NS16000
DATABOOK

NATIONAL
SEMICONDUCTOR
CORPORATION

NS16000
DATABOOK

NATIONAL
SEMICONDUCTOR
CORPORATION

Introduction

This NS16000 Databook presents technical descriptions of the entire NS16000 family of 8-, 16- and 32-bit microprocessors, slave processors, peripherals, software and development tools. It is designed to be updated frequently so that our customers can have the latest technical information on the NS16000 family.

The NS16000 family leads the way in state-of-the-art microprocessor designs because of its advanced architecture, which includes:

- 32-Bit Architecture
- Demand Paged Virtual Memory
- Fast Floating-Point Capability
- High-Level Language Support
- Symmetrical Architecture

When we at National Semiconductor began the design of the NS16000 microprocessor family, we decided to take a radical departure from popular trends in architectural design–trends that date back more than a decade. We chose to take the time to design it properly.

Working from the top down, we analyzed the issues and anticipated the computing needs of the 80's and 90's. The result is an advanced and efficient family of microprocessor hardware and software products.

Clearly, software productivity has become a major issue in computer-related product development. In microprocessor-based systems this issue centers around the capability of the microprocessor to maximize the utility of software relative to shorter development cycles, improved software reliability and extended software life cycles.

In short, the degree to which the microprocessor can maximize software utility directly affects the cost of a product, its reliability, and time to market. It also eliminates future software modification for product enhancement or because of rapid advances in hardware technology.

Our approach has been to define an architecture addressing these software issues most effectively. The NS16000 combines 32-bit performance with efficient management of large address space. It facilitates high-level language program development and efficient instruction execution. Floating-point is integrated into the architecture.

This combination gives the user large system computing power at two orders of magnitude less cost.

But we didn't stop there. Advanced architecture isn't enough. Our top-down approach includes the hardware, software, and development support products necessary for your design. The evaluation board, in-system emulator, software development tools, including a VAX–11 cross-software package, and third party software are also available now for your evaluation and development.

The NS16000 family is a solid foundation from which National can build solutions for your future designs while satisfying your needs today.

Training

In addition to customer training on National's microprocessors, Starplex II™ and ISE™, training on the NS16000 family is now being conducted. This includes "The NS16000 Architecture", "ISE/16", "NSX-16 Cross-Software Support", "GENIX Cross Support" and "SYS16".

The NS16000 family development tools are thoroughly explained and demonstrated through lectures and lab exercises. Depending on the topic, these courses take from two to five days.

National's Training Center is located in San Jose, California, about forty miles south of San Francisco International Airport, and only ten minutes from San Jose Airport. Upon request, National will conduct on-site customer training.

Service

The Service Organization offers three levels of technical support for the Microcomputer Systems Division's products:

1 The Response Center utilizes SPIRE, a computerized technical data base designed for rapid search, to solve customer and technical problems. Depot repair services are available for board and system products. Our customers can use our toll-free numbers to contact the Response Center for immediate solutions. (800) 538-1866 (California only) or (800) 672-1811.

2 When indicated, the Response Center will utilize our Application System Engineers who have in-depth product knowledge for dealing with application-oriented issues (both hardware and software) to help solve customer design problems. The Application System Engineers are supported by engineering and manufacturring resources.

3 National's Field Engineers are located in various cities in the United States and Canada, and are available for dispatch to customer sites to repair our development systems products. Each field engineering location maintains an extensive spare parts inventory.

Microcomputer Systems Division

The Microcomputer Systems Division's goal is to become a leading force in the microcomputer systems marketplace.

To achieve this goal, a total systems approach has been taken on the NS16000 program to provide the customer with the necessary hardware and software support, evaluation and development tools, training, service and technical literature.

The focus is on upward migration paths, system integration at all levels and the preservation of the user's software investment.

Four groups (Microprocessors, OEM Board Level Products, Software Products and Development Systems) offer a broad capability to solve customer needs at various levels of performance and integration.

Quality and Reliability

As electronic systems become more and more complex, the need for consistently high quality integrated circuits becomes increasingly important. Having recognized this need as far back as the 1970's, National Semiconductor initiated a unique, company-wide Quality Improvement Program. The results have been dramatic and, we believe, unmatched in this industry. Over the years, National has regularly been named by many major customers as "Quality Manufacturer of the Year". We are proud of our success, which sets a standard for others to achieve. And yet our quest for perfection is ongoing, so that customers can continue to rely on National Semiconductor integrated circuits and products in their system designs.

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 does not assume any responsibility for use of any circuitry described; no circuit patent licenses are implied, and National reserves the right, at any time without notice, to change said circuitry and specifications.

Information contained herein is intended to be a general product description and is subject to change.

Table of Contents

CPUs

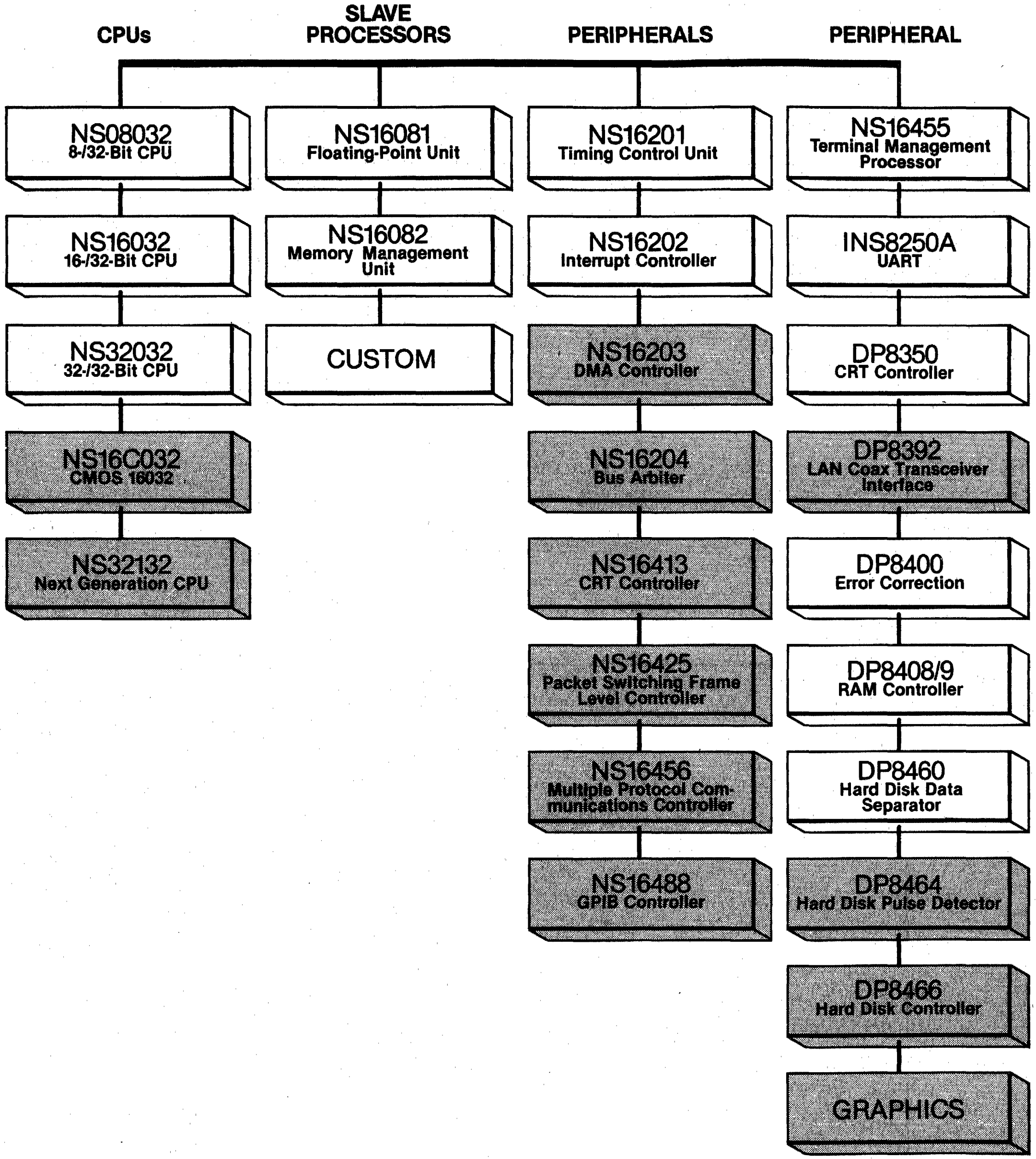
Slave Processors

Peripherals

Development Tools

Software

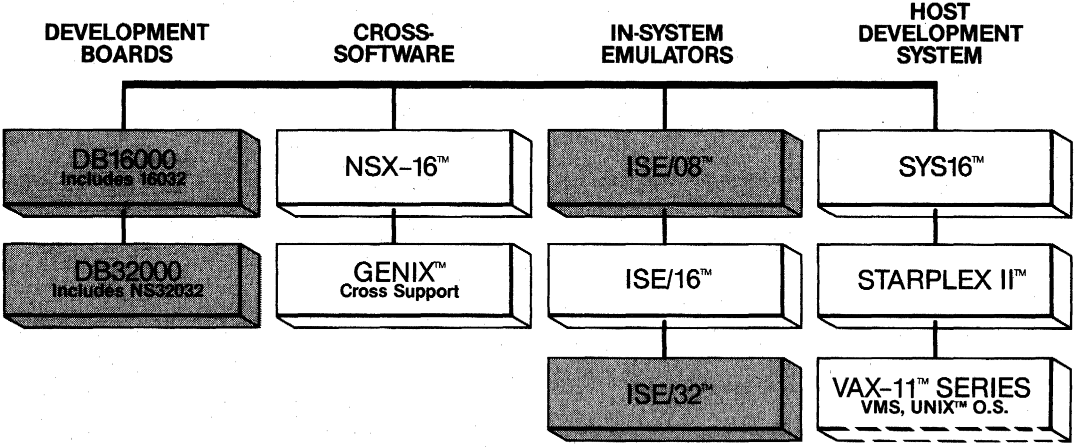
NS16000 Microprocessor Family



CPUs	SLAVE PROCESSORS	PERIPHERALS	PERIPHERAL
NS08032 8-/32-Bit CPU	NS16081 Floating-Point Unit	NS16201 Timing Control Unit	NS16455 Terminal Management Processor
NS16032 16-/32-Bit CPU	NS16082 Memory Management Unit	NS16202 Interrupt Controller	INS8250A UART
NS32032 32-/32-Bit CPU	CUSTOM	▨ NS16203 DMA Controller	DP8350 CRT Controller
▨ NS16C032 CMOS 16032		▨ NS16204 Bus Arbiter	▨ DP8392 LAN Coax Transceiver Interface
▨ NS32132 Next Generation CPU		▨ NS16413 CRT Controller	DP8400 Error Correction
		▨ NS16425 Packet Switching Frame Level Controller	DP8408/9 RAM Controller
		▨ NS16456 Multiple Protocol Communications Controller	DP8460 Hard Disk Data Separator
		▨ NS16488 GPIB Controller	▨ DP8464 Hard Disk Pulse Detector
			▨ DP8466 Hard Disk Controller
			▨ GRAPHICS

Note: Products in the Shaded boxes are additional hardware components planned to support the NS16000 CPUs. Please contact your local National Sales Office for further information on their availability.

NS16000 Development Tools



Note: Products in the Shaded boxes are additional hardware components planned to support the NS16000 CPUs. Please contact your local National Sales Office for further information on their availability.

CPUs

National Semiconductor

PRELIMINARY

NS08032-6, NS08032-4 High-Performance 8-Bit Microprocessors

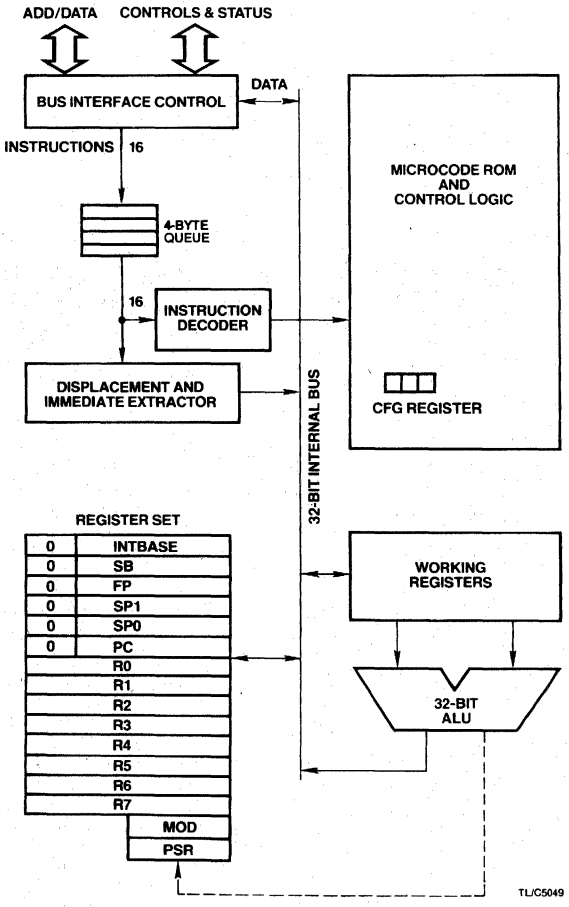
General Description

The NS08032 functions as a Central Processing Unit (CPU) in National Semiconductor's NS16000 microcomputer family. It has been designed to optimally support microprocessor users who need the ability to use a large addressing space for large programs and/or large data structures. Because large programs must realistically be generated and maintained in high-level languages, the NS16000 architecture provides for very efficient compilation while remaining easy to program at the assembler level for optimizations. The NS08032 represents an implementation of this architecture for 8-bit systems. High-performance Floating-Point instructions are provided with the NS16081 Floating-Point Unit (FPU). The NS08032-4 and NS08032-6 have different timing parameters. Refer to Section 4 for timing specifications.

Features

- 32-Bit Architecture and Implementation
- 8-bit Bus for Low System Cost
- 16-MByte Uniform Addressing Space
- Powerful Instruction Set
  - — General Two-Address Capability
  - — Very High Degree of Symmetry
  - — Addressing Modes Optimized for High-Level Language References
  - — Expansion via Slave Processors or Traps
- High-Speed XMOS Technology
- Single 5V Supply
- 48-Pin Dual-In-Line Package

NS08032-4/-6 CPU Block Diagram



Absolute Maximum Ratings

Temperature under bias	0 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 DC Electrical Characteristics.*

DC Electrical Characteristics:

NS08032-4 $T_A = 0$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, GND = 0V
NS08032-6 $T_A = 0$ to $+50°C$, $V_{CC} = 5V \pm 5\%$, GND = 0V

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{IH}	Logical 1 Input Voltage		2.0		V_{CC}+0.5	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	V_{CC}−0.5		V_{CC}+0.5	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CLT}	Logical 0 Clock Voltage, Transient (ringing tolerance)	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	Logical 1 Output Voltage	$I_{OH} = -400\mu A$	2.4			V
V_{OL}	Logical 0 Output Voltage	$I_{OL} = 2mA$			0.45	V
I_{ILS}	$\overline{\text{SPC}}$ Input Current (low)	$V_{IN} = 0.4V$, $\overline{\text{SPC}}$ in input mode	0.05		1.0	mA
I_I	Input Leakage Current	$0 \leqslant V_{IN} \leqslant V_{CC}$, All inputs except PHI1, PHI2, $\overline{\text{SPC}}$	−10		10	μA
$I_{O(OFF)}$	Output Leakage Current	$0.4 \leqslant V_{OUT} \leqslant V_{CC}$	−20		20	μA
I_{CC}	Active Supply Current	$I_{OUT} = 0$, $T_A = 25°C$		180	300	mA

1 NS08032 Pin Descriptions

The following is a brief description of all NS08032 pins. The descriptions reference portions of the Functional Description, Section 3.

1.1 Supplies

Power (V_{CC}): +5V Positive Supply. Section 3.1.

Logical 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.

Back-Bias Generator (BBG): Output of on-chip substrate voltage generator. Section 3.1.

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.

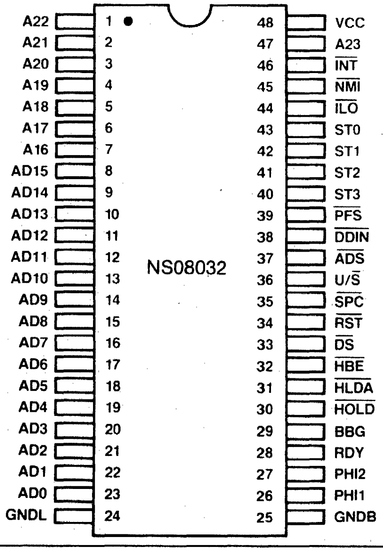
Hold Request ($\overline{\text{HOLD}}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Section 3.5.

Interrupt ($\overline{\text{INT}}$): Active low. Maskable Interrupt Request. Section 3.7.

Non-Maskable Interrupt ($\overline{\text{NMI}}$): Active low. Non-Maskable Interrupt Request. Section 3.7.

Reset ($\overline{\text{RST}}$): Active low. It initiates a Reset. Section 3.3.

Connection Diagram



TL/C5049

1.3 Output Signals

Address Bits 16-23 (A16-A23): Active high. These are the most significant eight 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.

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 Ack., Master
0101—Interrupt Ack., Cascaded
0110—End of Interrupt, Master
0111—End of Interrupt, Cascaded
1000—Sequential Instruction Fetch
1001—Nonsequential Instruction Fetch
1010—Data Transfer
1011—Read RMW 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 3.5.

User/Supervisor (U/$\overline{\text{S}}$): User or Supervisor Mode status. High state indicates User Mode, low indicates Supervisor Mode. Section 3.6.

Interlocked Operation ($\overline{\text{ILO}}$): Active low. Indicates that an interlocked instruction is being executed. Section 3.6.

Program Flow Status ($\overline{\text{PFS}}$): Active low. Pulse indicates beginning of an instruction execution. Section 3.6.

Data Strobe ($\overline{\text{DS}}$): Active low. Data strobe output. Section 3.4.

1.4 Input-Output Signals

Address/Data 0-15 (AD0-AD15): Active high. In all except Slave Processor bus cycles, pins AD0-AD7 serve as an 8-bit Multiplexed Address/Data bus, and pins AD8-AD15 hold address bits 8-15 throughout the bus cycle. Bit 0 is defined as the least-significant bit. Section 3.4.

In Slave Processor bus cycles, all 16 pins are used as a data bus (Section 3.4.6).

Slave Processor Control ($\overline{\text{SPC}}$): Active low. Used by the CPU as the data strobe output for Slave Processor transfers; used by Slave Processors to acknowledge completion of an instruction. Section 3.4.6 and Section 3.8.

Data Strobe: Active low. Data Strobe output. Section 3.4.

2 Architectural Description

2.1 Programming Model

The NS16000 architecture includes 16 registers on the NS08032 CPU. *Figure 2-1* shows the NS08032 registers.

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 32 bits in length. If a general register is specified for an operand which is 8 or 16 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 NS08032 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 NS08032, 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 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.

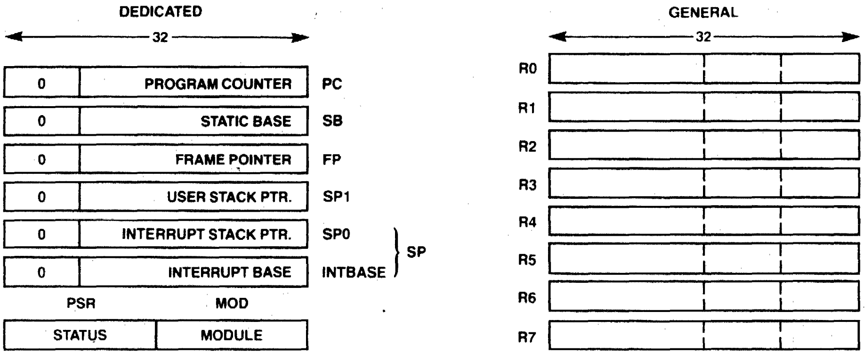
In this document, reference is made to the SP register. The terms "SP register" or "SP" refer to either SP0 or SP1, depending on the setting of the S bit in the PSR register. If the S bit in the PSR is 0, then SP refers to SP0. If the S bit in the PSR is 1, then SP refers to SP1. (In the NS08032, the upper eight bits of these registers are always zero.)

Stacks in the NS16000 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 lowest address in memory occupied by the old contents of the frame pointer. (In the NS08032, 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 NS08032, the upper eight bits of this register are always zero.)



TL/C5049

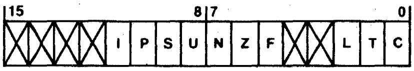
FIGURE 2-1. The General and Dedicated Registers

INTBASE: The INTERRUPT BASE register holds the address of the dispatch table for interrupts and traps (Section 3.7). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS08032, 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 16 bits long, therefore the module table must be contained within the first 64k bytes of memory.

PSR: The PROCESSOR STATUS REGISTER holds the status codes for the NS08032 microprocessor.

The PSR is 16 bits long, divided into two 8-bit halves (*Figure 2-2*). 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/C5049

FIGURE 2-2. The Processor Status Register

C: The C bit indicates that a carry or borrow occurred after an addition or subtraction instruction. It is used in the calculation of multiple precision numbers. 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 3.7.5).

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 NS08032 is said to be in Supervisor Mode; when U=1, the NS08032 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.

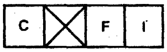
S: The S bit specifies whether the SP0 register or SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

P: The P bit prevents a TRC trap from occurring more than once for an instruction (Section 3.7.5). 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 3.7). If I=0, only the $\overline{\text{NMI}}$ interrupt is accepted. Trap enables are not affected by this bit.

2.1.3 The Configuration Register (CFG)

Within the Control section of the NS08032 CPU is the 4-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*.



TL/C5049

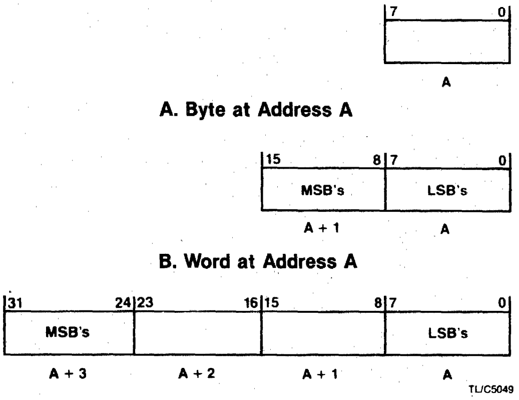
FIGURE 2-3. CFG Register

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS16202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the INT pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Section 3.7.

The F and C bits declare the presence of the FPU 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 NS08032 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 (*Figure 2-4A*). 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.



A. Byte at Address A

B. Word at Address A

C. Double Word at Address A

FIGURE 2-4. Data Formats for NS08032 Memory

Two contiguous bytes are called a word (*Figure 2-4B*). 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.

Two contiguous words are called a double word (*Figure 2-4C*). 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.

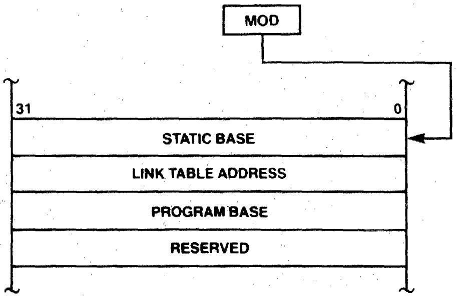
2.1.5 Dedicated Tables

Two of the NS08032 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 3.7.

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 NS08032. At any point in time, 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 2-5*. The Static Base entry contains the address of static data which is 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/C5049

FIGURE 2-5. Module Descriptor Format

The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information which is 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 2-6*. 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 NS16000 Programmer's Reference Manual.

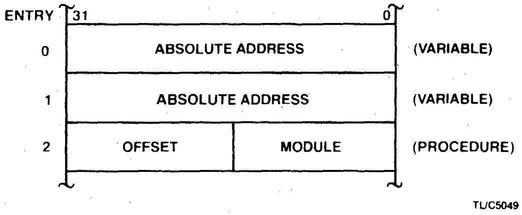


FIGURE 2-6. A Sample Link Table

2.2 Instruction Set

2.2.1 General Instruction Format

Figure 2-7 shows the general format of an NS16000 instruction. The Basic Instruction is one to three bytes long and contains the opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions which may appear, depending on the instruction and the addressing modes selected.

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

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Displacement/Immediate field may contain one or 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 2-9*, 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 backward from the usual 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).

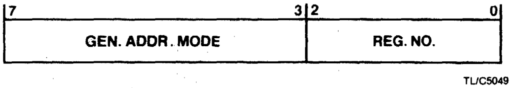


FIGURE 2-8. Index Byte Format

2.2.2 Addressing Modes

The NS08032 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 NS08032 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 which acts upon that variable. Extraneous data movement is therefore minimized.

NS08032 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 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.

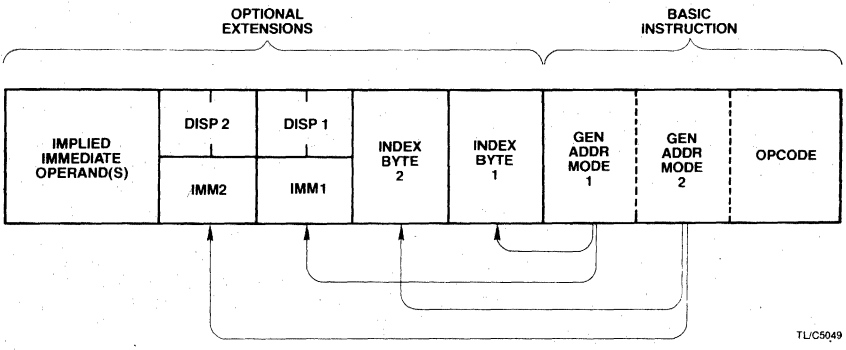
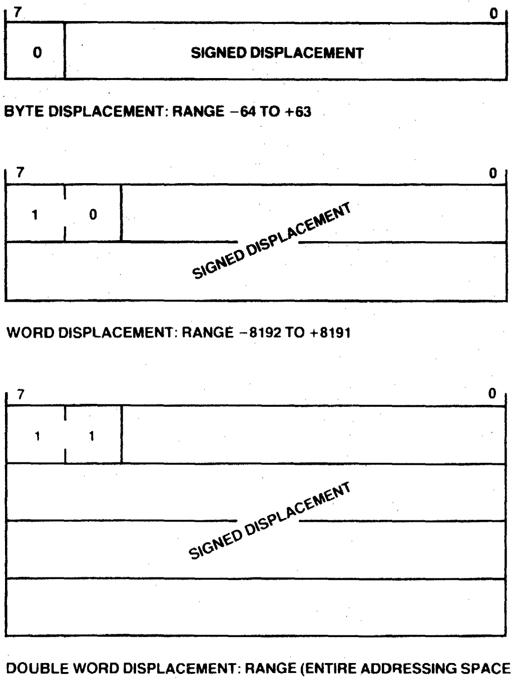


FIGURE 2-7. General Instruction Format



BYTE DISPLACEMENT: RANGE −64 TO +63

WORD DISPLACEMENT: RANGE −8192 TO +8191

DOUBLE WORD DISPLACEMENT: RANGE (ENTIRE ADDRESSING SPACE)

TL/C5049

FIGURE 2-9. Displacement Encodings

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

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

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

The following table, Table 2-1, is a brief summary of the addressing modes. For a complete description of their actions, see the Programmer's Reference Manual.

2.2.3 Instruction Set Summary

This section presents a brief description of the NS08032 instruction set. The instructions are functionally grouped in Table 2-2. The Format column of each table is a reference 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 Programmer's 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.

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

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

reg = Any General Purpose Register: R0–R7.

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

creg = A Custom Slave Processor Register (implementation dependent)

cond = Any condition code, encoded as a 4-bit field within the Basic Instruction.

Table 2-1. NS08032 Addressing Modes

Encoding	Mode	Assembler Syntax	Effective Address
Register			
00000	Register 0	R0 or F0	None: Operand is in the specified register.
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
Register Relative			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
Memory 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(PC)	
Memory Relative			
10000	Frame memory relative	disp2(disp1(FP))	Disp2 + Pointer; Pointer found at address Disp1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1(SP))	
10010	Static memory relative	disp2(disp1(SB))	
Immediate			
10100	Immediate	value	None: Operand is input from instruction queue.
Absolute			
10101	Absolute	@disp	Disp.
External			
10110	External	EXTERNAL (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.
Scaled Index			
11100	Index, bytes	mode[Rn:B]	Mode + Rn.
11101	Index, words	mode[Rn:W]	Mode + 2 × Rn.
11110	Index, double words	mode[Rn:D]	Mode + 4 × Rn.
11111	Index, quad words	mode[Rn:Q]	Mode + 8 × Rn. "Mode" and "n" are contained within the Index Byte.
10011	(Reserved for Future Use)		

Table 2-2. Instruction Set Summary

Format	Instruction		Description
Moves			
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a 4-bit constant.
7	MOVMi	gen,gen,disp	Move Multiple: disp bytes.
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			
4	ADDi	gen,gen	Add.
2	ADDQi	short,gen	Add 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)			
6	ADDPi	gen,gen	Add Packed.
6	SUBPi	gen,gen	Subtract Packed.
Integer Comparison			
4	CMPi	gen,gen	Compare.
2	CMPQi	short,gen	Compare to 4-bit constant.
7	CMPMi	gen,gen,disp	Compare Multiple: disp bytes.
Logical and Boolean			
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			
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			
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.

Table 2-2. **Instruction Set Summary** (Cont'd)

Format	Instruction		Description
Bit Fields (Note 1)			
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			
8	CHECKi	reg,gen,gen	Index bounds check.
8	INDEXi	reg,gen,gen	Recursive indexing step for multiple-dimensional arrays.
Strings (Note 2)			
5	MOVSi	options	Move String 1 to String 2.
5	MOVST	options	Move string, translating bytes.
5	CMPSi	options	Compare String 1 to String 2.
5	CMPST	options	Compare, translating String 1 bytes.
5	SKPSi	options	Skip over String 1 entries.
5	SKPST	options	Skip, translating bytes for Until/While.
Jumps and Linkage			
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 nonzero.
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			
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)

Table 2-2. Instruction Set Summary (Cont'd)

Format	Instruction		Description
Floating Point			
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.
Miscellaneous			
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			
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	CCMPc	gen,gen	Custom Compare.
15.1	CCV0ci	gen,gen	Custom Convert.
15.1	CCV1ci	gen,gen	
15.1	CCV2ci	gen,gen	
15.1	CCV3ci	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	
15.0	LCR	creg,gen	Load Custom Register. (Privileged)
15.0	SCR	creg,gen	Store Custom Register. (Privileged)

Note 1: Bit fields are values in memory which 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.

Note 2: String instructions assign specific functions to the General Purpose Registers:

R4—Comparison Value
R3—Translation Table Pointer
R2—String 2 Pointer
R1—String 1 Pointer
R0—Limit Count

Options on all string instructions are:

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

All string instructions end when R0 decrements to zero.

3 Functional Description

3.1 Power and Grounding

The NS08032 requires a single 5V power supply, applied on pin 48 (V_{CC}). See DC Electrical Characteristics.

Grounding connections are made on two pins. Logic Ground (GNDL, pin 24) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 25) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below (*Figure 3-1*).

In addition to V_{CC} and Ground, the NS08032 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (*Figure 3-1*) from the BBG pin to ground. Recommended values for these are:

C_1: 1μF, Tantalum.

C_2: 1000pF, low inductance. This should be either a disc or monolithic ceramic capacitor.

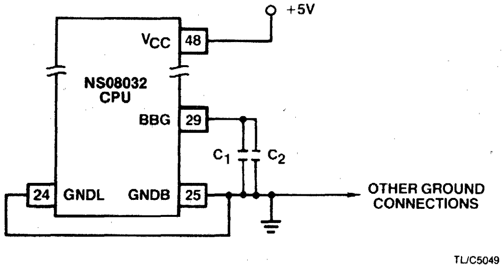


FIGURE 3-1. Recommended Supply Connections

3.2 Clocking

The NS08032 inputs clocking signals from the NS16201 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 positive 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 the AC Electrical Characteristics (Section 4) for complete specifications of PHI1 and PHI2.

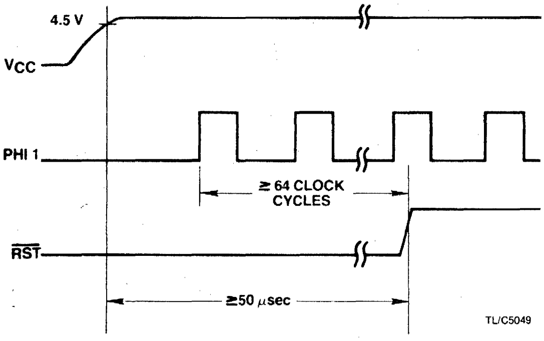


FIGURE 3-3. Power-On Reset Requirements

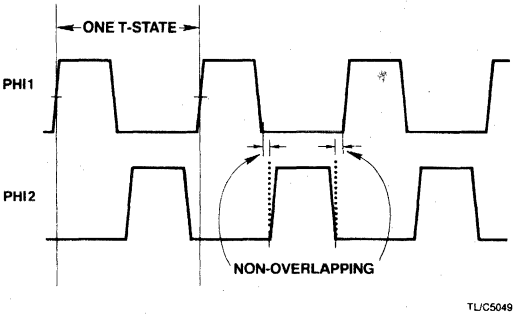


FIGURE 3-2. Clock Timing Relationships

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. A TTL clock signal (CTTL) is provided by the TCU for all other clocking.

3.3 Resetting

The $\overline{RST}$ pin serves as a reset for on-chip logic. The CPU may be reset at any time by pulling the $\overline{RST}$ 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, $\overline{RST}$ must be held low for at least 50μs 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 for not less than 64 clock cycles. The trailing (positive-going) edge must occur while PHI1 is high, and no later than 10ns before the PHI1 trailing edge. See *Figures 3-3* and *3-4*.

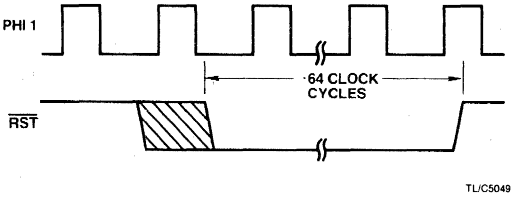


FIGURE 3-4. General Reset Timing

The NS16201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS08032 CPU. *Figure 3-5* shows the recommended connections.

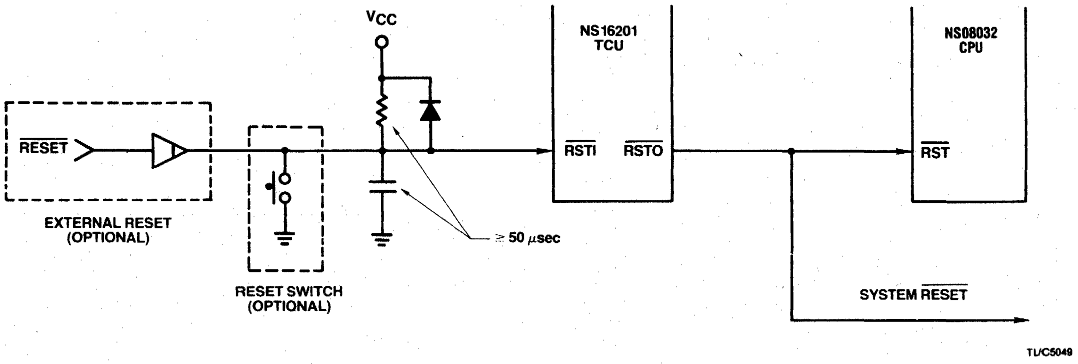


FIGURE 3-5. Recommended Reset Connections

3.4 Bus Cycles

The NS08032 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 NS16000 family.
2. To fetch instructions into the 4-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 4-bit code placed on the Bus Status pins (ST0–ST3). Slave Processor cycles differ in that separate control signals are applied and transfers are performed 16 bits at a time (Section 3.4.6).

Figure 3-6 shows typical bus connections for the NS08032. The address, data, and control signals referenced in the following discussion are shown in this figure.

The sequence of events in a non-Slave Processor 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).

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 $\overline{\mathrm{ADS}}$ 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 0-7 from the AD0-AD7 pins. See *Figure 3-6*. Also during this time the status signal $\overline{\mathrm{DDIN}}$, indicating the direction of the transfer, becomes valid.

During T2, the CPU switches the Data Bus, AD0-AD7, to either accept or present data. Note that the signals AD8-AD15 and A16-A23 remain valid, and need not be latched. It also starts the Data Strobe ($\overline{\mathrm{DS}}$), signaling the beginning of the data transfer. Associated signals from the NS16201 Timing Control Unit are also activated at this time: $\overline{\mathrm{RD}}$ (Read Strobe) or $\overline{\mathrm{WR}}$ (Write Strobe), $\overline{\mathrm{TSO}}$ (Timing State Output, indicating that T2 has been reached) and $\overline{\mathrm{DBE}}$ (Data Buffer Enable).

The T3 state provides for access time requirements, and it occurs at least once in a bus cycle. At the beginning of T3, on the rising edge of the PHI1 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-AD7) is sampled at the falling edge of PHI2 of the last T3 state. See Timing Specification, Section 4. Data must, however, be held at least until the beginning of T4. $\overline{\mathrm{DS}}$ and $\overline{\mathrm{RD}}$ 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 $\overline{\mathrm{DS}}$, $\overline{\mathrm{RD}}$ or $\overline{\mathrm{WR}}$, and $\overline{\mathrm{TSO}}$ signals go inactive, and at the rising edge of PHI2, $\overline{\mathrm{DBE}}$ goes inactive, having provided for necessary data hold times. Addresses (and Data during Write cycles) remain 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).

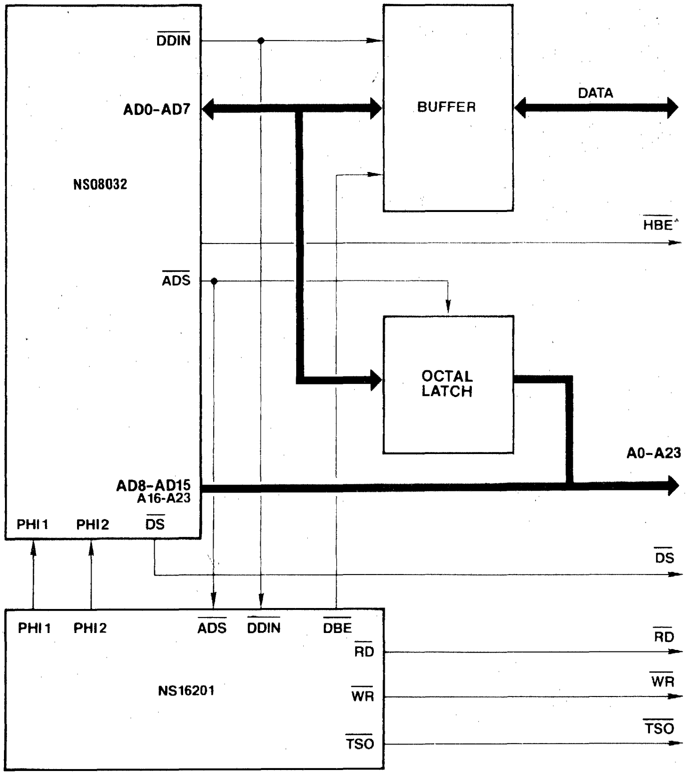


FIGURE 3-6. Bus Connections

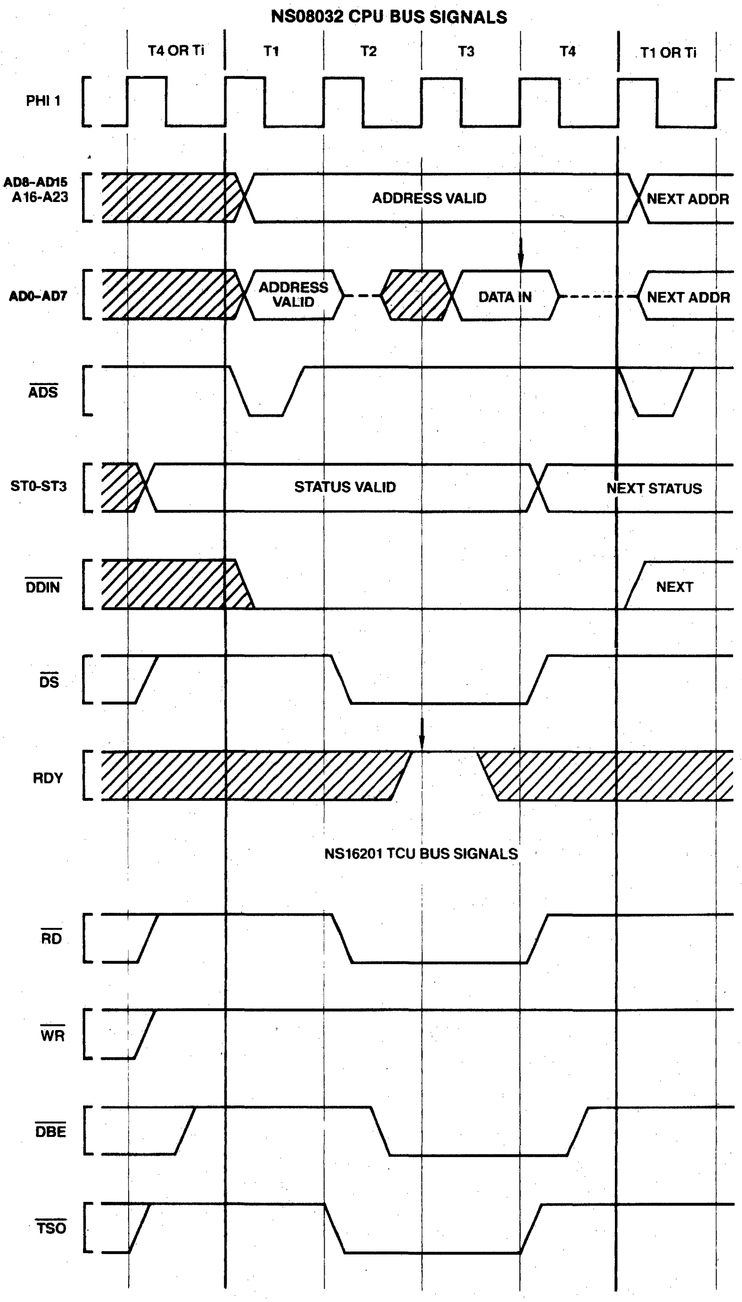
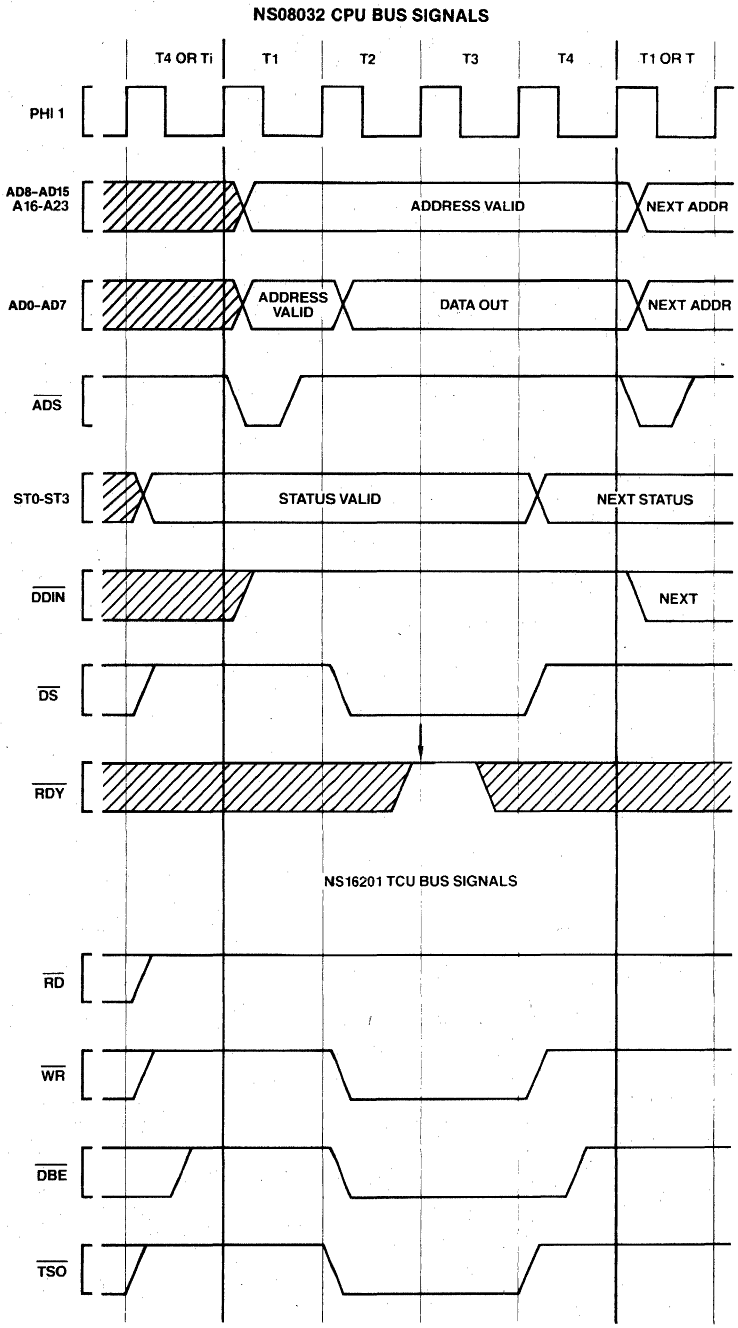


FIGURE 3-7. Read Cycle Timing



TL/C5049

FIGURE 3-8. Write Cycle Timing

3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS08032 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.

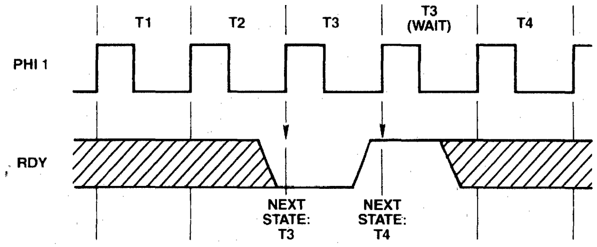
At the beginning of T3, on the rising edge of PHI1, the RDY line is sampled by the CPU. If RDY is high, the next T-State will be T4, ending the bus cycle. If it is sampled low, then the next T-State will be another T3, and the RDY line will again be sampled on PHI1. Each additional T3 state after the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

The RDY pin is driven by the NS16201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pins:

1. $\overline{\text{CWAIT}}$ (Continuous WAIT), which holds the CPU in WAIT States until removed.
2. $\overline{\text{WAIT1}}$, $\overline{\text{WAIT2}}$, $\overline{\text{WAIT4}}$, $\overline{\text{WAIT8}}$ (collectively, $\overline{\text{WAITn}}$), which may be given a 4-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 on their use, see the NS16201 Data Sheet.

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



TL/C5049

FIGURE 3-9. RDY Pin Timing

3.4.2 Bus Status

The NS08032 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, 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 4-bit value, with ST0 the least significant bit. Their values decode as follows:

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

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.

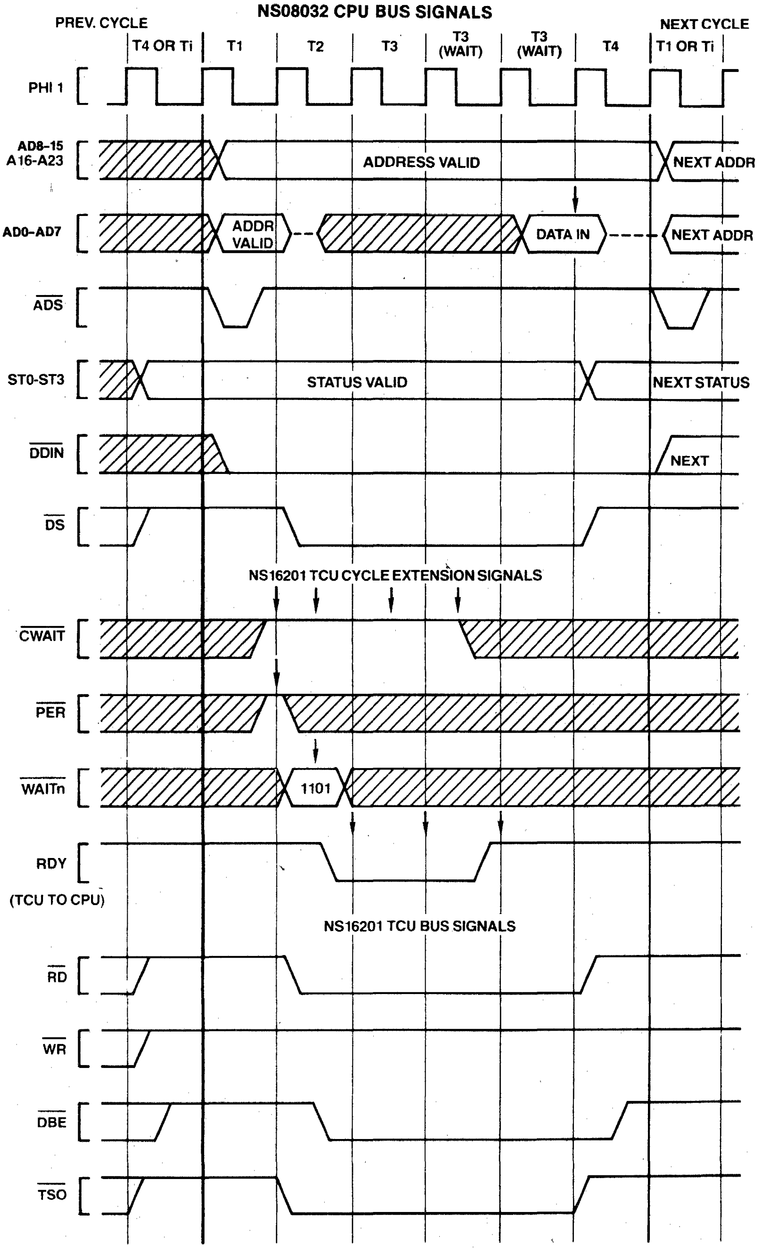
0100 Interrupt Acknowledge, Master.
The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{\text{NMI}}$), it will read from address FFFF00$_{16}$, but will ignore any data provided. To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$), it will read from address FFFE00$_{16}$, expecting a vector number to be provided from the Master NS16202 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 NS16202 is present. See Section 3.4.5.

0101 Interrupt Acknowledge, Cascaded.
The CPU is reading a vector number from a Cascaded NS16202 Interrupt Control Unit. The address provided is the address of the NS16202 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.



TL/C5049

Note: Arrows on $\overline{\text{CWAIT}}$, $\overline{\text{PER}}$, $\overline{\text{WAITn}}$ indicate points at which the TCU samples. Arrows on AD0–AD7 and RDY indicate points at which the CPU samples.

FIGURE 3-10. Extended Cycle Example

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.

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 3.8.1

1110 Read Slave Processor Status.
The CPU is reading a status word from a Slave Processor. This occurs after the Slave Processor has signaled 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 3.8.1.

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 3.8.1.

3.4.3 Data Access Sequences

The NS08032 accesses all memory and peripheral devices in sequences of single-byte transfers. Transfer of values larger than bytes is performed from least-significant byte (lowest address) to most-significant byte.

3.4.3.1 Bit Accesses

The bit instructions access 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 starting 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 operands which 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.

3.4.4 Instruction Fetches

Instructions for the NS08032 CPU are "prefetched"; that is, they are input before being needed into the next available entry of the 4-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.

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.

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 Read cycles. Table 3–1 summarizes NS08032 interrupt sequences.

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 NS08032 interrupt structure, see Section 3.7.

Table 3-1.
Interrupt Sequences

Cycle	Status	Address	$\overline{\text{DDIN}}$	Bus
A. Nonmaskable Interrupt Control Sequences.				
Interrupt Acknowledge				
1	0100	FFFF00$_{16}$	0	Don't Care
Interrupt Return				
None: Performed through Return from Trap (RETT) instruction.				
B. Nonvectored Interrupt Control Sequences.				
Interrupt Acknowledge				
1	0100	FFFE00$_{16}$	0	Don't Care
Interrupt Return				
None. Performed through return from Trap (RETT) instruction.				
C. Vectored Interrupt Sequences: Noncascaded.				
Interrupt Acknowledge				
1	0100	FFFE00$_{16}$	0	Vector: Range 0–127
Interrupt Return				
1	0110	FFFE00$_{16}$	0	Vector: Same as in Previous Interrupt Acknowledge Cycle
D. Vectored Interrupt Sequences: Cascaded.				
Interrupt Acknowledge				
1	0100	FFFE00$_{16}$	0	Cascade Index: Range −16 to −1
(The CPU here uses the Cascade Index to find the Cascade Address.)				
2	0101	Cascade Address	0	Vector: Range 0–255
Interrupt Return				
1	0110	FFFE00$_{16}$	0	Cascade Index: Same as in Previous Interrupt Acknowledge Cycle
(The CPU here uses the Cascade Index to find the Cascade Address.)				
2	0111	Cascade Address	0	Don't Care

3.4.6 Slave Processor Communication

The $\overline{SPC}$ pin is used as the data strobe for Slave Processor transfers. In a Slave Processor bus cycle, data is transferred 16 bits at a time on the Data Bus (AD0-AD15) and the least-significant two bits of CPU cycle status (ST0-ST1) are monitored by each Slave Processor in order to determine the type of transfer being performed. *Figure 3-11* shows typical Slave Processor connections. $\overline{SPC}$ is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Section 3.8 for full protocol sequences.

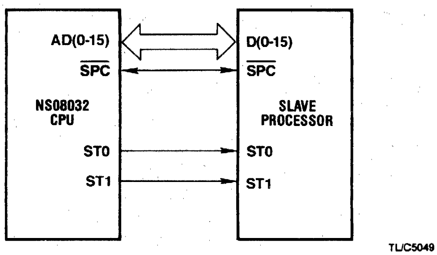
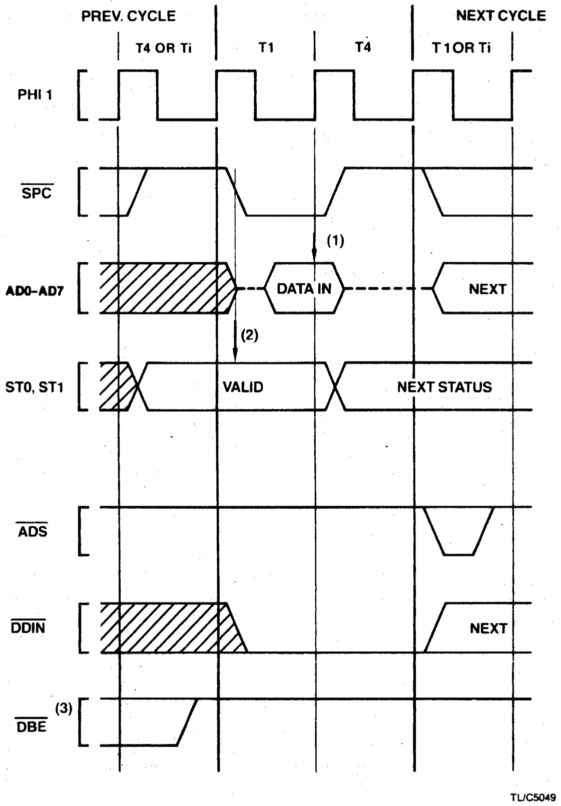


FIGURE 3-11. Slave Processor Connections



Note 1. CPU samples Data Bus here.

Note 2. Slave Processor samples CPU status here.

Note 3. $\overline{DBE}$ and all other NS16201 TCU bus signals remain inactive because no $\overline{ADS}$ pulse is received from the CPU.

FIGURE 3-12. CPU Read from Slave Processor

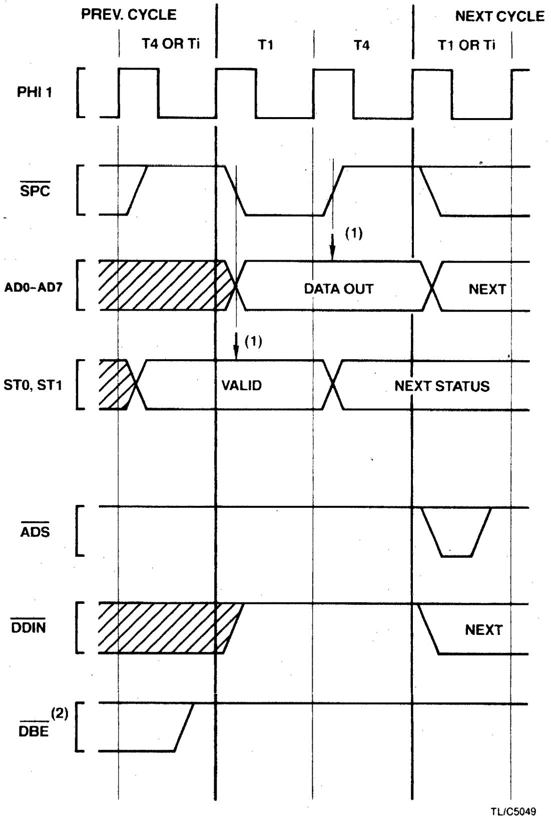
3.4.6.1 Slave Processor Bus Cycles

A Slave Processor bus cycle always takes exactly two clock cycles, labeled T1 and T4 (see *Figures 3-12* and *3-13*). During a Read cycle, $\overline{SPC}$ is activated at T1, data is sampled at T4, and $\overline{SPC}$ is removed. 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 NS16201 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 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 16-bit bus (AD0–AD15). 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 to most-significant word.



Note 1. Arrows indicate points at which the Slave Processor samples.

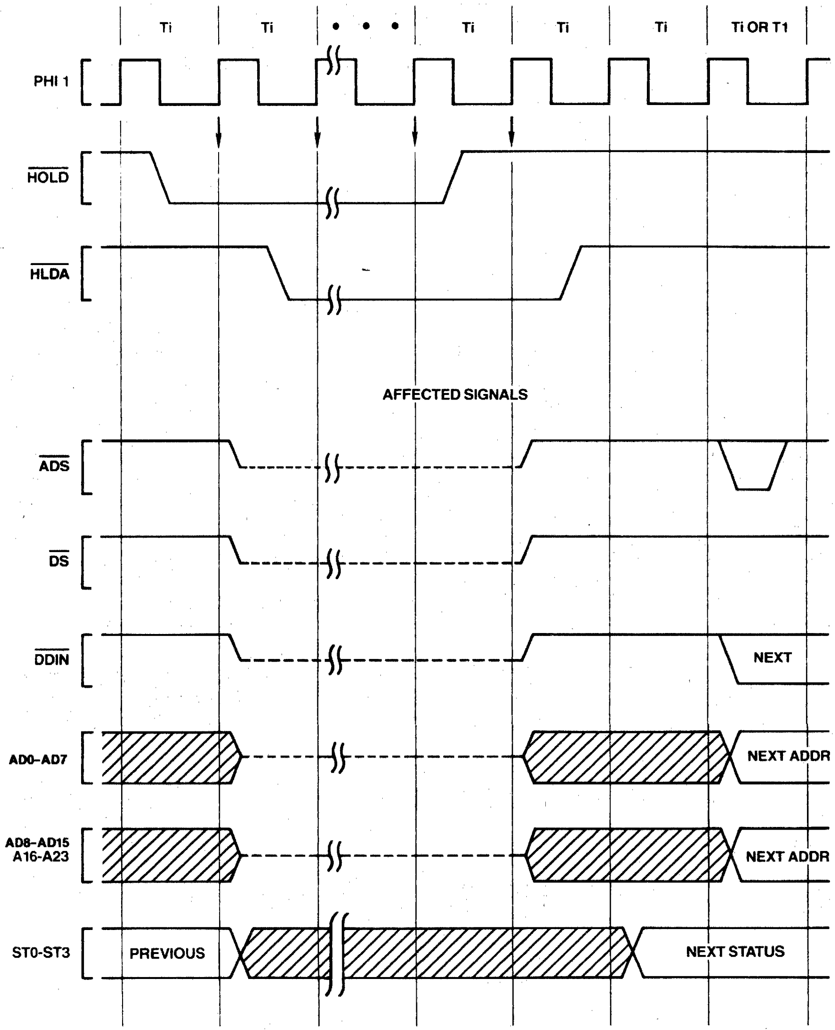
Note 2. $\overline{DBE}$, being provided by the NS16201 TCU, remains inactive due to the fact that no pulse is presented on $\overline{ADS}$, TCU signals $\overline{RD}$, $\overline{WR}$ and $\overline{TSO}$ also remain inactive.

FIGURE 3-13. CPU Write to Slave Processor

3.5 Bus Access Control

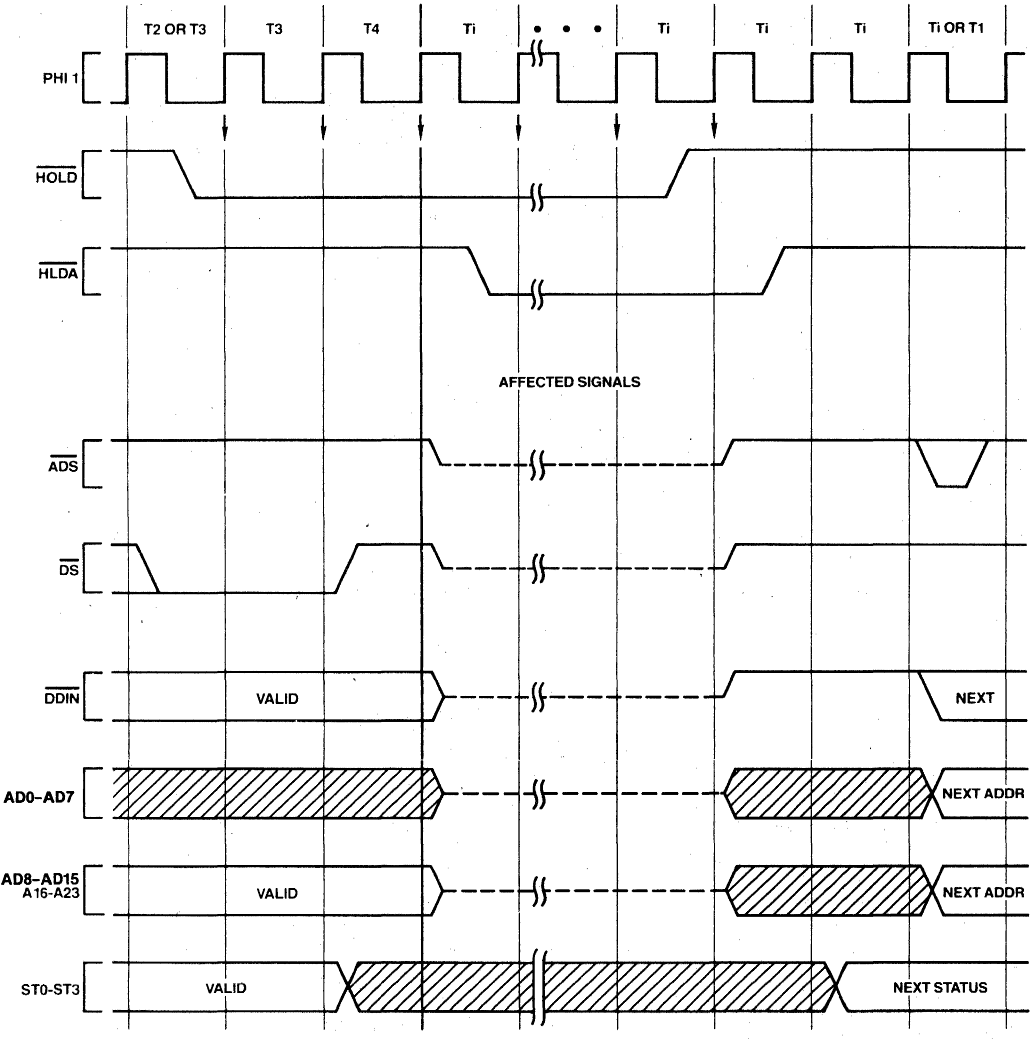
The NS08032 CPU has the capability of relinquishing its access to the bus upon request from a DMA device or another CPU. This capability is implemented on the HOLD (Hold Request) and HLDA (Hold Acknowledge) pins. By asserting HOLD low, an external device requests access to the bus. On receipt of HLDA from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0–AD15, A16–A23, ADS and DDIN pins to the TRI-STATE® condition. To return control of the bus to the CPU, the device sets HOLD inactive, and the CPU acknowledges return of the bus by setting HLDA inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the HOLD request is made, as the CPU must always complete the current bus cycle. *Figure 3-14* shows the timing sequence when the CPU is idle. In this case, the CPU grants the bus during the immediately following clock cycle. *Figure 3-15* shows the sequence if the CPU is using the bus at the time that the 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.



