National Semiconductor

PRELIMINARY
July 1989

NS32C016-10/NS32C016-15 High-Performance Microprocessors

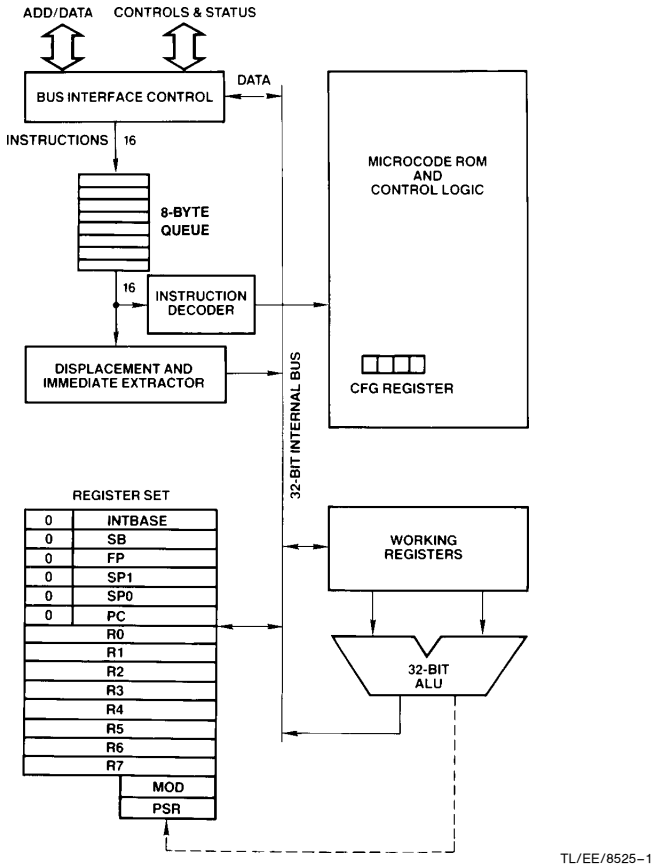
General Description

The NS32C016 is a 32-bit, CMOS microprocessor with TTL compatible inputs. The NS32C016 has a 16M byte linear address space and a 16-bit external data bus. It is fabricated with National Semiconductor's advanced CMOS process and is fully object code compatible with other Series 32000® CPU's. The NS32C016 has a 32-bit ALU, eight 32-bit general purpose registers, an eight-byte prefetch queue and a highly symmetric architecture. It also incorporates a slave processor interface and provides for full virtual memory capability in conjunction with the NS32082 memory management unit (MMU). High performance floating-point instructions are provided with the NS32081 floating-point unit (FPU). The NS32C016 is intended for a wide range of high performance computer applications.

Features

- 32-bit architecture and implementation
- 16M byte uniform addressing space
- Powerful instruction set
  - — General 2-address capability
  - — Very high degree of symmetry
  - — Addressing modes optimized for high-level Language references
- High-speed CMOS technology
- TTL compatible inputs
- Single 5V supply
- 48-pin dual-in-line package

Block Diagram



TL/EE/8525–1

STARPLEX II™ is a trademark of National Semiconductor Corp.
Series 32000® and TRI-STATE® are registered trademarks of National Semiconductor Corp.

 TL/EE/8525

RRD-B30M105/Printed in U. S. A.

Table of Contents

List of Illustrations

List of Illustrations (Continued)

List of Illustrations (Continued)

List of Tables

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 NS32C016 has 24-bit address pointers that can address up to 16 megabytes without 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

2.0 Architectural Description

2.1 PROGRAMMING MODEL

The Series 32000 architecture includes 16 registers on the NS32C016 CPU.

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 NS32C016 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. (In the NS32C016 the upper eight bits of this register are always zero.)

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 primarily for storing temporary data, and holding return information for operating system subroutines and interrupt and

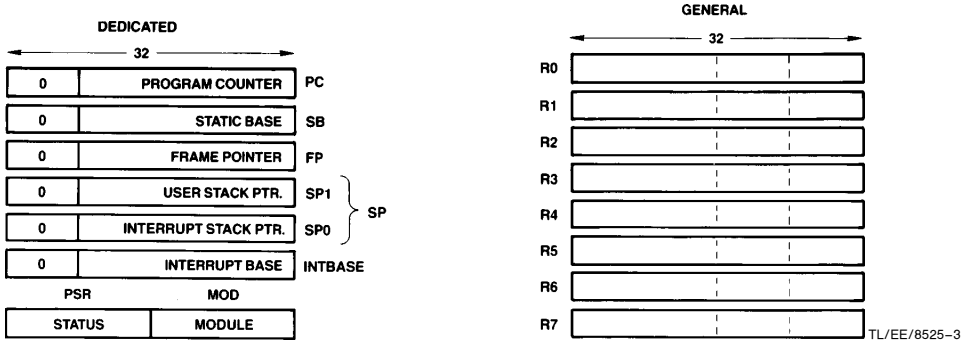


FIGURE 2-1. The General and Dedicated Registers

2.0 Architectural Description (Continued)

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 then SP refers to SP0. If the S bit in the PSR is 1 then SP refers to SP1. (In the NS32C016 the upper eight bits of these registers are always zero.)

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. (In the NS32C016 the upper eight bits of this register are always zero.)

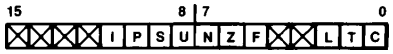
SB: The STATIC BASE register points to the global variables of a software module. This register is used to support relocatable global variables for software modules. The SB register holds the lowest address in memory occupied by the global variables of a module. (In the NS32C016 the upper eight bits of this register are always zero.)

INTBASE: The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Section 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS32C016 the upper eight bits of this register are always zero.)

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 NS32C016 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/8525–78

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 (Section 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 NS32C016 is said to be in Supervisor Mode; when U=1 the NS32C016 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).

P: The P bit prevents a TRC trap from occurring more than once for an instruction (Section 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 (Section 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 NS32C016 CPU is the four-bit CFG Register, which declares the presence of certain 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.*

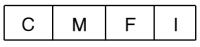


FIGURE 2-3. CFG Register

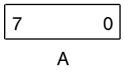
The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS32202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the $\overline{INT}$ pin are ''Vectored.'' If it is clear, these interrupts are ''Non-Vectored.'' See Section 3.8.

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.

2.1.4 Memory Organization

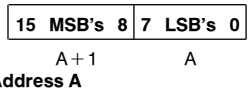
The main memory of the NS32C016 is a uniform linear address space. Memory locations are numbered sequentially starting at zero and ending at 2^{24} — 1. The number specifying a memory location is called an address. The contents of each memory location is a byte consisting of eight bits. Unless otherwise 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.



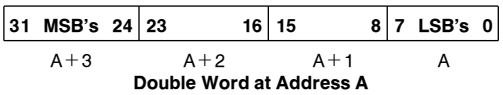
Byte at Address A

Two contiguous bytes are called a word. Except where noted (Section 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.



Word at Address A

Two contiguous words are called a double word. Except where noted (Section 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.



Double Word at Address A

Although memory is addressed as bytes, it is actually organized as words. Therefore, words and double words that are aligned to start at even addresses (multiples of two) are accessed more quickly than words and double words that are not so aligned.

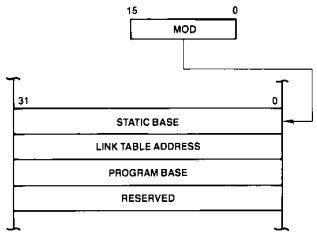
2.1.5 Dedicated Tables

Two of the NS32C016 dedicated registers (MOD and INTBASE) serve as pointers to dedicated tables in memory.

The INTBASE register points to the Interrupt Dispatch and Cascade tables. These are described in Section 3.8.

The MOD register contains a pointer into the Module Table, whose entries are called Module Descriptors. A Module Descriptor contains four pointers, three of which are used by the NS32C016. The MOD register contains the address of the Module Descriptor for the currently running module. It is automatically updated by the Call External Procedure instructions (CXP and CXPD).

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.



TL/EE/8525–4

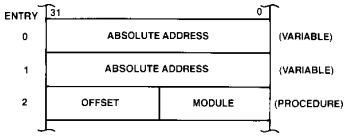
FIGURE 2-4. Module Descriptor Format

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/8525–5

FIGURE 2-5. A Sample Link Table

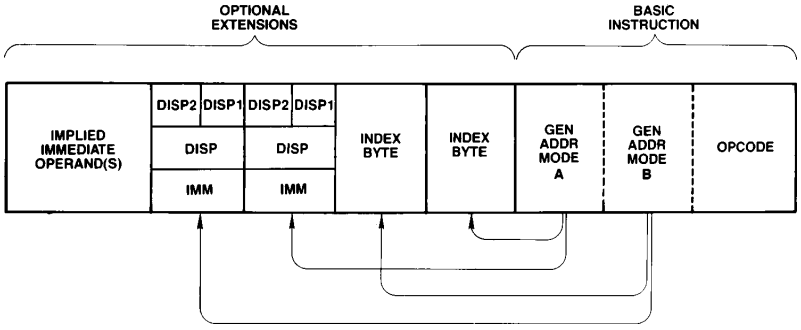
2.2 INSTRUCTION SET

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 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.

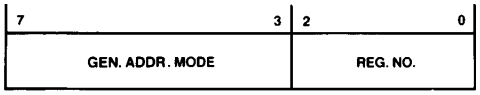
Index Bytes appear when either or both Gen fields specify Scaled Index. In this case, the Gen field specifies only the Scale Factor (1, 2, 4 or 8), and the Index Byte specifies which General Purpose Register to use as the index, and which addressing mode calculation to perform before indexing. See *Figure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

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain



TL/EE/8525–6

FIGURE 2-6. General Instruction Format



TL/EE/8525–7

FIGURE 2-7. Index Byte Format

one of two displacements, or one immediate value. The size of a Displacement field is encoded within 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 (Section 2.1.4).

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 (Section 2.2.3).

2.2.2 Addressing Modes

The NS32C016 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 in the NS32C016 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.

NS32C016 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.

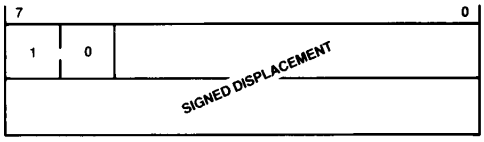
Register Relative: A General Purpose Register contains an address to which is added a displacement value from the instruction, yielding the Effective Address of the operand in memory.

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.

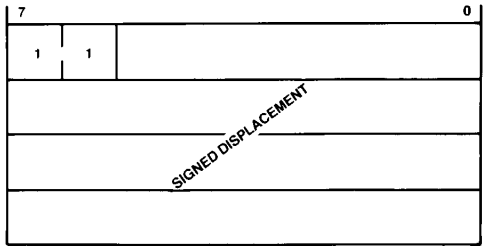
Memory Relative: A pointer variable is found within the memory space pointed to by the SP, SB or FP register. A

Byte Displacement: Range −64 to +63

Word Displacement: Range −8192 to +8191



Double Word Displacement: Range (Entire Addressing Space)



TL/EE/8525–8

FIGURE 2-8. Displacement Encodings

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.

2.0 Architectural Description (Continued)

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 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 Series 32000 Instruction Set Reference Manual.

TABLE 2-1. NS32C016 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	R6 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 Disp 1 + 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)

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS32C016 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 Series 32000 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).

TABLE 2-2. NS32C016 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.

TABLE 2-2. NS32C016 Instruction Set Summary (Continued)

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.

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.

2.0 Architectural Description (Continued)

TABLE 2-2. NS32C016 Instruction Set Summary (Continued)

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 strong 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	Description
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.

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)

2.0 Architectural Description (Continued)

TABLE 2-2. NS32C016 Instruction Set Summary (Continued)

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.
9	LFSR	gen	Load FSR.
9	SFSR	gen	Store FSR.

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.

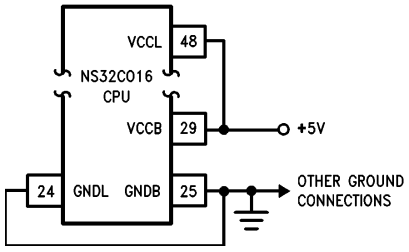
CUSTOM SLAVE

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

3.1 POWER AND GROUNDING

Power and ground connections for the NS32C016 are made on four pins. On-chip logic is connected to power through the logic power pin (VCCL, pin 48) and to ground through the logic ground pin (GNDL, pin 24). On-chip output drivers are connected to power through the buffer power pin (VCCP, pin 29) and to ground through the buffer ground pin (GNDB, pin 25). For optimal noise immunity, it is recommended that single conductors be connected directly from VCCL to VCCB and from GNDL to GNDB, as shown below (*Figure 3-1*).



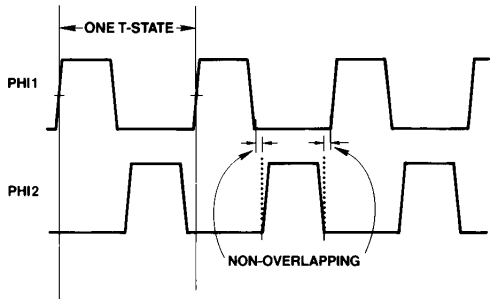
TL/EE/8525–9

FIGURE 3-1. Recommended Supply Connections

3.2 CLOCKING

The NS32C016 inputs clocking signals from the NS32C201 Timing Control Unit (TCU), which presents two non-overlapping phases of a single clock frequency. These phases are called PHI1 (pin 26) and PHI2 (pin 27). 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 Section 4 for complete specifications of PHI1 and PHI2.



TL/EE/8525–10

FIGURE 3-2. Clock Timing Relationships

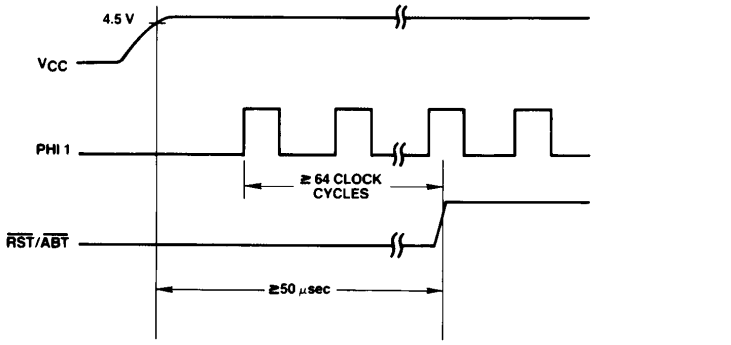
As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected anywhere except from the TCU to the CPU and, if present, the MMU. A TTL Clock signal (CTTL) is provided by the TCU for all 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 Section 3.5.4.

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 μs after V_{CC} is stable. This is to ensure that all on-chip voltages are completely stable before operation. Whenever a Reset is applied, it must also remain active



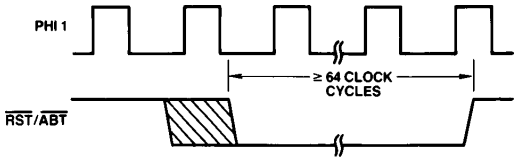
TL/EE/8525–11

FIGURE 3-3. Power-On Reset Requirements

3.0 Functional Description (Continued)

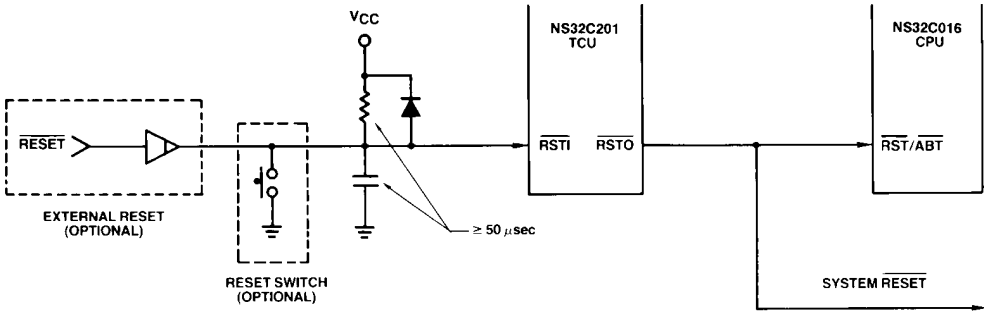
for not less than 64 clock cycles. The rising edge must occur while PHI1 is high. See *Figures 3-3* and *3-4.*

The NS32C201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32C016 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.



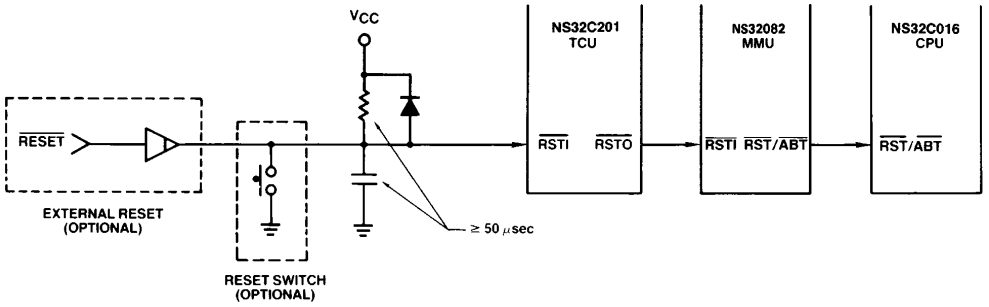
TL/EE/8525–12

FIGURE 3-4. General Reset Timing



TL/EE/8525–13

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



TL/EE/8525–14

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

3.4 BUS CYCLES

The NS32C016 CPU has a strap option which defines the Bus Timing Mode as either With or Without Address Translation. This section describes only bus cycles under the No Address Translation option. For details of the use of the strap and of bus cycles with address translation, see Section 3.5.

The CPU will perform a bus cycle for one of the following reasons:

1) To write or read data, to or from memory or a peripheral interface device. Peripheral input and output are memory-mapped in the Series 32000 family.
2) To fetch instructions into the eight-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 Section 4. The only external difference between them is the four-bit code placed on the Bus Status pins (ST0–ST3). Slave Processor cycles differ in that separate control signals are applied (Section 3.4.6).

The sequence of events in a non-Slave bus cycle is shown in *Figure 3-7* for a Read cycle and *Figure 3-8* 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 (Section 3.4.1).

3.0 Functional Description (Continued)

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

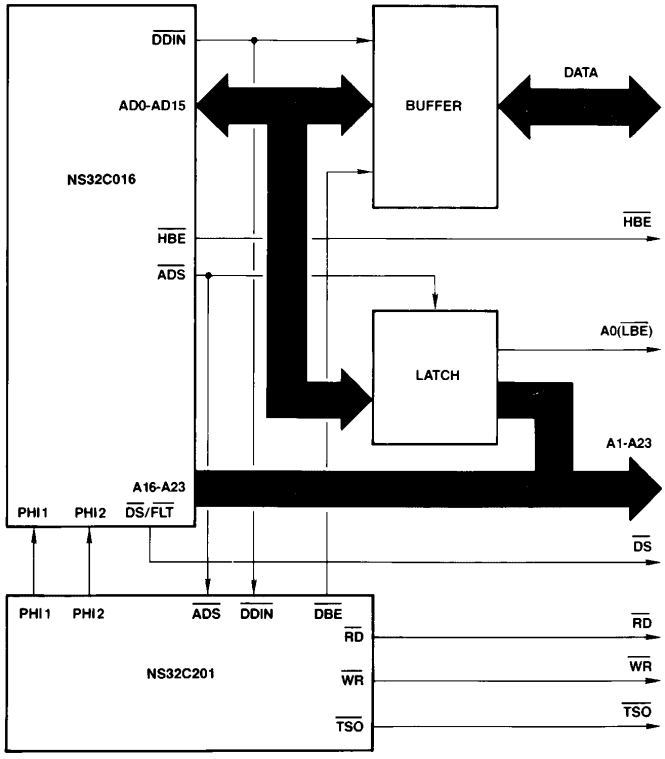
During T1, the CPU applies an address on pins AD0–AD15 and A16–A23. It also provides a low-going pulse on the $\overline{ADS}$ pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing Address bits 0–15 from the AD0–AD15 pins. See *Figure 3-6*. During this time also the status signals $\overline{DDIN}$, indicating the direction of the transfer, and $\overline{HBE}$, indicating whether the high byte (AD8–AD15) is to be referenced, become valid.

