$**	**$\overline{BE2}$**	**$\overline{BE1}$**	**$\overline{BE0}$**	**Byte 3**	**Byte 2**	**Byte 1**	**Byte 0**
					A. Non-Maskable Interrupt Control Sequences						
Interrupt Acknowledge											
1	0100	FFFFFF00_{16}	0	1	1	1	0	X	X	X	X
Interrupt Return											
None: Performed through Return from Trap (RETT) instruction.											
					B. Non-Vectored Interrupt Control Sequences						
Interrupt Acknowledge											
1	0100	FFFFFE00_{16}	0	1	1	1	0	X	X	X	X
Interrupt Return											
1	0110	FFFFFE00_{16}	0	1	1	1	0	X	X	X	X
					C. Vectored Interrupt Sequences: Non-Cascaded.						
Interrupt Acknowledge											
1	0100	FFFFFE00_{16}	0	1	1	1	0	X	X	X	Vector: Range: 0-127
Interrupt Return											
1	0110	FFFFFE00_{16}	0	1	1	1	0	X	X	X	Vector: Same as in Previous Int. Ack. Cycle
					D. Vectored Interrupt Sequences: Cascaded						
Interrupt Acknowledge											
1	0100	FFFFFE00_{16}	0	1	1	1	0	X	X	X	Cascade Index: range −16 to −1
(The CPU here uses the Cascade Index to find the Cascade Address.)											
2	0101	Cascade Address	0		See Note			Vector, range 9–255; on appropriate byte of data bus.			
Interrupt Return											
1	0110	FFFFFE00_{16}	0	1	1	1	0	X	X	X	Cascade Index: Same as in previous Int. Ack. Cycle
(The CPU here uses the Cascade Index to find the Cascade Address)											
2	0111	Cascade Address	0		See Note			X	X	X	X

X = Don't Care

Note: $\overline{BE0}$-$\overline{BE3}$ signals will be activated according to the cascaded ICU address. The cycle type can be 1, 2, 3 or 4, when reading the interrupt vector. The vector value can be in the range 0–255.

3.0 Functional Description (Continued)

3.4.7 Dynamic Bus Configuration

The NS32332 interfaces to external data buses with 3 different widths: 8-bit, 16-bit and 32-bit. The NS32332 can switch from one bus width to another dynamically i.e. on a cycle by cycle basis.

This feature allows the user to include in his system different bus sizes for different purposes, like 8-bit bus for bootstrap ROM and 32-bit bus for cache memory, etc.

In each memory cycle, the bus width is determined by the inputs BW0 and BW1.

Four combinations exist:

BW1	BW0	
0	0	reserved
0	1	8-bit bus
1	0	16-bit bus
1	1	32-bit bus

The dynamic bus configuration is not applicable for slave cycles (see Sec. 3.4.1).

The BW0–BW1 lines are sampled by the CPU in T3 with the falling edge of PHI1 (see *Figure 3-14*).

If the bus width didn't change from the previous memory cycle, the CPU terminates the cycle normally.

If the bus width of the current cycle is different from the bus width of the previous cycle, then two WAIT states (see Sec. 3.4.1) must be inserted in order to let the CPU switch to the new width.

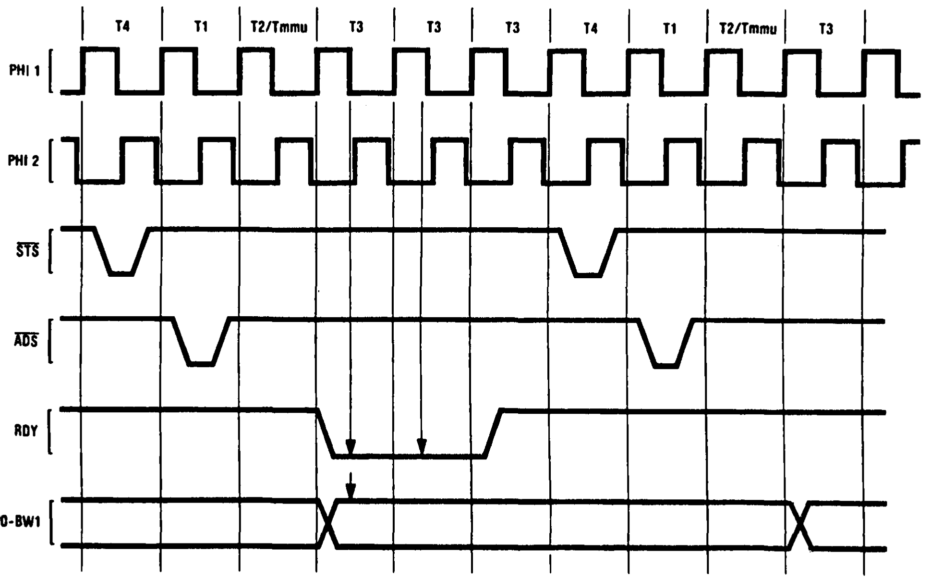
The additional 2 WAIT states count from the moment BW0 BW1 change. This can be overlapped with the wait states due to slow memories.

Note: BW0–BW1 can only be changed during the first T3 state of a memory access cycle. They should be externally latched and should not be changed at any other time.

In write cycles, the appropriate data will be present on the appropriate data lines. The CPU presents the data during T3 in a way that would fit any bus width.

If the operand being written is a byte, it will be duplicated on the 4 bytes AD0–AD31 depending on the operand address:

Address A0–1 =	00	XX	XX	XX	OP
	01	XX	XX	OP	OP
	10	XX	OP	XX	OP
	11	OP	XX	OP	OP



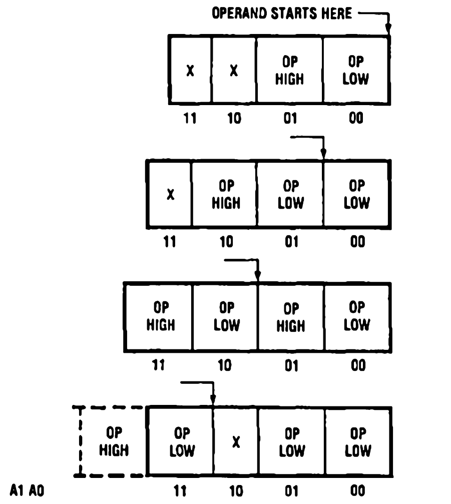
TL/EE/8673-23

FIGURE 3-14. Bus width changes. Two wait states are required after the signals BW0–BW1 change.

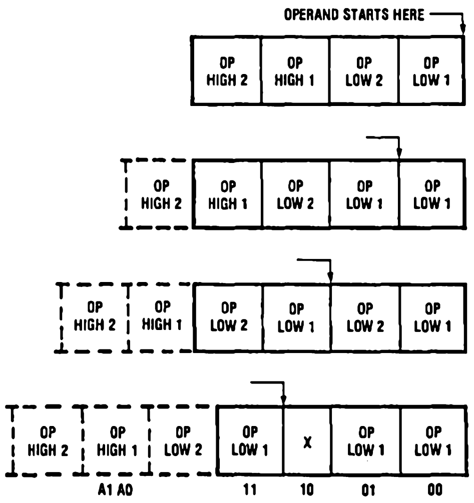
3.0 Functional Description (Continued)

If the operand being written is a word, 4 cases exist. The operand address can be x...x00 (binary) or x...x01 (binary) or x...x10 or x...x11 (binary).

See the duplications for each case:



TL/EE/8673-25



TL/EE/8673-26

If the operand being written is a double word 4 cases exist: The operand address can be x...x00 (binary) or x...x01 (binary) or x...x10 (binary) or x...x11 (binary).

See the duplications for each case:

Note that the organization of the operand described applies to the initial part of the operand cycle. For instance, if the CPU writes a double word operand to a 16-bit bus and the operand address is x...x11 (binary) it needs three memory cycles.

The description above applies to the first cycle. In the other 2 memory cycles belonging to the same operand, the data will be presented on the data bus lines to fit 16-bit bus width and take into account the operand length.

Example:

The CPU has to write a double word DDCCBBAA to address HEX 987653 which is in a 16-bit bus area. In the first cycle, the CPU does not know the width until T3 so it generates a cycle to address 987653 which activates the $\overline{BE3}$ line and puts on the data bus AA XX AA AA (X = don't care). After this cycle, the CPU knows it has a 16-bit bus and it generates a cycle to address 987654 which activates the BE0, BE1 and BE2 lines and puts on the data bus XX XX CC BB. The last cycle will address 987656, activate BE2, and put on the data bus XX XX XX DD. The BE0–BE3 lines are always activated as if the bus is 32-bit wide, regardless of BW0–BW1 state.

The CPU does not support a change of the bus width during a sequence of several memory references belonging to the same operand e.g. nonaligned double word. In other words, any operand should not be split between two memory spaces having different bus widths.

Instruction Fetches do not fall in this category and an Instruction Fetch can have its own bus width regardless of the bus width in the previous cycle.

3.4.8 Bus Exceptions

Any bus cycle may have a bus error during its execution. The error may be corrected during the current cycle or may be incorrectable. The NS32332 can handle both types of errors by means of BUS RETRY and BUS ERROR.

3.4.8.1 Bus Retry

If a bus error can be corrected, the CPU may be requested to repeat the erroneous bus cycle. The request is done by asserting the $\overline{BRT}$ (Bus Retry) signal.

The CPU response to Bus Retry depends on the cycle type:

Instruction Fetch Cycle—If the RETRY occurs during an instruction fetch, the fetch cycle will be retried as soon as possible. If the RETRY is requested during a burst chain, the burst is stopped and the fetch is retried. The only delay in retrying the instruction fetch may result from pending operand requests (and, of course, from hold or wait requests).

The fetch cycle will be retried only if there are no more than four bytes in the queue.

Operand Read Cycle—If the RETRY occurs on an operand read, the bus cycle is immediately repeated. If the data read is "multiple" e.g. non-aligned, only the problematic part will be repeated. For instance, if the cycle is a non-aligned double word and the second half failed, only the second part will be repeated. The same applies for a RETRY occurring during a burst chain. The repeated cycle will begin where the read operand failed (rather than the first address of the burst) and will finish the original burst.

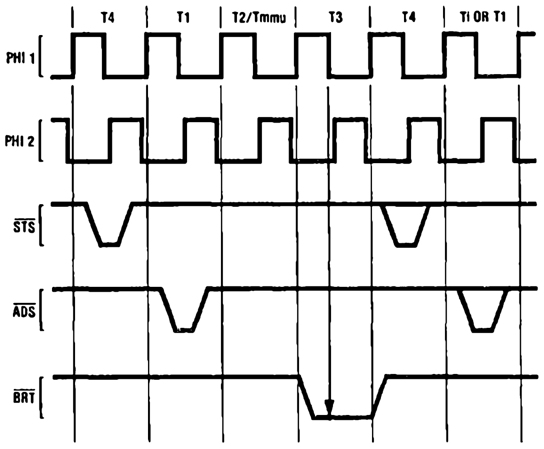
3.0 Functional Description (Continued)

Operand Write Cycle—If the RETRY occurs on a write, the bus cycle is immediately repeated. If the operand write is "multiple" e.g. non-aligned, only the problematic part will be repeated. For instance, if the cycle is a non-aligned double word and the second half failed, only the second part will be repeated.

A Bus Retry is requested by activating the $\overline{\text{BRT}}$ line (see *Figure 3-15*). $\overline{\text{BRT}}$ is sampled by the CPU during T3 on the falling edge of PHI1. If $\overline{\text{BRT}}$ is inactive, the cycle will be terminated in a regular way. In this case $\overline{\text{BRT}}$ must also be kept inactive during T4. If $\overline{\text{BRT}}$ is active, $\overline{\text{BRT}}$ will be sampled again during T4 on the falling edge of PHI1. If $\overline{\text{BRT}}$ is inactive, the cycle will be terminated in a regular way. If $\overline{\text{BRT}}$ is active, T4 will be followed by an idle state and the cycle will be repeated, i.e. a new T4 for setting the Status Bus and issuing $\overline{\text{STS}}$ and then T1 through T4 will be performed.

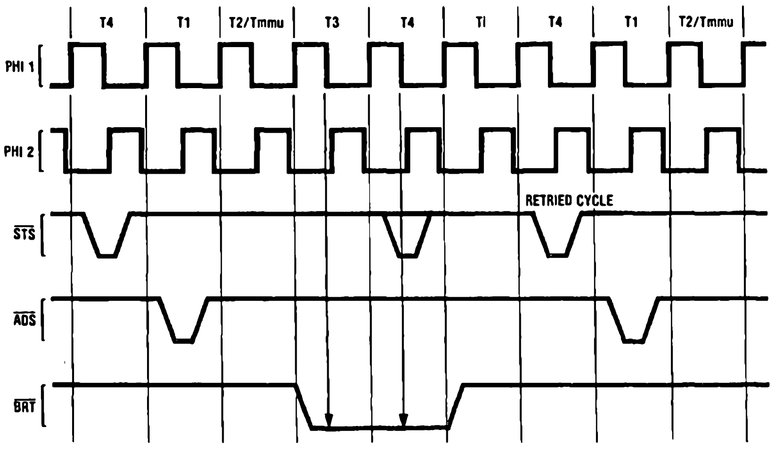
Although the decision about Retry is taken by the CPU on T4, $\overline{\text{BRT}}$ must have an early activation in T3 as described above in order to prevent the internal pipeline to advance. Holding the pipeline allows the repeated cycle to override the original one. If $\overline{\text{BRT}}$ is activated only in T3 and not in T4, there might be one cycle penalty in the performance of the execution unit in operand read cycles.

Retry is applicable for regular memory cycles and burst cycles, but not for Slave cycles.



TL/EE/8673-27

(a) Bus Cycle Not Retried



TL/EE/8673-28

(b) Bus Cycle Retried

FIGURE 3-15. Bus Cycle Retry

3.0 Functional Description (Continued)

3.4.8.2 Bus Error

If a Bus Error is incorrectable the CPU may be requested to abort the current process and branch to an appropriate routine to handle the error. The request is performed by activating the $\overline{\text{BER}}$ signal.

$\overline{\text{BER}}$ is sampled by the CPU during T4 on the falling edge of PHI1. If $\overline{\text{BER}}$ is active the bus will go to Tidle after T4 and the CPU will jump to the Bus Error handler (see Sec. 3.8).

The CPU response to Bus Error depends on the cycle type:

Instruction Fetch Cycles—If the bus error occurs on an instruction fetch, additional fetches are inhibited including the one which failed. If, after inhibiting instruction fetches, some operand cycles are still pending within the CPU, they are executed normally, delaying the access to the bus error exception. If and when the internal instruction queue becomes empty, the CPU will enter the BUS ERROR exception. This arrangement enables the CPU to ignore bus errors which belong to fetch ahead cycles if these fetches are not to be used as a result of a jump.

Operand Read Cycles—If the bus error occurs on an operand read, the bus error is immediately accepted, and the CPU enters the BUS ERROR exception.

Operand Write Cycles—If the bus error occurs on an operand write, the exception is immediately accepted.

Note 1: When a bus error occurs, the instruction that caused the error is generally not re-executable.

The process that was being executed should either be aborted or should be restarted from the last checkpoint.

Note 2: Bus error has top priority and is accepted even during the acknowledge sequence of another CPU exception (i.e. Abort, Interrupt, etc.).

It is the responsibility of the user software to detect such an occurence and to take the appropriate corrective actions.

3.4.8.3 Fatal Bus Error

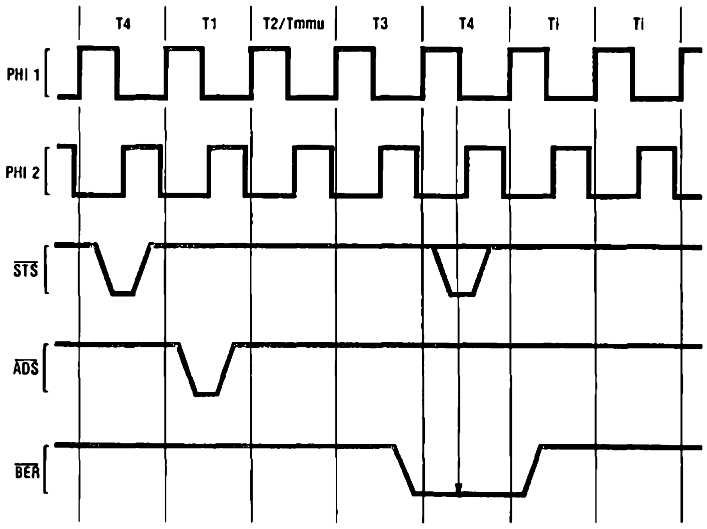
As previously mentioned, the CPU response to a bus error is to interrupt the current activity and enter the error routine.

An exception to this rule occurs when a bus error is signalled to the CPU during the acknowledge of a previous bus error. In this case the second error is interpreted by the CPU as a fatal bus error.

The CPU will respond to this event by halting execution and floating $\overline{\text{ADS}}$, $\overline{\text{BE0}}$–$\overline{\text{BE3}}$, $\overline{\text{DDIN}}$, $\overline{\text{STS}}$ and AD0–AD31.

The Halt condition is indicated by the setting of ST0–ST3 to zero and by the assertion of $\overline{\text{MC}}/\text{EXS}$ for more than one clock cycle (see Sec. 4.1.3).

The CPU can exit this condition only through a hardware reset.



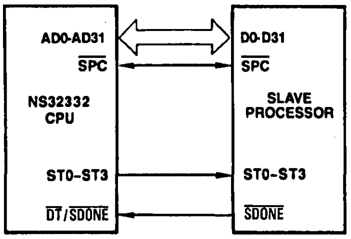
TL/EE/8673-30

FIGURE 3-16. Bus Error During Read or Write Cycle

3.0 Functional Description (Continued)

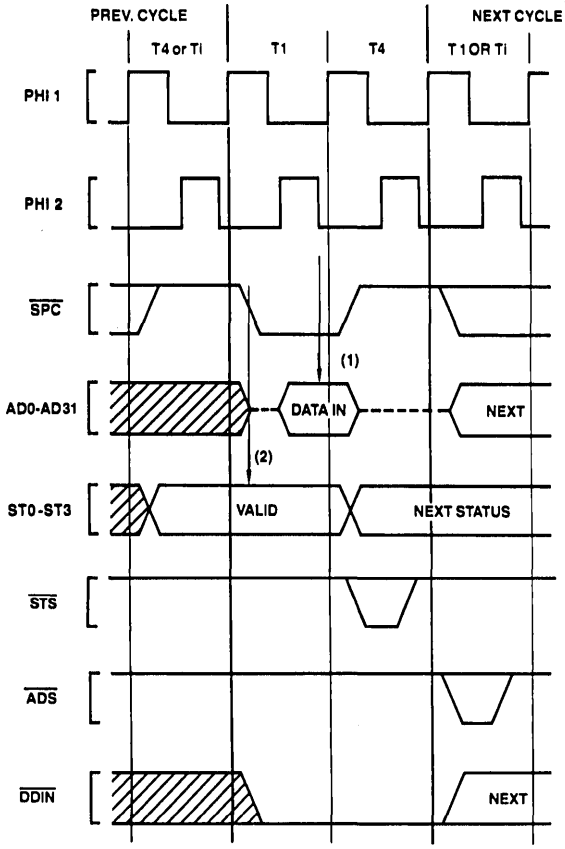
3.4.9 Slave Processor Communication

The $\overline{SPC}$ pin is used as the data strobe for Slave Processor transfers. In this role, it is referred to as Slave Processor Control ($\overline{SPC}$). In a Slave Processor bus cycle, data is transferred on the Data Bus and the status lines (ST0–ST3) are monitored by each Slave Processor in order to determine the type of transfer being performed. $\overline{SPC}$ is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Sec. 3.9 for full protocol sequences.



TL/EE/8673-31

FIGURE 3-17. Slave Processor Connections



TL/EE/8673-32

Notes:

(1) CPU samples Data Bus here.

(2) Slave Processor samples CPU Status here.

FIGURE 3-18. CPU Read from Slave Processor

3.0 Functional Description (Continued)

3.4.9.1 Slave Processor Bus Cycles

A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see *Figures 3-18* and *3-19*). During a Read cycle, $\overline{SPC}$ is active from the beginning of T1 to the beginning of T4, and the data is sampled at the end of T1. The Cycle Status pins lead the cycle by one clock period, and are sampled at the leading edge of $\overline{SPC}$. During a Write cycle, the CPU applies data and activates $\overline{SPC}$ at T1, removing $\overline{SPC}$ at T4. The Slave Processor latches status on the leading edge of $\overline{SPC}$ and latches data on the trailing edge.

The CPU does not pulse the address ($\overline{ADS}$) and status ($\overline{STS}$) strobes during a slave protocol. The direction of a transfer is determined by the sequence ("protocol") established by the instruction under execution; but the CPU indicates the direction on the $\overline{DDIN}$ pin for hardware debugging purposes.