TL/C5049

FIGURE 3-14. HOLD Timing, Bus Initially Idle



TL/C5049

FIGURE 3-15. $\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle

3.6 Instruction Status

In addition to the four bits of bus cycle status (ST0–ST3), the NS08032 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.

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

U/$\overline{\text{S}}$ originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, *Figure 4-19.*

$\overline{\text{ILO}}$ (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multiprocessor communication and resource sharing. As with the U/$\overline{\text{S}}$ pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification section, *Figures 4-16* and *4-17.*

3.7 NS08032 Interrupt Structure

The NS08032 CPU has two interrupt pins: INT, on which maskable interrupts may be requested, and NMI, on which nonmaskable interrupts may be requested.

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).

3.7.1 General Interrupt/Trap Sequence

Upon receipt of an interrupt or trap request, the CPU goes through four 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. Saving Processor Status.
   The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.
3. Vector Acquisition.
   A Vector is either obtained from the Data Bus or is supplied by default.
4. 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 3-16.* 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.

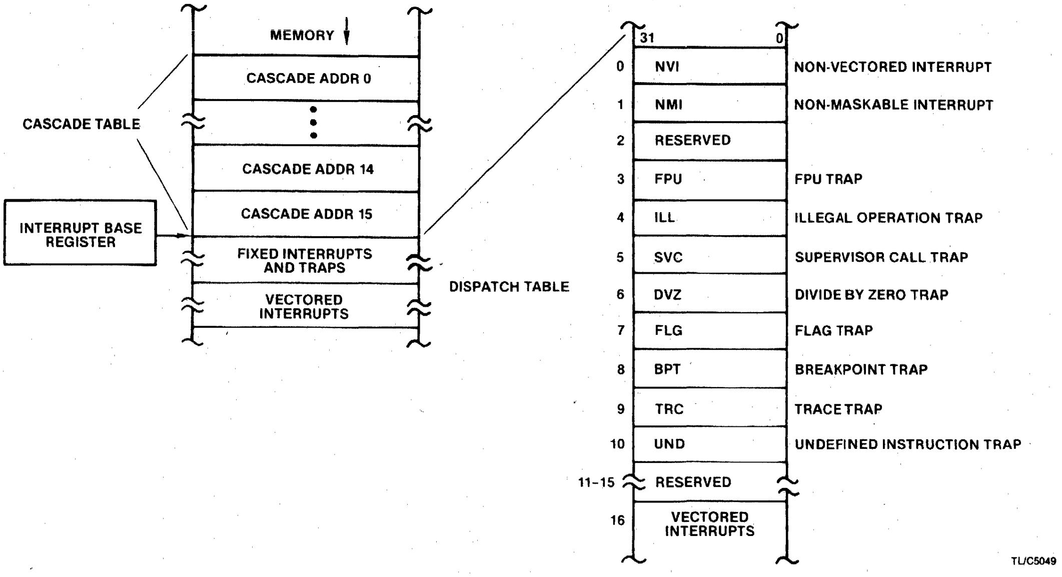


FIGURE 3-16. Interrupt Dispatch and Cascade Tables

This process is illustrated in *Figure 3-17*, from the viewpoint of the programmer.

Full sequences of events in processing interrupts and traps may be found as follows:

Interrupt on $\overline{INT}$ or $\overline{NMI}$ pin:	Sec. 3.7.7.1
Traps (except Trace):	Sec. 3.7.7.2
Trace Trap:	Sec. 3.7.7.3

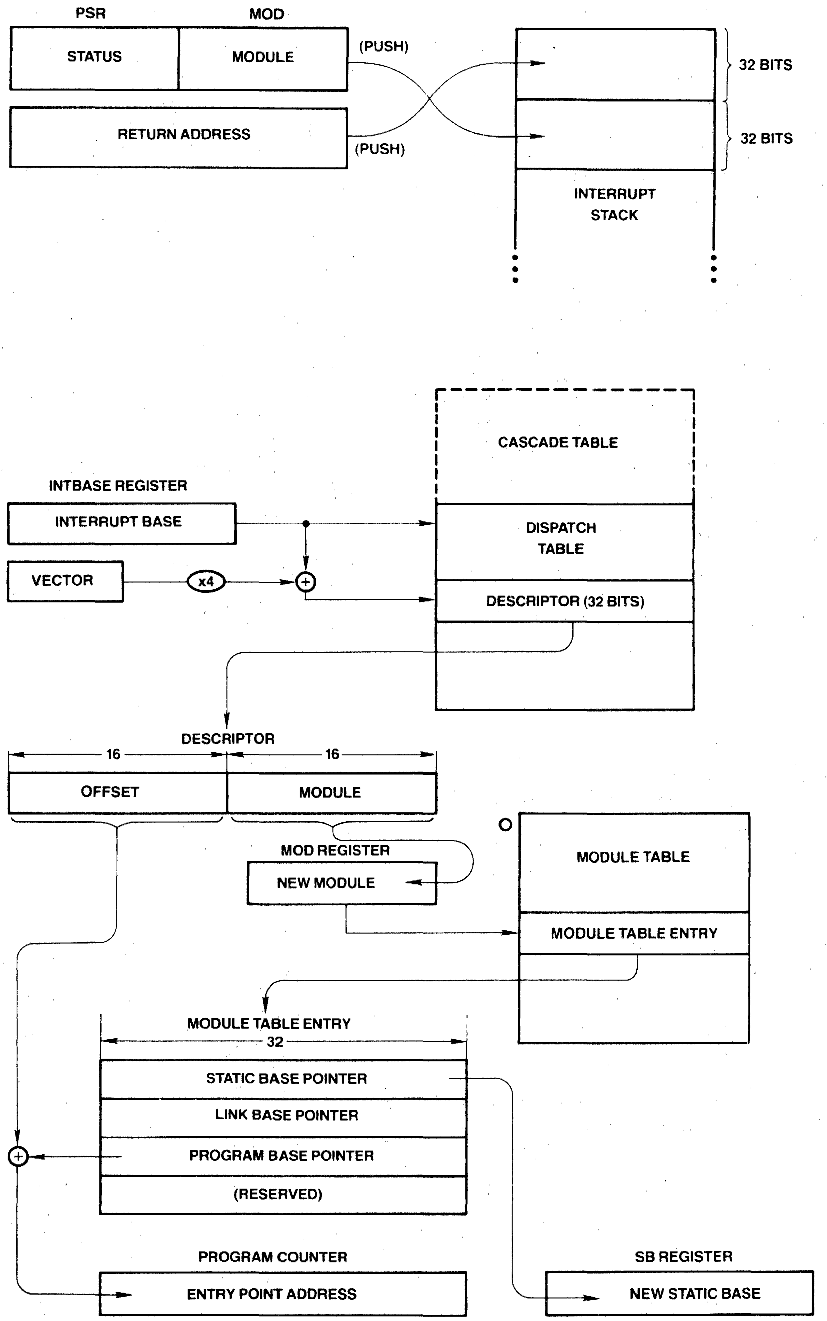


FIGURE 3-17. Interrupt/Trap Service Routine Calling Sequence

3.7.2 Interrupt/Trap Return

To return control to an interrupted program, one of two instructions is used. The RETT (Return From Trap) instruction (*Figure 3-18*) 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 been completed. Since interrupts are generally asynchronous external events, RETI does not pop parameters. See *Figure 3-19*.

3.7.3 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 $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ 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).

3.7.3.1 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. The RETT instruction should be used to return from an interrupt in Non-Vectored Mode.

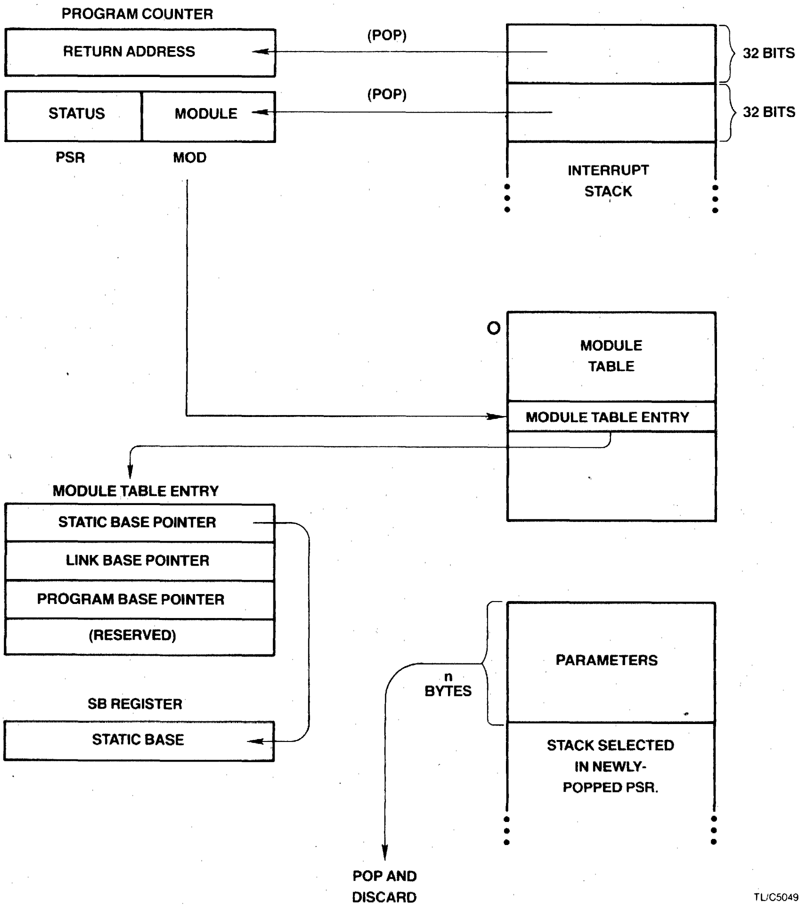


FIGURE 3-18. Return from Trap (RETTn) Instruction Flow

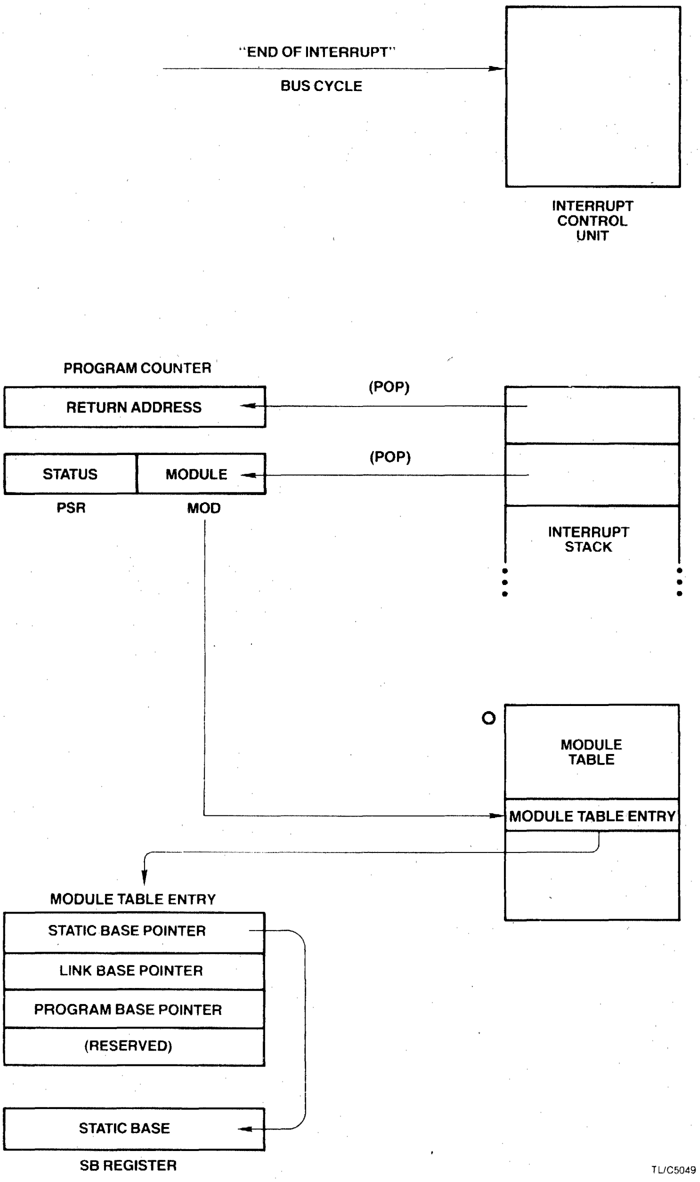


FIGURE 3-19. Return from Interrupt (RETI) Instruction Flow

3.7.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an NS16202 Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. *Figure 3-20* shows the connections required for a single ICU. Upon receipt of an interrupt request on the $\overline{\text{INT}}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Section 3.4.2) reading a vector value from 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 reprioritize 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).

3.7.3.3 Vectored Mode: Cascaded Case

In order to allow up to 256 levels of interrupt, provision is made both in the CPU and in the NS16202 Interrupt Control Unit (ICU) to transparently support cascading. *Figure 3-21* 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 $\overline{\text{INT}}$ 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 3-16 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 Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Section 3.4.2), informing the cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

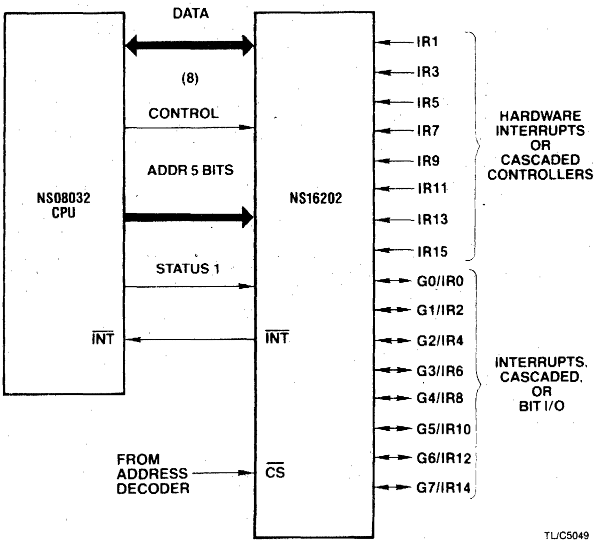


FIGURE 3-20. Interrupt Control Unit Connections (16 Levels)

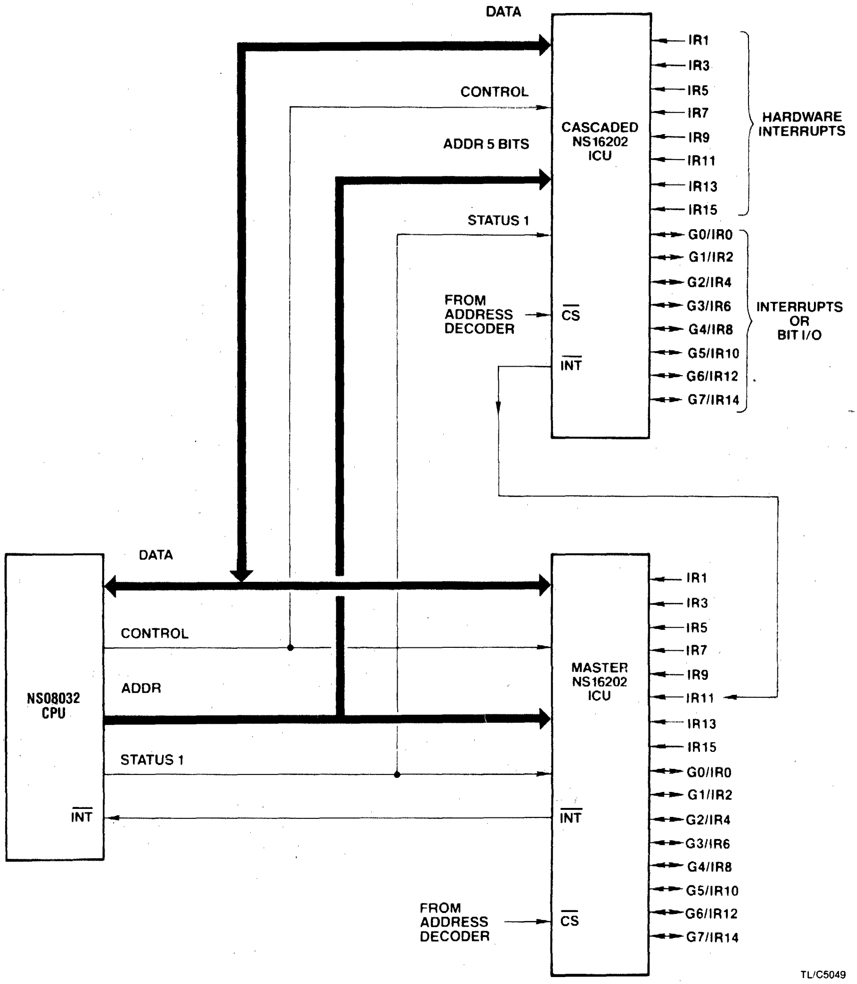


FIGURE 3-21. Cascaded Interrupt Control Unit Connections

3.7.4 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 3.7.7.1.

3.7.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. Traps recognized by the NS08032 are:

Trap (FPU): An exceptional condition was detected by the NS16081 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 3.8.1).

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 FPU 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.

The Return Address pushed by any trap except TRC is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events.

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.

3.7.6 Prioritization

The NS08032 CPU internally prioritizes simultaneous interrupt and trap requests as follows:

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

3.7.7 Interrupt/Trap Sequences: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in *Figure 3-22*. Upon detecting any interrupt request or trap condition, the CPU first performs a sequence dependent upon the type of interrupt or trap. This sequence will include pushing the Processor Status Register and establishing a Vector and a Return Address. The CPU then performs the Service sequence.

For the sequence followed in processing either Maskable or Non-Maskable interrupts (on the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pins, respectively), see Section 3.7.7.1. For the Trace Trap, see Section 3.7.7.3, and for all other traps, see Section 3.7.7.2.

3.7.7.1 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 interruptable 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 FFFE00$_{16}$, 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" $\geqslant$ 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 3-22.*

Service (Vector, Return Address):

1) Push MOD Register onto the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)
2) Push the Return Address onto the Interrupt Stack as a 32-bit quantity.
3) Read the 32-bit External Procedure Descriptor from the Interrupt Dispatch Table: address is Vector*4+INTBASE Register contents.
4) Move the Module field of the Descriptor into the MOD Register.
5) Read the new Static Base pointer from the memory address contained in MOD, placing it into the SB Register.
6) 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.

TL/C5049

FIGURE 3-22. Service Sequence
Invoked during all interrupt/trap sequences.

3.7.7.2 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.

FPU:	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, T and P.
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 3-22.*

3.7.7.3 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 3-22.*

3.8 Slave Processor Instructions

The NS08032 CPU recognizes two groups of instructions as being executable by external Slave Procesors:

Floating-Point 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. Slave Processor cycles use pins AD0–AD15 as a 16-bit data bus.

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).

TL/C5049

FIGURE 3-23. Slave Processor Protocol

3.8.1 Slave Processor Protocol

Slave Processor instructions have a 3-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 3-23.* 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, respectively, and bits 8–15 appear on pins AD0–AD7, respectively.

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).

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

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 SPC, the CPU uses 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 3-24.* 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

FPU 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 a Custom Slave instruction (LCR: Load Custom Register). In executing this instruction, 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.

3.8.2 Floating-Point Instructions

Table 3-2 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 Classes for each general operand, defining how the addressing modes are interpreted (see Programmer's 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 3-24*).

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.

Table 3-2.
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	write.D	N/A	N/A	N/A	D to Op. 1	none

Note:
D = Double word.
i = Integer size (B,W,D) specified in mnemonic.
f = Floating-point type (F,L) specified in mnemonic.
N/A = Not applicable to this instruction.

3.8.3 Custom Slave Instruction

Provided in the NS08032 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 opcode 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 3-3 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.

For the instruction encodings, see Appendix A.

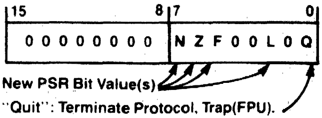


FIGURE 3-24. Slave Processor Status Word Format

Table 3-3.
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
CCMPc	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write. i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	write.D	N/A	N/A	N/A	D to Op. 2	none
CATST0*	addr	N/A	D	N/A	N/A	F
CATST1*	addr	N/A	D	N/A	N/A	F
LCR*	read.D	N/A	D	N/A	N/A	none
SCR*	write.D	N/A	N/A	N/A	D to Op. 1	none

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

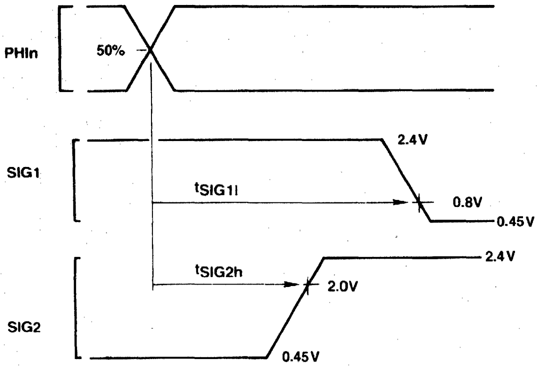
4 AC Electrical Characteristics

4.1 Definitions

All the timing specifications given in this section refer to 50% of the leading or trailing edges of the appropriate clock phase and 0.8V or 2.0V on the appropriate signal as illustrated in *Figures 4-1* and *4-2*, unless specifically stated otherwise.

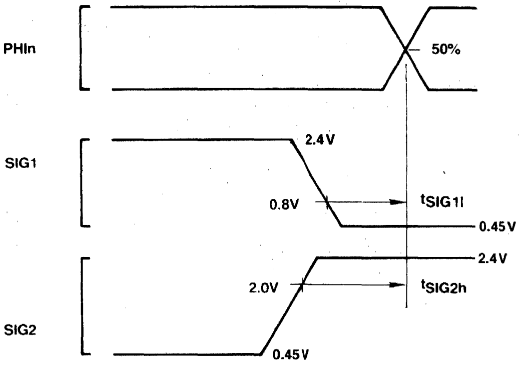
Abbreviations:

L.E.—leading edge
T.E.—trailing edge



TL/C5049

FIGURE 4-1. Timing Specification Standard (Signal Valid After Edge)



TL/C5049

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

4.2 Timing Tables

Table 4-1. Output Signals: Internal Propagation Delays, NS08032-4, NS08032-6
Maximum times assume capacitive loading of 100pF.

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{ALv}	Address bits 0-7 valid	4-3	after L.E., PHI1 T1			80	ns
t_{ALh}	Address bits 0-7 hold	4-3	after L.E., PHI1 Tmmu or T2	0			ns
t_{Dv}	Data valid (write cycle)	4-3	after L.E., PHI1 T2			80	ns
t_{Dh}	Data hold (write cycle)	4-3	after L.E., PHI1 next T1 or Ti	0			ns
t_{AHv}	Address bits 8-23 valid	4-3	after L.E., PHI1 T1			80	ns
t_{AHh}	Address bits 8-23 hold	4-3	after L.E., PHI1 next T1 or Ti	0			ns
t_{ALADSs}	Address bits 0-7 set up to $\overline{\text{ADS}}$ T.E.	4-4	before $\overline{\text{ADS}}$ reaches 2.0V	20			ns
t_{AHADSs}	Address bits 8-23 set up to $\overline{\text{ADS}}$ T.E.	4-4	before $\overline{\text{ADS}}$ reaches 2.0V	20			ns
t_{ALADSh}	Address bits 0-7 hold from $\overline{\text{ADS}}$ T.E.	4-9	after $\overline{\text{ADS}}$ reaches 2.0V	10			ns
t_{AHADSh}	Address bits 8-23 hold from $\overline{\text{ADS}}$ T.E.	4-8	after $\overline{\text{ADS}}$ reaches 2.0V	10			ns
t_{ALf}	Address bits 0-7 floating	4-4	after L.E., PHI1 T2			25	ns
t_{STv}	Status (ST0-ST3) valid	4-3	after L.E., PHI1 T4 (before T1, see note)			90	ns
t_{STh}	Status (ST0-ST3) hold	4-3	after L.E., PHI1 T4 (after T1)	0			ns
t_{DDINv}	$\overline{\text{DDIN}}$ signal valid	4-4	after L.E., PHI1 T1			110	ns
t_{DDINh}	$\overline{\text{DDIN}}$ signal hold	4-4	after L.E., PHI1 next T1 or Ti	0			ns
t_{ADSa}	$\overline{\text{ADS}}$ signal active (low)	4-3	after L.E., PHI1 T1			50	ns
t_{ADSia}	$\overline{\text{ADS}}$ signal inactive	4-3	after T.E., PHI1 T1			65	ns
t_{ADSw}	$\overline{\text{ADS}}$ pulse width	4-3	at 0.8V, both edges	60			ns
t_{DSa}	$\overline{\text{DS}}$ signal active (low)	4-3	after L.E., PHI1 T2			70	ns

Table 4-1. Output Signals: Internal Propagation Delays, NS08032-4, NS08032-6 (Cont'd)

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{DSia}	$\overline{DS}$ signal inactive	4-3	after L.E., PHI1 T4			60	ns
t_{ALf}	AD0-AD7 floating (caused by $\overline{HOLD}$)	4-5	after L.E., PHI1 T1			100	ns
t_{AHf}	A8-A23 floating (caused by $\overline{HOLD}$)	4-5	after L.E., PHI1 T1			100	ns
t_{ADSf}	$\overline{ADS}$ floating (caused by $\overline{HOLD}$)	4-5	after L.E., PHI1 Ti			100	ns
t_{DDINf}	$\overline{DDIN}$ floating (caused by $\overline{HOLD}$)	4-5	after L.E., PHI1 Ti			100	ns
t_{HLDAa}	$\overline{HLDA}$ signal active (low)	4-5	after L.E., PHI1 Ti			100	ns
t_{HLDAia}	$\overline{HLDA}$ signal inactive	4-7	after L.E., PHI1 Ti			100	ns
t_{ADSr}	$\overline{ADS}$ signal returns from floating (caused by $\overline{HOLD}$)	4-7	after L.E., PHI1 Ti			100	ns
t_{DDINr}	$\overline{DDIN}$ signal returns from floating (caused by $\overline{HOLD}$)	4-7	after L.E., PHI1 Ti			100	ns
t_{SPCa}	$\overline{SPC}$ output active (low)	4-12	after L.E., PHI1 T1			50	ns
t_{SPCia}	$\overline{SPC}$ output inactive	4-12	after L.E., PHI1 T4			50	ns
t_{SPCnf}	$\overline{SPC}$ output nonforcing	4-14	after L.E., PHI2 T4			40	ns
t_{Dv}	Data valid (slave processor write)	4-12	after L.E., PHI1 T1			80	ns
t_{Dh}	Data hold (slave processor write)	4-12	after L.E., PHI1 next T1 or Ti	0			ns
t_{PFSw}	$\overline{PFS}$ pulse width	4-17	at 0.8V, both edges	70			ns
t_{PFSa}	$\overline{PFS}$ pulse active (low)	4-17	after L.E., PHI2			70	ns
t_{PFSia}	$\overline{PFS}$ pulse inactive	4-17	after L.E., PHI2			70	ns
t_{ILOs}	$\overline{ILO}$ signal setup	4-19a	before L.E., PHI1 T1 of first interlocked write cycle	0			ns
t_{ILOh}	$\overline{ILO}$ signal hold	4-19b	after L.E., PHI1 T3 of last interlocked read cycle)	0			ns
t_{ILOa}	$\overline{ILO}$ signal active (low)	4-20	after L.E., PHI1			70	ns
t_{ILOia}	$\overline{ILO}$ signal inactive	4-20	after L.E., PHI1			70	ns
t_{USs}	U/$\overline{S}$ signal setup	4-21	before L.E., PHI1 T4	0			ns
t_{USh}	U/$\overline{S}$ signal hold	4-21	after L.E., PHI1 T1	2			t_{Cp}
t_{NSPF}	Nonsequential fetch to next $\overline{PFS}$ clock cycle	4-18b	after L.E., PHI1 T1	4			t_{Cp}
t_{PFNS}	$\overline{PFS}$ clock cycle to next nonsequential fetch	4-18a	before L.E., PHI1 T1	4			t_{Cp}
t_{LXPF}	Last operand transfer of an instruction to next $\overline{PFS}$ clock cycle	4-29	before L.E., PHI1 T1 of first bus cycle of transfer	0			t_{Cp}

Note 1. Timing parameters for components with an "S" suffix are not guaranteed compatible with an NS16081 Slave Processor at a clock rate greater than 4MHz.

Note 2. 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...".

Table 4-2. Input Signal Requirements: NS08032-4, NS08032-6

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{PWR}	Power stable to $\overline{\text{RST}}$ T.E.	4-24	after V_{CC} reaches 4.5V	50			µs
t_{DIs}	Data in setup (read cycle)	4-4	before T.E., PHI2 T3	20			ns
t_{DIh}	Data in hold (read cycle)	4-4	after L.E., PHI1 T4	0			ns
t_{HLDa}	$\overline{\text{HOLD}}$ active (low) setup time (See note)	4-5	before T.E., PHI2 TX1	25			ns
t_{HLDia}	$\overline{\text{HOLD}}$ inactive setup time	4-7	before T.E., PHI2 Ti	25			ns
t_{HLDh}	$\overline{\text{HOLD}}$ hold time	4-5	after L.E., PHI1 TX2	0			ns
t_{RDYs}	RDY setup time	4-10, 4-11	before T.E., PHI2 T2 or T3	25			ns
t_{RDYh}	RDY hold time	4-10, 4-11	after L.E., PHI2 T3	0			ns
t_{RSTs}	$\overline{\text{RST}}$ setup time	4-24, 4-25	before T.E., PHI1	20			ns
t_{RSTw}	$\overline{\text{RST}}$ pulse width	4-25	at 0.8V (both edges)	64			t_{Cp}
t_{INTa}	$\overline{\text{INT}}$ setup time	4-26	before T.E., PHI1	10			ns
t_{NMIw}	$\overline{\text{NMI}}$ pulsewidth	4-27	at 0.8V (both edges)	40			ns
t_{NMPF}	$\overline{\text{NMI}}$ setup time after a $\overline{\text{PFS}}$ clock cycle (for use of $\overline{\text{NMI}}$ as a breakpoint)	4-28	before L.E., PHI1 (as shown)	−10			ns
t_{DIs}	Data setup (slave read cycle)	4-13	before T.E., PHI2 T1	20			ns
t_{DIh}	Data hold (slave read cycle)	4-13	after L.E., PHI1 T4	0			ns
t_{SPCw}	$\overline{\text{SPC}}$ pulse width (from slave processor)	4-12	at 0.8V (both edges)	30			ns

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

Table 4-3. Clocking Requirements: NS08032-4

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{CLr}	PHI1, PHI2 rise time	4-16	to V_{CH} (see page 2)			9	ns
t_{CLf}	PHI1, PHI2 fall time	4-16	from 90-10% of V_{CH} (see page 2)			9	ns
t_{CLh}	PHI1, PHI2 high time	4-16		$0.5t_{Cp}$ −15			ns
t_{Cp}	Clock period	4-16		240		5000	ns
t_{OVL}	Non-overlap time	4-16	at 10% of V_{CH} (see page 2)	0		15	ns

Table 4-4. Clocking Requirements: NS08032-6

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{CLr}	PHI1, PHI2 rise time	4-16	to V_{CH} (see page 2)			9	ns
t_{CLf}	PHI1, PHI2 fall time	4-16	from 90-10% of V_{CH} (see page 2)			9	ns
t_{CLh}	PHI1, PHI2 high time	4-16		$0.5t_{Cp}$ −15			ns
t_{Cp}	Clock period	4-16		160		5000	ns
t_{OVL}	Non-overlap time	4-16	at 10% of V_{CH} (see page 2)	0		15	ns

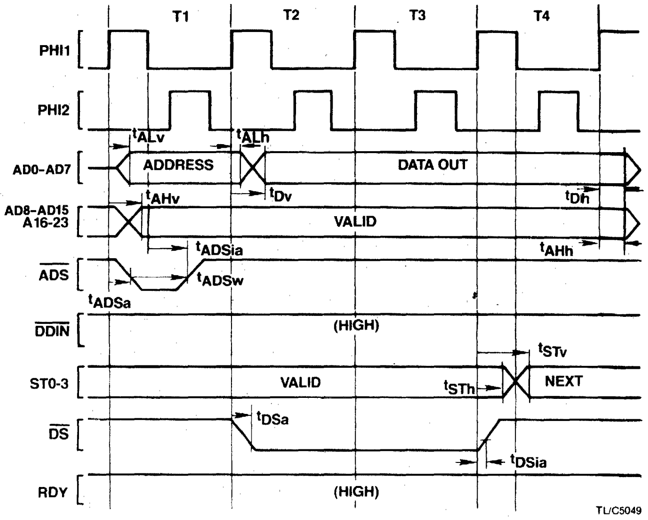


FIGURE 4-3. Write Cycle

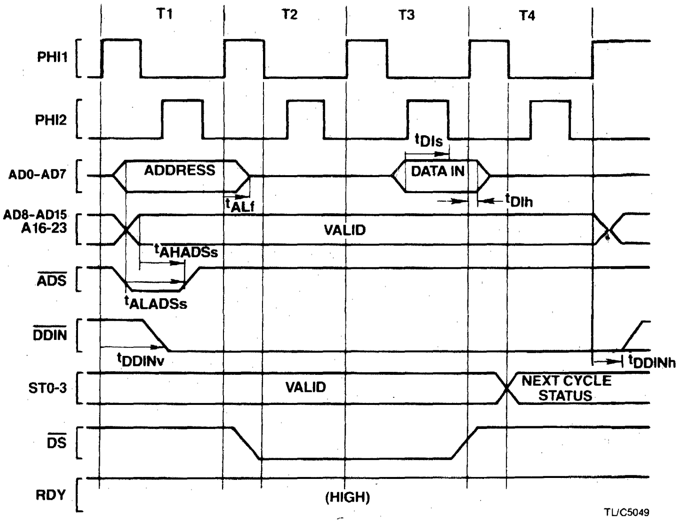
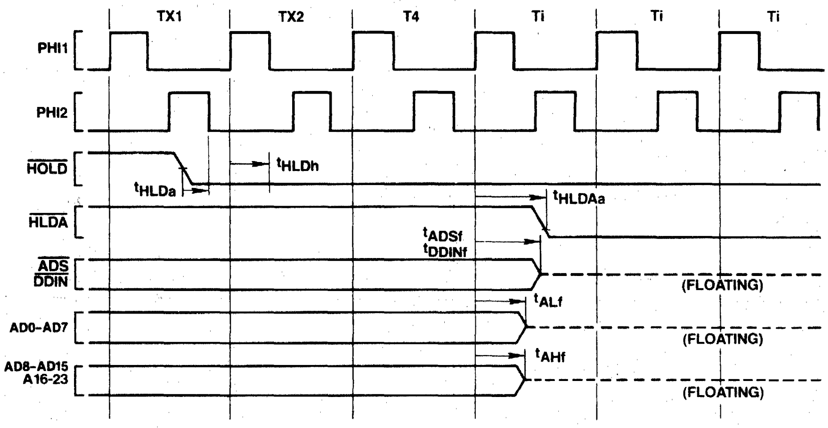


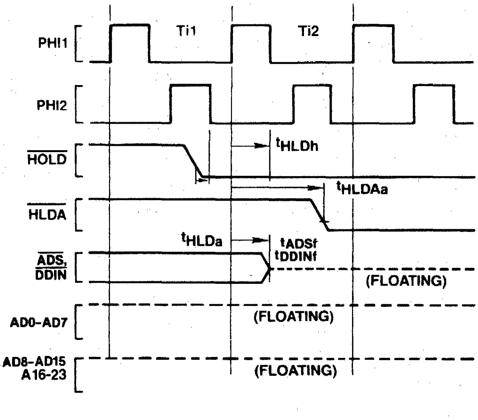
FIGURE 4-4. Read Cycle



TL/C5049

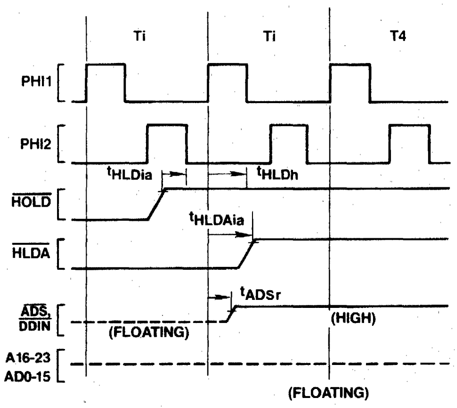
Note: Whenever the CPU is not idling (not in Ti), the $\overline{\text{HOLD}}$ request ($\overline{\text{HOLD}}$ Low) must be active t_{HLDa} before the trailing edge of PHI2 of the clock cycle that appears two clock cycles before T4 (TX1) and stay low until t_{HLDh} after the leading edge of PHI1 of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

FIGURE 4-5. Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially)



TL/C5049

FIGURE 4-6. Floating by $\overline{\text{HOLD}}$ Timing (CPU Initially Idle)



TL/C5049

FIGURE 4-7. Release from $\overline{\text{HOLD}}$

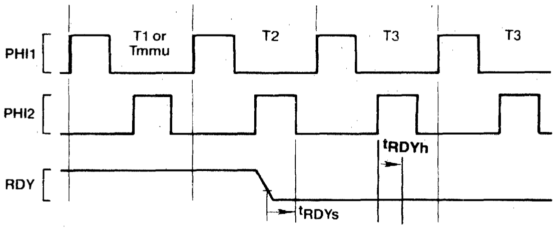


FIGURE 4-8. Ready Sampling (CPU Initially READY)

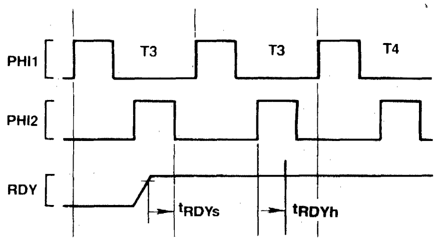


FIGURE 4-9. Ready Sampling (CPU Initially NOT READY)

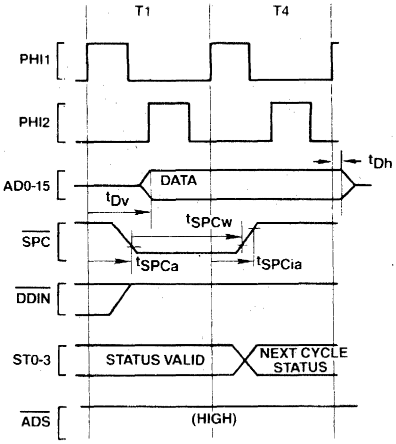


FIGURE 4-10. Slave Processor Write Timing

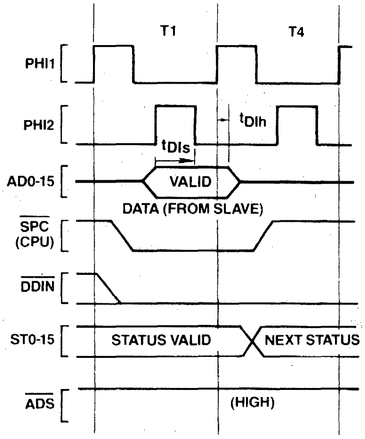
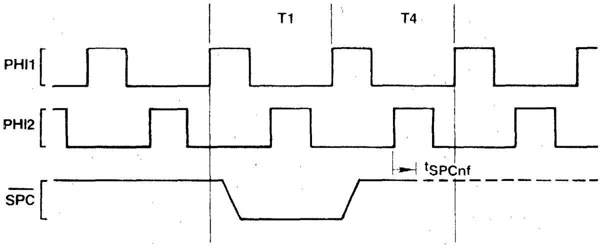


FIGURE 4-11. Slave Processor Read Timing



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

FIGURE 4-12. $\overline{\text{SPC}}$ Non-Forcing Delay

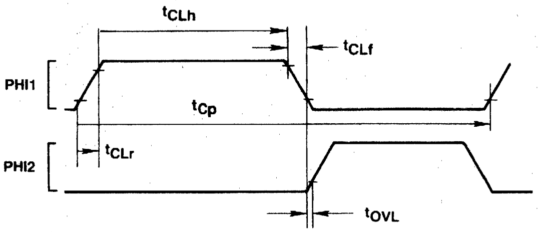


FIGURE 4-13. Clock Waveforms

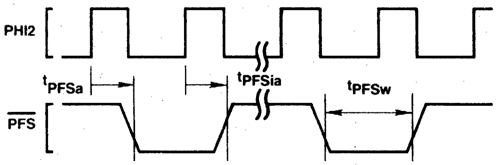


FIGURE 4-14. Relationship of $\overline{\text{PFS}}$ to Clock Cycles

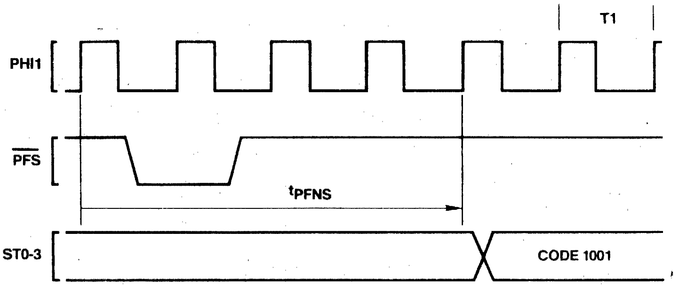


FIGURE 4-15a. Guaranteed Delay, $\overline{\text{PFS}}$ to Non-Sequential Fetch

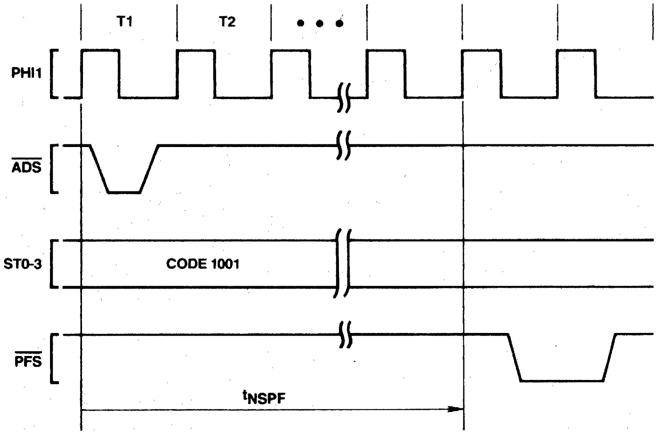


FIGURE 4-15b. Guaranteed, Delay, Non-Sequential Fetch to $\overline{\text{PFS}}$

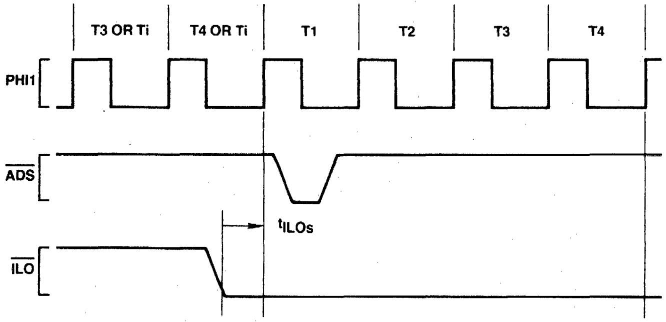


FIGURE 4-16. Relationship of $\overline{\text{ILO}}$ to First Operand Cycle of an Interlocked Instruction

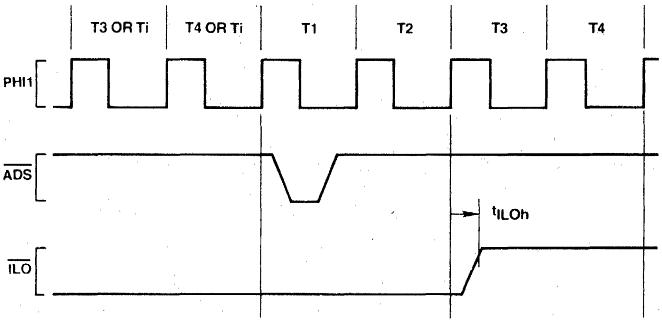


FIGURE 4-17. Relationship of $\overline{\text{ILO}}$ to Last Operand Cycle of an Interlocked Instruction

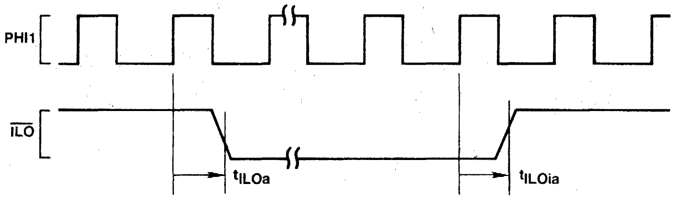


FIGURE 4-18. Relationship of $\overline{\text{ILO}}$ to Any Clock Cycle

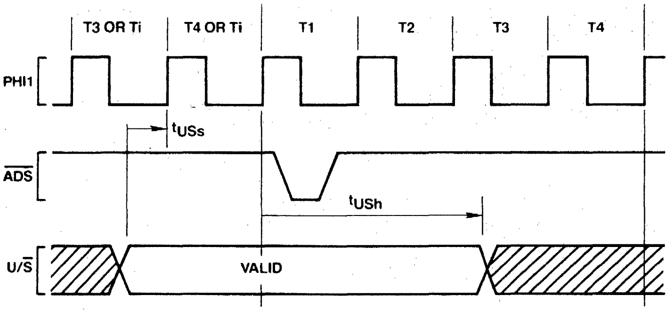


FIGURE 4-19. U/$\overline{\text{S}}$ Relationship to Any Bus Cycle— Guaranteee Valid Interval

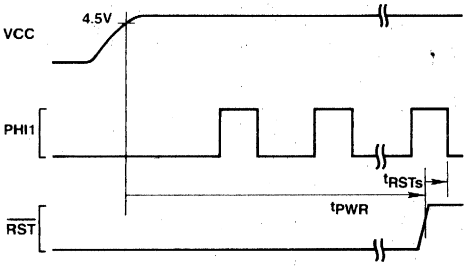


FIGURE 4-20. Power-On Reset

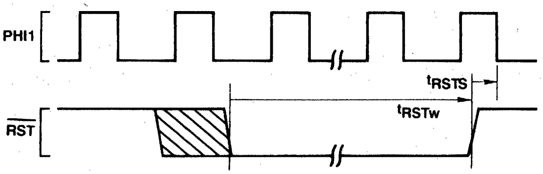
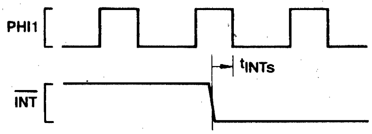


FIGURE 4-21. Non-Power—On Reset



Note: Violation of t_{INTs} timing is allowed, but detection then occurs one clock cycle later.

FIGURE 4-22. $\overline{\text{INT}}$ Interrupt Signal Detection

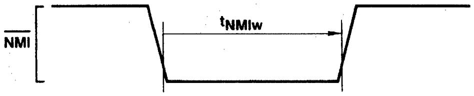


FIGURE 4-23. $\overline{\text{NMI}}$ Interrupt Signal Timing

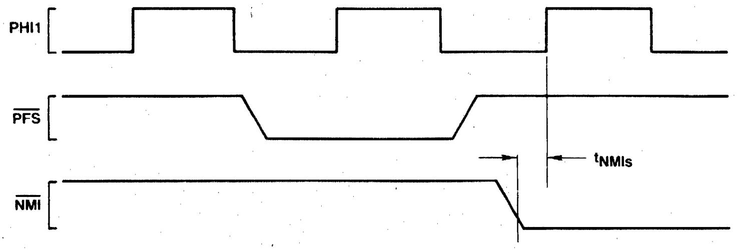
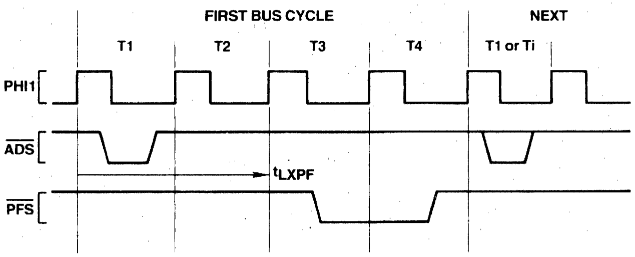


FIGURE 4-24. Required Relationship of $\overline{\text{NMI}}$ to $\overline{\text{PFS}}$ (Breakpoint Use)



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

FIGURE 4-25. Relationship Between Last Data Transfer of an Instruction and $\overline{\text{PFS}}$ Pulse of Next Instruction

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 (Standard 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, gen1,
gen2 . . . . General Addressing Mode Field.
See Section 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)

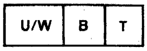
short . . . . Short Immediate Value. May contain:

quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB

cond: Condition Code (above), in Scond.

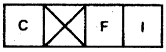
areg: CPU Dedicated Register, in LPR, SPR.
0000 = US
0001-0111 = (Reserved)
1000 = FP
1001 = SP
1010 = SB
1011 = (Reserved)
1100 = (Reserved)
1101 = PSR
1110 = INTBASE
1111 = MOD

Options: in String Instructions

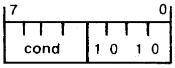


T = Translated
B = Backward
U/W = 00: None
01: While Match
11: Until Match

Configuration bits, in SETCFG:

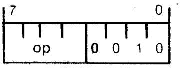


mreg: MMU Register number, in LMR, SMR.
0000 = BPR0
0001 = BPR1
0010 = (Reserved)
0011 = (Reserved)
0100 = PF0
0110 = (Reserved)
0111 = (Reserved)
1000 = SC
1001 = (Reserved)
1010 = MSR
1011 = BCNT
1100 = PTB0
1101 = PTB1
1110 = (Reserved)
1111 = EIA



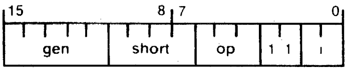
Format 0

Bcond (BR)



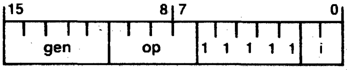
Format 1

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

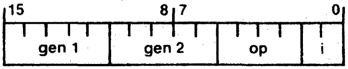
ADDQ	–000	ACB	–100
CMPQ	–001	MOVQ	–101
SPR	–010	LPR	–110
Scond	–011		



Format 3

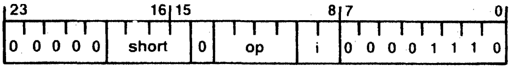
CXPD	−0000	ADJSP	−1010
BICPSR	−0010	JSR	−1100
JUMP	−0100	CASE	−1110
BISPSR	−0110		

Trap (UND) on XXX1, 1000



Format 4

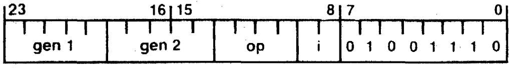
ADD	−0000	SUB	−1000
CMP	−0001	ADDR	−1001
BIC	−0010	AND	−1010
ADDC	−0100	SUBC	−1100
MOV	−0101	TBIT	−1101
OR	−0110	XOR	−1110



Format 5

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

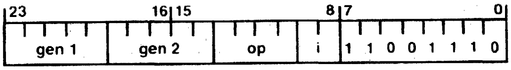
Trap (UND) on 1XXX, 01XX



Format 6

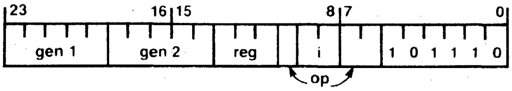
NEG	−1000		
ASH	−0001	NOT	−1001
CBIT	−0010	Trap (UND)	−1010
CBITI	−0011	SUBP	−1011
Trap (UND)	−0100	ABS	−1100
LSH	−0101	COM	−1101
SBIT	−0110	IBIT	−1110
SBITI	−0111	ADDP	−1111

Trap (UND) on all others



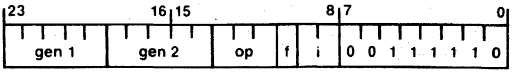
Format 7

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



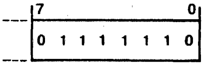
Format 8

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



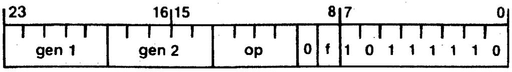
Format 9

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



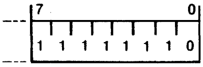
Format 10

Trap (UND) Always



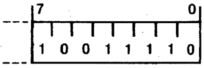
Format 11

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



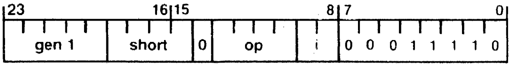
Format 12

Trap (UND) Always



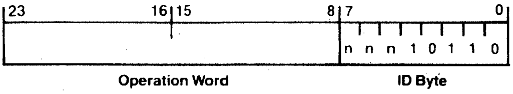
Format 13

Trap (UND) Always



Format 14

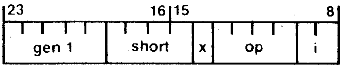
Trap (UND) Always



Format 15
(Custom Slave)

nnn **Operation Word Format**

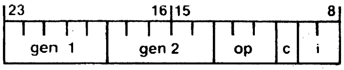
000



Format 15.0

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

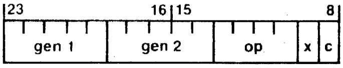
001



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	Trap (UND)	–1010
CCMP	–0010	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

7							0
0	1	0	1	1	1	1	0

Format 16

Trap (UND) Always

7							0
1	1	0	1	1	1	1	0

Format 17

Trap (UND) Always

7							0
1	0	0	0	1	1	1	0

Format 18

Trap (UND) Always

7							0
X	X	X	0	0	1	1	0

Format 19

Trap (UND) Always

Implied Immediate Encodings:

7							0
r7	r6	r5	r4	r3	r2	r1	r0

Register Mask, appended to SAVE, ENTER

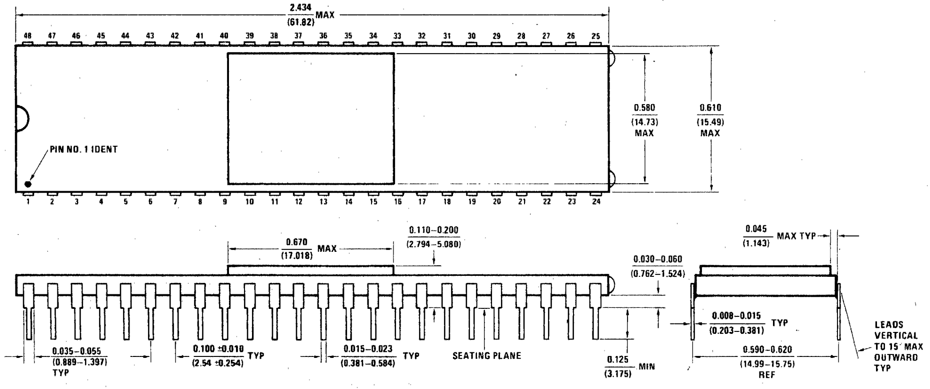
7							0
r0	r1	r2	r3	r4	r5	r6	r7

Register Mask, appended to RESTORE, EXIT

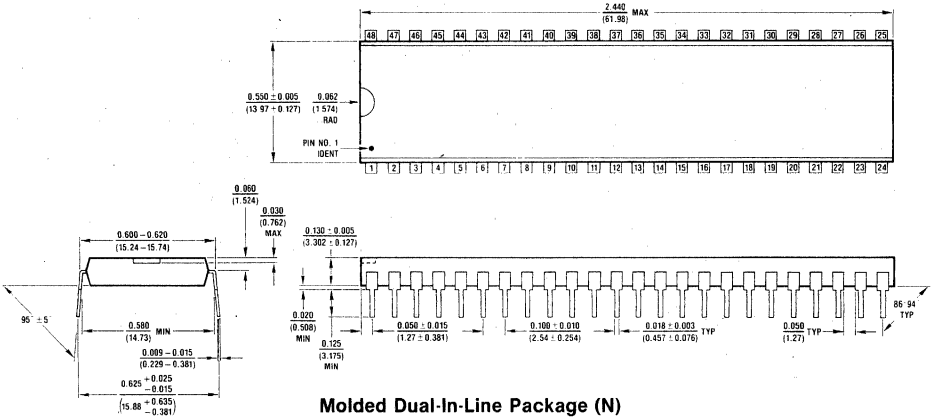
7							0
	offset				length – 1		

Offset/Length Modifier, appended to INSS, EXTS

Physical Dimensions inches (millimeters)



Ceramic Dual-In-Line Package (D)
NS Package D48A



Molded Dual-In-Line Package (N)
NS Package N48A

ORDERING INFORMATION

NS08032D-6, NS08032D-4
NS08032N-6, NS08032N-4

EVALUATION TOOLS ORDERING INFORMATION

DB16000*	Evaluation Board
NSX-16	NS16000 Cross Software Package (VAX/VMS)
SFW-90-A500	NS16000 Cross Software Package (STARPLEX II™)
SPX-90/51	STARPLEX II with Single-Sided, Double-Density System
SPX-90/61	STARPLEX II with Double-Sided, Double-Density System

*To be used with an NS08032 plug-in module for NS08032 evaluation and development.

National Semiconductor

NS16032-4, NS16032-6 High-Performance Microprocessors

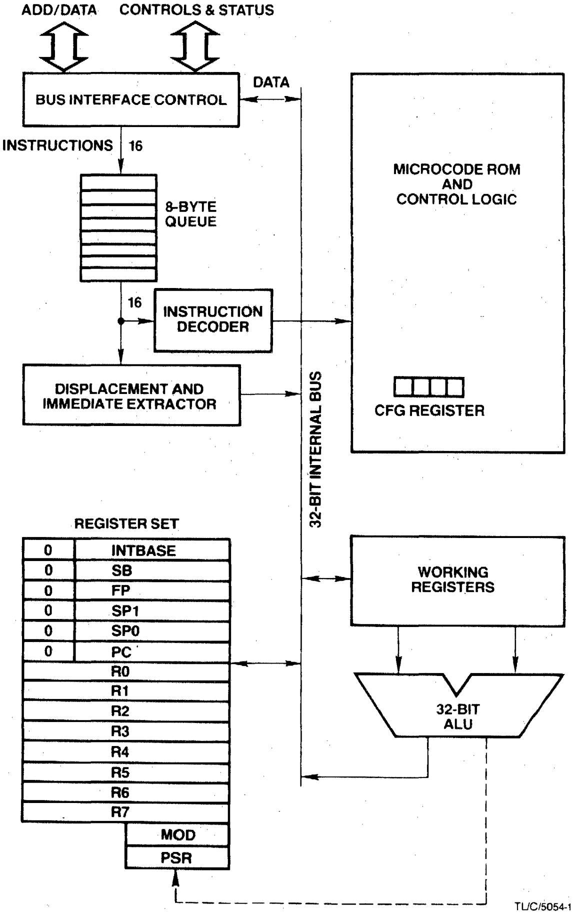
General Description

The NS16032 functions as a central processing unit (CPU) in National Semiconductor's NS16000™ microprocessor family. It has been designed to optimally support microprocessor users who need the ability to use a large addressing space for large programs and/or large data structures. Because large programs must realistically be generated and maintained in high-level languages, the NS16000 architecture provides for very efficient compilation while remaining easy to program at the assembler level for optimizations. NS16000 architecture provides for full virtual memory capability, in conjunction with with the NS16082 Memory Management Unit (MMU). High performance floating-point instructions are provided with the NS16081 Floating-Point Unit (FPU). The NS16032-4 and NS16032-6 have different timing parameters. Refer to Section 4 for timing specifications.

Features

- 32-bit Architecture and Implementation
- 16-MByte 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 XMOS™ Technology
- Single 5V Supply
- 48-pin Dual-In-Line Package

NS16032 CPU Block Diagram



TL/C/5054-1

Absolute Maximum Ratings

Temperature under bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All input or output voltages with respect to GND	−0.5V to +7V
Power Dissipation	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 DC Electrical Characteristics.*

DC Electrical Characteristics: NS16032: T_A = 0 to +70°C, V_{CC} = 5V ±5%, GND = 0V

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{IH}	Logical 1 Input Voltage		2.0		V_{CC}+0.5	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	V_{CC}−0.4		V_{CC}+0.5	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CLT}	Logical 0 Clock Voltage, Transient (ringing tolerance)	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	Logical 1 Output Voltage	I_{OH} = −400μA	2.4			V
V_{OL}	Logical 0 Output Voltage	I_{OL} = 2mA			0.45	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 Leakage Current	$0 \leqslant V_{IN} \leqslant V_{CC}$, All inputs except PHI1, PHI2, $\overline{AT}/\overline{SPC}$	−20		20	μA
$I_{O(OFF)}$	Output Leakage Current	$0.4 \leqslant V_{OUT} \leqslant V_{CC}$	−20		20	μA
I_{CC}	Active Supply Current	I_{OUT} = 0, T_A = 25°C		180	300	mA

1 NS16032 Pin Descriptions

The following is a brief description of all NS16032 pins. The descriptions reference portions of the Functional Description, Section 3.

1.1 SUPPLIES

Power (V_{CC}): +5V Positive Supply. Sec. 3.1.

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

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

Back-Bias Generator (BBG): Output of on-chip substrate voltage generator. Sec. 3.1.

1.2 INPUT SIGNALS

Clocks (PHI1, PHI2): Two-phase clocking signals. Sec. 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. Sec. 3.4.1.

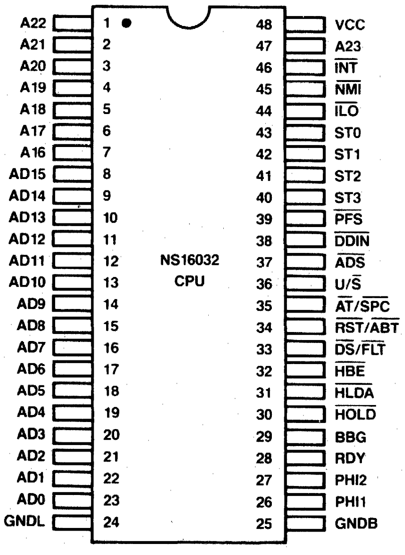
Hold Request ($\overline{\mathrm{HOLD}}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Sec. 3.6.

Interrupt ($\overline{\mathrm{INT}}$): Active low. Maskable Interrupt request. Sec. 3.8.

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

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

Connection Diagram



TL/C/5054-2

1.3 OUTPUT SIGNALS

Address Bits 16-23 (A16-A23): Active high. These are the most significant 8 bits of the memory address bus. Sec. 3.4.

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

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

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

Status (ST0-ST3): Active high. Bus cycle status code, ST0 least significant. Sec. 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 ($\overline{\mathrm{HLDA}}$): Active low. Applied by the CPU in response to $\overline{\mathrm{HOLD}}$ input, indicating that the bus has been released for DMA or multiprocessing purposes. Sec. 3.6.

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

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

Program Flow Status ($\overline{\mathrm{PFS}}$): Active low. Pulse indicates beginning of an instruction execution. Sec. 3.7.