During T2 the CPU switches the Data Bus, AD0–AD15, to either accept or present data. Note that the signals A16–A23 remain valid, and need not be latched. It also starts the data strobe ($\overline{DS}$), signaling the beginning of the data transfer. Associated signals from the NS32C201 Timing Control Unit are also activated at this time: $\overline{RD}$ (Read Strobe) or $\overline{WR}$ (Write Strobe), $\overline{TSO}$ (Timing State Output, indicating that T2 has been reached) and $\overline{DBE}$ (Data Buffer Enable).

The T3 state provides for access time requirements, and it occurs at least once in a bus cycle. At the end of T2, on the falling edge of the PHI2 clock, the RDY (Ready) line is sampled to determine whether the bus cycle will be extended (Section 3.4.1).

If the CPU is performing a Read cycle, the Data Bus (AD0–AD15) is sampled at the falling edge of PHI2 of the last T3 state, see Section 4. Data must, however, be held at least until the beginning of T4. $\overline{DS}$ and $\overline{RD}$ are guaranteed not to go inactive before this point, so the rising edge of either of them may safely be used to disable the device providing the input data.

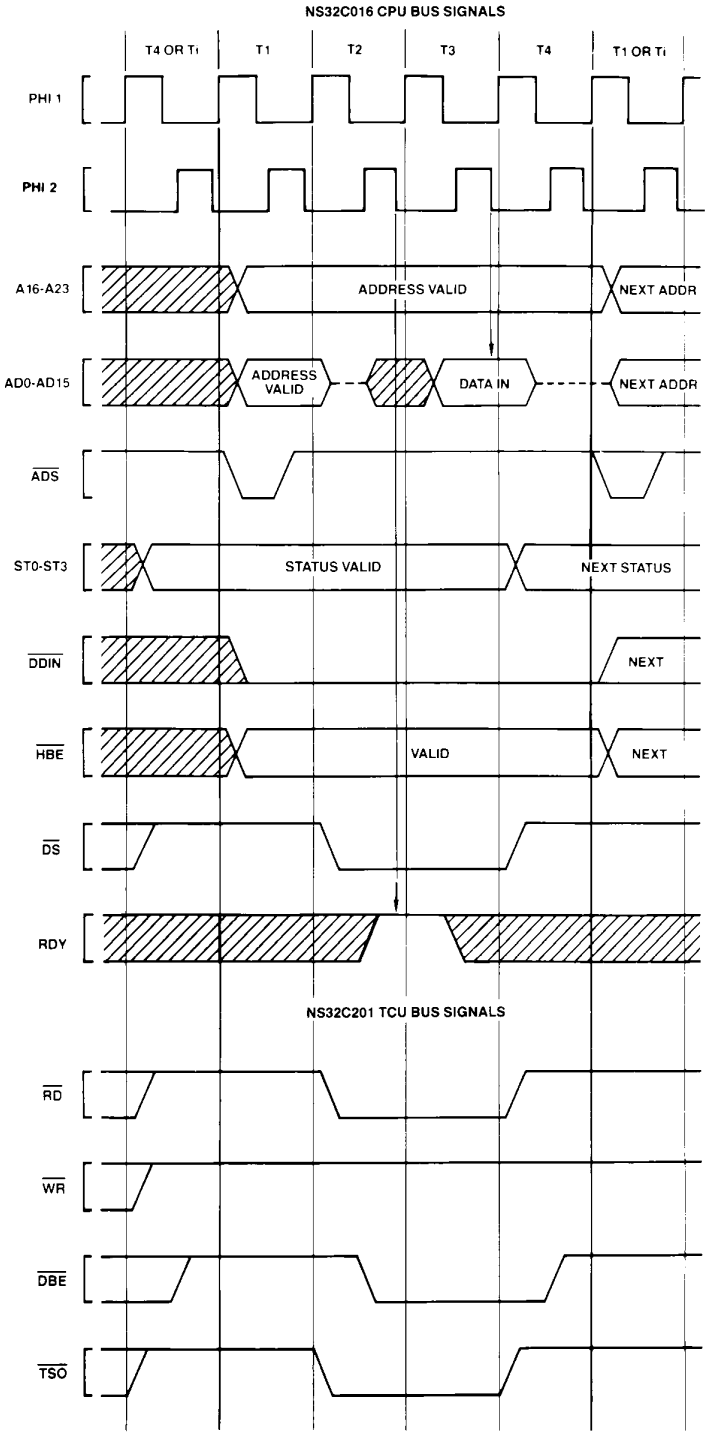
The T4 state finishes the bus cycle. At the beginning of T4, the $\overline{DS}$, $\overline{RD}$, or $\overline{WR}$, and $\overline{TSO}$ signals go inactive, and at the rising edge of PHI2, $\overline{DBE}$ goes inactive, having provided for necessary data hold times. Data during Write cycles remains valid from the CPU throughout T4. Note that the Bus Status lines (ST0–ST3) change at the beginning of T4, anticipating the following bus cycle (if any).



TL/EE/8525–15

FIGURE 3-6. Bus Connections

3.0 Functional Description (Continued)



TL/EE/8525–16

FIGURE 3-7. Read Cycle Timing

3.0 Functional Description (Continued)

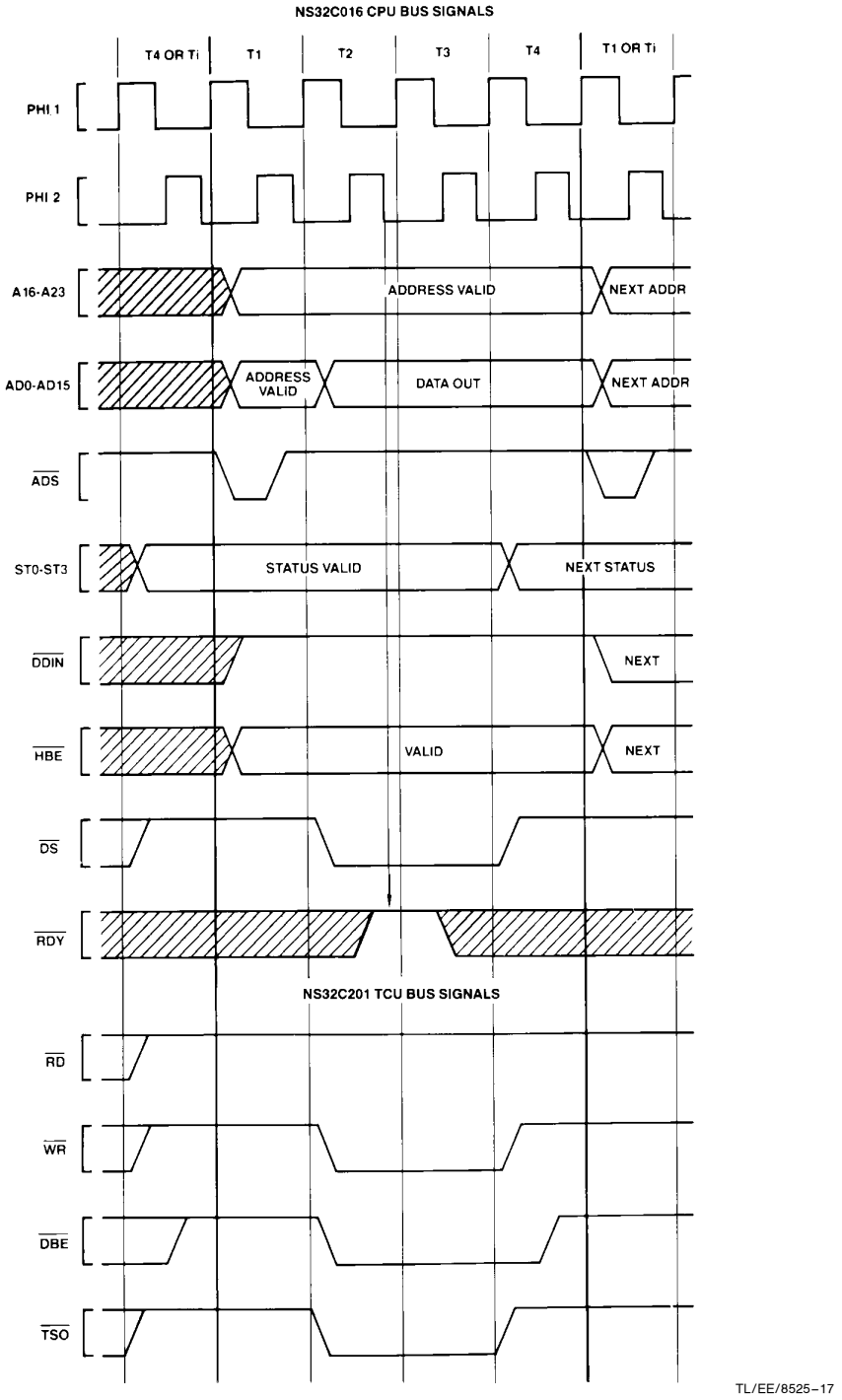


FIGURE 3-8. Write Cycle Timing

3.0 Functional Description (Continued)

3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS32C016 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.

At the end of T2 on the falling edge of PHI2, the RDY line is sampled by the CPU. If RDY is high, the next T-states will be T3 and then T4, ending the bus cycle. If it is sampled low, then another T3 state will be inserted after the next T-state and the RDY line will again be sampled on the falling edge of PHI2. Each additional T3 state after the first is referred to as a ''wait state.'' See *Figure 3-9.*

The RDY pin is driven by the NS32C201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pins:

1) $\overline{\text{CWAIT}}$ (Continues WAIT), which holds the CPU in WAIT states until removed.
2) $\overline{\text{WAIT1}}$, $\overline{\text{WAIT2}}$, $\overline{\text{WAIT4}}$, $\overline{\text{WAIT8}}$ (Collectively $\overline{\text{WAIT}}$n), which may be given a four-bit binary value requesting a specific number of WAIT States from 0 to 15.
3) $\overline{\text{PER}}$ (Peripheral), which inserts five additional WAIT states and causes the TCU to reshape the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ strobes. This provides the setup and hold times required by most MOS peripheral interface devices.

Combinations of these various WAIT requests are both legal and useful. For details of their use, see the NS32C201 TCU Data Sheet.

Figure 3-10 illustrates a typical Read cycle, with two WAIT states requested through the TCU $\overline{\text{WAIT}}$n pins.

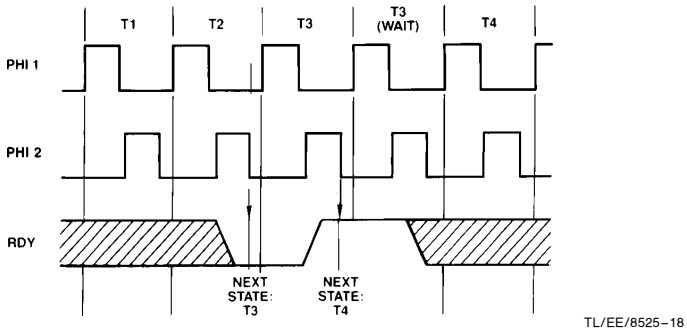


FIGURE 3-9. RDY Pin Timing

3.4.2 Bus Status

The NS32C016 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 it is idle.

Referring to *Figures 3-7* and *3-8,* 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 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 FFFF00_{16}, but will ignore any data provided.

To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$) it will read from address FFFE00_{16}, expecting a vector number to be provided from the Master NS32202 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, having assumed that no NS32202 is present. See Section 3.4.5.

0101 — Interrupt Acknowledge, Cascaded.

The CPU is reading a vector number from a Cascaded NS32202 Interrupt Control Unit. The address provided is the address of the NS32202 Hardware Vector register. See Section 3.4.5.

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 Section 3.4.5.

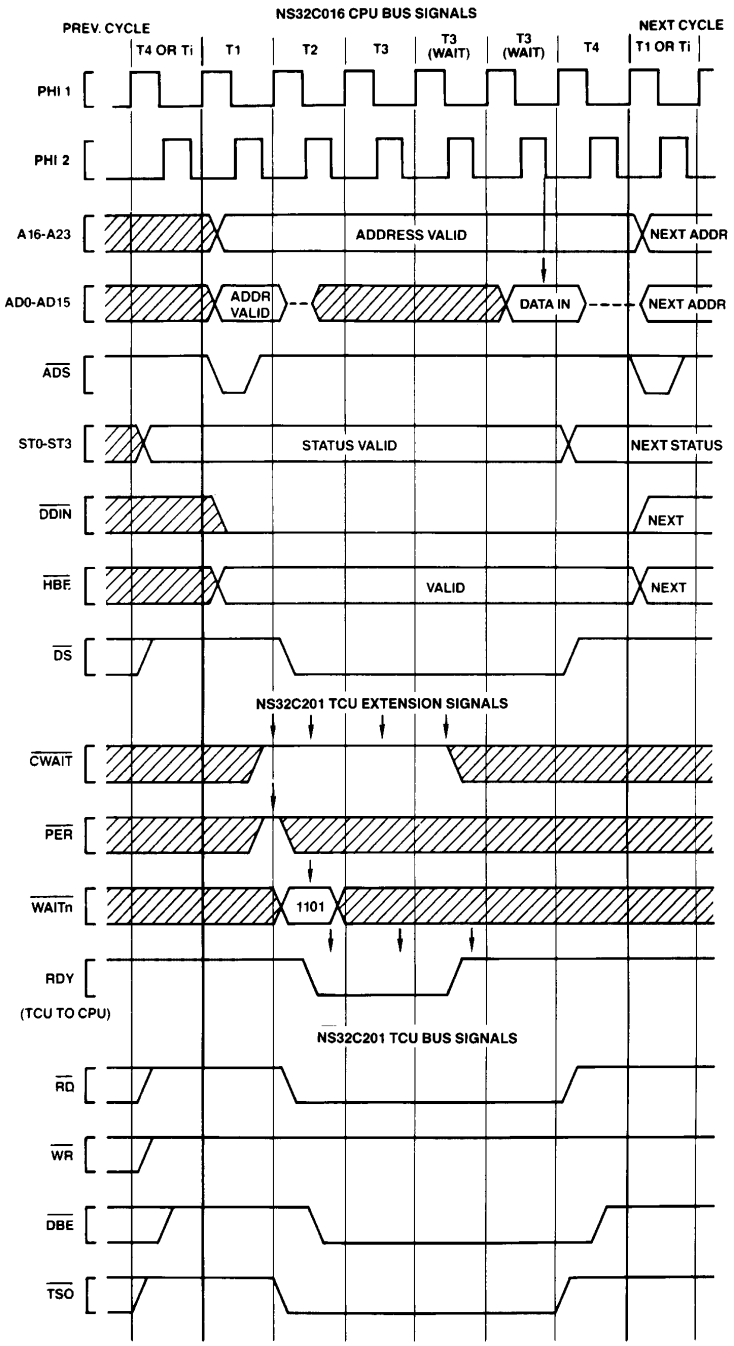
0111 — End of Interrupt, Cascaded.

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 Section 3.4.5.

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.

3.0 Functional Description (Continued)



TL/EE/8525–19

FIGURE 3-10. Extended Cycle Example

Note: Arrows on $\overline{\text{CWAIT}}$, $\overline{\text{PER}}$, $\overline{\text{WAITn}}$ indicate points at which the TCU samples. Arrows on AD0–AD15 and RDY indicate points at which the CPU samples.

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 Section 3.9.1.

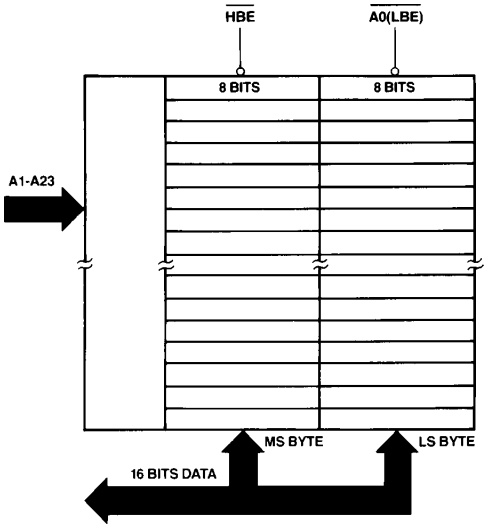
1110 — Read Slave Processor Status.
The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU Processor Status Register bits N, Z, L or F. See Section 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 Section 3.9.1.

3.4.3 Data Access Sequences

The 24-bit address provided by the NS32C016 is a byte address; that is, it uniquely identifies one of up to 16,777,216 eight-bit memory locations. An important feature of the NS32C016 is that the presence of a 16-bit data bus imposes no restrictions on data alignment; any data item, regardless of size, may be placed starting at any memory address. The NS32C016 provides a special control signal, High Byte Enable ($\overline{HBE}$), which facilitates individual byte addressing on a 16-bit bus.

Memory is organized as two eight-bit banks, each bank receiving the word address (A1–A23) in parallel. One bank, connected to Data Bus pins AD0–AD7, is enabled to respond to even byte addresses; i.e., when the least significant address bit (A0) is low. The other bank, connected to Data Bus pins AD8–AD15, is enabled when $\overline{HBE}$ is low. See *Figure 3-11.*



TL/EE/8525–20

FIGURE 3-11. Memory Interface

Any bus cycle falls into one of three categories: Even Byte Access, Odd Byte Access, and Even Word Access. All accesses to any data type are made up of sequences of these cycles. Table 3-1 gives the state of A0 and $\overline{HBE}$ for each category.

TABLE 3-1. Bus Cycle Categories

Category	$\overline{HBE}$	A0
Even Byte	1	0
Odd Byte	0	1
Even Word	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 (i.e., whether it starts on an even byte address or an odd byte address). Table 3-2 lists the bus cycle performed for each situation. For the timing of A0 and $\overline{HBE}$, see Section 3.4.

3.0 Functional Description (Continued)

TABLE 3-2. Access Sequences

Cycle	Type	Address	$\overline{HBE}$	A0	High Bus	Low Bus
A. Odd Word Access Sequence						
					BYTE 1	**BYTE 0** ← **A**
1	Odd Byte	A	0	1	Byte 0	Don't Care
2	Even Byte	A + 1	1	0	Don't Care	Byte 1
B. Even Double-Word Access Sequence						
			BYTE 3	**BYTE 2**	**BYTE 1**	**BYTE 0** ← **A**
1	Even Word	A	0	0	Byte 1	Byte 0
2	Even Word	A + 2	0	0	Byte 3	Byte 2
C. Odd Double-Word Access Sequence						
			BYTE 3	**BYTE 2**	**BYTE 1**	**BYTE 0** ← **A**
1	Odd Byte	A	0	1	Byte 0	Don't Care
2	Even Word	A + 1	0	0	Byte 2	Byte 1
3	Even Byte	A + 3	1	0	Don't Care	Byte 3
D. Even Quad-Word Access Sequence						
BYTE 7	**BYTE 6**	**BYTE 5**	**BYTE 4**	**BYTE 3**	**BYTE 2**	**BYTE 1**	**BYTE 0** ← **A**
1	Even Word	A		0	0	Byte 1	Byte 0
2	Even Word	A + 2		0	0	Byte 3	Byte 2
Other bus cycles (instruction prefetch or slave) can occur here.							
3	Even Word	A + 4		0	0	Byte 5	Byte 4
4	Even Word	A + 6		0	0	Byte 7	Byte 6
E. Odd Quad-Word Access Sequence						
BYTE 7	**BYTE 6**	**BYTE 5**	**BYTE 4**	**BYTE 3**	**BYTE 2**	**BYTE 1**	**BYTE 0** ← **A**
1	Odd Byte	A		0	1	Byte 0	Don't Care
2	Even Word	A + 1		0	0	Byte 2	Byte 1
3	Even Byte	A + 3		1	0	Don't Care	Byte 3
Other bus cycles (instruction prefetch or slave) can occur here.							
4	Odd Byte	A + 4		0	1	Byte 4	Don't Care
5	Even Word	A + 5		0	0	Byte 6	Byte 5
6	Even Byte	A + 7		1	0	Don't Care	Byte 7

3.4.3.1 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.