3.4.9.2 Slave Operand Transfer Sequences

A Slave Processor operand is transferred in one or more slave operand cycles. The NS32332 supports two slave protocols which can be selected by the configuration register (CFG).

1. The regular Slave protocol is fully compatible with NS32032, NS32016 and NS32008 slave protocols.

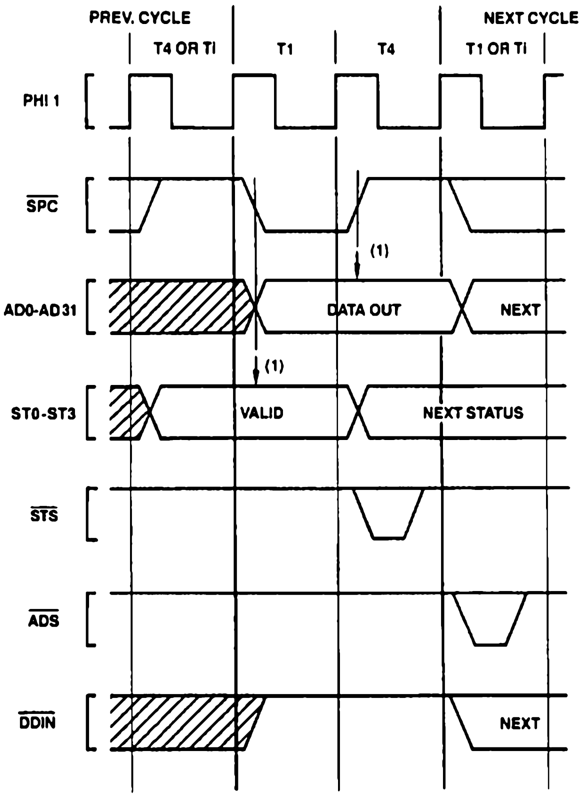
   In this protocol the NS32332 uses only the two least significant bytes of the data bus for slave cycles. This allows the NS32332 CPU to work with the current slaves (like NS32082, NS32081 etc.)

   A byte operand is transferred on the least significant byte of the data bus (AD0–AD15).

   A double word is transferred in a consecutive pair of bus cycles least significant word first. A quadword is transferred in two pairs of slave cycles.

2. The fast slave protocol is unique to the NS32332 CPU. In this protocol the NS32332 uses the full width of the data bus (AD0-AD31) for slave cycles.

   A byte operand is transferred on the least significant byte of the data bus (AD0-AD7), a word operand is transferred on bits AD0–AD15 and a double word operand is transferred on bits AD0–AD31. A quad word is transferred in two pairs of slave cycles with other bus cycles possibly occurring between them.



TL/EE/8673-33

Note:

(1) Arrows indicate points at which the Slave Processor samples.

FIGURE 3-19. CPU Write to Slave Processor

3.0 Functional Description (Continued)

3.5 MEMORY MANAGEMENT OPTION

The NS32332 CPU, in conjunction with the Memory Management Unit (MMU), provides full support for address translation, memory protection, and memory allocation techniques up to and including Virtual Memory.

When an MMU is used, the states T2 and TMMU are overlapped. During this time the CPU places AD0–AD31 into the TRI-STATE mode, allowing the MMU to assert the translated address and issue the physical address strobe $\overline{\text{PAV}}$. *Figure 3-20* shows the Bus Cycle timing with address translation.

Note 1: If an NS32382 MMU is used, the CPU can be selected to output data during write cycles in state T2, by forcing $\overline{\text{DT}}/\overline{\text{SDONE}}$ low during reset. This can be done because the NS32382 uses a separate physical address bus.

However, if a write cycle causes an MMU page table lookup, the CPU data will be valid in state T3. After $\overline{\text{FLT}}$ is deasserted, regardless of the data timing selected.

DT/$\overline{\text{SDONE}}$ must always be forced high during reset if an NS32082 MMU is used since, in this case, no separate physical address bus is provided.

Note 2: If an NS32082 MMU is used, in order for it to operate properly, it must be set to the 32-Bit mode by forcing a A24/$\overline{\text{HBF}}$ low during reset. In this mode the bus lines AD16–AD24 are floated after the MMU address has been latched, since they are used by the CPU to transfer data.

3.5.1 The $\overline{\text{FLT}}$ (Float) Pin

The $\overline{\text{FLT}}$ signal is used by the CPU for address translation support. Activating $\overline{\text{FLT}}$ during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the MMU in order to update its Translation Lookaside Buffer (TLB) from page tables in memory, or to update certain status bits within them.

Figure 3-21 shows the effect of $\overline{\text{FLT}}$. Upon sampling $\overline{\text{FLT}}$ low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

1) Sets AD0–AD31, and $\overline{\text{DDIN}}$ to the TRI-STATE condition ("floating").
2) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See $\overline{\text{RST}}/\overline{\text{ABT}}$ description.)

The above conditions remain in effect until $\overline{\text{FLT}}$ again goes high. See Sec. 4.

3.5.2 Aborting Bus Cycles

The $\overline{\text{RST}}/\overline{\text{ABT}}$ pin, apart from its Reset function (Sec. 3.3), also serves as the means to "abort", or cancel, a bus cycle and the instruction, if any, which initiated it. An Abort request is distinguished from a Reset in that the $\overline{\text{RST}}/\overline{\text{ABT}}$ pin is held active for only one clock cycle.

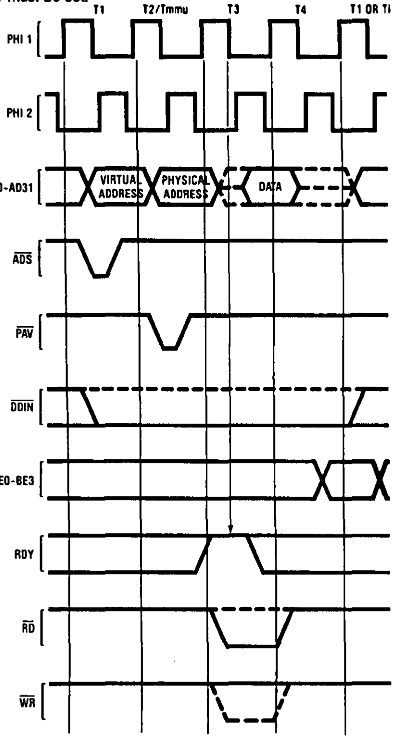
If $\overline{\text{RST}}/\overline{\text{ABT}}$ is pulled low during Tmmu or Tf, this signals that the cycle must be aborted. Since it is the MMU $\overline{\text{PAV}}$ signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was started.

The MMU will abort a bus cycle for either of two reasons:

1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The referenced page must be brought into physical memory from mass storage to make it accessible to the CPU.
2) The CPU is attempting to perform an access which is not allowed by the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction that caused it to occur is also aborted in such a manner that it is guaranteed re-executable later.

Note: To guarantee correct instruction reexecution, Bit M in the CFG Register must be set.



TL/EE/8673-87

FIGURE 3-20. Read (Write) Cycle with Address Translation

3.5.2.1 Instruction Abort

Upon aborting an instruction, the CPU immediately interrupts the instruction and performs an abort acknowledge using the ABT vector in the Interrupt Table (see Sec. 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, so that a Return from Trap (RETT) instruction will automatically retry it.

The one exception to this sequence occurs if the aborted bus cycle was an instruction prefetch. If so, it is not yet certain that the aborted prefetched code is to be executed. Instead of causing an interrupt, the CPU only aborts the bus cycle, and stops prefetching. If the information in the Instruction Queue runs out, meaning that the instruction will actually be executed, the Abort will occur, in effect aborting the instruction that was being fetched.

3.5.2.2 Hardware Considerations

In order to guarantee instruction retry, certain rules must be followed in applying an Abort to the CPU. These rules are followed by the Memory Management Unit.

1) If $\overline{\text{FLT}}$ has not been applied to the CPU, the Abort pulse must occur during Tmmu.

3.0 Functional Description (Continued)

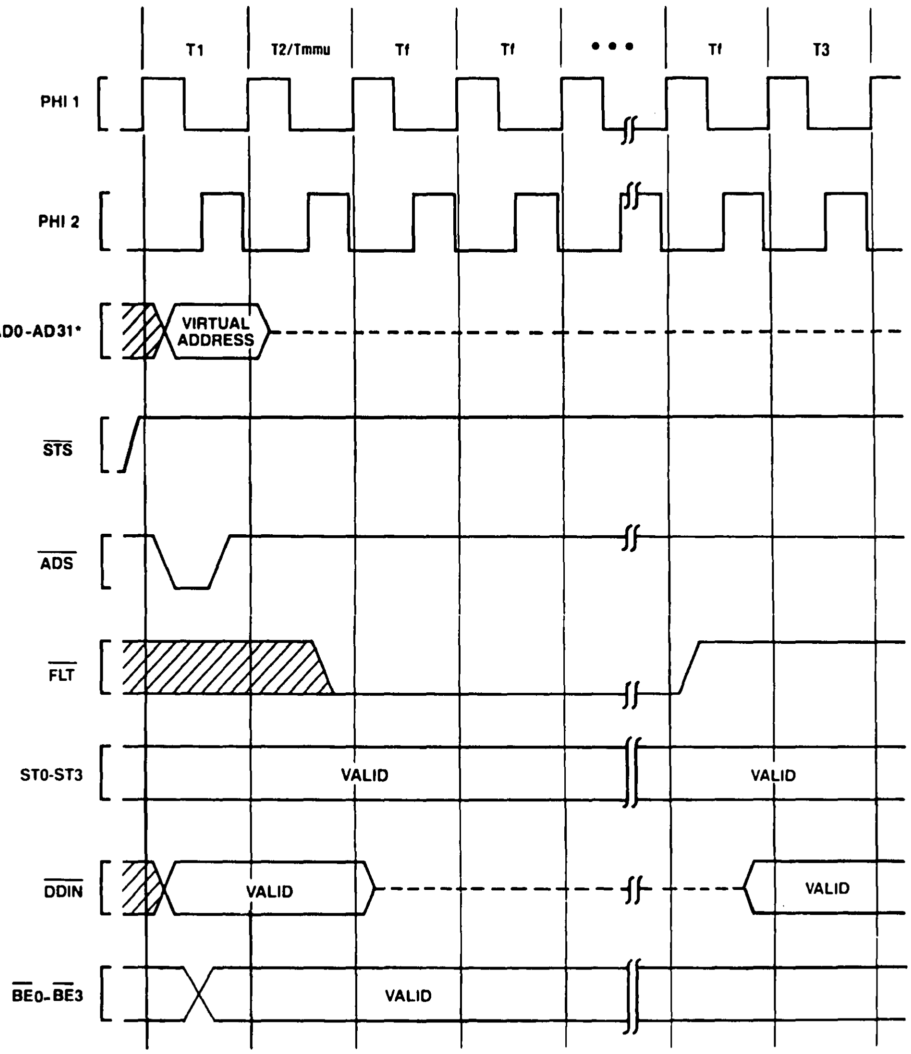
2) If $\overline{\text{FLT}}$ has been applied to the CPU, the Abort pulse must be applied before the T-State in which $\overline{\text{FLT}}$ goes inactive. The CPU will not actually respond to the Abort command until $\overline{\text{FLT}}$ is removed.

3) The Write half of a Read-Modify-Write operand access may not be aborted. The CPU guarantees that this will never be necessary for Memory Management functions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

If $\overline{\text{RST}}/\overline{\text{ABT}}$ is pulsed at any time other than as indicated above, it will abort either the instruction currently under execution or the next instruction and will act as a very high-priority interrupt. However, the program that was running at the time is not guaranteed recoverable.

3.6 BUS ACCESS CONTROL

The NS32332 CPU has the capability of relinquishing its access to the bus upon request from a DMA device or another CPU. This capability is implemented on the $\overline{\text{HOLD}}$ (Hold Request) and $\overline{\text{HLDA}}$ (Hold Acknowledge) pins. By asserting HOLD low, an external device requests access to the bus. On receipt of $\overline{\text{HLDA}}$ from the CPU, the device may perform bus cycles, as the CPU at this point has set the



*See MMU data sheet for details on physical address timing and MMU initiated Bus cycles.

TL/EE/8673-34

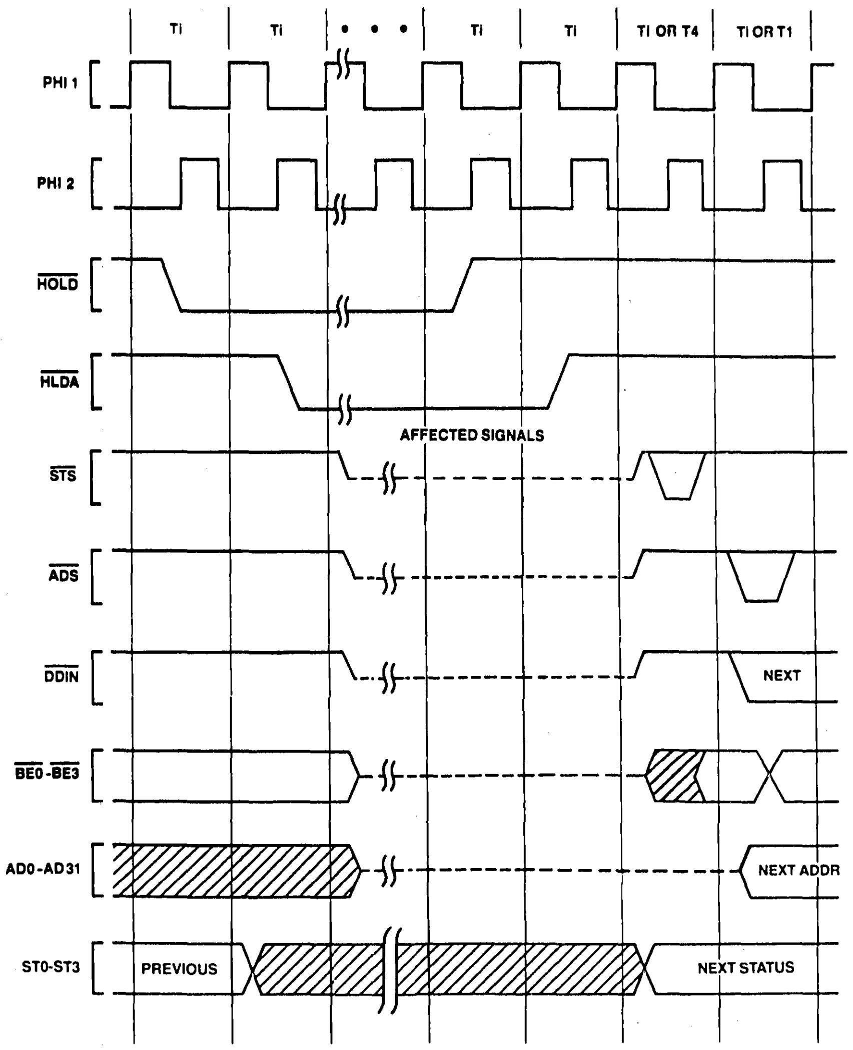
FIGURE 3-21. $\overline{\text{FLT}}$ Timing

3.0 Functional Description (Continued)

AD0–AD31, $\overline{\text{ADS}}$, $\overline{\text{STS}}$, $\overline{\text{DDIN}}$ and $\overline{\text{BE0}}$–$\overline{\text{BE3}}$ pins to the TRI-STATE condition. To return control of the bus to the CPU, the device sets $\overline{\text{HOLD}}$ inactive, and the CPU acknowledges return of the bus by setting $\overline{\text{HLDA}}$ inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the $\overline{\text{HOLD}}$ request is made, as the CPU must always complete the current bus cycle. *Figure 3-22* shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. *Figure 3-23* shows the sequence if the CPU is using the bus at the time that the $\overline{\text{HOLD}}$ request is made. If the request is made during or before the clock cycle shown (two clock cycles before T4), the CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case it will not grant the bus until after the next T4 state. Note that this situation will also occur if the CPU is idle on the bus but has initiated a bus cycle internally.

In a Memory-Managed system, the $\overline{\text{HLDA}}$ signal is connected in a daisy-chain through the MMU, so that the MMU can release the bus if it is using it.



TL/EE/8673-35

FIGURE 3-22. $\overline{\text{HOLD}}$ Timing, Bus Initially Idle

3.0 Functional Description (Continued)

3.7 INSTRUCTION STATUS

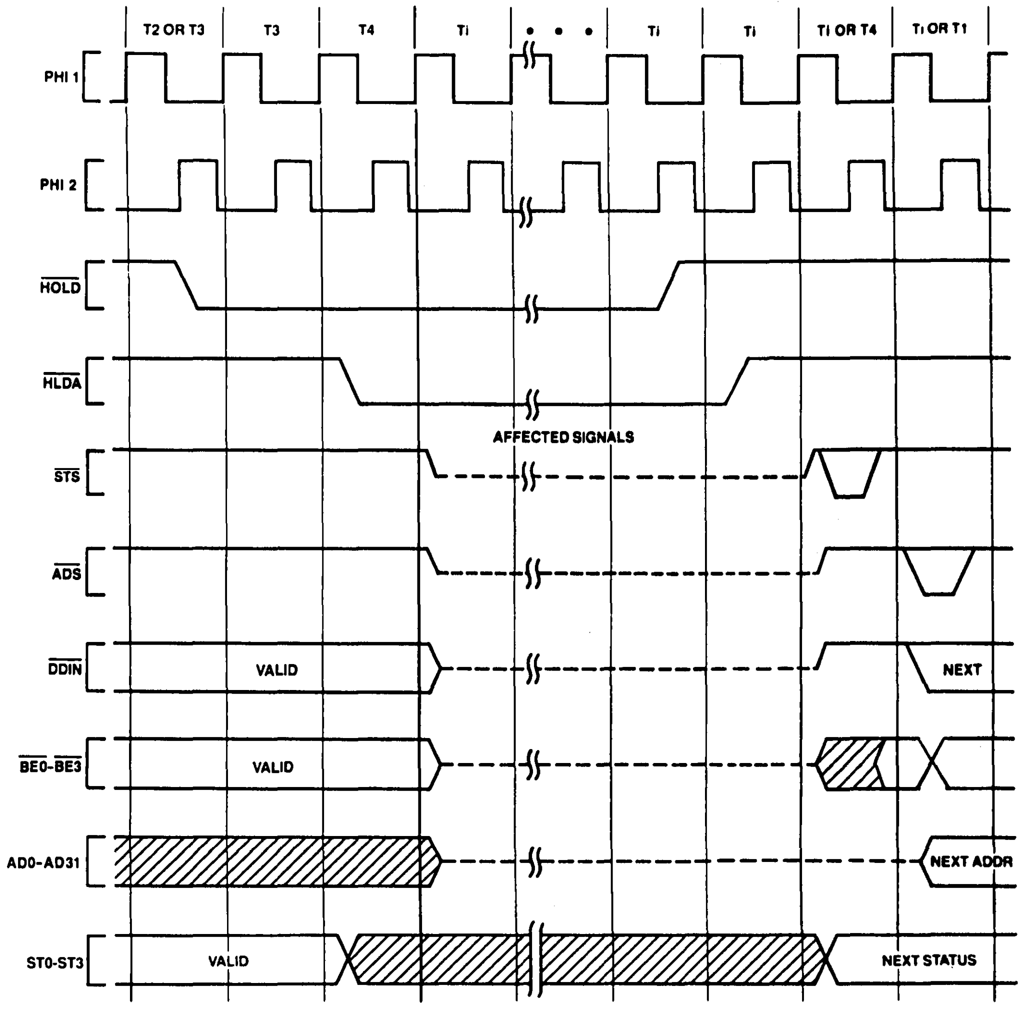
In addition to the four bits of Bus Cycle status (ST0–ST3), the NS32332 CPU also presents Instruction Status information on four separate pins. These pins differ from ST0–ST3 in that they are synchronous to the CPU's internal instruction execution section rather than to its bus interface section.

$\overline{\text{PFS}}$ (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes.

U/$\overline{\text{S}}$ originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. It is sampled by the MMU for mapping, protection, and debugging purposes. U/$\overline{\text{S}}$ line is updated every T4.

$\overline{\text{ILO}}$ (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multi-processor communication and resource sharing.

While $\overline{\text{ILO}}$ is active, the CPU inhibits instruction fetches. In order to prevent MMU cycles during $\overline{\text{ILO}}$, the CPU executes a dummy Read cycle with status code 1011 (RMW) prior to activating $\overline{\text{ILO}}$. Thereafter, $\overline{\text{ILO}}$ is activated and the Read is performed again but with status code 1010 (operand transfer). Refer to *Figure 3-24*.