1.4 INPUT-OUTPUT SIGNALS

Address/Data 0-15 (AD0-AD15): Active high. Multiplexed Address/Data information. Bit 0 is the least significant bit of each. Sec. 3.4.

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

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

2 Architectural Description

2.1 PROGRAMMING MODEL

The NS16000 architecture includes 16 registers on the NS16032 CPU.

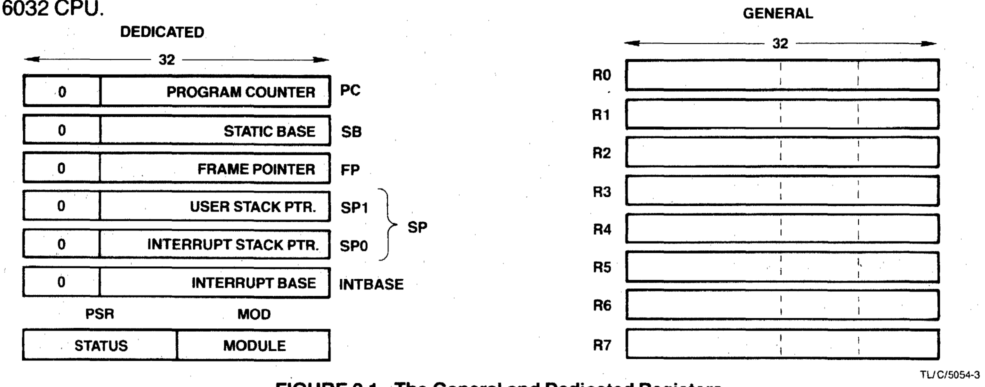


FIGURE 2-1. The General and Dedicated Registers.

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 NS16032 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 NS16032 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 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 NS16032 the upper eight bits of these registers are always zero).

Stacks in the NS16000 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 NS16032 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 NS16032 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 (Sec. 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS16032 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 NS16032 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.

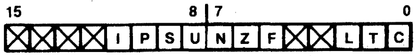


FIGURE 2-2. Processor Status Register.

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

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

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

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

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

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

U: If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When U = 0 the NS16032 is said to be in Supervisor Mode; when U = 1 the NS16032 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 (Sec. 3.8.5). It may have a setting of 0 (no trace pending) or 1 (trace pending).

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

2.1.3 The Configuration Register (CFG)

Within the Control section of the NS16032 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.

C	M	F	I

FIGURE 2-3. CFG Register.

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS16202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the $\overline{\text{INT}}$ pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Sec. 3.8.

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

2.1.4 Memory Organization

The main memory of the NS16032 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.

7 0
A

Byte at Address A

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

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

Word at Address A

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

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

Double Word at Address A

Although memory is addressed as bytes, it is actually organized as 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.

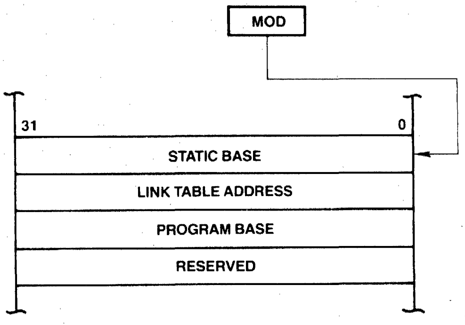
2.1.5 Dedicated Tables

Two of the NS16032 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 Sec. 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 NS16032. At any point in time, 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/C/5054-6

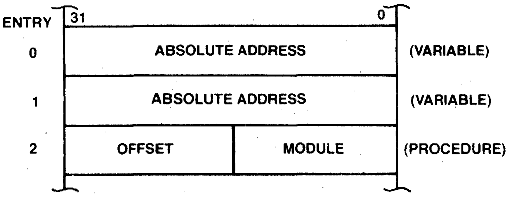
FIGURE 2-4. Module Descriptor Format.

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 NS16000 Programmer's Manual.



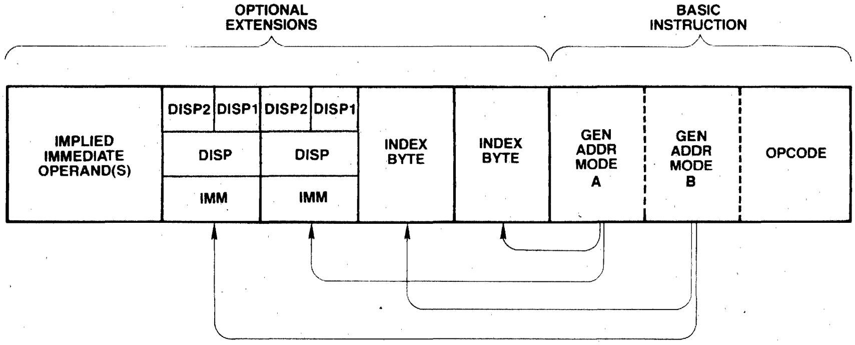
TL/C/5054-7

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 an NS16000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.



TL/C/5054-8

FIGURE 2-6. General Instruction Format.

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.

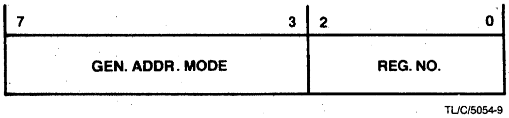


FIGURE 2-7. Index Byte Format.

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain one or 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 backward from the usual memory representation of data (Sec. 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 (Sec. 2.2.3).

2.2.2 Addressing Modes

The NS16032 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."

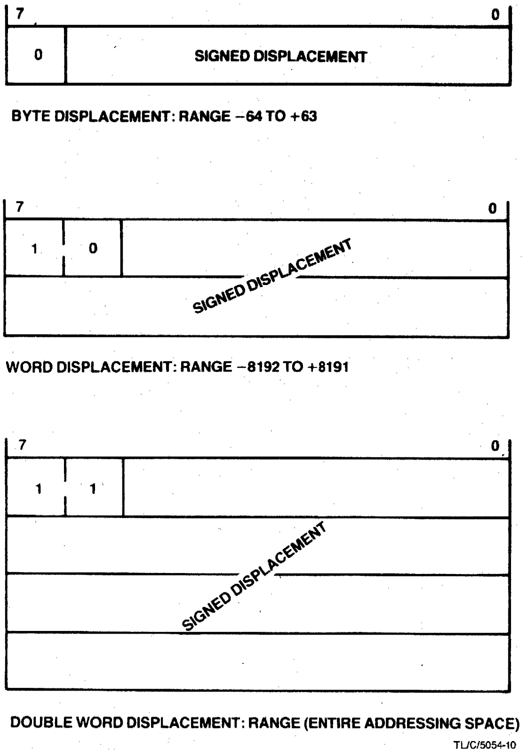
Addressing modes in the NS16032 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

NS16032 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.



BYTE DISPLACEMENT: RANGE −64 TO +63

WORD DISPLACEMENT: RANGE −8192 TO +8191

DOUBLE WORD DISPLACEMENT: RANGE (ENTIRE ADDRESSING SPACE)

FIGURE 2-8. Displacement Encodings.

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

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

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

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

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

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

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Programmer's Manual.

TABLE 2-1.
NS16032 Addressing Modes

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

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS16032 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 Programmer's 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.
NS16032 Instruction Set Summary

MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a 4-bit constant.
7	MOVMi	gen,gen,disp	Move Multiple: disp bytes.
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 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)

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

INTEGER COMPARISON

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

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

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

ARRAYS

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

STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 – Comparison Value
R3 – Translation Table Pointer
R2 – String 2 Pointer
R1 – String 1 Pointer
R0 – Limit Count

Options on all string instructions are:

B (Backward): Decrement string pointers after each step rather than incrementing.

U (Until match): End instruction if String 1 entry matches R4.

W (While match): End instruction if String 1 entry does not match R4.

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	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)

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	CCMPc	gen,gen	Custom Compare.
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 Functional Description

3.1 POWER AND GROUNDING

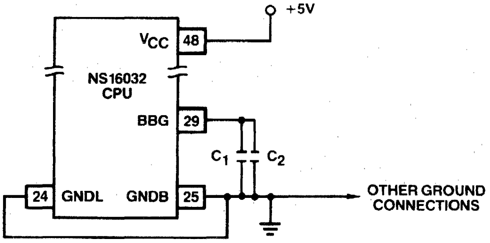
The NS16032 requires a single 5-volt power supply, applied on pin 48 (V_{CC}). See DC Specification Section.

Grounding connections are made on two pins. Logic Ground (GNDL, pin 24) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 25) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below (Figure 3-1).

In addition to V_{CC} and Ground, the NS16032 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (Fig. 3-1) from the BBG pin to ground. Recommended values for these are:

C_1: 1 μF, Tantalum.

C_2: 1000 pF, low inductance. This should be either a disc or monolithic ceramic capacitor.



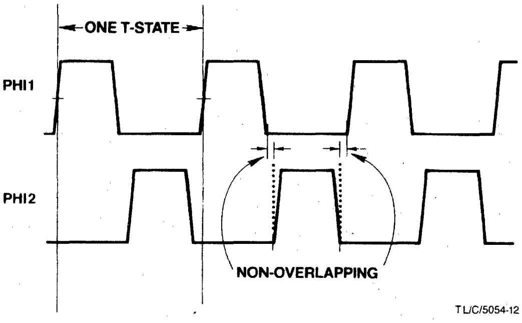
TL/C/5054-11

FIGURE 3-1. Recommended Supply Connections.

3.2 CLOCKING

The NS16032 inputs clocking signals from the NS16201 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 positive 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 the AC Specifications (Sec. 4) for complete specifications of PHI1 and PHI2.



TL/C/5054-12

FIGURE 3-2. Clock Timing Relationships.

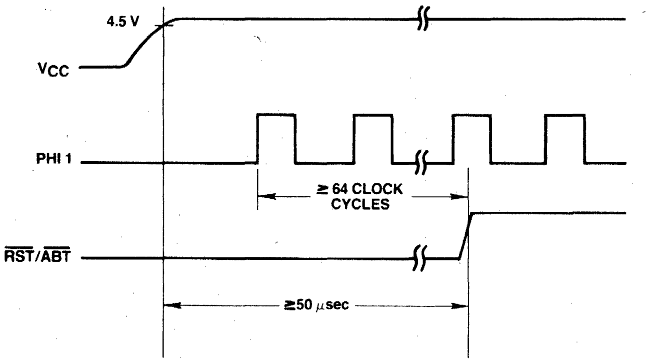
As the TCU presents signals with very fast transitions, it is recommended that the conductors carrying PHI1 and PHI2 be kept as short as possible, and that they not be connected anywhere except from the TCU to the CPU and, if present, the MMU. A TTL Clock signal (CTTL) is provided by the TCU for all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

3.3 RESETTING

The $\overline{\text{RST}}/\overline{\text{ABT}}$ pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Sec. 3.5.4.

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 μsec 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



TL/C/5054-13

FIGURE 3-3. Power-on Reset Requirements.

active for not less than 64 clock cycles. The trailing (positive-going) edge must occur while PHI1 is high, and no later than 10 ns before the PHI1 trailing edge. See Figures 3-3 and 3-4.

The NS16201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS16032 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.

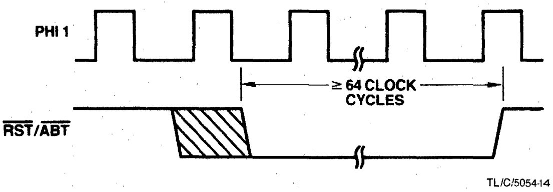


FIGURE 3-4. General Reset Timing.

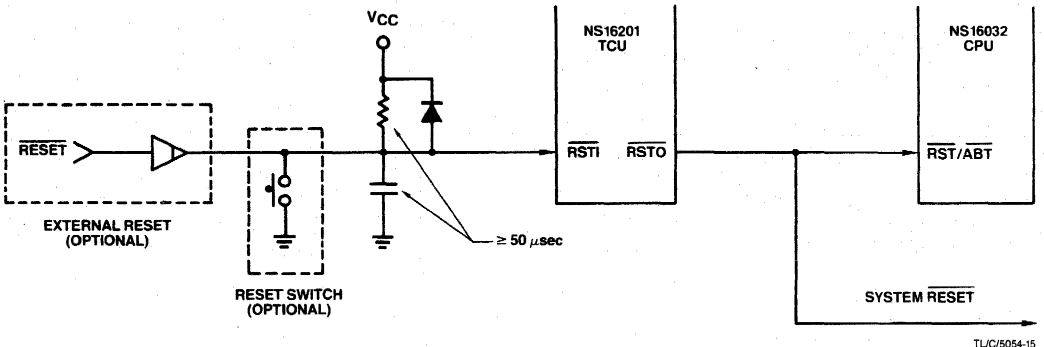


FIGURE 3-5a. Recommended Reset Connections, Non-Memory-Managed System.

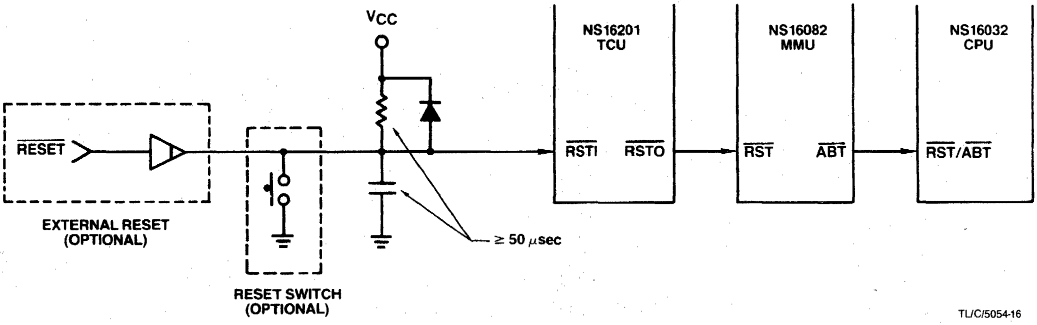


FIGURE 3-5b. Recommended Reset Connections, Memory-Managed System.

3.4 BUS CYCLES

The NS16032 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 Sec. 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 NS16000 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 Sec. 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 (Sec. 3.4.6).

The sequence of events in a non-Slave bus cycle is shown below 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 (Sec. 3.4.1).

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").

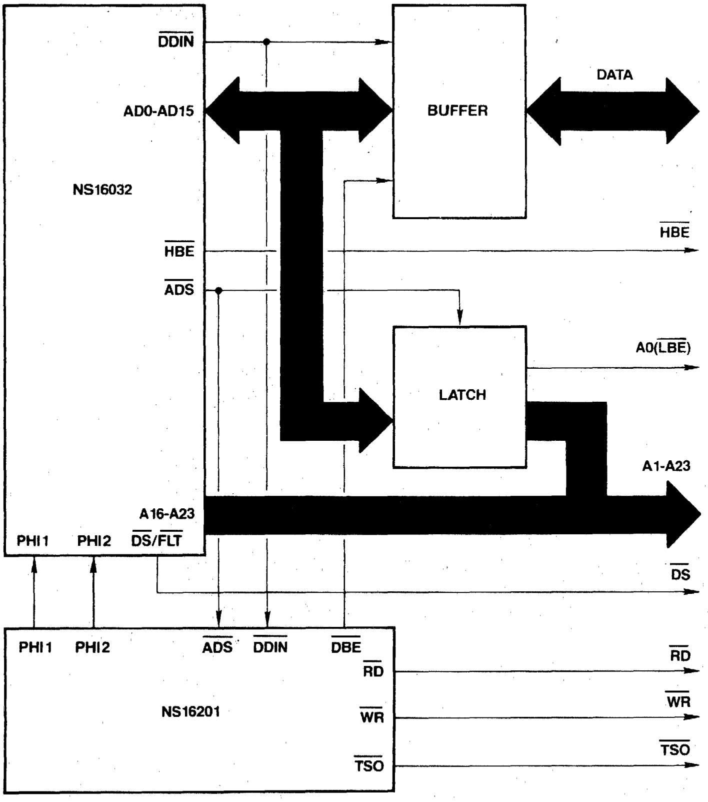
During T1, the CPU applies an address on pins AD0-AD15 and A16-A23. It also provides a low-going pulse on the ADS pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing Address bits 0-15 from the AD0-AD15 pins. See Figure 3-6. During this time also the status signals DDIN, indicating the direction of the transfer, and 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 (DS), signalling the beginning of the data transfer. Associated signals from the NS16201 Timing Control Unit are also activated at this time: RD (Read Strobe) or WR (Write Strobe), TSO (Timing State Output, indicating that T2 has been reached) and DBE (Data Buffer Enable).

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

If the CPU is performing a Read cycle, the Data Bus (AD0-AD15) is sampled at the falling edge of PHI2 of the last T3 state. See Timing Specification, Sec. 4. Data must, however, be held at least until the beginning of T4. DS and 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 DS, RD or WR, and TSO signals go inactive, and at the rising edge of PHI2, DBE goes inactive, having provided for necessary data hold times. Addresses (and Data during Write cycles) remain 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/C/5054-17

FIGURE 3-6. Bus Connections.

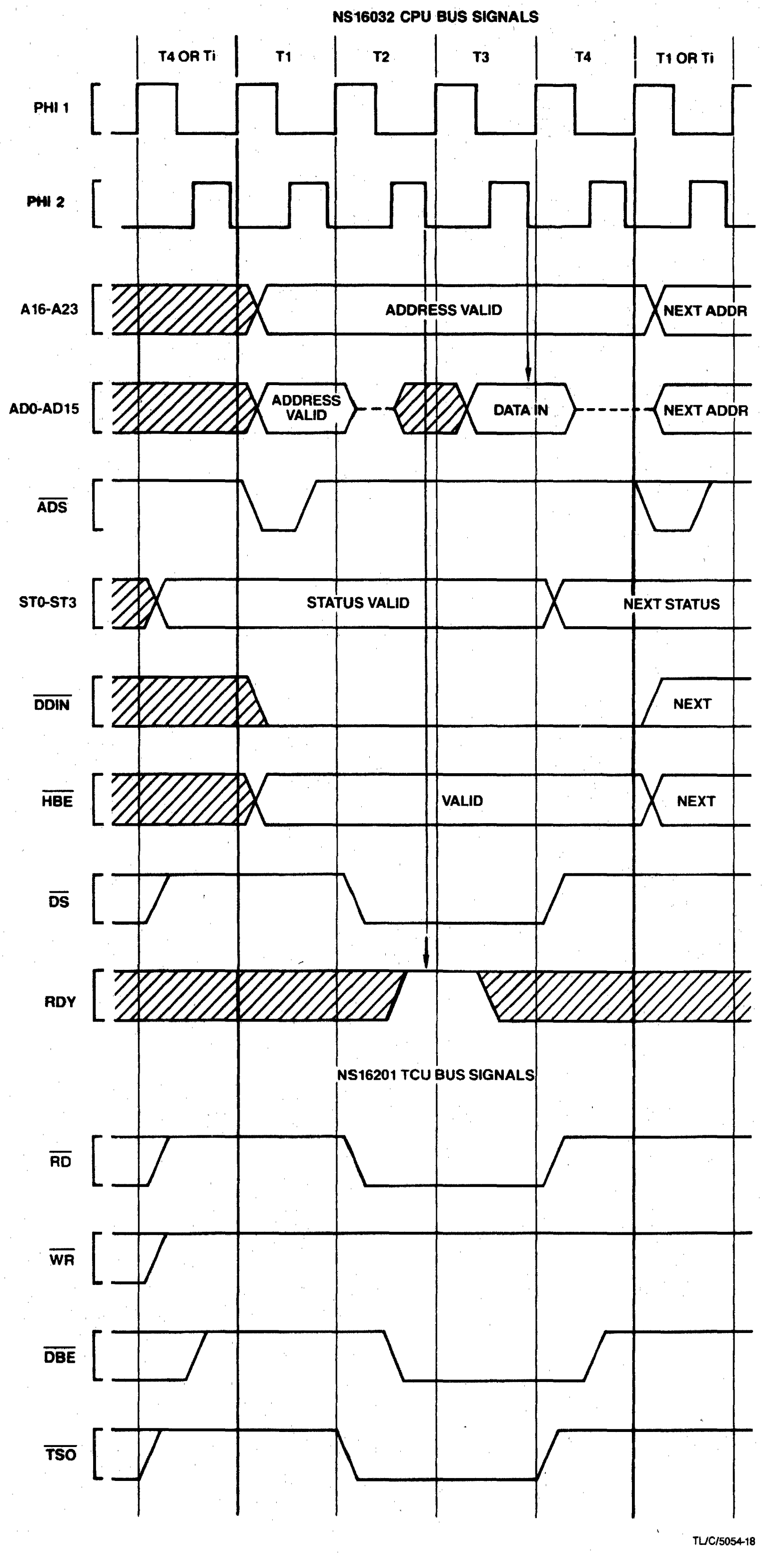
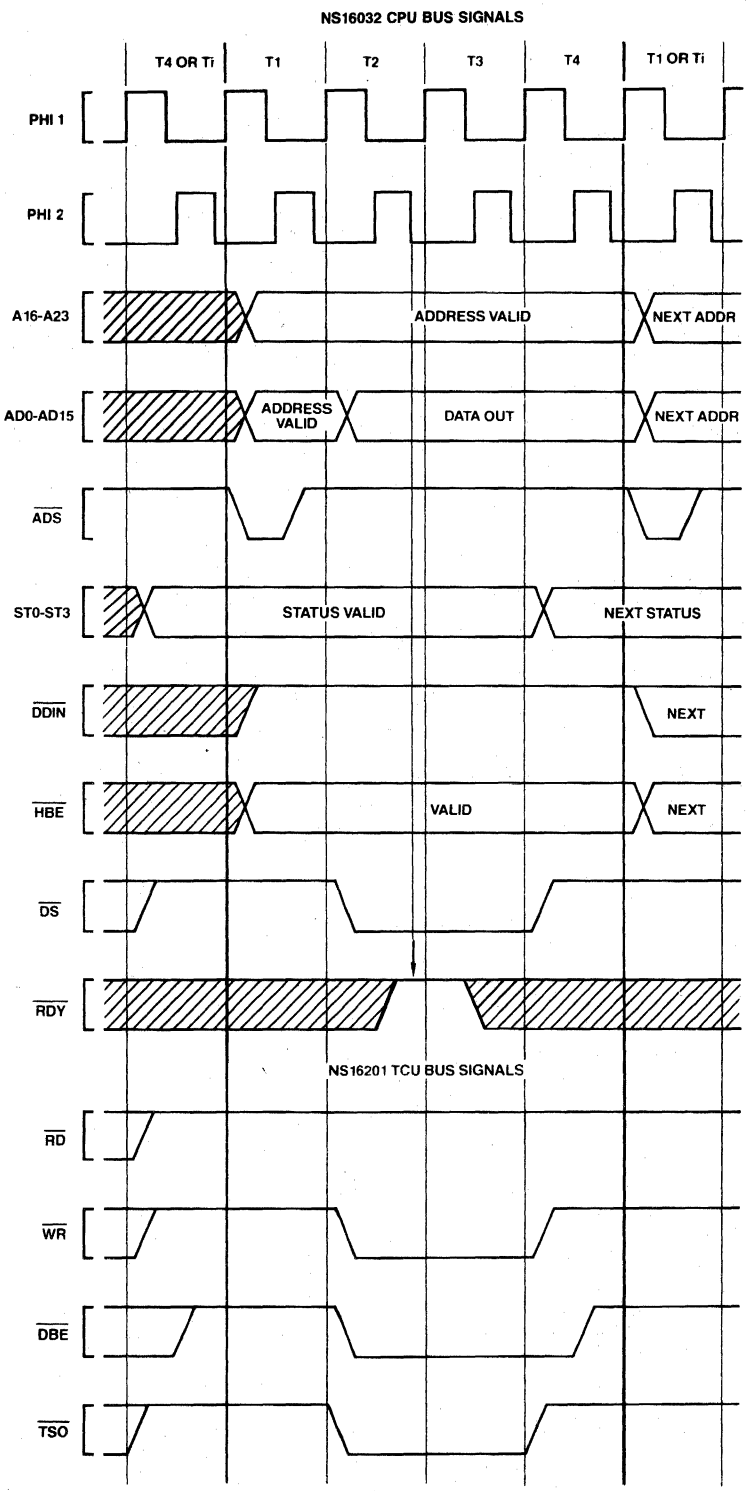


FIGURE 3-7. Read Cycle Timing.



TL/C/5054-19

FIGURE 3-8. Write Cycle Timing.

3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS16032 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 PHI 2, 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 PHI 2. 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 NS16201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pins:

1) $\overline{\text{CWAIT}}$ (Continuous WAIT), which holds the CPU in WAIT states until removed.

2) $\overline{\text{WAIT1}}$, $\overline{\text{WAIT2}}$, $\overline{\text{WAIT4}}$, $\overline{\text{WAIT8}}$ (Collectively $\overline{\text{WAITn}}$), 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 on their use, see the NS16201 Data Sheet.

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

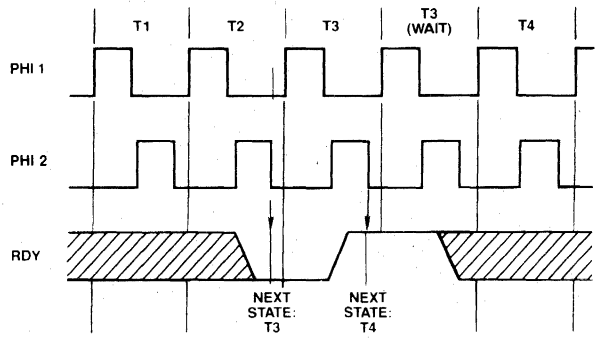


FIGURE 3-9. RDY Pin Timing.

3.4.2 Bus Status

The NS16032 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 yet need access to the bus.

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.

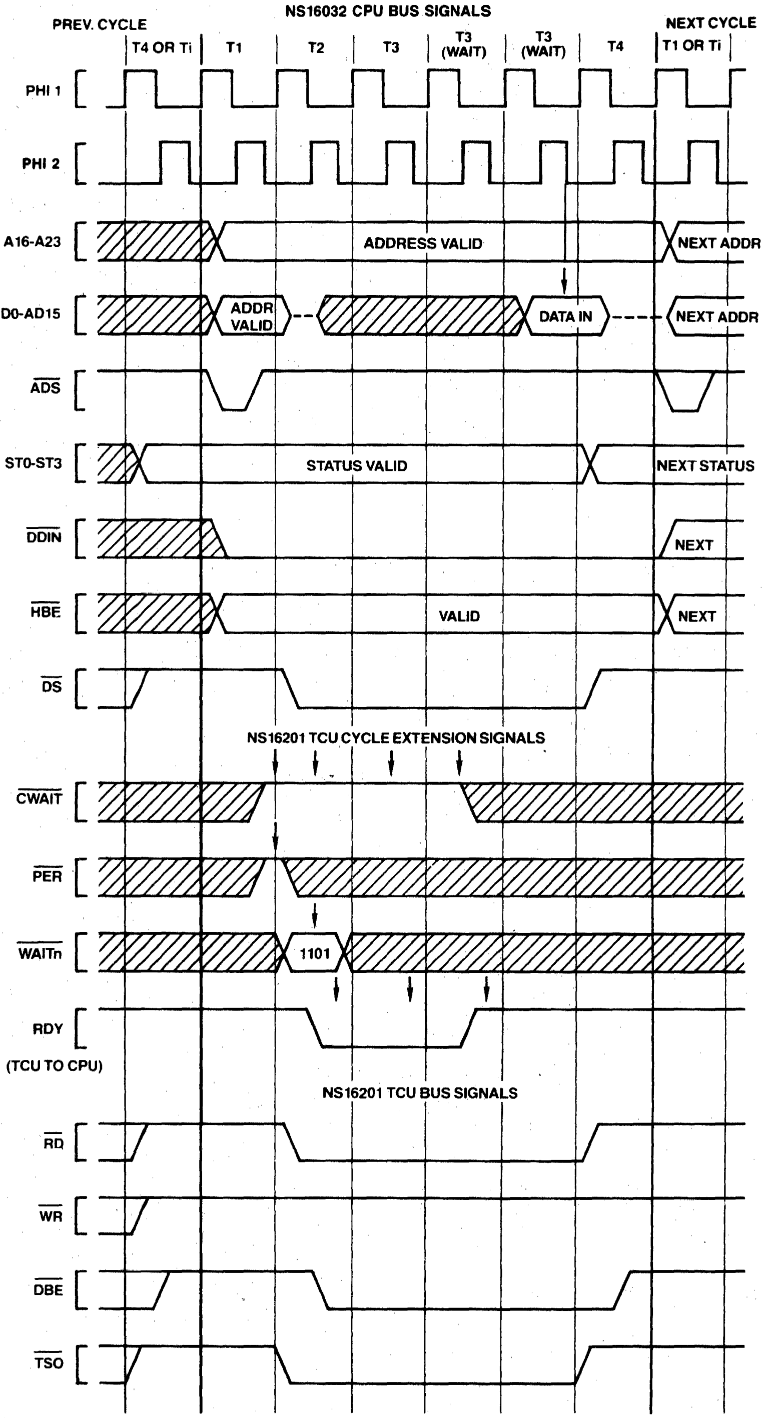
0100 – Interrupt Acknowledge, Master.
The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{\text{NMI}}$) it will read from address FFFF00_{16}, but will ignore any data provided.
To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$) it will read from address FFFE00_{16}, expecting a vector number to be provided from the Master NS16202 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 NS16202 is present. See Sec. 3.4.5.

0101 – Interrupt Acknowledge, Cascaded.
The CPU is reading a vector number from a Cascaded NS16202 Interrupt Control Unit. The address provided is the address of the NS16202 Hardware Vector register. See Sec. 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 Sec. 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 Sec. 3.4.5.

1000 – Sequential Instruction Fetch.
The CPU is reading the next sequential word from the instruction stream into the Instruction



TL/C/5054-21

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.

Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

1001 – Non-Sequential Instruction Fetch.
The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

1010 – Data Transfer.
The CPU is reading or writing an operand of an instruction.

1011 – Read RMW Operand.
The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.

1100 – Read for Effective Address Calculation.
The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

1101 – Transfer Slave Processor Operand.
The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Sec. 3.9.1.

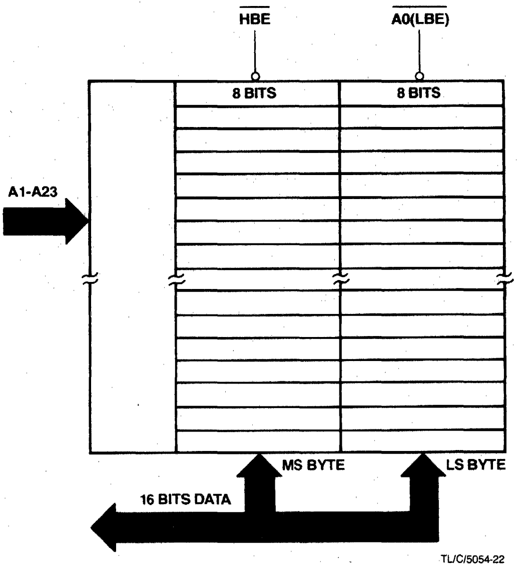
1110 – Read Slave Processor Status.
The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU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

1111 – Broadcast Slave ID.
The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Sec. 3.9.1.

3.4.3 Data Access Sequences

The 24-bit address provided by the NS16032 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 NS16032 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 NS16032 provides a special control signal, High Byte Enable ($\overline{\text{HBE}}$), which facilitates individual byte addressing on a 16-bit bus.

Memory is intended to be 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 $\overline{\text{HBE}}$ is low. See Figure 3-11.



TL/C/5054-22

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 states of A0 and $\overline{\text{HBE}}$ for each category.

Table 3-1.
Bus Cycle Categories

Category	$\overline{\text{HBE}}$	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 $\overline{\text{HBE}}$ see Sec. 3.4.

Table 3.2
Access Sequences

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus

A. Odd Word Access Sequence

BYTE 1	BYTE 0	← A

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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.

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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.

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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 operands which 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 NS16032 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 (Sec. 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 NS16032 interrupt structure, see Sec. 3.8.

Table 3-3
Interrupt Sequences

Cycle	Status	Address	$\overline{\text{DDIN}}$	$\overline{\text{HBE}}$	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 $\overline{\text{HBE}}$ 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 $\overline{\text{HBE}}$ low and reads the vector number from bits 8-15 of the Data Bus. The vector number may be in the range 0-255.

3.4.6 Slave Processor Communication

In addition to its use as the Address Translation strap (Sec. 3.5.1), the $\overline{\text{AT/SPC}}$ pin is used as the data strobe for Slave Processor transfers. In this role, it is referred to as Slave Processor Control ($\overline{\text{SPC}}$). In a Slave Processor bus cycle, data is transferred on the Data Bus (AD0-AD15), and the least significant two bits of CPU cycle status (ST0-ST1) are monitored by each Slave Processor in order to determine the type of transfer being performed. $\overline{\text{SPC}}$ is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Sec. 3.9 for full protocol sequences.

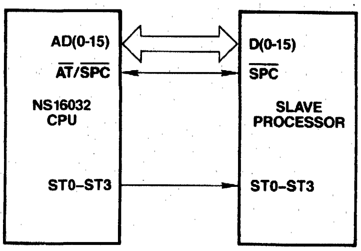
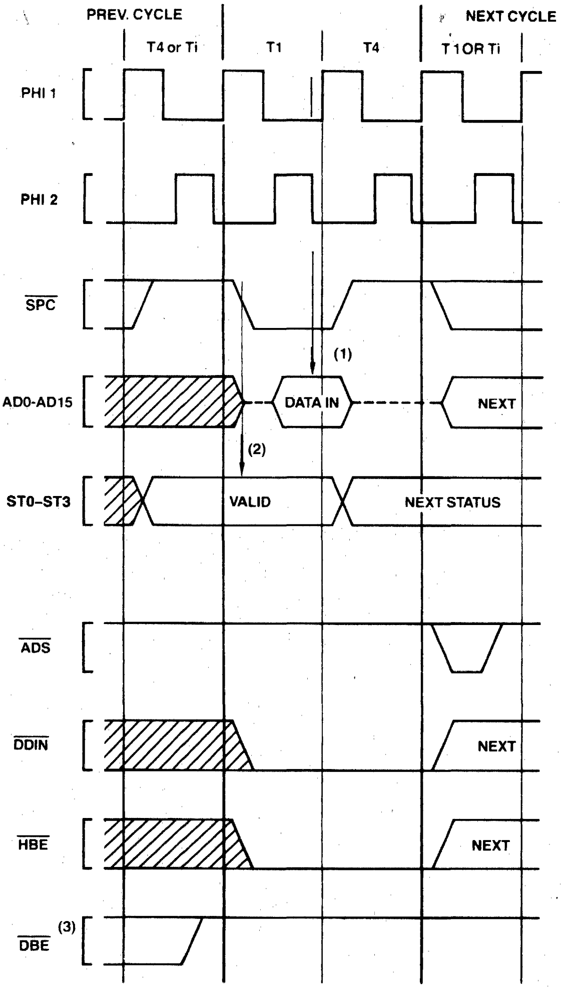


FIGURE 3-12. Slave Processor Connections.



NOTE:

(1) CPU samples Data Bus here.

(2) Slave Processor samples CPU Status here.

(3) $\overline{\text{DBE}}$ and all other NS16201 TCU bus signals remain inactive because no $\overline{\text{ADS}}$ pulse is received from the CPU.

FIGURE 3-13. CPU Read from Slave Processor.

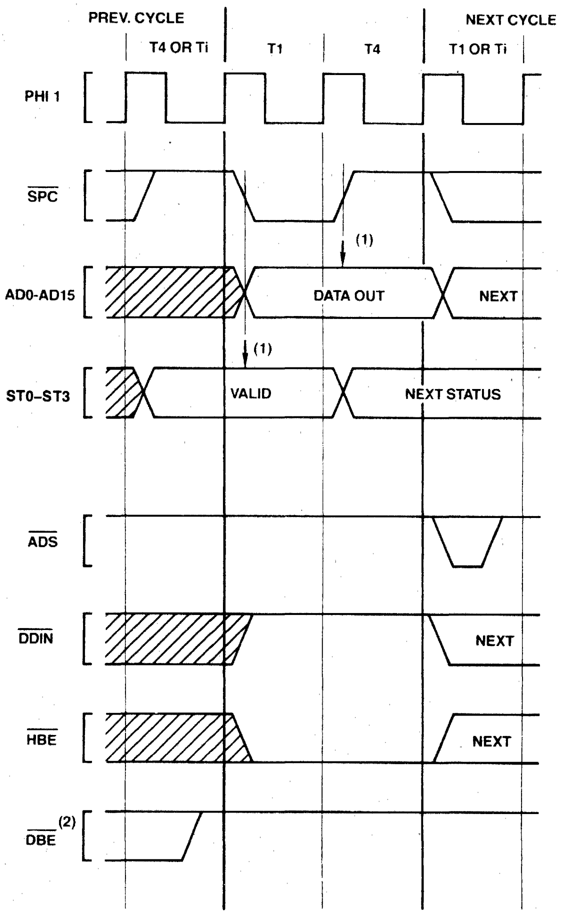
3.4.6.1 Slave Processor Bus Cycles

A Slave Processor bus cycle always takes exactly two clock cycles, labelled T1 and T4 (see Figures 3-13 and 3-14). During a Read cycle, $\overline{SPC}$ is activated at T1, data is sampled at T4, and $\overline{SPC}$ is removed. 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 NS16201 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 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.



NOTE:

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

(2) $\overline{DBE}$, being provided by the NS16201 TCU, remains inactive due to the fact that no pulse is presented on $\overline{ADS}$. TCU signals $\overline{RD}$, $\overline{WR}$ and $\overline{TSO}$ also remain inactive.

FIGURE 3-14. CPU Write to Slave Processor.

3.5 MEMORY MANAGEMENT OPTION

The NS16032 CPU, in conjunction with the NS16082 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

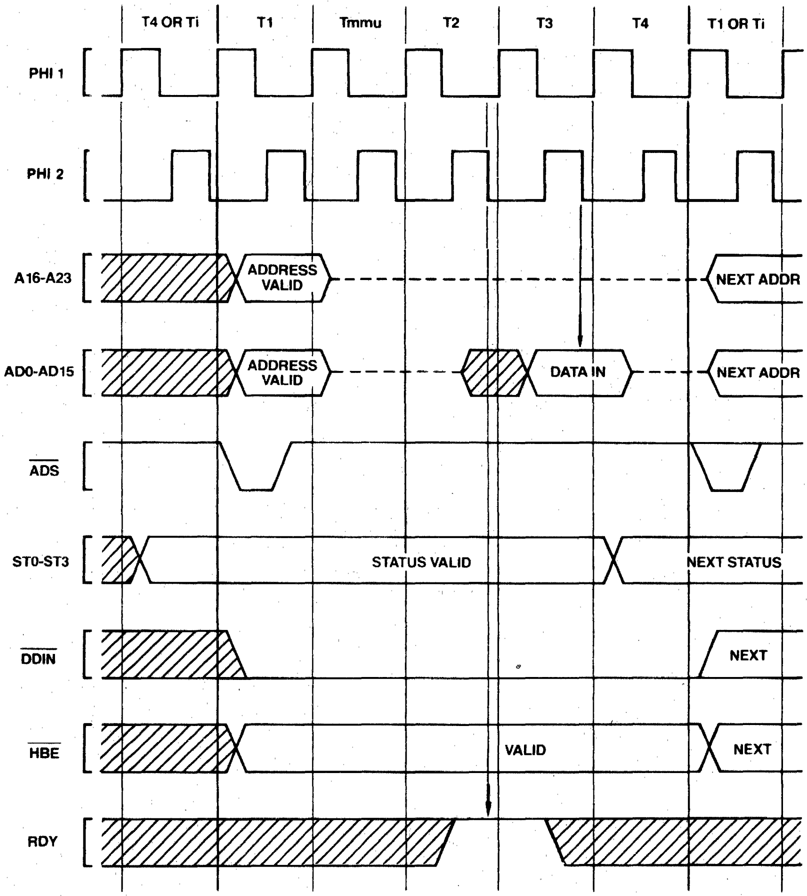
The Bus Interface Control section of the NS16032 CPU has two bus timing modes: With or Without Address Translation. The mode of operation is selected by the CPU by sampling the AT/SPC (Address Translation/Slave Processor Control) pin on the rising edge of the RST (Reset) pulse. If AT/SPC is sampled as high, the bus timing is as previously described in Sec. 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 DS/FLT pin changes in function from a Data Strobe output (DS) to a Float Command input (FLT).

The NS16082 MMU will itself pull the CPU AT/SPC pin low when it is reset, but this pin may be left floating in non-Memory-Managed systems.

Note that the Address Translation strap does not specifically declare the presence of an NS16082 MMU, but



TL/C/5054-26

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

only the 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 Sec. 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 $\overline{\text{PAV}}$. T2 through T4 of the cycle are identical to their counterparts 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.

Figures 3-17 and 3-18 show a Read cycle and a Write cycle as generated by the 16032/16082/16201 group. Note that with the CPU $\overline{\text{ADS}}$ signal going only to the MMU, and with the MMU $\overline{\text{PAV}}$ signal substituting for $\overline{\text{ADS}}$ everywhere else, Tmmu through T4 look exactly like T1 through T4 in a non-Memory-Managed system. For the connection diagram, see Appendix B.

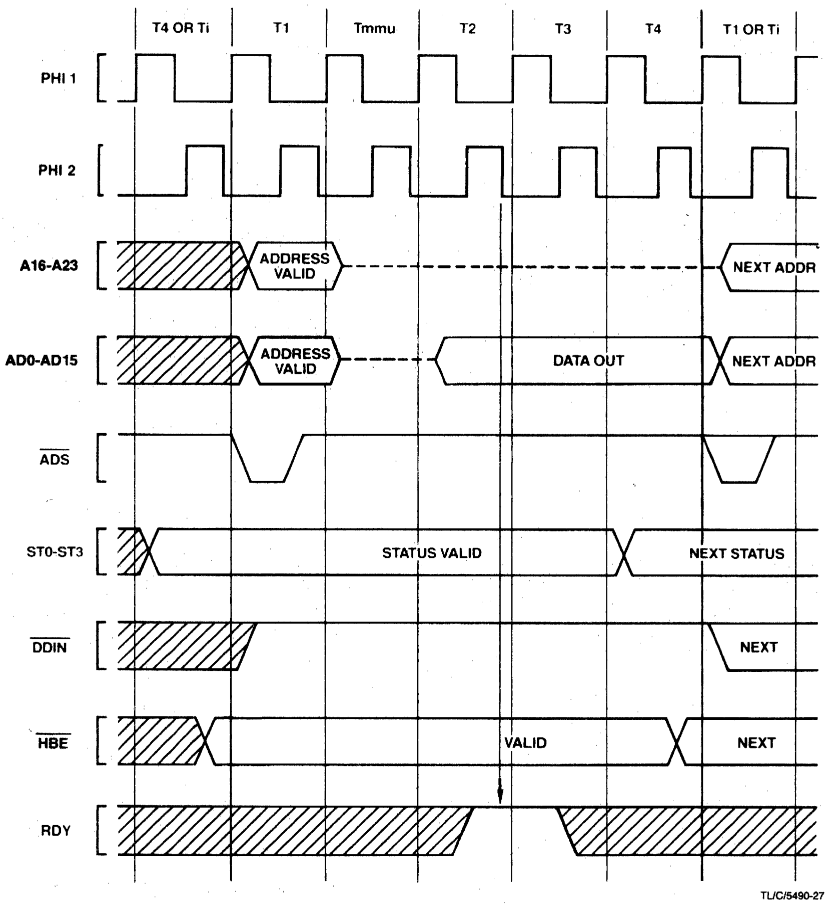
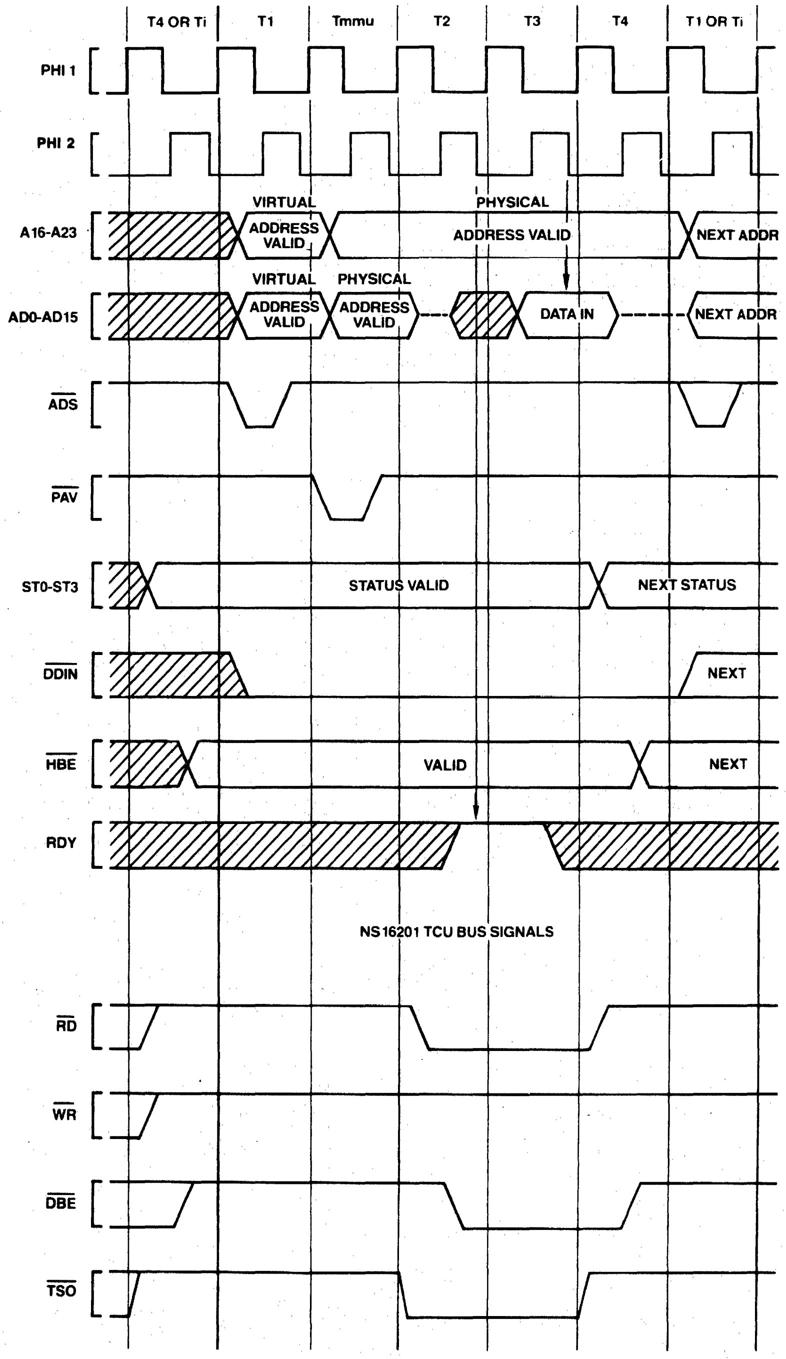


FIGURE 3-16. Write Cycle with Address Translation (CPU Action).



TL/C/5490-28

FIGURE 3-17. Memory-Managed Read Cycle.

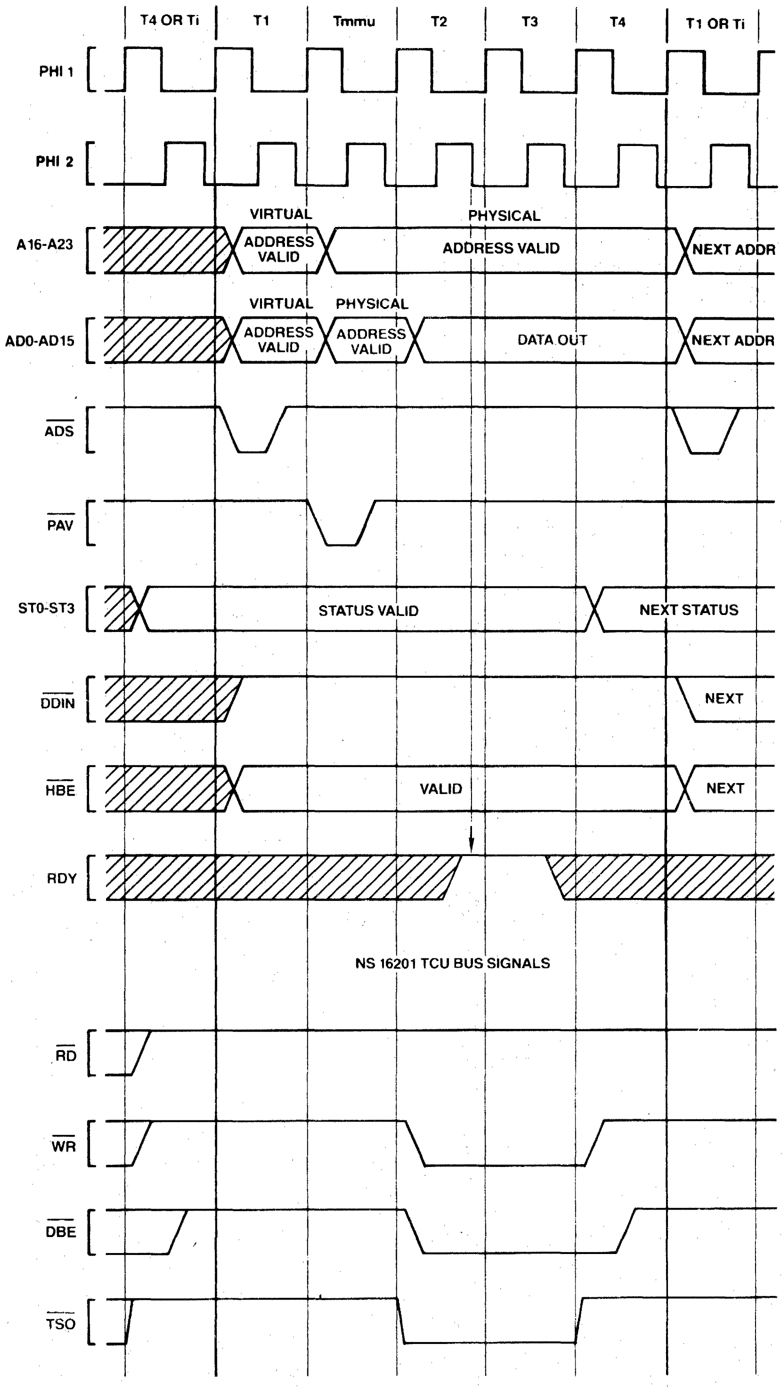


FIGURE 3-18. Memory-Managed Write Cycle.

]]>

at) Pin

n mode, the DS/FLT pin is treated d FLT (Float). Activating FLT during PU to wait longer than Tmmu for nd validation. This feature is used S16082 MMU in order to update its ache from page tables in memory, tatus bits within them.

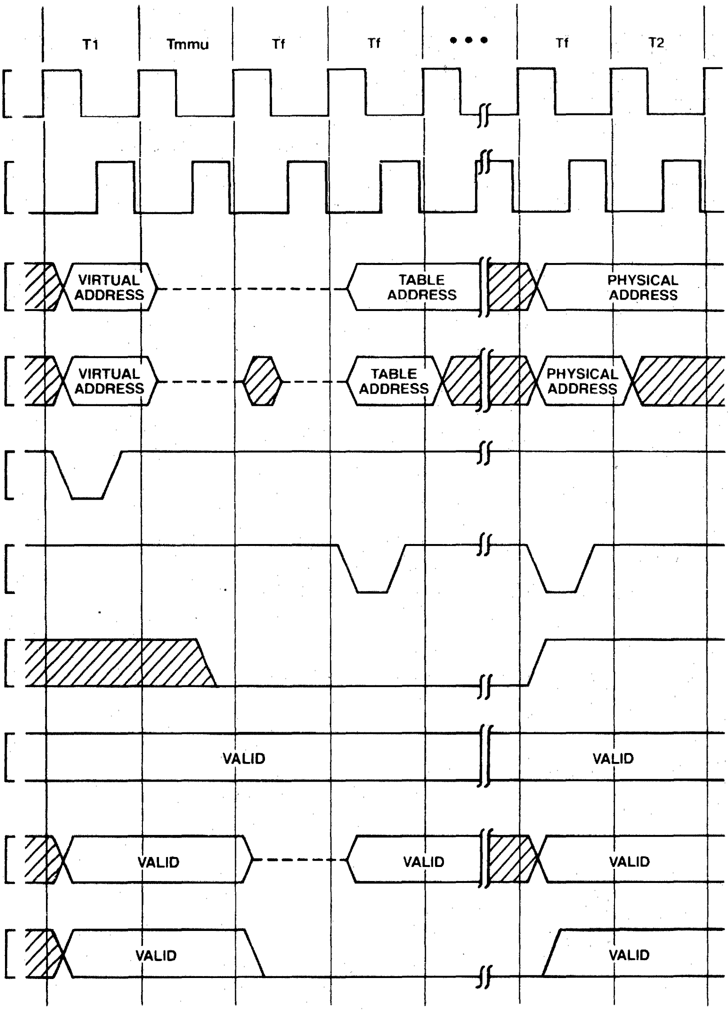
he effects of FLT. Upon sampling ı, the CPU enters idle T-States (Tf)

1) Sets AD0–AD15, A16–A23 and DDIN to the TRI-STATE condition ("floating").

2) Sets HBE low.

3) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See RST/ABT description, Sec. 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 FLT again goes high. See the Timing Specifications, Sec. 4.



TL/C/5054-30

FIGURE 3-19. FLT Float Command Timing.

3.5.4 Aborting Bus Cycles

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

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

The NS16082 MMU will abort a bus cycle for either of two reasons:

1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The referenced page must be brought into physical memory from mass storage to make it accessible to the CPU.

2) The CPU is attempting to perform an access which is not allowed due to the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction which caused it to occur is also aborted in such a manner that it is guaranteed re-executable later. Due to the NS16000 Family instruction set definition and its implementation in the NS16032 CPU, the only information which is changed irrecoverably by such partly-executed instructions is information which does not affect their 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 Sec. 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, such 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 which 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 NS16082 Memory Management Unit.

1) If $\overline{\text{FLT}}$ has not
pulse must oc
Timing Specifi

2) If $\overline{\text{FLT}}$ has bee
must be applie
inactive. The
Abort comma
4-23.

3) No bus cycle
of a Read-Mo
guarantees th
Memory Mana
RMW status (
half of the acc
status, that cyc
to write to any

If $\overline{\text{RST}}/\overline{\text{ABT}}$ is pu
cated above, it wil
under execution c
a very high-priori
which was runnin
erable, and shoul

3.6 BUS ACCE

The NS16032 CP
access to the bus
another CPU. Th
$\overline{\text{HOLD}}$ (Hold Req
pins. By asserting
access to the bu
the device may p
point has set the
$\overline{\text{HBE}}$ pins to the T
of the bus to the
and the CPU ack
$\overline{\text{HLDA}}$ inactive.

How quickly the
whether it is idl
request is made,
current bus cycle.
when the CPU is
bus during the im
3-21 shows the s
the time that the H
made during or b
cycles before T4
the clock cycle fo
to T4, the CPU
another bus cycl
until after the ne
also occur if the C
bus cycle interna

In a Memory-Ma
nected in a daisy
the MMU can rel

3.5.3 The FLT (Float) Pin

In Address Translation mode, the DS/FLT pin is treated as the input command FLT (Float). Activating FLT during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the NS16082 MMU in order to update its internal translation cache from page tables in memory, or to update certain status bits within them.

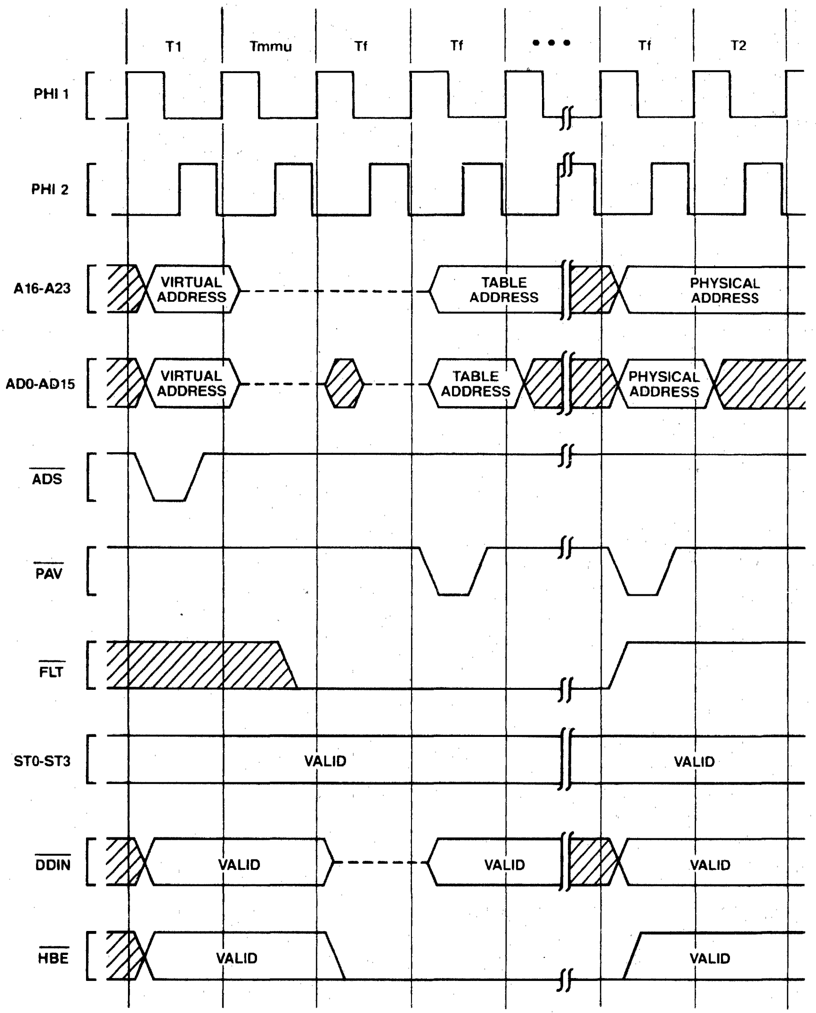
Figure 3-19 shows the effects of FLT. Upon sampling FLT low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

1) Sets AD0–AD15, A16–A23 and DDIN to the TRI-STATE condition ("floating").

2) Sets HBE low.

3) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See RST/ABT description, Sec. 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 FLT again goes high. See the Timing Specifications, Sec. 4.



TL/C/5054-30

FIGURE 3-19. FLT Float Command Timing.

3.5.4 Aborting Bus Cycles

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

If RST/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 PAV signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was even started.

The NS16082 MMU will abort a bus cycle for either of two reasons:

1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The referenced page must be brought into physical memory from mass storage to make it accessible to the CPU.

2) The CPU is attempting to perform an access which is not allowed due to the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction which caused it to occur is also aborted in such a manner that it is guaranteed re-executable later. Due to the NS16000 Family instruction set definition and its implementation in the NS16032 CPU, the only information which is changed irrecoverably by such partly-executed instructions is information which does not affect their 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 Sec. 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, such 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 which 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 NS16082 Memory Management Unit.

1) If FLT has not been applied to the CPU, the Abort pulse must occur during or before Tmmu. See the Timing Specifications, Figure 4-22.

2) If FLT has been applied to the CPU, the Abort pulse must be applied before the T-State in which FLT goes inactive. The CPU will not actually respond to the Abort command until FLT is removed. See Figure 4-23.

3) No bus cycle may be aborted which is the Write half of a Read-Modify-Write operand access. The CPU guarantees that this will never be necessary for Memory Management functions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

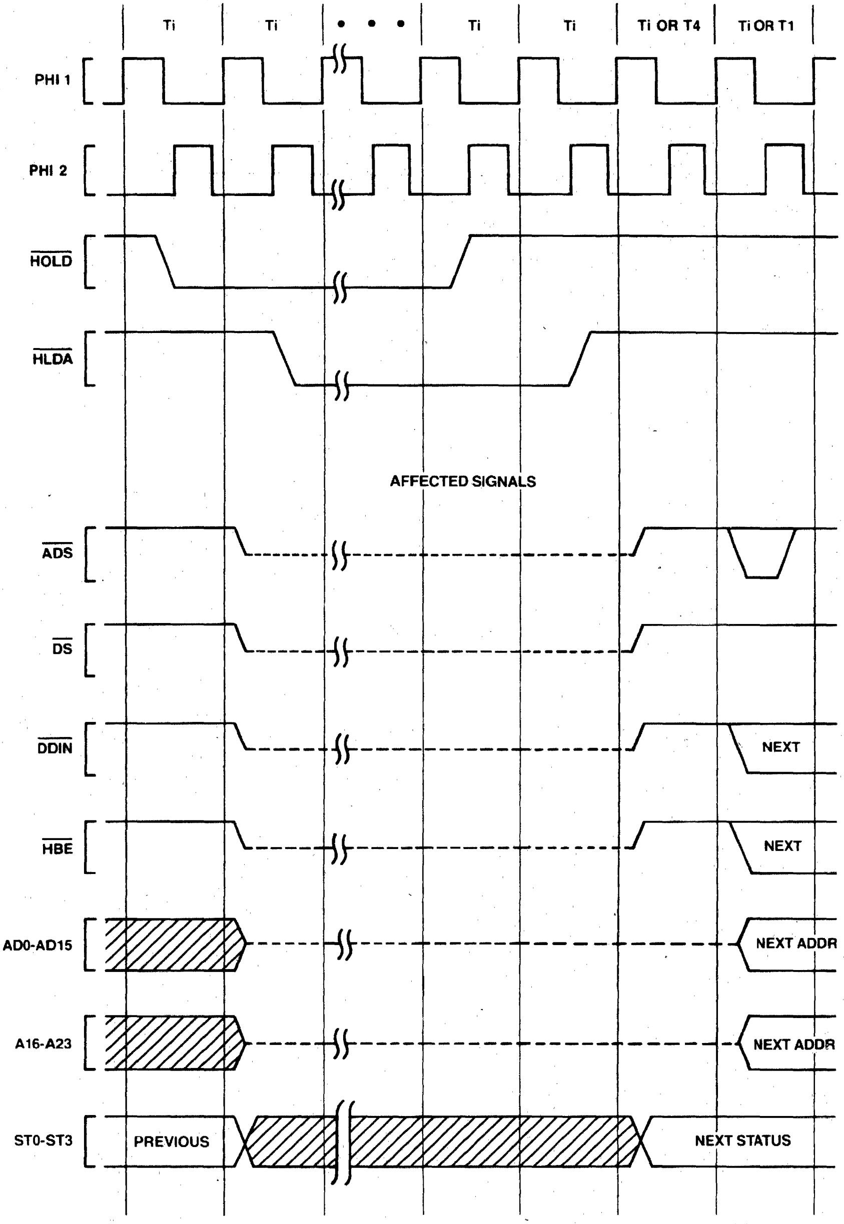
If RST/ABT is pulsed at any time other than as indicated above, it will abort either the instruction currently under execution or the next instruction and will act as a very high-priority interrupt. However, the program which was running at the time is not guaranteed recoverable, and should be terminated.

3.6 BUS ACCESS CONTROL

The NS16032 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 HOLD (Hold Request) and HLDA (Hold Acknowledge) pins. By asserting HOLD low, an external device requests access to the bus. On receipt of HLDA from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0-AD15, A16-A23, ADS, DDIN and HBE pins to the TRI-STATE® condition. To return control of the bus to the CPU, the device sets HOLD inactive, and the CPU acknowledges return of the bus by setting HLDA inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the 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 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 HLDA signal is connected in a daisy-chain through the NS16082, such that the MMU can release the bus if it is using it.



TL/C/5054-31

FIGURE 3-20. $\overline{\text{HOLD}}$ Timing, Bus Initially Idle.