3.4.3.2 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.

3.4.3.3 Extending Multiply 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.

3.4.4 Instruction Fetches

Instructions for the NS32C016 CPU are ''prefetched''; that is, they are input before being needed into the next available entry of the eight-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 (Section 3.4.2).

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 Even Word 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 is either an Even Word Read or an Odd Byte Read, depending on whether the destination address is even or odd.

3.4.5 Interrupt Control Cycles

Activating the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ 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. For full details of the NS32C016 interrupt structure, see Section 3.8.

3.0 Functional Description (Continued)

TABLE 3-3. Interrupt Sequences

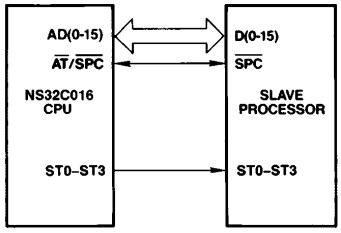
Cycle	Status	Address	$\overline{DDIN}$	$\overline{HBE}$	A0	High Bus	Low Bus
			A. Non-Maskable Interrupt Control Sequences.				
Interrupt Acknowledge							
1	0100	$FFFF00_{16}$	0	1	0	Don't Care	Don't Care
Interrupt Return							
None: Performed through Return from Trap (RETT) instruction.							
			B. Non-Vectored Interrupt Control Sequences.				
Interrupt Acknowledge							
1	0100	$FFFE00_{16}$	0	1	0	Don't Care	Don't Care
Interrupt Return							
None: Performed through Return from Trap (RETT) instruction.							
			C. Vectored Interrupt Sequences: Non-Cascaded.				
Interrupt Acknowledge							
1	0100	$FFFE00_{16}$	0	1	0	Don't Care	Vector: Range: 0−127
Interrupt Return							
1	0110	$FFFE00_{16}$	0	1	0	Don't Care	Vector: Same as in Previous Int. Ack. Cycle
			D. Vectored Interrupt Sequences: Cascaded.				
Interrupt Acknowledge							
1	0100	$FFFE00_{16}$	0	1	0	Don't Care	Cascade Index: range −16 to −1
(The CPU here uses the Cascade Index to find the Cascade Address.)							
2	0101	Cascade Address	0	1 or 0*	0 or 1*	Vector, range 0−255; on appropriate half of Data Bus for even/odd address	
Interrupt Return							
1	0110	$FFFE00_{16}$	0	1	0	Don't Care	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	1 or 0*	0 or 1*	Don't Care	Don't Care

* If the Cascaded ICU Address is Even (A0 is low), then the CPU applies $\overline{HBE}$ high and reads the vector number from bits 0−7 of the Data Bus.
If the address is Odd (A0 is high), then the CPU applies $\overline{HBE}$ low and reads the vector number from bits 8−15 of the Data Bus. The vector number may be in the range 0−255.

3.0 Functional Description (Continued)

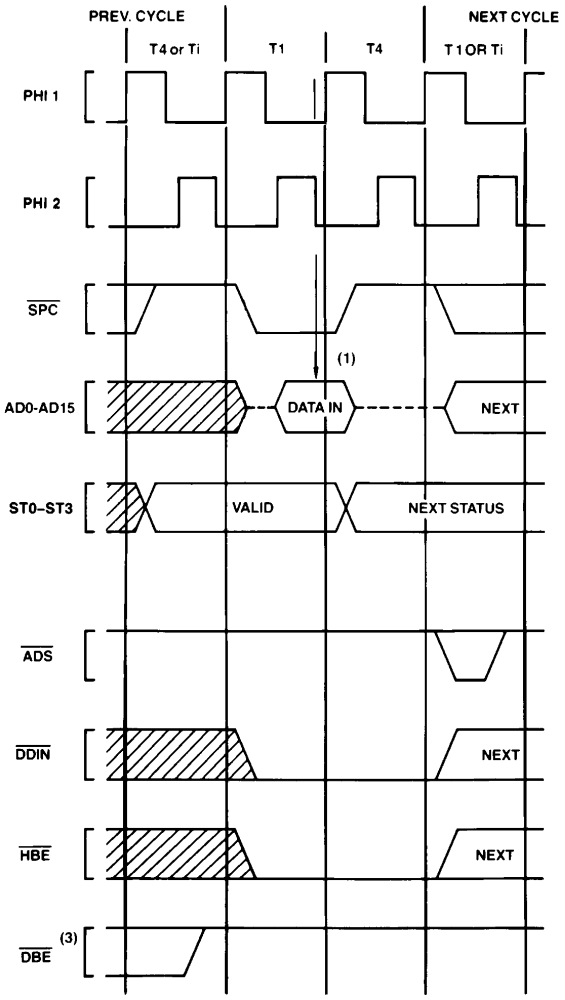
3.4.6 Slave Processor Communication

In addition to its use as the Address Translation strap (Section 3.5.1), the $\overline{AT}/\overline{SPC}$ 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 (AD0–AD15), 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 Section 3.9 for full protocol sequences.



TL/EE/8525–21

FIGURE 3-12. Slave Processor Connections



TL/EE/8525–22

Notes:

(1) CPU samples Data Bus here.

(2) $\overline{DBE}$ and all other NS32C201 TCU bus signals remain inactive because no $\overline{ADS}$ pulse is received from the CPU.

FIGURE 3-13. CPU Read from Slave Processor

3.0 Functional Description (Continued)

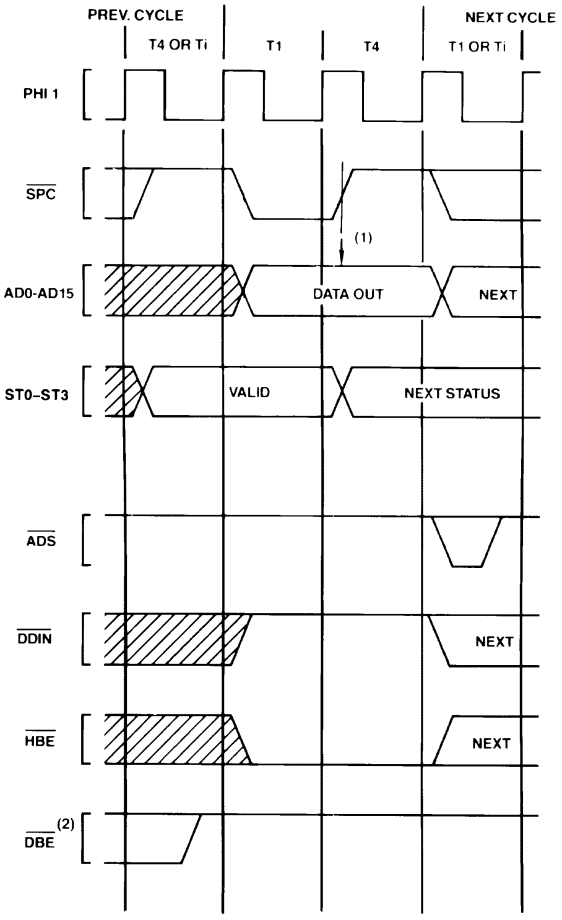
3.4.6.1 Slave Processor Bus Cycles

A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see *Figures 3-13* and *3-14*). 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.

Since the CPU does not pulse the Address Strobe ($\overline{ADS}$), no bus signals are generated by the NS32C201 Timing Control Unit. 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.

3.4.6.2 Slave Operand Transfer Sequences

A Slave Processor operand is transferred in one or more Slave bus cycles. A Byte operand is transferred on the least-significant byte of the Data Bus (AD0–AD7), and a Word operand is transferred on the entire bus. A Double Word is transferred in a consecutive pair of bus cycles, least-significant word first. A Quad Word is transferred in two pairs of Slave cycles, with other bus cycles possibly occurring between them. The word order is from least-significant word to most-significant.



TL/EE/8525–23

Notes:

(1) Slave Processor samples Data Bus here.

(2) $\overline{DBE}$, being provided by the NS32C201 TCU, remains inactive due to the fact that no pulse is presented on $\overline{ADS}$. $\overline{TCU}$ signals $\overline{RD}$, $\overline{WR}$ and $\overline{TSO}$ also remain inactive.

FIGURE 3-14. CPU Write to Slave Processor

3.0 Functional Description (Continued)

3.5 MEMORY MANAGEMENT OPTION

The NS32C016 CPU, in conjunction with the NS32082 Memory Management Unit (MMU), provides full support for address translation, memory protection, and memory allocation techniques up to and including Virtual Memory.

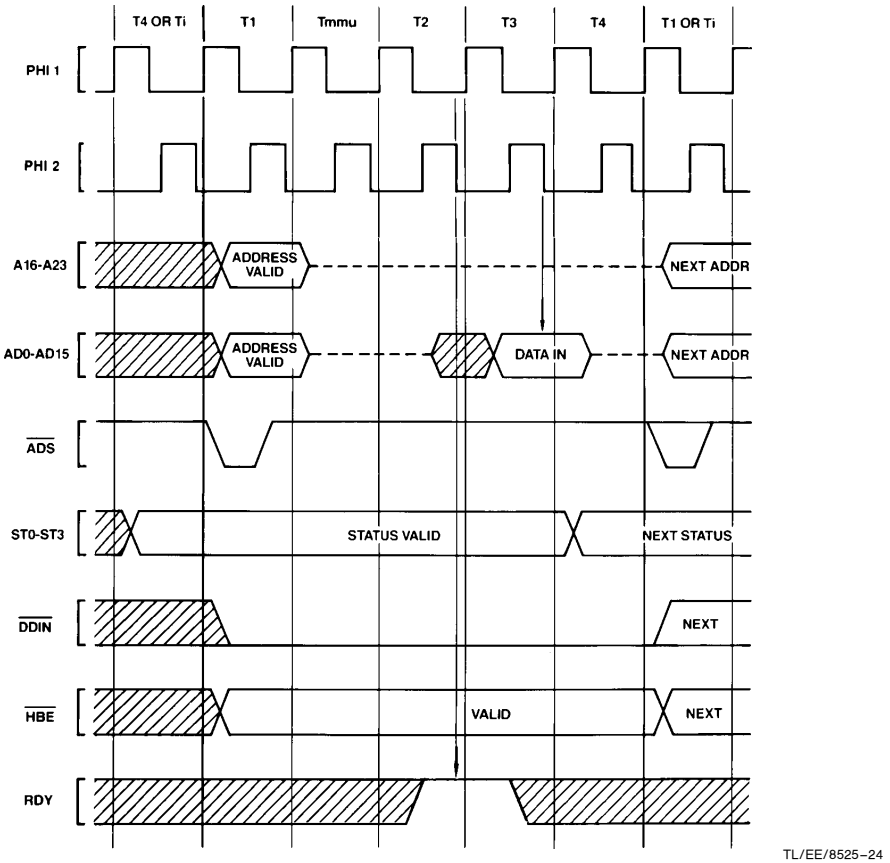
3.5.1 Address Translation Strap

The Bus Interface Control section of the NS32C016 CPU has two bus timing modes: With or Without Address Translation. The mode of operation is selected by the CPU by sampling the $\overline{AT}/\overline{SPC}$ (Address Translation/Slave Processor Control) pin on the rising edge of the $\overline{RST}$ (Reset) pulse. If $\overline{AT}/\overline{SPC}$ is sampled as high, the bus timing is as previously described in Section 3.4. If it is sampled as low, two changes occur:

1) An extra clock cycle, Tmmu, is inserted into all bus cycles except Slave Processor transfers.
2) The $\overline{DS}/\overline{FLT}$ pin changes in function from a Data Strobe output ($\overline{DS}$) to a Float Command input ($\overline{FLT}$).

The NS32082 MMU will itself pull the CPU $\overline{AT}/\overline{SPC}$ pin low when it is reset. In non-Memory-Managed systems this pin should be pulled up to V_{CC} through a 10 kΩ resistor.

Note that the Address Translation strap does not specifically declare the presence of an NS32082 MMU, but only the



TL/EE/8525–24

FIGURE 3-15. Read Cycle with Address Translation (CPU Action)

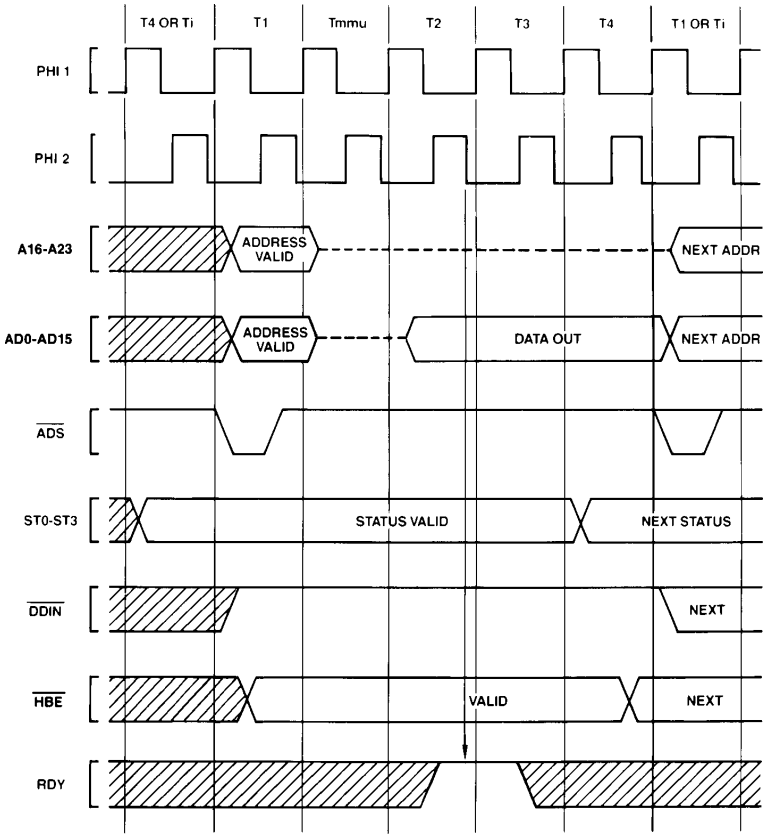
presence of external address translation circuitry. MMU instructions will still trap as being undefined unless the SETCFG (Set Configuration) instruction is executed to declare the MMU instruction set valid. See Section 2.1.3.

3.5.2 Translated Bus Timing

Figures 3-15 and *3-16* illustrate the CPU activity during a Read cycle and a Write cycle in Address Translation mode. The additional T-State, Tmmu, is inserted between T1 and T2. During this time the CPU places AD0–AD15 and A16–A23 into the TRI-STATE® mode, allowing the MMU to assert the translated address and issue the physical address strobe $\overline{PAV}$. T2 through T4 of the cycle are identical to their counter-parts without Address Translation, with the exception that the CPU Address lines A16–A23 remain in the TRI-STATE condition. This allows the MMU to continue asserting the translated address on those pins.

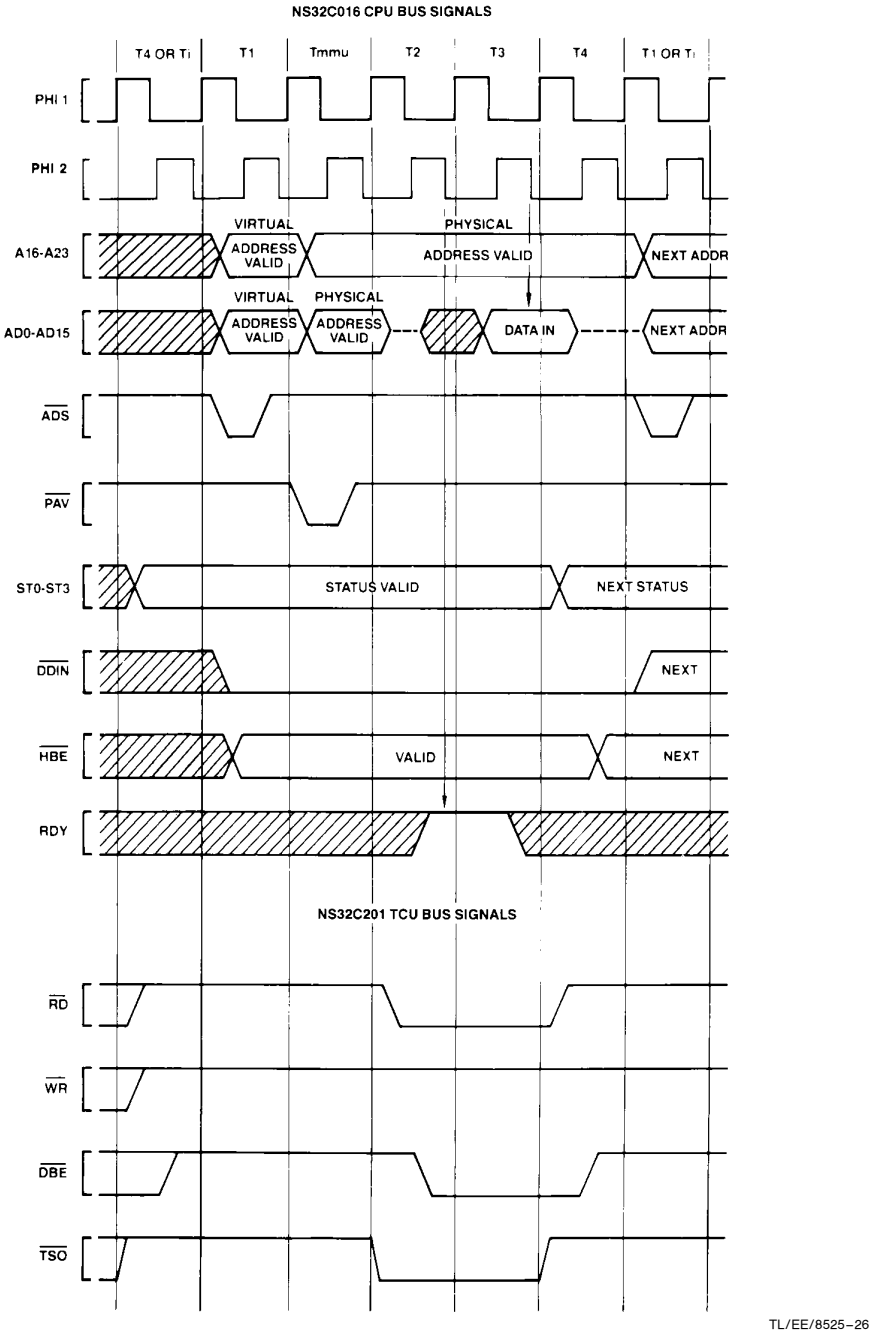
Note that in order for the NS32082 MMU to operate correctly, it must be set to the 32C016 mode by forcing A24 high during reset.

Figures 3-17 and *3-18* show a Read cycle and a Write cycle as generated by the 32C016/32082/32C201 group. Note that with the CPU $\overline{ADS}$ signal going only to the MMU, and with the MMU $\overline{PAV}$ signal substituting for $\overline{ADS}$ everywhere else, Tmmu through T4 look exactly like T1 through T4 in a non-Memory-Managed system. For the connection diagram, see Appendix B.