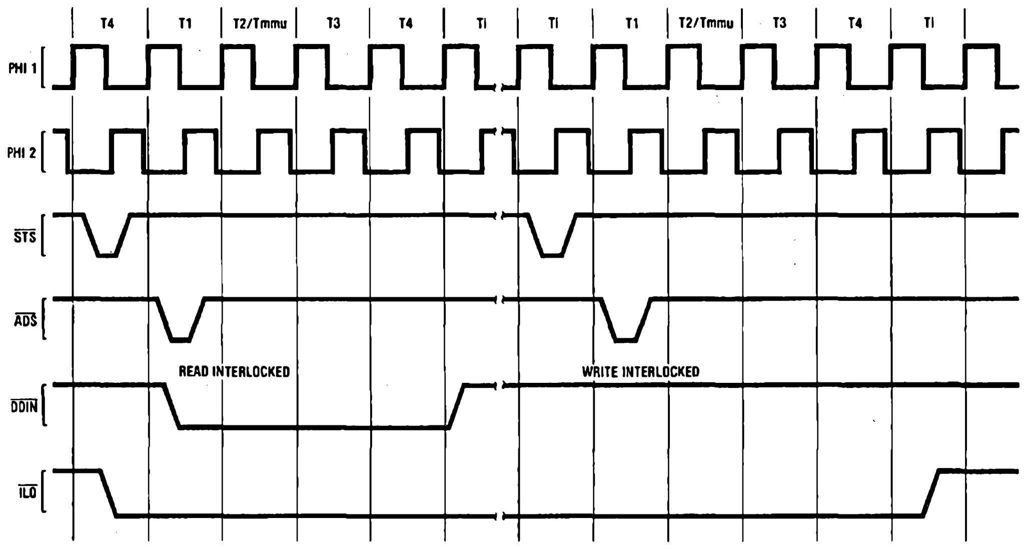
TL/EE/8673-36

FIGURE 3-23. $\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle

3.0 Functional Description (Continued)

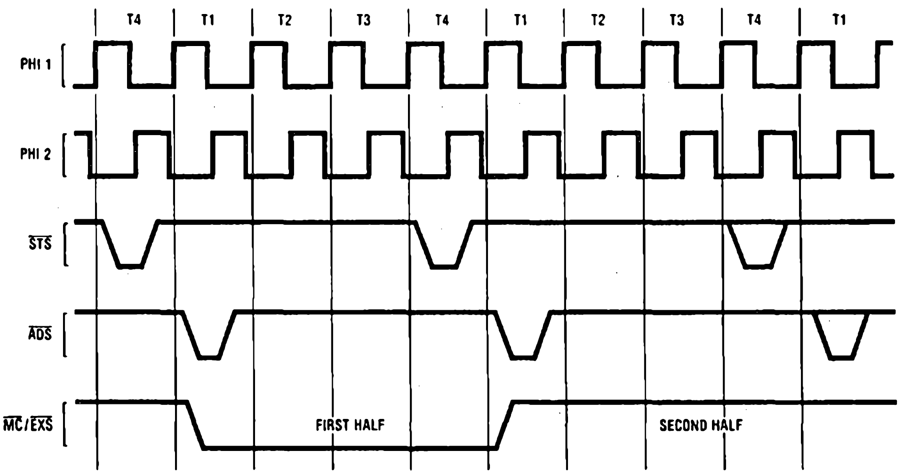
$\overline{\text{MC}}/\overline{\text{EXS}}$ (Multiple Cycle/Exception Status) is activated during the access of the first part of an operand that crosses a double-word address boundary. The activation of this signal is independent of the selected bus width. Its timing is shown in *Figure 3-25*. The MMU or other external circuitry can use it as an early indication of a CPU access to an operand that crosses a page boundary.

$\overline{\text{MC}}/\overline{\text{EXS}}$ is also activated during the first non-sequential instruction fetch (status code 1001) following an abort, and when the CPU enters the idle state (Status Code 0000) following a fatal bus error.



TL/EE/8673-37

FIGURE 3-24. $\overline{\text{ILO}}$ Timing



TL/EE/8673-38

FIGURE 3-25. Non-aligned Write Cycle—$\overline{\text{MC}}/\overline{\text{EXS}}$ Timing

3.0 Functional Description (Continued)

3.8 NS32332 INTERRUPT STRUCTURE

$\overline{\text{INT}}$, on which maskable interrupts may be requested,

$\overline{\text{NMI}}$, on which non-maskable interrupts may be requested, and

$\overline{\text{RST}}/\overline{\text{ABT}}$, which may be used to abort a bus cycle and any associated instruction. See Sec. 3.5.2.

In addition there is a set of internally-generated "traps" which cause interrupt service to be performed as a result either of exceptional conditions (e.g., attempted division by zero) or of specific instructions whose purpose is to cause a trap to occur (e.g., the Supervisor Call instruction).

3.8.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through three major steps:

1) Adjustment of Registers.

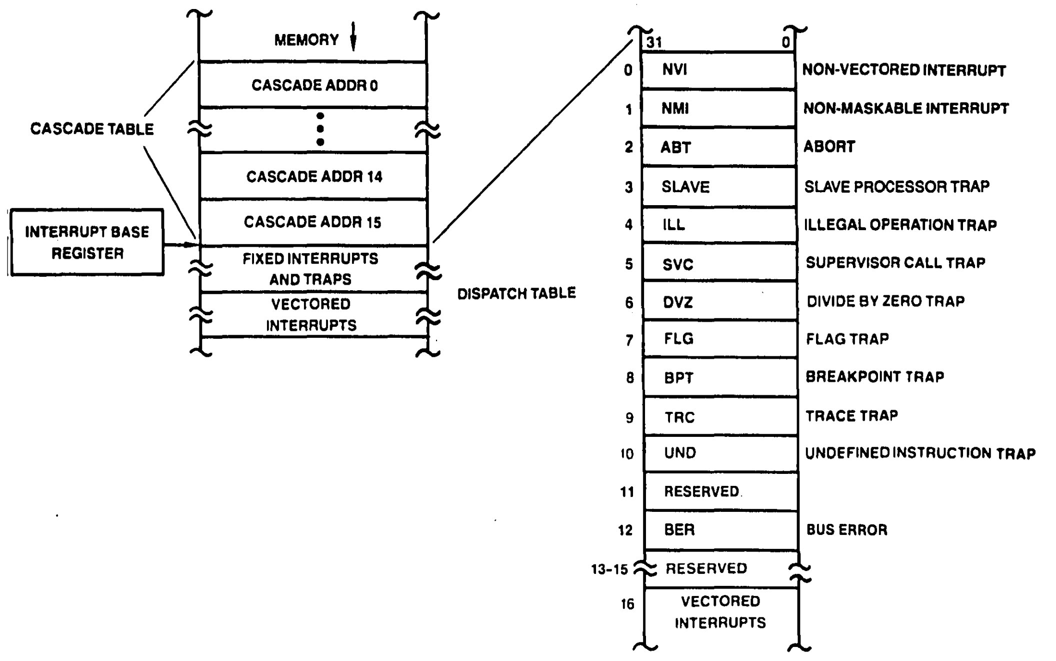
Depending on the source of the interrupt or trap, the CPU may restore and/or adjust the contents of the Program Counter (PC), the Processor Status Register (PSR) and the currently-selected Stack Pointer (SP). A copy of the PSR is made, and the PSR is then set to reflect Supervisor Mode and selection of the Interrupt Stack.

2) Vector Acquisition.

A Vector is either obtained from the Data Bus or is supplied by default.

3) Service Call.

The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See *Figure 3-26.* A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.

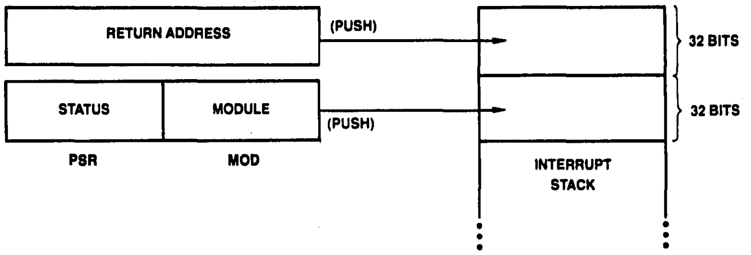


TL/EE/8673-39

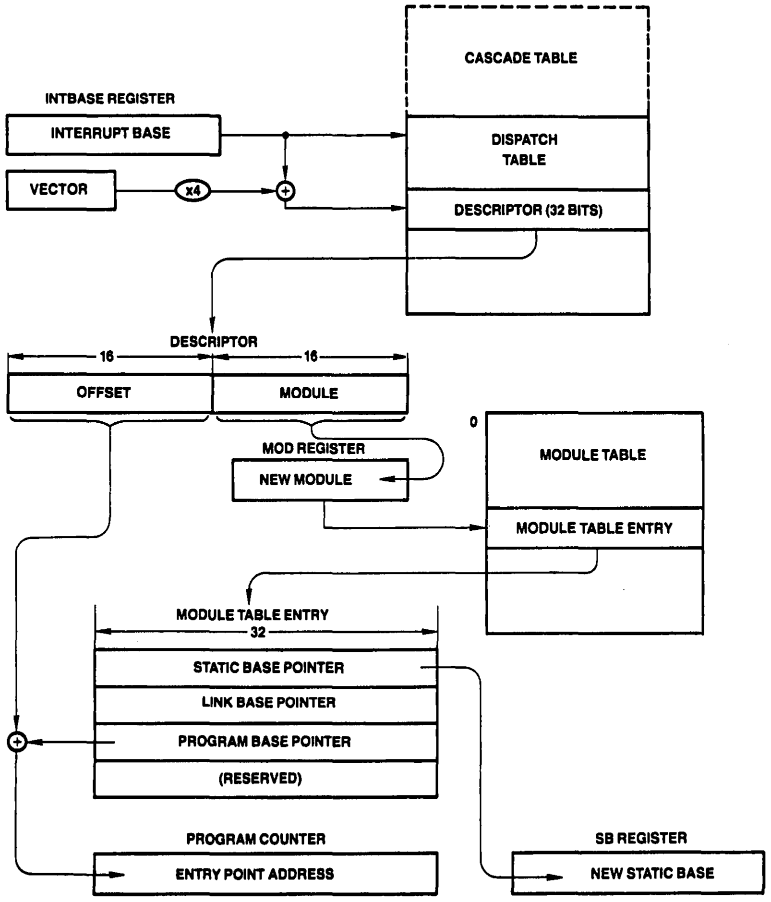
FIGURE 3-26. Interrupt Dispatch Table

3.0 Functional Description (Continued)

This process is illustrated in *Figure 3-27*, from the viewpoint of the programmer.



TL/EE/8673-40



TL/EE/8673-41

FIGURE 3-27. Interrupt/Trap Service Routine Calling Sequence

3.0 Functional Description (Continued)

3.8.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instructions is used. The RETT (Return from Trap) instruction *(Figure 3-28)* restores the PSR, MOD, PC and SB registers to their previous contents and, since traps are often used deliberately as a call mechanism for Supervisor Mode procedures, it also discards a specified number of bytes from the original stack as surplus parameter space. RETT is used to return from any trap or interrupt except the Maskable Interrupt. For this, the RETI (Return from Interrupt) instruction is used, which also informs any external Interrupt Control Units that interrupt service has completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See *Figure 3-29.*

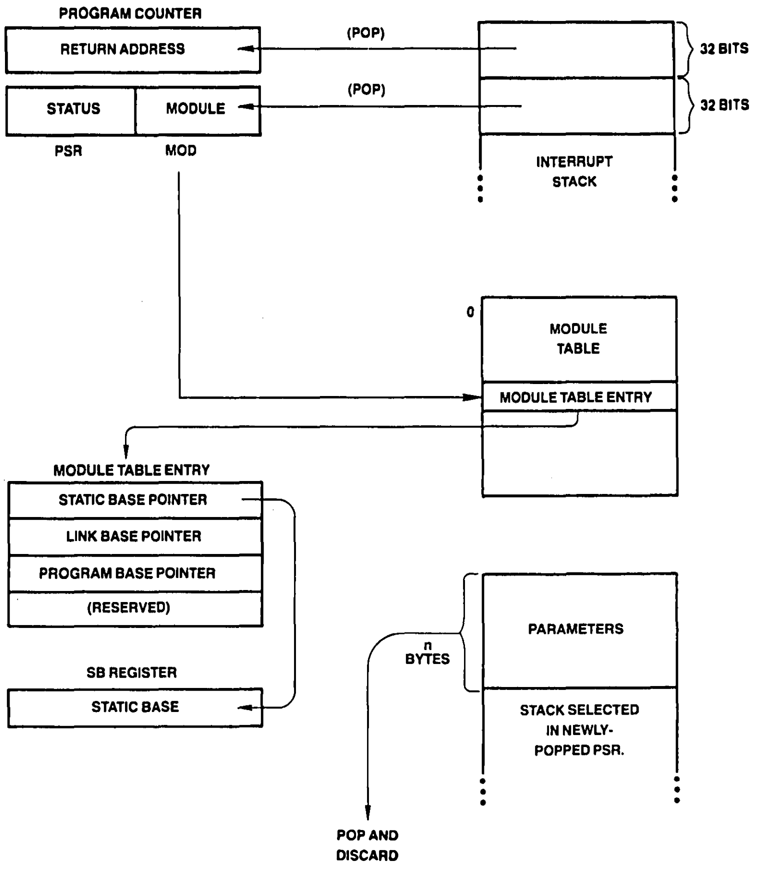
3.8.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)

The $\overline{\text{INT}}$ pin is a level-sensitive input. A continuous low level is allowed for generating multiple interrupt requests. The input is maskable, and is therefore enabled to generate interrupt requests only while the Processor Status Register I bit is set. The I bit is automatically cleared during service of an $\overline{\text{INT}}$, $\overline{\text{NMI}}$ or Abort request, and is restored to its original setting upon return from the interrupt service routine via the RETT or RETI instruction.

The $\overline{\text{INT}}$ pin may be configured via the SETCFG instruction as either Non-Vectored (CFG Register bit I = 0) or Vectored (bit I = 1).

3.8.3.1 Non-Vectored Mode

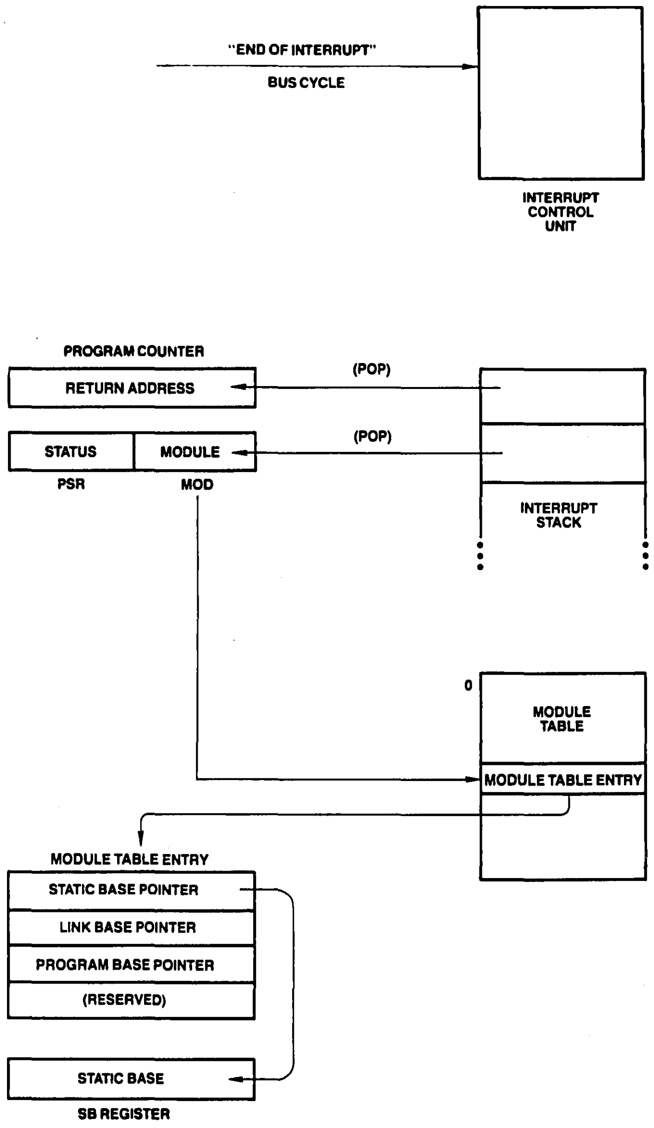
In the Non-Vectored mode, an interrupt request on the $\overline{\text{INT}}$ pin will cause an Interrupt Acknowledge bus cycle, but the CPU will ignore any value read from the bus and use instead a default vector of zero. This mode is useful for small systems in which hardware interrupt prioritization is unnecessary.



TL/EE/8673-42

FIGURE 3-28. Return from Trap (RETT n) Instruction Flow

3.0 Functional Description (Continued)



TL/EE/8673-43

FIGURE 3-29. Return from Interrupt (RETI) Instruction Flow

3.0 Functional Description (Continued)

3.8.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an Interrupt Control Unit (ICU) to prioritize many interrupt requests. Upon receipt of an interrupt request on the $\overline{\text{INT}}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.3) reading a vector value from the low-order byte of the Data Bus. This vector is then used as an index into the Dispatch Table in order to find the External Procedure Descriptor for the proper interrupt service procedure. The service procedure eventually returns via the Return from Interrupt (RETI) instruction, which performs an End of Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. The ICU provides the vector number again, which the CPU uses to determine whether it needs also to inform a Cascaded ICU (see below).

In a system with only one ICU (16 levels of interrupt), the vectors provided must be in the range of 0 through 127; that is, they must be positive numbers in eight bits. By providing a negative vector number, an ICU flags the interrupt source as being a Cascaded ICU (see below).

3.8.3.3 Vectored Mode: Cascaded Case

In order to allow more levels of interrupt, provision is made in the CPU to transparently support cascading. Note that the Interrupt output from a Cascaded ICU goes to an Interrupt Request input of the Master ICU, which is the only ICU which drives the CPU $\overline{\text{INT}}$ pin. Refer to the ICU data sheet for details.

In a system which uses cascading, two tasks must be performed upon initialization:

1) For each Cascaded ICU in the system, the Master ICU must be informed of the line number on which it receives the cascaded requests.
2) A Cascade Table must be established in memory. The Cascade Table is located in a NEGATIVE direction from the location indicated by the CPU Interrupt Base (INTBASE) Register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

Figure 3-26 illustrates the position of the Cascade Table. To find the Cascade Table entry for a Cascaded ICU, take its Master ICU line number (0 to 15) and subtract 16 from it, giving an index in the range −16 to −1. Multiply this value by 4, and add the resulting negative number to the contents of the INTBASE Register. The 32-bit entry at this address must be set to the address of the Hardware Vector Register of the Cascaded ICU. This is referred to as the "Cascade Address."

Upon receipt of an interrupt request from a Cascaded ICU, the Master ICU interrupts the CPU and provides the negative Cascade Table index instead of a (positive) vector number. The CPU, seeing the negative value, uses it as an index into the Cascade Table and reads the Cascade Address from the referenced entry. Applying this address, the CPU performs an "Interrupt Acknowledge, Cascaded" bus cycle (Sec. 3.4.3), reading the final vector value. This vector is interpreted by the CPU as an unsigned byte, and can therefore be in the range of 0 through 255.

In returning from a Cascaded interrupt, the service procedure executes the Return from Interrupt (RETI) instruction, as it would for any Maskable Interrupt. The CPU performs an "End of Interrupt, Master" bus cycle (Sec. 3.4.3), whereupon the Master ICU again provides the negative Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Sec. 3.4.3), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

Note: If an interrupt must be masked off, the CPU can do so by setting the corresponding bit in the interrupt mask register of the interrupt controller.

However, if an interrupt is set pending during the CPU instruction that masks off that interrupt, the CPU may still perform an interrupt acknowledge cycle following that instruction since it might have sampled the $\overline{\text{INT}}$ line before the ICU deasserted it. This could cause the ICU to provide an invalid vector. To avoid this problem the above operation should be performed with the CPU interrupt disabled.

3.8.4 Non-Maskable Interrupt (The $\overline{\text{NMI}}$ Pin)

The Non-Maskable Interrupt is triggered whenever a falling edge is detected on the $\overline{\text{NMI}}$ pin. The CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.3) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is FFFFFF00_{16}. The vector value used for the Non-Maskable Interrupt is taken as 1, regardless of the value read from the bus.

The service procedure returns from the Non-Maskable Interrupt using the Return from Trap (RETT) instruction. No special bus cycles occur on return.

For the full sequence of events in processing the Non-Maskable Interrupt, see Sec. 3.8.7.1.

3.8.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by the NS32332 CPU are:

Trap (SLAVE): An exceptional condition was detected by the Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Sec. 3.9.1).

Trap (ILL): Illegal operation. A privileged operation was attempted while the CPU was in User Mode (PSR bit U = 1).

Trap (SVC): The Supervisor Call (SVC) instruction was executed.

Trap (DVZ): An attempt was made to divide an integer by zero. (The Slave trap is used for Floating Point division by zero.)

Trap (FLG): The FLAG instruction detected a "1" in the CPU PSR F bit.

Trap (BPT): The Breakpoint (BPT) instruction was executed.

Trap (TRC): The instruction just completed is being traced. See below.

Trap (UND): An undefined opcode was encountered by the CPU.