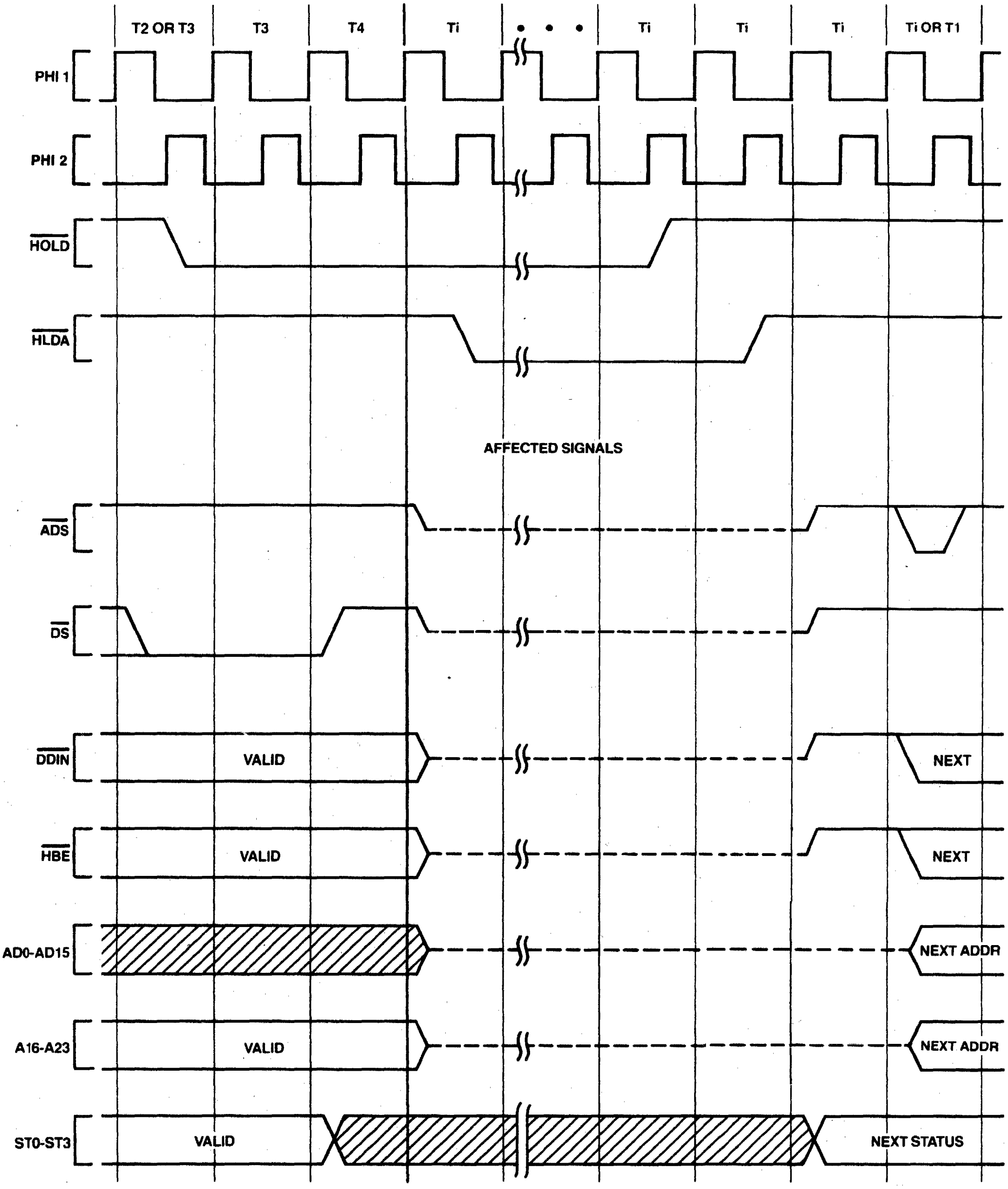


FIGURE 3-21. $\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle.

3.7 INSTRUCTION STATUS

In addition to the four bits of Bus Cycle status (ST0-ST3), the NS16032 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.

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 NS16082 Memory Management Unit.

U/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 4-21.

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 multiprocessor communication and resource sharing. As with the U/S pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification Section, Figure 4-19.

3.8 NS16032 INTERRUPT STRUCTURE

INT, on which maskable interrupts may be requested,

NMI, on which non-maskable interrupts may be requested, and

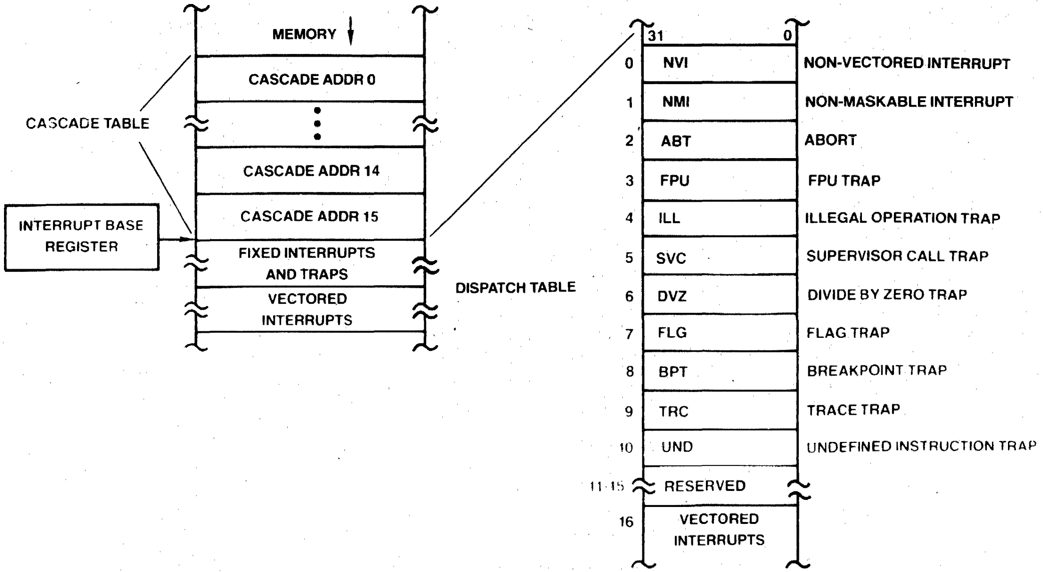
RST/ABT, which may be used to abort a bus cycle and any associated instruction. It generates an interrupt request if an instruction was aborted. See Sec. 3.5.4.

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 four 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) Saving Processor Status.
   The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.
3) Vector Acquisition.
   A Vector is either obtained from the Data Bus or is supplied by default.
4) 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 oh the Interrupt Stack.



TL/C/5054-33

FIGURE 3-22. Interrupt Dispatch and Cascade Tables.

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

Full sequences of events in processing interrupts and traps may be found as follows:

Interrupt on $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pin:	Sec. 3.8.7.1.
Abort Interrupt:	Sec. 3.8.7.4.
Traps (except Trace):	Sec. 3.8.7.2.
Trace Trap:	Sec. 3.8.7.3.

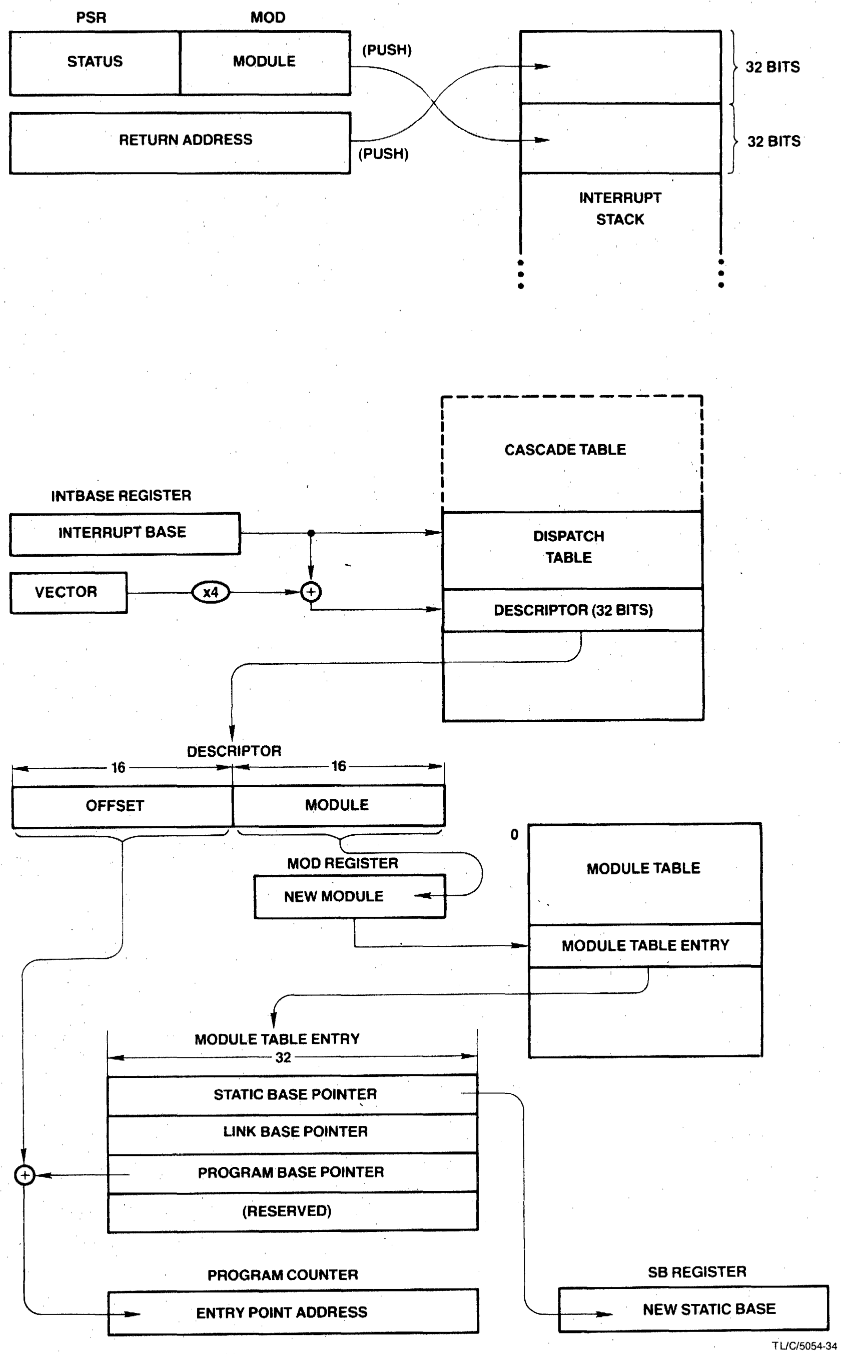


FIGURE 3-23. Interrupt/Trap Service Routine Calling Sequence.

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.

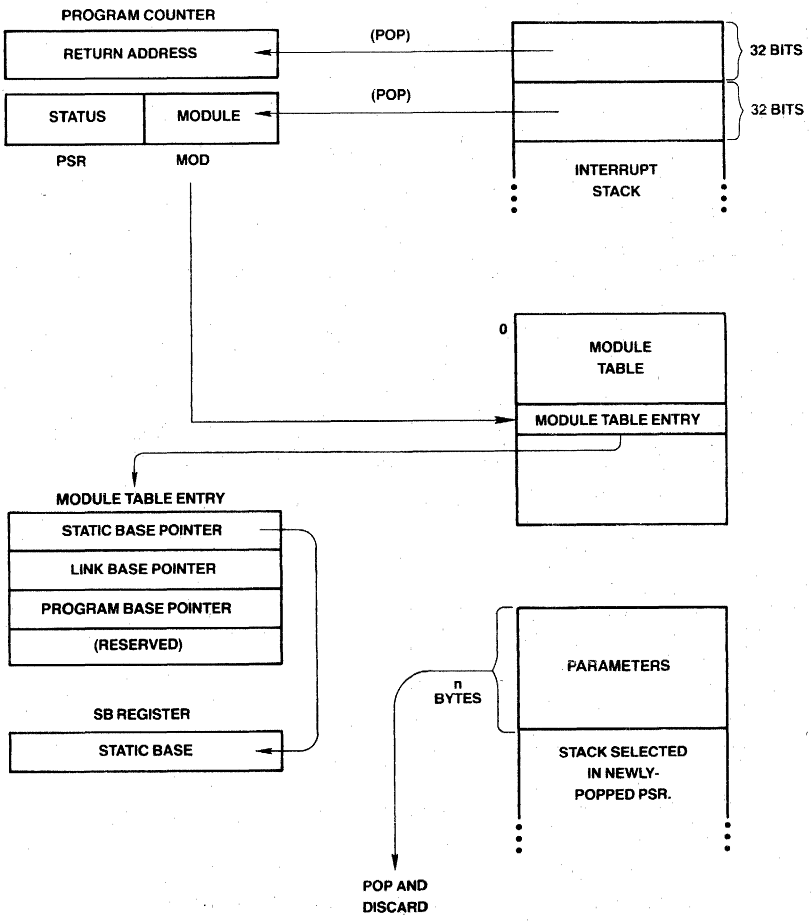
3.8.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)

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/C/5054-35

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

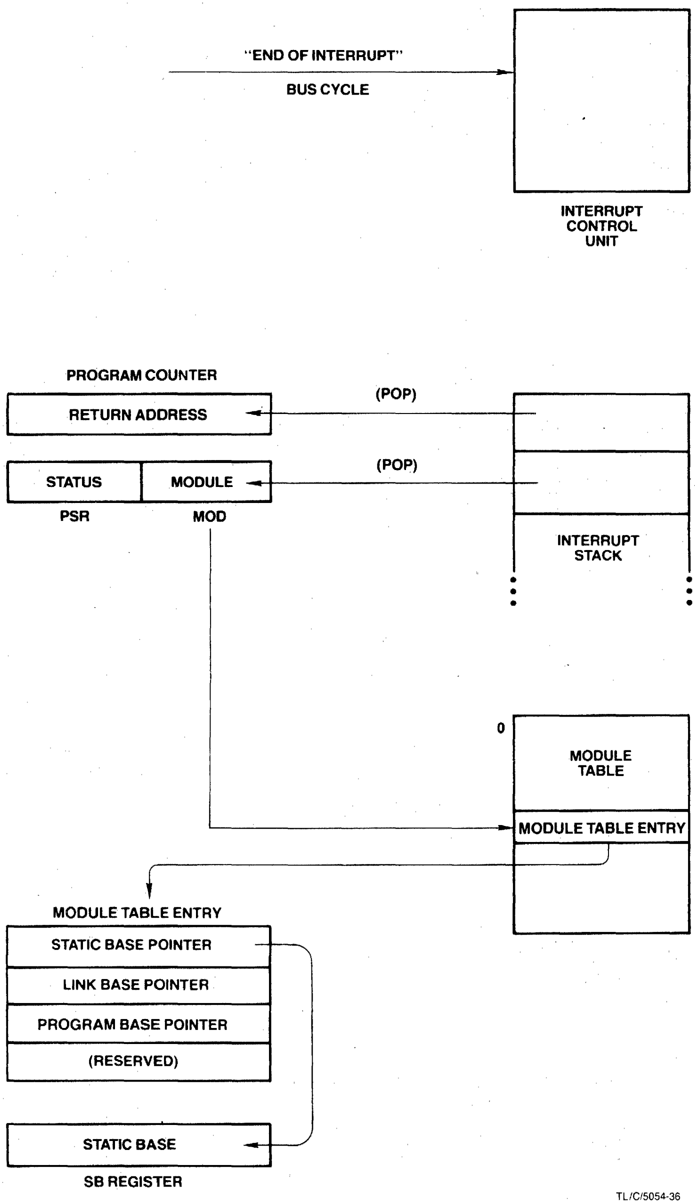


FIGURE 3-25. Return from Interrupt (RETI) Instruction Flow.

3.8.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an NS16202 Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Upon receipt of an interrupt request on the $\overline{\text{INT}}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.2) 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 NS16202 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 $\overline{\text{INT}}$ pin.

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

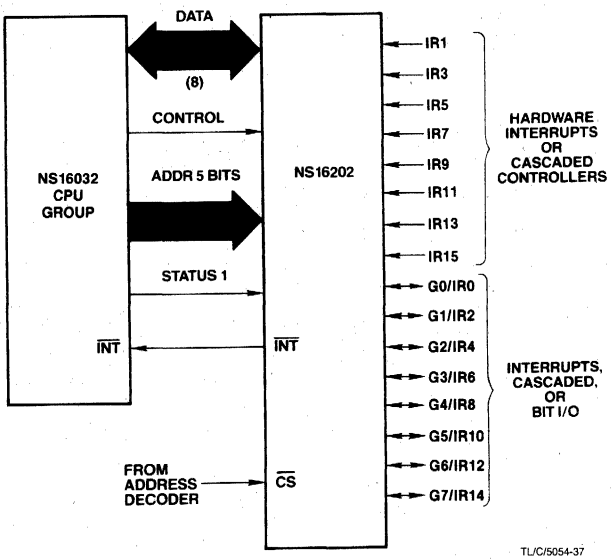
1) For each Cascaded ICU in the system, the Master ICU must be informed of the line number (0 to 15) on which it receives the cascaded requests.

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 (Sec. 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 (Sec. 3.4.2), whereupon the Master ICU again provides the negative Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Sec. 3.4.2), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.



TL/C/5054-37

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

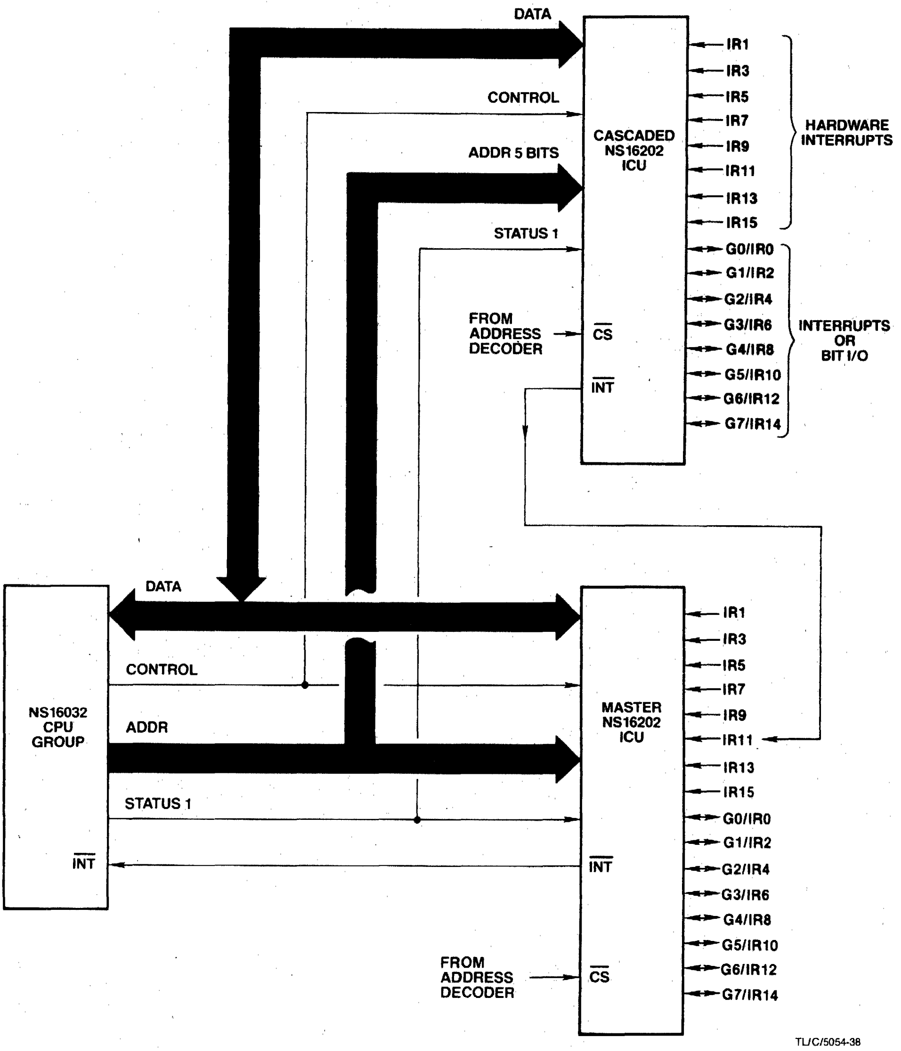


FIGURE 3-27. Cascaded Interrupt Control Unit Connections.

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 (Sec. 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 Sec. 3.8.7.1.

3.8.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) 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 NS16032 CPU are:

Trap (FPU): An exceptional condition was detected by the NS16081 Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Sec. 3.9.1).

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

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

Trap (DVZ): An attempt was made to divide an integer by zero. (The FPU 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 NS16032 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: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in Figure 3-28. 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 sequence followed in processing either Maskable or Non-Maskable interrupts (on the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pins, respectively), see Sec. 3.8.7.1. For Abort interrupts, see Sec. 3.8.7.4. For the Trace Trap, see Sec. 3.8.7.3, and for all other traps see Sec. 3.8.7.2.

3.8.7.1 Maskable/Non-Maskable Interrupt Sequence

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

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

   Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
   a. Read a byte from address 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) Push MOD Register onto the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)

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

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

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

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

6) 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.

TL/C/5054-39

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.

FPU:	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 NS16032 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 (Sec. 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

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, Sec. 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, Sec. 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

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).

TL/C/5054-40

FIGURE 3-29. Slave Processor Protocol.

solely responsible for memory accesses, these extensions are not sent to the Slave processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Sec. 3.4.2).

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

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

Upon receiving the pulse on $\overline{\text{SPC}}$, the CPU uses $\overline{\text{SPC}}$ to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Sec. 3.4.2). 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 FPU vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2).

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 Programmer's 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

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

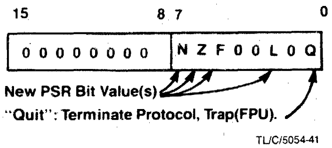


FIGURE 3-30. Slave Processor Status Word Format.

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 futher details of the Memory Management Instruction set, see the Programmer's Manual and the NS16082 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 Programmer's Manual and the NS16082 Memory Management Unit Data Sheet.

D = Double Word.
* = Privileged Instruction: will trap if CPU is in User Mode.
N/A = Not Applicable to this instruction.

3.9.4 Custom Slave Instructions

Provided in the NS16032 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
CCMPc	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to Op. 2	none
CATST0 *	addr	N/A	D	N/A	N/A	F
CATST1 *	addr	N/A	D	N/A	N/A	F
LCR *	read.D	N/A	D	N/A	N/A	none
SCR *	write.D	N/A	N/A	N/A	D to Op. 1	none

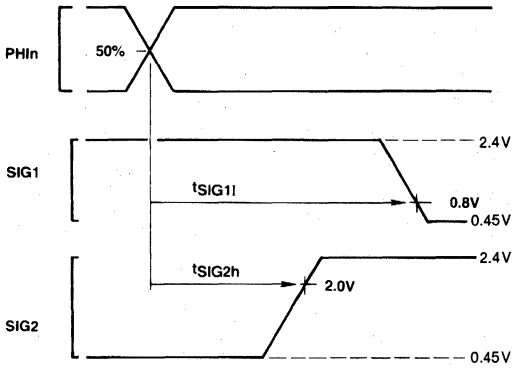
NOTE:
D = Double Word
i = integer size (B,W,D) specified in mnemonic.
f = Floating Point type (F,L) specified in mnemonic.
N/A = Not Applicable to this instruction.

4 AC Electrical Characteristics

4.1 Definitions

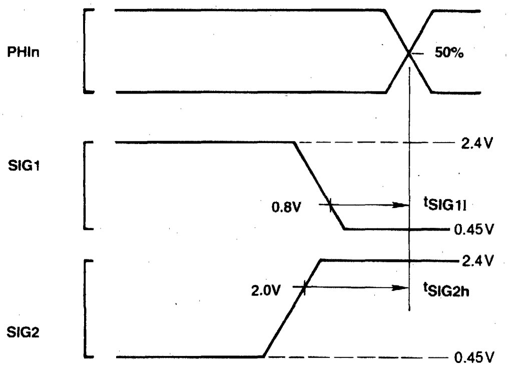
All the timing specifications given in this section refer to 50% of the leading or trailing edges of the appropriate clock phase and 0.8V or 2.0V on the appropriate signal as illustrated in *Figures 4-1* and *4-2*, unless specifically stated otherwise.

Abbreviations:
L.E — leading edge
T.E. — trailing edge



TL/C/5054-42

FIGURE 4-1. Timing Specification Standard (Signal Valid After Clock Edge)



TL/C/5054-43

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

4.2 Timing Tables

4.2.1 Output Signals: Internal Propagation Delays, NS16032-4, NS16032-6

Maximum times assume capacitive loading of 100 pF.

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{ALv}	Address bits 0–15 valid	4-3	after L.E., PHI1 T1			80	ns
t_{ALh}	Address bits 0–15 hold	4-3	after L.E., PHI1 Tmmu or T2	0			ns
t_{Dv}	Data valid (write cycle)	4-3	after L.E., PHI1 T2			80	ns
t_{Dh}	Data hold (write cycle)	4-3	after L.E., PHI1 next T1 or Ti	0			ns
t_{AHv}	Address bits 16–23 valid	4-3	after L.E., PHI1 T1			95	ns
t_{AHh}	Address bits 16–23 hold	4-3	after L.E., PHI1 next T1 or Ti	0			ns
t_{ALADSs}	Address bits 0–15 set up to $\overline{ADS}$ T.E.	4-4	before $\overline{ADS}$ reaches 2.0V	20			ns
t_{AHADSs}	Address bits 16–23 set up to $\overline{ADS}$ T.E.	4-4	before $\overline{ADS}$ reaches 2.0V	20			ns
t_{ALADSh}	Address bits 0–15 hold from $\overline{ADS}$ T.E.	4-9	after $\overline{ADS}$ reaches 2.0V	10			ns
t_{AHADSh}	Address bits 16–23 hold from $\overline{ADS}$ T.E.	4-8	after $\overline{ADS}$ reaches 2.0V	10			ns
t_{ALf}	Address bits 0–15 floating (no MMU)	4-4	after L.E., PHI1 T2			25	ns
t_{ALMf}	Address bits 0–15 floating (with MMU)	4-8	after L.E., PHI1 Tmmu			25	ns
t_{AHMf}	Address bits 16–23 floating (with MMU)	4-8	after L.E., PHI1 Tmmu			25	ns
t_{HBEv}	$\overline{HBE}$ signal valid	4-3	after L.E., PHI1 T1			95	ns
t_{HBEh}	$\overline{HBE}$ signal hold	4-3	after L.E., PHI1 next T1 or Ti	0			ns
t_{STv}	Status (ST0–ST3) valid	4-3	after L.E., PHI1 T4 (before T1, see note)			90	ns
t_{STh}	Status (ST0–ST3) hold	4-3	after L.E., PHI1 T4 (after T1)	0			ns
t_{DDINv}	$\overline{DDIN}$ signal valid	4-4	after L.E., PHI1 T1			110	ns
t_{DDINh}	$\overline{DDIN}$ signal hold	4-4	after L.E., PHI1 next T1 or Ti	0			ns
t_{ADSa}	$\overline{ADS}$ signal active (low)	4-3	after L.E., PHI1 T1			55	ns
t_{ADSia}	$\overline{ADS}$ signal inactive	4-3	after T.E., PHI1 T1			60	ns
t_{ADSw}	$\overline{ADS}$ pulse width	4-3	at 0.8V, both edges	60			ns
t_{DSa}	$\overline{DS}$ signal active (low)	4-3	after L.E., PHI1 T2			70	ns

4.2.1 Output Signals: Internal Propagation Delays, NS16032-4, NS16032-6 (continued)

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{DSia}	$\overline{DS}$ signal inactive	4-3	after L.E., PHI1 T4			60	ns
t_{ALf}	AD0-AD15 floating (caused by $\overline{HOLD}$)	4-5	after L.E., PHI1 T1			100	ns
t_{AHf}	A16-A23 floating (caused by $\overline{HOLD}$)	4-5	after L.E., PHI1 T1			100	ns
t_{ADSf}	$\overline{ADS}$ floating (caused by $\overline{HOLD}$)	4-5	after L.E., PHI1 Ti			100	ns
t_{HBEf}	$\overline{HBE}$ floating (caused by $\overline{HOLD}$)	4-5	after L.E., PHI1 Ti			100	ns
t_{DDINf}	$\overline{DDIN}$ floating (caused by $\overline{HOLD}$)	4-5	after L.E., PHI1 Ti			100	ns
t_{HLDAa}	$\overline{HLDA}$ signal active (low)	4-5	after L.E., PHI1 Ti			100	ns
t_{HLDAia}	$\overline{HLDA}$ signal inactive	4-7	after L.E., PHI1 Ti			100	ns
t_{ADSr}	$\overline{ADS}$ signal returns from floating (caused by $\overline{HOLD}$)	4-7	after L.E., PHI1 Ti			100	ns
t_{HBEr}	$\overline{HBE}$ signal returns from floating (caused by $\overline{HOLD}$)	4-7	after L.E., PHI1 Ti			100	ns
t_{DDINr}	$\overline{DDIN}$ signal returns from floating (caused by $\overline{HOLD}$)	4-7	after L.E., PHI1 Ti			100	ns
t_{ALf}	AD0-AD15 floating (caused by $\overline{FLT}$)	4-8	after L.E., PHI1 Tf			60	ns
t_{DDINf}	$\overline{DDIN}$ signal floating (caused by $\overline{FLT}$)	4-8	after $\overline{FLT}$ reaches 0.8V			80	ns
t_{HBEl}	$\overline{HBE}$ signal low (caused by $\overline{FLT}$)	4-8	after $\overline{FLT}$ reaches 0.8V			100	ns
t_{DDINr}	$\overline{DDIN}$ signal returns from floating (caused by $\overline{FLT}$)	4-9	after $\overline{FLT}$ reaches 2.0V			75	ns
t_{HBEr}	$\overline{HBE}$ signal returns from floating (caused by $\overline{FLT}$)	4-9	after $\overline{FLT}$ reaches 2.0V			90	ns
t_{SPCa}	$\overline{SPC}$ output active (low)	4-12	after L.E., PHI1 T1			50	ns
t_{SPCia}	$\overline{SPC}$ output inactive	4-12	after L.E., PHI1 T4			50	ns
t_{SPCnf}	$\overline{SPC}$ output nonforcing	4-14	after L.E., PHI2 T4			40	ns
t_{Dv}	Data valid (slave processor write)	4-12	after L.E., PHI1 T1			80	ns
t_{Dh}	Data hold (slave processor write)	4-12	after L.E., PHI1 next T1 or Ti	0			ns
t_{PFSw}	$\overline{PFS}$ pulse width	4-17	at 0.8V, both edges	70			ns
t_{PFSa}	$\overline{PFS}$ pulse active (low)	4-17	after L.E., PHI2			70	ns
t_{PFSia}	$\overline{PFS}$ pulse inactive	4-17	after L.E., PHI2			70	ns
t_{ILOs}	$\overline{ILO}$ signal setup	4-19a	before L.E., PHI1 T1 of first interlocked write cycle	0			ns
t_{ILOh}	$\overline{ILO}$ signal hold	4-19b	after L.E., PHI1 T3 of last interlocked read cycle	0			ns
t_{ILOa}	$\overline{ILO}$ signal active (low)	4-20	after L.E., PHI1			70	ns
t_{ILOia}	$\overline{ILO}$ signal inactive	4-20	after L.E., PHI1			70	ns
t_{USs}	U/$\overline{S}$ signal setup	4-21	before T.E., PHI1 T4 or Ti	15			ns
t_{USh}	U/$\overline{S}$ signal hold	4-21	after L.E., PHI1 T1	2			t_{Cp}
t_{NSPF}	Nonsequential fetch to next $\overline{PFS}$ clock cycle	4-18b	after L.E., PHI1 T1	4			t_{Cp}
t_{PFNS}	$\overline{PFS}$ clock cycle to next nonsequential fetch	4-18a	before L.E., PHI1 T1	4			t_{Cp}
t_{LXPF}	Last operand transfer of an instruction to next $\overline{PFS}$ clock cycle	4-28	before L.E., PHI1 T1 of first bus cycle of transfer	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...".

4.2.2 Input Signal Requirements: NS16032-4, NS16032-6

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{PWR}	Power stable to $\overline{\text{RST}}$ T.E.	4-24	after V_{CC} reaches 4.5V	50			μs
t_{DIs}	Data in setup (read cycle)	4-4	before T.E., PHI2 T3	20			ns
t_{DIh}	Data in hold (read cycle)	4-4	after T.E., PHI2 T3	10			ns
t_{HLDa}	$\overline{\text{HOLD}}$ active (low) setup time (See note)	4-5	before T.E., PHI2 TX1	25			ns
t_{HLDia}	$\overline{\text{HOLD}}$ inactive setup time	4-7	before T.E., PHI2 Ti	25			ns
t_{HLDh}	$\overline{\text{HOLD}}$ hold time	4-5	after L.E., PHI1 TX2	0			ns
t_{FLTa}	$\overline{\text{FLT}}$ active (low) setup time	4-8	before T.E., PHI2 Tmmu	25			ns
t_{FLTia}	$\overline{\text{FLT}}$ inactive setup time	4-9	before T.E., PHI2 T2	25			ns
t_{RDYs}	RDY setup time	4-10, 4-11	before T.E., PHI2 T2 or T3	25			ns
t_{RDYh}	RDY hold time	4-10, 4-11	after T.E., PHI1 T3	0			ns
t_{ABTs}	$\overline{\text{ABT}}$ setup time ($\overline{\text{FLT}}$ inactive)	4-22	before T.E., PHI2 Tmmu	30			ns
t_{ABTs}	$\overline{\text{ABT}}$ setup time ($\overline{\text{FLT}}$ active)	4-23	before T.E., PHI2 T2	30			ns
t_{ABTh}	$\overline{\text{ABT}}$ hold time	4-22	after L.E., PHI1	0			ns
t_{RSTs}	$\overline{\text{RST}}$ setup time	4-24, 4-25	before T.E., PHI1	20			ns
t_{RSTw}	$\overline{\text{RST}}$ pulse width	4-25	at 0.8V (both edges)	64			t_{Cp}
t_{INTs}	$\overline{\text{INT}}$ setup time	4-26	before T.E., PHI1	20			ns
t_{NMIw}	$\overline{\text{NMI}}$ pulsewidth	4-27	at 0.8V (both edges)	40			ns
t_{DIs}	Data setup (slave read cycle)	4-13	before T.E., PHI2 T1	20			ns
t_{DIh}	Data hold (slave read cycle)	4-13	after T.E., PHI2 T1	10			ns
t_{SPCw}	$\overline{\text{SPC}}$ pulse width (from slave processor)	4-12	at 0.8V (both edges)	30			ns
t_{ATs}	AT/$\overline{\text{SPC}}$ setup for address translation strap	4-15	before L.E., PHI1 of cycle during which $\overline{\text{RST}}$ pulse is removed	1			t_{Cp}
t_{ATh}	AT/$\overline{\text{SPC}}$ hold for address translation strap	4-15	after T.E., PHI1 of cycle during which $\overline{\text{RST}}$ pulse is removed	2			t_{Cp}

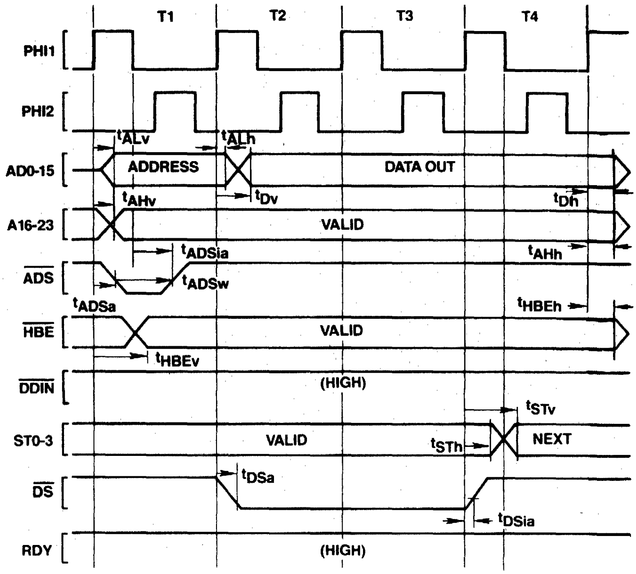
NOTE:
This setup time is necessary to ensure prompt acknowledgement via $\overline{\text{HLDA}}$ and the ensuing floating of CPU off the buses. Note that the time from the receipt of the $\overline{\text{HOLD}}$ signal until the CPU floats is a function of the time $\overline{\text{HOLD}}$ signal goes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

4.2.3 Clocking Requirements: NS16032-4

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{CLr}	PHI1, PHI2 rise time	4-16	to V_{CH} (see page 2)			9	ns
t_{CLf}	PHI1, PHI2 fall time	4-16	from 90-10% of V_{CH} (see page 2)			9	ns
t_{CLh}	PHI1, PHI2 high time	4-16		0.4			t_{Cp}
t_{CLl}	PHI1, PHI2 low time	4-16		0.35			t_{Cp}
t_{Cp}	Clock period	4-16		240		5000	ns
t_{OVL}	Non-overlap time	4-16	at 10% of V_{CH} (see page 2)	0			ns

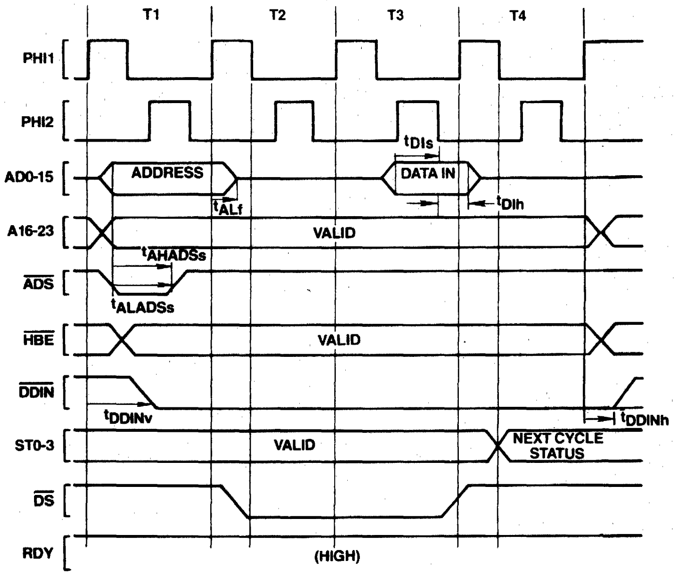
4.2.4 Clocking Requirements: NS16032-6

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{CLr}	PHI1, PHI2 rise time	4-16	to V_{CH} (see page 2)			9	ns
t_{CLf}	PHI1, PHI2 fall time	4-16	from 90-10% of V_{CH} (see page 2)			9	ns
t_{CLh}	PHI1, PHI2 high time	4-16		0.4			t_{Cp}
t_{CLl}	PHI1, PHI2 low time	4-16		0.35			t_{Cp}
t_{Cp}	Clock period	4-16		160		5000	ns
t_{OVL}	Non-overlap time	4-16	at 10% of V_{CH} (see page 2)	0			ns



TL/C/5054-44

FIGURE 4-3. Write Cycle.



TL/C/5054-45

FIGURE 4-4. Read Cycle.

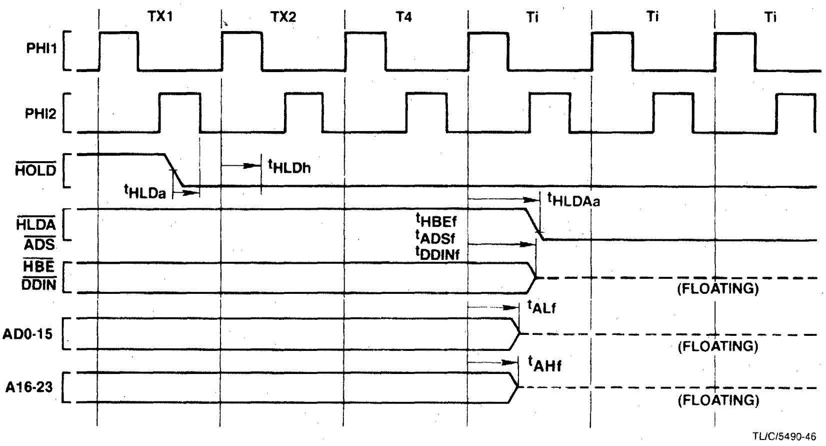


FIGURE 4-5. Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially).

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

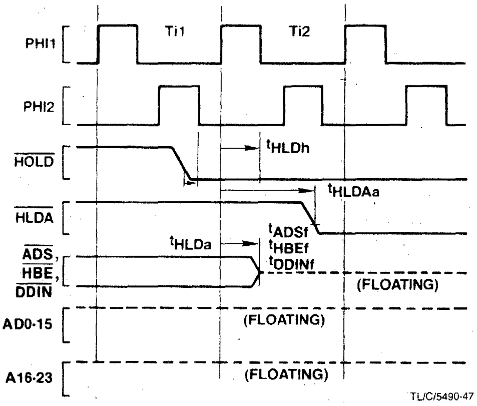


FIGURE 4-6. Floating by $\overline{\text{HOLD}}$ Timing (CPU Initially Idle).

Note that during Ti1 the CPU is already idling.

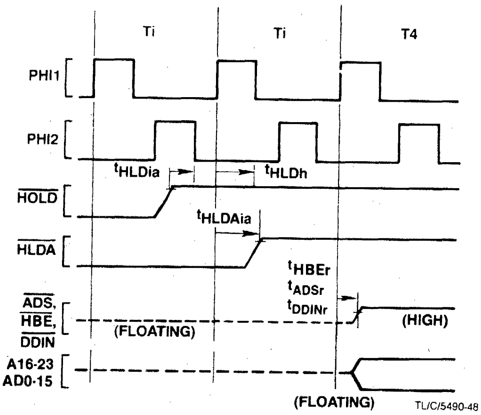
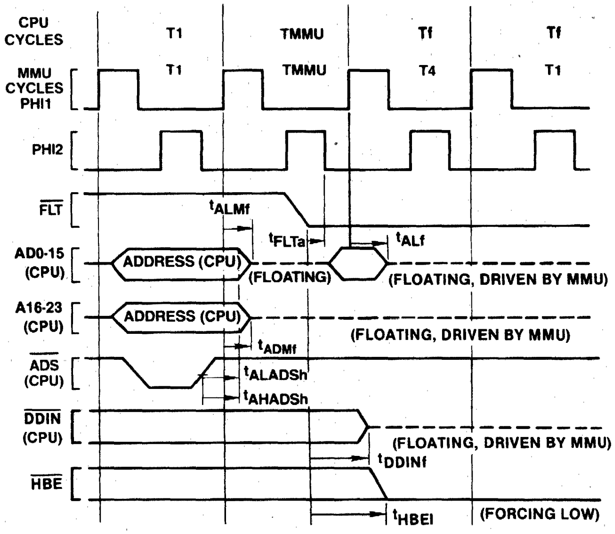
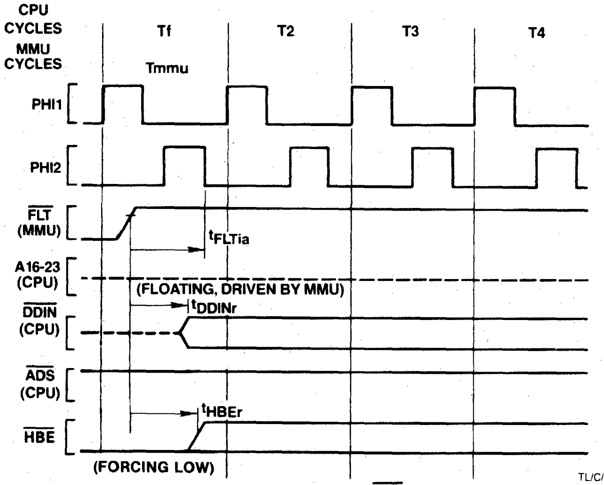


FIGURE 4-7. Release from $\overline{\text{HOLD}}$.



TL/C/5490-49

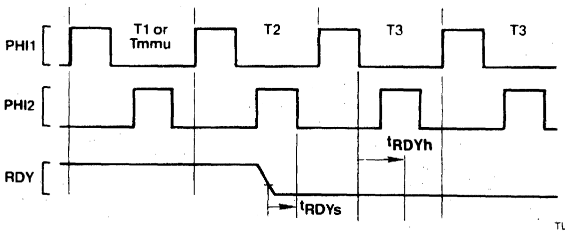
FIGURE 4-8. $\overline{\text{FLT}}$ Initiated Float Cycle Timing.



TL/C/5490-50

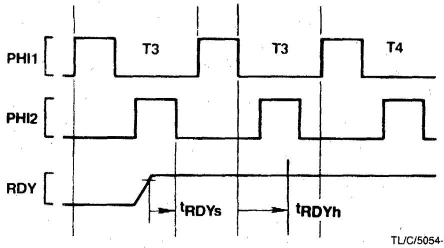
FIGURE 4-9. Release from $\overline{\text{FLT}}$ Timing.

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



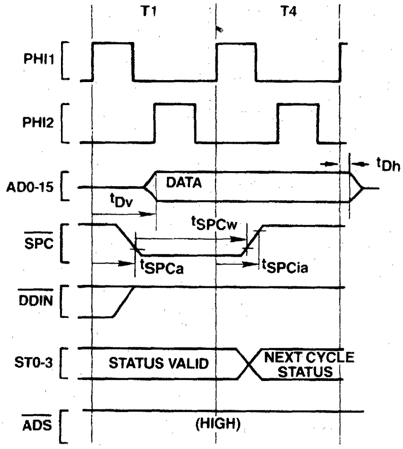
TL/C/5490-51

FIGURE 4-10. Ready Sampling (CPU Initially READY).



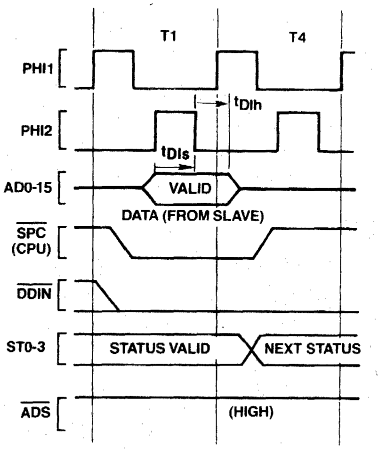
TL/C/5054-52

FIGURE 4-11. Ready Sampling (CPU Initially NOT READY).



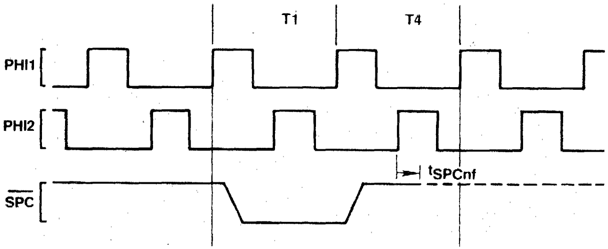
TL/C/5054-53

FIGURE 4-12. Slave Processor Write Timing.



TL/C/5054-54

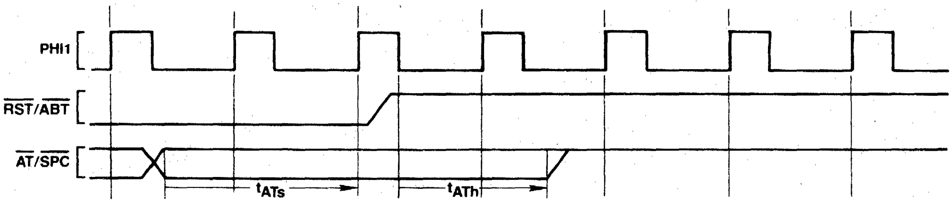
FIGURE 4-13. Slave Processor Read Timing.



TL/C/5054-55

FIGURE 4-14. $\overline{\text{SPC}}$ Non-Forcing Delay.

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


PHI1
$\overline{\text{RST}}/\overline{\text{ABT}}$
$\overline{\text{AT}}/\overline{\text{SPC}}$
t_{ATs}
t_{ATh}

TL/C/5054-56

FIGURE 4-15. Reset Configuration Timing.

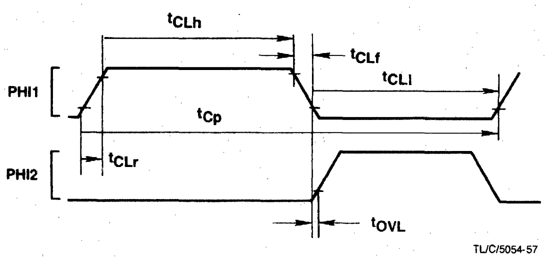


FIGURE 4-16. Clock Waveforms.

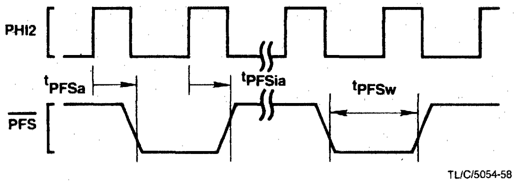


FIGURE 4-17. Relationship of PFS to Clock Cycles.

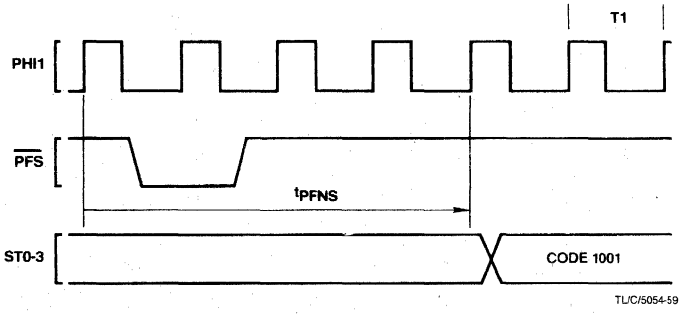


FIGURE 4-18a. Guaranteed Delay, PFS to Non-Sequential Fetch.

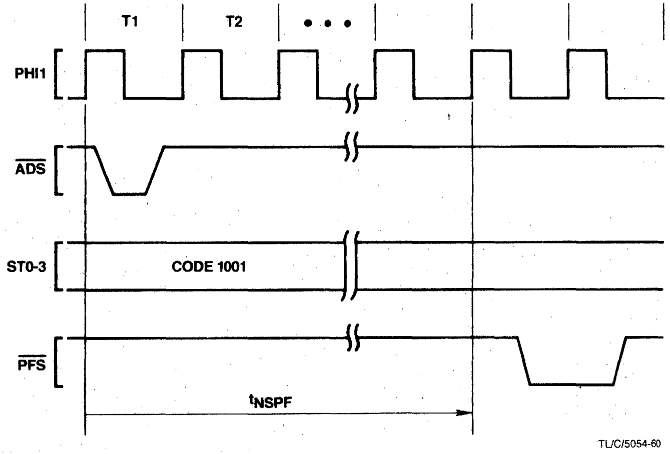
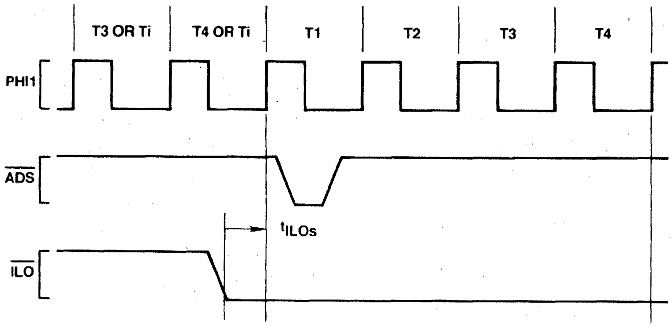
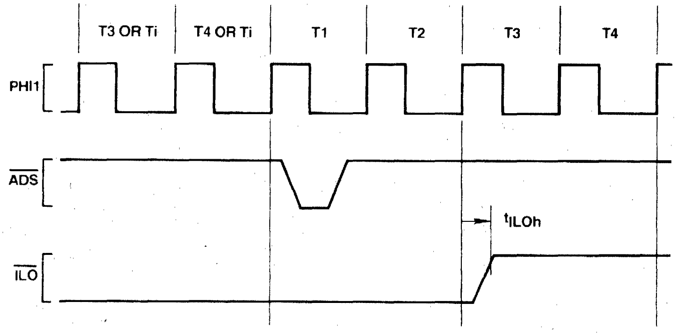


FIGURE 4-18b. Guaranteed Delay, Non-Sequential Fetch to PFS.



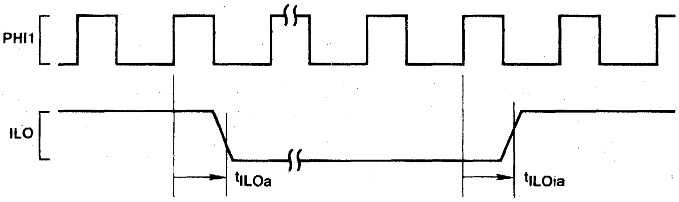
TL/C/5054-61

FIGURE 4-19a. Relationship of $\overline{\text{ILO}}$ to First Operand Cycle of an Interlocked Instruction.



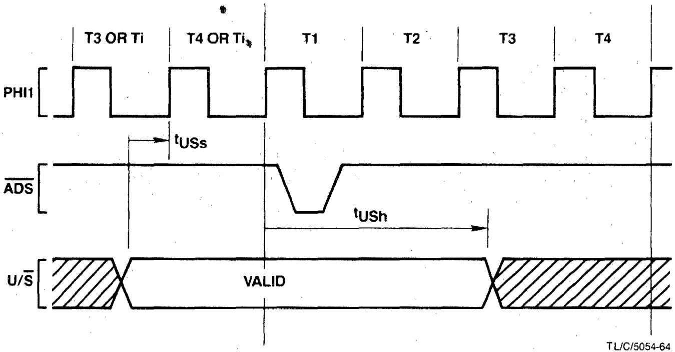
TL/C/5054-62

FIGURE 4-19b. Relationship of $\overline{\text{ILO}}$ to Last Operand Cycle of an Interlocked Instruction.



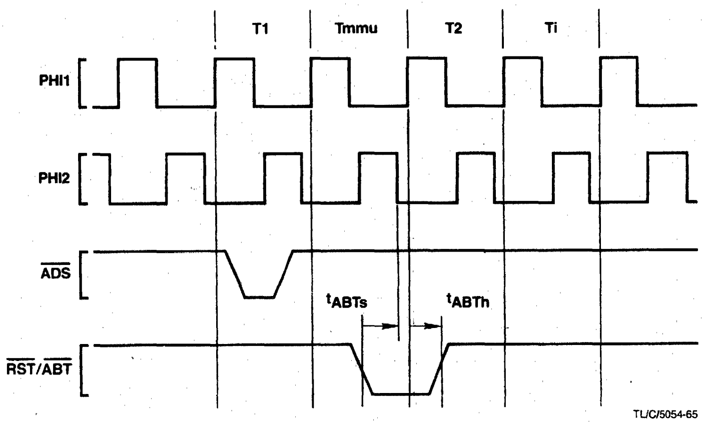
TL/C/5054-63

FIGURE 4-20. Relationship of $\overline{\text{ILO}}$ to Any Clock Cycle.



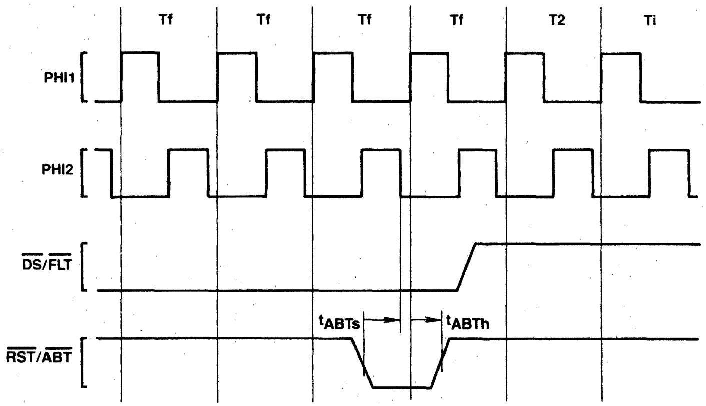
TL/C/5054-64

FIGURE 4-21. U/$\overline{\text{S}}$ Relationship to Any Bus Cycle — Guaranteed Valid Interval.



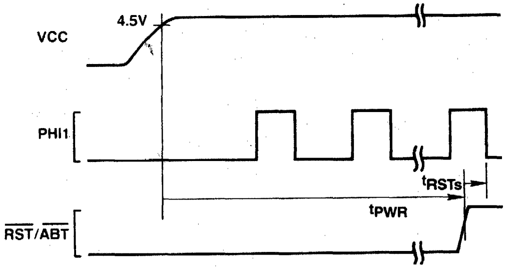
TL/C/5054-65

FIGURE 4-22. Abort Timing, $\overline{FLT}$ Not Applied.


Tf
Tf
Tf
Tf
T2
Ti
PHI1
PHI2
$\overline{DS}/\overline{FLT}$
t_{ABTs}
t_{ABTh}
$\overline{RST}/\overline{ABT}$

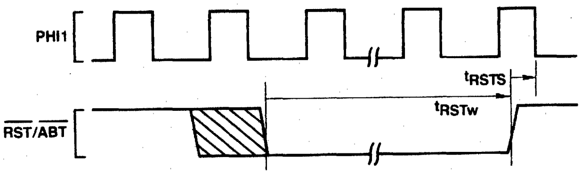
TL/C/5054-66

FIGURE 4-23. Abort Timing, $\overline{FLT}$ Applied.



TL/C/5054-67

FIGURE 4-24. Power-On Reset.



TL/C/5054-68

FIGURE 4-25. Non-Power-On Reset.

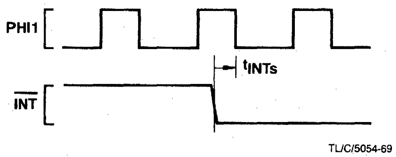


FIGURE 4-26. $\overline{INT}$ Interrupt Signal Detection.

Violation of tINTs timing is allowed, but detection then occurs one clock cycle later.

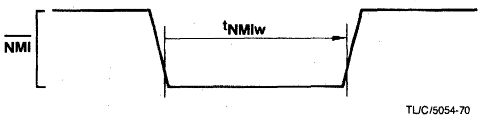


FIGURE 4-27. $\overline{NMI}$ Interrupt Signal Timing.

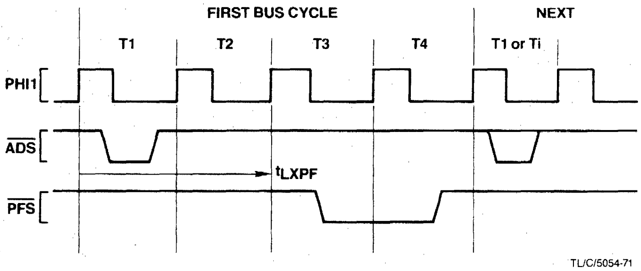


FIGURE 4-28. Relationship Between Last Data Transfer of an Instruction and $\overline{PFS}$ Pulse of Next Instruction.

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.

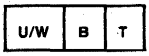
gen, gen 1, gen 2 = General Addressing Mode Field
See Sec. 2.2 for encodings.

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)

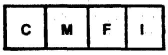
short = Short Immediate value. May contain:
quick: Signed 4-bit value, in MOVQ, ADDQ, CMPQ, ACB.
cond: Condition Code (above), in Scond.
areg: CPU Dedicated Register, in LPR, SPR.
0000 = US
0001 – 0111 = (Reserved)
1000 = FP
1001 = SP
1010 = SB
1011 = (Reserved)
1100 = (Reserved)
1101 = PSR
1110 = INTBASE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

Options: in String Instructions

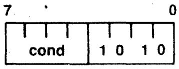


T = Translated
B = Backward
U/W = 00: None
01: While Match
11: Until Match

Configuration bits, in SETCFG:

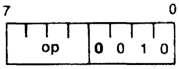


mreg: MMU Register number, in LMR, SMR.
0000 = BPR0
0001 = BPR1
0010 = (Reserved)
0011 = (Reserved)
0100 = PF0
0101 = PF1
0110 = (Reserved)
0111 = (Reserved)
1000 = SC
1001 = (Reserved)
1010 = MSR
1011 = BCNT
1100 = PTB0
1101 = PTB1
1110 = (Reserved)
1111 = EIA



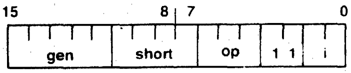
Format 0

Bcond (BR)



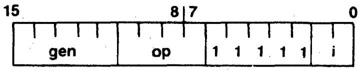
Format 1

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

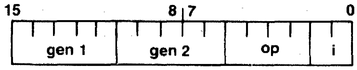
ADDQ	–000	ACB	–100
CMPQ	–001	MOVQ	–101
SPR	–010	LPR	–110
Scond	–011		



Format 3

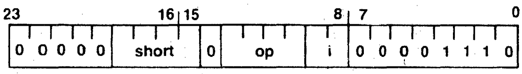
CXPD	−0000	ADJSP	−1010
BICPSR	−0010	JSR	−1100
JUMP	−0100	CASE	−1110
BISPSR	−0110		

Trap (UND) on XXX1, 1000



Format 4

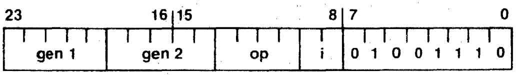
ADD	−0000	SUB	−1000
CMP	−0001	ADDR	−1001
BIC	−0010	AND	−1010
ADDC	−0100	SUBC	−1100
MOV	−0101	TBIT	−1101
OR	−0110	XOR	−1110



Format 5

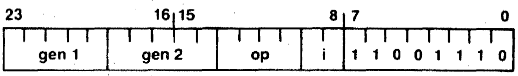
MOVS	−0000	SETCFG	−0010
CMPS	−0001	SKPS	−0011

Trap (UND) on 1XXX, 01XX



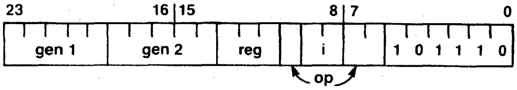
Format 6

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



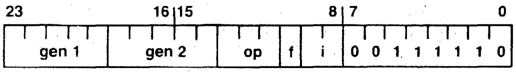
Format 7

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



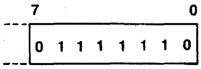
Format 8

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



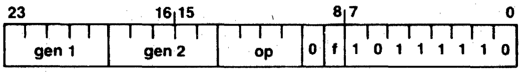
Format 9

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



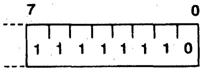
Format 10

Trap (UND) Always



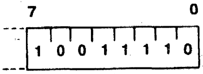
Format 11

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



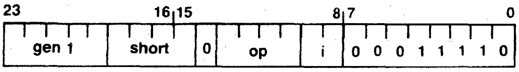
Format 12

Trap (UND) Always



Format 13

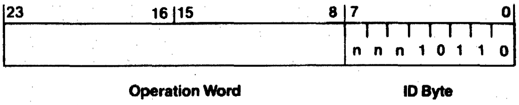
Trap (UND) Always



Format 14

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

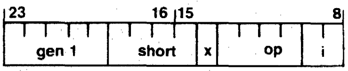
Trap (UND) on 01XX, 1XXX



Format 15
(Custom Slave)

nnn **Operation Word Format**

000

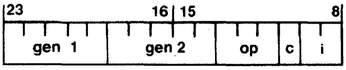


Format 15.0

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

Trap (UND) on all others

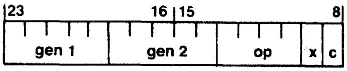
001



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	Trap (UND)	−1010
CCMP	−0010	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

7							0
0	1	0	1	1	1	1	0

Format 16

Trap (UND) Always

7							0
1	1	0	1	1	1	1	0

Format 17

Trap (UND) Always

7							0
1	0	0	0	1	1	1	0

Format 18

Trap (UND) Always

7							0
X	X	X	0	0	1	1	0

Format 19

Trap (UND) Always

Implied Immediate Encodings:

7							0
r7	r6	r5	r4	r3	r2	r1	r0

Register Mask, appended to SAVE, ENTER

7							0
r0	r1	r2	r3	r4	r5	r6	r7

Register Mask, appended to RESTORE, EXIT

7							0
offset			length – 1				

Offset/Length Modifier appended to INSS, EXTS

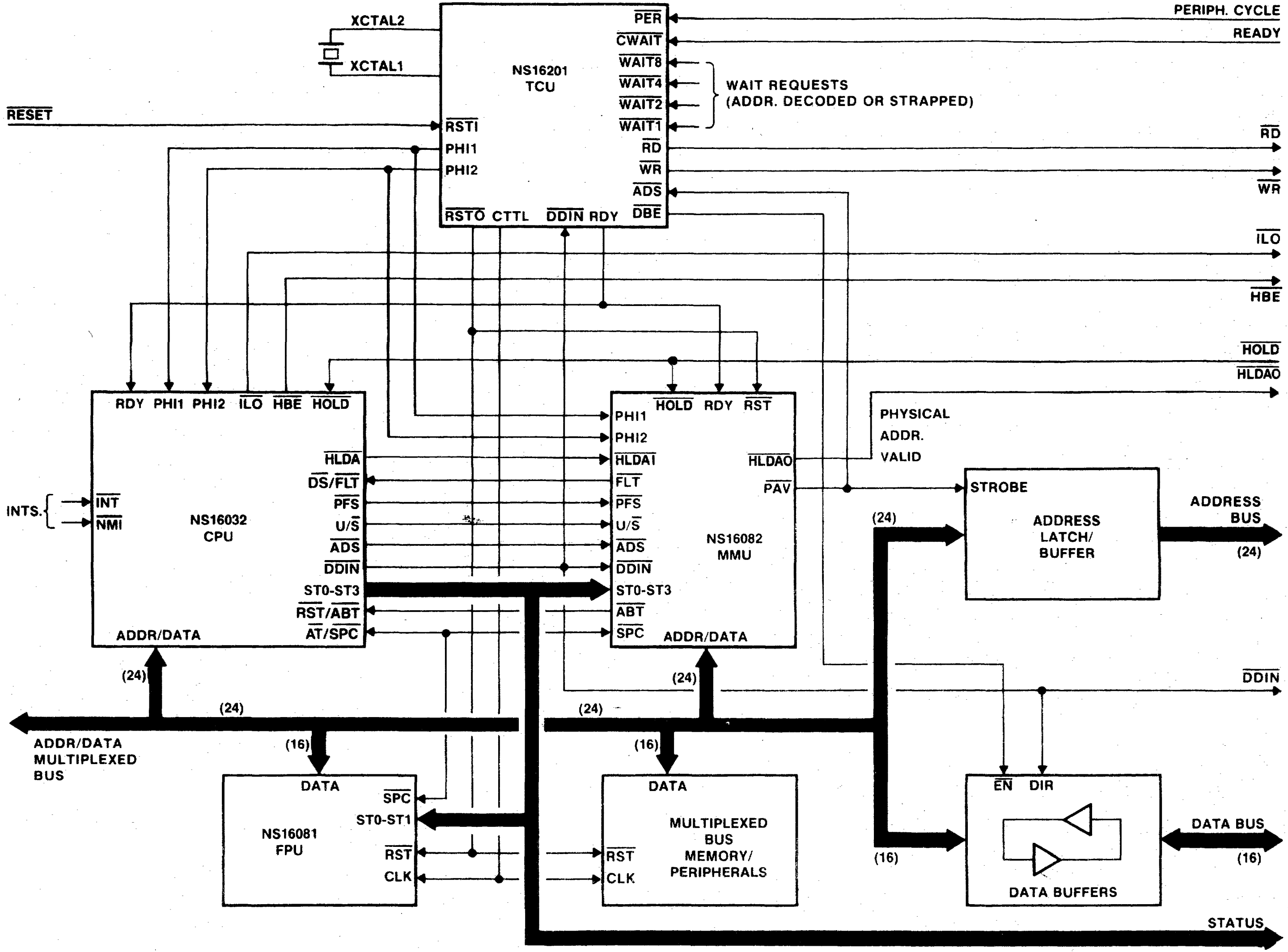


FIGURE B-1. System Connection Diagram.