TL/EE/8525–25

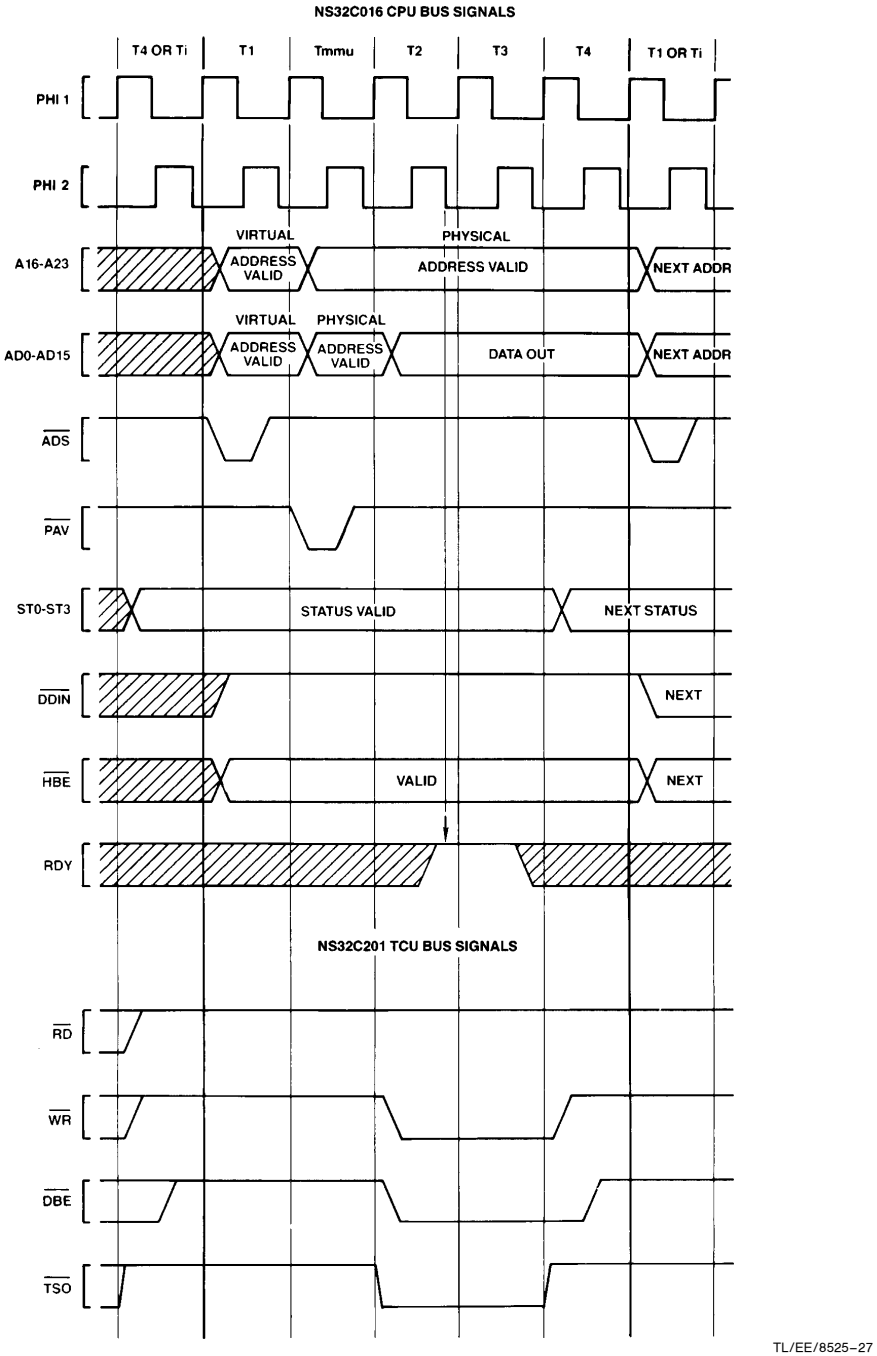
FIGURE 3-16. Write Cycle with Address Translation (CPU Action)



TL/EE/8525–26

FIGURE 3-17. Memory-Managed Read Cycle

3.0 Functional Description (Continued)



TL/EE/8525–27

FIGURE 3-18. Memory-Managed Write Cycle

3.0 Functional Description (Continued)

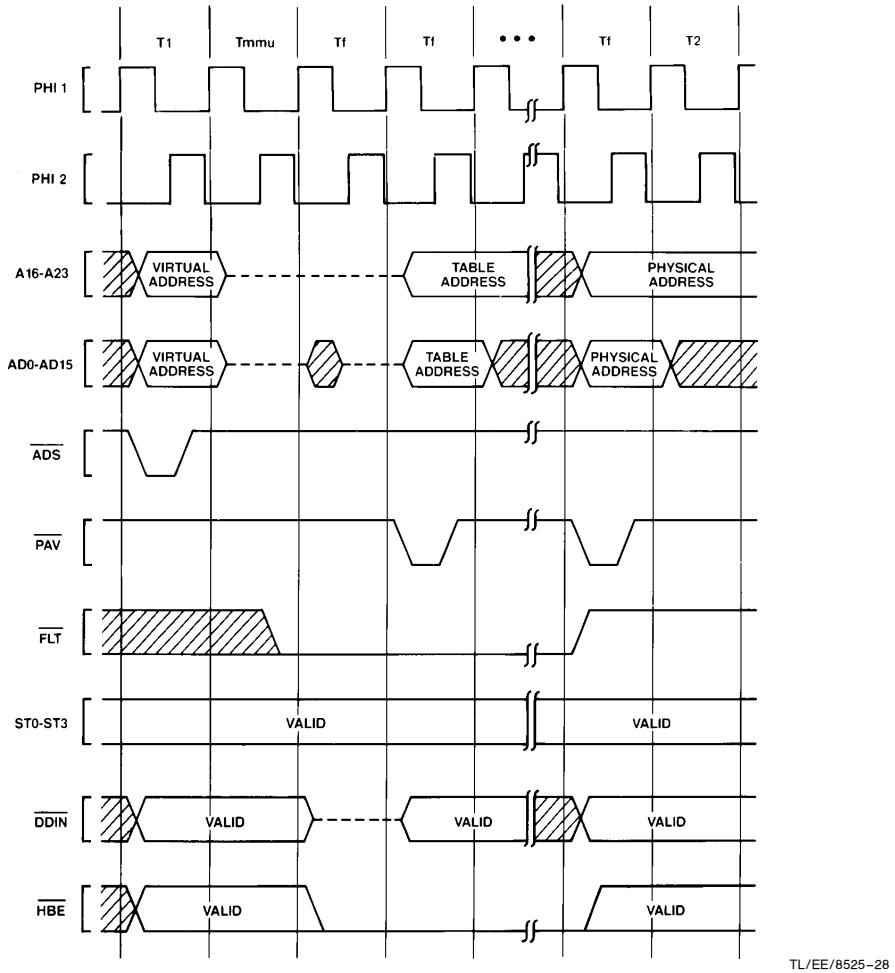
3.5.3 The $\overline{FLT}$ (Float) Pin

The $\overline{FLT}$ pin is used by the CPU for address translation support. Activating $\overline{FLT}$ during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the NS32082 MMU in order to update its internal translation Look-Aside Buffer (TLB) from page tables in memory, or to update certain status bits within them.

Figure 3-19 shows the effects of $\overline{FLT}$. Upon sampling $\overline{FLT}$ low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

1) Sets AD0–AD15, A16–A23 and $\overline{DDIN}$ to the TRI-STATE condition (“floating”).
2) Sets $\overline{HBE}$ low.
3) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See $\overline{RST}$/$\overline{ABT}$ description, Section 3.5.4.)

Note that the AD0–AD15 pins may be briefly asserted during the first idle T-State. The above conditions remain in effect until $\overline{FLT}$ again goes high. See the Timing Specifications, Section 4.



TL/EE/8525–28

FIGURE 3-19. $\overline{FLT}$ Timing

3.5.4 Aborting Bus Cycles

The $\overline{RST}/\overline{ABT}$ pin, apart from its Reset function (Section 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 $\overline{RST}/\overline{ABT}$ pin is held active for only one clock cycle.

If $\overline{RST}/\overline{ABT}$ is pulled low during Tmmu or Tf, this signals that the cycle must be aborted. The CPU itself will enter T2 and then Ti, thereby terminating the cycle. Since it is the MMU $\overline{PAV}$ signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was started.

The NS32082 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 reference 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. The information that is changed irrecoverably by such a partly-executed instruction does not affect its re-execution.

3.5.4.1 The Abort Interrupt

Upon aborting an instruction, the CPU immediately performs an interrupt through the ABT vector in the Interrupt Table (see Section 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 ABT interrupt will occur, in effect aborting the instruction that was being fetched.

3.5.4.2 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 NS32082 Memory Management Unit.

1) If $\overline{FLT}$ has not been applied to the CPU, the Abort pulse must occur during or before Tmmu. See the Timing Specifications, *Figure 4-23.*
2) If $\overline{FLT}$ has been applied to the CPU, the Abort pulse must be applied before the T-State in which $\overline{FLT}$ goes inactive. The CPU will not actually respond to the Abort command until $\overline{FLT}$ is removed. See *Figure 4-24.*
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 funtions 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 $\overline{RST}/\overline{ABT}$ 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.

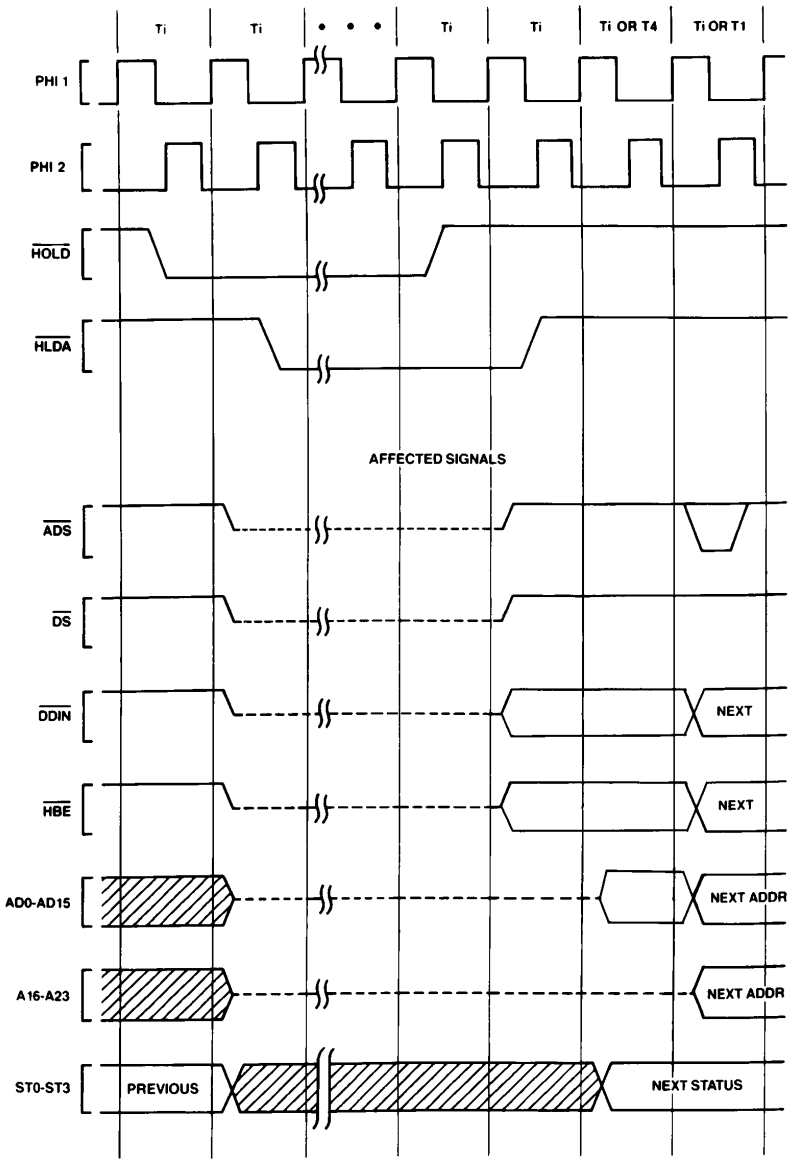
3.6 BUS ACCESS CONTROL

The NS32C016 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 $\overline{HOLD}$ (Hold Request) and $\overline{HLDA}$ (Hold Acknowledge) pins. By asserting $\overline{HOLD}$ low, an external device requests access to the bus. On receipt of $\overline{HLDA}$ from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0–AD15, A16–A23, $\overline{ADS}$, $\overline{DDIN}$ and $\overline{HBE}$ pins to the TRI-STATE condition. To return control of the bus to the CPU, the device sets $\overline{HOLD}$ inactive, and the CPU acknowledges return of the bus by setting $\overline{HLDA}$ inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the $\overline{HOLD}$ request is made, as the CPU must always complete the current bus cycle. *Figure 3-20* shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. *Figure 3-21* shows the sequence if the CPU is using the bus at the time that the $\overline{HOLD}$ 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.

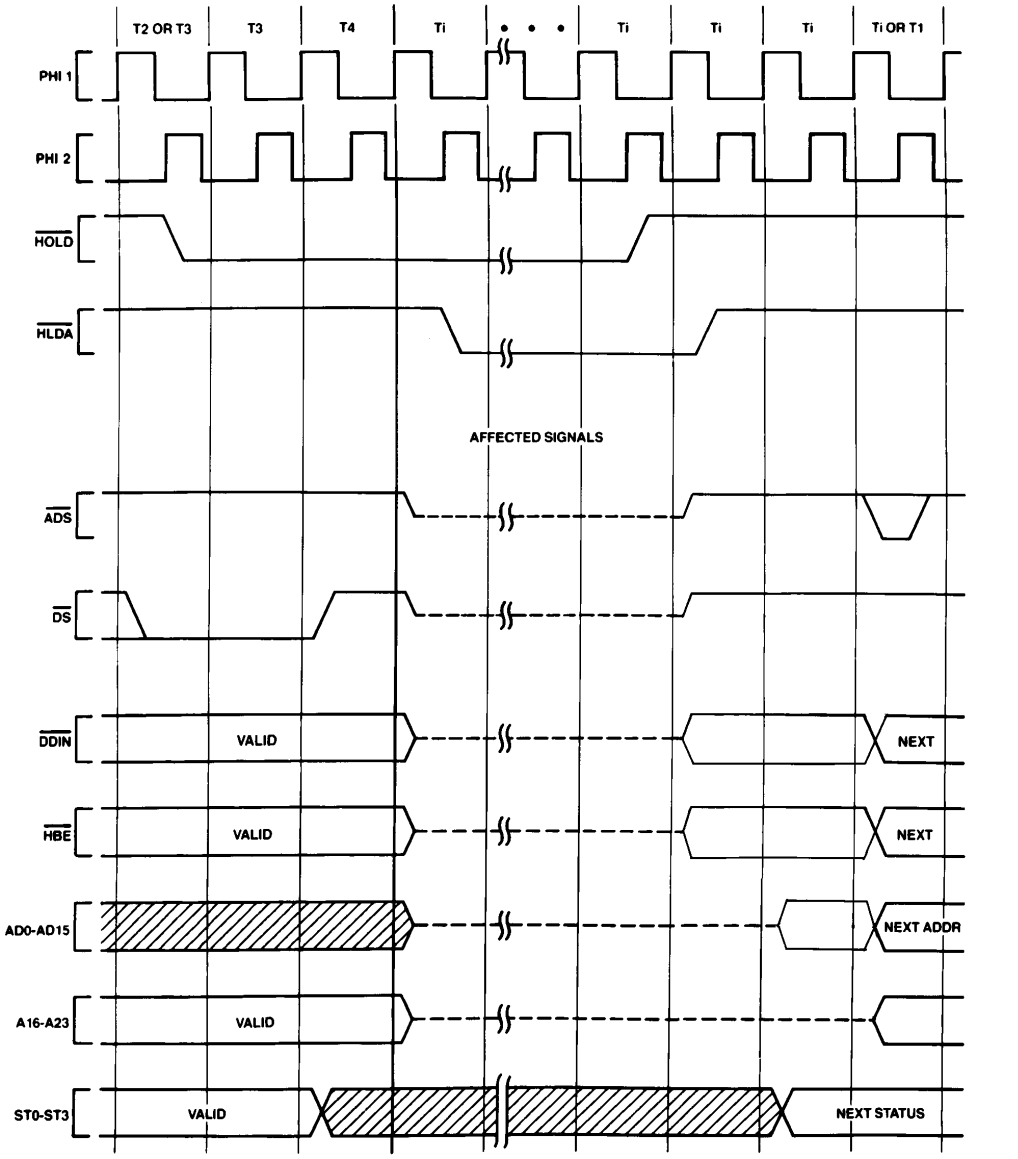
In a Memory-Managed system, the $\overline{HLDA}$ signal is connected in a daisy-chain through the NS32082, so that the MMU can release the bus if it is using it.

3.0 Functional Description (Continued)



TL/EE/8525–29

FIGURE 3-20. HOLD Timing, Bus Initially Idle



TL/EE/8525–30

FIGURE 3-21. HOLD Timing, Bus Initially Not Idle

3.0 Functional Description (Continued)

3.7 INSTRUCTION STATUS

In addition to the four bits of Bus Cycle status (ST0–ST3), the NS32C016 CPU also presents Instruction Status information on three 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.

$\overline{PFS}$ (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes, and is used that way by the NS32082 Memory Management Unit.

U/$\overline{S}$ 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. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, *Figure 4-22.*

$\overline{ILO}$ (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. As with the U/$\overline{S}$ pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification Section, *Figure 4-20.*

3.8 NS32C016 INTERRUPT STRUCTURE

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

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

$\overline{RST}$/$\overline{ABT}$, which may be used to abort a bus cycle and any associated instruction. See Section 3.5.4.

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.

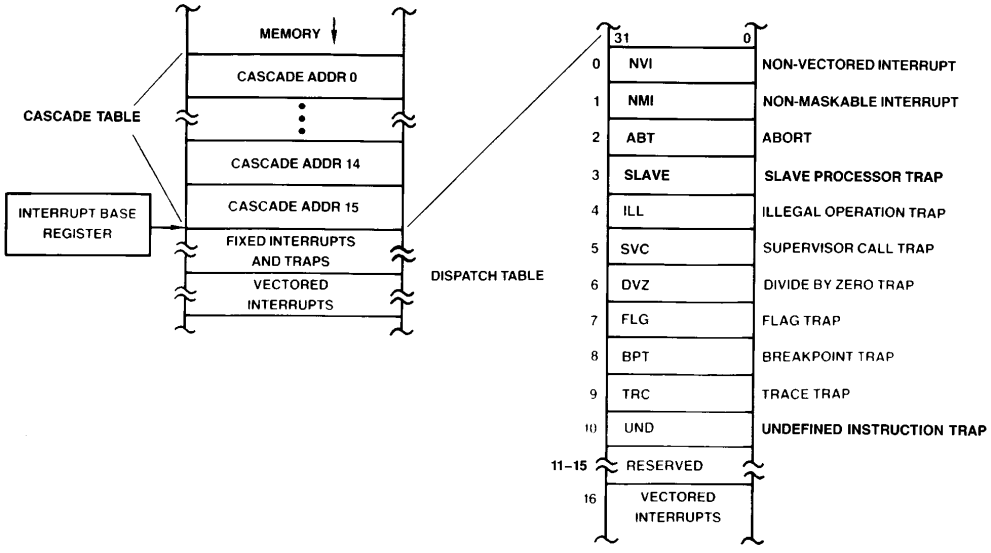
2) Vector Acquisition.