3.0 Functional Description (Continued)

A special case is the Trace Trap (TRC), which is enabled by setting the T bit in the Processor Status Register (PSR). At the beginning of each instruction, the T bit is copied into the PSR P (Trace "Pending") bit. If the P bit is set at the end of an instruction, then the Trace Trap is activated. If any other trap or interrupt request is made during a traced instruction, its entire service procedure is allowed to complete before the Trace Trap occurs. Each interrupt and trap sequence handles the P bit for proper tracing, guaranteeing one and only one Trace Trap per instruction, and guaranteeing that the Return Address pushed during a Trace Trap is always the address of the next instruction to be traced.

Note: A slight difference exists between the NS32332 and previous Series 32000 CPUs when tracing is enabled.

The NS32332 always clears the P bit in the PSR before pushing the PSR on the stack. Previous CPUs do not clear it when a trap (ILL) occurs.

The result is that an instruction that causes a trap (ILL) exception is traced by previous Series 32000 CPUs, but is never traced by the NS32332.

3.8.6 Prioritization

The NS32332 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

1) Traps other than Trace	(Highest priority)
2) Abort	
3) Bus Error	
4) Non-Maskable Interrupt	
5) Maskable Interrupts	
6) Trace Trap	(Lowest priority)

3.8.7 Interrupt/Trap Sequences: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in *Figure 3-30*. Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequence followed in processing either Maskable or Non-Maskable interrupts (on the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pins, respectively), see Sec. 3.8.7.1 For Abort Interrupts, see Sec. 3.8.7.4. For the Trace Trap, see Sec. 3.8.7.3, and for all other traps see Sec. 3.8.7.2.

3.8.7.1 Maskable/Non-Maskable Interrupt Sequence

This sequence is performed by the CPU when the $\overline{\text{NMI}}$ pin receives a falling edge, or the $\overline{\text{INT}}$ pin becomes active with the PSR I bit set. The interrupt sequence begins either at the next instruction boundary or, in the case of the String instructions, at the next interruptible point during its execution.

1. If a String instruction was interrupted and not yet completed:
   a. Clear the Processor Status Register P bit.
   b. Set "Return Address" to the address of the first byte of the interrupted instruction.

   Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
   a. Read a byte from address FFFFFF00_{16}, applying Status Code 0100 (Interrupt Acknowledge, Master, Sec. 3.4.3). Discard the byte read.
   b. Set "Vector" to 1.
   c. Go to Step 8.
4. If the interrupt is Non-Vectored:
   a. Read a byte from address FFFFFE00_{16}, applying Status Code 0100 (Interrupt Acknowledge, Master: Sec. 3.4.3). Discard the byte read.
   b. Set "Vector" to 0.
   c. Go to Step 8.
5. Here the interrupt is Vectored. Read "Byte" from address FFFFFE00_{16}, applying Status Code 0100 (Interrupt Acknowledge, Master: Sec. 3.4.3).
6. If "Byte" ≥ 0, then set "Vector" to "Byte" and go to Step 8.
7. If "Byte" is in the range -16 through -1, then the interrupt source is Cascaded. (More negative values are reserved for future use.) Perform the following:
   a. Read the 32-bit Cascade Address from memory. The address is calculated as INTBASE $+4*$ Byte.
   b. Read "Vector," applying the Cascade Address just read and Status Code 0101 (Interrupt Acknowledge, Cascaded: Sec. 3.4.3).
8. Perform Service (Vector, Return Address), *Figure 3-30*.

Service (Vector, Return Address):

1) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table: address is Vector* 4 + INTBASE Register contents.

2) Move the Module field of the Descriptor into the MOD Register.

3) Read the Program Base pointer from memory address MOD + 8, and add to it the Offset field from the Descriptor, placing the result in the Program Counter.

4) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.

5) Flush queue: Non-sequentially fetch first instruction of Interrupt routine.

6) Push the PSR copy onto the Interrupt Stack as a 16-bit value.

7) Push MOD Register into the Interrupt Stack as a 16-bit value.

8) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

FIGURE 3-30. Service Sequence

Invoked during all interrupt/trap sequences.

3.8.7.2 Trap Sequence: Traps Other Than Trace

1) Restore the currently selected Stack Pointer and the Processor Status Register to their original values at the start of the trapped instruction.

2) Set "Vector" to the value corresponding to the trap type.

SLAVE:	Vector = 3.
ILL:	Vector = 4.
SVC:	Vector = 5.
DVZ:	Vector = 6.
FLG:	Vector = 7.
BPT:	Vector = 8.
UND:	Vector = 10.

3) Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, P and T.

3.0 Functional Description (Continued)

4) Set "Return Address" to the address of the first byte of the trapped instruction.

5) Perform Service (Vector, Return Address), *Figure 3-30.*

3.8.7.3 Trace Trap Sequence

1) In the Processor Status Register (PSR), clear the P bit.

2) Copy the PSR into a temporary register, then clear PSR bits S, U and T.

3) Set "Vector" to 9.

4) Set "Return Address" to the address of the next instruction.

5) Perform Service (Vector, Return Address), *Figure 3-30.*

3.8.7.4 Abort Sequence

1) Restore the currently selected Stack Pointer to its original contents at the beginning of the aborted instruction.

2) Clear the PSR P bit.

3) Copy the PSR into a temporary register, then clear PSR bits S, U, T and I.

4) Set "Vector" to 2.

5) Set "Return Address" to the address of the first byte of the aborted instruction.

6) Perform Service (Vector, Return Address), *Figure 3-30.*

3.8.7.5 Bus Error Sequence

1) The same as Abort sequence above, but set vector to 12.

3.9 SLAVE PROCESSOR INSTRUCTIONS

The NS32332 CPU recognizes three groups of instructions being executable by external Slave Processor:

Floating Point Instruction Set
Memory Management Instruction Set
Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Sec. 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

In addition, each slave instruction will be performed either through the regular (32032 compatible) slave protocol or through a fast slave protocol according to the relevent bit in the configuration register (Sec. 2.1.3).

A combination of one slave communicating with an old protocol and another with a new protocol is allowed, e.g. 16-bit FPU (32081) and 32-bit MMU (32382) or vice versa.

3.9.1 16-Bit Slave Processor Protocol (32032 Compatible)

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. The ID Byte has three functions:

1) It identifies the instruction as being a Slave Processor instruction.

2) It specifies which Slave Processor will execute it.

3) It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-31.* While applying Status Code 1111 (Broadcast ID, Sec. 3.4.3), the CPU transfers the ID Byte on bits AD0–AD7 and a non-used byte xxxxxxx1 (x = don't care) on bits AD24–AD31. All Slave Processors input this byte and decode it. The Slave Processor selected by the ID Byte is activated, and from this point the CPU is communicating only with it. If any other slave protocol was in progress (e.g., an aborted Slave instruction), this transfer cancels it.

The CPU next sends the Operation Word while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.3). Upon receiving it, the Slave Processor decodes it, and at this point both the CPU and the Slave Processor are aware of the number of operands to be transferred and their sizes. The operation Word is swapped on the Data Bus, that is, bits 0–7 appear on pins AD8–AD15 and bits 8–15 appear on pins AD0–AD7.

Using the Address Mode fields within the Operation Word, the CPU starts fetching operand and issuing them to the Slave Processor. To do so, it references any Addressing Mode extensions which may be appended to the Slave Processor instruction. Since the CPU is solely responsible for memory accesses, these extensions are not sent to the Slave processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Sec. 3.4.3).

After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{\text{SPC}}$ low. To allow for this $\overline{\text{SPC}}$ is normally held high only by an internal pull-up device of approximately 5 kΩ.

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills the queue before the Slave Processor finishes, the CPU will wait, applying Status Code 0011 (Waiting for Slave, Sec. 3.4.3).

Upon receiving the pulse on $\overline{\text{SPC}}$, the CPU uses $\overline{\text{SPC}}$ to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Sec. 3.4.3). This word has the format shown in *Figure 3-34.* If the Q bit ("Quit", Bit 0) is set, this indicates that an error was detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the SLAVE vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.3).

An exception to the protocol above is the LMR (Load Memory Management Register) instruction, and a corresponding Custom Slave instruction (LCR: Load Custom Register). In executing these instructions, the protocol ends after the CPU has issued the last operand. The CPU does not wait for an acknowledgement from the Slave Processor, and it does not read status.

3.0 Functional Description (Continued)

Status Combinations:

Send ID (ID): Code 1111

Xfer Operand (OP): Code 1101

Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU Send ID Byte.
2	OP	CPU Sends Operaton Word.
3	OP	CPU Sends Required Operands
4	—	Slave Starts Execution. CPU Pre-fetches.
5	—	Slave Pulses $\overline{SPC}$ Low.
6	ST	CPU Reads Status Word. (Trap? Alter Flags?)
7	OP	CPU Reads Results (If Any).

FIGURE 3-31. 16-Bit Slave Processor Protocol

3.9.2 32-Bit Fast Slave Protocol

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in *Figure 3-32*. While applying Status code 1111 (Broadcast ID Sec. 3.4.2), the CPU transfers the ID Byte on bits AD24–AD31, the operation word on bits AD8–AD23 in a swapped order of bytes and a non-used byte XXXXXXX1 (X = don't care) on bits AD0–AD7 (*Figure 3-33*).

Using the addressing mode fields within the Operation word, the CPU fetches operands and sends them to the Slave Processor. Since the CPU is solely responsible for memory accesses, addressing mode extensions are not sent to the Slave Processor. The Status Code applied is 1101 (Transfer Slave Processor Operand Sec. 3.4.2). After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{SDONE}$ or $\overline{SPC}$ low for one clock cycle.

Unlike the old protocol, the SLAVE may request the CPU to read the status by activating the $\overline{SDONE}$ or $\overline{SPC}$ line for two clock cycles instead of one. The CPU will then read the slave status word and update the PSR Register, unless a trap is signalled. If this happens, the CPU will immediately abort the protocol and start a trap sequence using either the SLAVE or the UND vector in the interrupt table as specified in the Status Word.

Note: The PSR update is presently restricted to three instructions: CMPf, RDVAL, WRVAL and their custom slave equivalents.

While the Slave Processor is executing the instruction, the CPU is free to prefetch instructions into its queue. If it fills its queue before the Slave Processor finishes, the CPU will wait applying status code 0011 (waiting for Slave, Sec. 3.4.2).

Upon receiving the pulse on either $\overline{SDONE}$ or $\overline{SPC}$, the CPU uses $\overline{SPC}$ to read the result from the Slave Processor and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2).

Status Combinations:

Send ID (ID): Code 1111

Xfer Operand (OP): Code 1101

Read Status (ST): Code 1110

Step	Status	Action
1	ID	CPU sends ID and Operation Word.
2	OP	CPU sends required operands (if any).
3	—	Slave starts execution (CPU prefetches).*
4	—	Slave pulses $\overline{SDONE}$ or $\overline{SPC}$ low.
5	ST	CPU Reads Status word (only if $\overline{SDONE}$ or $\overline{SPC}$ pulse is two clock cycles wide).
6	OP	CPU Reads Results (if any).

FIGURE 3-32. 32-Bit Fast Slave Protocol

Certain Slave Processor instructions affect CPU PSR. For these instructions only the CPU will perform a Read Slave status cycle as described in 3.9.1.1 before reading the result. The relevent PSR bits will be loaded from the status word.

byte 3	byte 2	byte 1	byte 0
ID	OPCODE low	OPCODE high	Don't Care

FIGURE 3-33. ID and Opcode Format for Fast Slave Protocol

3.9.3 Floating Point Instructions

Table 3-4 gives the protocols followed for each Floating Point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Appendix A.

The Operand class columns give the Access Class for each general operand, defining how the addressing modes are interpreted (see Instruction Set Reference Manual).

The Operand Issued columns show the sizes of the operands issued to the Floating Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "f" indicates that the instruction specifies a Floating Point size for the operand (F = 32-bit Standard Floating, L = 64-bit Long Floating).

The Returned Value Type and Destination column gives the size of any returned value and where the CPU places it. The PSR Bits Affected column indicates which PSR bits, if any, are updated from the Slave Processor Status Word (*Figure 3-34*).

3.0 Functional Description (Continued)

TABLE 3-4

Floating Point Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
ADDf	read.f	rmw.f	f	f	f to Op. 2	none
SUBf	read.f	rmw.f	f	f	f to Op. 2	none
MULf	read.f	rmw.f	f	f	f to Op. 2	none
DIVf	read.f	rmw.f	f	f	f to Op. 2	none
MOVf	read.f	write.f	f	N/A	f to Op. 2	none
ABSf	read.f	write.f	f	N/A	f to Op. 2	none
NEGf	read.f	write.f	f	N/A	f to Op. 2	none
CMPf	read.f	read.f	f	f	N/A	N,Z,L
FLOORfi	read.f	write.i	f	N/A	i to Op. 2	none
TRUNCfi	read.f	write.i	f	N/A	i to Op. 2	none
ROUNDfi	read.f	write.i	f	N/A	i to Op. 2	none
MOVFL	read.F	write.L	F	N/A	L to Op. 2	none
MOVLF	read.L	write.F	L	N/A	F to Op. 2	none
MOVif	read.i	write.f	i	N/A	f to Op. 2	none
POLYf	read.f	read.f	f	f	f to F0	none
DOTf	read.f	read.f	f	f	f to F0	none
SCALBf	read.f	rmw.f	f	f	f to Op.2	none
LOGBf	read.f	write.f	f	N/A	f to Op.2	none
LFSR	read.D	N/A	D	N/A	N/A	none
SFSR	N/A	write.D	N/A	N/A	D to Op. 2	none

Note 1:

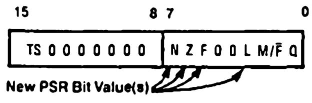
D = Double Word

i = Integer size (B,W,D) specified in mnemonic.

f = Floating Point type (F,L) specified in mnemonic.

N/A = Not Applicable to this instruction.

Any operand indicated as being of type "f" will not cause a transfer if the Register addressing mode is specified. This is because the Floating Point Registers are physically on the Floating Point Unit and are therefore available without CPU assistance.



TL/EE/8673-44

FIGURE 3-34. Slave Processor Status Word Format

Note 1: Q is the Trap Bit. It is set to 1 by the Slave whenever a trap is requested.

Note 2: TS is the Trap Select Bit. When a trap is requested (Q = 1), TS tells the CPU whether a SLAVE or an UND trap is to be generated. TS is 0 for a slave trap and 1 for an UND trap.

Note 3: M/F̄ should be set for a RDVAL, WRVAL, or Custom Slave Equivalent instruction. It should be cleared for CMPf and CCMP0c and CCMPc. When M/F̄ is cleared, the F bit should also be cleared.

3.9.4 Memory Management Instructions

Table 3-5 gives the protocols for Memory Management instructions. Encodings for these instructions may be found in Appendix A.

In executing the RDVAL and WRVAL instructions, the CPU calculates and issues the 32-bit Effective Address of the single operand. The CPU then performs a single-byte Read cycle from that address, allowing the MMU to safely abort the instruction if the necessary information is not currently in physical memory. Upon seeing the memory cycle complete, the MMU continues the protocol, and returns the validation result in the F bit of the Slave Status Word.

The size of a Memory Management operand is always a 32-bit Double Word. For further details of the Memory Management Instruction set, see the Instruction Set Reference Manual and the MMU Data Sheet.

3.0 Functional Description (Continued)

TABLE 3-5
Memory Management Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
RDVAL*	addr	N/A	D	N/A	N/A	F
WRVAL*	addr	N/A	D	N/A	N/A	F
LMR*	read.D	N/A	D	N/A	N/A	none
SMR*	write.D	N/A	N/A	N/A	D to Op. 1	none

Note:
In the RDVAL and WRVAL instructions, the CPU issues the address as a Double Word, and performs a single-byte Read cycle from that memory address. For details, see the Instruction Set Reference Manual and the Memory Management Unit Data Sheet.

D = Double Word

* = Privileged Instruction: will trap if CPU is in User Mode.

N/A = Not Applicable to this instruction.

3.9.5 Custom Slave Instructions

Provided in the NS32332 is the capability of communicating with a user-defined, "Custom" Slave Processor. The instruction set provided for a Custom Slave Processor defines the instruction formats, the operand classes and the communication protocol. Left to the user are the interpretations of the Op Code fields, the programming model of the Custom Slave and the actual types of data transferred. The protocol specifies only the size of an operand, not its data type.

Table 3-6 lists the relevant information for the Custom Slave instruction set. The designation "c" is used to represent an operand which can be a 32-bit ("D") or 64-bit ("Q") quantity in any format; the size is determined by the suffix on the mnemonic. Similarly, an "i" indicates an integer size (Byte, Word, Double Word) selected by the corresponding mnemonic suffix.

Any operand indicated as being of type "c" will not cause a transfer if the register addressing mode is specified. It is assumed in this case that the slave processor is already holding the operand internally.

For the instruction encodings, see Appendix A.

3.0 Functional Description (Continued)

TABLE 3-6
Custom Slave Instruction Protocols.

Mnemonic	Operand 1 Class	Operand 2 Class	Operand 1 Issued	Operand 2 Issued	Returned Value Type and Dest.	PSR Bits Affected
CCAL0c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL1c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL2c	read.c	rmw.c	c	c	c to Op. 2	none
CCAL3c	read.c	rmw.c	c	c	c to Op. 2	none
CMOV0c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV1c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV2c	read.c	write.c	c	N/A	c to Op. 2	none
CMOV3c	read.c	write.c	c	N/A	c to Op.2	none
CCMP0c	read.c	read.c	c	c	N/A	N,Z,L
CCMP1c	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to OP. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op.1	none

Note:
D = Double Word
i = Integer size (B,W,D) specified in mnemonic.
c = Custom size (D:32 bits or Q:64 bits) specified in mnemonic.
* = Privileged instruction: will trap if CPU is in User Mode.
N/A = Not Applicable to this instruction.

4.0 Device Specifications

4.1 NS32332 PIN DESCRIPTIONS

The following is a brief description of all NS32332 pins. The descriptions reference portions of the Functional Description, Section 3.

Unless otherwise indicated, reserved pins should be left open.

4.1.1 Supplies

Logic Power ($V_{CCL1, 2}$): +5V positive supply.

Buffers Power ($V_{CCB1, 2, 3, 4, 5}$): +5V positive supply.

Logic Ground (GNDL1, GNDL2): Ground reference for on-chip logic.

Buffer Grounds (GNDB1, GNDB2, GNDB3, GNDB4, GNDB5, GNDB6): Ground references for on-chip drivers.

Back Bias Generator (BBG): Output of on-chip substrate voltage generator.

4.1.2 Input Signals

Clocks (PHI1, PHI2): Two-phase clocking signals.

Ready (RDY): Active high. While RDY is not active, the CPU adds wait cycles to the current bus cycle. Not applicable for slave cycles.

Hold Request ($\overline{\text{HOLD}}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes.

Note: If the $\overline{\text{HOLD}}$ signal is generated asynchronously, it's set up and hold times may be violated. In this case it is recommended to synchronize it with CTTL to minimize the possibility of metastable states.

The CPU provides only one synchronization stage to minimize the $\overline{\text{HLDA}}$ latency. This is to avoid speed degradations in cases of heavy $\overline{\text{HOLD}}$ activity (i.e. DMA controller cycles interleaved with CPU cycles.)

Interrupt ($\overline{\text{INT}}$): Active low. Maskable Interrupt request.

Non-Maskable Interrupt ($\overline{\text{NMI}}$): Active low. Non-Maskable Interrupt request.

Reset/Abort ($\overline{\text{RST}}/\overline{\text{ABT}}$): Active low. If held active for one clock cycle and released, this pin causes an ABORT. If held longer, it is interpreted as RESET.

Bus Error ($\overline{\text{BER}}$): Active low. When active, indicates that an error occurred during a bus cycle. It is treated by the CPU as the highest priority exception after RESET. Not applicable for slave cycles.

Bus Retry ($\overline{\text{BRT}}$): Active low. When active, the CPU will re-execute the last bus cycle. Not applicable for slave cycles.

Bus Width (BW1, BW0): Define the bus width (8, 16, 32) in every bus cycle. 01–8 bits, 10–16 bits, 11–32 bits. 00 is a reserved combination. Not applicable for slave cycles.

Burst in ($\overline{\text{BIN}}$): Active low. When active, the CPU may perform burst cycles.

Float ($\overline{\text{FLT}}$): Active low. Float command input. In non-memory managed systems, this pin should be tied to V_{CC} through a 10 kΩ resistor.

Data Timing/Slave Done ($\overline{\text{DT}}/\overline{\text{SDONE}}$): Active low. Used by a 32-bit slave processor to acknowledge the completion of an instruction and/or indicate that the slave status should be read (Section 3.9.2). Sampled during reset to select the data timing during write cycles (Section 3.3).

4.1.3 Output Signals

Address Strobe ($\overline{\text{ADS}}$): Active low. Controls address latches, indicates the start of a bus cycle.

Data Direction In ($\overline{\text{DDIN}}$): Active low. Indicates the directions of data transfers.