TL/C/5054-72

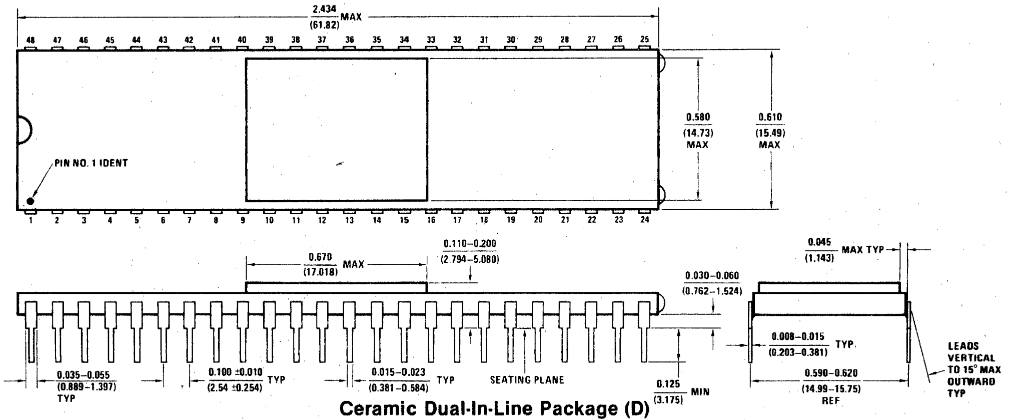
ORDERING INFORMATION

NS16032D-6 NS16032D-4
NS16032N-6 NS16032N-4

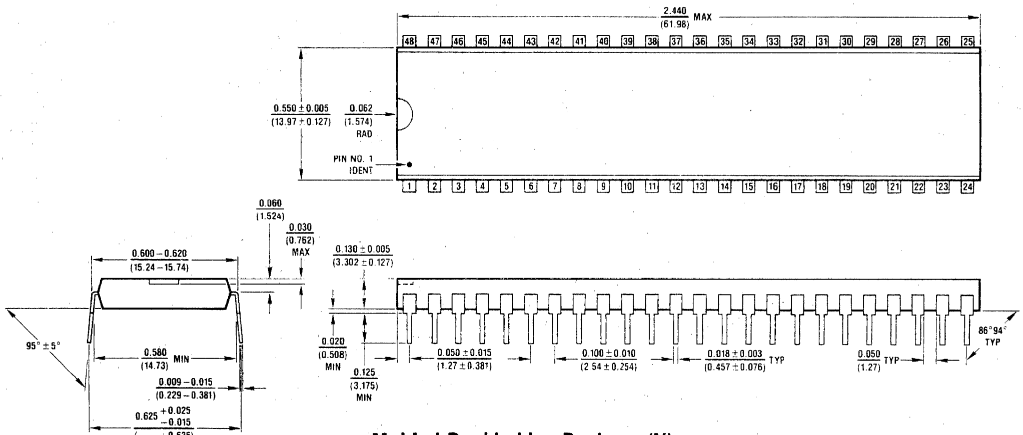
DEVELOPMENT TOOLS ORDERING INFORMATION

DB16000	Evaluation Board
NSX-16	Cross Software Package (VAX/VMS)
SFW-90-A010	Cross Software Package (STARPLEX II™)
NS-ISE-16	In-System Emulator (VAX/VMS)
SPM-90-A1632	In-System Emulator (STARPLEX II)

Physical Dimensions inches (millimeters)



Ceramic Dual-In-Line Package (D)
NS Package D48A



Molded Dual-In-Line Package (N)
NS Package N48A

National Semiconductor

NS16032-10
High-Performance Microprocessor

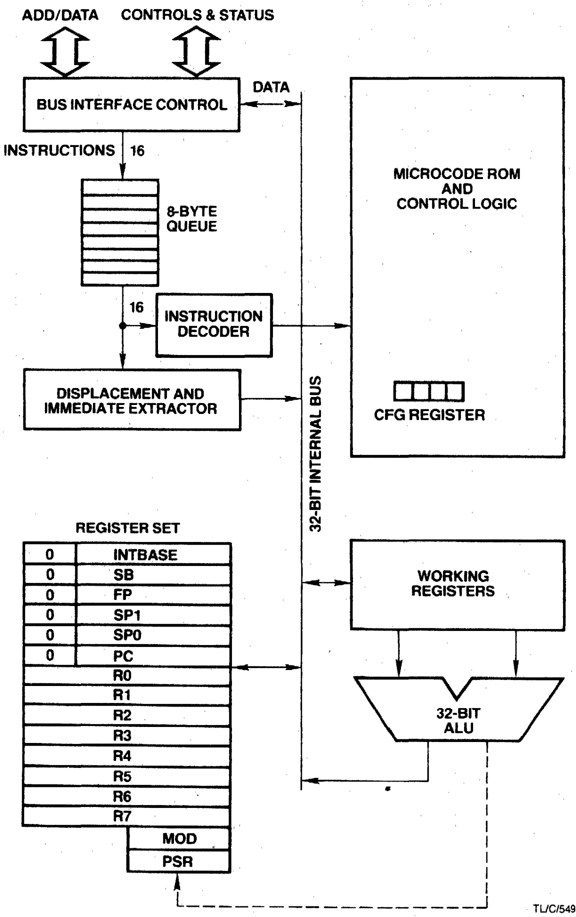
General Description

The NS16032 functions as a central processing unit (CPU) in National Semiconductor's NS16000™ microprocessor family. It has been designed to optimally support microprocessor users who need the ability to use a large addressing space for large programs and/or large data structures. Because large programs must realistically be generated and maintained in high-level languages, the NS16000 architecture provides for very efficient compilation while remaining easy to program at the assembler level for optimizations. NS16000 architecture provides for full virtual memory capability, in conjunction with with the NS16082 Memory Management Unit (MMU). High performance floating-point instructions are provided with the NS16081 Floating-Point Unit (FPU).

Features

- 32-bit Architecture and Implementation
- 16-MByte 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 XMOS™ Technology
- Single 5V Supply
- 48-pin Dual-In-Line Package

NS16032 CPU Block Diagram



TL/C/5490-1

Absolute Maximum Ratings

Temperature under bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All input or output voltages with respect to GND	−0.5V to +7V
Power Dissipation	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 DC Electrical Characteristics.*

DC Electrical Characteristics: T_A = 0 to +70°C, V_{CC} = 5V ±5%, GND = 0V

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{IH}	Logical 1 Input Voltage		2.0		V_{CC}+0.5	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	V_{CC}−0.4		V_{CC}+0.5	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CLT}	Logical 0 Clock Voltage, Transient (ringing tolerance)	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	Logical 1 Output Voltage	$I_{OH} = -400\mu A$	2.4			V
V_{OL}	Logical 0 Output Voltage	$I_{OL} = 2mA$			0.45	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	$0 \leq V_{IN} \leq V_{CC}$, All inputs except PHI1, PHI2, $\overline{AT}/\overline{SPC}$	−20		20	μA
$I_{O(OFF)}$	Output Leakage Current	$0.4 \leq V_{OUT} \leq V_{CC}$	−20		20	μA
I_{CC}	Active Supply Current	$I_{OUT} = 0$, $T_A = 25°C$		180	300	mA

1 NS16032 Pin Descriptions

The following is a brief description of all NS16032 pins. The descriptions reference portions of the Functional Description, Section 3.

1.1 SUPPLIES

Power (V_{CC}): +5V Positive Supply. Sec. 3.1.

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

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

Back-Bias Generator (BBG): Output of on-chip substrate voltage generator. Sec. 3.1.

1.2 INPUT SIGNALS

Clocks (PHI1, PHI2): Two-phase clocking signals. Sec. 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. Sec. 3.4.1.

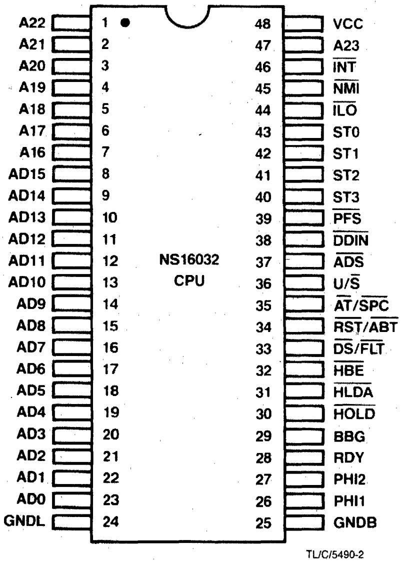
Hold Request ($\overline{\text{HOLD}}$): Active low. Causes the CPU to release the bus for DMA or multiprocessing purposes. Sec. 3.6.

Interrupt ($\overline{\text{INT}}$): Active low. Maskable Interrupt request. Sec. 3.8.

Non-Maskable Interrupt ($\overline{\text{NMI}}$): Active low. Non-Maskable Interrupt request. Sec. 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, Sec. 3.5.4. If held longer, it initiates a Reset, Sec. 3.3.

Connection Diagram



A22 1 48 VCC
A21 2 47 A23
A20 3 46 $\overline{\text{INT}}$
A19 4 45 $\overline{\text{NMI}}$
A18 5 44 $\overline{\text{ILO}}$
A17 6 43 ST0
A16 7 42 ST1
AD15 8 41 ST2
AD14 9 40 ST3
AD13 10 39 $\overline{\text{PFS}}$
AD12 11 38 $\overline{\text{DDIN}}$
AD11 12 37 $\overline{\text{ADS}}$
AD10 13 36 U/$\overline{\text{S}}$
AD9 14 35 $\overline{\text{AT}}/\text{SPC}$
AD8 15 34 $\overline{\text{RST}}/\overline{\text{ABT}}$
AD7 16 33 $\overline{\text{DS}}/\overline{\text{FLT}}$
AD6 17 32 $\overline{\text{HBE}}$
AD5 18 31 $\overline{\text{HLDA}}$
AD4 19 30 $\overline{\text{HOLD}}$
AD3 20 29 BBG
AD2 21 28 RDY
AD1 22 27 PHI2
AD0 23 26 PHI1
GNDL 24 25 GNDB
NS16032
CPU

TL/C/5490-2

1.3 OUTPUT SIGNALS

Address Bits 16-23 (A16-A23): Active high. These are the most significant 8 bits of the memory address bus. Sec. 3.4.

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

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

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

Status (ST0-ST3): Active high. Bus cycle status code, ST0 least significant. Sec. 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. Sec. 3.6.

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

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

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

1.4 INPUT-OUTPUT SIGNALS

Address/Data 0-15 (AD0-AD15): Active high. Multiplexed Address/Data information. Bit 0 is the least significant bit of each. Sec. 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 an instruction. Sec. 3.4.6; Sec. 3.9. Sampled on trailing edge of Reset pulse as Address Translation Strap. Sec. 3.5.1.

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

2 Architectural Description

2.1 PROGRAMMING MODEL

The NS16000 architecture includes 16 registers on the NS16032 CPU.

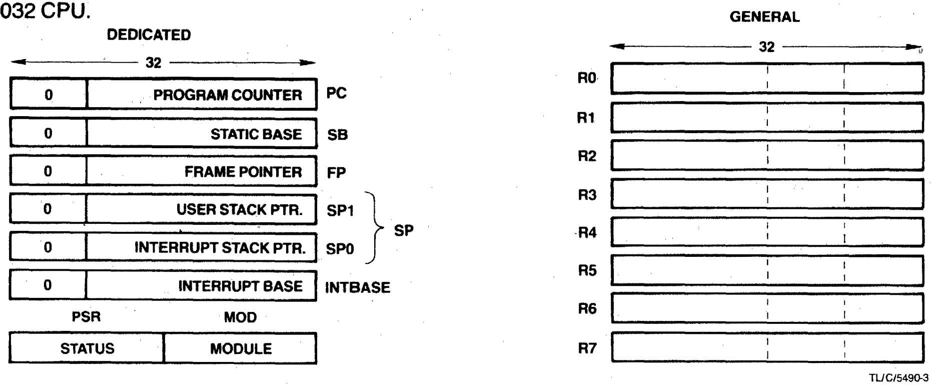


FIGURE 2-1. The General and Dedicated Registers.

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 NS16032 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 NS16032 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 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 NS16032 the upper eight bits of these registers are always zero).

Stacks in the NS16000 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 NS16032 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 NS16032 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 (Sec. 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS16032 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 NS16032 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.

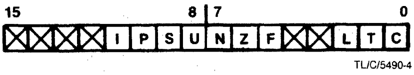


FIGURE 2-2. Processor Status Register.

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

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

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

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

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

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

U: If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When U = 0 the NS16032 is said to be in Supervisor Mode; when U = 1 the NS16032 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.

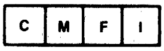
S: The S bit specifies whether the SP0 register or SP1 register is used as the stack pointer. The bit is automatically cleared on interrupts and traps. It may have a setting of 0 (use the SP0 register) or 1 (use the SP1 register).

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

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

2.1.3 The Configuration Register (CFG)

Within the Control section of the NS16032 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.



TL/C/5490-5

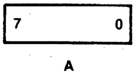
FIGURE 2-3. CFG Register.

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS16202 Interrupt Control Unit). If the CFG I bit is set, interrupts requested through the $\overline{\text{INT}}$ pin are "Vectored." If it is clear, these interrupts are "Non-Vectored." See Sec. 3.8.

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

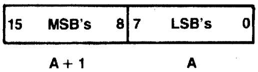
2.1.4 Memory Organization

The main memory of the NS16032 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 nµmbered in increasing significance and toward the left.



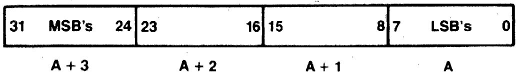
Byte at Address A

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



Word at Address A

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



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 NS16032 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 Sec. 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 NS16032. At any point in time, 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.

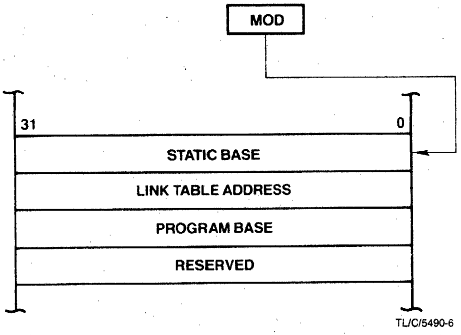


FIGURE 2-4. Module Descriptor Format.

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 NS16000 Programmer's Manual.

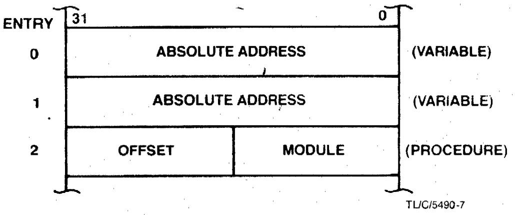


FIGURE 2-5. A Sample Link Table.

2.2 INSTRUCTION SET

2.2.1 General Instruction Format

Figure 2-6 shows the general format of an NS16000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

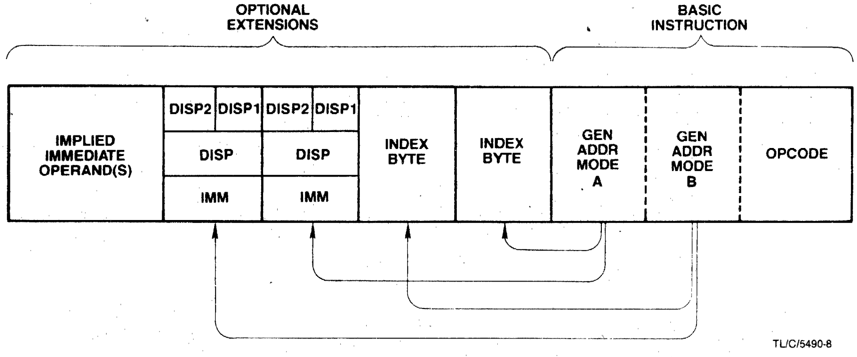


FIGURE 2-6. General Instruction Format.

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.

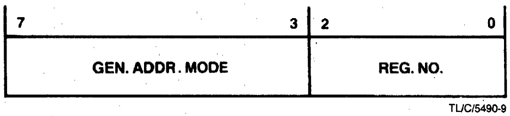


FIGURE 2-7. Index Byte Format.

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain one or 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 backward from the usual memory representation of data (Sec. 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 (Sec. 2.2.3).

2.2.2 Addressing Modes

The NS16032 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."

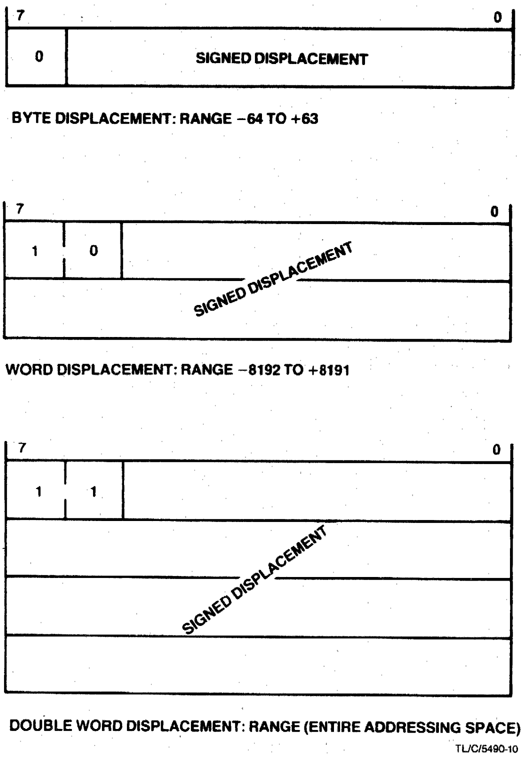
Addressing modes in the NS16032 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

NS16032 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.



BYTE DISPLACEMENT: RANGE −64 TO +63

WORD DISPLACEMENT: RANGE −8192 TO +8191

DOUBLE WORD DISPLACEMENT: RANGE (ENTIRE ADDRESSING SPACE)

FIGURE 2-8. Displacement Encodings.

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

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

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

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

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

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

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Programmer's Manual.

TABLE 2-1.
NS16032 Addressing Modes

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

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS16032 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 Programmer's 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.
NS16032 Instruction Set Summary

MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a 4-bit constant.
7	MOVMi	gen,gen,disp	Move Multiple: disp bytes.
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 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)

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

INTEGER COMPARISON

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

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

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

ARRAYS

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

STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 – Comparison Value
R3 – Translation Table Pointer
R2 – String 2 Pointer
R1 – String 1 Pointer
R0 – Limit Count

Options on all string instructions are:

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

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	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)

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	CCMPc	gen,gen	Custom Compare.
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 Functional Description

3.1 POWER AND GROUNDING

The NS16032 requires a single 5-volt power supply, applied on pin 48 (V_{CC}). See DC Specification Section.

Grounding connections are made on two pins. Logic Ground (GNDL, pin 24) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 25) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below (Figure 3-1).

In addition to V_{CC} and Ground, the NS16032 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (Fig. 3-1) from the BBG pin to ground. Recommended values for these are:

C_1: 1 μF, Tantalum.

C_2: 1000 pF, low inductance. This should be either a disc or monolithic ceramic capacitor.

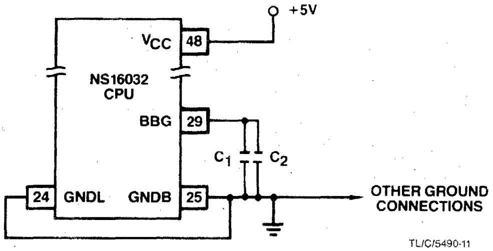


FIGURE 3-1. Recommended Supply Connections.

3.2 CLOCKING

The NS16032 inputs clocking signals from the NS16201 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 positive 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 the AC Specifications (Sec. 4) for complete specifications of PHI1 and PHI2.

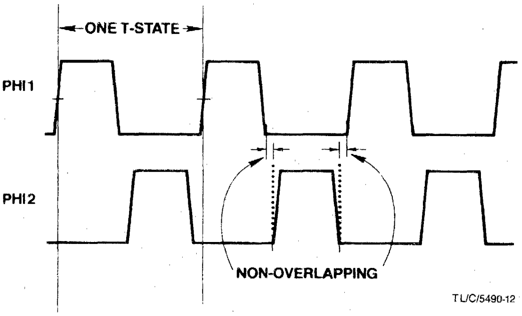


FIGURE 3-2. Clock Timing Relationships.

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

3.3 RESETTING

The $\overline{RST}/\overline{ABT}$ pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Sec. 3.5.4.

The CPU may be reset at any time by pulling the $\overline{RST}/\overline{ABT}$ 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, $\overline{RST}/\overline{ABT}$ must be held low for at least 50 μsec 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

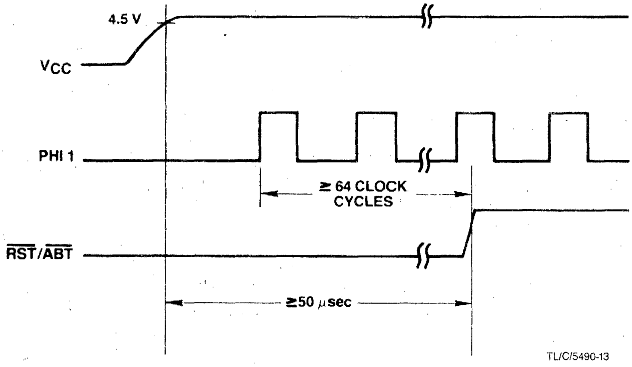


FIGURE 3-3. Power-on Reset Requirements.

active for not less than 64 clock cycles. The trailing (positive-going) edge must occur while PHI1 is high, and no later than 10 ns before the PHI1 trailing edge. See Figures 3-3 and 3-4.

The NS16201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS16032 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.

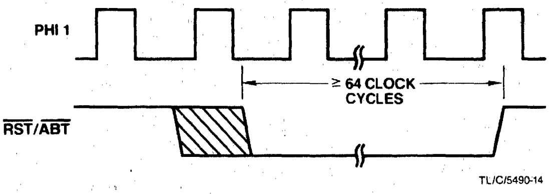


FIGURE 3-4. General Reset Timing.

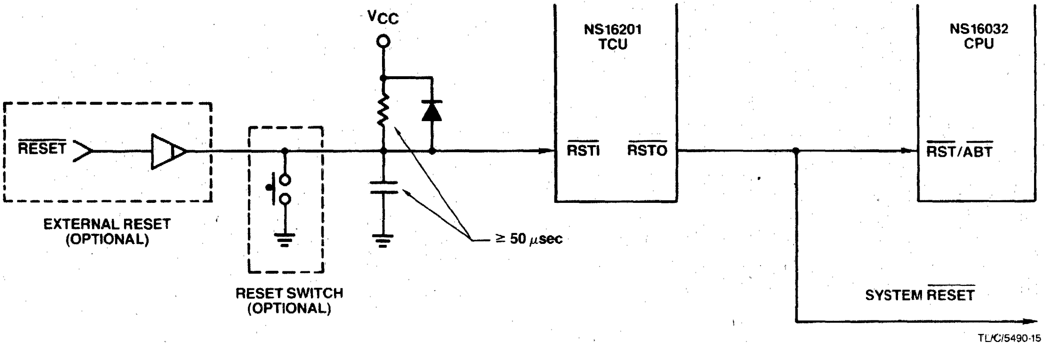


FIGURE 3-5a. Recommended Reset Connections, Non-Memory-Managed System.

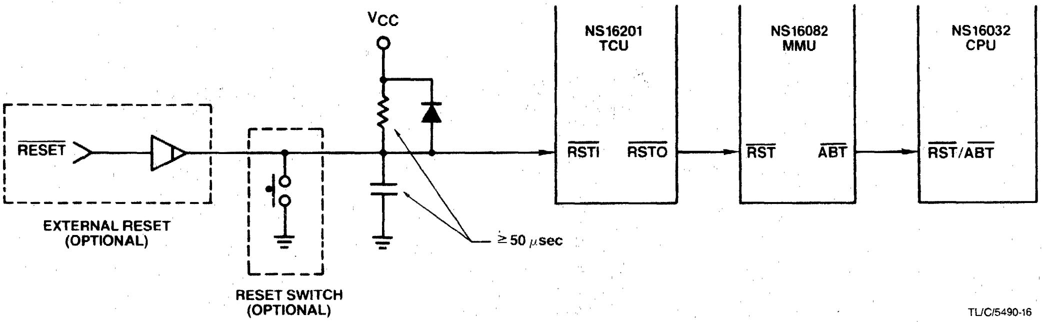


FIGURE 3-5b. Recommended Reset Connections, Memory-Managed System.

3.4 BUS CYCLES

The NS16032 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 Sec. 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 NS16000 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 Sec. 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 (Sec. 3.4.6).

The sequence of events in a non-Slave bus cycle is shown below 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 (Sec. 3.4.1).

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 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 DDIN, indicating the direction of the transfer, and 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 (DS), signalling the beginning of the data transfer. Associated signals from the NS16201 Timing Control Unit are also activated at this time: RD (Read Strobe) or WR (Write Strobe), TSO (Timing State Output, indicating that T2 has been reached) and DBE (Data Buffer Enable).

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

If the CPU is performing a Read cycle, the Data Bus (AD0-AD15) is sampled at the falling edge of PHI2 of the last T3 state. See Timing Specification, Sec. 4. Data must, however, be held at least until the beginning of T4. DS and 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 DS, RD or WR, and TSO signals go inactive, and at the rising edge of PHI2, DBE goes inactive, having provided for necessary data hold times. Addresses (and Data during Write cycles) remain 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).

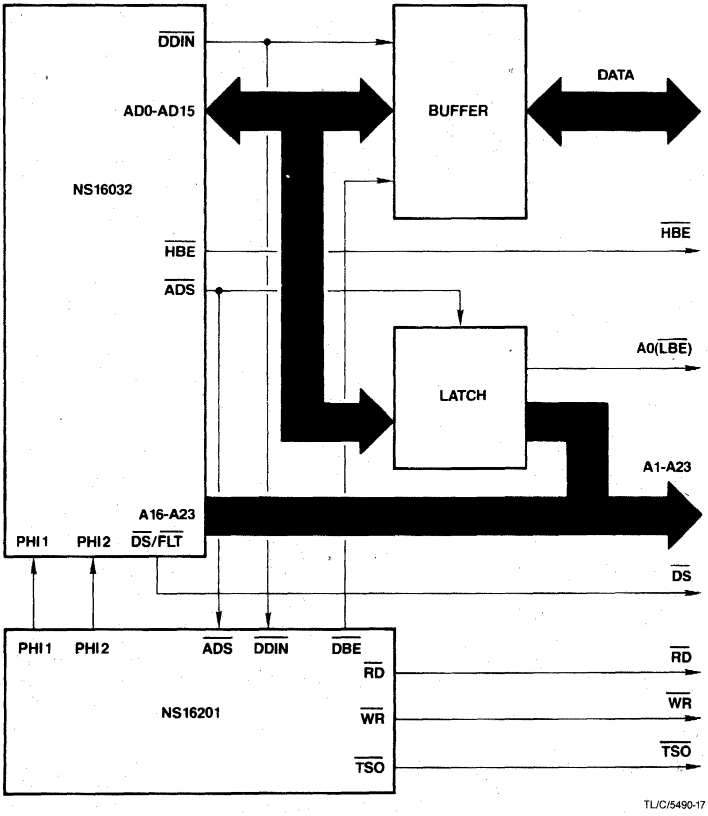


FIGURE 3-6. Bus Connections.

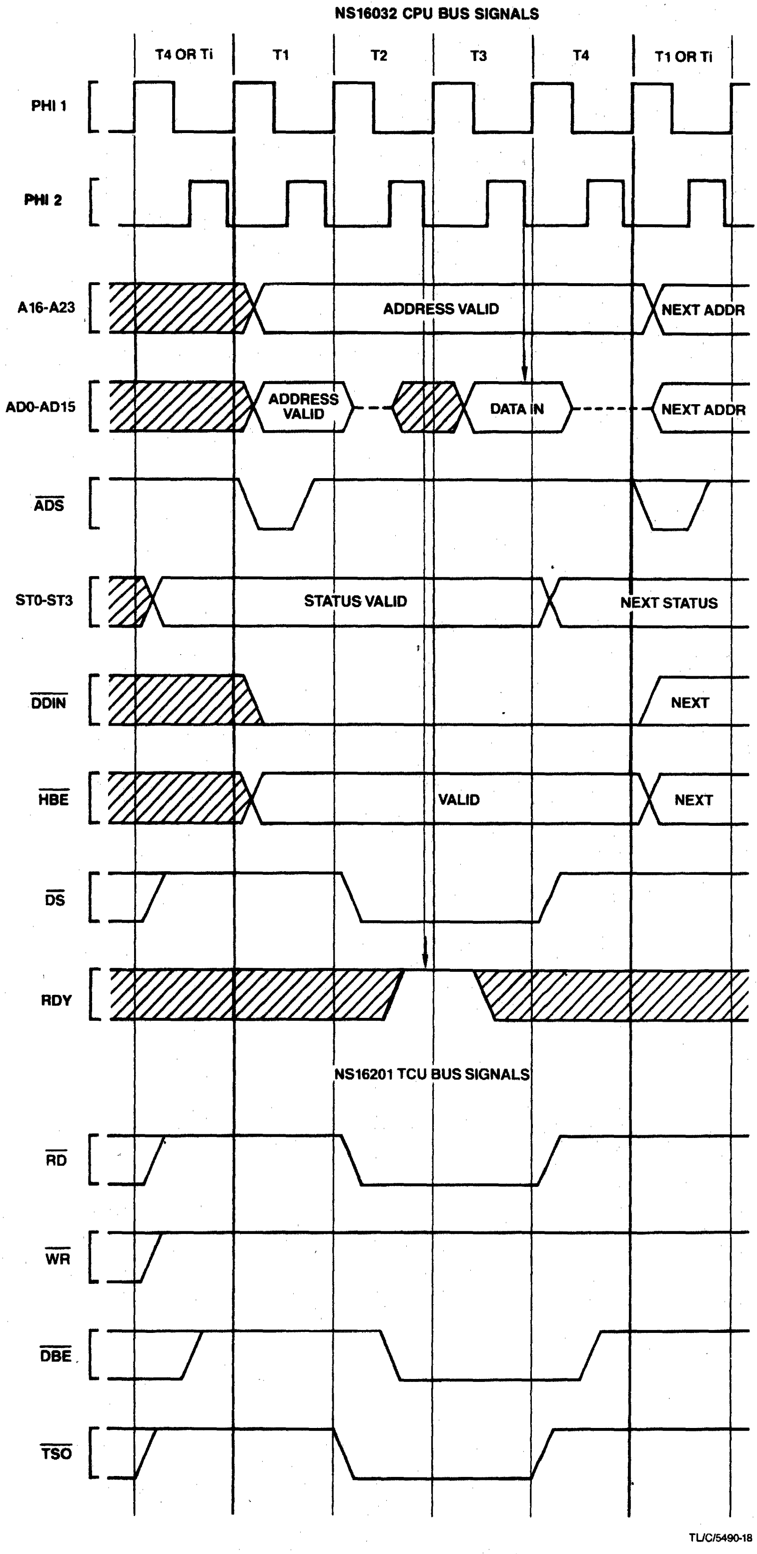
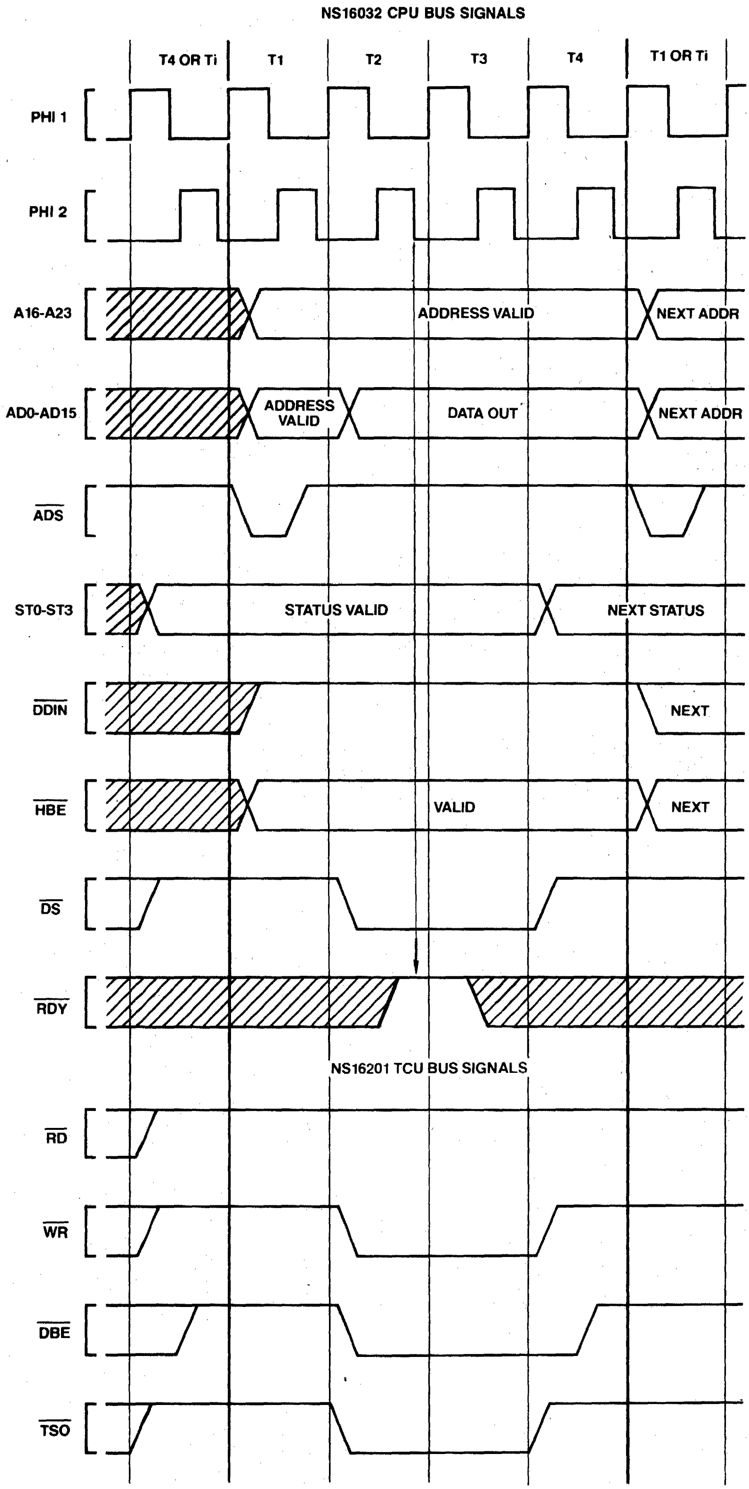


FIGURE 3-7. Read Cycle Timing.



TL/C/5490-19

FIGURE 3-8. Write Cycle Timing.

3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS16032 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 PHI 2, 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 PHI 2. 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 NS16201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pins:

1) $\overline{\text{CWAIT}}$ (Continuous WAIT), which holds the CPU in WAIT states until removed.

2) $\overline{\text{WAIT1}}$, $\overline{\text{WAIT2}}$, $\overline{\text{WAIT4}}$, $\overline{\text{WAIT8}}$ (Collectively $\overline{\text{WAITn}}$), 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 on their use, see the NS16201 Data Sheet.

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

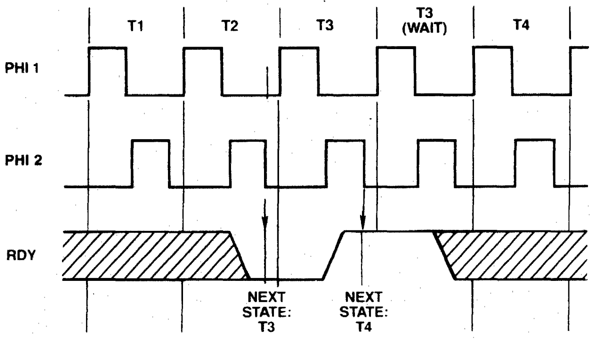


FIGURE 3-9. RDY Pin Timing.

3.4.2 Bus Status

The NS16032 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 yet need access to the bus.

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.

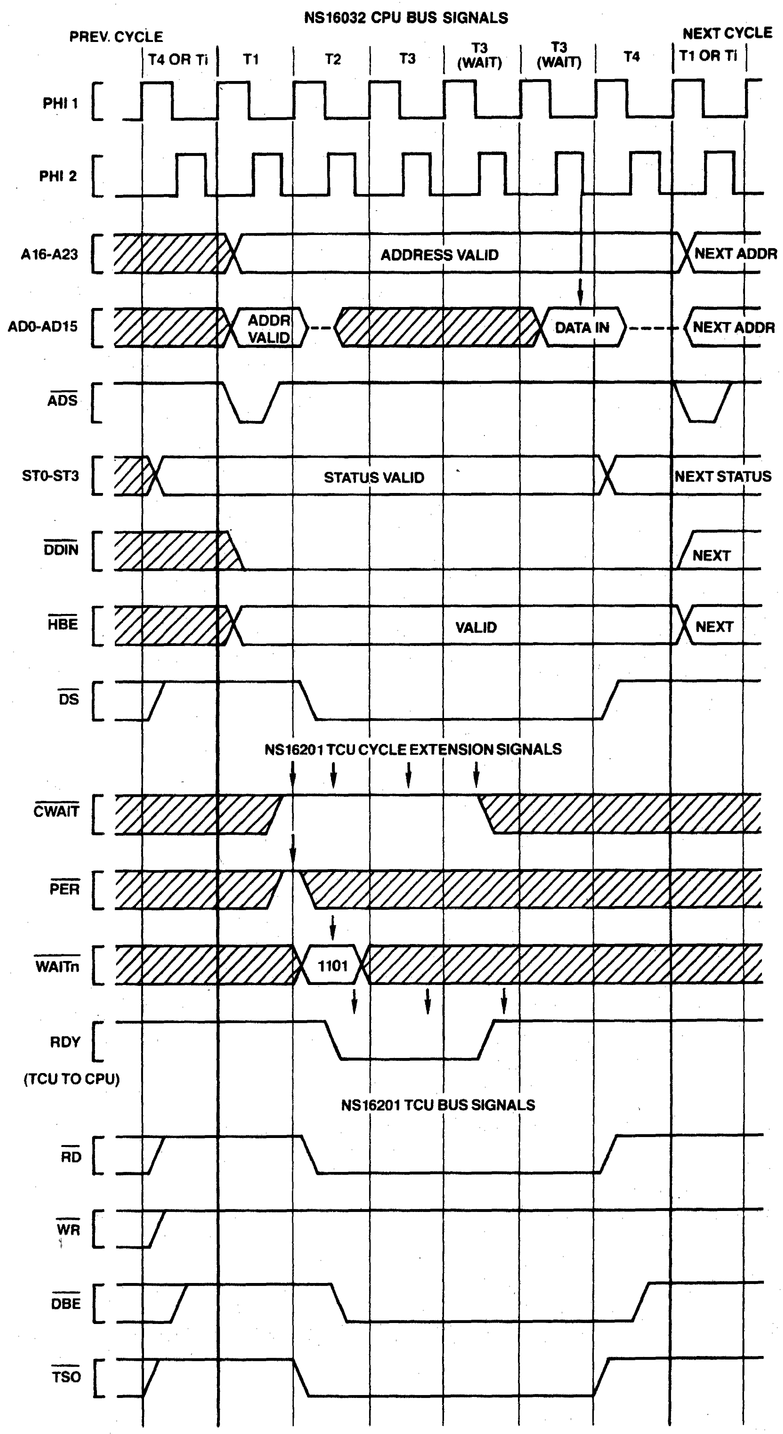
0100 – Interrupt Acknowledge, Master.
The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{\text{NMI}}$) it will read from address FFFF00_{16}, but will ignore any data provided.
To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$) it will read from address FFFE00_{16}, expecting a vector number to be provided from the Master NS16202 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 NS16202 is present. See Sec. 3.4.5.

0101 – Interrupt Acknowledge, Cascaded.
The CPU is reading a vector number from a Cascaded NS16202 Interrupt Control Unit. The address provided is the address of the NS16202 Hardware Vector register. See Sec. 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 Sec. 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 Sec. 3.4.5.

1000 – Sequential Instruction Fetch.
The CPU is reading the next sequential word from the instruction stream into the Instruction



TL/C/5490-21

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.

Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

1001 – Non-Sequential Instruction Fetch.
The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

1010 – Data Transfer.
The CPU is reading or writing an operand of an instruction.

1011 – Read RMW Operand.
The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.

1100 – Read for Effective Address Calculation.
The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

1101 – Transfer Slave Processor Operand.
The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Sec. 3.9.1.

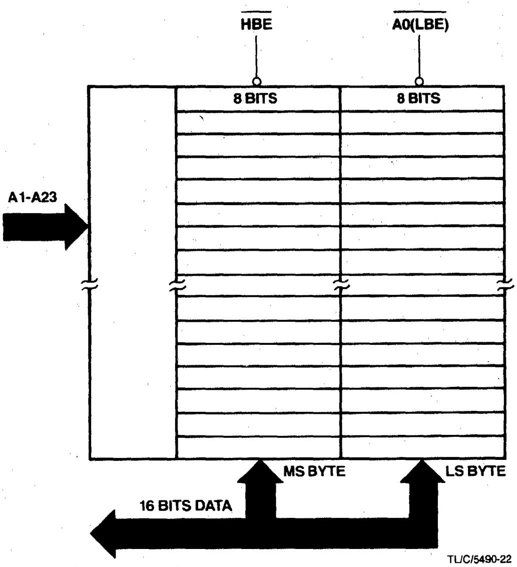
1110 – Read Slave Processor Status.
The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU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

1111 – Broadcast Slave ID.
The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Sec. 3.9.1.

3.4.3 Data Access Sequences

The 24-bit address provided by the NS16032 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 NS16032 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 NS16032 provides a special control signal, High Byte Enable ($\overline{\text{HBE}}$), which facilitates individual byte addressing on a 16-bit bus.

Memory is intended to be 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 $\overline{\text{HBE}}$ is low. See Figure 3-11.



TL/C/5490-22

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 states of A0 and $\overline{\text{HBE}}$ for each category.

**Table 3-1.
Bus Cycle Categories**

Category	$\overline{\text{HBE}}$	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 $\overline{\text{HBE}}$ see Sec. 3.4.

Table 3.2
Access Sequences

A. Odd Word Access Sequence

BYTE 1	BYTE 0	←A

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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.

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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.

Cycle	Type	Address	$\overline{\text{HBE}}$	A0	High Bus	Low Bus
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 operands which 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 NS16032 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 (Sec. 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 NS16032 interrupt structure, see Sec. 3.8.

Table 3-3
Interrupt Sequences

Cycle	Status	Address	$\overline{\text{DDIN}}$	$\overline{\text{HBE}}$	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 $\overline{\text{HBE}}$ 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 $\overline{\text{HBE}}$ low and reads the vector number from bits 8-15 of the Data Bus. The vector number may be in the range 0-255.

3.4.6 Slave Processor Communication

In addition to its use as the Address Translation strap (Sec. 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 least significant two bits of CPU cycle status (ST0-ST1) are monitored by each Slave Processor in order to determine the type of transfer being performed. $\overline{SPC}$ is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Sec. 3.9 for full protocol sequences.

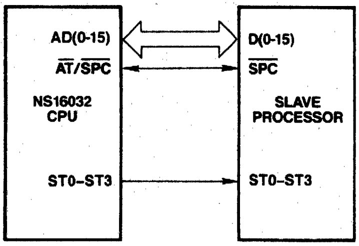
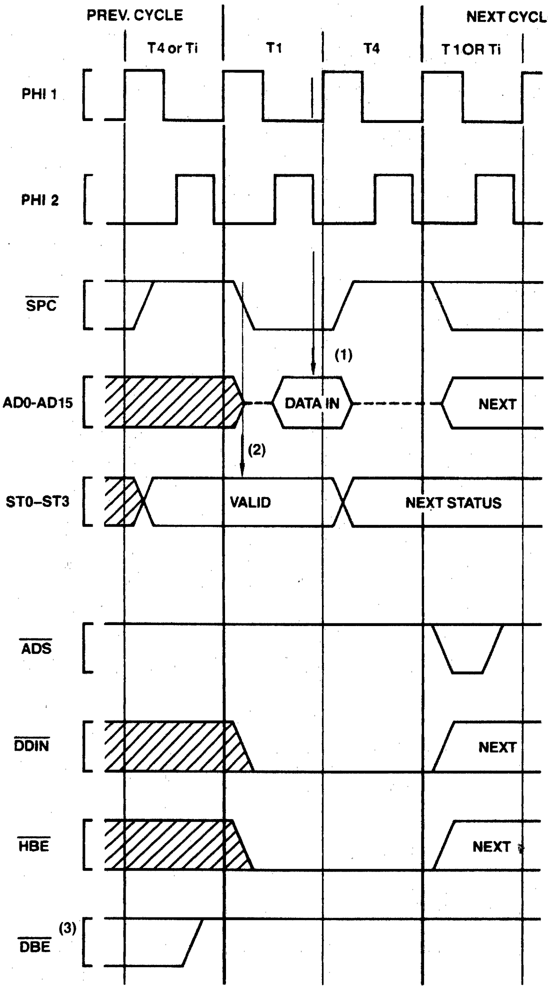


FIGURE 3-12. Slave Processor Connections.



TL/C/5490-24

NOTE:

(1) CPU samples Data Bus here.

(2) Slave Processor samples CPU Status here.

(3) $\overline{DBE}$ and all other NS16201 TCU bus signals remain inactive because no $\overline{ADS}$ pulse is received from the CPU.

FIGURE 3-13. CPU Read from Slave Processor.

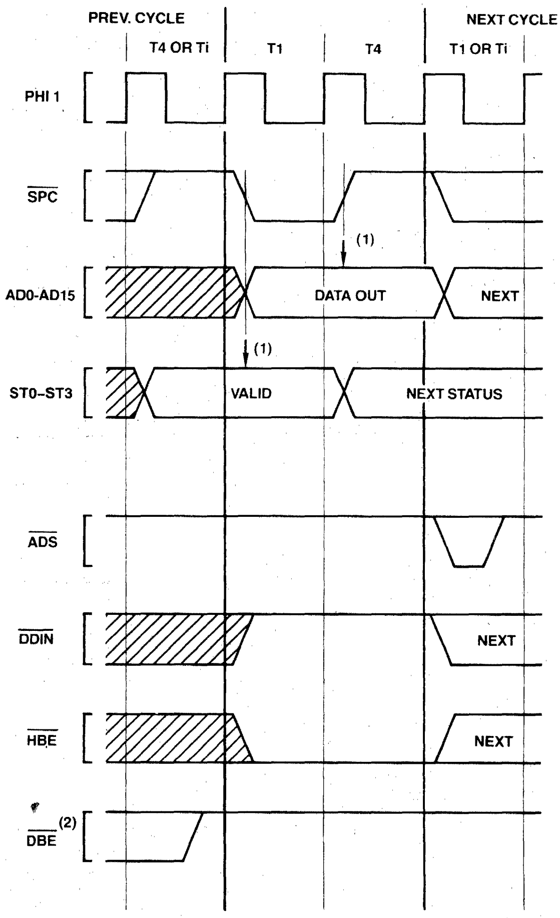
3.4.6.1 Slave Processor Bus Cycles

A Slave Processor bus cycle always takes exactly two clock cycles, labelled T1 and T4 (see Figures 3-13 and 3-14). During a Read cycle, $\overline{SPC}$ is activated at T1, data is sampled at T4, and $\overline{SPC}$ is removed. 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 NS16201 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 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.



NOTE:

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

(2) $\overline{DBE}$, being provided by the NS16201 TCU, remains inactive due to the fact that no pulse is presented on $\overline{ADS}$. TCU signals $\overline{RD}$, $\overline{WR}$ and $\overline{TSO}$ also remain inactive.

FIGURE 3-14. CPU Write to Slave Processor.

3.5 MEMORY MANAGEMENT OPTION

The NS16032 CPU, in conjunction with the NS16082 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 NS16032 CPU has two bus timing modes: With or Without Address Translation. The mode of operation is selected by the CPU by sampling the $\overline{AT}$/SPC (Address Translation/ Slave Processor Control) pin on the rising edge of the $\overline{RST}$ (Reset) pulse. If $\overline{AT}$/SPC is sampled as high, the bus timing is as previously described in Sec. 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 NS16082 MMU will itself pull the CPU $\overline{AT}$/SPC pin low when it is reset, but this pin may be left floating in non-Memory-Managed systems.

Note that the Address Translation strap does not specifically declare the presence of an NS16082 MMU, but

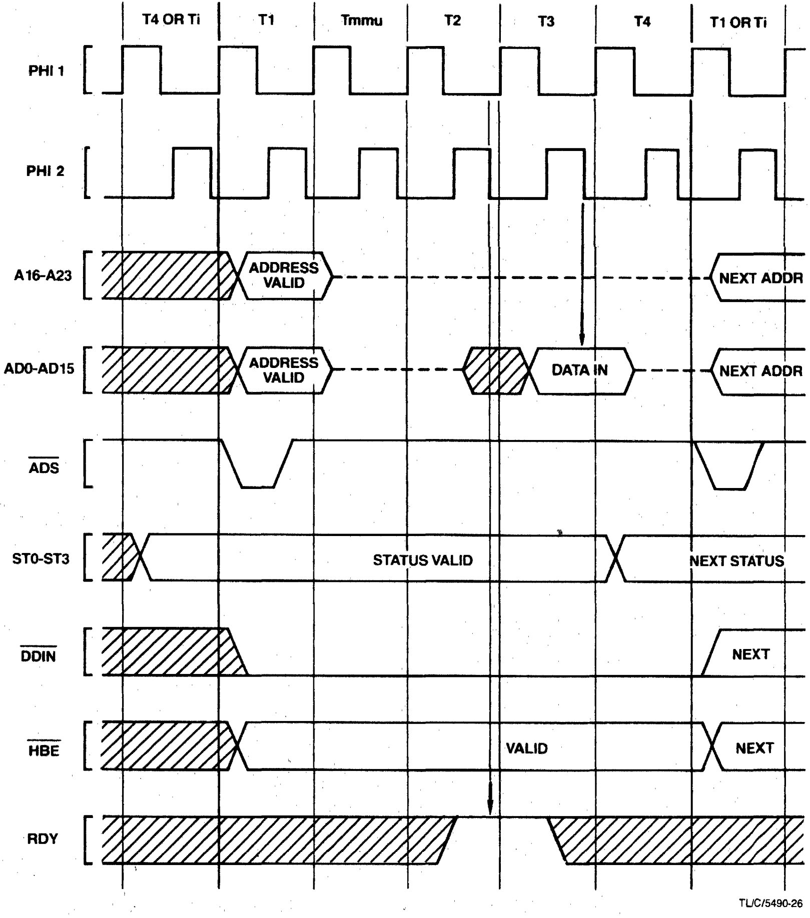


FIGURE 3-15. Read Cycle with Address Translation (CPU Action).

only the 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 Sec. 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 $\overline{\text{PAV}}$. T2 through T4 of the cycle are identical to their counterparts 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.

Figures 3-17 and 3-18 show a Read cycle and a Write cycle as generated by the 16032/16082/16201 group. Note that with the CPU $\overline{\text{ADS}}$ signal going only to the MMU, and with the MMU $\overline{\text{PAV}}$ signal substituting for $\overline{\text{ADS}}$ everywhere else, Tmmu through T4 look exactly like T1 through T4 in a non-Memory-Managed system. For the connection diagram, see Appendix B.

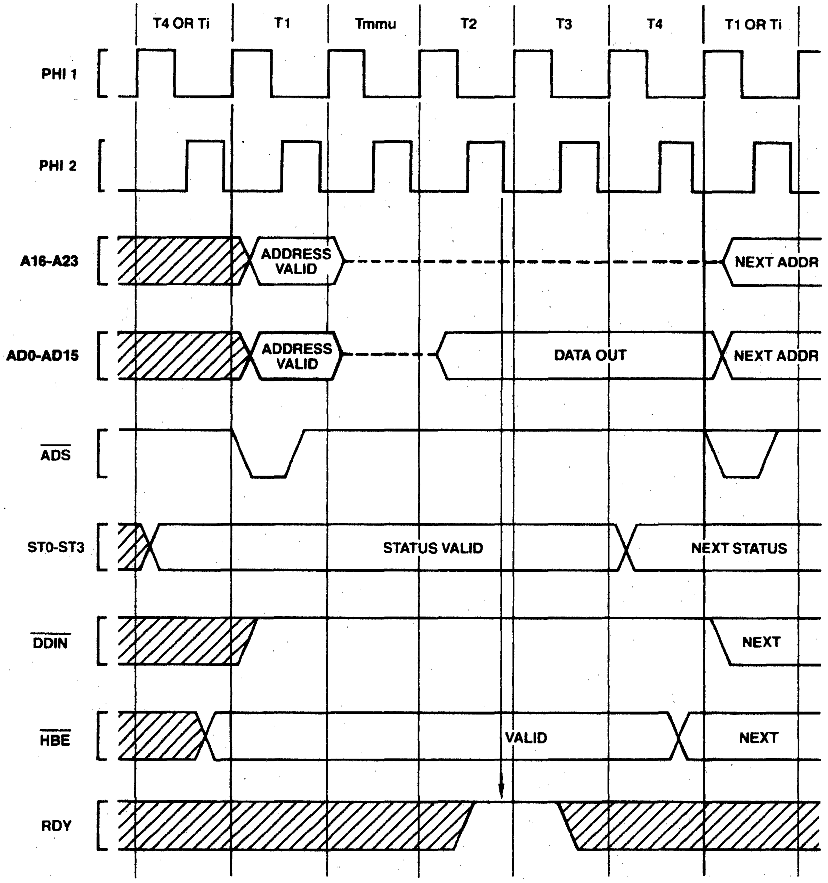
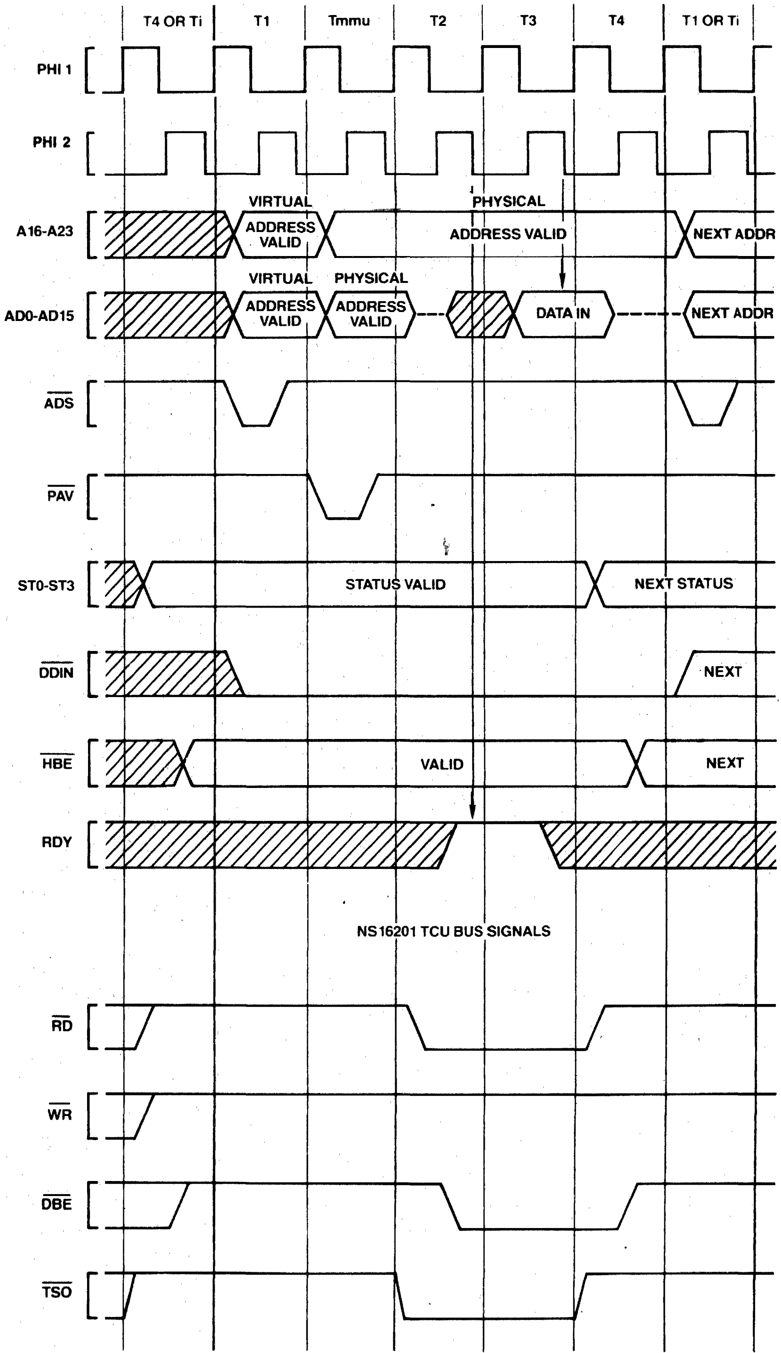
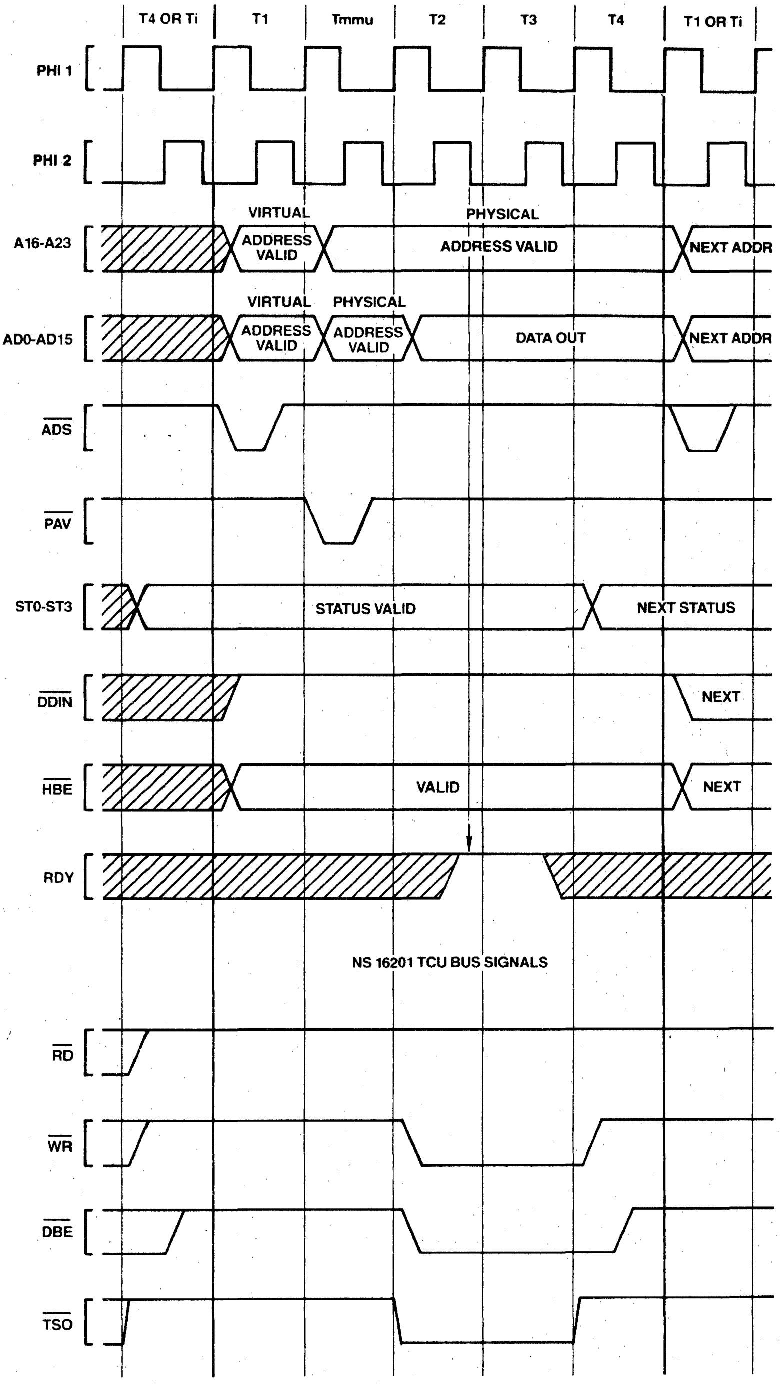


FIGURE 3-16. Write Cycle with Address Translation (CPU Action).