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

3) Service Call.

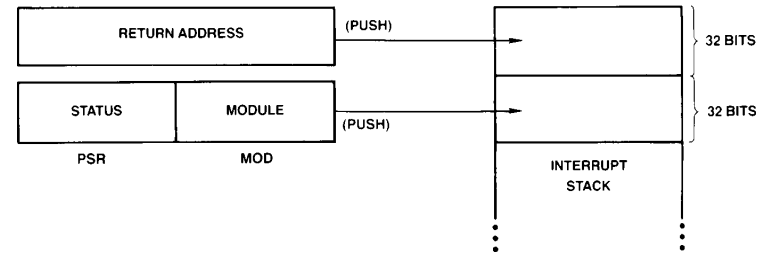
   The Vector is used as an index into the Interrupt Dispatch Table, whose base address is taken from the CPU Interrupt Base (INTBASE) Register. See *Figure 3-22.* A 32-bit External Procedure Descriptor is read from the table entry, and an External Procedure Call is performed using it. The MOD Register (16 bits) and Program Counter (32 bits) are pushed on the Interrupt Stack.

This process is illustrated in *Figure 3-23,* from the viewpoint of the programmer.

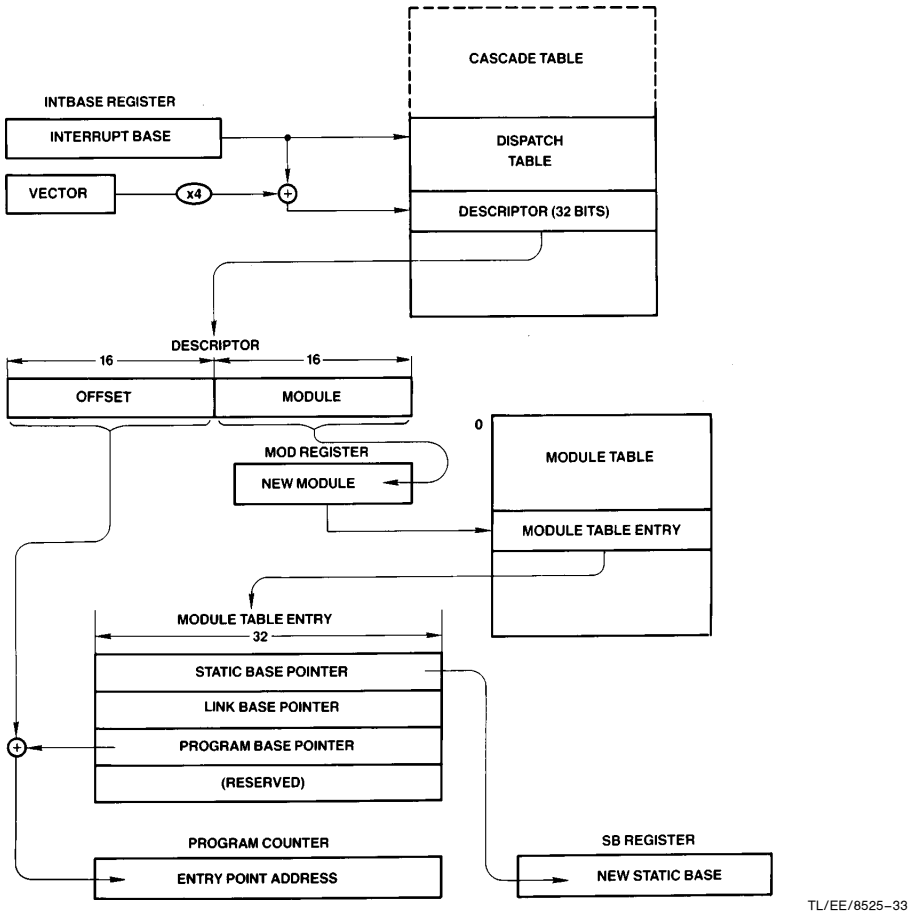


TL/EE/8525–31

FIGURE 3-22. Interrupt Dispatch and Cascade Tables



TL/EE/8525–32



TL/EE/8525–33

FIGURE 3-23. Interrupt/Trap Service Routine Calling Sequence

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 3-24)* 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 3-25*.

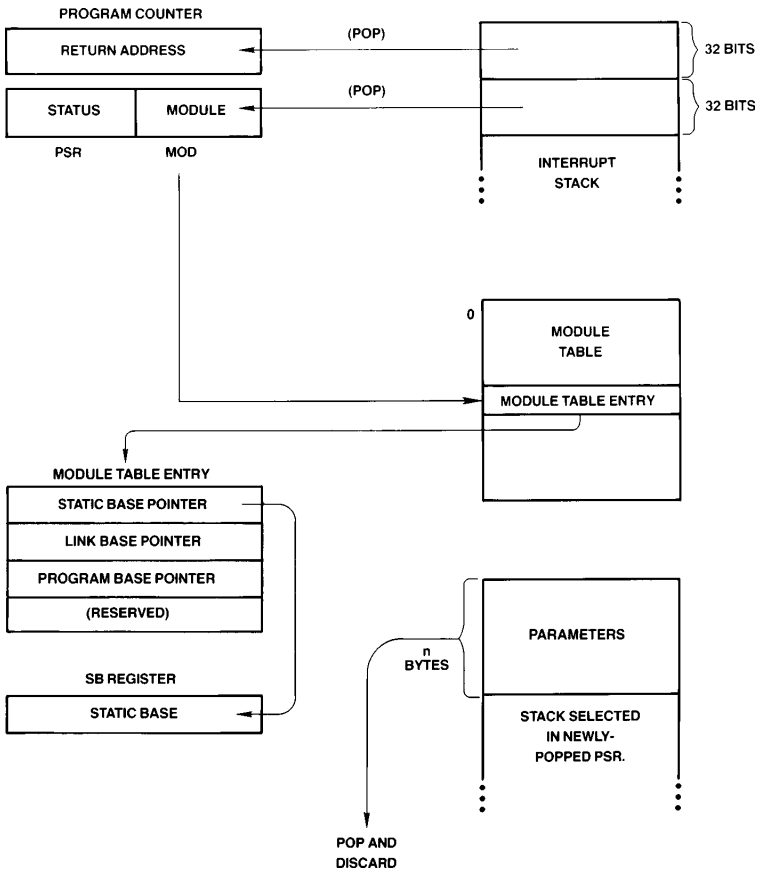
3.8.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)

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).

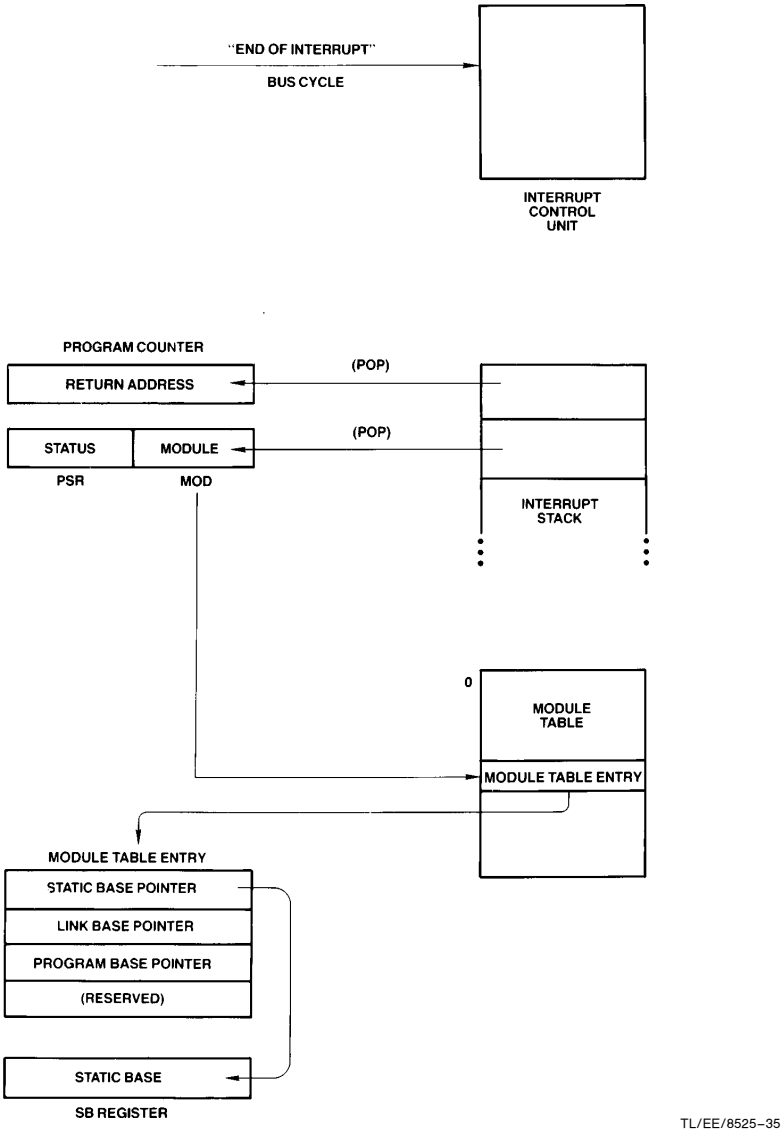
3.8.3.1 Non-Vectored Mode

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/8525–34

FIGURE 3-24. Return from Trap (RETT n) Instruction Flow



TL/EE/8525–35

FIGURE 3-25. Return from Interrupt (RET I) Instruction Flow

3.8.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Upon receipt of an interrupt request on the $\overline{INT}$ pin, the CPU performs an ''Interrupt Acknowledge, Master'' bus cycle (Section 3.4.2) 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 up to 256 levels of interrupt, provision is made both in the CPU and in the NS32202 Interrupt Control Unit (ICU) to transparently support cascading. *Figure 3-27* shows a typical cascaded configuration. 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 $\overline{INT}$ pin.

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

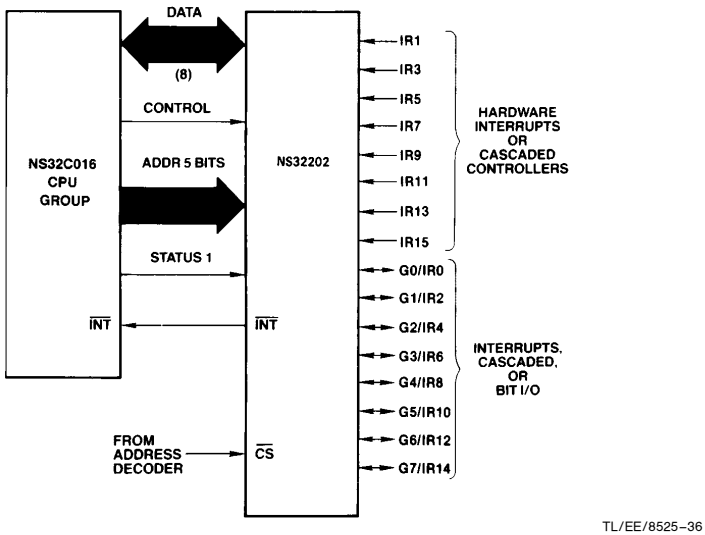
1) For each Cascaded ICU in the system, the Master ICU must be informed of the line number (0 to 15) on which it receives the cascaded requests.
2) A Cascade Table must be established in memory. The Cascade Table is located in a NEGATIVE direction from the location indicated by the CPU Interrupt Base (INTBASE) Register. Its entries are 32-bit addresses, pointing to the Vector Registers of each of up to 16 Cascaded ICUs.

Figure 3-22 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 (Section 3.4.2), 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 (Section 3.4.2), whereupon the Master ICU again provides the negative Cascaded 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 (Section 3.4.2), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

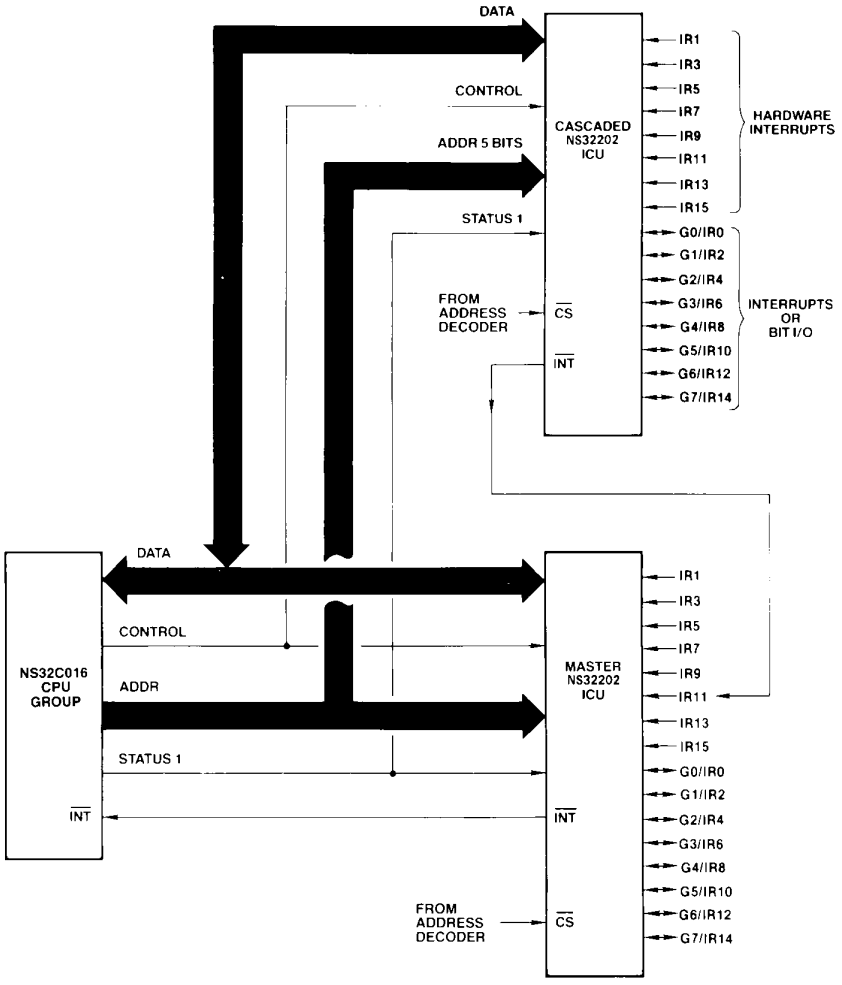
Note: If an interrupt must be masked off, the CPU can do so by setting the corresponding bit in the Interrupt Mask Register of the Interrupt Controller. However, if an interrupt is set pending during the CPU instruction that masks off that interrupt, the CPU may still perform an interrupt acknowledge cycle following that instruction since it might have sampled the $\overline{INT}$ 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.



TL/EE/8525–36

FIGURE 3-26. Interrupt Control Unit Connections (16 Levels)

3.0 Functional Description (Continued)



TL/EE/8525–37

FIGURE 3-27. Cascaded Interrupt Control Unit Connections

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 (Section 3.4.2) when processing of this interrupt actually begins. The Interrupt Acknowledge cycle differs from that provided for Maskable Interrupts in that the address presented is $FFFF00_{16}$. 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 Section 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) below 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 NS32C016 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 (Section 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.

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.

3.8.6 Prioritization

The NS32C016 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

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

3.8.7 Interrupt/Trap Sequences: Detail Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called ''Service'' is defined in *Figure 3-28.* 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 sequenced followed in processing either Maskable or Non-Maskable Interrupts (on the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pins, respectively), see Section 3.8.7.1. For Abort interrupts, see Section 3.8.7.4. For the Trace Trap, see Section 3.8.7.3, and for all other traps see Section 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 $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). 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 $FFFF00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2). Discard the byte read.
   b. Set ''Vector'' to 0.
   c. Go to Step 8.
5. Here the interrupt is Vectored. Read ''Byte'' from address $FFFE00_{16}$, applying Status Code 0100 (Interrupt Acknowledge, Master: Section 3.4.2).
6. If ''Byte'' $\geq$ 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: Section 3.4.2).
8. Push the PSR copy (from Step 2) onto the Interrupt Stack as a 16-bit value.
9. Perform Service (Vector, Return Address), *Figure 3-28.*

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 new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.
4) 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.
5) Flush Queue: Non-sequentially fetch first instruction of Interrupt Routine.
6) Push MOD Register onto the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)
7) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.

FIGURE 3-28. 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.
4) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set ''Return Address'' to the address of the first byte of the trapped instruction.
6) Perform Service (Vector, Return Address), *Figure 3-28.*

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) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
4) Set ''Vector'' to 9.
5) Set ''Return Address'' to the address of the next instruction.
6) Perform Service (Vector, Return Address), *Figure 3-28.*

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) Push the PSR copy onto the Interrupt Stack as a 16-bit value.
5) Set ''Vector'' to 2.
6) Set ''Return Address'' to the address of the first byte of the aborted instruction.
7) Perform Service (Vector, Return Address), *Figure 3-28.*

3.9 SLAVE PROCESSOR INSTRUCTIONS

The NS32C016 CPU recognizes three groups of instructions as being executable by external Slave Processors:

Floating Point Instruction Set

Memory Management Instruction Set

Custom Instruction Set

Each Slave Instruction Set is validated by a bit in the Configuration Register (Section 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.

3.9.1 Slave Processor Protocol

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 3-29.* While applying Status Code 1111 (Broadcast ID, Section 3.4.2), the CPU transfers the ID Byte on the least-significant half of the Data Bus (AD0–AD7). 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, Section 3.4.2). 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 Addressing Mode fields within the Operation Word, the CPU starts fetching operands 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, Section 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 Send ID Byte.
2	OP	CPU Sends Operation Word.
3	OP	CPU Sends Required Operands.
4	—	Slave Starts Execution. CPU Pre-Fetches.
5	—	Slave Pulses $\overline{\text{SPC}}$ Low.
6	ST	CPU Reads Status Word. (Trap? Alter Flags?)
7	OP	CPU Reads Results (If Any).

FIGURE 3-29. Slave Processor Protocol

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 $\overline{SPC}$ low. To allow for this, and for the Address Translation strap function, $\overline{AT}/\overline{SPC}$ 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, Section 3.4.2).

Upon receiving the pulse on $\overline{SPC}$, the CPU uses $\overline{SPC}$ to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Section 3.4.2). This word has the format shown in *Figure 3-30.* 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, Section 3.4.2).

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.

3.9.2 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 Series 32000 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 3-30).*

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
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

Notes:

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.

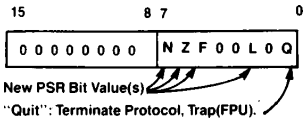


FIGURE 3-30. Slave Processor Status Word Format

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.

3.9.3 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 Series 32000 Instruction Set Reference Manual and the NS32082 MMU Data Sheet.

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 Series 32000 Instruction Set Reference Manual and the NS32082 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.

3.0 Functional Description (Continued)

3.9.4 Custom Slave Instructions

Provided in the NS32C016 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.

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	readi	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

Notes:

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 NS32C016 PIN DESCRIPTIONS

The following is a brief description of all NS32C016 pins. The descriptions reference portions of the Functional Description, Section 3.

4.1.1 Supplies

Logic Power (V_{CCL}): +5V positive supply for on-chip logic. Section 3.1.

Buffer Power (V_{CCB}): +5V positive supply for on-chip output buffers. Section 3.1.

Logic Ground (GNDL): Ground reference for on-chip logic. Section 3.1.

Buffer Ground (GNDB): Ground reference for on-chip drivers connected to output pins. Section 3.1.

4.1.2 Input Signals

Clocks (PHI1, PHI2): Two-phase clocking signals. Section 3.2.

Ready (RDY): Active high. While RDY is inactive, the CPU extends the current bus cycle to provide for a slower memory or peripheral reference. Upon detecting RDY active, the CPU terminates the bus cycle. Section 3.4.1.