Byte Enables ($\overline{\text{BE0}}-\overline{\text{BE3}}$): Active low. Enable the access of bytes 0–3 in a 32 bit system.

Status (ST0–ST3): Bus cycle status code, ST0 least significant. Encodings are:

0000	— Idle: CPU Inactive on Bus.
0001	— Idle: WAIT Instruction.
0010	— (Reserved).
0011	— Idle: Waiting for Slave.
0100	— Interrupt Acknowledge, Master.
0101	— Interrupt Acknowledge, Cascaded.
0110	— End of Interrupt, Master.
0111	— End of Interrupt, Cascaded.
1000	— Sequential Instruction Fetch.
1001	— Non-Sequential Instruction Fetch.
1010	— Data Transfer.
1011	— Read Read-Modify-Write Operand.
1100	— Read for Effective Address.
1101	— Transfer Slave Operand.
1110	— Read Slave Status Word.
1111	— Broadcast Slave ID.

Status Strobe ($\overline{\text{STS}}$): Active low. Indicates that a new status (ST0–ST3) is valid. Not applicable for slave cycles.

Multiple Cycle/Exception Status ($\overline{\text{MC}}/\overline{\text{EXS}}$): Active low. This signal is activated during the access of the first part of an operand that crosses a double word address boundary.

It is also activated in conjunction with status codes 1001 and 0000 during Abort Acknowledge and when a fatal bus error occurs.

Note: $\overline{\text{MC}}/\overline{\text{EXS}}$ indicates a fatal bus error only when it has been active for more than one clock cycle.

Hold Acknowledge ($\overline{\text{HLDA}}$): Active low. Activated by the CPU in response to HOLD input. Indicates that the CPU has released the bus.

User/Supervisor (U/$\overline{\text{S}}$): User or Supervisor Mode status.

Interlocked Operation ($\overline{\text{ILO}}$): Active low. Indicates that an interlocked cycle is being performed.

Program Flow Status ($\overline{\text{PFS}}$): Active low. A pulse that indicates the beginning of an instruction execution.

Burst Out ($\overline{\text{BOUT}}$): Active low. When active, indicates that the CPU will perform burst cycles.

4.1.4 Input/Output Signals

Address/Data 0–31 (AD0–AD31): Multiplexed address and data lines.

Slave Processor Control ($\overline{\text{SPC}}$): Active low. Used by the CPU as a data strobe output for slave processor transfers. Used by a 16-bit slave processor to acknowledge the completion of an instruction.

4.0 Device Specifications (Continued)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

4.2 ABSOLUTE MAXIMUM RATINGS

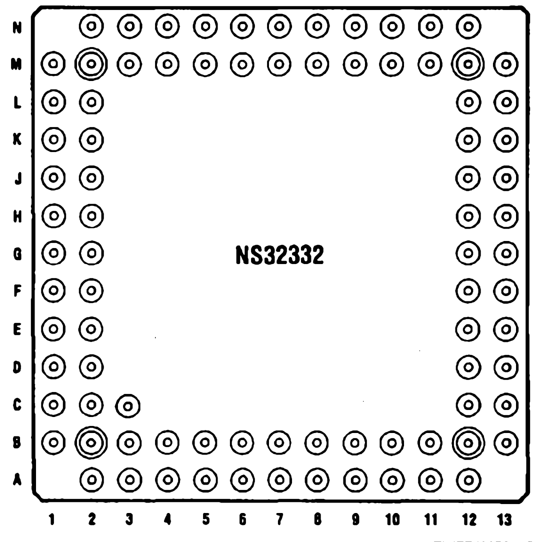
Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to GND	−0.5V to +7V
Power Dissipation	3 Watt

Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS T_A = 0° to +70°C, V_{CC} = 5V ±5%, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	High Level Input Voltage		2.0		V_{CC} +0.5	V
V_{IL}	Low Level Input Voltage		−0.5		0.8	V
V_{CH}	High Level Clock Voltage	PHI1, PHI2 pins only	V_{CC} −0.5		V_{CC} +0.5	V
V_{CL}	Low Level Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CRT}	Clock Input Ringing Tolerance	PHI1, PHI2 pins only	−0.5		0.5	V
V_{OH}	High Level Output Voltage	I_{OH} = −400 μA	2.4			V
V_{OL}	Low Level Output Voltage	I_{OL} = 2 mA			0.45	V
I_{ILS}	$\overline{\text{SPC}}$ and $\overline{\text{DT}}/\overline{\text{SDONE}}$ Input Current (low)	V_{IN} = 0.4V, $\overline{\text{SPC}}$ in input mode	0.05		1.0	mA
I_I	Input Load Current	0 ≤ V_{IN} ≤ V_{CC}, Input Pins except PHI1, PHI2, $\overline{\text{DT}}/\overline{\text{SDONE}}$	−20		20	μA
I_L	Leakage Current (Output and I/O pins in TRI-STATE/Input Mode)	0.4 ≤ V_{IN} ≤ V_{CC}	−80		80	μA
I_{CC}	Active Supply Current	I_{OUT} = 0, T_A = 25°C		450	600	mA

Connection Diagram*



TL/EE/8673-45

Bottom View

NS32332 Pinout Descriptions
84 Pin Grid Array

Desc	Pin	Desc	Pin	Desc	Pin
GNDB1	B1	AD29	N6	$\overline{\text{BOUT}}$	E12
AD6	B2	AD30	M6	$\overline{\text{SPC}}$	D13
AD7	C1	AD31	N7	$\overline{\text{MC/EXS}}$	D12
AD8	C2	VCCL1	M7	VCCB5	C13
AD9	D1	VCCL2	N8	$\overline{\text{ADS}}$	C12
AD10	D2	$\overline{\text{INT}}$	M8	GNDB6	B13
AD11	E1	$\overline{\text{NMI}}$	N9	$\overline{\text{DDIN}}$	A12
GNDB2	E2	RESERVED	M9	$\overline{\text{BE0}}$	B12
AD12	F1	RESERVED	N10	$\overline{\text{BE1}}$	A11
AD13	F2	RESERVED	M10	$\overline{\text{BE2}}$	B11
AD14	G1	RESERVED	N11	$\overline{\text{BE3}}$	A10
AD15	G2	$\overline{\text{ILO}}$	M11	$\overline{\text{HLDA}}$	B10
VCCB2	H1	VCCB4	N12	$\overline{\text{HOLD}}$	A9
AD16	H2	ST3	M13	RDY	B9
AD17	J1	ST2	M12	$\overline{\text{DT}}/\overline{\text{SDONE}}$	A8
AD18	J2	ST1	L13	PHI 2	B8
AD19	K1	ST0	L12	PHI 1	A7
GNDB3	K2	$\overline{\text{STS}}$	K13	BBG	B7
AD20	L1	GNDB5	K12	GNDL2	A6
AD21	L2	$\overline{\text{PFS}}$	J13	GNDL1	B6
AD22	M1	U/$\overline{\text{S}}$	J12	VCCB1	A5
AD23	N2	BW1	H13	AD0	B5
VCCB3	M2	BW0	H12	AD1	A4
AD24	N3	$\overline{\text{BIN}}$	G13	AD2	B4
AD25	M3	$\overline{\text{FLT}}$	G12	AD3	A3
AD26	N4	$\overline{\text{RST}}/\overline{\text{ABT}}$	F13	AD4	B3
AD27	M4	$\overline{\text{BRT}}$	F12	AD5	A2
GNDB4	N5	$\overline{\text{BER}}$	E13	POSITION PIN	C3
AD28	M5				

Order Number NS32332U-10 or NS32332U-15
See NS Package Number U84C

FIGURE 4-1. Pin Grid Array Package

*AMP sockets are recommended for use with NS32332 CPU. AMP sockets are manufactured by AMP INCORPORATED, Harrisburg PA.

4.0 Device Specifications (Continued)

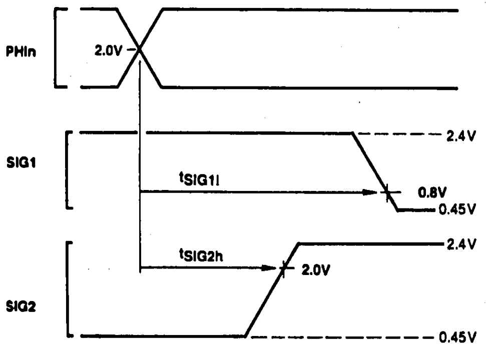
4.4 SWITCHING CHARACTERISTICS

4.4.1 Definitions

All the timing specifications given in this section refer to 2.0V on the rising or falling edges of the clock phases PHI1 and PHI2 and 0.8V or 2.0V on all other signals as illustrated below, unless specifically stated otherwise.

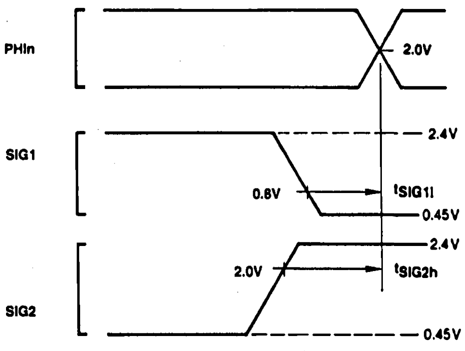
ABBREVIATIONS:

L.E. — leading edge	R.E. — rising edge
T.E. — trailing edge	F.E. — falling edge



TL/EE/8673-46

FIGURE 4-2. Timing Specification Standard (Signal Valid After Clock Edge)



TL/EE/8673-47

FIGURE 4-3. Timing Specification Standard (Signal Valid Before Clock Edge)

4.4.2 Timing Tables

4.4.2.1 Output Signals: Internal Propagation Delays, NS32332-10, NS32332-15

Maximum times assume capacitive loading of 100 pF.

AD0–31, $\overline{\text{ADS}}$ and $\overline{\text{BOUT}}$ timings are defined with a capacitive loading of 75 pF.

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-15		Units
				Min	Max	Min	Max	
t_{ALv}	4-5	Address bits 0–31 valid	after R.E., PHI1 T1		30		20	ns
t_{ALh}	4-5	Address bits 0–31 hold	after R.E., PHI1 T2/Tmmu	10		6		ns
t_{Dv}	4-5	Data valid (write cycle)	after R.E., PHI1 T3 or T2		50		38	ns
t_{Dh}	4-5	Data hold (write cycle)	after R.E., PHI1 next T1 or Ti	0		0		ns
t_{ALADSs}	4-4	Address bits 0–31 setup	before $\overline{\text{ADS}}$ T.E.	25		20		ns
t_{ALADSh}	4-18	Address bits 0–31 hold	after $\overline{\text{ADS}}$ T.E.	10		10		ns
t_{ALf}	4-4	Address bits 0–31 floating (no MMU)	after R.E., PHI1 T2/Tmmu		25		24	ns
t_{ALMf}	4-18	Address bits 0–31 floating (by $\overline{\text{FLT}}$ line)	after R.E., PHI1 Tf		40		40	ns
t_{STSa}	4-3,4-5	$\overline{\text{STS}}$ signal active (low)	after R.E., PHI1 T4 of previous bus cycle or Ti		35		25	ns
t_{STSia}	4-3,4-5	$\overline{\text{STS}}$ signal inactive	after R.E., PHI2 T4 of previous bus cycle or Ti		45		30	ns
t_{STSw}	4-3	$\overline{\text{STS}}$ pulse width	at 0.8V (both edges)	35		24		ns
t_{BErv}	4-4,4-6	$\overline{\text{BEn}}$ signals valid (Operand Read Cycles Only)	after R.E., PHI2, T4 or Ti		140		95	ns
t_{BEv}	4-5,4-6	$\overline{\text{BEn}}$ signals valid	after R.E., PHI2, T4 or Ti		85		58	ns
t_{BEh}	4-4	$\overline{\text{BEn}}$ signals hold	after R.E., PHI2, T4	0		0		ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32332-10, NS32332-15 (Continued)

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-15		Units
				Min	Max	Min	Max	
t_{STv}	4-5	Status (ST0–ST3) valid	after R.E., PHI1 T4 (before T1, see note)		50		35	ns
t_{STSTSs}	4-5	Status Signals Setup	Before $\overline{\text{STS}}$ T.E.	10		6		ns
t_{STh}	4-5	Status (ST0–ST3) hold	after R.E., PHI1 T4 (after T1)	0		0		ns
t_{DDINv}	4-4	$\overline{\text{DDIN}}$ signal valid	after R.E., PHI1 T1		35		25	ns
t_{DDINh}	4-4	$\overline{\text{DDIN}}$ signal hold	after R.E., PHI1 next T1 or Ti	0		0		ns
t_{ADSa}	4-5	$\overline{\text{ADS}}$ signal active (low)	after R.E., PHI1 T1		25		17	ns
t_{ADSia}	4-5	$\overline{\text{ADS}}$ signal inactive	after R.E., PHI2 T1		45		29	ns
t_{ADSw}	4-5	$\overline{\text{ADS}}$ pulse width	at 0.8V (both edges)	35		24		ns
t_{MCa}	4-4,4-5	$\overline{\text{MC}}$ signal active (low)	after R.E., PHI1 T1		70		50	ns
t_{MCia}	4-4,4-5	$\overline{\text{MC}}$ signal inactive	after R.E., PHI1 T1 or T3 (burst)		70		50	ns
t_{ALf}	4-15	AD0–AD31 floating (caused by $\overline{\text{HOLD}}$)	after R.E., PHI1 T1		25		24	ns
t_{ADSf}	4-15, 4-17	$\overline{\text{ADS}}$ floating (caused by $\overline{\text{HOLD}}$)	after R.E., PHI1 Ti		55		40	ns
t_{BEf}	4-15, 4-17	$\overline{\text{BEn}}$ floating (caused by $\overline{\text{HOLD}}$)	after R.E., PHI1 Ti		55		40	ns
t_{DDINf}	4-15, 4-17	$\overline{\text{DDIN}}$ floating (caused by $\overline{\text{HOLD}}$)	after R.E., PHI1 Ti		55		45	ns
t_{HLDAa}	4-15, 4-16	$\overline{\text{HLDA}}$ signal active (low)	after R.E., PHI1 T4		60		45	ns
t_{HLDAia}	4-18	$\overline{\text{HLDA}}$ signal inactive	after R.E., PHI1 Ti		60		45	ns
t_{ADSr}	4-18	$\overline{\text{ADS}}$ signal returns from floating (caused by $\overline{\text{HOLD}}$)	after R.E., PHI1 Ti		55		40	ns
t_{BEr}	4-18	$\overline{\text{BEn}}$ signals return from floating (caused by $\overline{\text{HOLD}}$)	after R.E., PHI1 Ti		55		40	ns
t_{DDINr}	4-18	$\overline{\text{DDIN}}$ signal returns from floating (caused by $\overline{\text{HOLD}}$)	after R.E., PHI1 Ti		55		40	ns
t_{DDINf}	4-19	$\overline{\text{DDIN}}$ signal floating (caused by $\overline{\text{FLT}}$)	after $\overline{\text{FLT}}$ F.E.		50		45	ns
t_{DDINr}	4-20	$\overline{\text{DDIN}}$ signal returns from floating (caused by $\overline{\text{FLT}}$)	after $\overline{\text{FLT}}$ R.E.		40		28	ns
t_{SPCa}	4-21	$\overline{\text{SPC}}$ output active (low)	after R.E., PHI1 T1		30		21	ns
t_{SPCia}	4-21	$\overline{\text{SPC}}$ output inactive	after R.E., PHI1 T4	2	35	2	26	ns
t_{SPCnf}	4-24	$\overline{\text{SPC}}$ output nonforcing	after R.E., PHI2 T4		10		8	ns
t_{Dv}	4-21	Data valid (slave processor write)	after R.E., PHI1 T1		50		38	ns
t_{Dh}	4-21	Data hold (slave processor write)	after R.E., PHI1 next T1 or Ti	0		0		ns
t_{PFSw}	4-26	$\overline{\text{PFS}}$ pulse width	at 0.8V (both edges)	70		45		ns
t_{PFSa}	4-26	$\overline{\text{PFS}}$ pulse active (low)	after R.E., PHI2		50		38	ns
t_{PFSia}	4-26	$\overline{\text{PFS}}$ pulse inactive	after R.E., PHI2		50		38	ns
t_{USv}	4-33	U/$\overline{\text{S}}$ signal valid	after R.E., PHI1 T4		48		35	ns
t_{USh}	4-33	U/$\overline{\text{S}}$ signal hold	after R.E., PHI1 T4	10		6		ns
t_{NSPF}	4-28	Nonsequential fetch to next $\overline{\text{PFS}}$ clock cycle	after R.E., PHI1 T1	4		4		t_{Cp}

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32332-10, NS32332-15 (Continued)

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-15		Units
				Min	Max	Min	Max	
t_{PFNS}	4-27	$\overline{PFS}$ clock cycle to next non-sequential fetch	before R.E., PHI1 T1	4		4		t_{Cp}
t_{STSf}	4-15, 4-16	$\overline{STS}$ floating (HOLD)	after R.E., PHI1 Ti		55		44	ns
t_{STSr}	4-18	$\overline{STS}$ not floating (HOLD)	after R.E., PHI1 Ti, T4		55		40	ns
t_{BOUTa}	4-6, 4-10	$\overline{BOUT}$ output active	after R.E., PHI2 Tmmu		100		66	ns
t_{BOUTia}	4-6, 4-10	$\overline{BOUT}$ output inactive	after R.E., PHI2 T3 or T4		75		40	ns
t_{ILOa}	4-14	$\overline{ILO}$ signal active	after R.E., PHI1 T4		50		38	ns
t_{ILOia}	4-14	$\overline{ILO}$ signal inactive	after R.E., PHI1 Ti		50		38	ns

Note: Every memory cycle starts with T4, during which Cycle Status is applied. If the CPU was idling, the sequence will be: ". . . Ti, T4, T1 . . .". If the CPU was not idling, the sequence will be: ". . . T4, T1 . . .".