TL/C/5490-28

FIGURE 3-17. Memory-Managed Read Cycle.



TL/C/5490-29

FIGURE 3-18. Memory-Managed Write Cycle.

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

In Address Translation mode, the $\overline{\text{DS}}/\overline{\text{FLT}}$ pin is treated as the input command $\overline{\text{FLT}}$ (Float). Activating $\overline{\text{FLT}}$ during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the NS16082 MMU in order to update its internal translation cache from page tables in memory, or to update certain status bits within them.

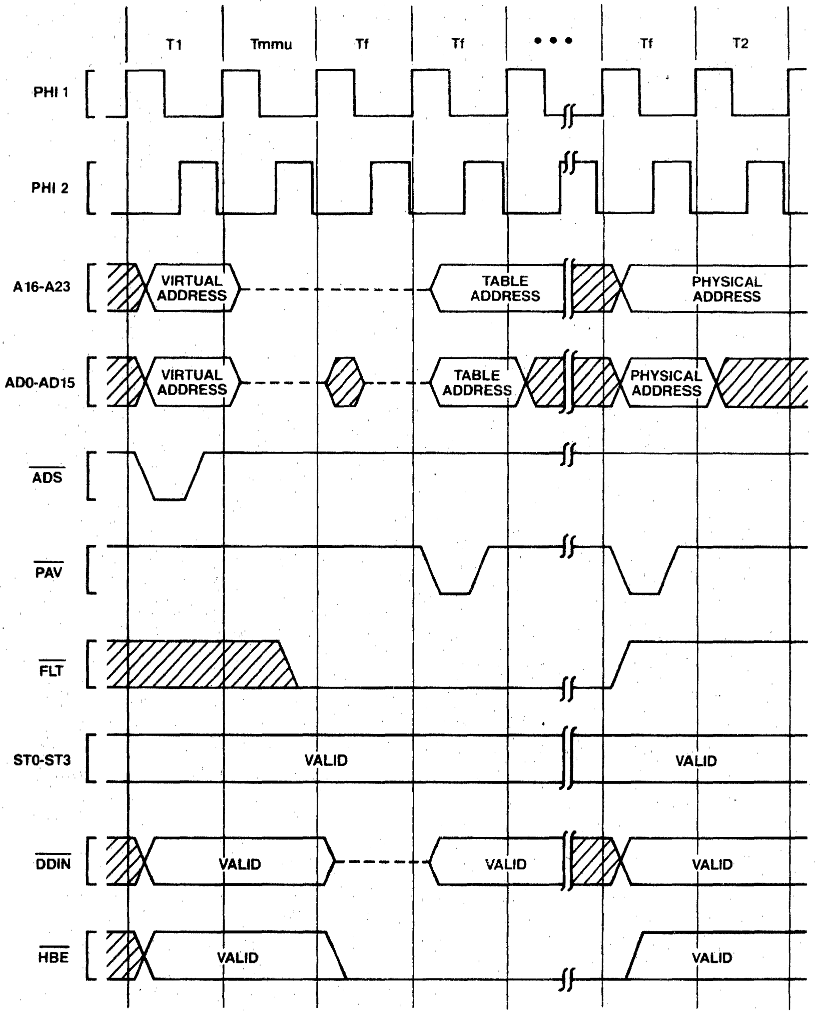
Figure 3-19 shows the effects of $\overline{\text{FLT}}$. Upon sampling $\overline{\text{FLT}}$ low, late in Tmmu, the CPU enters idle T-States (Tf) during which it:

1) Sets AD0–AD15, A16–A23 and $\overline{\text{DDIN}}$ to the TRI-STATE condition ("floating").

2) Sets $\overline{\text{HBE}}$ low.

3) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See $\overline{\text{RST}}/\overline{\text{ABT}}$ description, Sec. 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 $\overline{\text{FLT}}$ again goes high. See the Timing Specifications, Sec. 4.



TL/C/5490-30

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

3.5.4 Aborting Bus Cycles

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

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

The NS16082 MMU will abort a bus cycle for either of two reasons:

1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The referenced page must be brought into physical memory from mass storage to make it accessible to the CPU.

2) The CPU is attempting to perform an access which is not allowed due to the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction which caused it to occur is also aborted in such a manner that it is guaranteed re-executable later. Due to the NS16000 Family instruction set definition and its implementation in the NS16032 CPU, the only information which is changed irrecoverably by such partly-executed instructions is information which does not affect their 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 Sec. 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, such 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 which 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 NS16082 Memory Management Unit.

1) If $\overline{\text{FLT}}$ has not been applied to the CPU, the Abort pulse must occur during or before Tmmu. See the Timing Specifications, Figure 4-22.

2) If $\overline{\text{FLT}}$ has been applied to the CPU, the Abort pulse must be applied before the T-State in which $\overline{\text{FLT}}$ goes inactive. The CPU will not actually respond to the Abort command until $\overline{\text{FLT}}$ is removed. See Figure 4-23.

3) No bus cycle may be aborted which is the Write half of a Read-Modify-Write operand access. The CPU guarantees that this will never be necessary for Memory Management functions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

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

3.6 BUS ACCESS CONTROL

The NS16032 CPU has the capability of relinquishing its access to the bus upon request from a DMA device or another CPU. This capability is implemented on the $\overline{\text{HOLD}}$ (Hold Request) and $\overline{\text{HLDA}}$ (Hold Acknowledge) pins. By asserting $\overline{\text{HOLD}}$ low, an external device requests access to the bus. On receipt of $\overline{\text{HLDA}}$ from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0-AD15, A16-A23, $\overline{\text{ADS}}$, $\overline{\text{DDIN}}$ and $\overline{\text{HBE}}$ pins to the TRI-STATE® condition. To return control of the bus to the CPU, the device sets $\overline{\text{HOLD}}$ inactive, and the CPU acknowledges return of the bus by setting $\overline{\text{HLDA}}$ inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the $\overline{\text{HOLD}}$ request is made, as the CPU must always complete the current bus cycle. Figure 3-20 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 $\overline{\text{HOLD}}$ request is made. If the request is made during or before the clock cycle shown (two clock cycles before T4), the CPU will release the bus during the clock cycle following T4. If the request occurs closer to T4, the CPU may already have decided to initiate another bus cycle. In that case it will not grant the bus until after the next T4 state. Note that this situation will also occur if the CPU is idle on the bus but has initiated a bus cycle internally.

In a Memory-Managed system, the $\overline{\text{HLDA}}$ signal is connected in a daisy-chain through the NS16082, such that the MMU can release the bus if it is using it.

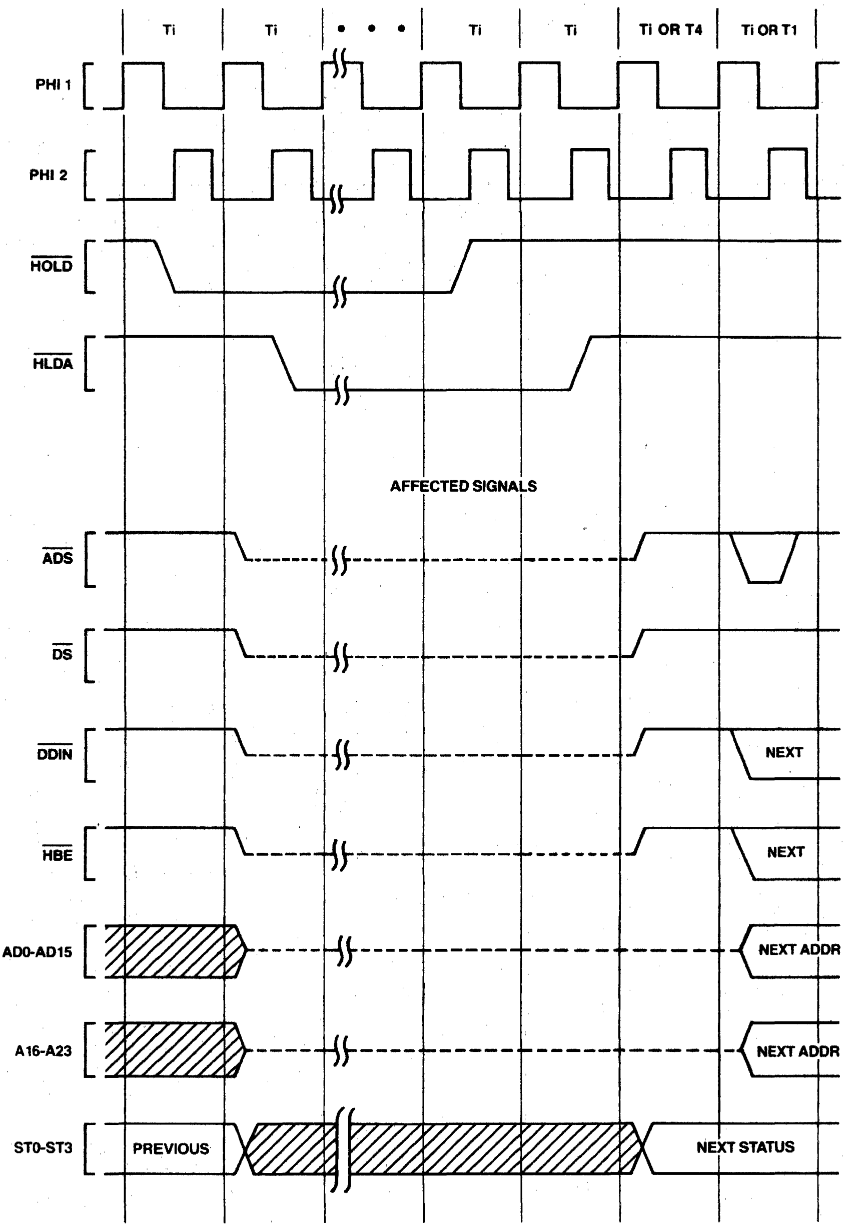


FIGURE 3-20. $\overline{\text{HOLD}}$ Timing, Bus Initially Idle.

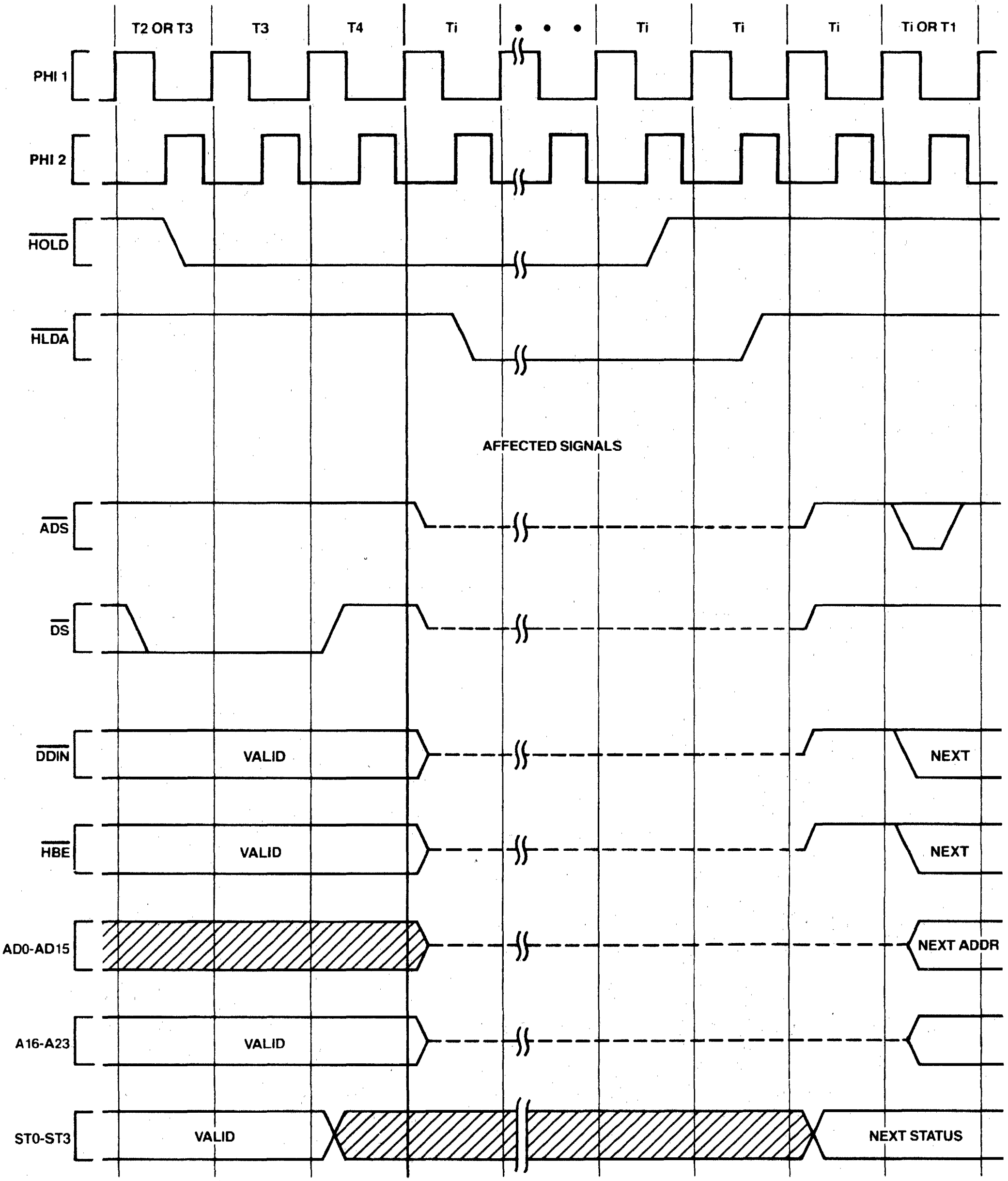


FIGURE 3-21. $\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle.

3.7 INSTRUCTION STATUS

In addition to the four bits of Bus Cycle status (ST0-ST3), the NS16032 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.

$\overline{\text{PFS}}$ (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes, and is used that way by the NS16082 Memory Management Unit.

U/$\overline{\text{S}}$ originates from the U bit of the Processor Status Register, and indicates whether the CPU is currently running in User or Supervisor mode. It is sampled by the MMU for mapping, protection and debugging purposes. Although it is not synchronous to bus cycles, there are guarantees on its validity during any given bus cycle. See the Timing Specifications, Figure 4-21.

$\overline{\text{ILO}}$ (Interlocked Operation) is activated during an SBITI (Set Bit, Interlocked) or CBITI (Clear Bit, Interlocked) instruction. It is made available to external bus arbitration circuitry in order to allow these instructions to implement the semaphore primitive operations for multiprocessor communication and resource sharing. As with the U/$\overline{\text{S}}$ pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification Section, Figure 4-19.

3.8 NS16032 INTERRUPT STRUCTURE

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

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

$\overline{\text{RST/ABT}}$, which may be used to abort a bus cycle and any associated instruction. It generates an interrupt request if an instruction was aborted. See Sec. 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 four 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) Saving Processor Status.
The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.

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

4) 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.

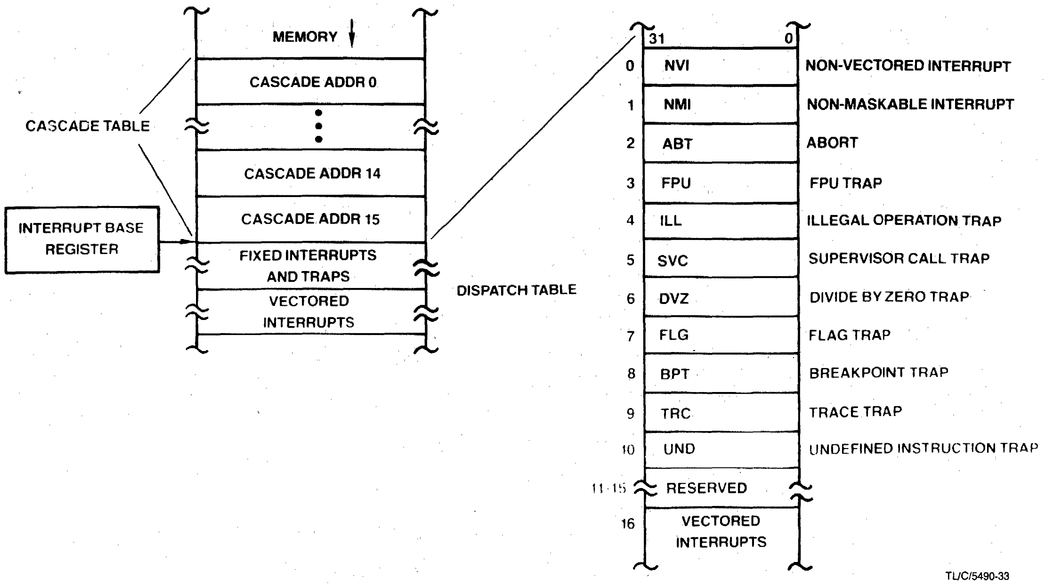
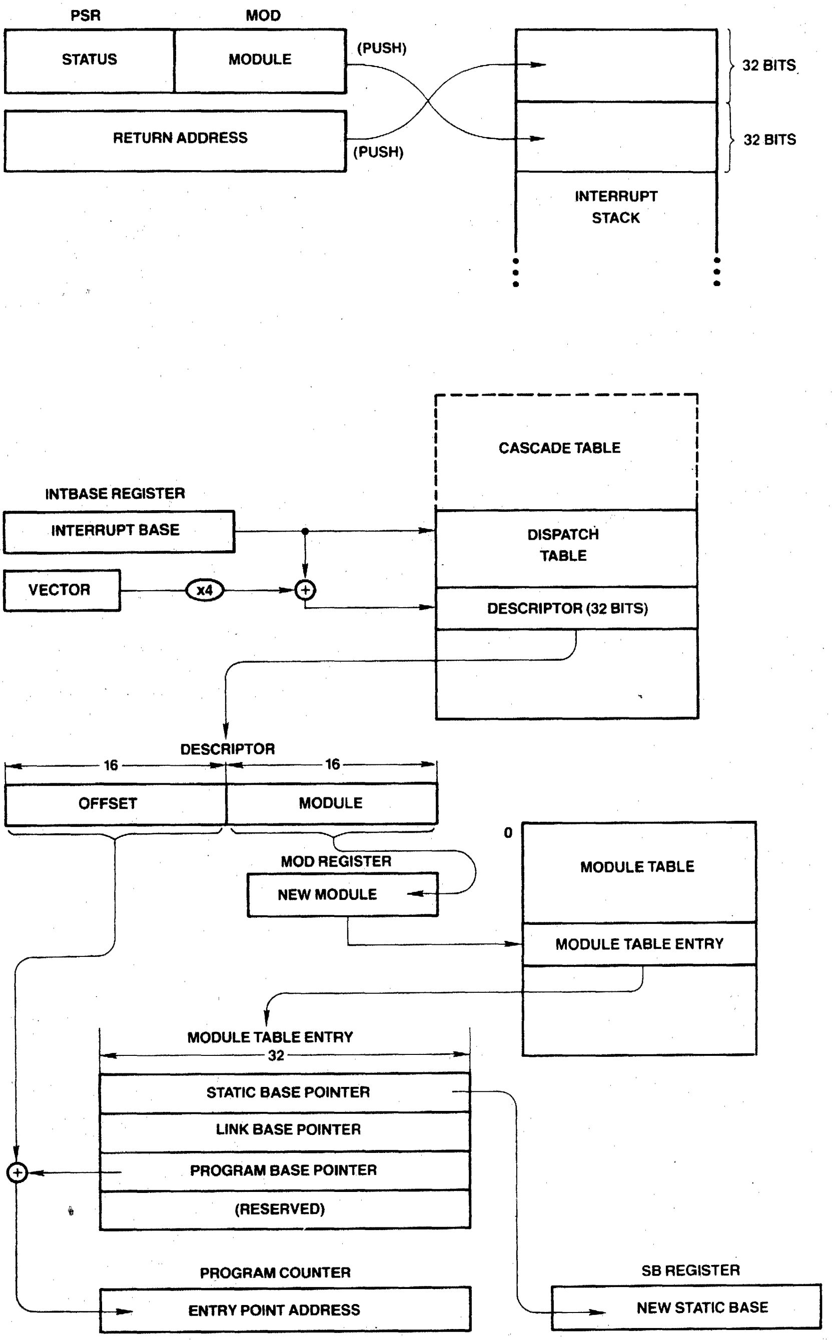


FIGURE 3-22. Interrupt Dispatch and Cascade Tables.

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

Full sequences of events in processing interrupts and traps may be found as follows:

Interrupt on $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pin:	Sec. 3.8.7.1.
Abort Interrupt:	Sec. 3.8.7.4.
Traps (except Trace):	Sec. 3.8.7.2.
Trace Trap:	Sec. 3.8.7.3.



TL/C/5490-34

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.

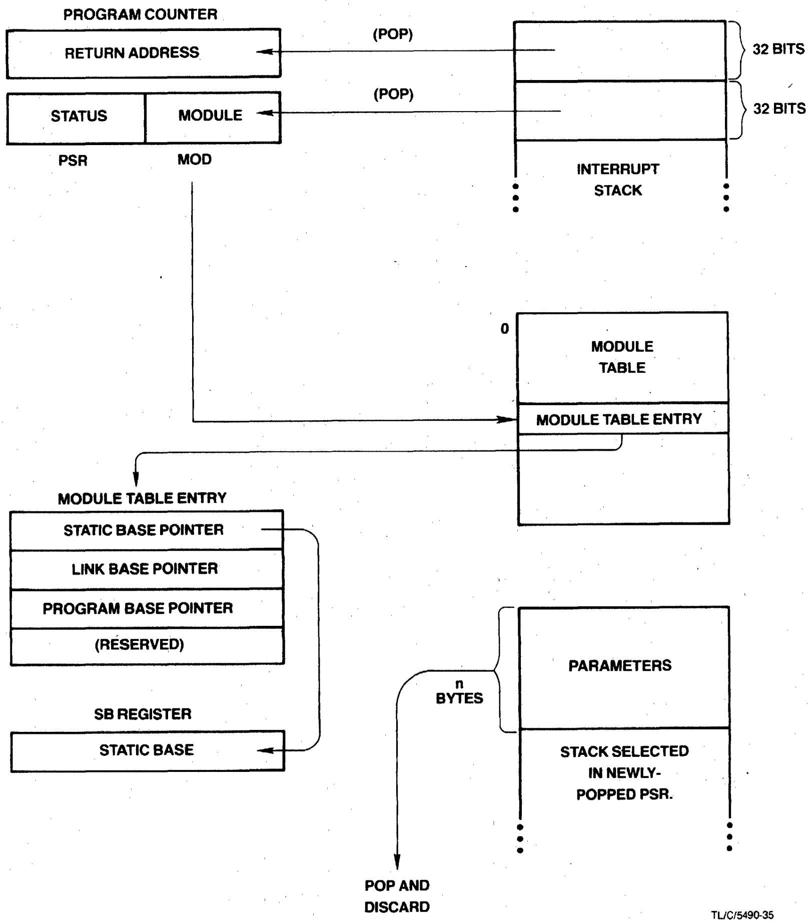
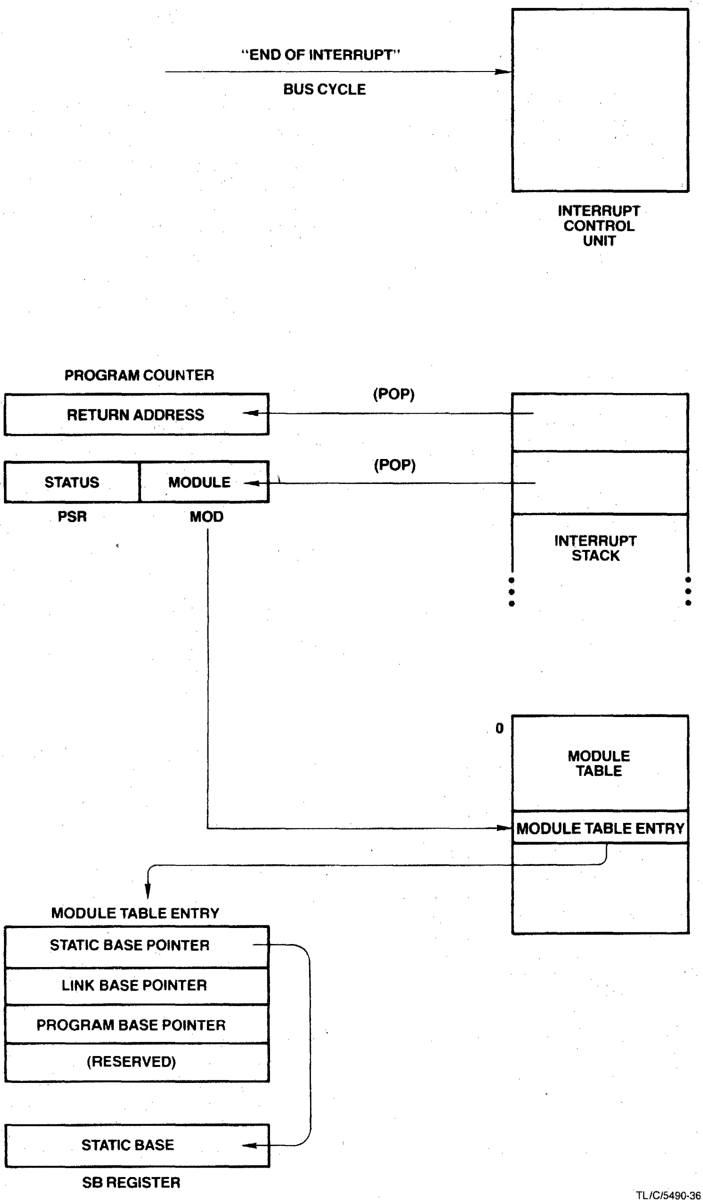


FIGURE 3-24. Return from Trap (RETT n) Instruction Flow.



TL/C/5490-36

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

3.8.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an NS16202 Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Upon receipt of an interrupt request on the $\overline{\text{INT}}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.2) 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 NS16202 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 $\overline{\text{INT}}$ 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 (Sec. 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 (Sec. 3.4.2), whereupon the Master ICU again provides the negative Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Sec. 3.4.2), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

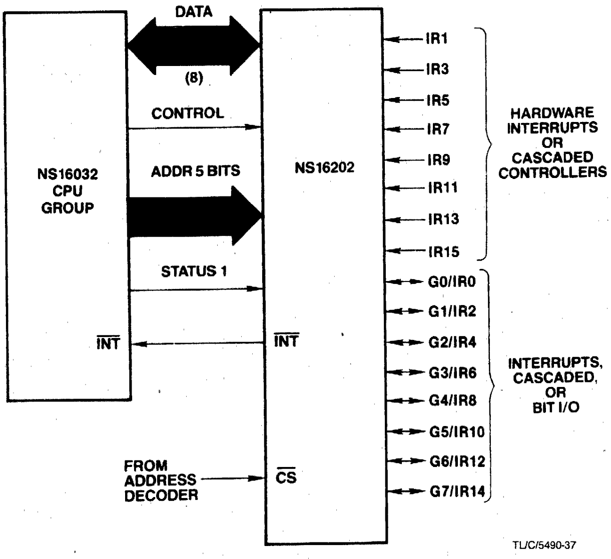


FIGURE 3-26. Interrupt Control Unit Connections (16 Levels).

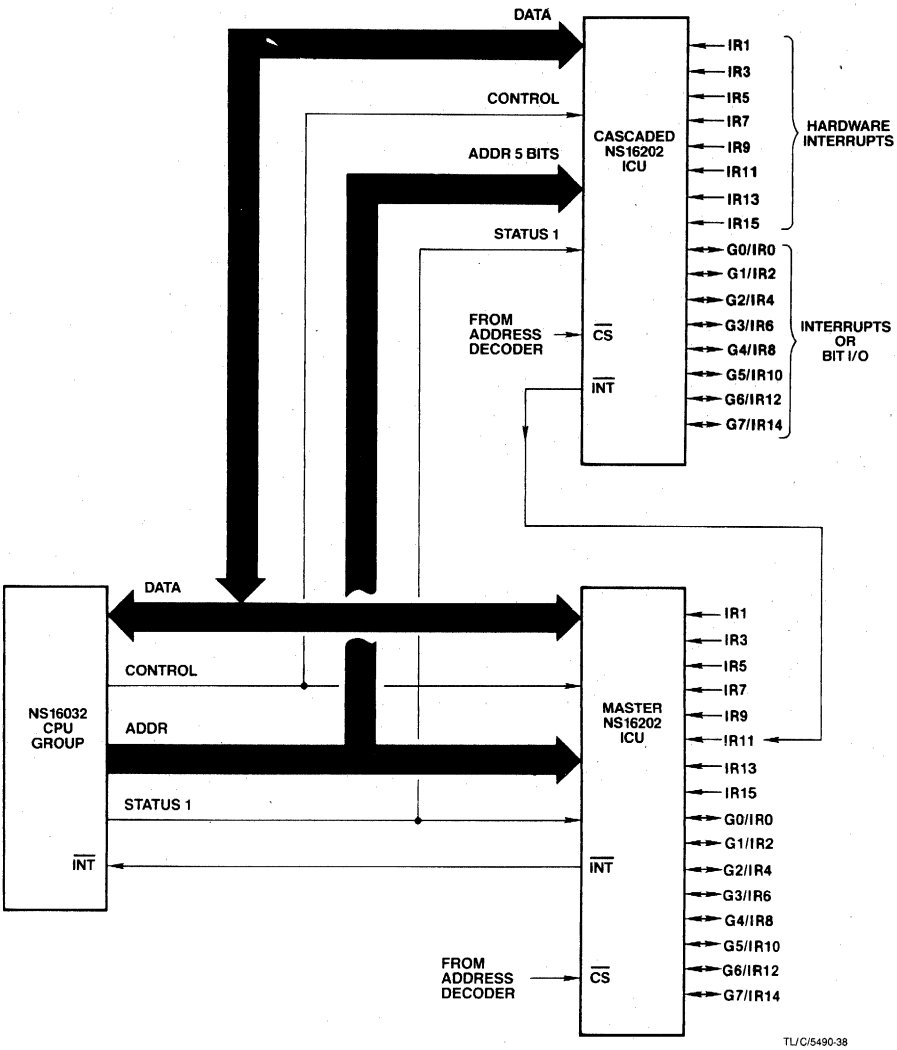


FIGURE 3-27. Cascaded Interrupt Control Unit Connections.

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 (Sec. 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 Sec. 3.8.7.1.

3.8.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) 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 NS16032 CPU are:

Trap (FPU): An exceptional condition was detected by the NS16081 Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Sec. 3.9.1).

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

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

Trap (DVZ): An attempt was made to divide an integer by zero. (The FPU 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 NS16032 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: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in Figure 3-28. 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 sequence followed in processing either Maskable or Non-Maskable interrupts (on the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pins, respectively), see Sec. 3.8.7.1. For Abort interrupts, see Sec. 3.8.7.4. For the Trace Trap, see Sec. 3.8.7.3, and for all other traps see Sec. 3.8.7.2.

3.8.7.1 Maskable/Non-Maskable Interrupt Sequence

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

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

   Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
   a. Read a byte from address $FFFF00_{16}$, 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" ≥ 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) Push MOD Register onto the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)

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

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

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

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

6) 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.

TL/C/5490-39

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.

FPU:	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 NS16032 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 (Sec. 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

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, Sec. 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, Sec. 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

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).

TL/C/5490-40

FIGURE 3-29. Slave Processor Protocol.

solely responsible for memory accesses, these extensions are not sent to the Slave processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Sec. 3.4.2).

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

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

Upon receiving the pulse on $\overline{\text{SPC}}$, the CPU uses $\overline{\text{SPC}}$ to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Sec. 3.4.2). 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 FPU vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2).

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 Programmer's Manual).

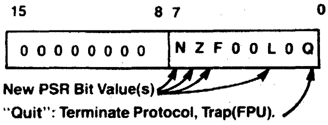
The Operand Issued columns show the sizes of the operands issued to the Floating Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "f" indicates that the instruction specifies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

NOTE:
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.



TL/C/5490-41

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 futher details of the Memory Management Instruction set, see the Programmer's Manual and the NS16082 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 Programmer's Manual and the NS16082 Memory Management Unit Data Sheet.

D = Double Word.

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

N/A = Not Applicable to this instruction.

3.9.4 Custom Slave Instructions

Provided in the NS16032 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
CCMPc	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to Op. 2	none
CATST0 *	addr	N/A	D	N/A	N/A	F
CATST1 *	addr	N/A	D	N/A	N/A	F
LCR *	read.D	N/A	D	N/A	N/A	none
SCR *	write.D	N/A	N/A	N/A	D to Op. 1	none

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

4 AC Electrical Characteristics

4.1 Definitions

All the timing specifications given in this section refer to 50% of the leading or trailing edges of the appropriate clock phase and 0.8V or 2.0V on the appropriate signal as illustrated in *Figures 4-1* and *4-2*, unless specifically stated otherwise.

Abbreviations:
L.E — leading edge
T.E. — trailing edge

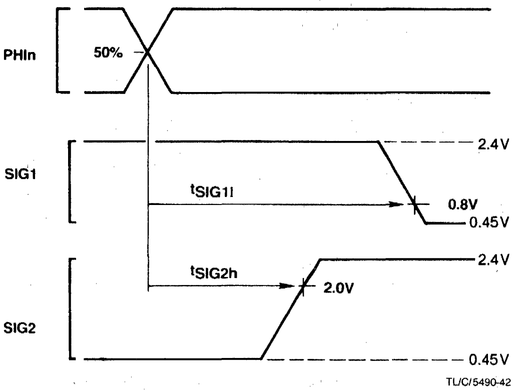


FIGURE 4-1. Timing Specification Standard (Signal Valid After Clock Edge)

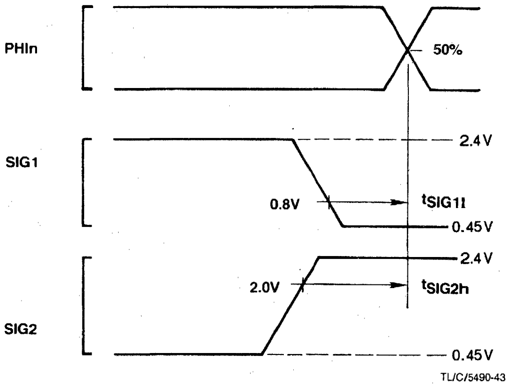


FIGURE 4-2. Timing Specification Standard (Signal Valid Before Clock Edge)

4.2 Timing Tables

4.2.1 Output Signals: Internal Propagation Delays

Maximum times assume capacitive loading of 100pF.

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{ALv}	Address bits 0-15 valid	4-3	after L.E., PHI1 T1			50	ns
t_{ALh}	Address bits 0-15 hold	4-3	after L.E., PHI1 Tmmu or T2	0			ns
t_{Dv}	Data valid (write cycle)	4-3	after L.E., PHI1 T2			50	ns
t_{Dh}	Data hold (write cycle)	4-3	after L.E., PHI1 next T1 or Ti	0			ns
t_{AHv}	Address bits 16-23 valid	4-3	after L.E., PHI1 T1			50	ns
t_{AHh}	Address bits 16-23 hold	4-3	after L.E., PHI1 next T1 or Ti	0			ns
t_{ALADSs}	Address bits 0-15 set up to $\overline{\text{ADS}}$ T.E.	4-4	before $\overline{\text{ADS}}$ reaches 2.0V	20			ns
t_{AHADSs}	Address bits 16-23 set up to $\overline{\text{ADS}}$ T.E.	4-4	before $\overline{\text{ADS}}$ reaches 2.0V	20			ns
t_{ALADSh}	Address bits 0-15 hold from $\overline{\text{ADS}}$ T.E.	4-9	after $\overline{\text{ADS}}$ reaches 2.0V	10			ns
t_{AHADSh}	Address bits 16-23 hold from $\overline{\text{ADS}}$ T.E.	4-8	after $\overline{\text{ADS}}$ reaches 2.0V	10			ns
t_{ALf}	Address bits 0-15 floating (no MMU)	4-4	after L.E., PHI1 T2			25	ns
t_{ALMf}	Address bits 0-15 floating (with MMU)	4-8	after L.E., PHI1 Tmmu			25	ns
t_{AHMf}	Address bits 16-23 floating (with MMU)	4-8	after L.E., PHI1 Tmmu			25	ns
t_{HBEv}	$\overline{\text{HBE}}$ signal valid	4-3	after L.E., PHI1 T1			70	ns
t_{HBEh}	$\overline{\text{HBE}}$ signal hold	4-3	after L.E., PHI1 next T1 or Ti	0			ns
t_{STv}	Status (ST0-ST3) valid	4-3	after L.E., PHI1 T4 (before T1, see note)			45	ns
t_{STh}	Status (ST0-ST3) hold	4-3	after L.E., PHI1 T4 (after T1)	0			ns
t_{DDINv}	$\overline{\text{DDIN}}$ signal valid	4-4	after L.E., PHI1 T1			65	ns
t_{DDINh}	$\overline{\text{DDIN}}$ signal hold	4-4	after L.E., PHI1 next T1 or Ti	0			ns
t_{ADSa}	$\overline{\text{ADS}}$ signal active (low)	4-3	after L.E., PHI1 T1			40	ns
t_{ADSia}	$\overline{\text{ADS}}$ signal inactive	4-3	after T.E., PHI1 T1			45	ns
t_{ADSw}	$\overline{\text{ADS}}$ pulse width	4-3	at 0.8V, both edges	35			ns
t_{DSa}	$\overline{\text{DS}}$ signal active (low)	4-3	after L.E., PHI1 T2			45	ns

4.2.1 Output Signals: Internal Propagation Delays (continued)

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{DSia}	$\overline{\text{DS}}$ signal inactive	4-3	after L.E., PHI1 T4			40	ns
t_{ALf}	AD0-AD15 floating (caused by $\overline{\text{HOLD}}$)	4-5	after L.E., PHI1 T1			25	ns
t_{AHf}	A16-A23 floating (caused by $\overline{\text{HOLD}}$)	4-5	after L.E., PHI1 T1			25	ns
t_{ADSf}	$\overline{\text{ADS}}$ floating (caused by $\overline{\text{HOLD}}$)	4-5	after L.E., PHI1 Ti			55	ns
t_{HBEf}	$\overline{\text{HBE}}$ floating (caused by $\overline{\text{HOLD}}$)	4-5	after L.E., PHI1 Ti			55	ns
t_{DDINf}	$\overline{\text{DDIN}}$ floating (caused by $\overline{\text{HOLD}}$)	4-5	after L.E., PHI1 Ti			55	ns
t_{HLDAa}	$\overline{\text{HLDA}}$ signal active (low)	4-5	after L.E., PHI1 Ti			75	ns
t_{HLDAia}	$\overline{\text{HLDA}}$ signal inactive	4-7	after L.E., PHI1 Ti			75	ns
t_{ADSr}	$\overline{\text{ADS}}$ signal returns from floating (caused by $\overline{\text{HOLD}}$)	4-7	after L.E., PHI1 Ti			55	ns
t_{HBEr}	$\overline{\text{HBE}}$ signal returns from floating (caused by $\overline{\text{HOLD}}$)	4-7	after L.E., PHI1 Ti			55	ns
t_{DDINr}	$\overline{\text{DDIN}}$ signal returns from floating (caused by $\overline{\text{HOLD}}$)	4-7	after L.E., PHI1 Ti			55	ns
t_{ALf}	AD0-AD15 floating (caused by $\overline{\text{FLT}}$)	4-8	after L.E., PHI1 Tf			30	ns
t_{DDINf}	$\overline{\text{DDIN}}$ signal floating (caused by $\overline{\text{FLT}}$)	4-8	after $\overline{\text{FLT}}$ reaches 0.8V			50	ns
t_{HBEl}	$\overline{\text{HBE}}$ signal low (caused by $\overline{\text{FLT}}$)	4-8	after $\overline{\text{FLT}}$ reaches 0.8V			65	ns
t_{DDINr}	$\overline{\text{DDIN}}$ signal returns from floating (caused by $\overline{\text{FLT}}$)	4-9	after $\overline{\text{FLT}}$ reaches 2.0V			50	ns
t_{HBEr}	$\overline{\text{HBE}}$ signal returns from floating (caused by $\overline{\text{FLT}}$)	4-9	after $\overline{\text{FLT}}$ reaches 2.0V			75	ns
t_{SPCa}	$\overline{\text{SPC}}$ output active (low)	4-12	after L.E., PHI1 T1			35	ns
t_{SPCia}	$\overline{\text{SPC}}$ output inactive	4-12	after L.E., PHI1 T4			35	ns
t_{SPCnf}	$\overline{\text{SPC}}$ output nonforcing	4-14	after L.E., PHI2 T4			10	ns
t_{Dv}	Data valid (slave processor write)	4-12	after L.E., PHI1 T1			50	ns
t_{Dh}	Data hold (slave processor write)	4-12	after L.E., PHI1 next T1 or Ti	0			ns
t_{PFSw}	$\overline{\text{PFS}}$ pulse width	4-17	at 0.8V, both edges	70			ns
t_{PFSa}	$\overline{\text{PFS}}$ pulse active (low)	4-17	after L.E., PHI2			50	ns
t_{PFSia}	$\overline{\text{PFS}}$ pulse inactive	4-17	after L.E., PHI2			50	ns
t_{ILOs}	$\overline{\text{ILO}}$ signal setup	4-19a	before L.E., PHI1 T1 of first interlocked write cycle	0			ns
t_{ILOh}	$\overline{\text{ILO}}$ signal hold	4-19b	after L.E., PHI1 T3 of last interlocked read cycle	0			ns
t_{ILOa}	$\overline{\text{ILO}}$ signal active (low)	4-20	after L.E., PHI1			70	ns
t_{ILOia}	$\overline{\text{ILO}}$ signal inactive	4-20	after L.E., PHI1			70	ns
t_{USs}	U/$\overline{\text{S}}$ signal setup	4-21	before T.E., PHI1 T4 or Ti	10			ns
t_{USh}	U/$\overline{\text{S}}$ signal hold	4-21	after L.E., PHI1 T1	2			t_{Cp}
t_{NSPF}	Nonsequential fetch to next $\overline{\text{PFS}}$ clock cycle	4-18b	after L.E., PHI1 T1	4			t_{Cp}
t_{PFNS}	$\overline{\text{PFS}}$ clock cycle to next nonsequential fetch	4-18a	before L.E., PHI1 T1	4			t_{Cp}
t_{LXPF}	Last operand transfer of an instruction to next $\overline{\text{PFS}}$ clock cycle	4-28	before L.E., PHI1 T1 of first bus cycle of transfer	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...".

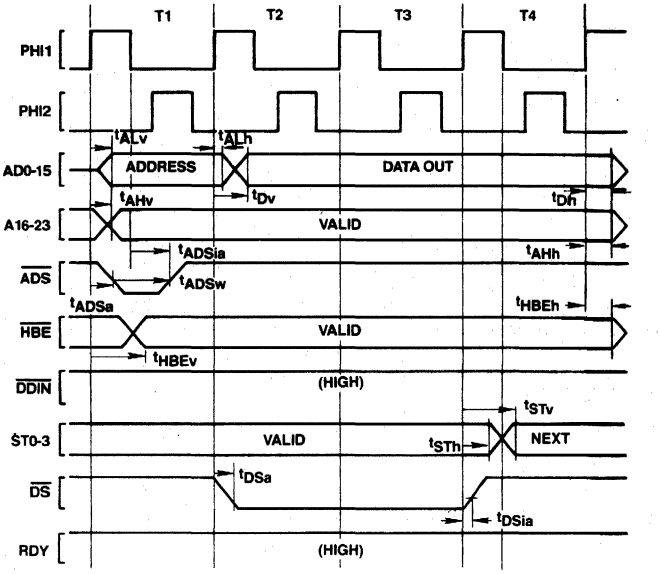
4.2.2 Input Signal Requirements:

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{PWR}	Power stable to $\overline{RST}$ T.E.	4-24	after V_{CC} reaches 4.5V	50			μs
t_{DIs}	Data in setup (read cycle)	4-4	before T.E., PHI2 T3	10			ns
t_{DIh}	Data in hold (read cycle)	4-4	after T.E., PHI2 T3	10			ns
t_{HLDa}	$\overline{HOLD}$ active (low) setup time (See note)	4-5	before T.E., PHI2 TX1	25			ns
t_{HLDia}	$\overline{HOLD}$ inactive setup time	4-7	before T.E., PHI2 Ti	25			ns
t_{HLDh}	$\overline{HOLD}$ hold time	4-5	after L.E., PHI1 TX2	0			ns
t_{FLTa}	$\overline{FLT}$ active (low) setup time	4-8	before T.E., PHI2 Tmmu	25			ns
t_{FLTia}	$\overline{FLT}$ inactive setup time	4-9	before T.E., PHI2 T2	25			ns
t_{RDYs}	RDY setup time	4-10, 4-11	before T.E., PHI2 T2 or T3	15			ns
t_{RDYh}	RDY hold time	4-10, 4-11	after T.E., PHI1 T3	0			ns
t_{ABTs}	$\overline{ABT}$ setup time ($\overline{FLT}$ inactive)	4-22	before T.E., PHI2 Tmmu	20			ns
t_{ABTs}	$\overline{ABT}$ setup time ($\overline{FLT}$ active)	4-23	before T.E., PHI2 T2	20			ns
t_{ABTh}	$\overline{ABT}$ hold time	4-22	after L.E., PHI1	0			ns
t_{RSTs}	$\overline{RST}$ setup time	4-24, 4-25	before T.E., PHI1	20			ns
t_{RSTw}	$\overline{RST}$ pulse width	4-25	at 0.8V (both edges)	64			t_{Cp}
t_{INTs}	$\overline{INT}$ setup time	4-26	before T.E., PHI1	20			ns
t_{NMIw}	$\overline{NMI}$ pulsewidth	4-27	at 0.8V (both edges)	40			ns
t_{DIs}	Data setup (slave read cycle).	4-13	before T.E., PHI2 T1	10			ns
t_{DIh}	Data hold (slave read cycle)	4-13	after T.E., PHI2 T1	10			ns
t_{SPCw}	$\overline{SPC}$ pulse width (from slave processor)	4-12	at 0.8V (both edges)	20			ns
t_{ATs}	AT/$\overline{SPC}$ setup for address translation strap	4-15	before L.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	1			t_{Cp}
t_{ATh}	AT/$\overline{SPC}$ hold for address translation strap	4-15	after T.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	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.

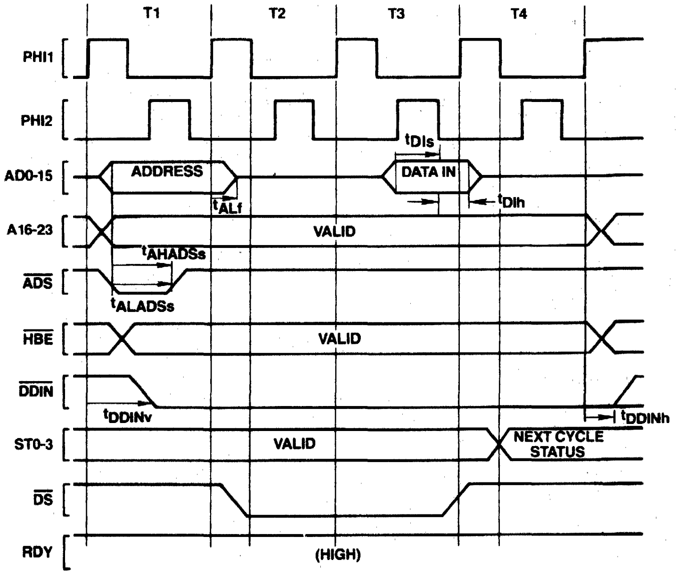
4.2.3 Clocking Requirements:

Name	Description	Figure	Reference/Conditions	Min.	Typ.	Max.	Unit
t_{CLr}	PHI1, PHI2 rise time	4-16	to V_{CH} (see page 2)			5	ns
t_{CLf}	PHI1, PHI2 fall time	4-16	from 90–10% of V_{CH} (see page 2)			5	ns
t_{CLh}	PHI1, PHI2 high time	4-16		0.4			t_{Cp}
t_{CLl}	PHI1, PHI2 low time	4-16		0.35			t_{Cp}
t_{Cp}	Clock period	4-16		100		5000	ns
t_{OVL}	Non-overlap time	4-16	at 10% of V_{CH} (see page 2)	0			ns



TL/C/5490-44

FIGURE 4-3. Write Cycle.



TL/C/5490-45

FIGURE 4-4. Read Cycle.

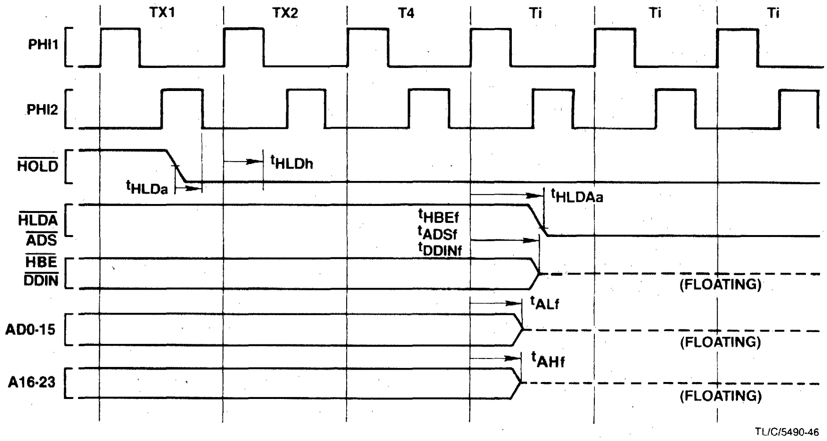


FIGURE 4-5. Floating by HOLD Timing (CPU Not Idle Initially).

Note that whenever the CPU is not idling (not in Ti), the HOLD request (HOLD low) must be active tHLDa before the trailing edge of PHI2 of the clock cycle that appears two clock cycles before T4 (TX1) and stay low until tHLDh after the leading edge of PHI1 of the clock cycle that precedes T4 (TX2) for the request to be acknowledged.

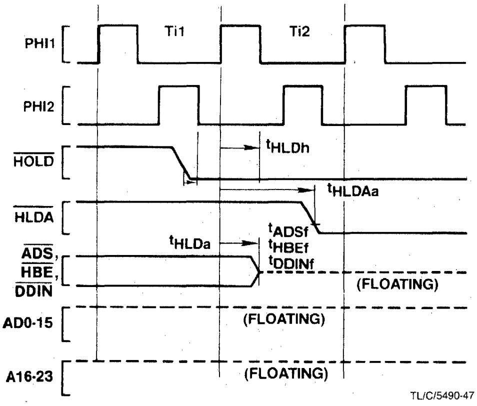


FIGURE 4-6. Floating by HOLD Timing (CPU Initially Idle).

Note that during Ti1 the CPU is already idling.

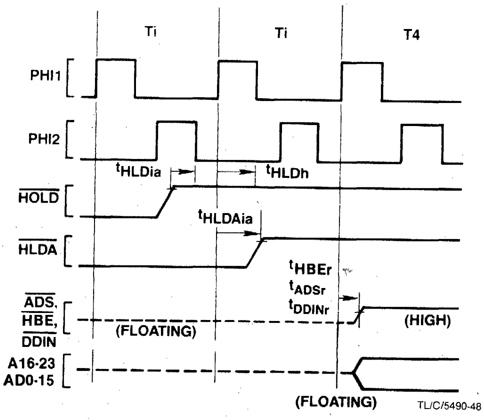


FIGURE 4-7. Release from HOLD.

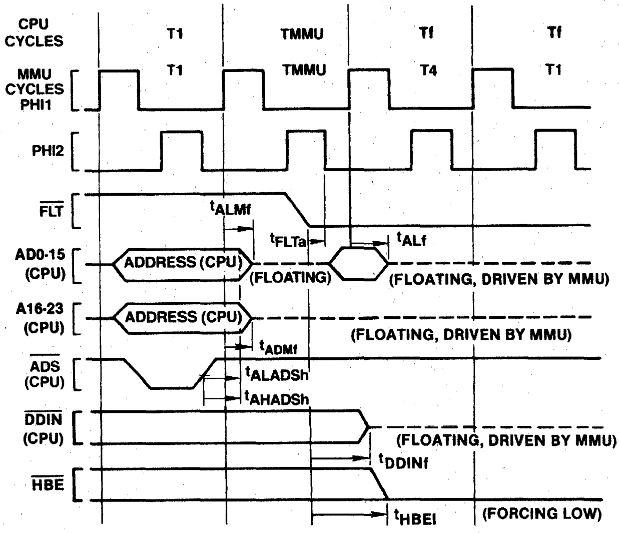
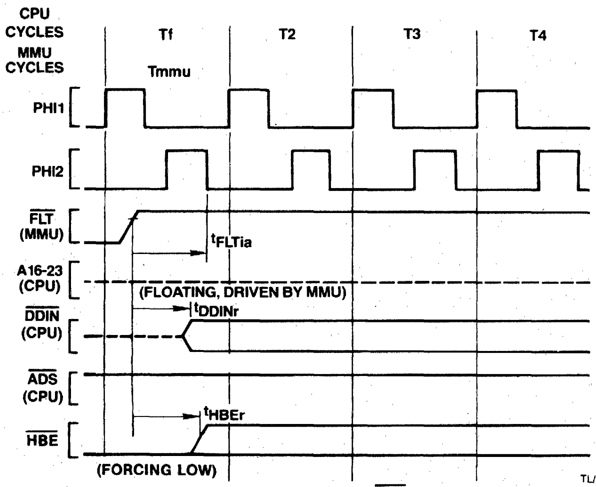


FIGURE 4-8. $\overline{\text{FLT}}$ Initiated Float Cycle Timing.

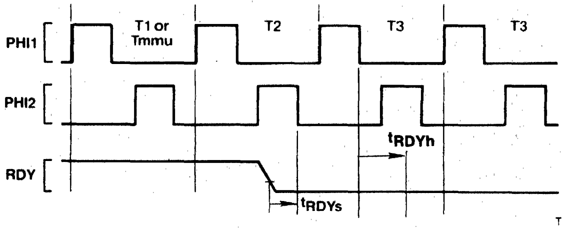
TL/C/5490-49



TL/C/5490-50

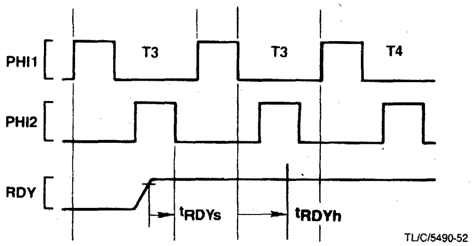
FIGURE 4-9. Release from $\overline{\text{FLT}}$ Timing.

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



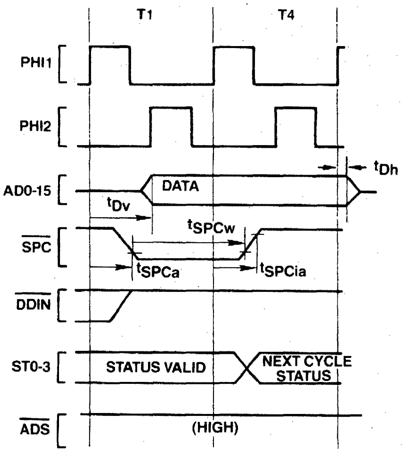
TL/C/5490-51

FIGURE 4-10. Ready Sampling (CPU Initially READY).



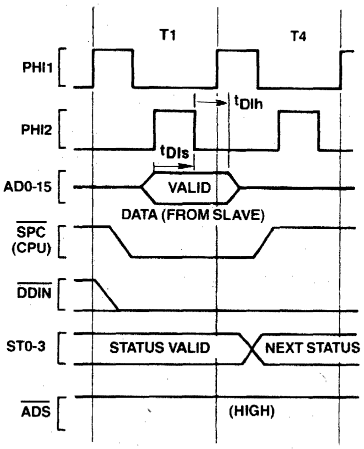
TL/C/5490-52

FIGURE 4-11. Ready Sampling (CPU Initially NOT READY).



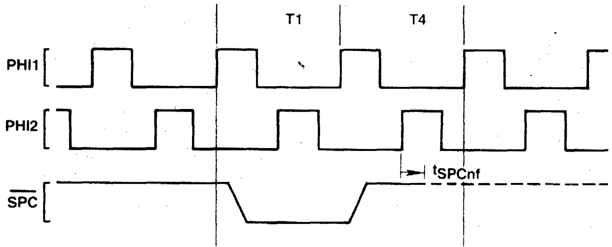
TL/C/5490-53

FIGURE 4-12. Slave Processor Write Timing.



TL/C/5490-54

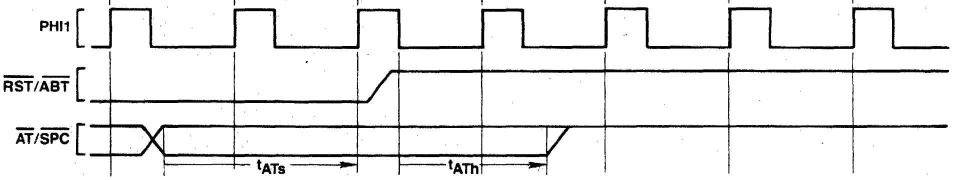
FIGURE 4-13. Slave Processor Read Timing.



TL/C/5490-55

FIGURE 4-14. $\overline{\text{SPC}}$ Non-Forcing Delay.

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



TL/C/5490-56

FIGURE 4-15. Reset Configuration Timing.

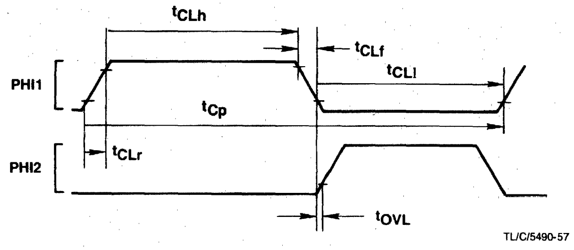


FIGURE 4-16. Clock Waveforms.

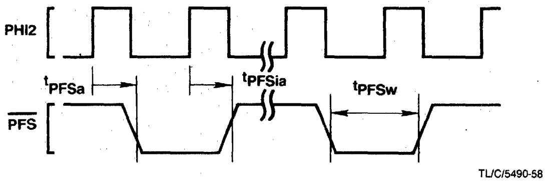


FIGURE 4-17. Relationship of $\overline{\text{PFS}}$ to Clock Cycles.

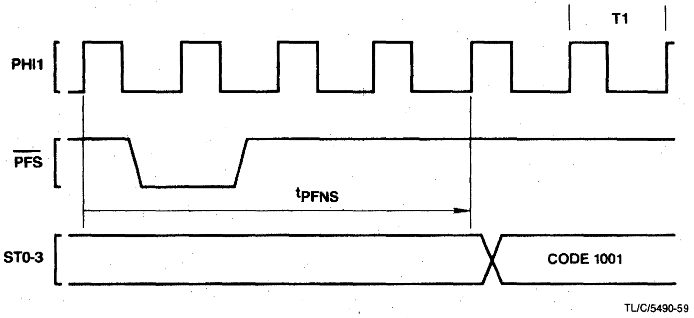


FIGURE 4-18a. Guaranteed Delay, $\overline{\text{PFS}}$ to Non-Sequential Fetch.

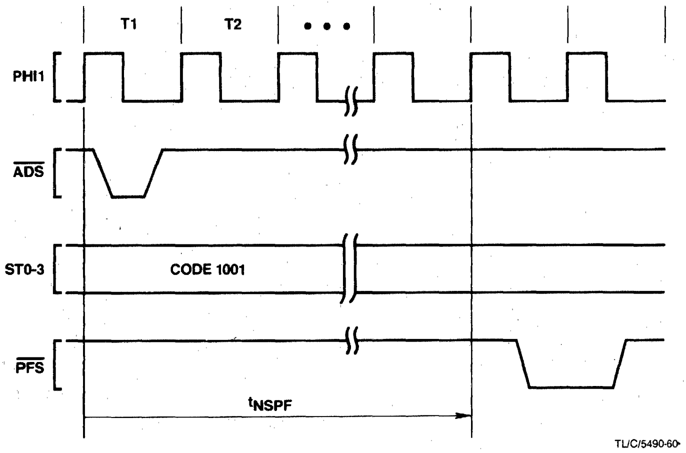
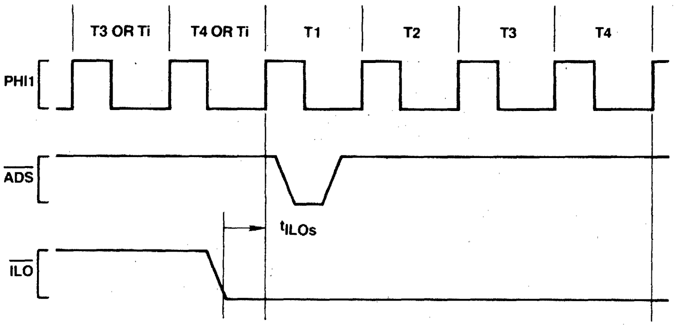
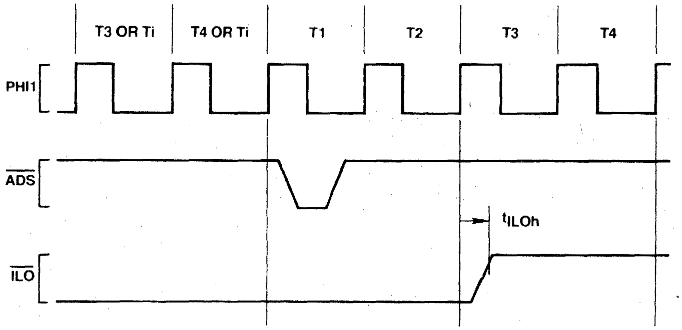


FIGURE 4-18b. Guaranteed Delay, Non-Sequential Fetch to $\overline{\text{PFS}}$.



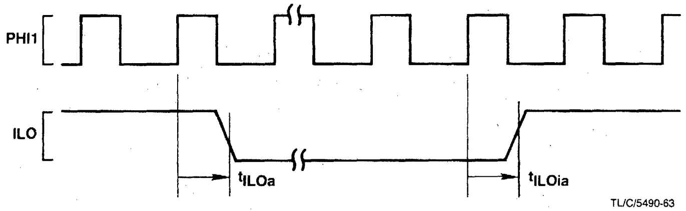
TL/C/5490-61

FIGURE 4-19a. Relationship of $\overline{\text{ILO}}$ to First Operand Cycle of an Interlocked Instruction.