Hold Request ($\overline{\text{HOLD}}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Section 3.6.

Note: If the $\overline{\text{HOLD}}$ signal is generated asynchronously, its 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. Section 3.8.

Non-Maskable Interrupt ($\overline{\text{NMI}}$): Active low. Non-Maskable Interrupt request. Section 3.8.

Reset/Abort ($\overline{\text{RST}}/\overline{\text{ABT}}$): Active low. If held active for one clock cycle and released, this pin causes an Abort Command, Section 3.5.4. If held longer, it initiates a Reset, Section 3.3.

4.1.3 Output Signals

Address Bits 16–23 (A16–A23): These are the most significant 8 bits of the memory address bus. Section 3.4.

Address Strobe ($\overline{\text{ADS}}$): Active low. Controls address latches; indicates start of a bus cycle. Section 3.4.

Data Direction In ($\overline{\text{DDIN}}$): Active low. Status signal indicating direction of data transfer during a bus cycle. Section 3.4.

High Byte Enable ($\overline{\text{HBE}}$): Active low. Status signal enabling transfer on the most significant byte of the Data Bus. Section 3.4; Section 3.4.3.

Note: In the current NS32C016, the $\overline{\text{HBE}}$ signal is forced low by the CPU when $\overline{\text{FLT}}$ is asserted by the MMU. However, in future revisions of the CPU, $\overline{\text{HBE}}$ will no longer be affected by $\overline{\text{FLT}}$. Therefore, in a memory managed system, an external 'AND' gate is required. This is shown in *Figure B-1* in Appendix B.

Status (ST0–ST3): Active high. Bus cycle status code, ST0 least significant. Section 3.4.2. 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.

Hold Acknowledge ($\overline{\text{HLDA}}$): Active low. Applied by the CPU in response to $\overline{\text{HOLD}}$ input, indicating that the bus has been released for DMA or multiprocessing purposes. Section 3.6.

User/Supervisor (U/$\overline{\text{S}}$): User or Supervisor Mode status. Section 3.7. High state indicates User Mode, low indicates Supervisor Mode. Section 3.7.

Interlocked Operation ($\overline{\text{ILO}}$): Active low. Indicates that an interlocked instruction is being executed. Section 3.7.

Program Flow Status ($\overline{\text{PFS}}$): Active Low. Pulse indicates beginning of an instruction execution. Section 3.7.

4.1.4 Input-Output Signals

Address/Data 0–15 (AD0–AD15): Multiplexed Address/Data information. Bit 0 is the least significant bit of each. Section 3.4.

Address Translation/Slave Processor Control ($\overline{\text{AT}}/\overline{\text{SPC}}$): Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by Slave Processors to acknowledge completion of a slave instruction. Section 3.4.6; Section 3.9. Sampled on the rising edge of Reset as Address Translation Strap. Section 3.5.1.

In non-memory-managed systems this pin should be pulled up to V_{CC} through a 10 kΩ resistor.

Data Strobe/Float ($\overline{\text{DS}}/\overline{\text{FLT}}$): Active low. Data Strobe output, Section 3.4, or Float Command input, Section 3.5.3. Pin function is selected on $\overline{\text{AT}}/\overline{\text{SPC}}$ pin, Section 3.5.1.

4.0 Device Specifications (Continued)

4.2 ABSOLUTE MAXIMUM RATINGS

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

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	1.5 Watt

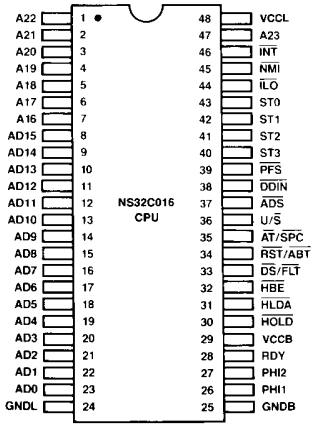
Note: *Absolute maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended; operation should be limited to those conditions specified under Electrical Characteristics.*

4.3 ELECTRICAL CHARACTERISTICS: $T_A = -40°C$ to $+85°C$, $V_{CC} = 5V \pm 10\%$, 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	0.90 V_{CC}		$V_{CC} + 0.5$	V
V_{CL}	Low Level Clock Voltage	PHI1, PHI2 pins only	−0.5		0.10 V_{CC}	V
V_{CRT}	Clock Input Ringing Tolerance	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	High Level Output Voltage	$I_{OH} = -400\ \mu A$	0.90 V_{CC}			V
V_{OL}	Low Level Output Voltage	$I_{OL} = 2$ mA			0.10 V_{CC}	V
I_{ILS}	$\overline{AT}/\overline{SPC}$ Input Current (low)	$V_{IN} = 0.4V$, $\overline{AT}/\overline{SPC}$ in input mode	0.05		1.0	mA
I_I	Input Load Current	$0 \le V_{IN} \le V_{CC}$, All inputs except PHI1, PHI2, $\overline{AT}/\overline{SPC}$	−20		20	μA
I_L	Leakage Current Output and IO Pins in TRI-STATE/ Input Mode	$0.4 \le V_{IN} \le V_{CC}$	−20		20	μA
I_{CC}	Active Supply Current	$I_{OUT} = 0$, $T_A = 25°C$		70	100	mA

Connection Diagram

Dual-In-Line Package



TL/EE/8525–2

Top View

FIGURE 4-1

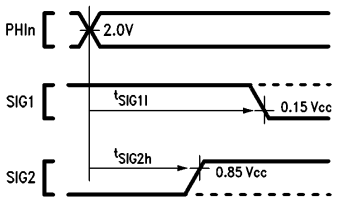
Order Number NS32C016D-10, NS32C016D-15, NS32C016N-10 or NS32C016N-15
See NS Package Number D48A or N48A

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; to 15% or 85% of V_{CC} on all the CMOS output signals, and to 0.8V or 2.0V on all the TTL input signals as illustrated in *Figures 4-2* and *4-3* unless specifically stated otherwise.



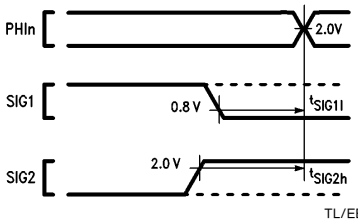
TL/EE/8525–39

FIGURE 4-2. Timing Specification Standard (CMOS Output Signals)

ABBREVIATIONS:

L.E. — leading edge R.E. — rising edge

T.E. — trailing edge F.E. — falling edge



TL/EE/8525–40

FIGURE 4-3. Timing Specification Standard (TTL Input Signals)

4.4.2 Timing Tables

4.4.2.1 Output Signals: Internal Propagation Delays, NS32C016-10 and NS32C016-15

Maximum times assume capacitive loading of 75 pF, on the address/data bus signals and 50 pF on all other signals.

Name	Figure	Description	Reference/Conditions	NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	
t_{ALv}	4-4	Address bits 0–15 valid	after R.E., PHI1 T1		40		35	ns
t_{ALh}	4-4	Address bits 0–15 hold	after R.E., PHI1 Tmmu or T2	5		5		ns
t_{Dv}	4-4	Data valid (write cycle)	after R.E., PHI1 T2		50		35	ns
t_{Dh}	4-4	Data hold (write cycle)	after R.E., PHI1 next T1 or Ti	0		0		ns
t_{AHv}	4-4	Address bits 16–23 valid	after R.E., PHI1 T1		40		35	ns
t_{AHh}	4-4	Address bits 16–23 hold	after R.E., PHI1 next T1 or Ti	0		0		ns
t_{ALADSs}	4-5	Address bits 0–15 set up	before $\overline{ADS}$ T.E.	25		20		ns
t_{AHADSs}	4-5	Address bits 16–23 set up	before $\overline{ADS}$ T.E.	25		20		ns
t_{ALADSh}	4-9	Address bits 0–15 hold	after $\overline{ADS}$ T.E.	15		10		ns
t_{AHADSh}	4-9	Address bits 16–23 hold	after $\overline{ADS}$ T.E.	15		10		ns
t_{ALf}	4-5	Address bits 0–15 floating (no MMU)	after R.E., PHI1 T2		25		20	ns
t_{ALMf}	4-9	Address bits 0–15 floating (with MMU)	after R.E., PHI1 TMMU		25		20	ns
t_{AHMf}	4-9	Address bits 16–23 floating (with MMU)	after R.E., PHI1 TMMU		25		20	ns
t_{HBEv}	4-4	$\overline{HBE}$ signal valid	after R.E., PHI1 T1		45		35	ns
t_{HBEh}	4-4	$\overline{HBE}$ signal hold	after R.E., PHI1 next T1 or Ti	0		0		ns
t_{STv}	4-4	Status (ST0–ST3) valid	after R.E., PHI1 T4 (before T1, see note)		45		35	ns
t_{STh}	4-4	Status (ST0–ST3) hold	after R.E., PHI1 T4 (after T1)	0		0		ns
t_{DDINv}	4-5	$\overline{DDIN}$ signal valid	after R.E., PHI1 T1		50		35	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32C016-10 and NS32C016-15 (Continued)

Name	Figure	Description	Reference/Conditions	NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	
t_{DDINh}	4-5	$\overline{DDIN}$ signal hold	after R.E., PHI1 next T1 or Ti	0		0		ns
t_{ADSa}	4-4	$\overline{ADS}$ signal active (low)	after R.E., PHI1 T1		35		26	ns
t_{ADSia}	4-4	$\overline{ADS}$ signal inactive	after R.E., PHI2 T1		40		30	ns
t_{ADSw}	4-4	$\overline{ADS}$ pulse width	at 15% V_{CC} (both edges)	30		25		ns
t_{DSa}	4-4	$\overline{DS}$ signal active (low)	after R.E., PHI1 T2		40		30	ns
t_{DSia}	4-4	$\overline{DS}$ signal inactive	after R.E., PHI1 T4		40		30	ns
t_{ALf}	4-6	AD0–AD15 floating	after R.E., PHI1 T1 (caused by $\overline{HOLD}$)		25		20	ns
t_{AHf}	4-6	A16–A23 floating	after R.E., PHI1 T1 (caused by $\overline{HOLD}$)		25		20	ns
t_{DSf}	4-6	$\overline{DS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		50		40	ns
t_{ADSf}	4-6	$\overline{ADS}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		50		40	ns
t_{HBEf}	4-6	$\overline{HBE}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		50		40	ns
t_{DDINf}	4-6	$\overline{DDIN}$ floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		50		40	ns
t_{HLDAa}	4-6	$\overline{HLDA}$ signal active (low)	after R.E., PHI1 Ti		30		25	ns
t_{HLDAia}	4-8	$\overline{HLDA}$ signal inactive	after R.E., PHI1 Ti		40		30	ns
t_{DSr}	4-8	$\overline{DS}$ signal returns from floating (caused by HOLD)	after R.E., PHI1 Ti		55		40	ns
t_{ADSr}	4-8	$\overline{ADS}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		55		40	ns
t_{HBEr}	4-8	$\overline{HBE}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		55		40	ns
t_{DDINr}	4-8	$\overline{DDIN}$ signal returns from floating (caused by $\overline{HOLD}$)	after R.E., PHI1 Ti		55		40	ns
t_{DDINf}	4-9	$\overline{DDIN}$ signal floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		55		50	ns
t_{HBEl}	4-9	$\overline{HBE}$ signal low (caused by $\overline{FLT}$)	after $\overline{FLT}$ F.E.		40		30	ns
t_{DDINr}	4-10	$\overline{DDIN}$ signal returns from floating (caused by $\overline{FLT}$)	after $\overline{FLT}$ R.E.		40		30	ns
t_{HBEr}	4-10	$\overline{HBE}$ signal returns from low (caused by $\overline{FLT}$)	after $\overline{FLT}$ R.E.		35		25	ns
t_{SPCa}	4-13	$\overline{SPC}$ output active (low)	after R.E., PHI1 T1		35		26	ns
t_{SPCia}	4-13	$\overline{SPC}$ output inactive	after R.E., PHI1 T4		35		26	ns
t_{SPCnf}	4-15	$\overline{SPC}$ output nonforcing	after R.E., PHI2 T4		30		25	ns
t_{Dv}	4-13	Data valid (slave processor write)	after R.E., PHI1 T1		50		35	ns
t_{Dh}	4-13	Data hold (slave processor write)	after R.E., PHI1 next T1 or Ti	0		0		ns
t_{PFSw}	4-18	$\overline{PFS}$ pulse width	at 15% V_{CC} (both edges)	50		40		ns
t_{PFSa}	4-18	$\overline{PFS}$ pulse active (low)	after R.E., PHI2		40		35	ns
t_{PFSia}	4-18	$\overline{PFS}$ pulse inactive	after R.E., PHI2		40		35	ns
t_{ILOs}	4-20a	$\overline{ILO}$ signal setup	before R.E., PHI1 T1 of first interlocked write cycle	50		35		ns
t_{ILOh}	4-20b	$\overline{ILO}$ signal hold	after R.E., PHI1 T3 of last interlocked read cycle	10		7		ns
t_{ILOa}	4-21	$\overline{ILO}$ signal active (low)	after R.E., PHI1		35		30	ns

4.0 Device Specifications (Continued)

4.4.2.1 Output Signals: Internal Propagation Delays, NS32C016-10 and NS32C016-15 (Continued)

Name	Figure	Description	Reference/Conditions	NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	
t_{ILOia}	4-21	$\overline{ILO}$ signal inactive	after R.E., PHI1		35		30	ns
t_{USV}	4-22	U/$\overline{S}$ signal valid	after R.E., PHI1 T4		35		30	ns
t_{USh}	4-22	U/$\overline{S}$ signal hold	after R.E., PHI1 T4	8		6		ns
t_{NSPF}	4-19b	Nonsequential fetch to next $\overline{PFS}$ clock cycle	after R.E., PHI1 T1	4		4		t_{Cp}
t_{PFNS}	4-19a	$\overline{PFS}$ clock cycle to next nonsequential fetch	before R.E., PHI1 T1	4		4		t_{Cp}
t_{LXPF}	4-29	Last operand transfer of an instruction to next $\overline{PFS}$ clock cycle	before R.E., PHI1 T1 of first bus cycle of transfer	0		0		t_{Cp}

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: NS32C016-10 and NS32C016-15

Name	Figure	Description	Reference/Conditions	NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	
t_{PWR}	4-25	Power stable to $\overline{RST}$ R.E.	after V_{CC} reaches 4.5V	50		33		μs
t_{DIs}	4-5	Data in setup (read cycle)	before F.E., PHI2 T3	15		10		ns
t_{DIh}	4-5	Data in hold (read cycle)	after F.E., PHI1 T4	3		3		ns
t_{HLDa}	4-6	$\overline{HOLD}$ active (low) setup time (see note)	before F.E., PHI2 TX1	25		17		ns
t_{HLDia}	4-8	$\overline{HOLD}$ inactive setup time	before F.E., PHI2 Ti	25		17		ns
t_{HLDh}	4-6	$\overline{HOLD}$ hold time	after R.E., PHI1 TX2	0		0		ns
t_{FLTa}	4-9	$\overline{FLT}$ active (low) setup time	before F.E., PHI2 Tmmu	25		17		ns
t_{FLTia}	4-10	$\overline{FLT}$ inactive setup time	before F.E., PHI2 T2	25		17		ns
t_{RDYs}	4-11, 4-12	RDY setup time	before F.E., PHI2 T2 or T3	15		10		ns
t_{RDYh}	4-11, 4-12	RDY hold time	after F.E., PHI1 T3	5		5		ns
t_{ABTs}	4-23	$\overline{ABT}$ setup time ($\overline{FLT}$ inactive)	before F.E., PHI2 Tmmu	20		13		ns
t_{ABTs}	4-24	$\overline{ABT}$ setup time ($\overline{FLT}$ active)	before F.E., PHI2 Tf	20		13		ns
t_{ABTh}	4-23	$\overline{ABT}$ hold time	after R.E., PHI1	0		0		ns
t_{RSTs}	4-25, 4-26	$\overline{RST}$ setup time	before F.E., PHI1	10		8		ns
t_{RSTw}	4-26	$\overline{RST}$ pulse width	at 0.8V (both edges)	64		64		t_{Cp}
t_{INTs}	4-27	$\overline{INT}$ setup time	before R.E., PHI1	20		15		ns
t_{NMIw}	4-28	$\overline{NMI}$ pulse width	at 0.8V (both edges)	70		70		ns
t_{DIs}	4-14	Data setup (slave read cycle)	before F.E., PHI2 T1	15		10		ns
t_{DIh}	4-14	Data hold (slave read cycle)	after R.E., PHI1 T4	3		3		ns

4.0 Device Specifications (Continued)

4.4.2.2 Input Signal Requirements: NS32C016-10 and NS32C016-15 (Continued)

Name	Figure	Description	Reference/Conditions	NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	
t_{SPCd}	4-15	$\overline{SPC}$ pulse delay from slave	after R.E., PHI2 T4	30		25		ns
t_{SPCs}	4-15	$\overline{SPC}$ setup time	before F.E., PHI1	30		25		ns
t_{SPCw}	4-15	$\overline{SPC}$ pulse width from slave processor (async. input)	at 0.8V (both edges)	20		20		ns
t_{ATs}	4-16	$\overline{AT}/\overline{SPC}$ setup for address translation strap	before R.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	1		1		t_{Cp}
t_{ATh}	4-16	$\overline{AT}/\overline{SPC}$ hold for address translation strap	after F.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	2		2		t_{Cp}

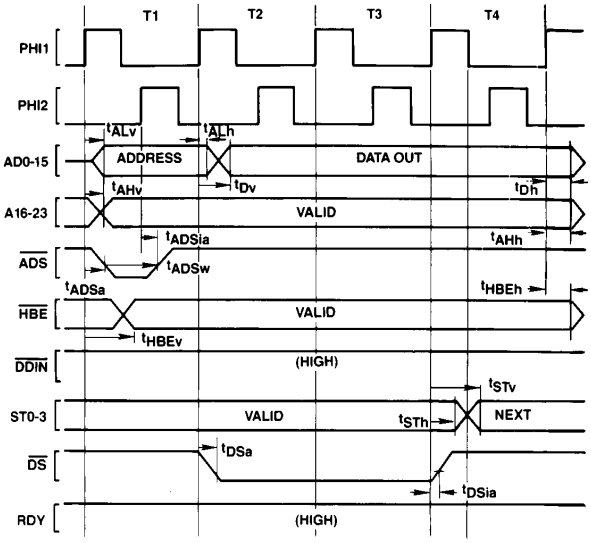
Note: This setup time is necessary to ensure prompt acknowledgement via $\overline{HLDA}$ and the ensuing floating of CPU off the buses. Note that the time from the receipt of the $\overline{HOLD}$ signal until the CPU floats is a function of the time $\overline{HOLD}$ signal goes low, the state of the RDY input (in MMU systems), and the length of the current MMU cycle.