4.4.2.2 Input Signal Requirements: NS32332-10, NS32332-15

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-15		Units
				Min	Max	Min	Max	
t_{PWR}	4-31	Power stable to $\overline{RST}$ R.E.	after V_{CC} reaches 4.5V	50		33		μs
t_{DIs}	4-4	Data in setup (read cycle)	before F.E., PHI2 T3	12		10		ns
t_{DIh}	4-4	Data in hold (read cycle)	after R.E., PHI1 T4	3		3		ns
t_{HLDa}	4-15 4-16,	$\overline{HOLD}$ active setup time	before F.E., PHI2 T2/Tmmu or T3 or Ti	25		17		ns
t_{HLDia}	4-18	$\overline{HOLD}$ inactive setup time	before F.E., PHI2 Ti	25		17		ns
t_{HLDh}	4-15, 4-17, 4-18	$\overline{HOLD}$ hold time	after R.E., PHI1 Ti or T3	0		0		ns
t_{FLTa}	4-19	$\overline{FLT}$ active (low) setup time	before F.E., PHI2 Tmmu	25		17		ns
t_{FLTia}	4-20	$\overline{FLT}$ inactive setup time	before F.E., PHI2 T3	25		17		ns
t_{RDYs}	4-4, 4-5, 4-6	RDY setup time	before F.E., PHI1 T3	20		12		ns
t_{RDYh}	4-4, 4-5, 4-6	RDY hold time	after R.E., PHI2 T3	4		3		ns
t_{ABTs}	4-29	$\overline{ABT}$ setup time ($\overline{FLT}$ inactive)	before F.E., PHI2 T2/Tmmu	20		13		ns
t_{ABTs}	4-30	$\overline{ABT}$ setup time ($\overline{FLT}$ active)	before F.E., PHI2 Tf	20		13		ns
t_{ABTh}	4-29, 4-30	$\overline{ABT}$ hold time	after R.E., PHI1 T3	0		0		ns
t_{RSTs}	4-31, 4-32	$\overline{RST}$ setup time	before F.E., PHI1	20		13		ns
t_{RSTw}	4-31, 4-32	$\overline{RST}$ pulse width	at 0.8V (both edges)	64		64		t_{Cp}
t_{INTs}	4-34	$\overline{INT}$ setup time	before F.E., PHI2	20		13		ns
t_{NMIw}	4-35	$\overline{NMI}$ pulse width	at 0.8V (both edges)	40		27		ns

4.0 Device Specifications (Continued)

4.4.2.2 Input Signal Requirements: NS32332-10, NS32332-15 (Continued)

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-15		Units
				Min	Max	Min	Max	
t_{DIs}	4-24	Data setup (slave read cycle)	before F.E., PHI2 T1	12		10		ns
t_{DIh}	4-24	Data hold (slave read cycle)	after R.E., PHI1 T4	3		3		ns
t_{DTs}	4-31	$\overline{DT}$ setup time	before F.E., PHI1	0		0		ns
t_{DTh}	4-31	$\overline{DT}$ hold time	after R.E., PHI1	0		0		ns
t_{SPCd}	4-24	$\overline{SPC}$ pulse delay from slave	after R.E., PHI2 T4	10		8		ns
t_{SPCs}	4-24	$\overline{SPC}$ setup time	before F.E., PHI1	25		15		ns
t_{SPCw}	4-24	$\overline{SPC}$ pulse width	at 0.8V (both edges)	20	100	13	66	ns
t_{SDNd}	4-23	$\overline{SDONE}$ pulse delay from slave	after R.E., PHI2 T4	10		8		ns
t_{SDNs}	4-23	$\overline{SDONE}$ setup time	before F.E., PHI1	25		15		ns
t_{SDNw}	4-23	$\overline{SDONE}$ pulse width	at 0.8V (both edges)	20	100	13	66	ns
t_{SDNSTw}	4-23	$\overline{SDONE}$ pulse width (to force CPU to read slave status)	at 0.8V (both edges)	175	275	115	200	ns
t_{BWs}	4-4, 4-5 4-6	$\overline{BW}$ 0–1 setup time	before F.E., PHI1 T3	25		13		ns
t_{BWh}	4-6	$\overline{BW}$0-1 hold time	after R.E., PHI1 T3 of Next Memory Access Cycle	0		0		ns
t_{BINs}	4-6, 4-7	$\overline{BIN}$ setup time (for each cycle of the burst)	before F.E., PHI1 T3	25		12		ns
t_{BINh}	4-6, 4-7	$\overline{BIN}$ hold time	after R.E., PHI1 T4	0		0		ns
t_{BERs}	4-12, 4-13	$\overline{BER}$ setup time	before F.E., PHI1 T4	25		14		ns
t_{BERh}	4-12, 4-13	$\overline{BER}$ hold time (see note)	after R.E., PHI1 Ti	0		0		ns
t_{BRTs}	4-8, 4-9, 4-10, 4-11	$\overline{BRT}$ setup time	before F.E., PHI1 T3 and T4	25		14		ns
t_{BRTh}	4-8, 4-9, 4-10	$\overline{BRT}$ hold time	after R.E., PHI1 T4 or Ti	0		0		ns

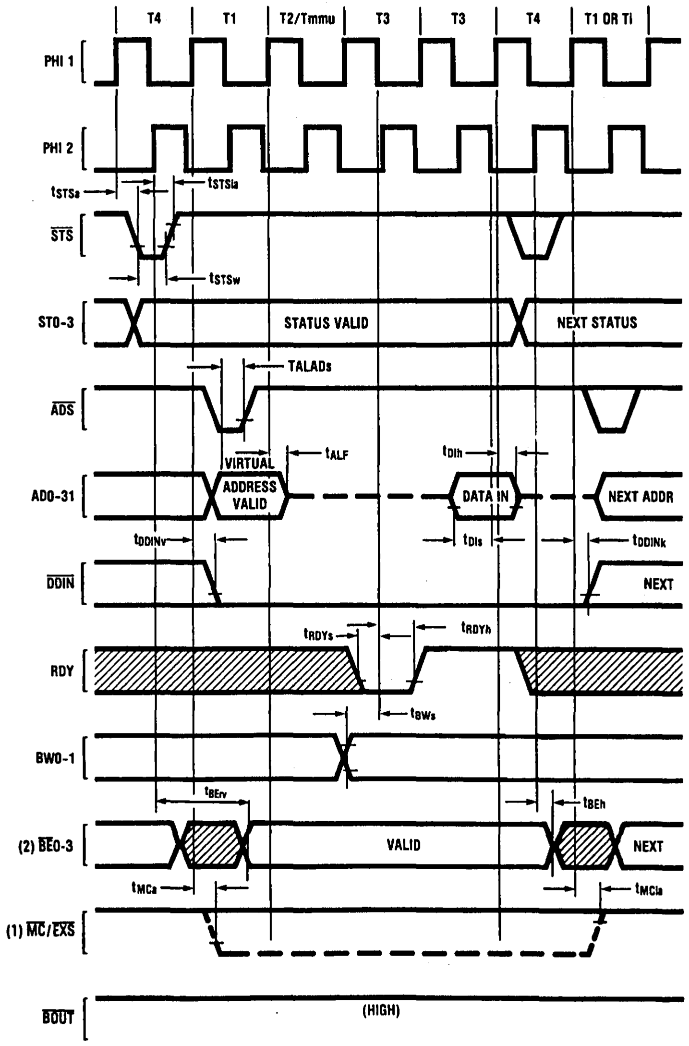
Note: A Ti state follows T4 when $\overline{BER}$ is asserted. $\overline{BER}$ should be deasserted at the latest in the beginning of the cycle following this Ti state.

4.4.2.3 Clocking Requirements: NS32332-10, NS32332-15

Symbol	Figure	Description	Reference/ Conditions	NS32332-10		NS32332-15		Units
				Min	Max	Min	Max	
t_{Cp}	4-25	Clock period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	100	250	66	250	ns
$t_{CLw(1,2)}$	4-25	PHI1, PHI2 Pulse Width	At 2.0V on PHI1, PHI2 (Both Edges)	0.5 t_{cp} − 10 ns		0.5 t_{cp} − 6 ns		
$t_{CLh(1,2)}$	4-25	PHI1, PHI2 high time	At V_{CC}−0.9V on PHI1, PHI2 (Both Edges)	0.5 t_{cp} − 15 ns		0.5 t_{cp} − 10 ns		
t_{CLl}	4-25	PHI1, PHI2 low time	At 0.8V on PHI1, PHI2 (Both Edges)	0.5 t_{cp} −5 ns		0.5 t_{cp} −5 ns		
$t_{nOVL(1,2)}$	4-25	Non-overlap time	0.8V on F.E., PHI1, PHI2 to 0.8V on R.E., PHI2, PHI1	−2	2	−2	2	ns
t_{nOVLas}		Non-overlap asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	At 0.8V on PHI1, PHI2	−3	3	−3	3	ns
t_{CLhas}		PHI1, PHI2 asymmetry ($t_{CLh(1)} - t_{CLh(2)}$)	At V_{CC}−0.9V on PHI1, PHI2	−5	5	−3	3	ns

4.0 Device Specifications (Continued)

4.4.3 Timing Diagrams



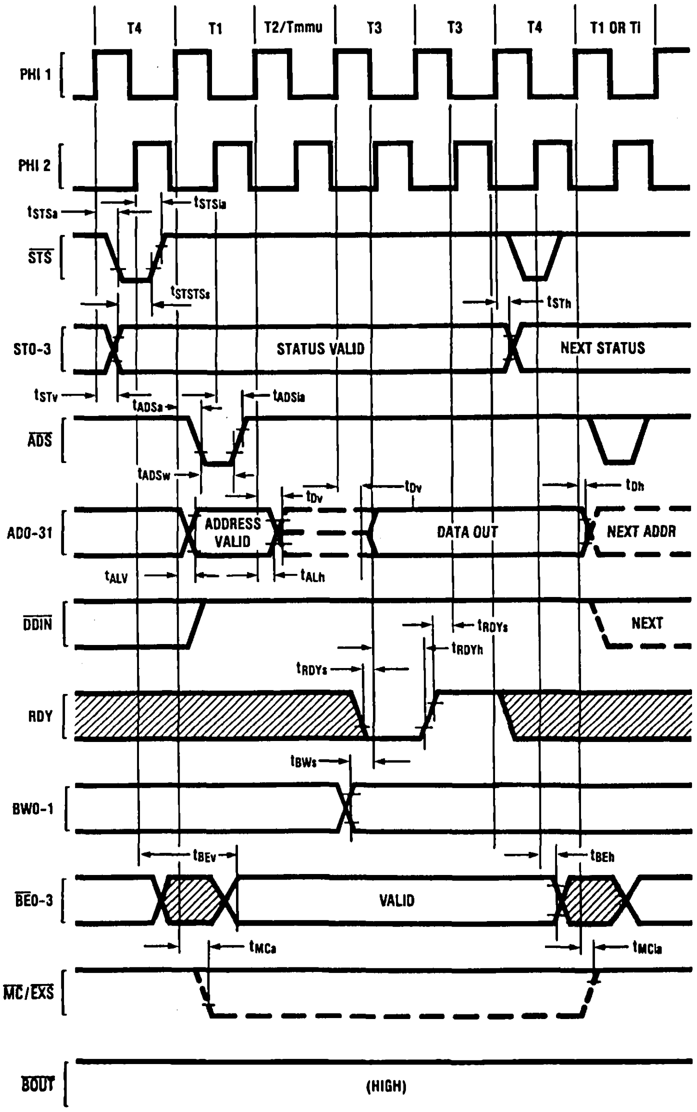
TL/EE/8673-48

Note 1: Asserted (low) when the bus transaction crosses a double-word boundary (address bits A0-1 wrap around during the transaction).

Note 2: $\overline{BE0}$-$\overline{BE3}$ are all active during instruction fetch cycles.

FIGURE 4-4. NS32332 Read Cycle Timing

4.0 Device Specifications (Continued)

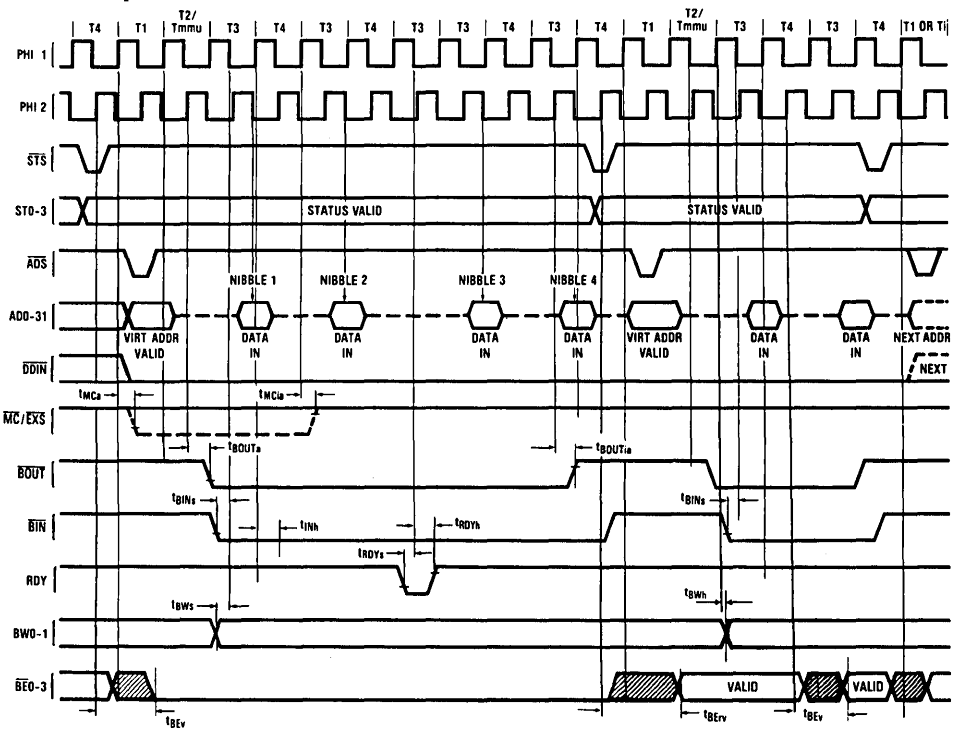


TL/EE/8673-49

Note: If DT/SDONE is sampled low during reset, the CPU outputs the data during T2/TMMU (see Section 3.3).

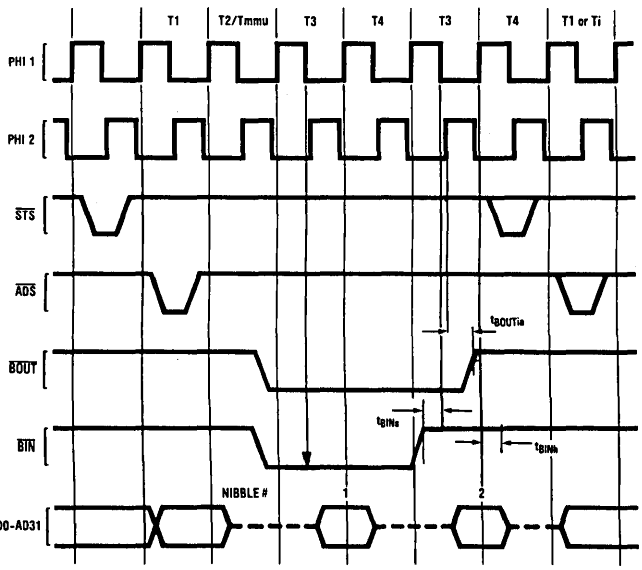
FIGURE 4-5. NS32332 Write Cycle Timing

4.0 Device Specifications (Continued)



TL/EE/8673-50

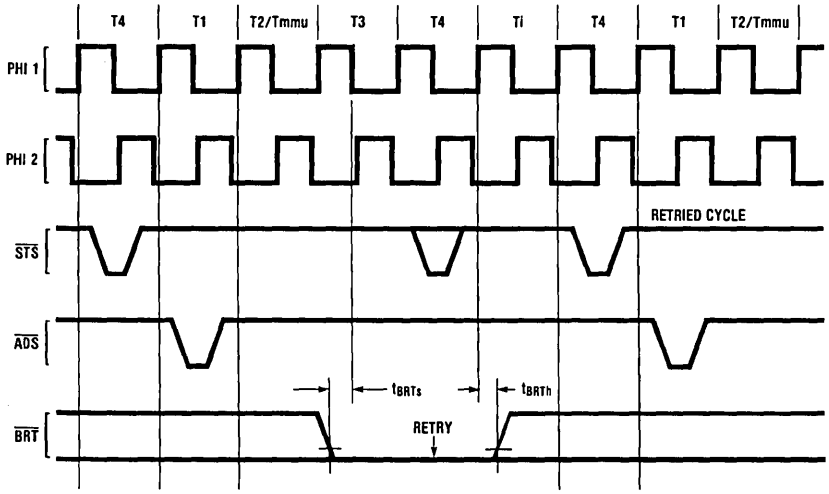
FIGURE 4-6. NS32332 Burst Cycle Timing (Instruction fetches followed by Operand Reads)



TL/EE/8673-94

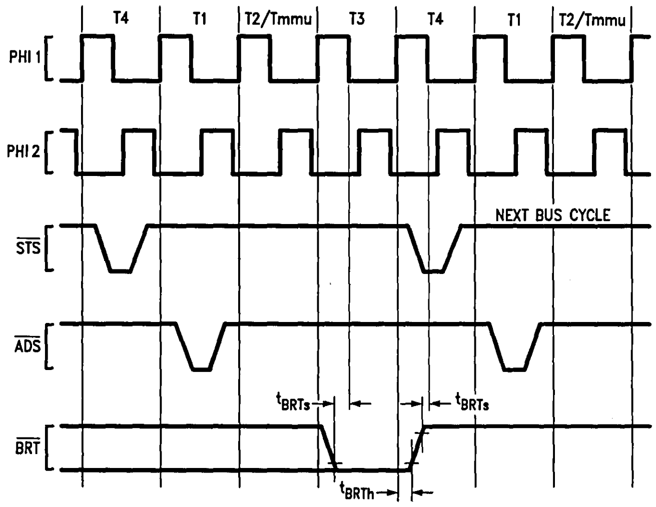
FIGURE 4-7. External Termination of Burst Cycle

4.0 Device Specifications (Continued)



TL/EE/8673-51

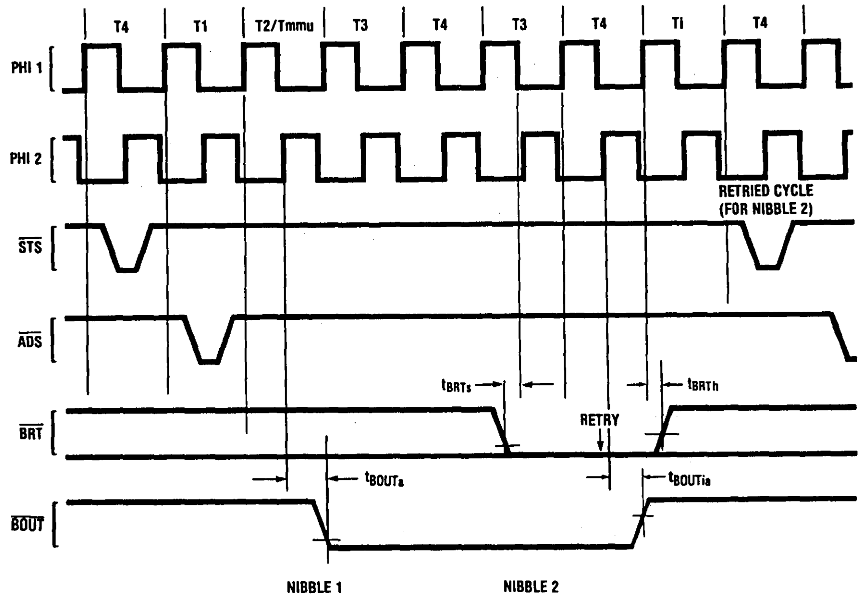
FIGURE 4-8. Bus Retry During Normal Bus Cycle



TL/EE/8673-52

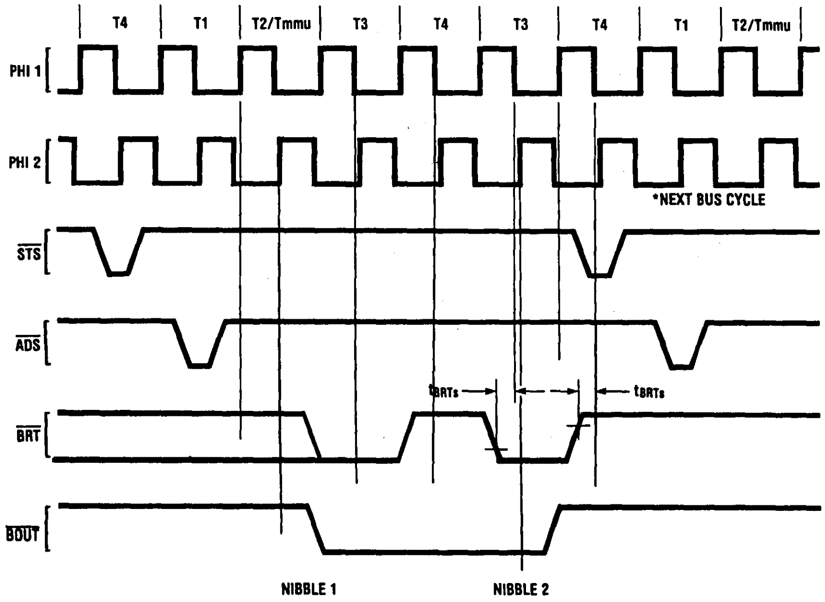
FIGURE 4-9. $\overline{\text{BRT}}$ Activated, but no Bus Retry

4.0 Device Specifications (Continued)



TL/EE/8673-53

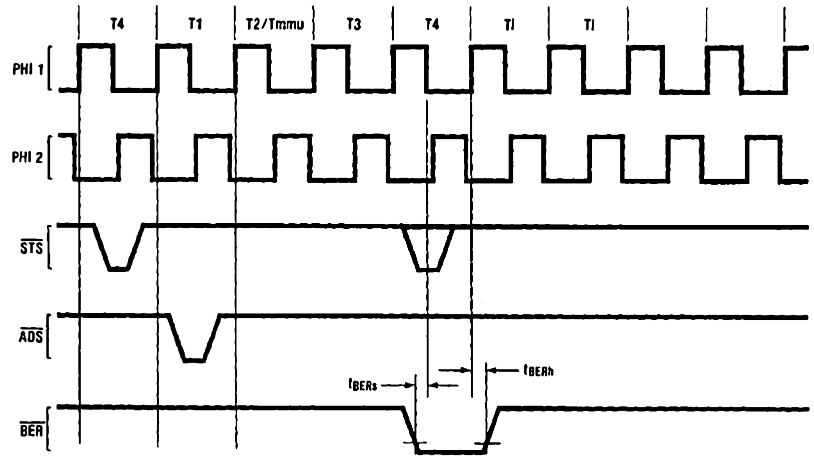
FIGURE 4-10. Bus Retry During Burst Bus Cycle



TL/EE/8673-54

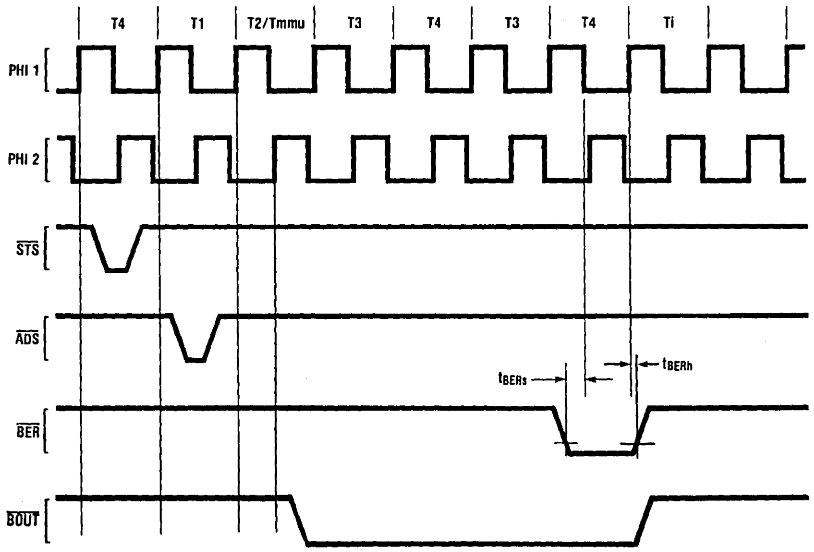
FIGURE 4-11. $\overline{\text{BRT}}$ Activated During Burst Bus Cycle, but no Bus Retry