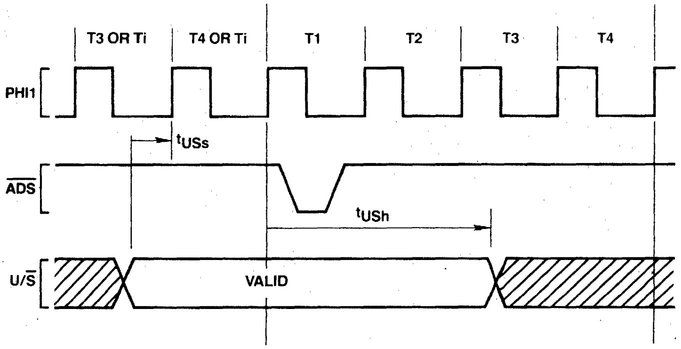
TL/C/5490-62

FIGURE 4-19b. Relationship of $\overline{\text{ILO}}$ to Last Operand Cycle of an Interlocked Instruction.



TL/C/5490-63

FIGURE 4-20. Relationship of $\overline{\text{ILO}}$ to Any Clock Cycle.



TL/C/5490-64

FIGURE 4-21. U/$\overline{\text{S}}$ Relationship to Any Bus Cycle — Guaranteed Valid Interval.

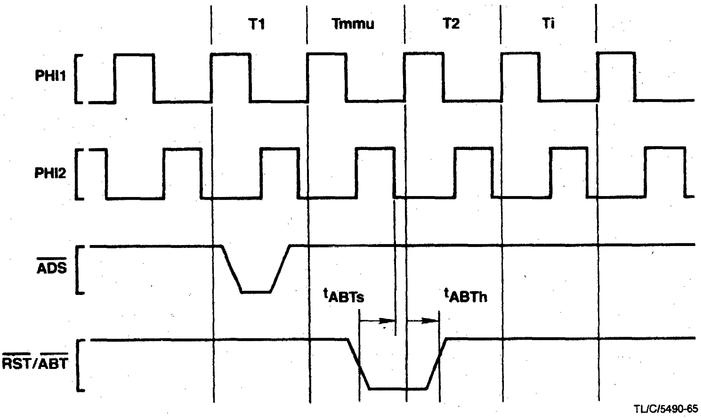


FIGURE 4-22. Abort Timing, FLT Not Applied.

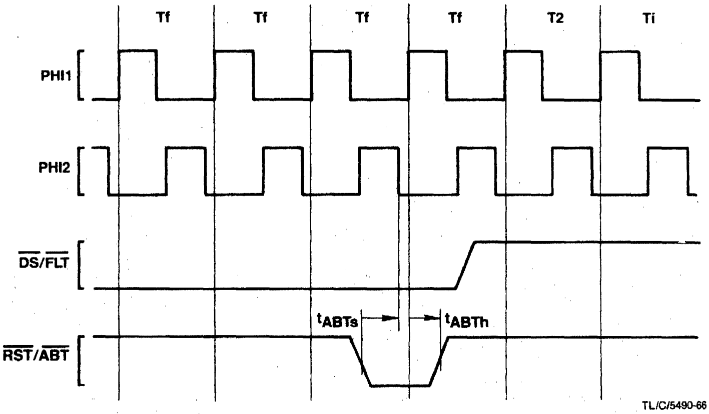


FIGURE 4-23. Abort Timing, FLT Applied.

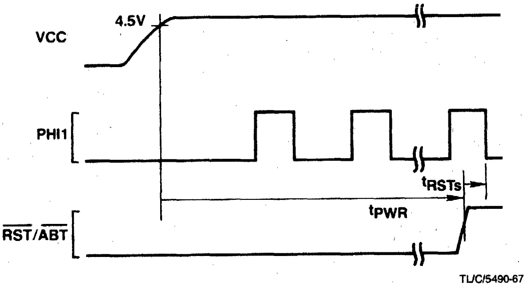


FIGURE 4-24. Power-On Reset.

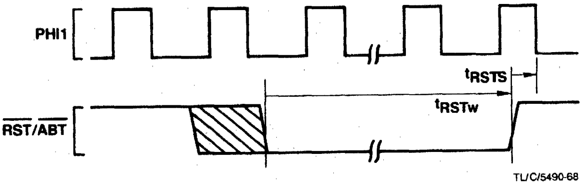


FIGURE 4-25. Non-Power-On Reset.

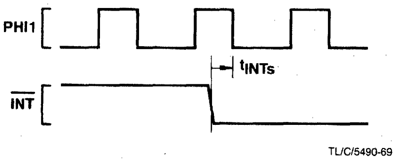


FIGURE 4-26. INT Interrupt Signal Detection.

Violation of tINTs timing is allowed, but detection then occurs one clock cycle later.

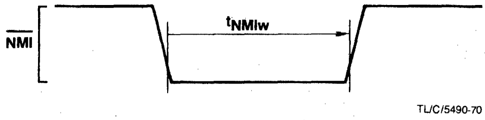


FIGURE 4-27. NMI Interrupt Signal Timing.

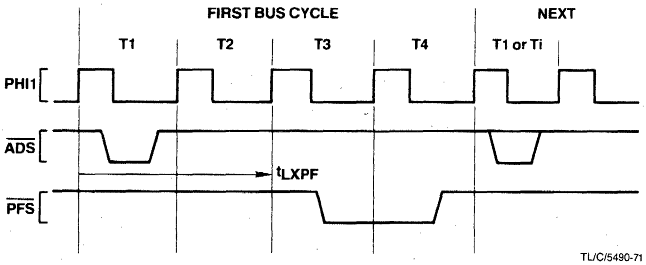


FIGURE 4-28. Relationship Between Last Data Transfer of an Instruction and PFS Pulse of Next Instruction.

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

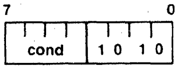
U/W	B	T

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

Configuration bits, in SETCFG:

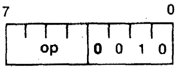
C	M	F	I

mreg: MMU Register number, in LMR, SMR.
0000 = BPR0
0001 = BPR1
0010 = (Reserved)
0011 = (Reserved)
0100 = PF0
0101 = PF1
0110 = (Reserved)
0111 = (Reserved)
1000 = SC
1001 = (Reserved)
1010 = MSR
1011 = BCNT
1100 = PTB0
1101 = PTB1
1110 = (Reserved)
1111 = EIA



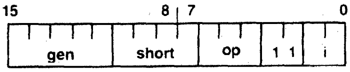
Format 0

Bcond (BR)



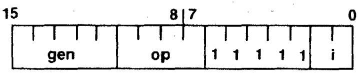
Format 1

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

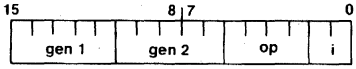
ADDQ	–000	ACB	–100
CMPQ	–001	MOVQ	–101
SPR	–010	LPR	–110
Scond	–011		



Format 3

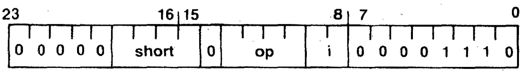
CXPD	−0000	ADJSP	−1010
BICPSR	−0010	JSR	−1100
JUMP	−0100	CASE	−1110
BISPSR	−0110		

Trap (UND) on XXX1, 1000



Format 4

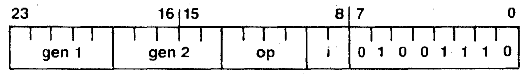
ADD	−0000	SUB	−1000
CMP	−0001	ADDR	−1001
BIC	−0010	AND	−1010
ADDC	−0100	SUBC	−1100
MOV	−0101	TBIT	−1101
OR	−0110	XOR	−1110



Format 5

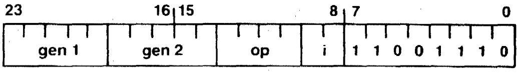
MOVS	−0000	SETCFG	−0010
CMPS	−0001	SKPS	−0011

Trap (UND) on 1XXX, 01XX



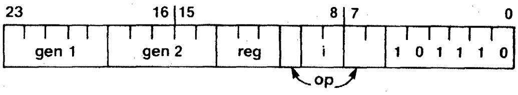
Format 6

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



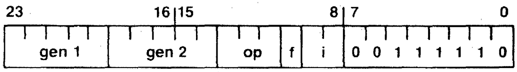
Format 7

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



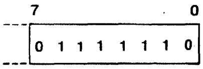
Format 8

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



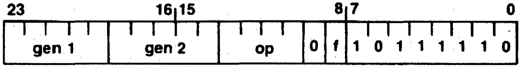
Format 9

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



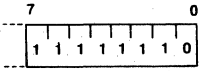
Format 10

Trap (UND) Always



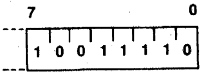
Format 11

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



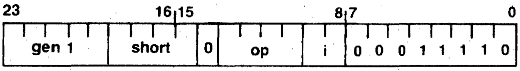
Format 12

Trap (UND) Always



Format 13

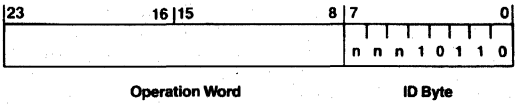
Trap (UND) Always



Format 14

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

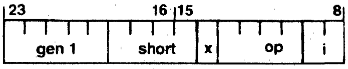
Trap (UND) on 01XX, 1XXX



Format 15
(Custom Slave)

nnn **Operation Word Format**

000

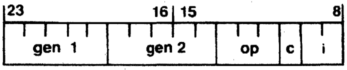


Format 15.0

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

Trap (UND) on all others

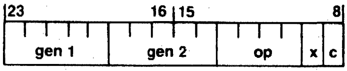
001



Format 15.1

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

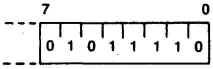
101



Format 15.5

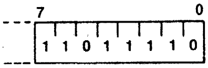
CCAL0	–0000	CCAL3	–1000
CMOV0	–0001	Trap (UND)	–1010
CCMP	–0010	Trap (UND)	–1011
CCAL1	–0100	CCAL2	–1100
CMOV2	–0101	CMOV1	–1101
Trap (UND)	–0110	Trap (UND)	–1110
Trap (UND)	–0111	Trap (UND)	–1111

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



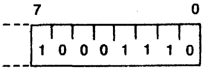
Format 16

Trap (UND) Always



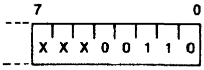
Format 17

Trap (UND) Always



Format 18

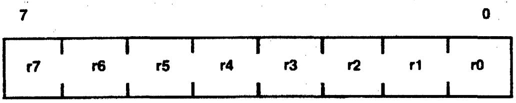
Trap (UND) Always



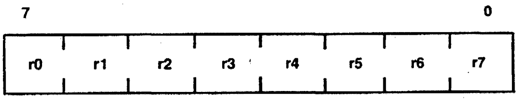
Format 19

Trap (UND) Always

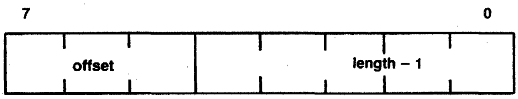
Implied Immediate Encodings:



Register Mask, appended to SAVE, ENTER



Register Mask, appended to RESTORE, EXIT



Offset/Length Modifier appended to INSS, EXTS

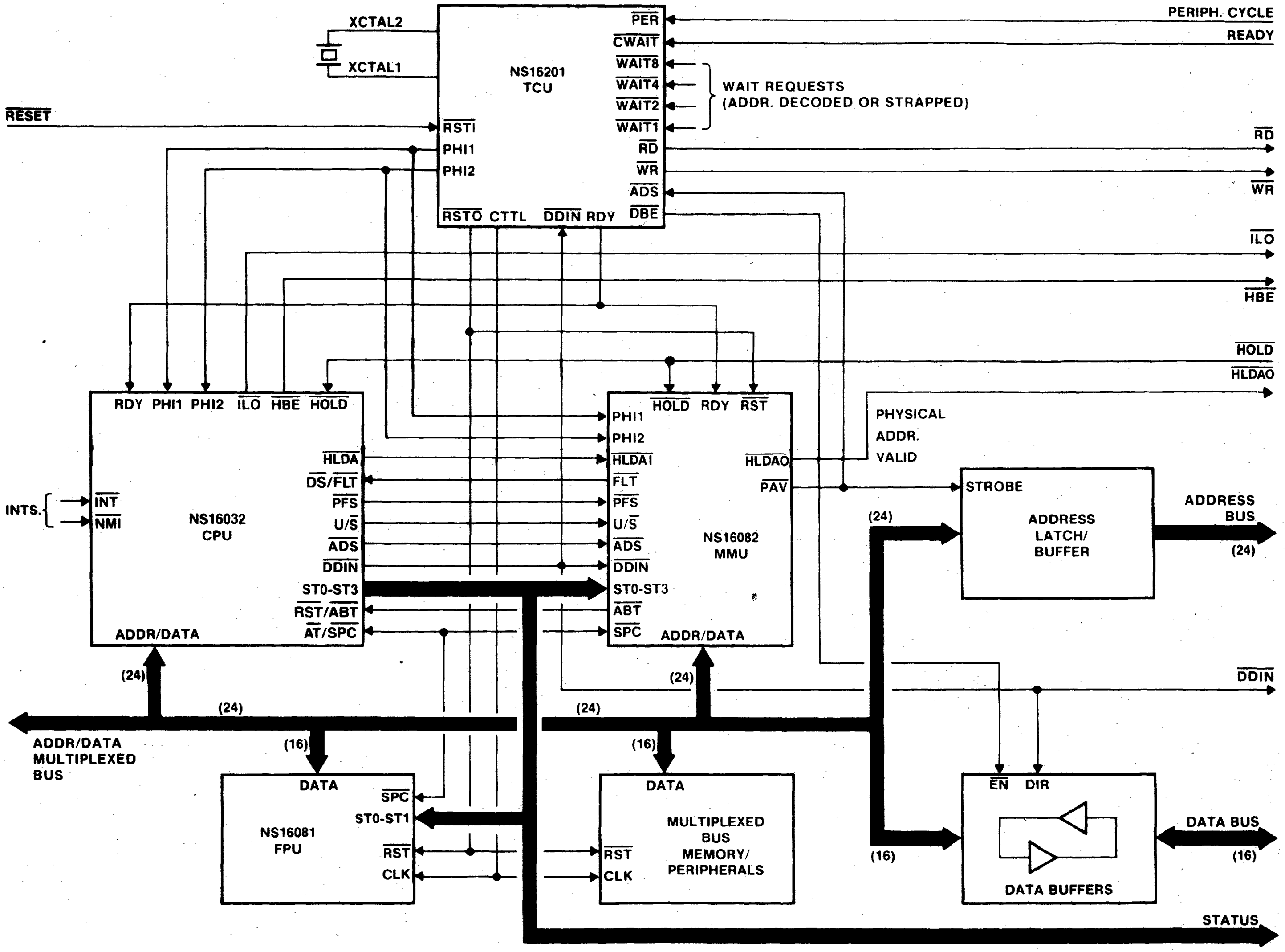


FIGURE B-1. System Connection Diagram.

TL/C/5054-72

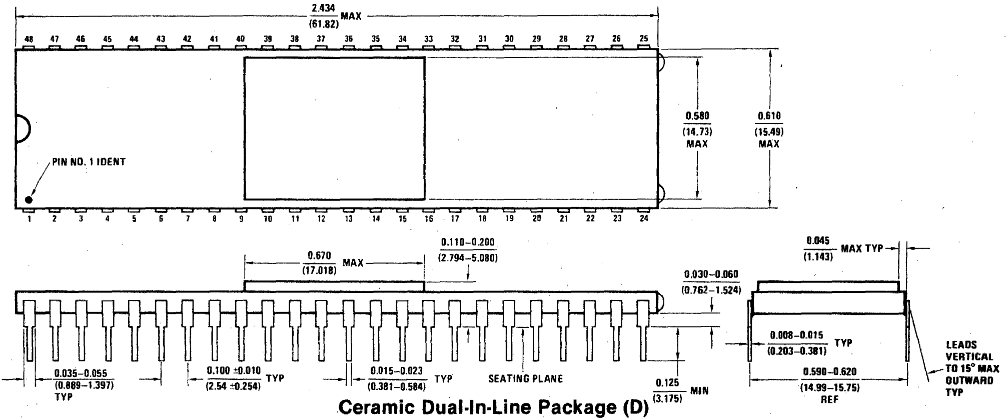
ORDERING INFORMATION

NS16032D-10
NS16032N-10

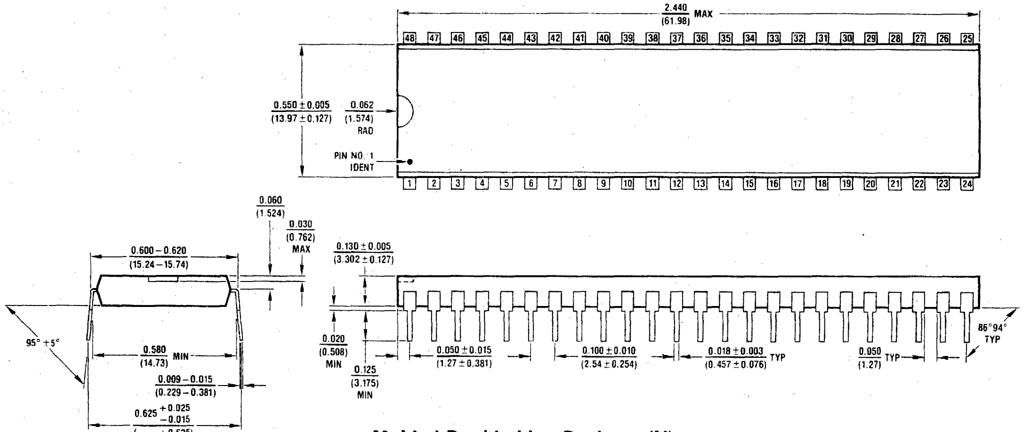
DEVELOPMENT TOOLS ORDERING INFORMATION

DB16000	Evaluation Board
NSX-16	Cross Software Package (VAX/VMS)
SFW-90-A010	Cross Software Package (STARPLEX II™)
NS-ISE-16	In-System Emulator (VAX/VMS)
SPM-90-A1632	In-System Emulator (STARPLEX II)

Physical Dimensions inches (millimeters)



Ceramic Dual-In-Line Package (D)
NS Package D48A



Molded Dual-In-Line Package (N)
NS Package N48A

National Semiconductor

PRELIMINARY

NS32032-6, NS32032-4 High-Performance Microprocessors

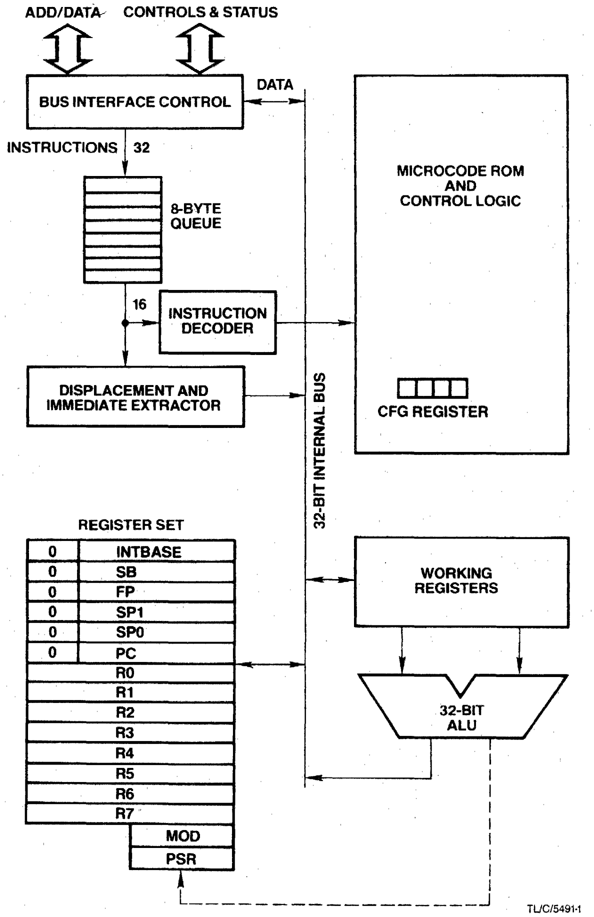
General Description

The NS32032 functions as a central processing unit (CPU) in National Semiconductor's NS16000™ microprocessor family. It has been designed to optimally support microprocessor users who need the ability to use a large addressing space for large programs and/or large data structures. Because large programs must realistically be generated and maintained in high-level languages, the NS16000 architecture provides for very efficient compilation while remaining easy to program at the assembler level for optimizations. NS16000 architecture provides for full virtual memory capability, in conjunction with the NS16082 Memory Management Unit (MMU). High performance floating-point instructions are provided with the NS16081 Floating-Point Unit (FPU). The NS32032-4 and NS32032-6 have different timing parameters. Refer to Section 4 for timing specifications.

Features

- 32-bit Architecture and Implementation
- 32-bit Data Bus
- 16-Mbyte Uniform Addressing Space
- Powerful Instruction Set
  - General 2-Address Capability
  - Very High Degree of Symmetry
  - Addressing Modes Optimized for High Level Lanuguage References
- NS16000 Slave Processor Support
- High-Speed XMOS™ Technology
- Single 5V Supply
- 68-pin Leadless Chip Carrier

NS32032 CPU Block Diagram



TL/C/5491-1

Absolute Maximum Ratings

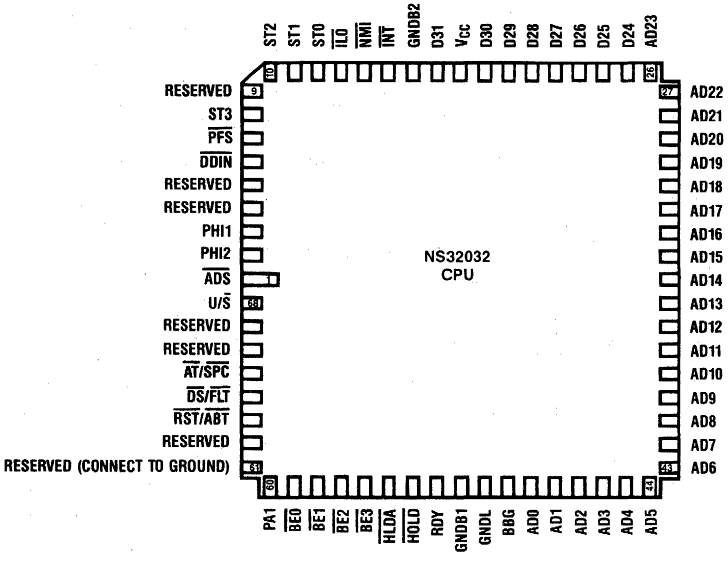
Temperature under bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All input or output voltages with respect to GND	−0.5V to +7V
Power Dissipation	1.5 Watt

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 DC Electrical Characteristics.*

DC Electrical Characteristics: NS32032-4 $T_A = 0$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, GND = 0V
NS32032-6 $T_A = 0$ to $+70°C$, $V_{CC} = 5V \pm 5\%$, GND = 0V

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Unit
V_{IH}	Logical 1 Input Voltage		2.0		V_{CC}+0.5	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{CH}	Logical 1 Clock Voltage	PHI1, PHI2 pins only	V_{CC}−0.5		V_{CC}+0.5	V
V_{CL}	Logical 0 Clock Voltage	PHI1, PHI2 pins only	−0.5		0.3	V
V_{CLT}	Logical 0 Clock Voltage, Transient (ringing tolerance)	PHI1, PHI2 pins only	−0.5		0.6	V
V_{OH}	Logical 1 Output Voltage	$I_{OH} = -400\mu A$	2.4			V
V_{OL}	Logical 0 Output Voltage	$I_{OL} = 2mA$			0.45	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	$0 \leqslant V_{IN} \leqslant V_{CC}$, All inputs except PHI1, PHI2, $\overline{AT}/\overline{SPC}$	−20		20	μA
$I_{O(OFF)}$	Output Leakage Current	$0.4 \leqslant V_{OUT} \leqslant V_{CC}$	−20		20	μA
I_{CC}	Active Supply Current	$I_{OUT} = 0$, $T_A = 25°C$		180	300	mA

Connection Diagram



Bottom View

TL/C/5491-2

1 NS32032 Pin Descriptions

The following is a brief description of all NS32032 pins. The descriptions reference portions of the Functional Description, Section 3.

Unless otherwise indicated (see pin 34) reserved pins should be left open.

1.1 SUPPLIES

Power (V_{CC}): +5V Positive Supply. Sec. 3.1.

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

Buffer Ground #1 (GNDB1): Ground reference for half of the on-chip drivers connected to output pins. Sec. 3.1.

Buffer Ground #2 (GNDB2): Ground reference for the other half of on-chip drivers connected to output pins. Sec. 3.1.

Back-Bias Generator (BBG): Output of on-chip substrate voltage generator. Sec. 3.1.

1.2 INPUT SIGNALS

Clocks (PHI1, PHI2): Two-phase clocking signals. Sec. 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. Sec. 3.4.1.

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

Interrupt ($\overline{\text{INT}}$): Active low. Maskable Interrupt request. Sec. 3.8.

Non-Maskable Interrupt ($\overline{\text{NMI}}$): Active low. Non-Maskable Interrupt request. Sec. 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, Sec. 3.5.4. If held longer, it initiates a Reset, Sec. 3.3.

Physical Address, bit 1 (PA1): Active high. This input is connected to address A1. It is used during MMU cycles to tell the CPU which word the MMU is accessing, so the appropriate byte enable control signals can be activated.

1.3 OUTPUT SIGNALS

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

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

Byte Enable ($\overline{\text{BE0}}$–$\overline{\text{BE3}}$): Active low. Four control signals enabling data transfers on individual bus bytes. Sec. 3.4.3.

Status (ST0-ST3): Active high. Bus cycle status code, ST0 least significant. Sec. 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. Sec. 3.6.

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

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

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

1.4 INPUT-OUTPUT SIGNALS

Address/Data 0-23 (AD0-AD23): Active high. Multiplexed Address/Data information. Bit 0 is the least significant bit of each. Sec. 3.4.

Data Bits 24-31 (D24-D31): Active high. The high order 8 bits of the data bus.

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 an instruction. Sec. 3.4.6; Sec. 3.9. Sampled on trailing edge of Reset pulse as Address Translation Strap. Sec. 3.5.1.

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

2 Architectural Description

2.1 PROGRAMMING MODEL

The NS16000 architecture includes 16 registers on the NS32032 CPU.

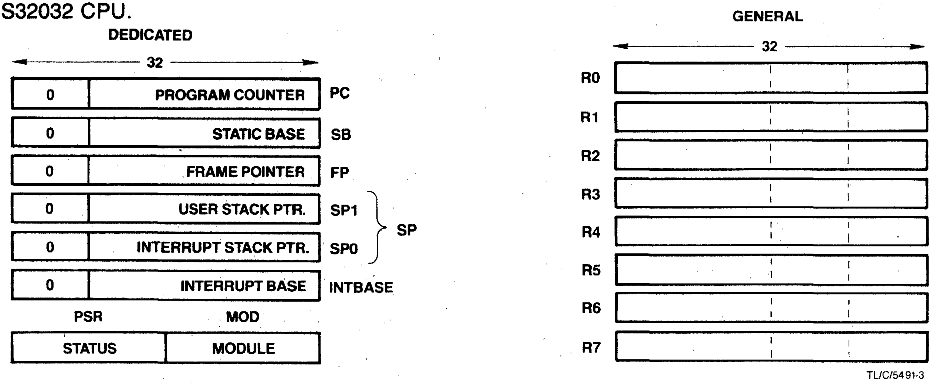


FIGURE 2-1. The General and Dedicated Registers.

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 NS32032 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 NS32032 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 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 NS32032 the upper eight bits of these registers are always zero).

Stacks in the NS16000 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 NS32032 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 NS32032 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 (Sec. 3.8). The INTBASE register holds the lowest address in memory occupied by the dispatch table. (In the NS32032 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 NS32032 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.

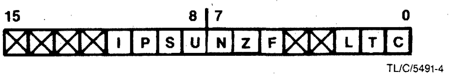


FIGURE 2-2. Processor Status Register.

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

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

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

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

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

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

U: If the U bit is "1" no privileged instructions may be executed. If the U bit is "0" then all instructions may be executed. When U = 0 the NS32032 is said to be in Supervisor Mode; when U = 1 the NS32032 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 (Sec. 3.8.5). It may have a setting of 0 (no trace pending) or 1 (trace pending).

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

2.1.3 The Configuration Register (CFG)

Within the Control section of the NS32032 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.

C	M	F	I

TL/C/5491-5

FIGURE 2-3. CFG Register.

The CFG I bit declares the presence of external interrupt vectoring circuitry (specifically, the NS32032 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 Sec. 3.8.

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

2.1.4 Memory Organization

The main memory of the NS32032 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.

7 0
A

Byte at Address A

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

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

Word at Address A

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

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

Double Word at Address A

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

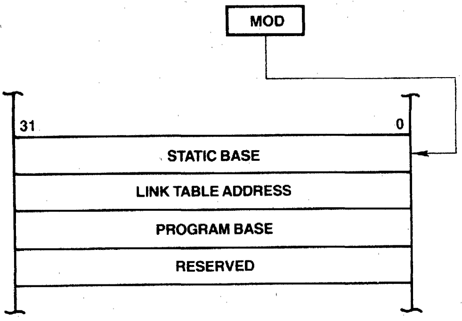
2.1.5 Dedicated Tables

Two of the NS32032 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 Sec. 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 NS32032. At any point in time, 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/C/5491-6

FIGURE 2-4. Module Descriptor Format.

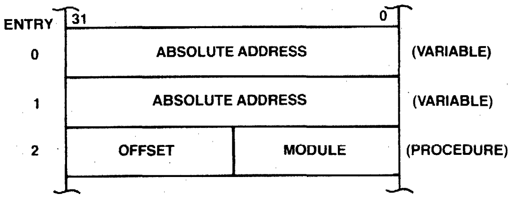
The Link Table Address points to the Link Table for the currently running module. The Link Table provides the information needed for:

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 NS16000 Programmer's Manual.



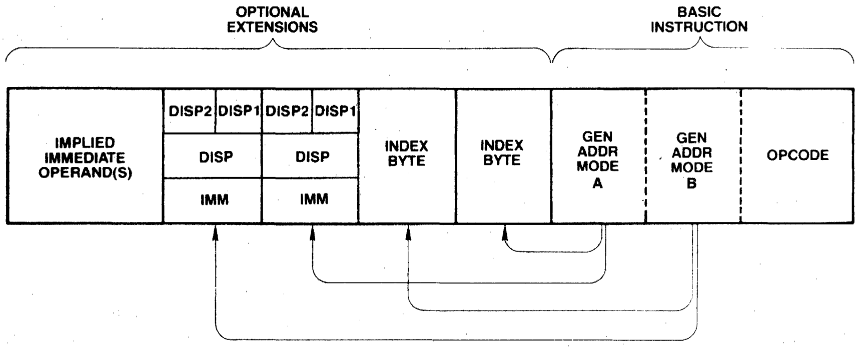
TL/C/5491-7

FIGURE 2-5. A Sample Link Table.

2.2 INSTRUCTION SET

2.2.1 General Instruction Format

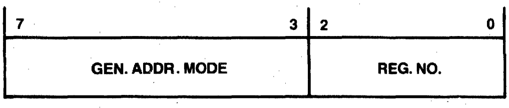
Figure 2-6 shows the general format of an NS16000 instruction. The Basic Instruction is one to three bytes long and contains the Opcode and up to two 5-bit General Addressing Mode ("Gen") fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.



TL/C/5491-8

FIGURE 2-6. General Instruction Format.

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.



TL/C/5491-9

FIGURE 2-7. Index Byte Format.

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain one or 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 backward from the usual memory representation of data (Sec. 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 (Sec. 2.2.3).

2.2.2 Addressing Modes

The NS32032 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."

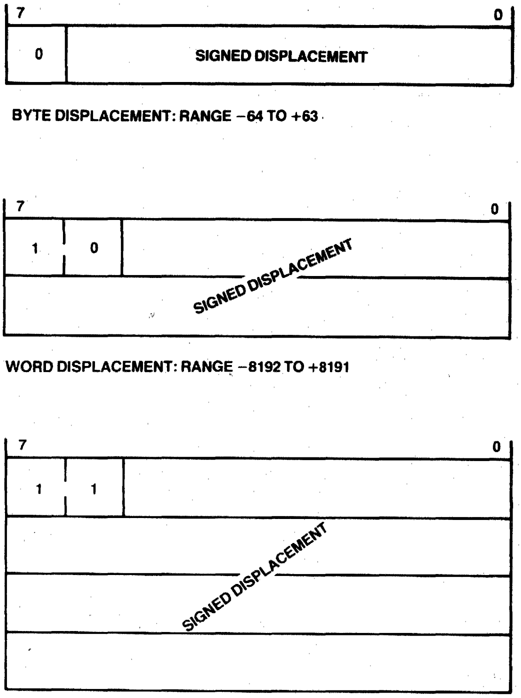
Addressing modes in the NS32032 are designed to optimally support high-level language accesses to variables. In nearly all cases, a variable access requires only one addressing mode, within the instruction that acts upon that variable. Extraneous data movement is therefore minimized.

NS32032 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.



BYTE DISPLACEMENT: RANGE −64 TO +63

WORD DISPLACEMENT: RANGE −8192 TO +8191

DOUBLE WORD DISPLACEMENT: RANGE (ENTIRE ADDRESSING SPACE)

TL/C/5491-10

FIGURE 2-8. Displacement Encodings.

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

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

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

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

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

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

Table 2-1 is a brief summary of the addressing modes. For a complete description of their actions, see the Programmer's Manual.

TABLE 2-1.
NS32032 Addressing Modes

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

2.2.3 Instruction Set Summary

Table 2-2 presents a brief description of the NS32032 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 Programmer's 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.
NS32032 Instruction Set Summary

MOVES

Format	Operation	Operands	Description
4	MOVi	gen,gen	Move a value.
2	MOVQi	short,gen	Extend and move a 4-bit constant.
7	MOVMi	gen,gen,disp	Move Multiple: disp bytes.
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 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)

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

INTEGER COMPARISON

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

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

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

ARRAYS

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

STRINGS

String instructions assign specific functions to the General Purpose Registers:

R4 – Comparison Value
R3 – Translation Table Pointer
R2 – String 2 Pointer
R1 – String 1 Pointer
R0 – Limit Count

Options on all string instructions are:

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

All string instructions end when R0 decrements to zero.

Format	Operation	Operands	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)

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	CCMPc	gen,gen	Custom Compare.
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 Functional Description

3.1 POWER AND GROUNDING

The NS32032 requires a single 5-volt power supply, applied on pin 9 (V_{CC}). See DC Specification Section.

Grounding connections are made on three pins. Logic Ground (GNDL, pin 44) is the common pin for on-chip logic, and Buffer Grounds (GNDB1, pin 43 and GNDB2, pin 11) are the common pins for the output drivers. For optimal noise immunity it is recommended that GNDB1 and GNDB2 be connected together through a single conductor, and GNDL be directly connected to the middle point of this conductor. All other ground connections should be made to the common line as shown in Figure 3-1.

In addition to V_{CC} and Ground, the NS32032 CPU uses an internally-generated negative voltage. It is necessary to filter this voltage externally by attaching a pair of capacitors (Fig. 3-1) from the BBG pin to ground. Recommended values for these are:

C_1: 1 μF, Tantalum.

C_2: 1000 pF, low inductance. This should be either a disc or monolithic ceramic capacitor.

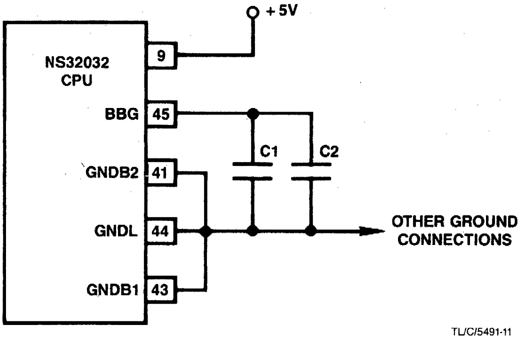


FIGURE 3-1. Recommended Supply Connections.

3.2 CLOCKING

The NS32032 inputs clocking signals from the NS16201 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 positive 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 the AC Specifications (Sec. 4) for complete specifications of PHI1 and PHI2.

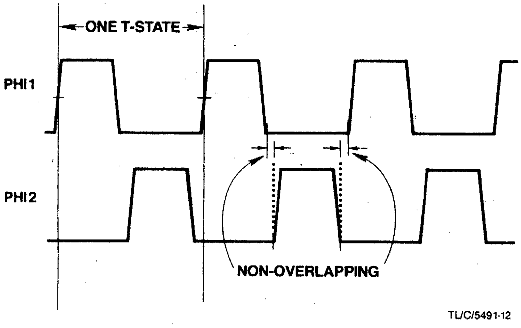


FIGURE 3-2. Clock Timing Relationships.

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

3.3 RESETTING

The $\overline{\text{RST}}/\overline{\text{ABT}}$ pin serves both as a Reset for on-chip logic and as the Abort input for Memory-Managed systems. For its use as the Abort Command, see Sec. 3.5.4.

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 μsec 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

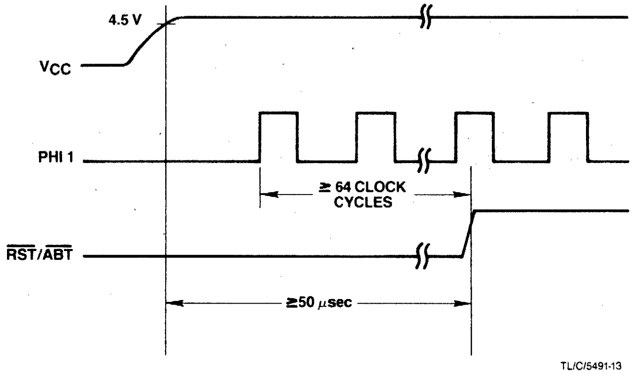
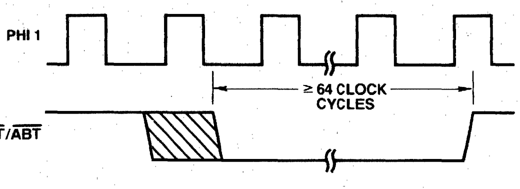


FIGURE 3-3. Power-on Reset Requirements.

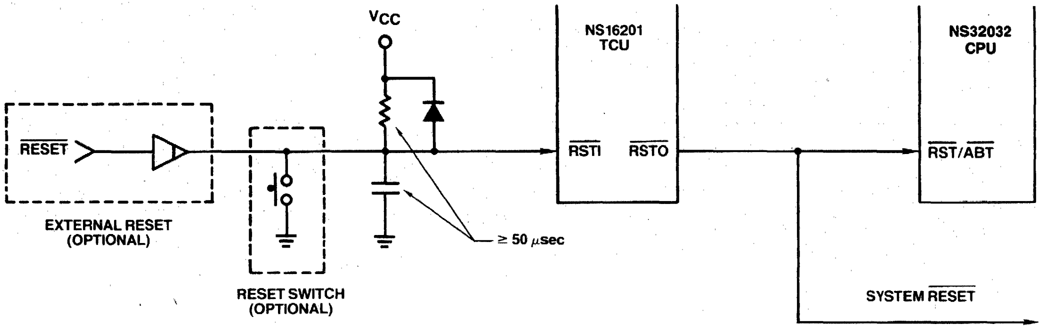
active for not less than 64 clock cycles. The trailing (positive-going) edge must occur while PHI1 is high, and no later than 10 ns before the PHI1 trailing edge. See Figures 3-3 and 3-4.

The NS16201 Timing Control Unit (TCU) provides circuitry to meet the Reset requirements of the NS32032 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.



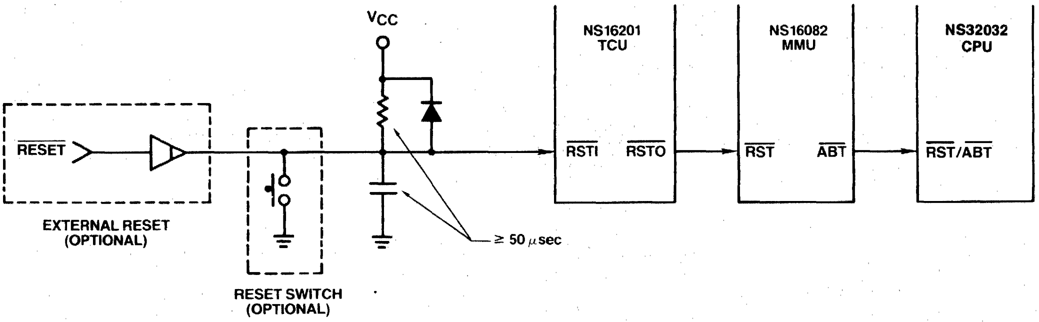
TL/C/5491-14

FIGURE 3-4. General Reset Timing.



TL/C/5491-15

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



TL/C/5491-16

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

3.4 BUS CYCLES

The NS32032 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 Sec. 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 NS16000 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 Sec. 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 (Sec. 3.4.6).

The sequence of events in a non-Slave bus cycle is shown below 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 (Sec. 3.4.1).

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").

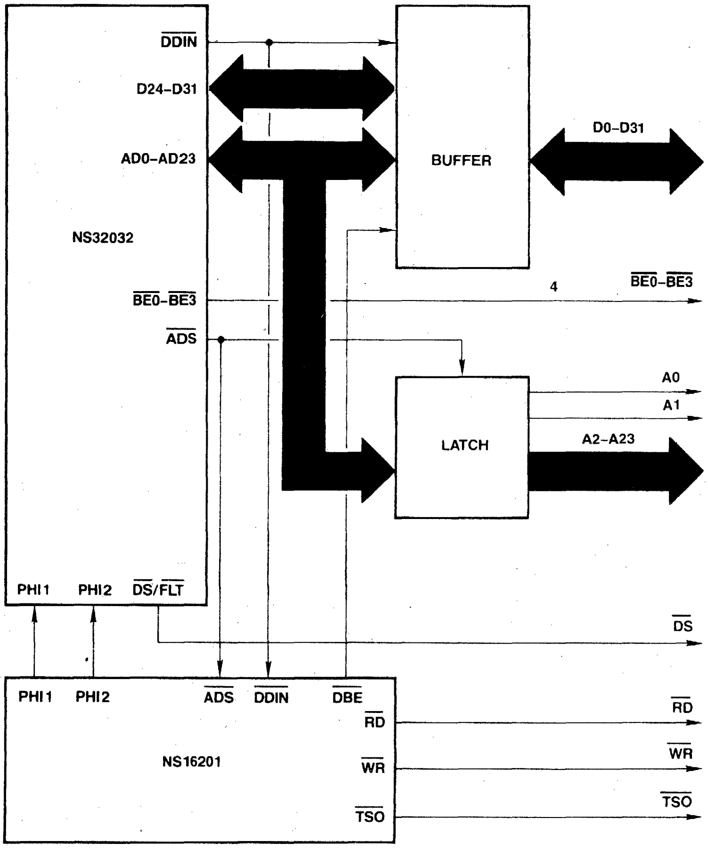
During T1, the CPU applies an address on pins AD0–AD23. It also provides a low-going pulse on the ADS pin, which serves the dual purpose of informing external circuitry that a bus cycle is starting and of providing control to an external latch for demultiplexing Address bits 0–23 from the AD0–AD23 pins. See Figure 3-6. During this time also the status signals $\overline{\text{DDIN}}$, indicating the direction of the transfer, and $\overline{\text{BE0}}$–$\overline{\text{BE3}}$, indicating which of the four bus bytes are to be referenced, become valid.

During T2 the CPU switches the Data Bus, AD0–AD31 to either accept or present data. It also starts the data strobe ($\overline{\text{DS}}$), signalling the beginning of the data transfer. Associated signals from the NS16201 Timing Control Unit are also activated at this time: $\overline{\text{RD}}$ (Read Strobe) or $\overline{\text{WR}}$ (Write Strobe), $\overline{\text{TSO}}$ (Timing State Output, indicating that T2 has been reached) and DBE (Data Buffer Enable).

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

If the CPU is performing a Read cycle, the Data Bus (AD0-AD31) is sampled at the falling edge of PHI2 of the last T3 state. See Timing Specification, Sec. 4. Data must, however, be held at least until the beginning of T4. $\overline{\text{DS}}$ and $\overline{\text{RD}}$ 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 $\overline{\text{DS}}$, $\overline{\text{RD}}$ or $\overline{\text{WR}}$, and $\overline{\text{TSO}}$ signals go inactive, and at the rising edge of PHI2, $\overline{\text{DBE}}$ goes inactive, having provided for necessary data hold times. Addresses (and Data during Write cycles) remain 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/C/5491-17

FIGURE 3-6. Bus Connections.

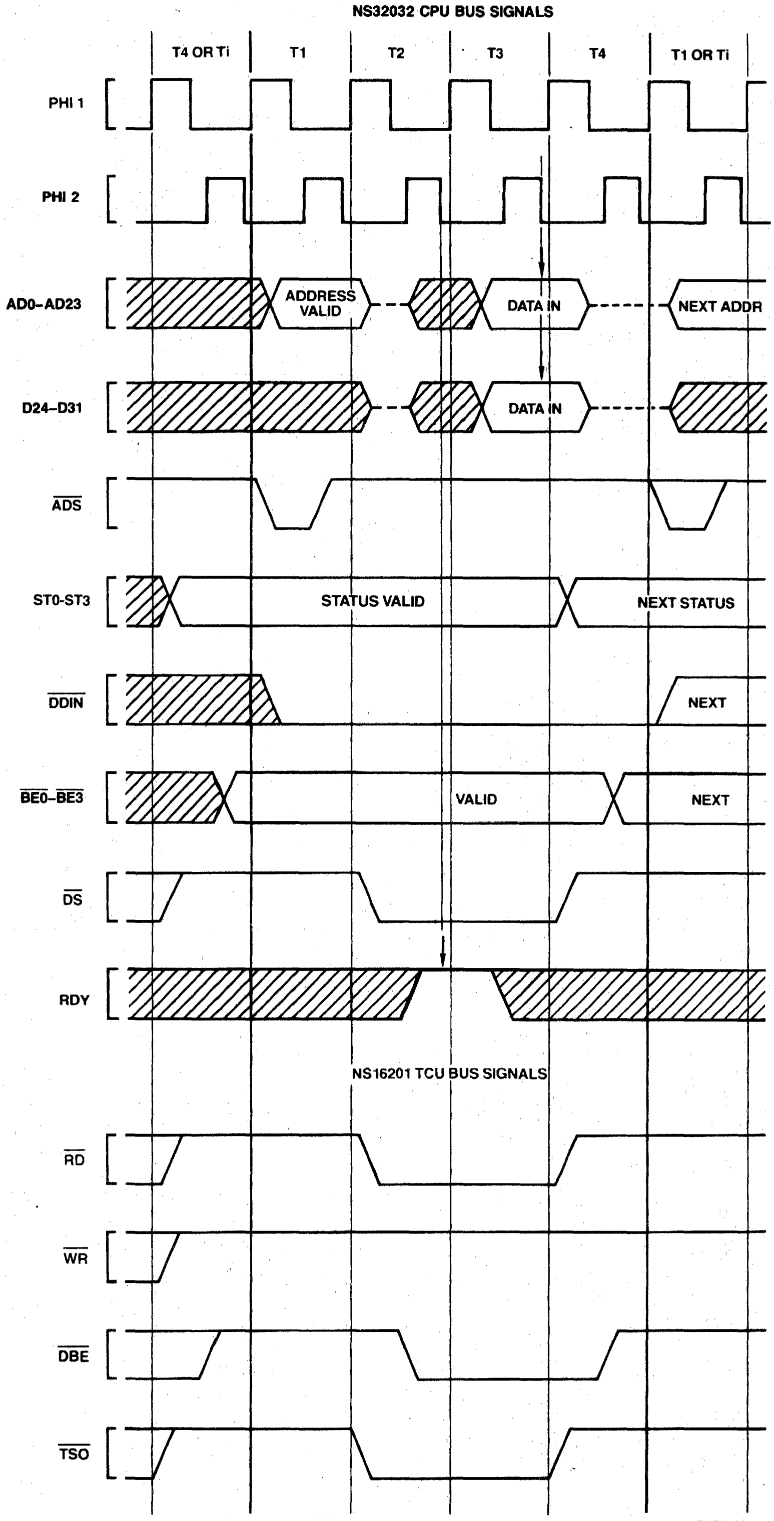


FIGURE 3-7. Read Cycle Timing.

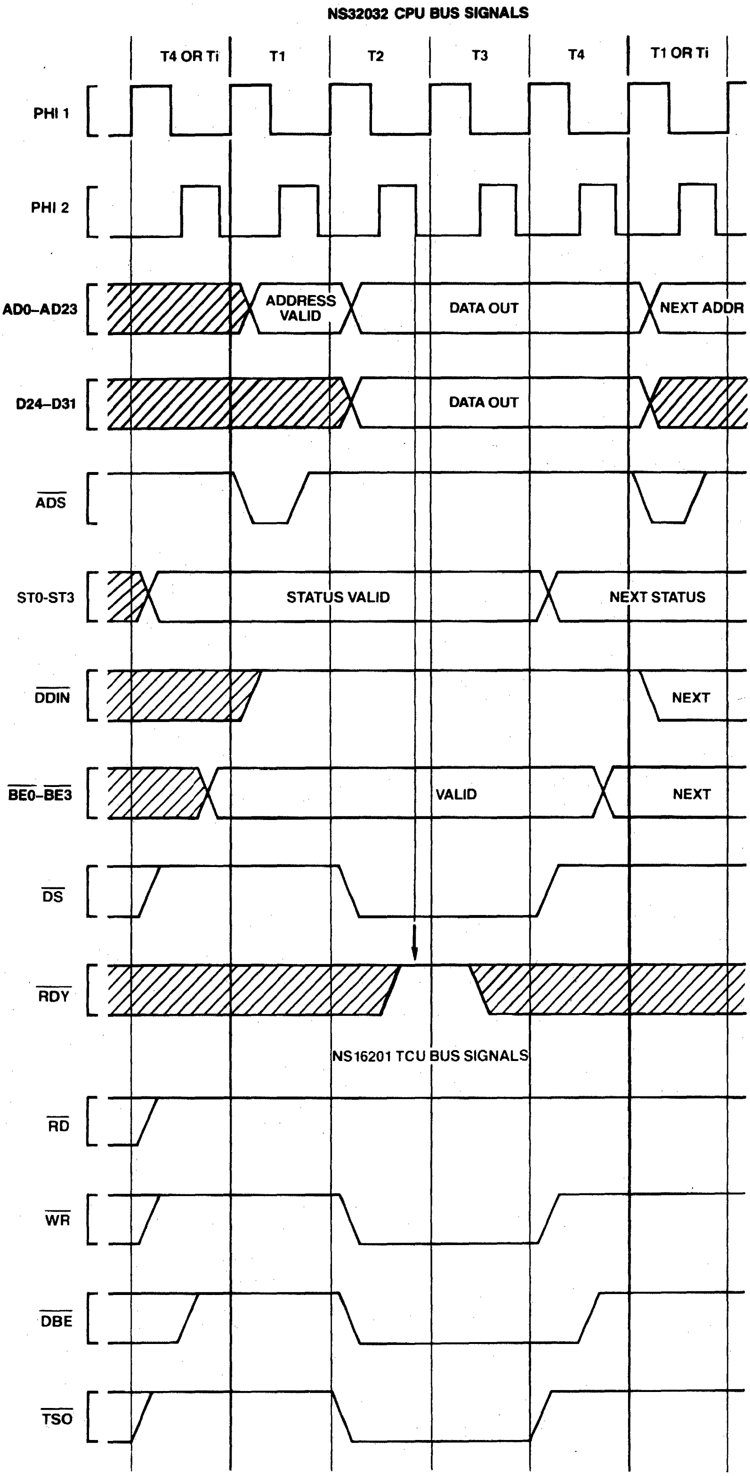


FIGURE 3-8. Write Cycle Timing.

3.4.1 Cycle Extension

To allow sufficient strobe widths and access times for any speed of memory or peripheral device, the NS16032 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 RDY is 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 NS16201 Timing Control Unit, which applies WAIT States to the CPU as requested on three sets of pins:

1) $\overline{\text{CWAIT}}$ (Continuous WAIT), which holds the CPU in WAIT states until removed.

2) $\overline{\text{WAIT1}}$, $\overline{\text{WAIT2}}$, $\overline{\text{WAIT4}}$, $\overline{\text{WAIT8}}$ (Collectively $\overline{\text{WAITn}}$), 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 on their use, see the NS16201 Data Sheet.

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

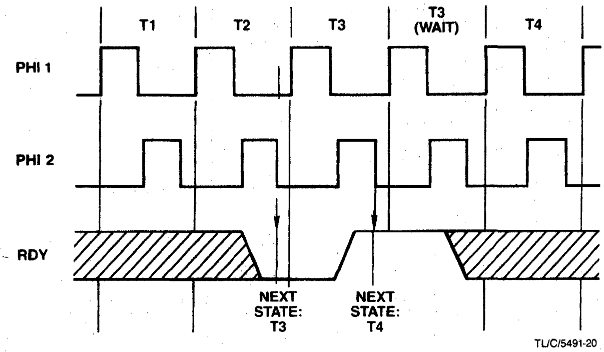


FIGURE 3-9. RDY Pin Timing.

3.4.2 Bus Status

The NS32032 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 yet need access to the bus.

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.

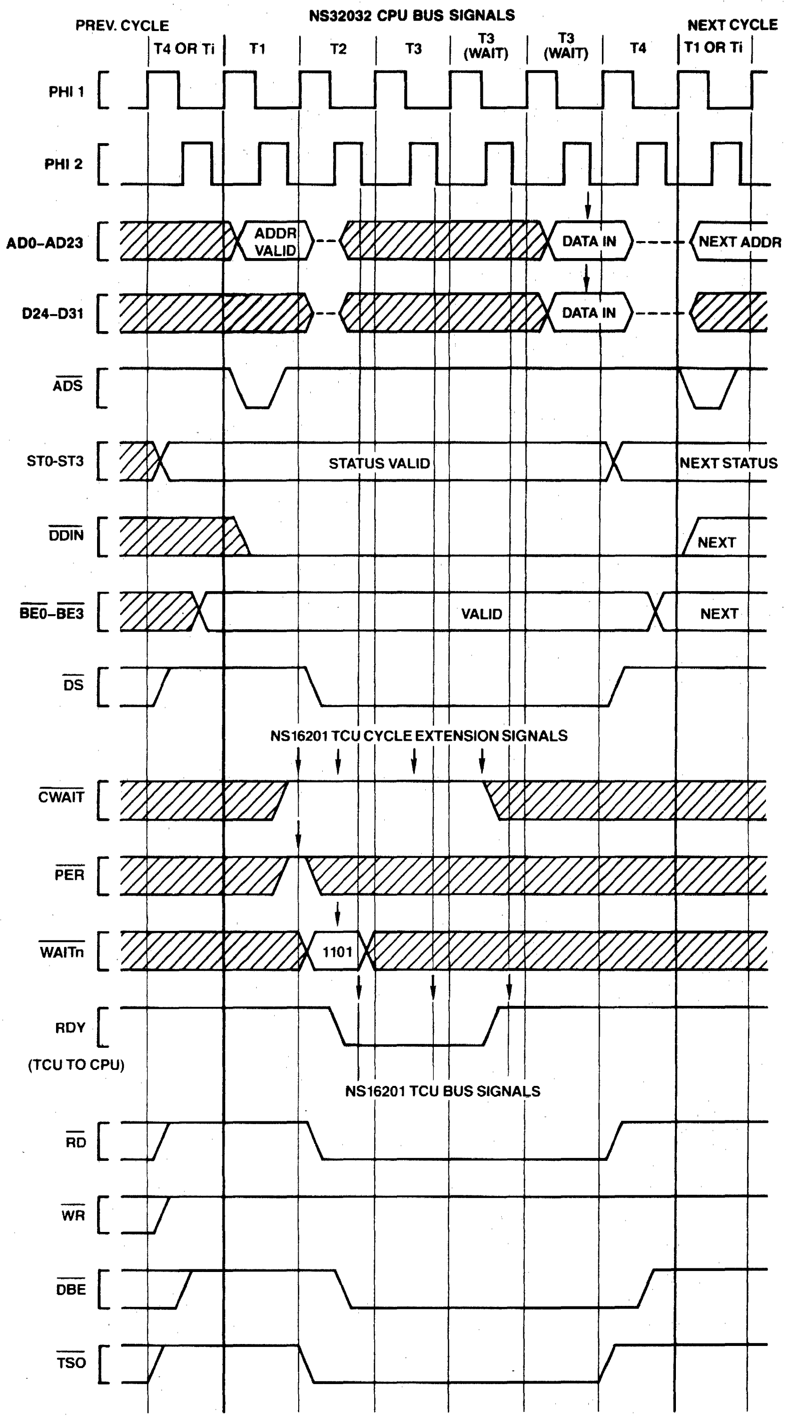
0100 – Interrupt Acknowledge, Master.
The CPU is performing a Read cycle. To acknowledge receipt of a Non-Maskable Interrupt (on $\overline{\text{NMI}}$) it will read from address FFFF00_{16}, but will ignore any data provided.
To acknowledge receipt of a Maskable Interrupt (on $\overline{\text{INT}}$) it will read from address FFFE00_{16}, expecting a vector number to be provided from the Master NS16202 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 NS16202 is present. See Sec. 3.4.5.

0101 – Interrupt Acknowledge, Cascaded.
The CPU is reading a vector number from a Cascaded NS16202 Interrupt Control Unit. The address provided is the address of the NS16202 Hardware Vector register. See Sec. 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 Sec. 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 Sec. 3.4.5.

1000 – Sequential Instruction Fetch.
The CPU is reading the next sequential word from the instruction stream into the Instruction



TL/C/5491-21

FIGURE 3-10. Extended Cycle Example.

NOTE:

Arrows on $\overline{\text{CWAIT}}$, $\overline{\text{PER}}$, $\overline{\text{WAIT}}$n indicate points at which the TCU samples. Arrows on AD0-AD15 and RDY indicate points at which the CPU samples.

Queue. It will do so whenever the bus would otherwise be idle and the queue is not already full.

1001 – Non-Sequential Instruction Fetch.
The CPU is performing the first fetch of instruction code after the Instruction Queue is purged. This will occur as a result of any jump or branch, or any interrupt or trap, or execution of certain instructions.

1010 – Data Transfer.
The CPU is reading or writing an operand of an instruction.

1011 – Read RMW Operand.
The CPU is reading an operand which will subsequently be modified and rewritten. If memory protection circuitry would not allow the following Write cycle, it must abort this cycle.

1100 – Read for Effective Address Calculation.
The CPU is reading information from memory in order to determine the Effective Address of an operand. This will occur whenever an instruction uses the Memory Relative or External addressing mode.

1101 – Transfer Slave Processor Operand.
The CPU is either transferring an instruction operand to or from a Slave Processor, or it is issuing the Operation Word of a Slave Processor instruction. See Sec. 3.9.1.

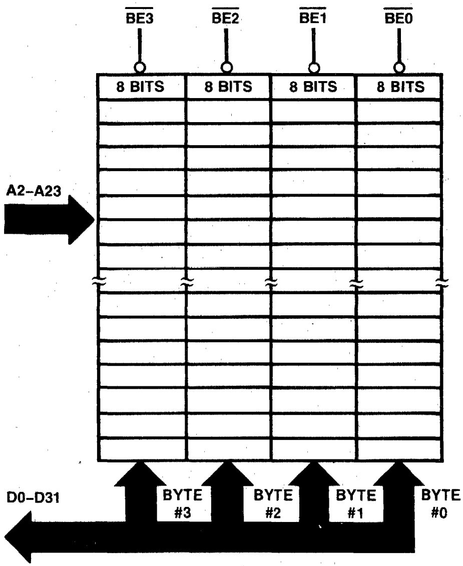
1110 – Read Slave Processor Status.
The CPU is reading a Status Word from a Slave Processor. This occurs after the Slave Processor has signalled completion of an instruction. The transferred word tells the CPU whether a trap should be taken, and in some instructions it presents new values for the CPU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

1111 – Broadcast Slave ID.
The CPU is initiating the execution of a Slave Processor instruction. The ID Byte (first byte of the instruction) is sent to all Slave Processors, one of which will recognize it. From this point the CPU is communicating with only one Slave Processor. See Sec. 3.9.1.

3.4.3 Data Access Sequences

The 24-bit address provided by the NS32032 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 NS32032 is that the presence of a 32-bit data bus imposes no restrictions on data alignment; any data item, regardless of size, may be placed starting at any memory address. The NS32032 provides special control signals, Byte Enable ($\overline{BE0}$–$\overline{BE3}$) which facilitate individual byte accessing on a 32-bit bus.

Memory is organized as four eight-bit banks, each bank receiving the double-word address (A2–A23) in parallel. One bank, connected to Data Bus pins AD0–AD7 is enabled when $\overline{BE0}$ is low. The second bank, connected to data bus pins AD8–AD15 is enabled when $\overline{BE1}$ is low. The third and fourth banks are enabled by $\overline{BE2}$ and $\overline{BE3}$, respectively. See Figure 3-11.



TL/C/5491-22

FIGURE 3-11. Memory Interface.

There are 12 combinations of operand lengths and address bits A1, A0, which imply 10 different types of bus accesses. Table 3-1 lists the bus access types, the least significant address bits, and the byte enable levels.

TABLE 3-1.
Bus Access Types

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

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

TABLE 3-2.
Access Sequences

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

X = Don't Care

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 operands which 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 NS32032 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 (Sec. 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 type 8 Read cycles (Table 3-1).

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

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 NS32032 interrupt structure, see Sec. 3.8.

TABLE 3-3.
Interrupt Sequences

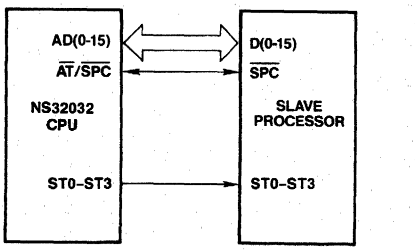
								Data Bus			
Cycle	**Status**	**Address**	**$\overline{\text{DDIN}}$**	**$\overline{\text{BE3}}$**	**$\overline{\text{BE2}}$**	**$\overline{\text{BE1}}$**	**$\overline{\text{BE0}}$**	**Byte 3**	**Byte 2**	**Byte 1**	**Byte 0**
				A. Non-Maskable Interrupt Control Sequences							
Interrupt Acknowledge											
1	0100	$FFFF00_{16}$	0	1	1	1	0	X	X	X	X
Interrupt Return											
None: Performed through Return from Trap (RETT) instruction.											
				B. Non-Vectored Interrupt Control Sequences							
Interrupt Acknowledge											
1	0100	$FFFE00_{16}$	0	1	1	1	0	X	X	X	X
Interrupt Return											
1	0110	$FFFE00_{16}$	0	1	1	1	0	X	X	X	X
				C. Vectored Interrupt Sequences: Non-Cascaded.							
Interrupt Acknowledge											
1	0100	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Vector: Range: 0–127
Interrupt Return											
1	0110	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Vector: Same as in Previous Int. Ack. Cycle
				D. Vectored Interrupt Sequences: Cascaded							
Interrupt Acknowledge											
1	0100	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Cascade Index: range − 16 to − 1
(The CPU here uses the Cascade Index to find the Cascade Address.)											
2	0101	Cascade Address	0		See Note			Vector, range 0–255; on appropriate byte of data bus.			
Interrupt Return											
1	0110	$FFFE00_{16}$	0	1	1	1	0	X	X	X	Cascade Index: Same as in previous Int. Ack. Cycle
(The CPU here uses the Cascade Index to find the Cascade Address.)											
2	0111	Cascade Address	0		See Note			X	X	X	X

X = Don't Care

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

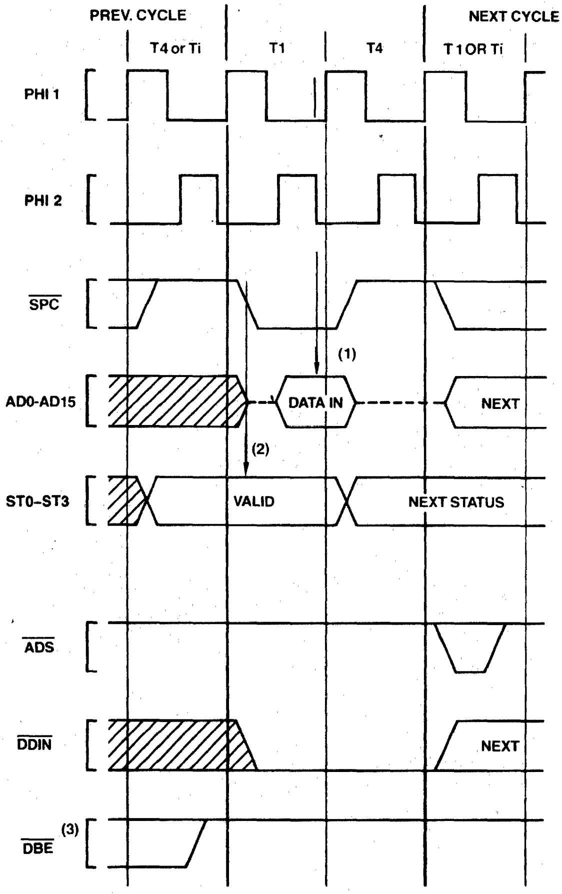
3.4.6 Slave Processor Communication

In addition to its use as the Address Translation strap (Sec. 3.5.1), the AT/SPC pin is used as the data strobe for Slave Processor transfers. In this role, it is referred to as Slave Processor Control (SPC). In a Slave Processor bus cycle, data is transferred on the Data Bus (AD0-AD15), and the least significant two bits of CPU cycle status (ST0-ST1) are monitored by each Slave Processor in order to determine the type of transfer being performed. SPC is bidirectional, but is driven by the CPU during all Slave Processor bus cycles. See Sec. 3.9 for full protocol sequences.