4.4.2.3 Clocking Requirements: NS32C016-10 and NS32C016-15

Name	Figure	Description	Reference/Conditions	NS32C016-10		NS32C016-15		Units
				Min	Max	Min	Max	
t_{Cp}	4-17	Clock period	R.E., PHI1, PHI2 to next R.E., PHI1, PHI2	100	250	66	250	ns
t_{CLw}	4-17	PHI1, PHI2 pulse width	At 2.0V on PHI1, PHI2 (both edges)	$0.5t_{Cp}$ -10 ns		$0.5t_{Cp}$ -6 ns		
t_{CLh}	4-17	PHI1, PHI2 High Time	At 90% V_{CC} on PHI1, PHI2	$0.5t_{Cp}$ -15 ns		$0.5t_{Cp}$ -10 ns		
t_{CLl}	4-17	PHI1, PHI2 Low Time	At 15% V_{CC} on PHI1, PHI2	$0.5t_{Cp}$ -5 ns		$0.5t_{Cp}$ -5 ns		
$t_{nOVL(1,2)}$	4-17	Non-overlap time	At 15% V_{CC} on PHI1, PHI2	-2	2	-2	2	ns
t_{nOVLas}		Non-overlap asymmetry ($t_{nOVL(1)} - t_{nOVL(2)}$)	At 15% V_{CC} on PHI1, PHI2	-3	3	-3	3	ns
t_{CLwas}		PHI1, PHI2 asymmetry ($t_{CLw(1)} - t_{CLw(2)}$)	At 2.0V on PHI1, PHI2	-5	5	-3	3	ns

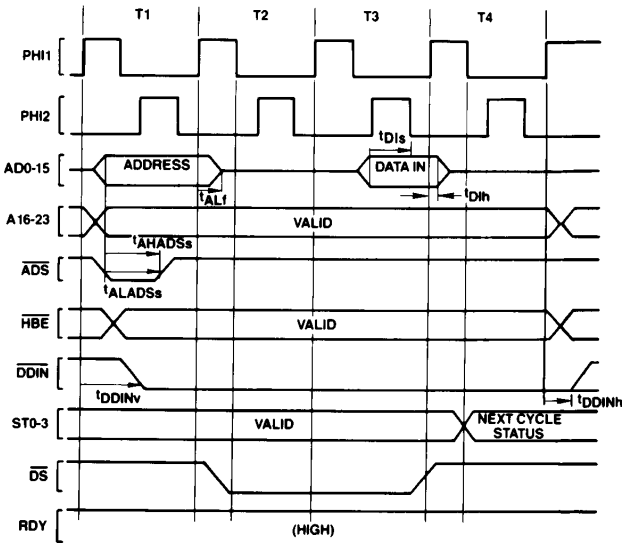
4.0 Device Specifications (Continued)

4.4.3 Timing Diagrams



TL/EE/8525–41

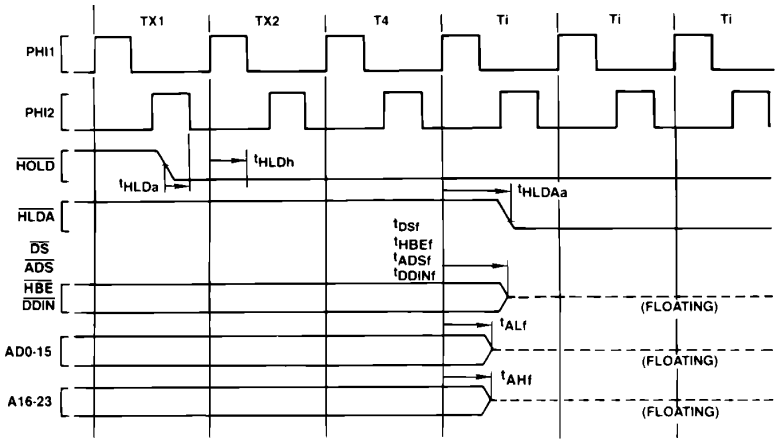
FIGURE 4-4. Write Cycle



TL/EE/8525–42

FIGURE 4-5. Read Cycle

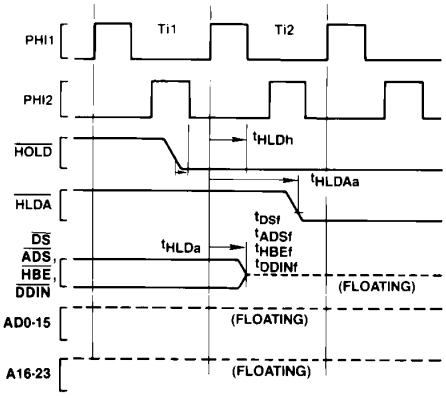
4.0 Device Specifications (Continued)



TL/EE/8525–43

FIGURE 4-6. Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially)

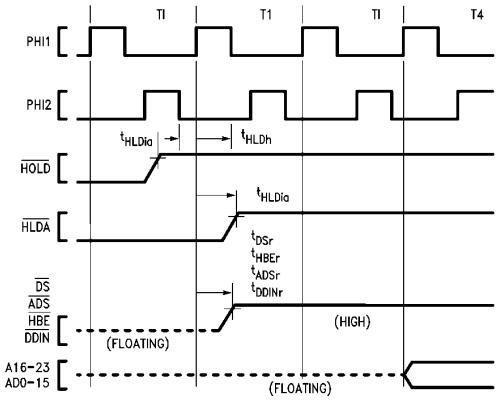
Note that whenever the CPU is not idling (not in Ti), the $\overline{\text{HOLD}}$ request ($\overline{\text{HOLD}}$ low) must be active t_{HLDa} before the falling edge of PHI2 of the clock cycle that appears two clock cycles before T4 (TX1) and stay low until t_{HLDh} after the rising edge of PHI1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



TL/EE/8525–44

FIGURE 4-7. Floating by $\overline{\text{HOLD}}$ Timing (CPU Initially Idle)

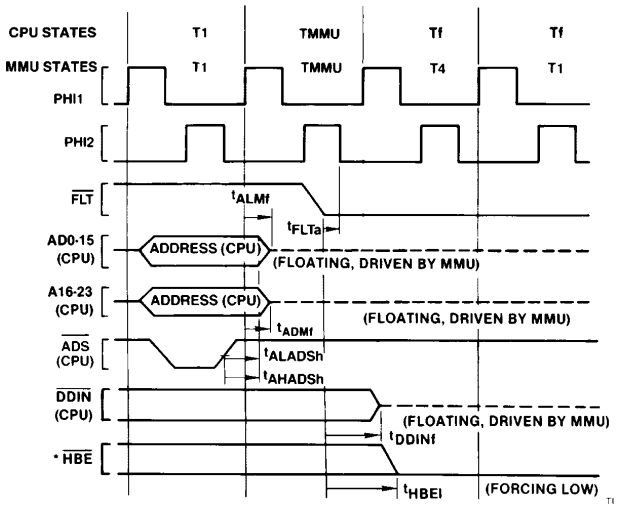
Note that during Ti1 the CPU is already idling.



TL/EE/8525–80

FIGURE 4-8. Release from $\overline{\text{HOLD}}$

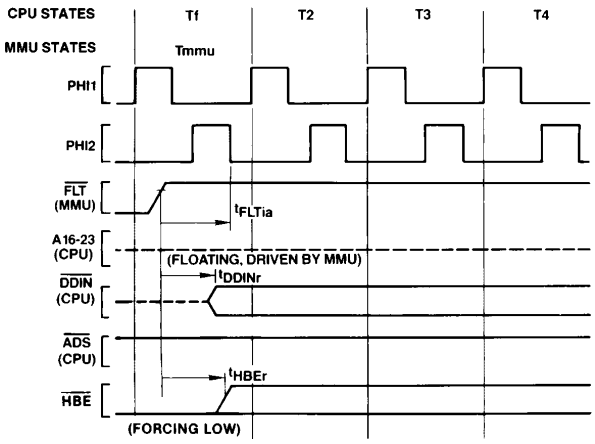
4.0 Device Specifications (Continued)



TL/EE/8525–46

*Note: In future higher speed versions of the NS32C016, $\overline{HBE}$ will no longer be affected by $\overline{FLT}$. See *Figure B-1* in Appendix B for the required modification to the interface logic.

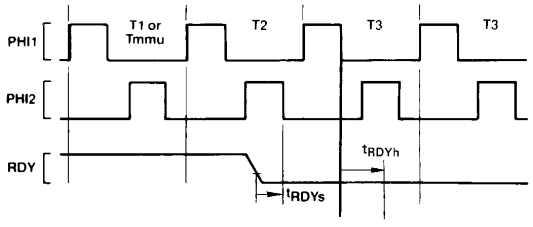
FIGURE 4-9. $\overline{FLT}$ Initiated Cycle Timing



TL/EE/8525–47

Note that 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.

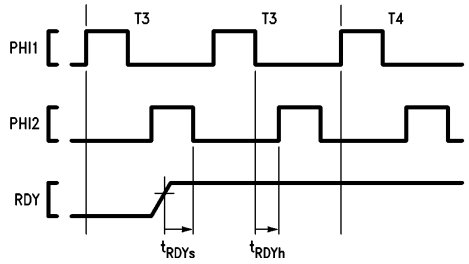
FIGURE 4-10. Release from $\overline{FLT}$ Timing



TL/EE/8525–48

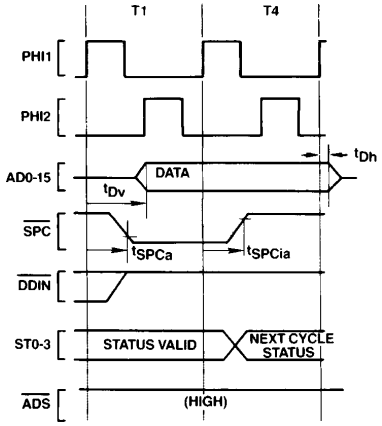
FIGURE 4-11. Ready Sampling (CPU Initially READY)

4.0 Device Specifications (Continued)



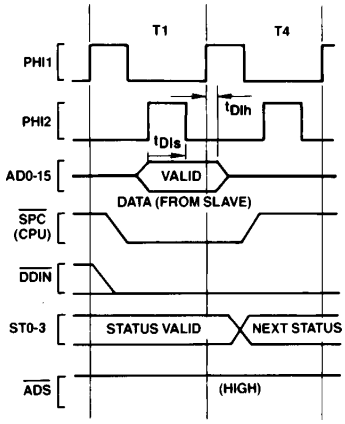
TL/EE/8525–49

FIGURE 4-12. Ready Sampling (CPU Initially NOT READY)



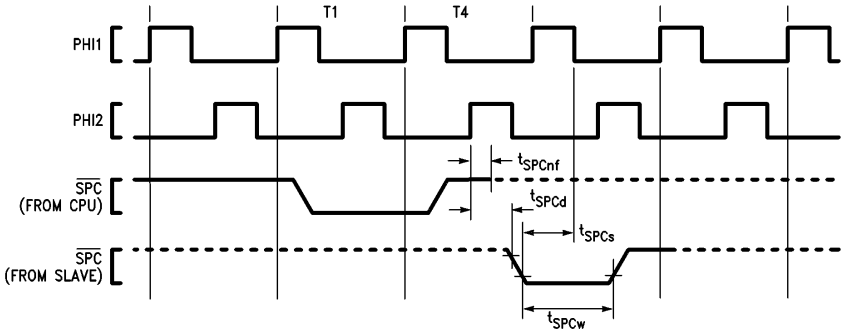
TL/EE/8525–50

FIGURE 4-13. Slave Processor Write Timing



TL/EE/8525–51

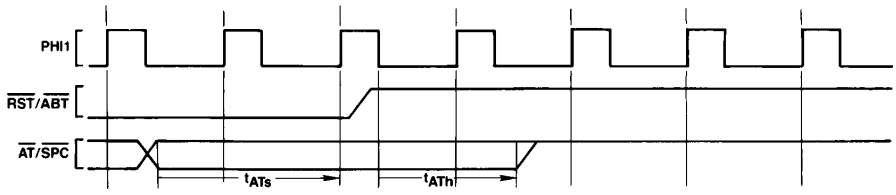
FIGURE 4-14. Slave Processor Read Timing



TL/EE/8525–81

FIGURE 4-15. $\overline{SPC}$ Timing

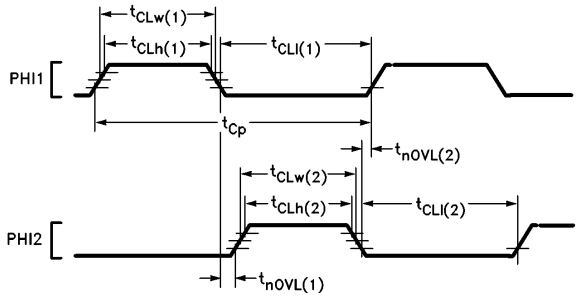
After transferring last operand to a Slave Processor, CPU turns OFF driver and holds $\overline{SPC}$ high with internal 5 kΩ pullup.



TL/EE/8525–53

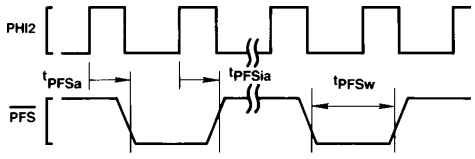
FIGURE 4-16. Reset Configuration Timing

4.0 Device Specifications (Continued)



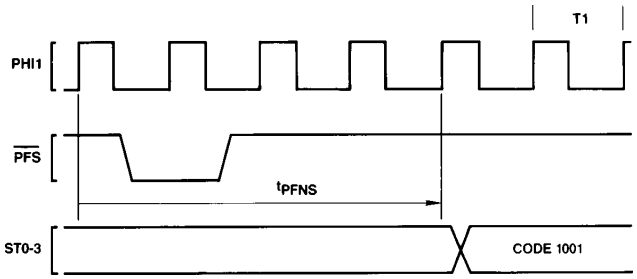
TL/EE/8525–54

FIGURE 4-17. Clock Waveforms



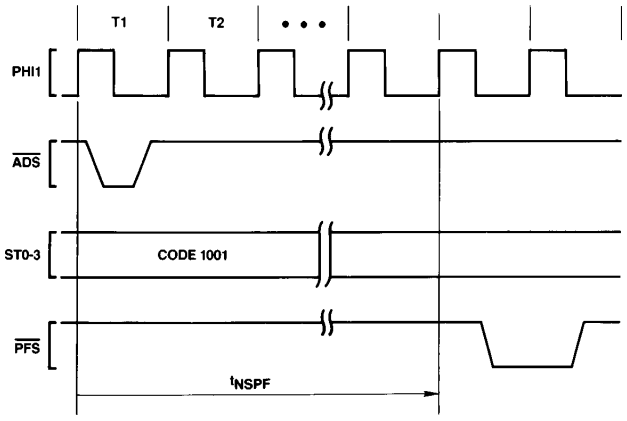
TL/EE/8525–55

FIGURE 4-18. Relationship of $\overline{PFS}$ to Clock Cycles



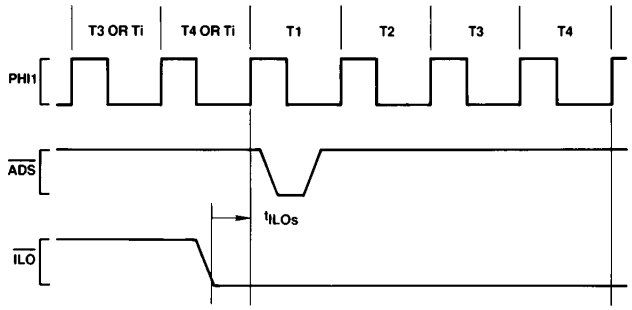
TL/EE/8525–56

FIGURE 4-19a. Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch



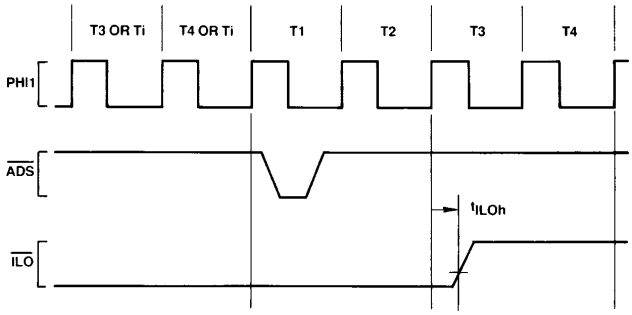
TL/EE/8525–57

FIGURE 4-19b. Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$



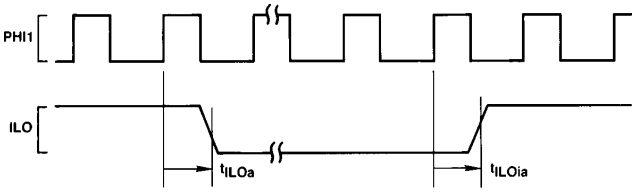
TL/EE/8525–58

FIGURE 4-20a. Relationship of $\overline{ILO}$ to First Operand Cycle of an Interlocked Instruction



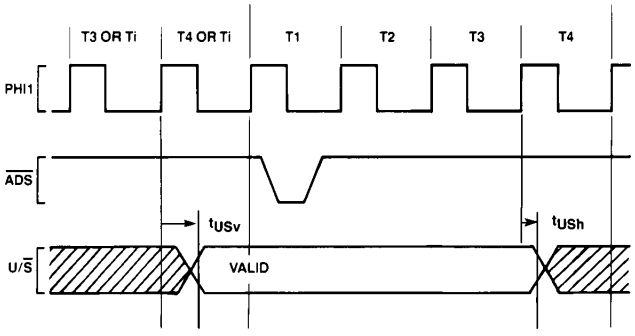
TL/EE/8525–59

FIGURE 4-20b. Relationship of $\overline{ILO}$ to Last Operand Cycle of an Interlocked Instruction



TL/EE/8525–60

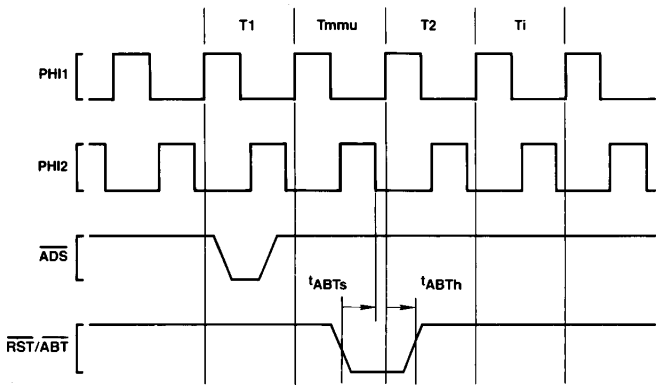
FIGURE 4-21. Relationship of $\overline{ILO}$ to Any Clock Cycle



TL/EE/8525–61

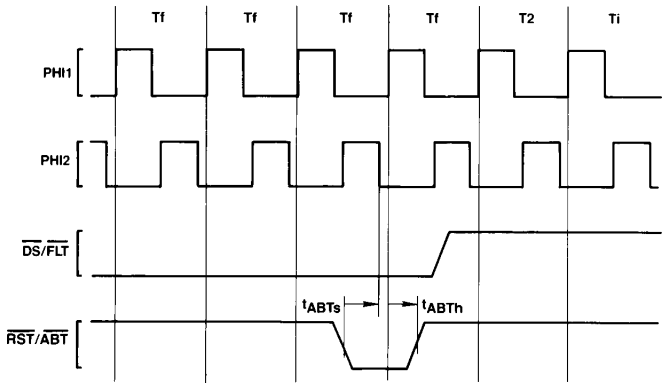
FIGURE 4-22. U/$\overline{S}$ Relationship to Any Bus Cycle— Guaranteed Valid Interval

4.0 Device Specifications (Continued)



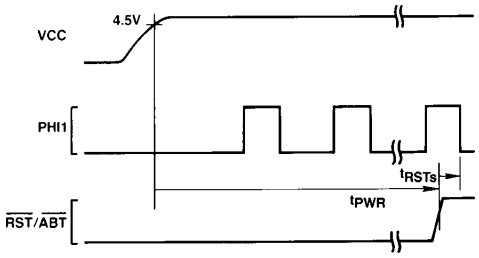
TL/EE/8525–62

FIGURE 4-23. Abort Timing, $\overline{FLT}$ Not Applied



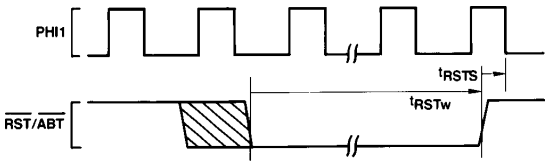
TL/EE/8525–63

FIGURE 4-24. Abort Timing, $\overline{FLT}$ Applied



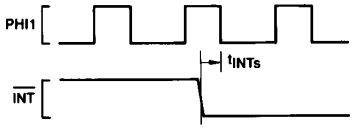
TL/EE/8525–64

FIGURE 4-25. Power-On Reset



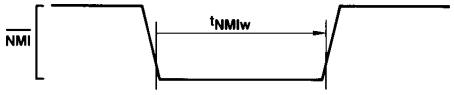
TL/EE/8525–65

FIGURE 4-26. Non-Power-On Reset



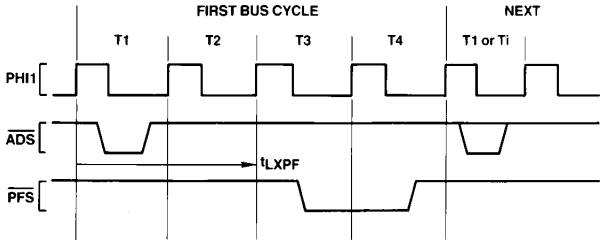
TL/EE/8525–66

FIGURE 4-27. $\overline{INT}$ Interrupt Signal Detection



TL/EE/8525–67

FIGURE 4-28. $\overline{NMI}$ Interrupt Signal Timing



TL/EE/8525–68

FIGURE 4-29. Relationship Between Last Data Transfer of an Instruction and $\overline{PFS}$ Pulse of Next Instruction

NOTE:

In a transfer of a Read-Modify-Write type operand, this is the Read transfer, displaying RMW Status (Code 1011).