4.0 Device Specifications (Continued)



TL/EE/8673-55

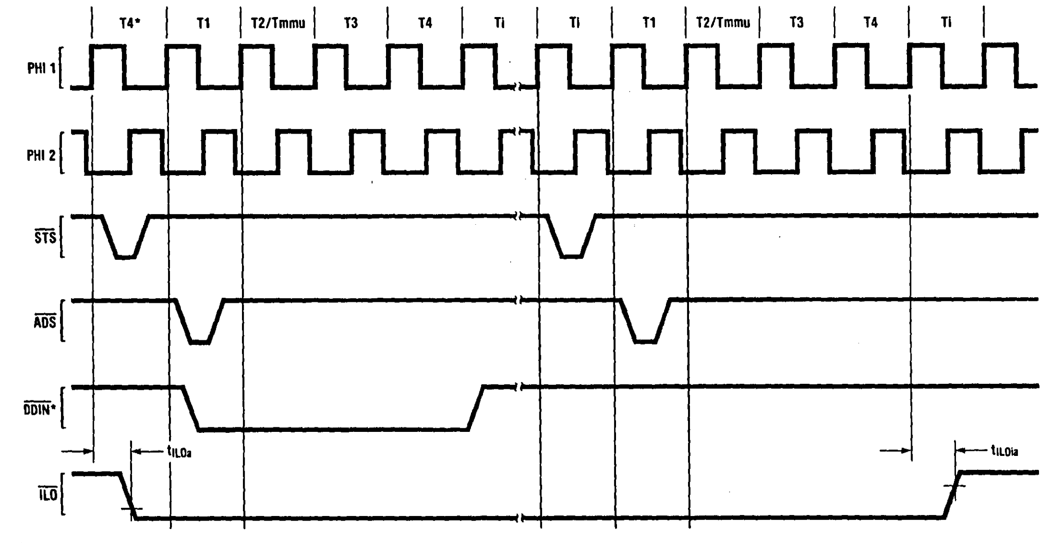
FIGURE 4-12. Bus Error During Normal Bus Cycle



TL/EE/8673-56

FIGURE 4-13. Bus Error During Burst Bus Cycle

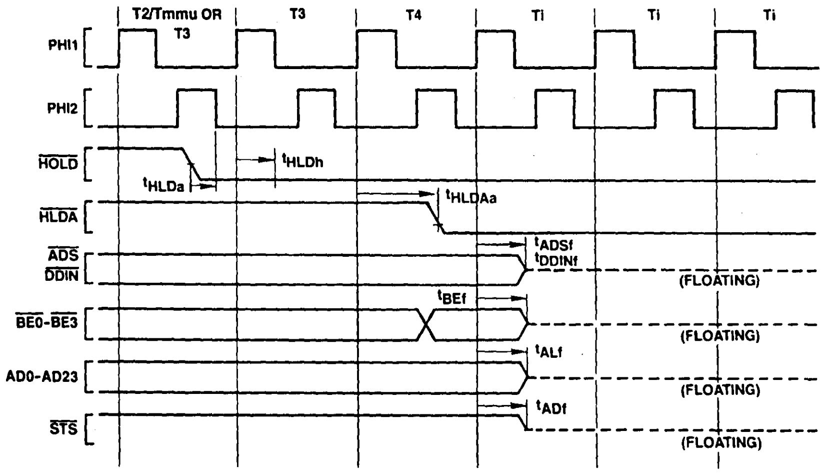
4.0 Device Specifications (Continued)



*End of Dummy Read cycle with the address of the interlocked operand.

TL/EE/8673-57

FIGURE 4-14. Timing of Interlocked Bus Transactions

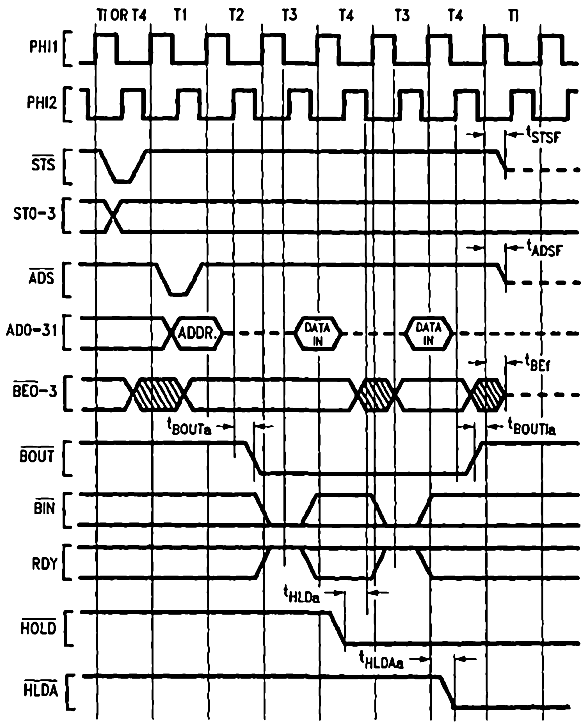


TL/EE/8673-58

FIGURE 4-15. Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially)

Note: Whenever the CPU is not idling (not in Ti), the $\overline{\text{HOLD}}$ signal must be active before the falling edge of PHI2 of the clock cycle that appears two clock cycles before T4 (TX1) and stay low until after the rising edge of PHI1 of the clock cycle that precedes T4 (TX2) for the request to be acknowledged.

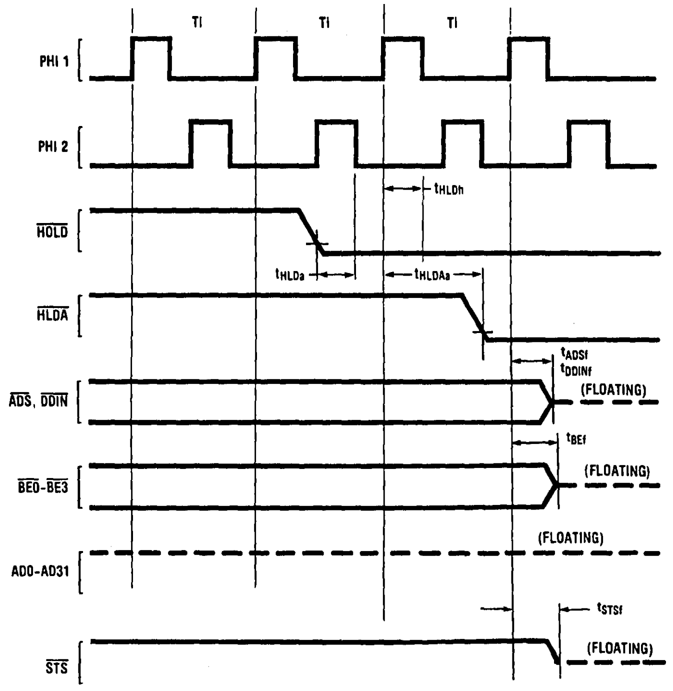
4.0 Device Specifications (Continued)



TL/EE/8673-90

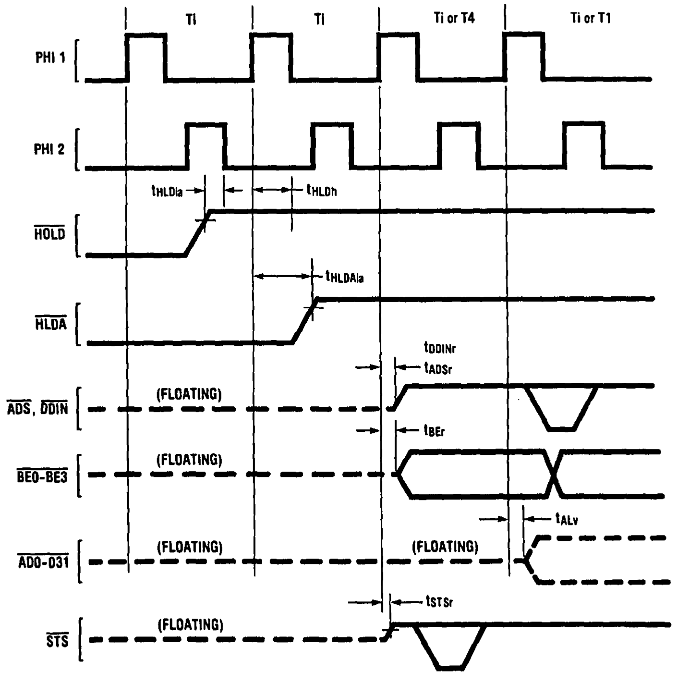
FIGURE 4-16. Floating by $\overline{\text{HOLD}}$ Timing (Burst Cycle Ended by $\overline{\text{HOLD}}$ Assertion)

4.0 Device Specifications (Continued)



TL/EE/8673-59

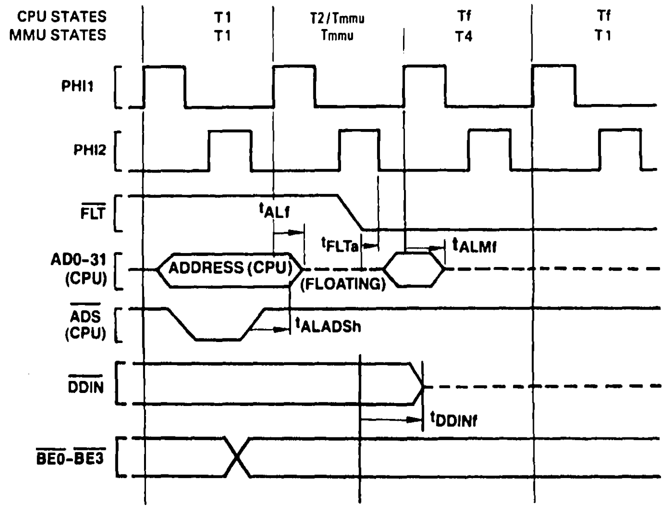
FIGURE 4-17. Floating by $\overline{\text{HOLD}}$ Timing (CPU Initially Idle)



TL/EE/8673-60

FIGURE 4-18. Release from $\overline{\text{HOLD}}$

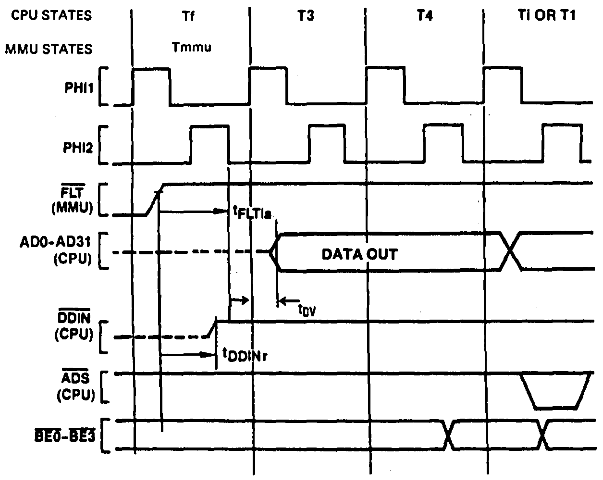
4.0 Device Specifications (Continued)



TL/EE/8673-61

Note: The bus lines AD0-31 are temporarily driven in T2/TMMU and T_f when $\overline{FLT}$ is asserted only if $\overline{DT}/\overline{SDONE}$ is sampled low during reset (see Section 3.3).

FIGURE 4-19. $\overline{FLT}$ Initiated Cycle Timing

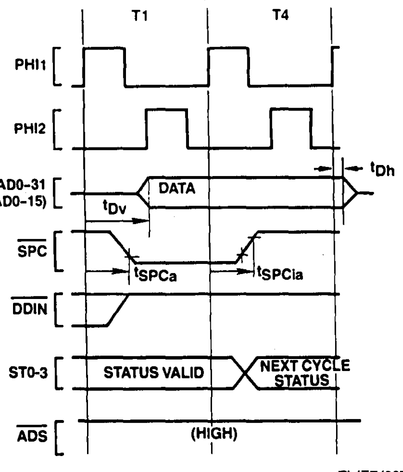


TL/EE/8673-62

FIGURE 4-20. Release from $\overline{FLT}$ Timing (CPU Write Cycle)

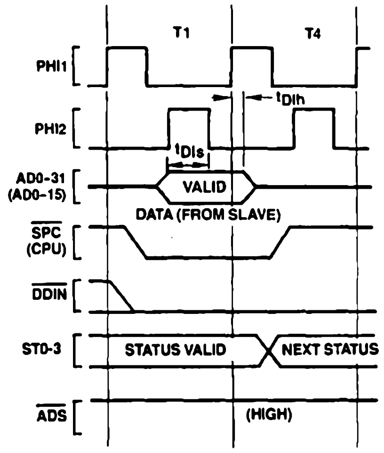
Note: When $\overline{FLT}$ is deasserted the CPU restarts driving $\overline{DDIN}$ before the MMU releases it. This, however, does not cause any conflict, since both CPU and MMU force $\overline{DDIN}$ to the same logic level.

4.0 Device Specifications (Continued)



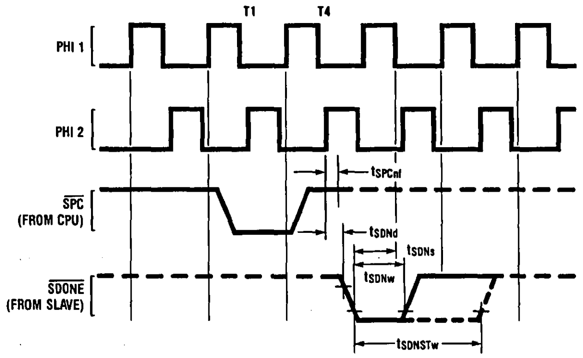
TL/EE/8673-64

FIGURE 4-21. Slave Processor Write Timing



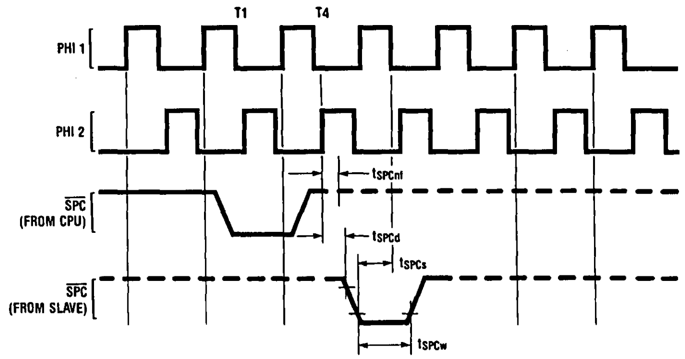
TL/EE/8673-65

FIGURE 4-22. Slave Processor Read Timing



TL/EE/8673-63

FIGURE 4-23. $\overline{\text{DT}}/\overline{\text{SDONE}}$ Timing (32-Bit Slave Protocol)

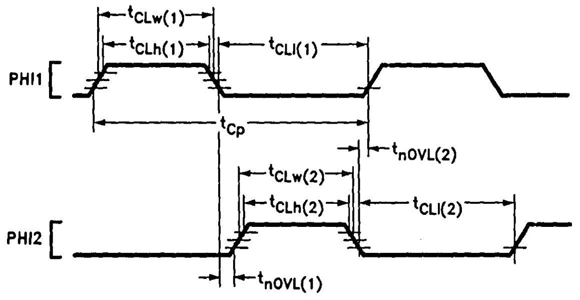


TL/EE/8673-66

FIGURE 4-24. $\overline{\text{SPC}}$ Timing (16-Bit Slave Protocol)

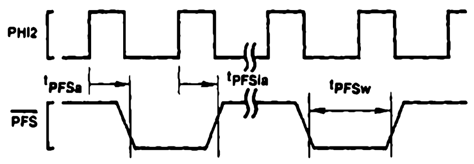
Note: After transferring last operand to a Slave Processor, CPU turns OFF driver and holds $\overline{\text{SPC}}$ high with internal 5 kΩ pullup.

4.0 Device Specifications (Continued)



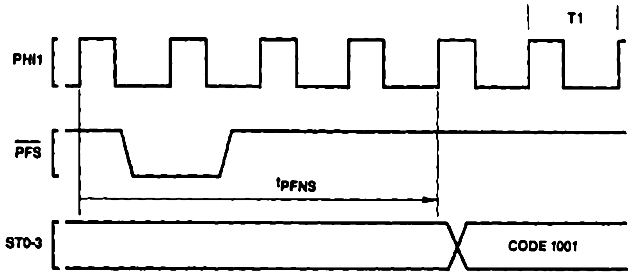
TL/EE/8673-91

FIGURE 4-25. Clock Waveforms



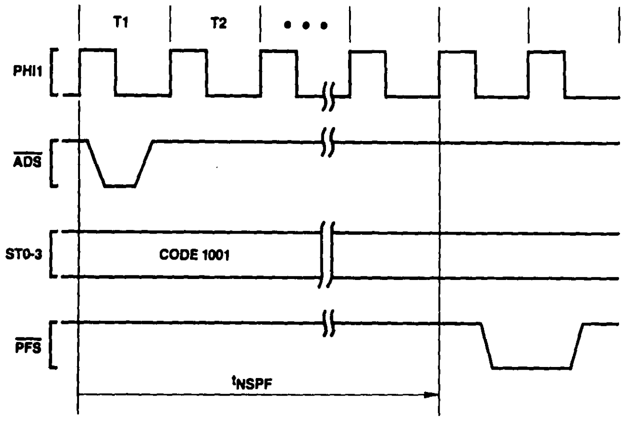
TL/EE/8673-68

FIGURE 4-26. Relationship of $\overline{PFS}$ to Clock Cycles



TL/EE/8673-69

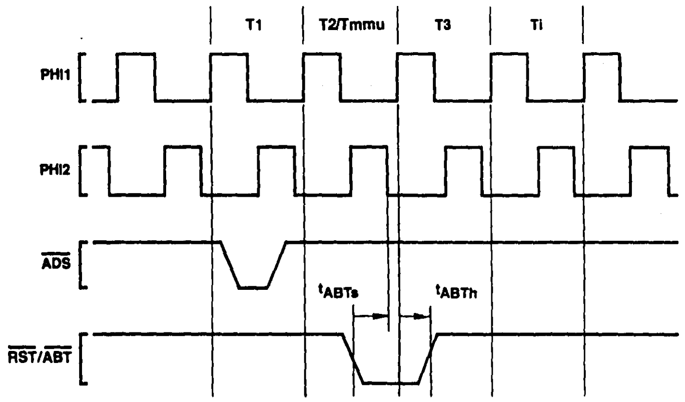
FIGURE 4-27. Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch



TL/EE/8673-70

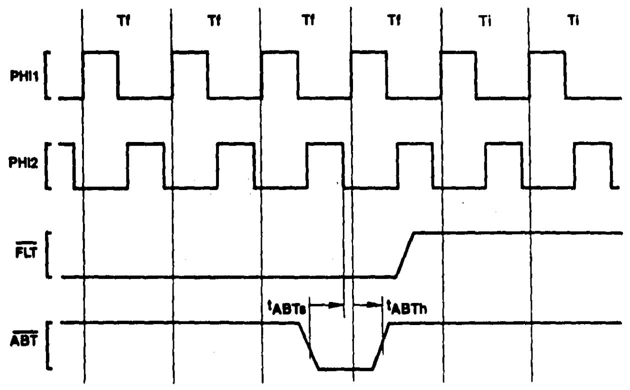
FIGURE 4-28. Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$

4.0 Device Specifications (Continued)



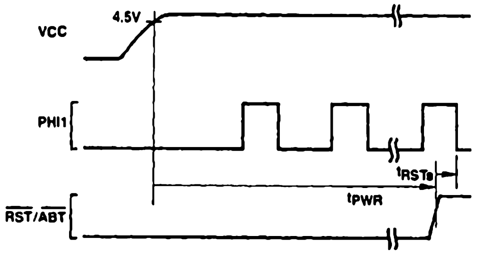
TL/EE/8673-71

FIGURE 4-29. Abort Timing, $\overline{\text{FLT}}$ Not Applied



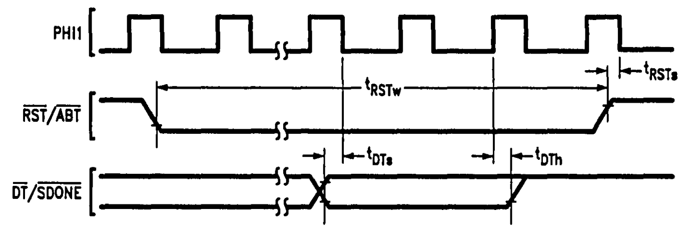
TL/EE/8673-72

FIGURE 4-30. Abort Timing, $\overline{\text{FLT}}$ Applied



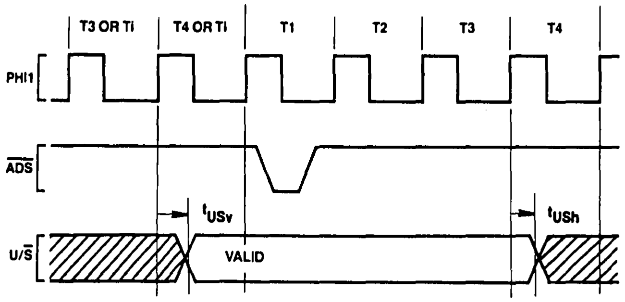
TL/EE/8673-73

FIGURE 4-31. Power-On Reset



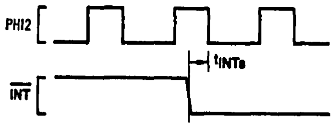
TL/EE/8673-92

FIGURE 4-32. Non-Power-On Reset



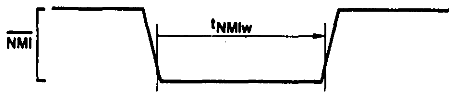
TL/EE/8673-75

FIGURE 4-33. U/$\overline{S}$ Relationship to Any Bus Cycle — Guaranteed Valid Interval



TL/EE/8673-76

FIGURE 4-34. $\overline{INT}$ Interrupt Signal Detection



TL/EE/8673-77

FIGURE 4-35. $\overline{NMI}$ Interrupt Signal Timing

Appendix A: Instruction Formats

NOTATIONS

i = Integer Type Field
- B = 00 (Byte)
- W = 01 (Word)
- D = 11 (Double Word)

f = Floating Point Type Field
- F = 1 (Std. Floating: 32 bits)
- L = 0 (Long Floating: 64 bits)

c = Custom Type Field
- D = 1 (Double Word)
- Q = 0 (Quad Word)

op = Operation Code
- Valid encodings shown with each format.

gen, gen 1, gen 2 = General Addressing Mode Field
- See Sec. 2.2 for encodings.

reg = General Purpose Register Number

cond = Condition Code Field

0000	= EQual: Z = 1
0001	= Not Equal: Z = 0
0010	= Carry Set: C = 1
0011	= Carry Clear: C = 0
0100	= HIgher: L = 1
0101	= Lower or Same: L = 0
0110	= Greater Than: N = 1
0111	= Less or Equal: N = 0
1000	= Flag Set: F = 1
1001	= Flag Clear: F = 0
1010	= LOwer: L = 0 and Z = 0
1011	= Higher or Same: L = 1 or Z = 1
1100	= Less Than: N = 0 and Z = 0
1101	= Greater or Equal: N = 1 or Z = 1
1110	= (Unconditionally True)
1111	= (Unconditionally False)

short = Short Immediate value. May contain
- quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.
- cond: Condition Code (above), in Scond.
- areg: CPU Dedicated Register, in LPR, SPR.

0000	= US
0001 – 0111	= (Reserved)
1000	= FP
1001	= SP
1010	= SB
1011	= (Reserved)
1100	= (Reserved)
1101	= PSR
1110	= INTBASE
1111	= MOD

Options: in String Instructions

U/W	B	T

T = Translated

B = Backward

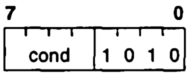
U/W = 00: None
- 01: While Match
- 11: Until Match

Configuration bits in SETCFG Instruction:

P	FC	FM	FF	C	M	F	I

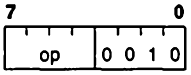
mreg: NS32382 Register number, in LMR, SMR.

0000	= BAR
0001	= (Reserved)
0010	= BMR
0011	= BDR
0100	= (Reserved)
0101	= (Reserved)
0110	= BEAR
0111	= (Reserved)
1000	= (Reserved)
1001	= MCR
1010	= MSR
1011	= TEAR
1100	= PTB0
1101	= PTB1
1110	= IVAR0
1111	= IVAR1



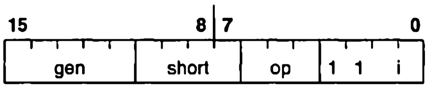
Format 0

Bcond (BR)



Format 1

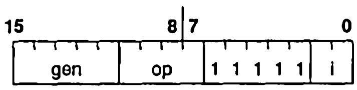
BSR	-0000	ENTER	-1000
RET	-0001	EXIT	-1001
CXP	-0010	NOP	-1010
RXP	-0011	WAIT	-1011
RETT	-0100	DIA	-1100
RETI	-0101	FLAG	-1101
SAVE	-0110	SVC	-1110
RESTORE	-0111	BPT	-1111



Format 2

ADDQ	-000	ACB	-100
CMPQ	-001	MOVQ	-101
SPR	-010	LPR	-110
Scond	-011		

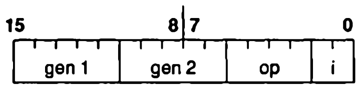
Appendix A: Instruction Formats (Continued)



Format 3

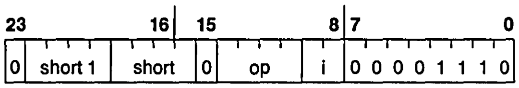
CXPD	-0000	ADJSP	-1010
BICPSR	-0010	JSR	-1100
JUMP	-0100	CASE	-1110
BISPSR	-0110		

Trap (UND) on XXX1, 1000



Format 4

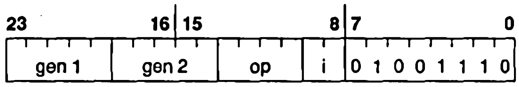
ADD	-0000	SUB	-1000
CMP	-0001	ADDR	-1001
BIC	-0010	AND	-1010
ADDC	-0100	SUBC	-1100
MOV	-0101	TBIT	-1101
OR	-0110	XOR	-1110



Format 5

MOVS	-0000	SETCFG*	-0010
CMPS	-0001	SKPS	-0011

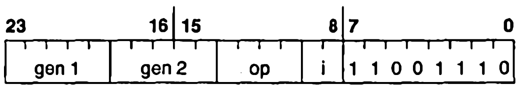
Trap (UND) on 1XXX, 01XX



Format 6

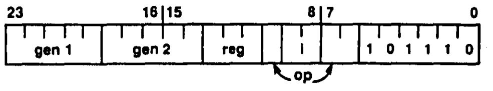
ROT	-0000	NEG	-1000
ASH	-0001	NOT	-1001
CBIT	-0010	Trap (UND)	-1010
CBITI	-0011	SUBP	-1011
Trap (UND)	-0100	ABS	-1100
LSH	-0101	COM	-1101
SBIT	-0110	IBIT	-1110
SBITI	-0111	ADDP	-1111

*Short 1 in format 5 applies only for SETCFG instruction. In other instructions this field is 0.



Format 7

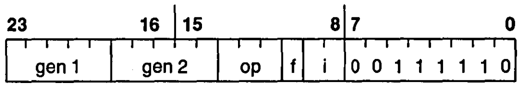
MOVM	-0000	MUL	-1000
CMPM	-0001	MEI	-1001
INSS	-0010	Trap (UND)	-1010
EXTS	-0011	DEI	-1011
MOVXBW	-0100	QUO	-1100
MOVZBW	-0101	REM	-1101
MOVZiD	-0110	MOD	-1110
MOVXiD	-0111	DIV	-1111



TL/EE/8673-78

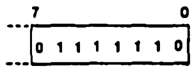
Format 8

EXT	-0 00	INDEX	-1 00
CVTP	-0 01	FFS	-1 01
INS	-0 10		
CHECK	-0 11		
MOVSU	-110, reg = 001		
MOVUS	-110, reg = 011		



Format 9

MOVif	-000	ROUND	-100
LFSR	-001	TRUNC	-101
MOVLF	-010	SFSR	-110
MOVFL	-011	FLOOR	-111

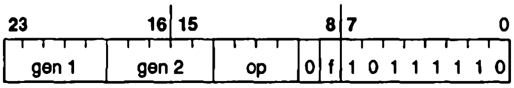


TL/EE/8673-79

Format 10

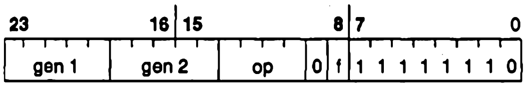
Trap (UND) Always

Appendix A: Instruction Formats (Continued)



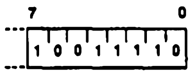
Format 11

ADDf	-0000	DIVf	-1000
MOVf	-0001	Note 1	-1001
CMPf	-0010	Trap (UND)	-1010
Note 3	-0011	Trap (UND)	-1011
SUBf	-0100	MULf	-1100
NEGf	-0101	ABSf	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111



Format 12

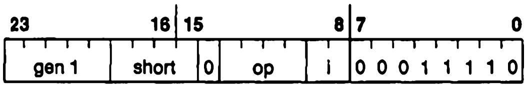
Note 2	-0000	Note 2	-1000
Note 1	-0001	Note 1	-1001
POLYf	-0010	Trap (UND)	-1010
DOTf	-0011	Trap (UND)	-1011
SCALBf	-0100	Note 2	-1100
LOGBf	-0101	Note 1	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111



TL/EE/8673-81

Format 13

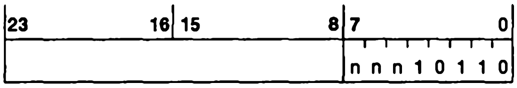
Trap (UND) Always



Format 14

RDVAL	-0000	LMR	-0010
WRVAL	-0001	SMR	-0011

Trap (UND) on 01XX, 1XXX



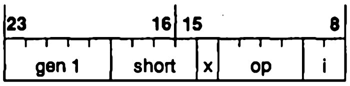
Operation Word ID Byte

Format 15

(Custom Slave)

nnn **Operation Word Format**

000

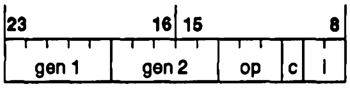


Format 15.0

CATST0	-0000	LCR	-0010
CATST1	-0001	SCR	-0011

Trap (UND) on all others

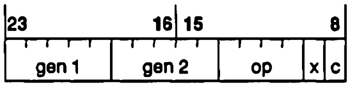
001



Format 15.1

CCV3	-000	CCV2	-100
LCSR	-001	CCV1	-101
CCV5	-010	SCSR	-110
CCV4	-011	CCV0	-111

101

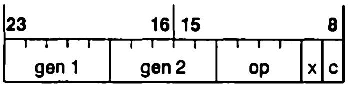


Format 15.5

CCAL0	-0000	CCAL3	-1000
CMOV0	-0001	CMOV3	-1001
CCMP0	-0010	Trap (UND)	-1010
CCMP1	-0011	Trap (UND)	-1011
CCAL1	-0100	CCAL2	-1100
CMOV2	-0101	CMOV1	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111

Appendix A: Instruction Formats (Continued)

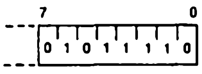
111



Format 15.7

Note 2	-0000	Note 2	-1000
Note 1	-0001	Note 1	-1001
Note 3	-0010	Trap (UND)	-1010
Note 3	-0011	Trap (UND)	-1011
Note 2	-0100	Note 2	-1100
Note 1	-0101	Note 1	-1101
Trap (UND)	-0110	Trap (UND)	-1110
Trap (UND)	-0111	Trap (UND)	-1111

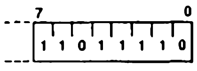
If nnn = 010, 011, 100, 110 then Trap (UND) Always.



TL/EE/8673-82

Format 16

Trap (UND) Always



TL/EE/8673-83

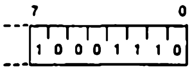
Note 1: Opcode not defined; CPU treats like MOV_f or $CMOV_c$. First operand has access class of read; second operand has access class of write; f or c field selects 32- or 64-bit data.

Note 2: Opcode not defined; CPU treats like ADD_f or $CCAL_c$. First operand has access class of read; second operand has access class of read-modify-write; f or c field selects 32- or 64-bit data.

Note 3: Opcode not defined; CPU treats like CMP_f or $CCMP_c$. First operand has access class of read; second operand has access class of read; f or c field selects 32- or 64-bit data.

Format 17

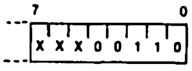
Trap (UND) Always



TL/EE/8673-84

Format 18

Trap (UND) Always

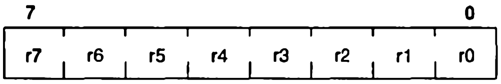


TL/EE/8673-85

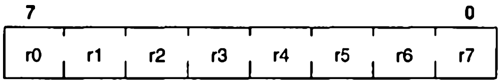
Format 19

Trap (UND) Always

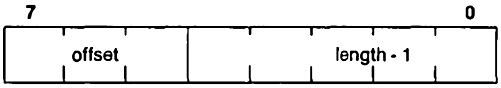
Implied Immediate Encodings:



Register Mark, appended to SAVE, ENTER



Register Mark, appended to RESTORE, EXIT



Offset/Length Modifier appended to INSS, EXTS

Appendix B: Interfacing Suggestions

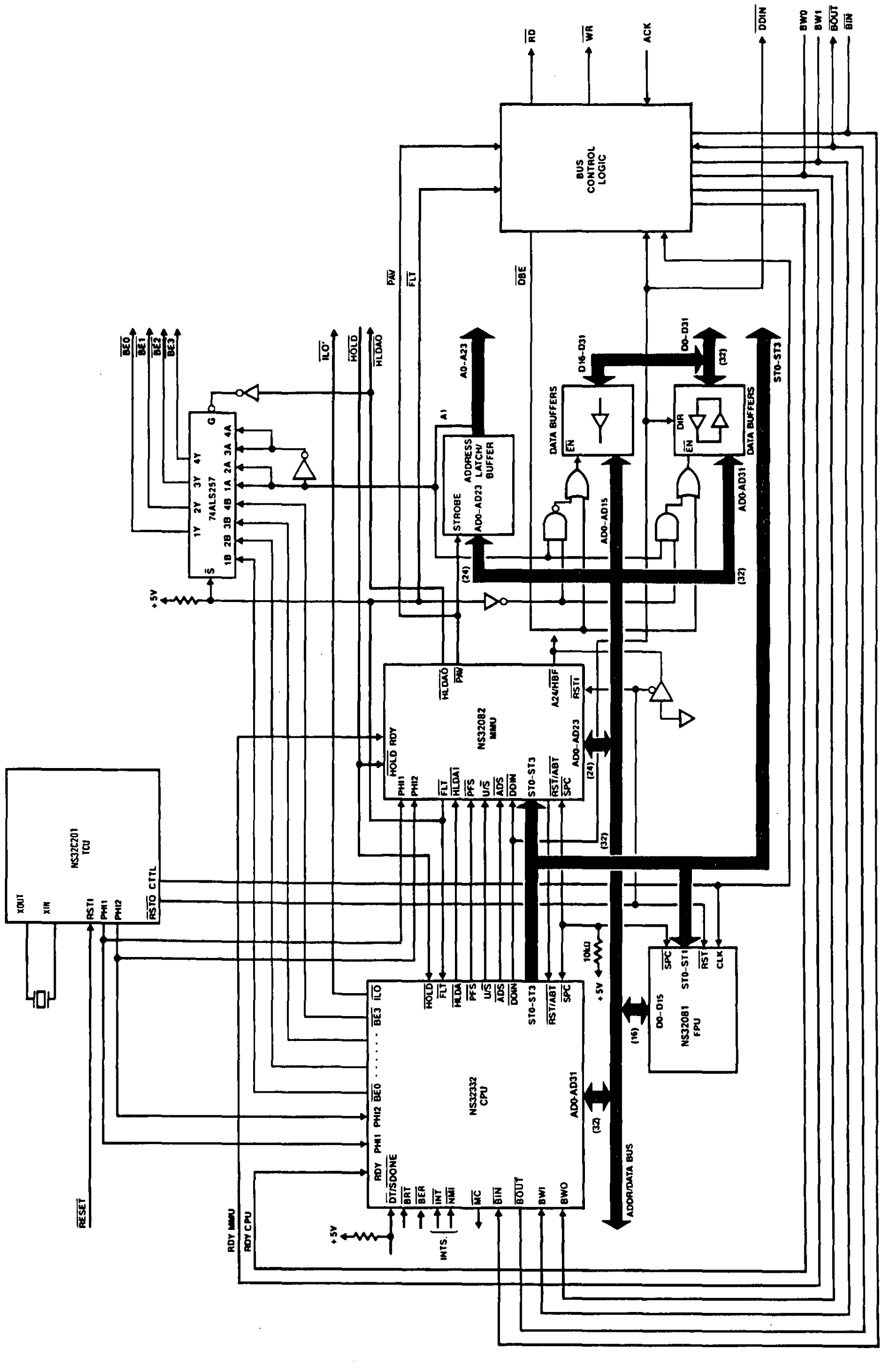
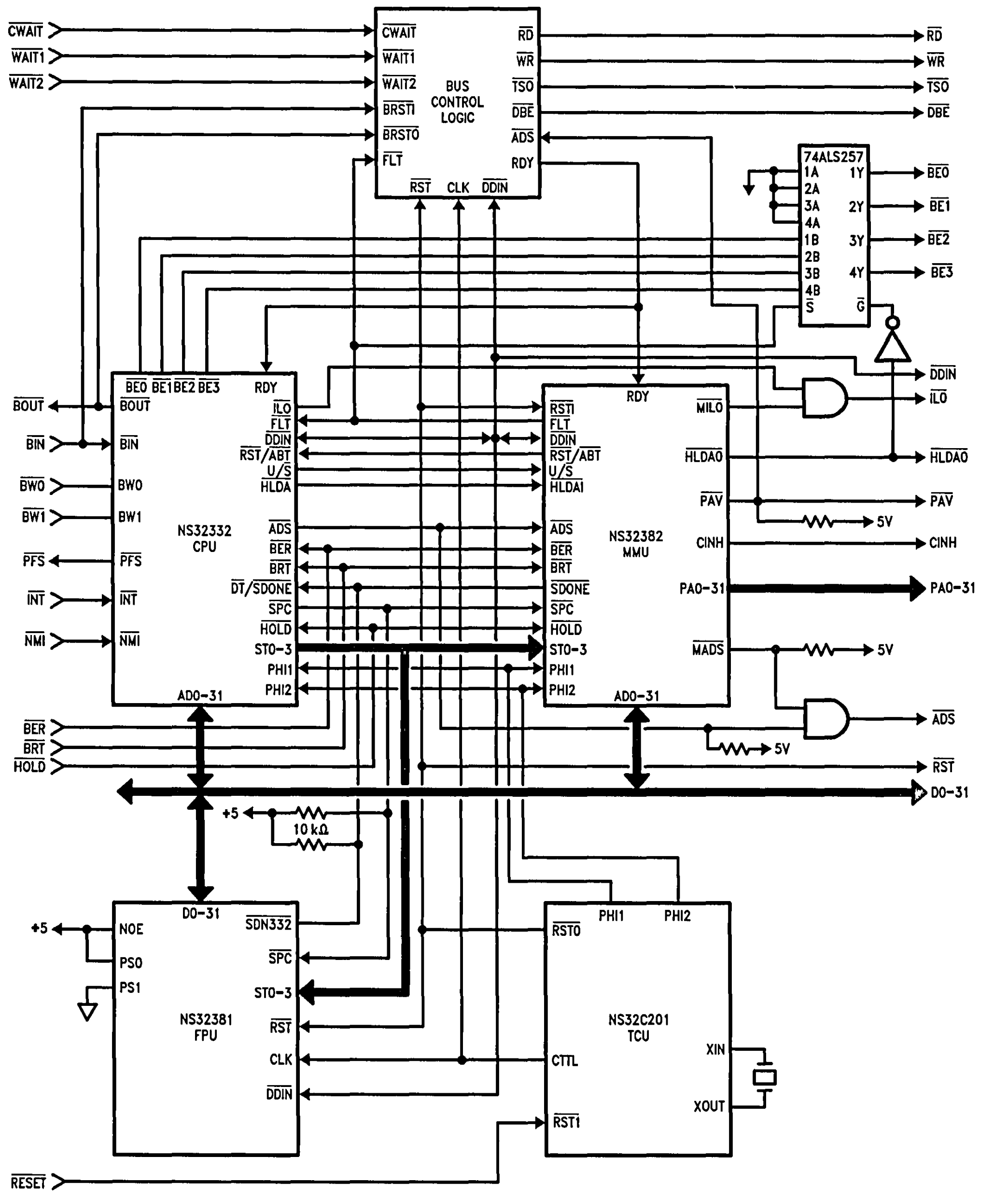


FIGURE B-1. System Connection Diagram (32332, 32081 & 32082)

Appendix B: Interfacing Suggestions (Continued)



TL/EE/8673-93

FIGURE B-2. System Connection Diagram (32332, 32381 & 32382)