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.

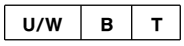
gen, gen 1, gen 2 = General Addressing Mode Field
- See Sec. 2.2 for encodings.

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)

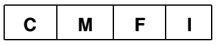
short = Short Immediate Value. May contain:
- quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.
- cond: Condition Code (above), in Scond.
- areg: CPU Dedicated Register, in LPR, SPR.
  - 0000 = US
  - 0001 – 0111 = (Reserved)
  - 1000 = FP
  - 1001 = SP
  - 1010 = SB
  - 1011 = (Reserved)
  - 1100 = (Reserved)
  - 1101 = PSR
  - 1110 = INTBASE
  - 1111 = MOD

Options: in String Instructions

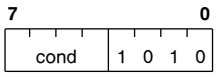


T = Translated
B = Backward
U/W = 00: None
01: While Match
11: Until Match

Configuration bits, in SETCFG:

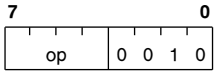


mreg: NS32082 Register number, in LMR, SMR.
- 0000 = BPR0
- 0001 = BPR1
- 0010 = (Reserved)
- 0011 = (Reserved)
- 0100 = (Reserved)
- 0101 = (Reserved)
- 0110 = (Reserved)
- 0111 = (Reserved)
- 1000 = (Reserved)
- 1001 = (Reserved)
- 1010 = MSR
- 1011 = BCNT
- 1100 = PTB0
- 1101 = PTB1
- 1110 = (Reserved)
- 1111 = EIA



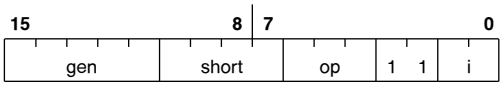
Format 0

Bcond (BR)



Format 1

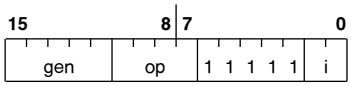
BSR	–0000	ENTER	–1000
RET	–0001	EXIT	–1001
CXP	–0010	NOP	–1010
RXP	–0011	WAIT	–1011
RETT	–0100	DIA	–1100
RETI	–0101	FLAG	–1101
SAVE	–0110	SVC	–1110
RESTORE	–0111	BPT	–1111



Format 2

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

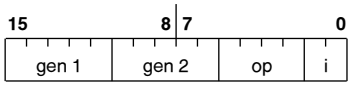
Appendix A: Instruction Formats (Continued)



Format 3

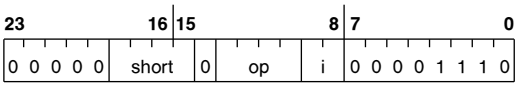
CXPD	−0000	ADJSP	−1010
BICPSR	−0010	JSR	−1100
JUMP	−0100	CASE	−1110
BISPSR	−0110		

Trap (UND) on XXX1, 1000



Format 4

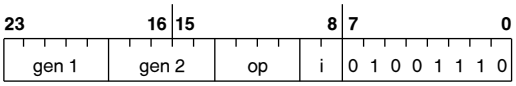
ADD	−0000	SUB	−1000
CMP	−0001	ADDR	−1001
BIC	−0010	AND	−1010
ADDC	−0100	SUBC	−1100
MOV	−0101	TBIT	−1101
OR	−0110	XOR	−1110



Format 5

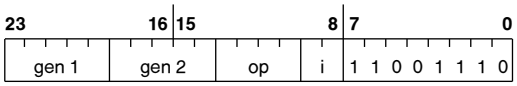
MOVS	−0000	SETCFG	−0010
CMPS	−0001	SKPS	−0011

Trap (UND) on 1XXX, 01XX



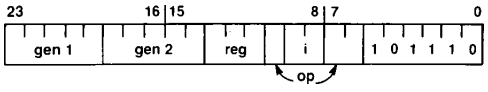
Format 6

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



Format 7

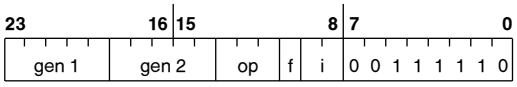
MOVM	−0000	MUL	−1000
CMPM	−0001	MEI	−1001
INSS	−0010	Trap (UND)	−1010
EXTS	−0011	DEI	−1011
MOVXBW	−0100	QUO	−1100
MOVZBW	−0101	REM	−1101
MOVZiD	−0110	MOD	−1110
MOVXiD	−0111	DIV	−1111



TL/EE/8525–69

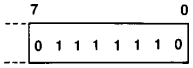
Format 8

EXT	−000	INDEX	−100
CVTP	−001	FFS	−101
INS	−010		
CHECK	−011		
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

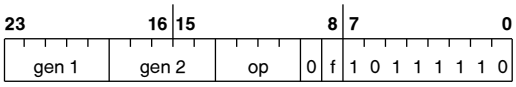


TL/EE/8525–70

Format 10

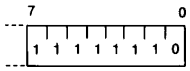
Trap (UND) Always

Appendix A: Instruction Formats (Continued)



Format 11

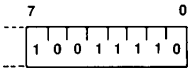
ADDf	−0000	DIVf	−1000
MOVf	−0001	Trap (SLAVE)	−1001
CMPf	−0010	Trap (UND)	−1010
Trap (SLAVE)	−0011	Trap (UND)	−1011
SUBf	−0100	MULf	−1100
NEGf	−0101	ABSf	−1101
Trap (UND)	−0110	Trap (UND)	−1110
Trap (UND)	−0111	Trap (UND)	−1111



TL/EE/8525–71

Format 12

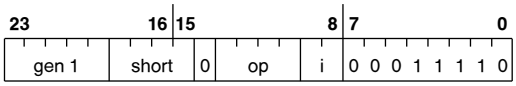
Trap (UND) Always



TL/EE/8525–72

Format 13

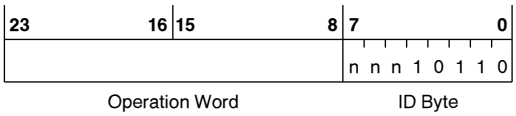
Trap (UND) Always



Format 14

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

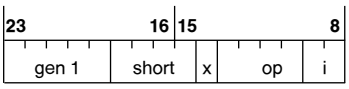
Trap (UND) on 01XX, 1XXX



Format 15
(Custom Slave)

nnn **Operation Word Format**

000

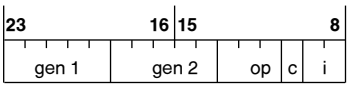


Format 15.0

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

Trap (UND) on all others

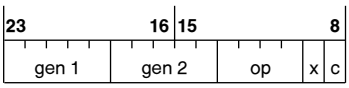
001



Format 15.1

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

101

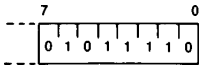


Format 15.5

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

If nnn = 010, 011, 100, 110, 111
then Trap (UND) Always

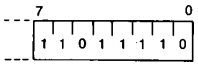
Appendix A: Instruction Formats (Continued)



TL/EE/8525–73

Format 16

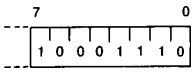
Trap (UND) Always



TL/EE/8525–75

Format 17

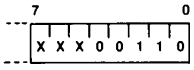
Trap (UND) Always



TL/EE/8525–76

Format 18

Trap (UND) Always

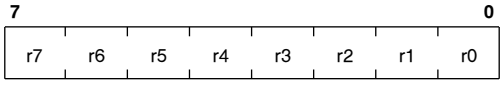


TL/EE/8525–74

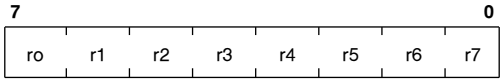
Format 19

Trap (UND) Always

Implied Immediate Encodings:



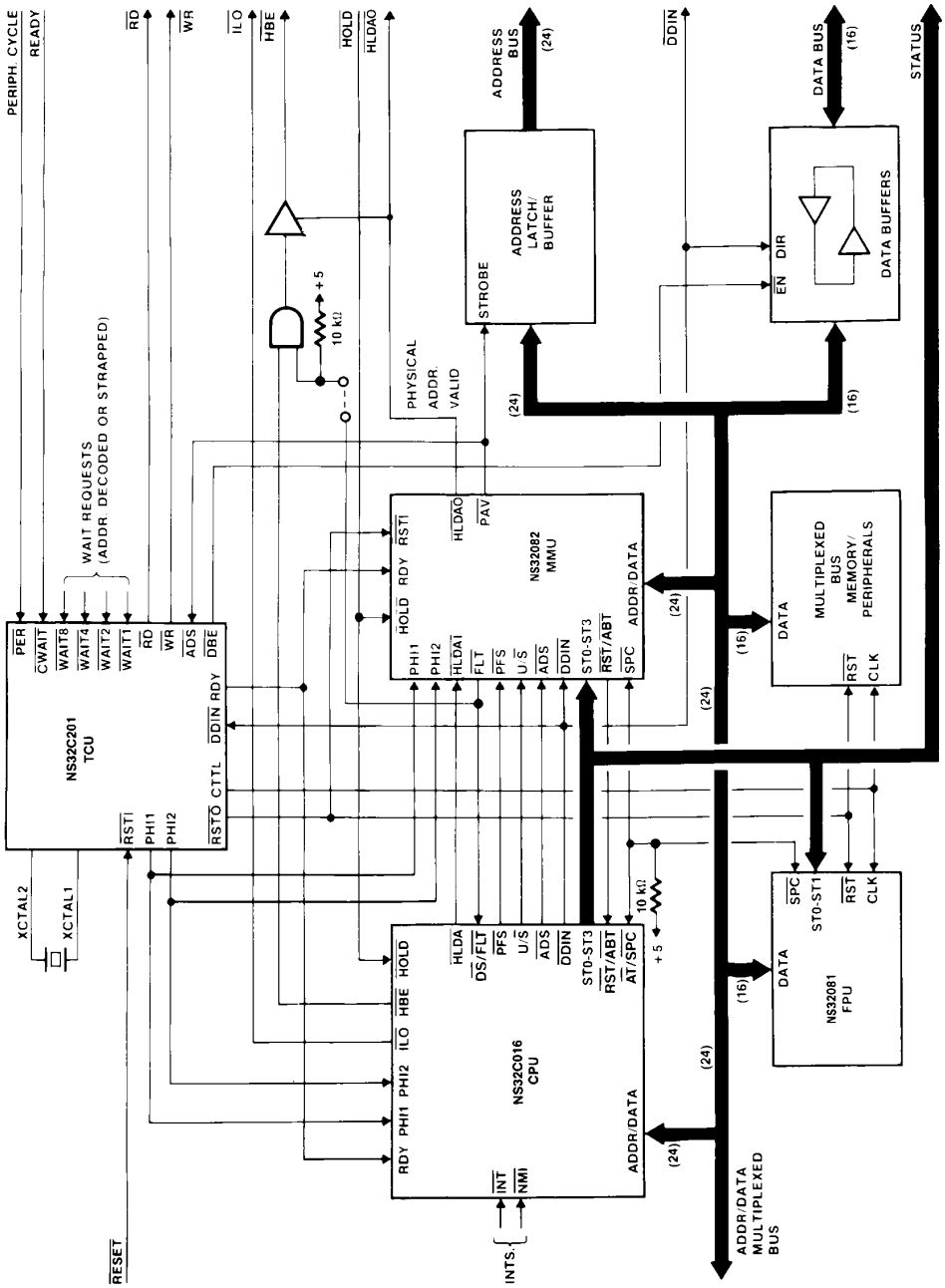
Register Mask, appended to SAVE, ENTER



Register Mask, appended to RESTORE, EXIT

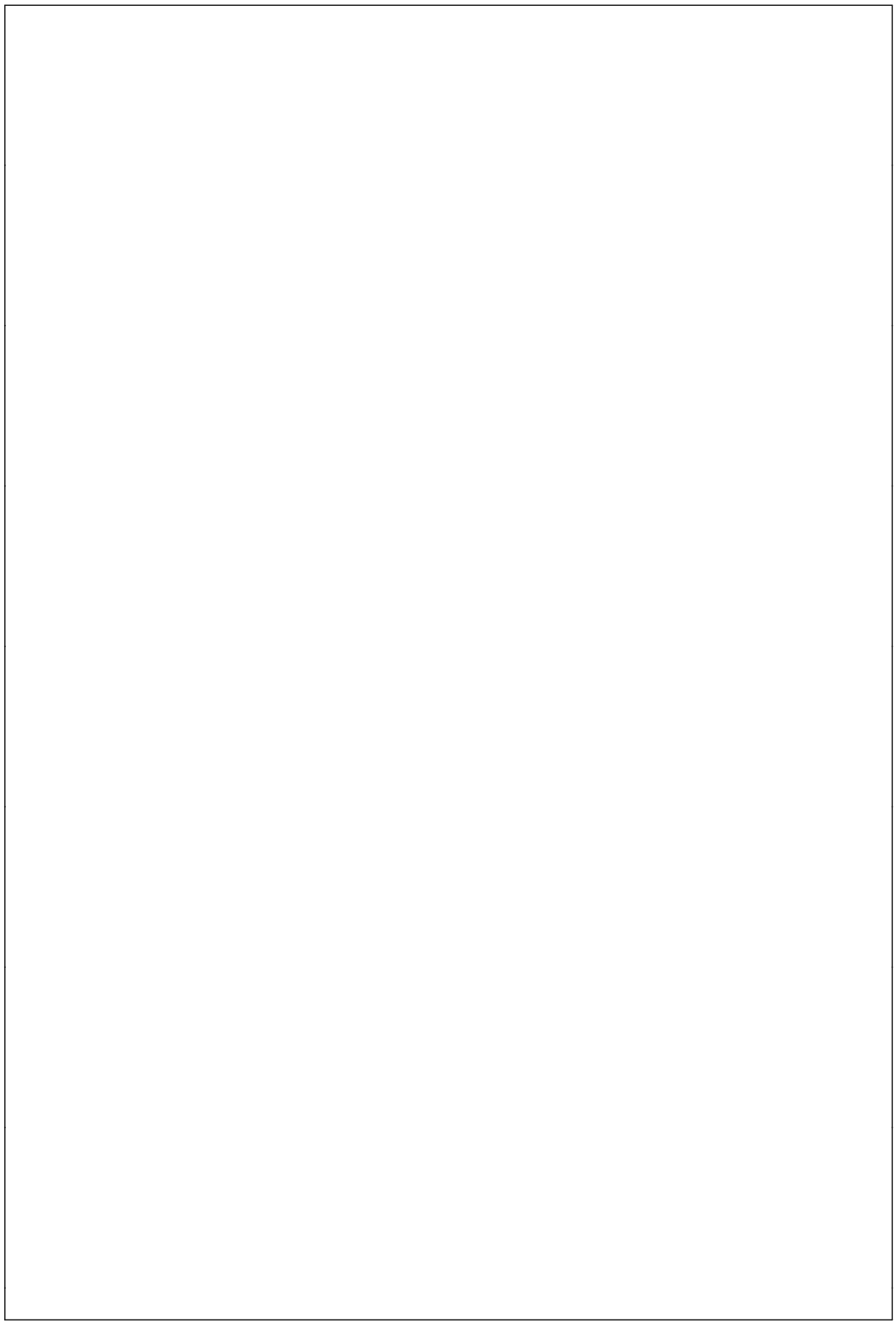
Offset/Length Modifier appended to INSS, EXTS

Appendix B: Interfacing Suggestions



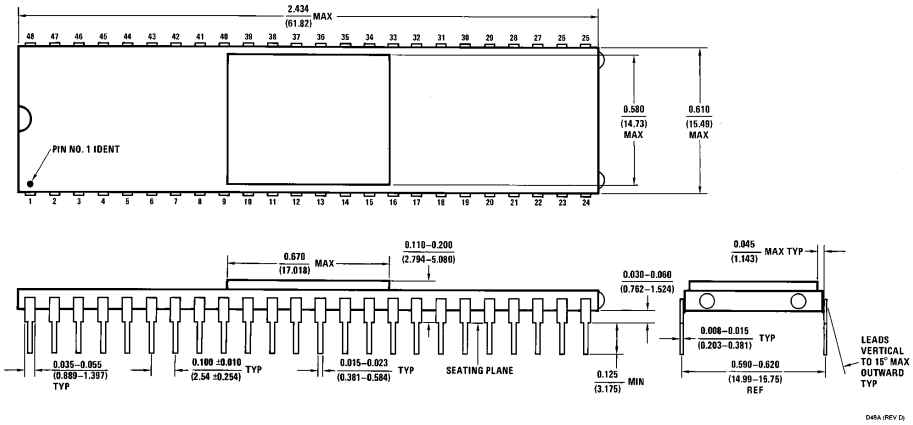
TL/EE/8525–77

FIGURE B-1. System Connection Diagram

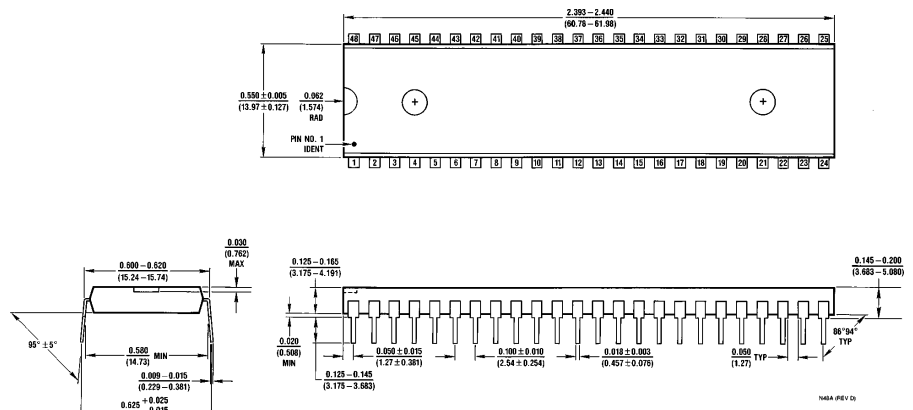


Physical Dimensions inches (millimeters)

Lit # 114273



Ceramic Dual-In-Line Package (D)
Order Number NS32C016D-10 or NS32C016D-15
NS Package Number D48A



Molded Dual-In-Line Package (N)
Order Number NS32C016N-10 or NS32C016N-15
NS Package Number N48A

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408