TL/C/5491-23

FIGURE 3-12. Slave Processor Connections.



TL/C/5491-24

NOTE:

(1) CPU samples Data Bus here.

(2) Slave Processor samples CPU Status here.

(3) DBE and all other NS16201 TCU bus signals remain inactive because no ADS pulse is received from the CPU.

FIGURE 3-13. CPU Read from Slave Processor.

3.4.6.1 Slave Processor Bus Cycles

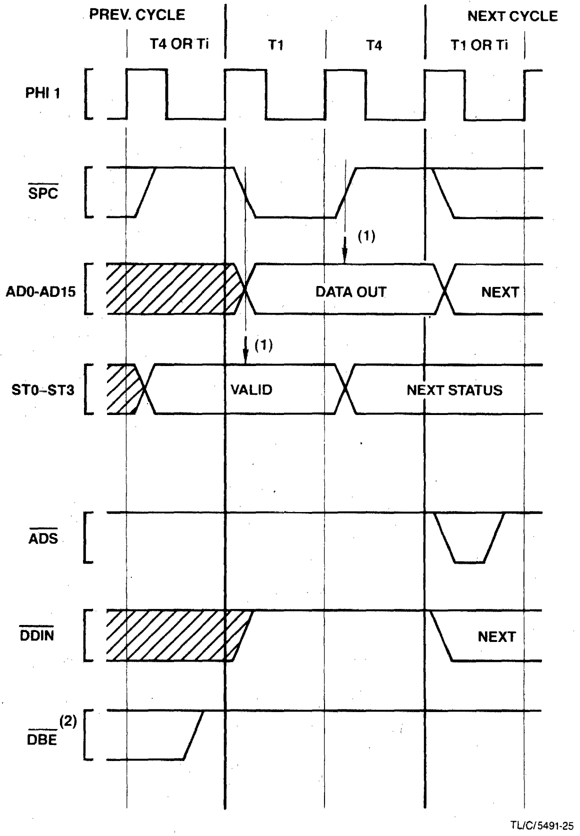
A Slave Processor bus cycle always takes exactly two clock cycles, labelled T1 and T4 (see Figures 3-13 and 3-14). During a Read cycle, $\overline{SPC}$ is activated at T1, data is sampled at T4, and $\overline{SPC}$ is removed. The Cycle Status pins lead the cycle by one clock period, and are sampled 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 NS16201 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 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 bits AD0–AD15. 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.

Note that the NS32032 uses only the two least significant bytes of the data bus for slave cycles. This is to maintain compatibility with existing slave processors.



NOTE:

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

(2) $\overline{DBE}$, being provided by the NS16201 TCU, remains inactive due to the fact that no pulse is presented on $\overline{ADS}$. TCU signals $\overline{RD}$, $\overline{WR}$ and $\overline{TSO}$ also remain inactive.

FIGURE 3-14. CPU Write to Slave Processor.

3.5 MEMORY MANAGEMENT OPTION

The NS32032 CPU, in conjunction with the NS16082 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 NS32032 CPU has two bus timing modes: With or Without Address Translation. The mode of operation is selected by the CPU by sampling the AT/SPC (Address Translation/Slave Processor Control) pin on the rising edge of the RST (Reset) pulse. If AT/SPC is sampled as high, the bus timing is as previously described in Sec. 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 DS/FLT pin changes in function from a Data Strobe output (DS) to a Float Command input (FLT).

The NS16082 MMU will itself pull the CPU AT/SPC pin low when it is reset, but this pin may be left floating in non-Memory-Managed systems.

Note that the Address Translation strap does not specifically declare the presence of an NS16082 MMU, but

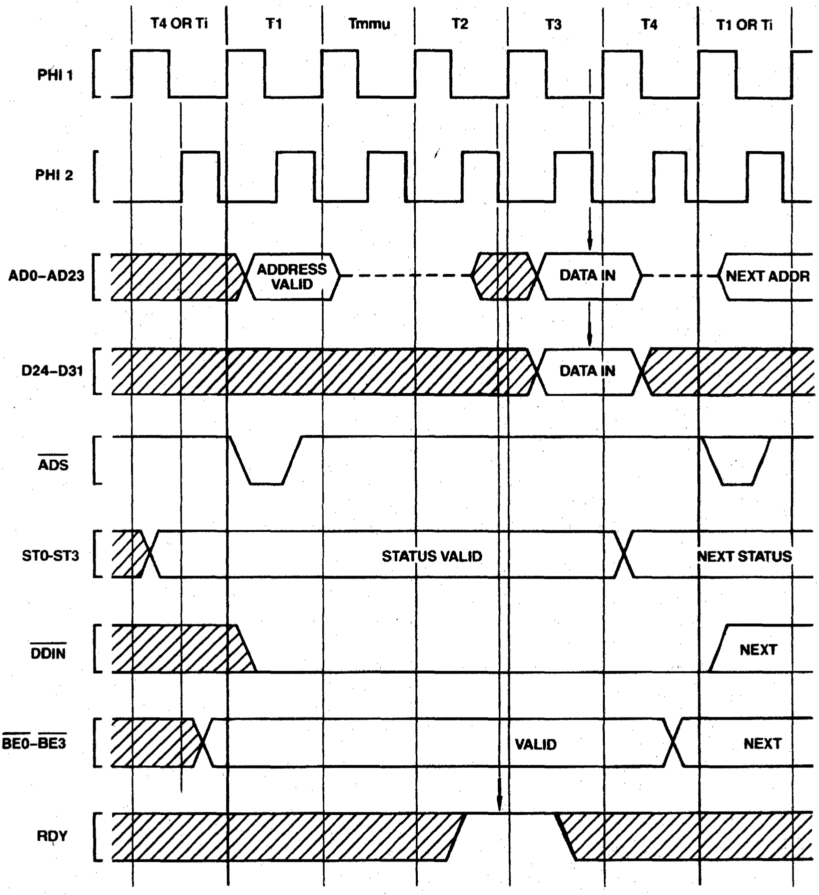


FIGURE 3-15. Read Cycle with Address Translation (CPU Action).

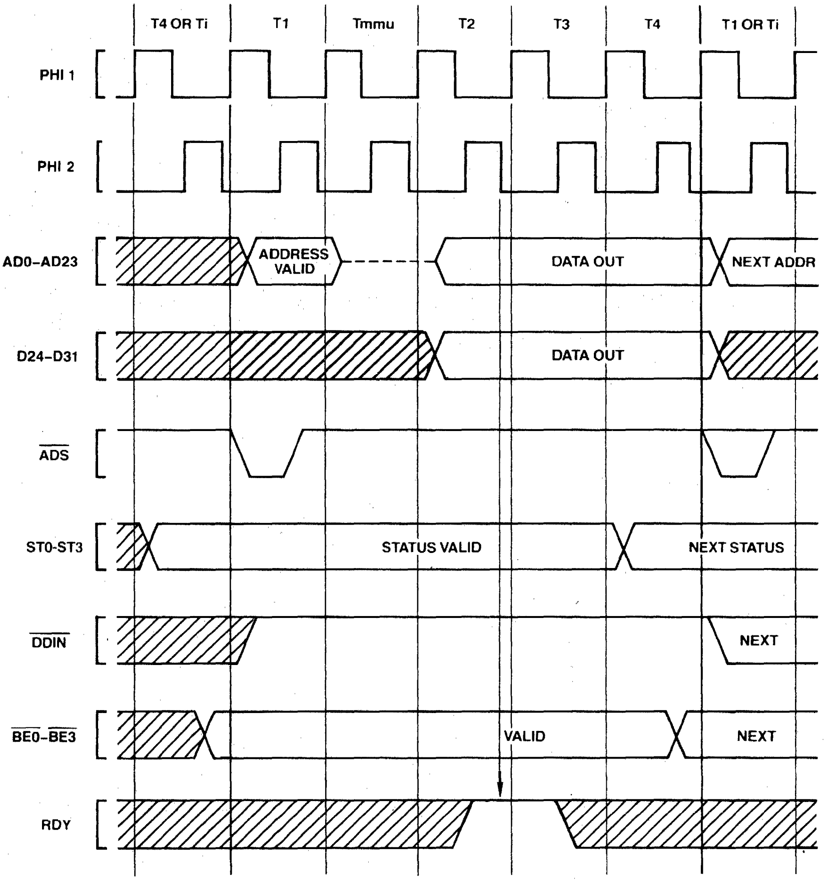
only the presence of external address translation circuitry. MMU instructions will still trap as being undefined unless the SETCFG (Set Configuration) instruction is executed to declare the MMU instruction set valid. See Sec. 2.1.3.

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–AD23 into the TRI-STATE® mode, allowing the MMU to assert the translated address and issue the physical address strobe $\overline{\text{PAV}}$. T2 through T4 of the cycle are identical to their counterparts without Address Translation. Note that in order for the NS16082 MMU to operate correctly it must be set to the 32032 mode by strapping A24 to ground during reset.

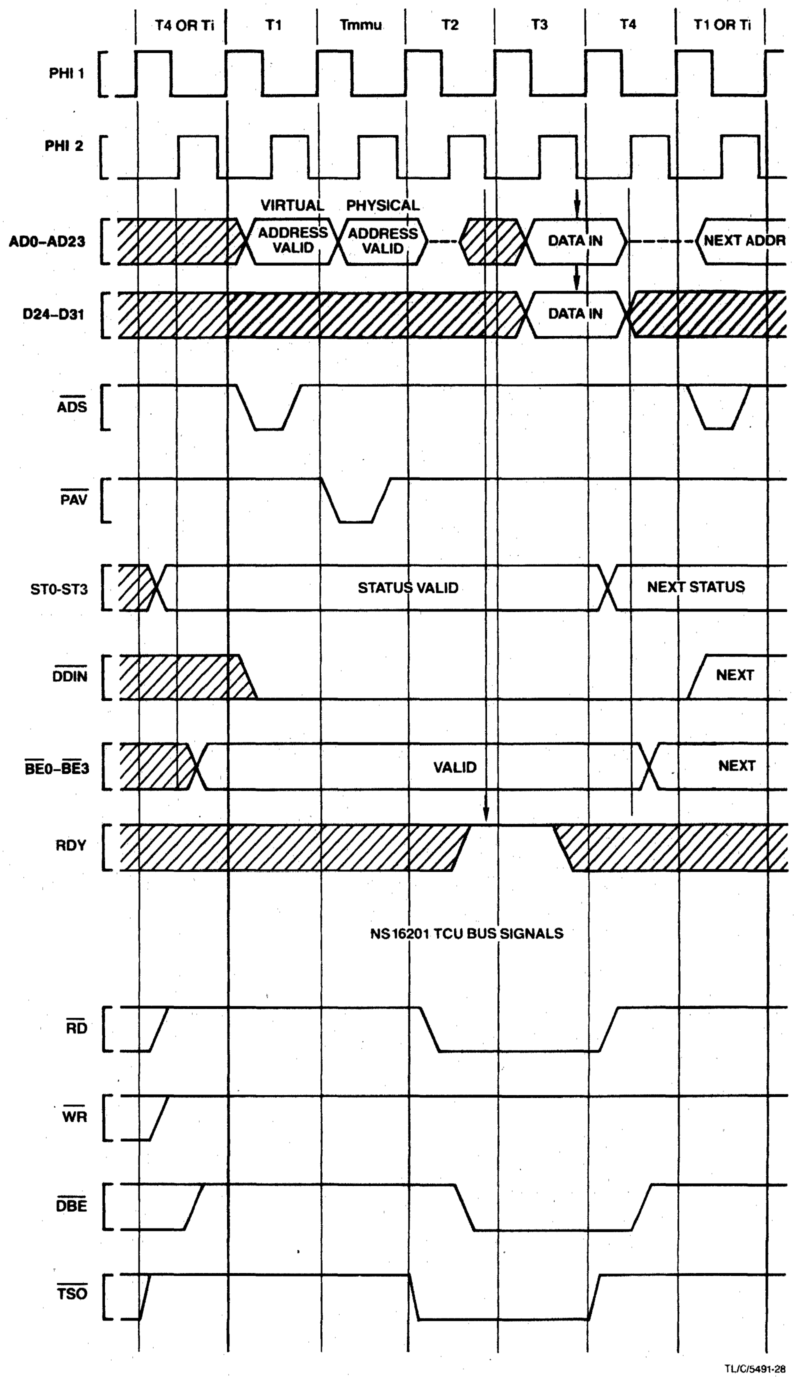
In this mode the bus lines AD16–AD23 are floated after the MMU address has been latched since they are used by the CPU to transfer data.

Figures 3-17 and 3-18 show a Read cycle and a Write cycle as generated by the 32032/16082/16201 group. Note that with the CPU $\overline{\text{ADS}}$ signal going only to the MMU, and with the MMU $\overline{\text{PAV}}$ signal substituting for $\overline{\text{ADS}}$ everywhere else, Tmmu through T4 look exactly like T1 through T4 in a non-Memory-Managed system. For the connection diagram, see Appendix B.



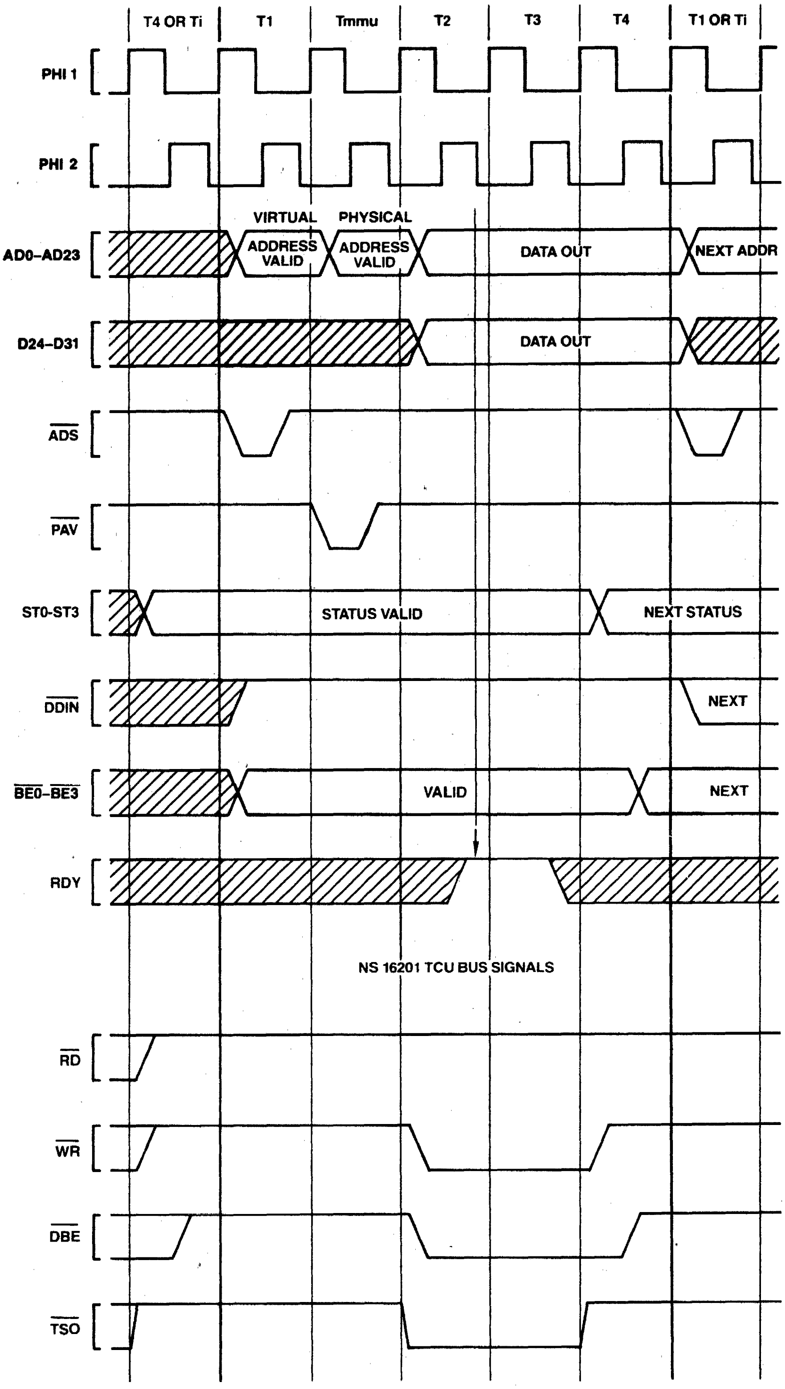
TL/C/5491-27

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



TL/C/5491-28

FIGURE 3-17. Memory-Managed Read Cycle.



TL/C/5491-29

FIGURE 3-18. Memory-Managed Write Cycle.

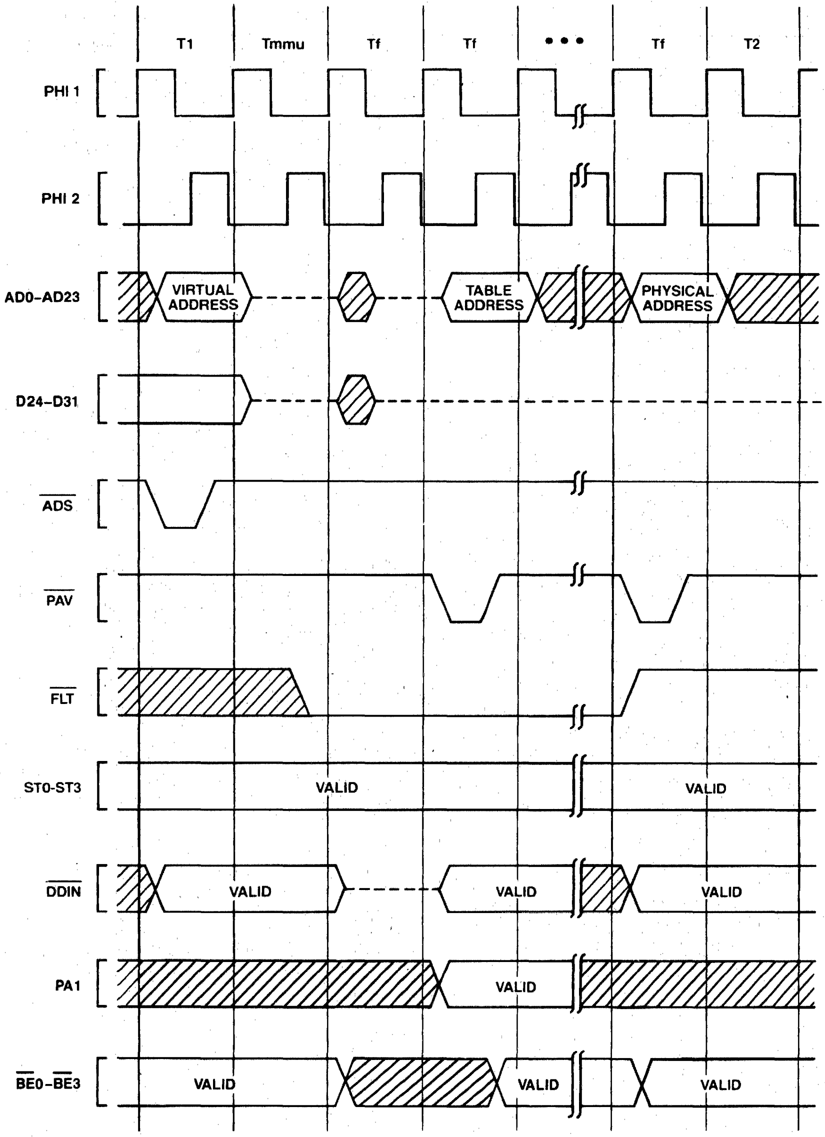
3.5.3 The $\overline{\text{FLT}}$ (Float) and PA1 (Physical A1) Pins

In Address Translation mode, the $\overline{\text{DS}}/\overline{\text{FLT}}$ pin is treated as the input command $\overline{\text{FLT}}$ (Float). Activating $\overline{\text{FLT}}$ during Tmmu causes the CPU to wait longer than Tmmu for address translation and validation. This feature is used occasionally by the NS16082 MMU in order to update its internal translation cache from page tables in memory, or to update certain status bits within them.

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

1) Sets AD0–AD23, D24–D31 and $\overline{\text{DDIN}}$ to the TRI-STATE condition ("floating").
2) Sets $\overline{\text{BE3}}$–$\overline{\text{BE0}}$ according to PA1.
3) Suspends further internal processing of the current instruction. This ensures that the current instruction remains abortable with retry. (See $\overline{\text{RST}}/\overline{\text{ABT}}$ description, Sec. 3.5.4.)

Note that the AD0–AD23 pins may be briefly asserted during the first idle T-State. The above conditions remain in effect until $\overline{\text{FLT}}$ again goes high. See the Timing Specifications, Sec. 4.



TL/C/5491-30

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

3.5.4 Aborting Bus Cycles

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

If RST/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 PAV signal which triggers a physical cycle, the rest of the system remains unaware that a cycle was even started.

The NS16082 MMU will abort a bus cycle for either of two reasons:

1) The CPU is attempting to access a virtual address which is not currently resident in physical memory. The referenced page must be brought into physical memory from mass storage to make it accessible to the CPU.

2) The CPU is attempting to perform an access which is not allowed due to the protection level assigned to that page.

When a bus cycle is aborted by the MMU, the instruction which caused it to occur is also aborted in such a manner that it is guaranteed re-executable later. Due to the NS16000 Family instruction set definition and its implementation in the NS32032 CPU, the only information which is changed irrecoverably by such partly-executed instructions is information which does not affect their 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 Sec. 3.8). The Return Address pushed on the Interrupt Stack is the address of the aborted instruction, such 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 which 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 NS16082 Memory Management Unit.

1) If FLT has not been applied to the CPU, the Abort pulse must occur during or before Tmmu. See the Timing Specifications, Figure 4-22.

2) If FLT has been applied to the CPU, the Abort pulse must be applied before the T-State in which FLT goes inactive. The CPU will not actually respond to the Abort command until FLT is removed. See Figure 4-23.

3) No bus cycle may be aborted which is the Write half of a Read-Modify-Write operand access. The CPU guarantees that this will never be necessary for Memory Management functions by applying a special RMW status (Status Code 1011) during the Read half of the access. When the CPU presents RMW status, that cycle must be aborted if it would be illegal to write to any of the accessed addresses.

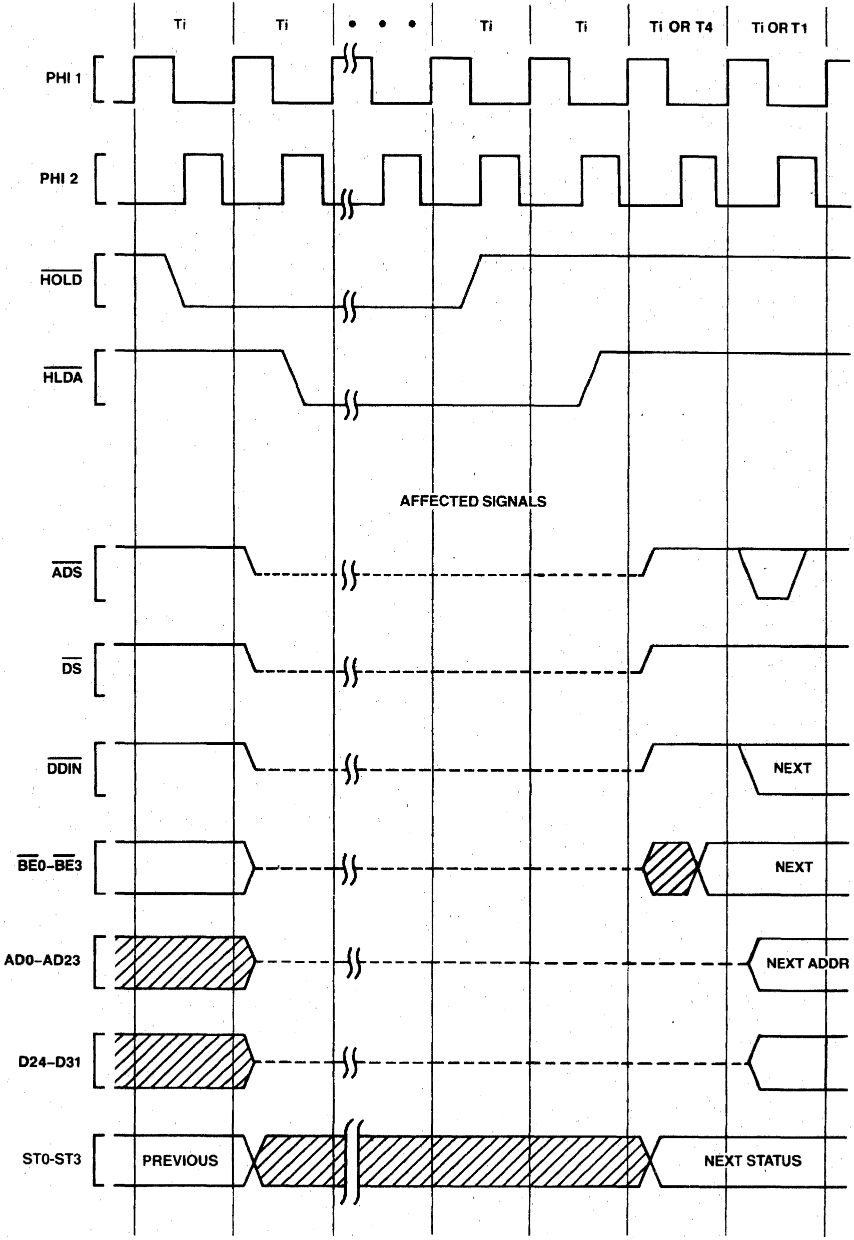
If RST/ABT is pulsed at any time other than as indicated above, it will abort either the instruction currently under execution or the next instruction and will act as a very high-priority interrupt. However, the program which was running at the time is not guaranteed recoverable, and should be terminated.

3.6 BUS ACCESS CONTROL

The NS32032 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 HOLD (Hold Request) and HLDA (Hold Acknowledge) pins. By asserting HOLD low, an external device requests access to the bus. On receipt of HLDA from the CPU, the device may perform bus cycles, as the CPU at this point has set the AD0–AD23, D24–D31, ADS, DDIN and BE0–BE3 pins to the TRI-STATE® condition. To return control of the bus to the CPU, the device sets HOLD inactive, and the CPU acknowledges return of the bus by setting HLDA inactive.

How quickly the CPU releases the bus depends on whether it is idle on the bus at the time the 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 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 HLDA signal is connected in a daisy-chain through the NS16082, such that the MMU can release the bus if it is using it.



TL/C/5491-31

FIGURE 3-20. $\overline{\text{HOLD}}$ Timing, Bus Initially Idle.

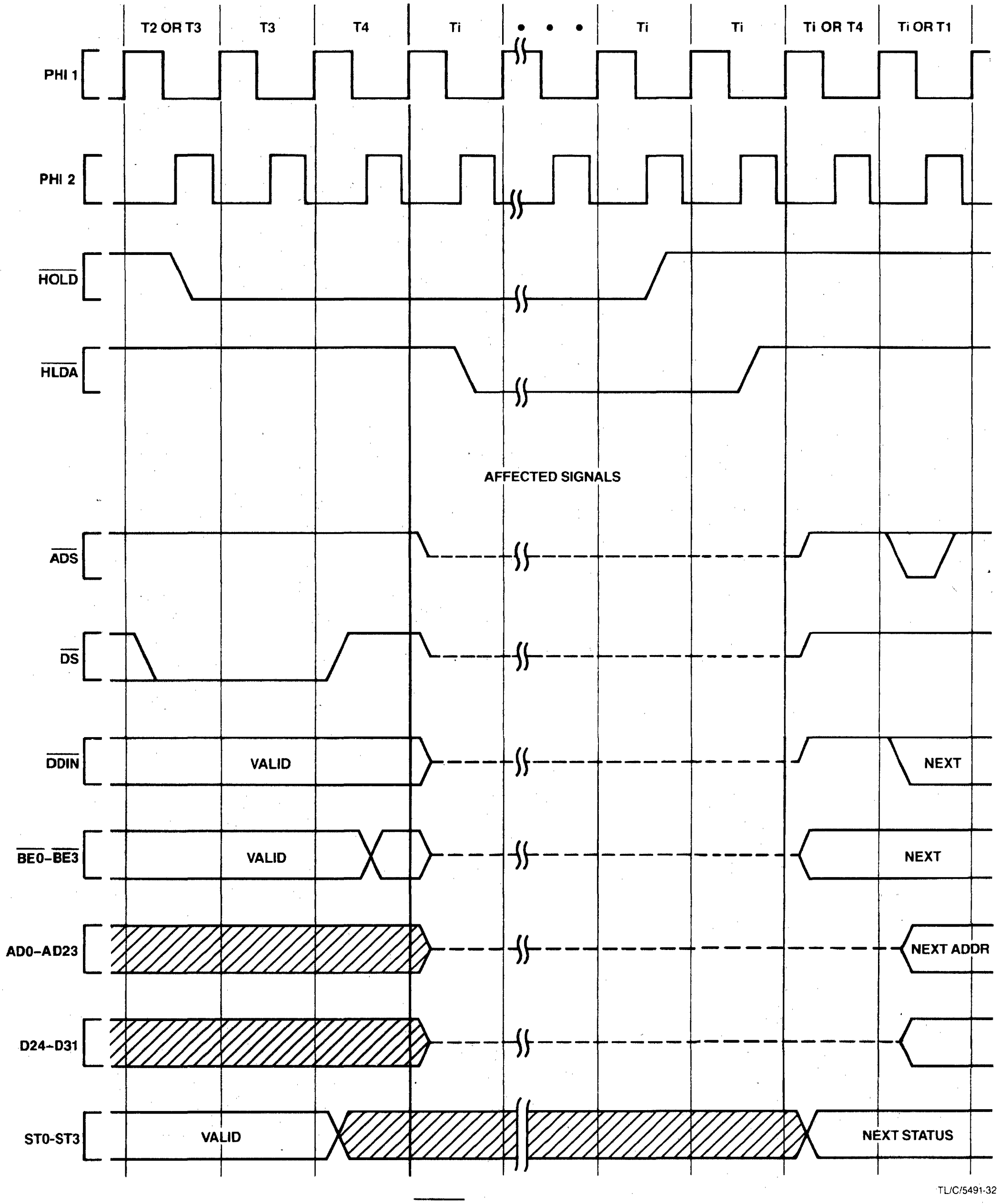


FIGURE 3-21. $\overline{\text{HOLD}}$ Timing, Bus Initially Not Idle.

3.7 INSTRUCTION STATUS

In addition to the four bits of Bus Cycle status (ST0-ST3), the NS32032 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.

$\overline{\mathrm{PFS}}$ (Program Flow Status) is pulsed low as each instruction begins execution. It is intended for debugging purposes, and is used that way by the NS16082 Memory Management Unit.

U/$\overline{\mathrm{S}}$ 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 4-21.

$\overline{\mathrm{ILO}}$ (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 multiprocessor communication and resource sharing. As with the U/$\overline{\mathrm{S}}$ pin, there are guarantees on its validity during the operand accesses performed by the instructions. See the Timing Specification Section, Figure 4-19.

3.8 NS32032 INTERRUPT STRUCTURE

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

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

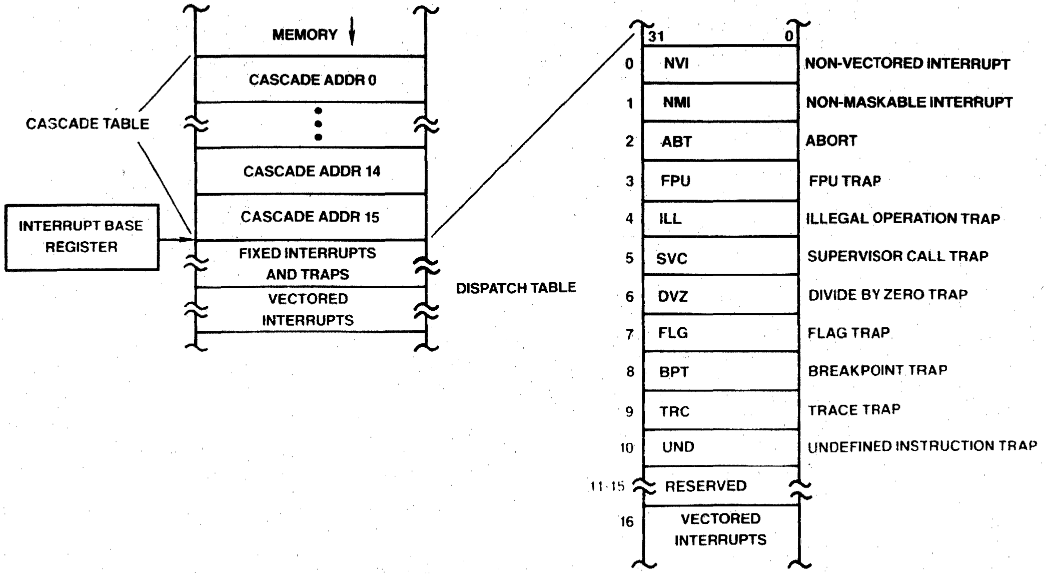
$\overline{\mathrm{RST/ABT}}$, which may be used to abort a bus cycle and any associated instruction. It generates an interrupt request if an instruction was aborted. See Sec. 3.5.4.

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 four 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) Saving Processor Status.
The PSR copy is pushed onto the Interrupt Stack as a 16-bit quantity.
3) Vector Acquisition.
A Vector is either obtained from the Data Bus or is supplied by default.
4) 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.



TL/C/5491-33

FIGURE 3-22. Interrupt Dispatch and Cascade Tables.

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

Full sequences of events in processing interrupts and traps may be found as follows:

Interrupt on $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pin:	Sec. 3.8.7.1.
Abort Interrupt:	Sec. 3.8.7.4.
Traps (except Trace):	Sec. 3.8.7.2.
Trace Trap:	Sec. 3.8.7.3.

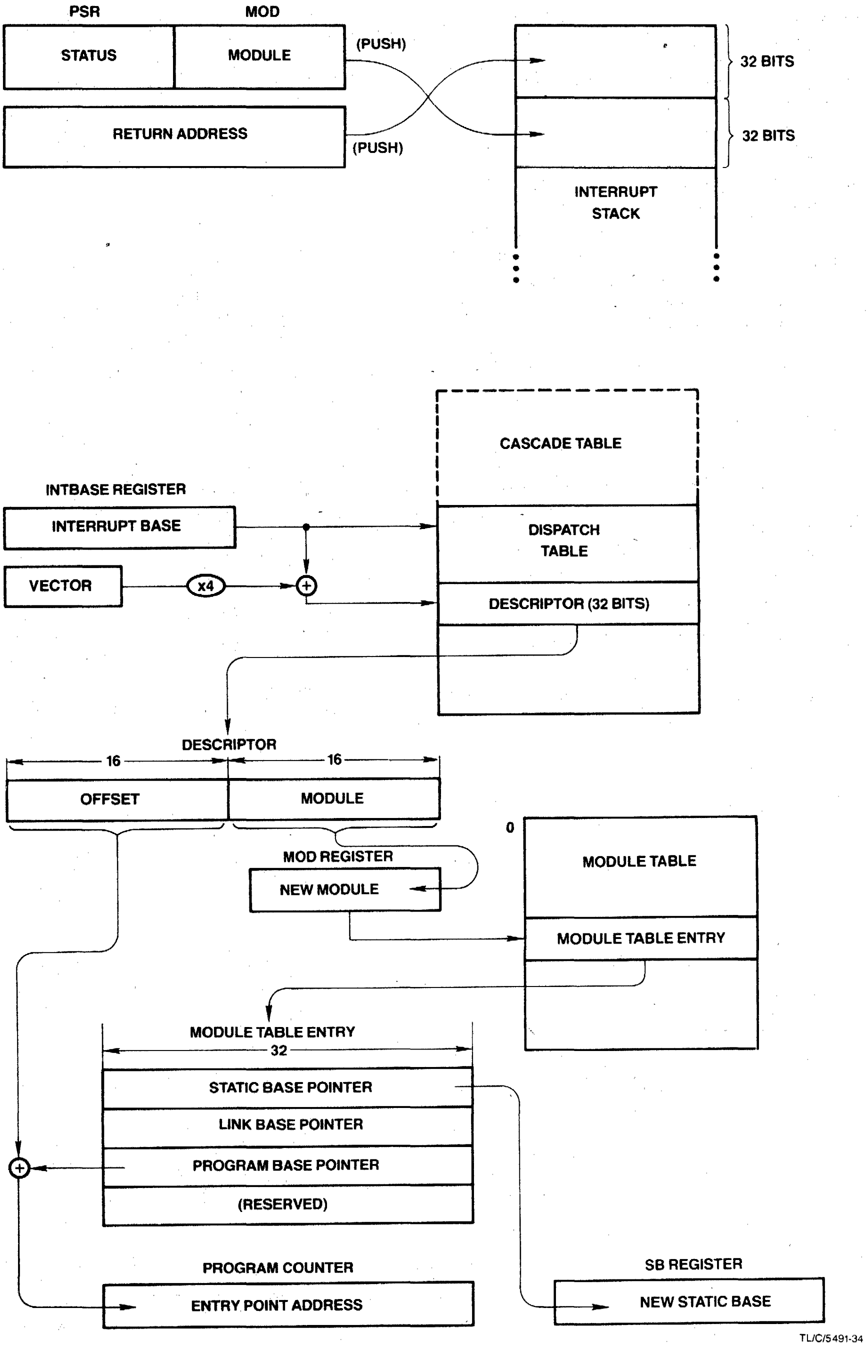


FIGURE 3-23. Interrupt/Trap Service Routine Calling Sequence.

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.

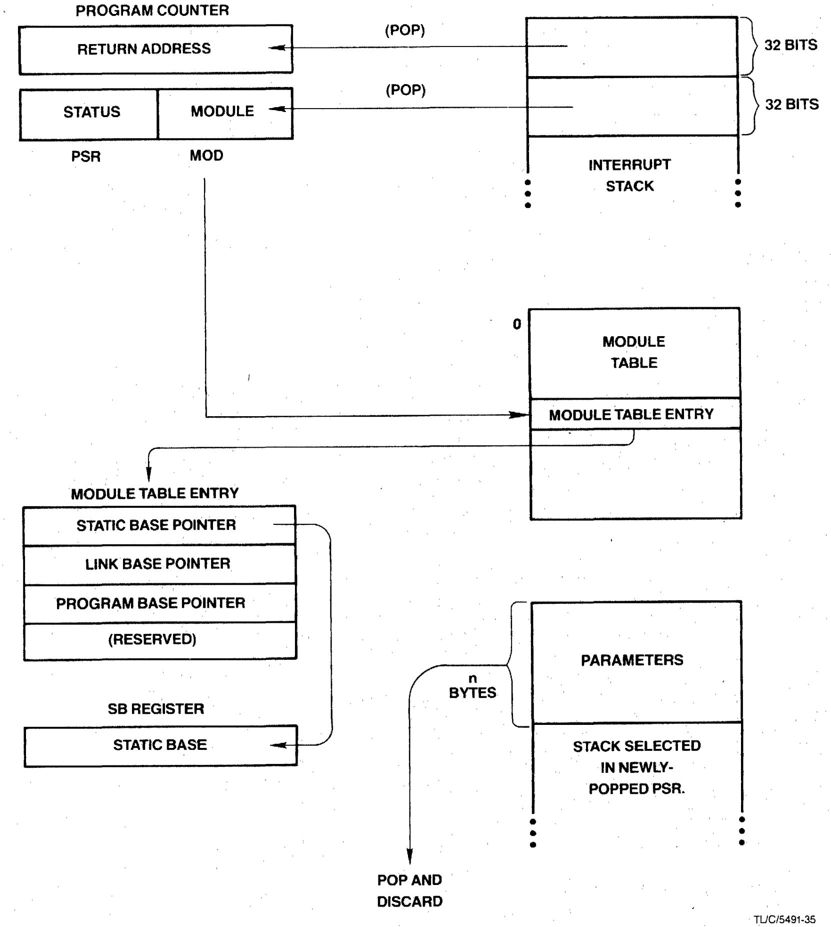
3.8.3 Maskable Interrupts (The $\overline{\text{INT}}$ Pin)

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).

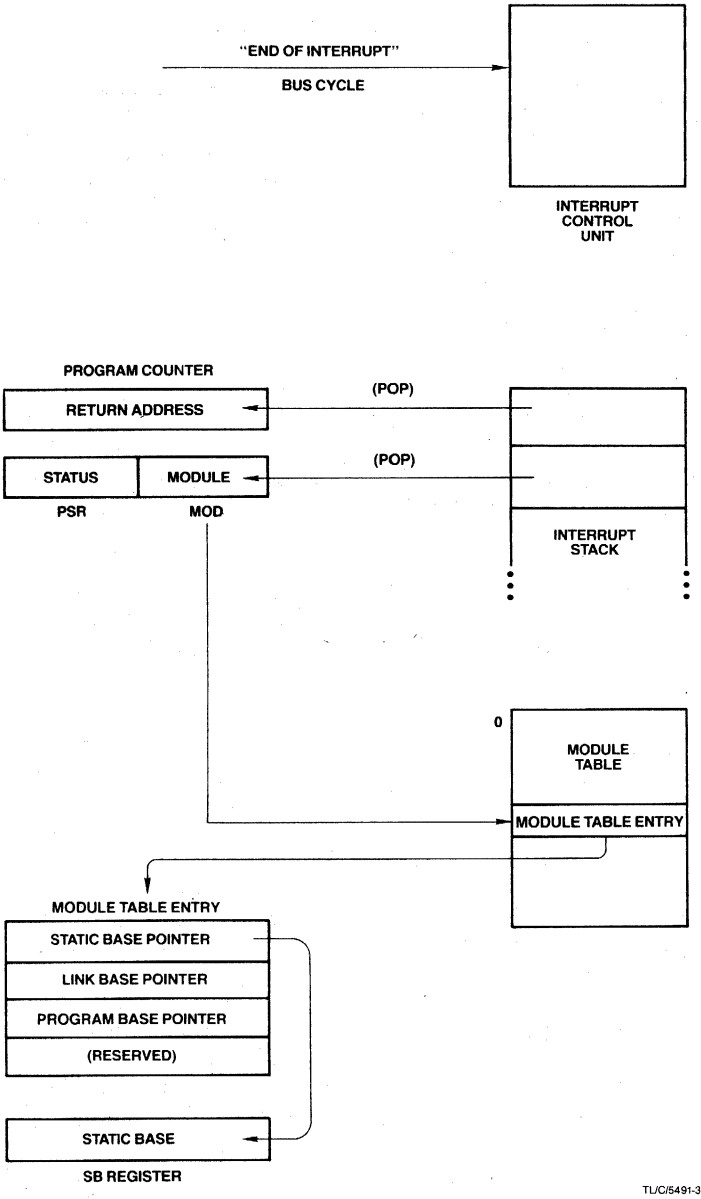
3.8.3.1 Non-Vectored Mode

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/C/5491-35

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



TL/C/5491-36

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

3.8.3.2 Vectored Mode: Non-Cascaded Case

In the Vectored mode, the CPU uses an NS16202 Interrupt Control Unit (ICU) to prioritize up to 16 interrupt requests. Upon receipt of an interrupt request on the $\overline{\text{INT}}$ pin, the CPU performs an "Interrupt Acknowledge, Master" bus cycle (Sec. 3.4.2) 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 NS16202 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 $\overline{\text{INT}}$ 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 (Sec. 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 (Sec. 3.4.2), whereupon the Master ICU again provides the negative Cascade Table index. The CPU, seeing a negative value, uses it to find the corresponding Cascade Address from the Cascade Table. Applying this address, it performs an "End of Interrupt, Cascaded" bus cycle (Sec. 3.4.2), informing the Cascaded ICU of the completion of the service routine. The byte read from the Cascaded ICU is discarded.

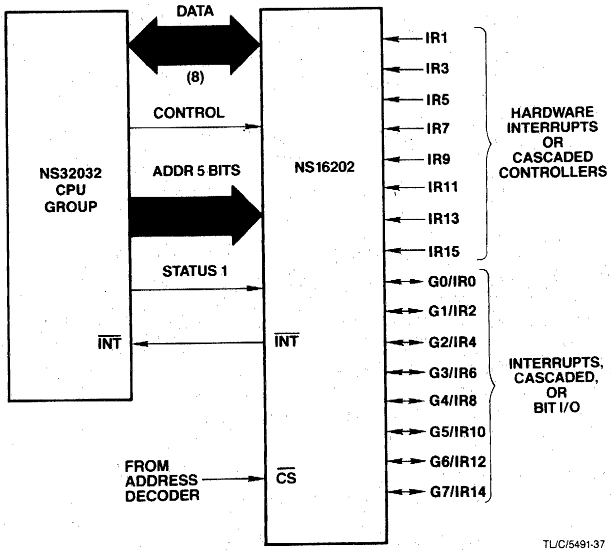


FIGURE 3-26. Interrupt Control Unit Connections (16 Levels).

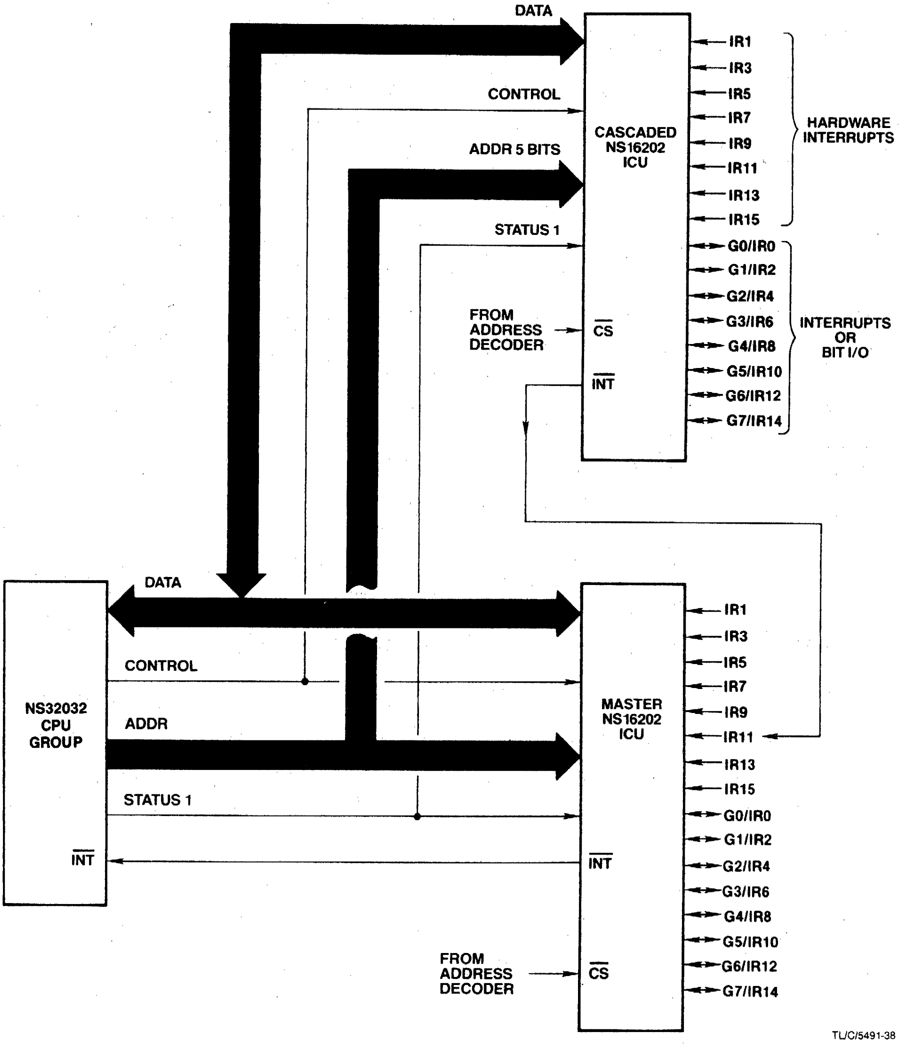


FIGURE 3-27. Cascaded Interrupt Control Unit Connections.

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 (Sec. 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 Sec. 3.8.7.1.

3.8.5 Traps

A trap is an internally-generated interrupt request caused as a direct and immediate result of the execution of an instruction. The Return Address pushed by any trap except Trap (TRC) is the address of the first byte of the instruction during which the trap occurred. Traps do not disable interrupts, as they are not associated with external events. Traps recognized by the NS32032 CPU are:

Trap (FPU): An exceptional condition was detected by the NS16081 Floating Point Unit or another Slave Processor during the execution of a Slave Instruction. This trap is requested via the Status Word returned as part of the Slave Processor Protocol (Sec. 3.9.1).

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

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

Trap (DVZ): An attempt was made to divide an integer by zero. (The FPU 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 NS16032 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: Detailed Flow

For purposes of the following detailed discussion of interrupt and trap service sequences, a single sequence called "Service" is defined in Figure 3-28. 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 sequence followed in processing either Maskable or Non-Maskable interrupts (on the $\overline{\text{INT}}$ or $\overline{\text{NMI}}$ pins, respectively), see Sec. 3.8.7.1. For Abort interrupts, see Sec. 3.8.7.4. For the Trace Trap, see Sec. 3.8.7.3, and for all other traps see Sec. 3.8.7.2.

3.8.7.1 Maskable/Non-Maskable Interrupt Sequence

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

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

   Otherwise, set "Return Address" to the address of the next instruction.
2. Copy the Processor Status Register (PSR) into a temporary register, then clear PSR bits S, U, T, P and I.
3. If the interrupt is Non-Maskable:
   a. Read a byte from address 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" ≥ 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) Push MOD Register onto the Interrupt Stack as a 16-bit value. (The PSR has already been pushed as a 16-bit value.)

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

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

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

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

6) 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.

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.

FPU:	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 NS32032 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 (Sec. 2.1.3). Any Slave Instruction which does not have its corresponding Configuration Register bit set will trap as undefined, without any Slave Processor communication attempted by the CPU. This allows software simulation of a non-existent Slave Processor.

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, Sec. 3.4.2), the CPU transfers the ID Byte on the least-significant byte 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, Sec. 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

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.

solely responsible for memory accesses, these extensions are not sent to the Slave processor. The Status Code applied is 1101 (Transfer Slave Processor Operand, Sec. 3.4.2).

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

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

Upon receiving the pulse on $\overline{\text{SPC}}$, the CPU uses $\overline{\text{SPC}}$ to read a Status Word from the Slave Processor, applying Status Code 1110 (Read Slave Status, Sec. 3.4.2). 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 FPU vector in the Interrupt Table. Certain Slave Processor instructions cause CPU PSR bits to be loaded from the Status Word.

The last step in the protocol is for the CPU to read a result, if any, and transfer it to the destination. The Read cycles from the Slave Processor are performed by the CPU while applying Status Code 1101 (Transfer Slave Operand, Sec. 3.4.2).

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 Programmer's Manual).

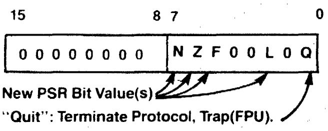
The Operand Issued columns show the sizes of the operands issued to the Floating Point Unit by the CPU. "D" indicates a 32-bit Double Word. "i" indicates that the instruction specifies an integer size for the operand (B = Byte, W = Word, D = Double Word). "f" indicates that the instruction specifies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

NOTE:
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.



TL/C/5491-41

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 futher details of the Memory Management Instruction set, see the Programmer's Manual and the NS16082 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 Programmer's Manual and the NS16082 Memory Management Unit Data Sheet.

D = Double Word.

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

N/A = Not Applicable to this instruction.

3.9.4 Custom Slave Instructions

Provided in the NS32032 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
CCMPc	read.c	read.c	c	c	N/A	N,Z,L
CCV0ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV1ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV2ci	read.c	write.i	c	N/A	i to Op. 2	none
CCV3ic	read.i	write.c	i	N/A	c to Op. 2	none
CCV4DQ	read.D	write.Q	D	N/A	Q to Op. 2	none
CCV5QD	read.Q	write.D	Q	N/A	D to Op. 2	none
LCSR	read.D	N/A	D	N/A	N/A	none
SCSR	N/A	write.D	N/A	N/A	D to Op. 2	none
CATST0 *	addr	N/A	D	N/A	N/A	F
CATST1 *	addr	N/A	D	N/A	N/A	F
LCR *	read.D	N/A	D	N/A	N/A	none
SCR *	write.D	N/A	N/A	N/A	D to Op. 1	none

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

4 AC Electrical Characteristics

4.1 Definitions

All the timing specifications given in this section refer to 50% of the leading or trailing edges of the appropriate clock phase and 0.8V or 2.0V on the appropriate signal as illustrated in *Figures 4-1* and *4-2*, unless specifically stated otherwise.

Abbreviations:
L.E — leading edge
T.E. — trailing edge

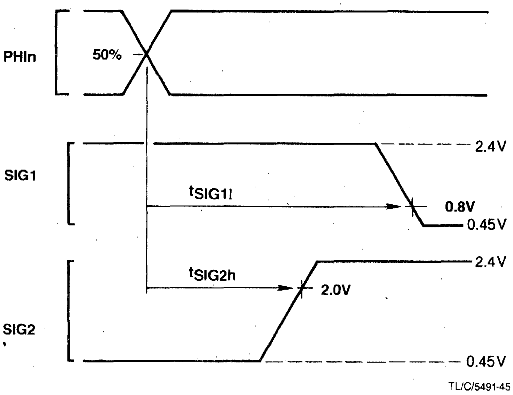


FIGURE 4-1. Timing Specification Standard (Signal Valid After Clock Edge)

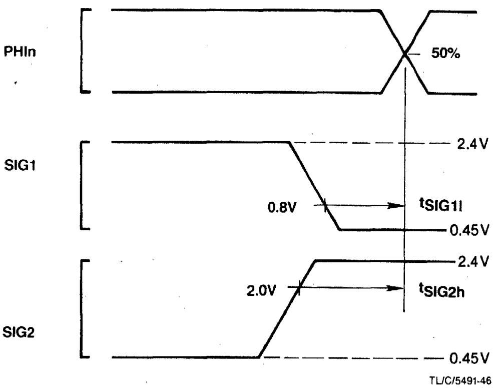


FIGURE 4-2. Timing Specification Standard (Signal Valid Before Clock Edge)

4.2 Timing Tables

4.2.1 Output Signals: Internal Propagation Delays, NS32032-4, NS32032-6

Maximum times assume capacitive loading of 100 pF.

Name	Description	Figure	Reference/Conditions	Min	Typ	Max	Unit
t_{ALv}	Address bits 0–23 valid	4-3	after L.E., PHI1 T1			80	ns
t_{ALh}	Address bits 0–23 hold	4-3	after L.E., PHI1 Tmmu or T2	0			ns
t_{Dv}	Data valid (write cycle)	4-3	after L.E., PHI1 T2			80	ns
t_{Dh}	Data hold (write cycle)	4-3	after L.E., PHI1 next T1 or Ti	0			ns
t_{ALADSs}	Address bits 0–23 set up to $\overline{ADS}$ T.E.	4-4	before $\overline{ADS}$ reaches 2.0V	20			ns
t_{ALADSh}	Address bits 0–23 hold from $\overline{ADS}$ T.E.	4-9	after $\overline{ADS}$ reaches 2.0V	10			ns
t_{ALf}	Address bits 0–23 floating (no MMU)	4-4	after L.E., PHI1 T2			25	ns
t_{ADf}	Data bits D24–D31 floating (no MMU)	4-4	after L.E., PHI1 T2			25	ns
t_{ALMf}	Address bits 0–23 floating (with MMU)	4-8	after L.E., PHI1 Tmmu			25	ns
t_{ADMf}	Data bits 21–31 floating (with MMU)	4-8	after L.E., PHI1 Tmmu			25	ns
t_{BEv}	$\overline{BEn}$ signals valid	4-3	after L.E., PHI2 T4			95	ns
t_{BEh}	$\overline{BEn}$ signals hold	4-3	after L.E., PHI2 T4 or Ti	0			ns
t_{STv}	Status (ST0–ST3) valid	4-3	after L.E., PHI1 T4 (before T1, see note)			90	ns
t_{STh}	Status (ST0–ST3) hold	4-3	after L.E., PHI1 T4 (after T1)	0			ns
t_{DDINv}	$\overline{DDIN}$ signal valid	4-4	after L.E., PHI1 T1			110	ns
t_{DDINh}	$\overline{DDIN}$ signal hold	4-4	after L.E., PHI1 next T1 or Ti	0			ns
t_{ADSa}	$\overline{ADS}$ signal active (low)	4-3	after L.E., PHI1 T1			50	ns
t_{ADSia}	$\overline{ADS}$ signal inactive	4-3	after T.E., PHI1 T1			65	ns
t_{ADSw}	$\overline{ADS}$ pulse width	4-3	at 0.8V, both edges	60			ns
t_{DSa}	$\overline{DS}$ signal active (low)	4-3	after L.E., PHI1 T2			70	ns

4.2.1 Output Signals: Internal Propagation Delays, NS32032-4, NS32032-6 (continued)

Name	Description	Figure	Reference/Conditions	Min	Typ	Max	Unit
t_{DSia}	$\overline{\text{DS}}$ signal inactive	4-3	after L.E., PHI1 T4			60	ns
t_{ALf}	AD0-AD23 floating (caused by $\overline{\text{HOLD}}$)	4-5	after L.E., PHI1 T1			100	ns
t_{ADf}	D24-D31 floating (caused by $\overline{\text{HOLD}}$)	4-5	after L.E., PHI1 T1			100	ns
t_{ADSf}	$\overline{\text{ADS}}$ floating (caused by $\overline{\text{HOLD}}$)	4-5	after L.E., PHI1 Ti			100	ns
t_{BEf}	$\overline{\text{BEn}}$ floating (caused by $\overline{\text{HOLD}}$)	4-5	after L.E., PHI1 Ti			100	ns
t_{DDINf}	$\overline{\text{DDIN}}$ floating (caused by $\overline{\text{HOLD}}$)	4-5	after L.E., PHI1 Ti			100	ns
t_{HLDAa}	$\overline{\text{HLDA}}$ signal active (low)	4-5	after L.E., PHI1 Ti			100	ns
t_{HLDAia}	$\overline{\text{HLDA}}$ signal inactive	4-7	after L.E., PHI1 Ti			100	ns
t_{ADSr}	$\overline{\text{ADS}}$ signal returns from floating (caused by $\overline{\text{HOLD}}$)	4-7	after L.E., PHI1 Ti			100	ns
t_{BEr}	$\overline{\text{BEn}}$ signals return from floating (caused by $\overline{\text{HOLD}}$)	4-7	after L.E., PHI1 Ti			100	ns
t_{DDINr}	$\overline{\text{DDIN}}$ signal returns from floating (caused by $\overline{\text{HOLD}}$)	4-7	after L.E., PHI1 Ti			100	ns
t_{ALf}	AD0-AD15 floating (caused by $\overline{\text{FLT}}$)	4-8	after L.E., PHI1 Tf			60	ns
t_{DDINf}	$\overline{\text{DDIN}}$ signal floating (caused by $\overline{\text{FLT}}$)	4-8	after $\overline{\text{FLT}}$ reaches 0.8V			80	ns
	$\overline{\text{ADS}}$ signal floating (caused by $\overline{\text{FLT}}$)	4-8	after $\overline{\text{FLT}}$ reaches 0.8V			80	ns
t_{BEI}	$\overline{\text{BEn}}$ signals invalid (caused by $\overline{\text{FLT}}$)	4-8	after $\overline{\text{FLT}}$ reaches 0.8V			100	ns
t_{PABE}	Delay from PA1 to $\overline{\text{BEn}}$	4-8	after PA1 reaches .08V or 2.0V			50	ns
t_{DDINr}	$\overline{\text{DDIN}}$ signal returns from floating (caused by $\overline{\text{FLT}}$)	4-9	after $\overline{\text{FLT}}$ reaches 2.0V			75	ns
t_{BEr}	$\overline{\text{BEn}}$ signals return from floating (caused by $\overline{\text{FLT}}$)	4-9	after $\overline{\text{FLT}}$ reaches 2.0V			90	ns
t_{SPCa}	$\overline{\text{SPC}}$ output active (low)	4-12	after L.E., PHI1 T1			50	ns
t_{SPCia}	$\overline{\text{SPC}}$ output inactive	4-12	after L.E., PHI1 T4			50	ns
t_{SPCnf}	$\overline{\text{SPC}}$ output nonforcing	4-14	after L.E., PHI2 T4			40	ns
t_{Dv}	Data valid (slave processor write)	4-12	after L.E., PHI1 T1			80	ns
t_{Dh}	Data hold (slave processor write)	4-12	after L.E., PHI1 next T1 or Ti	0			ns
t_{PFSw}	$\overline{\text{PFS}}$ pulse width	4-17	at 0.8V, both edges	70			ns
t_{PFSa}	$\overline{\text{PFS}}$ pulse active (low)	4-17	after L.E., PHI2			70	ns
t_{PFSia}	$\overline{\text{PFS}}$ pulse inactive	4-17	after L.E., PHI2			70	ns
t_{ILOs}	$\overline{\text{ILO}}$ signal setup	4-19a	before L.E., PHI1 T1 of first interlocked write cycle	0			ns
t_{ILOh}	$\overline{\text{ILO}}$ signal hold	4-19b	after L.E., PHI1 T3 of last interlocked read cycle	0			ns
t_{ILOa}	$\overline{\text{ILO}}$ signal active (low)	4-20	after L.E., PHI1			70	ns
t_{ILOia}	$\overline{\text{ILO}}$ signal inactive	4-20	after L.E., PHI1			70	ns
t_{USs}	U/$\overline{\text{S}}$ signal setup	4-21	before T.E., PHI1 T4 or Ti	15			ns
t_{USh}	U/$\overline{\text{S}}$ signal hold	4-21	after L.E., PHI1 T1	2			t_{Cp}
t_{NSPF}	Nonsequential fetch to next $\overline{\text{PFS}}$ clock cycle	4-18b	after L.E., PHI1 T1	4			t_{Cp}
t_{PFNS}	$\overline{\text{PFS}}$ clock cycle to next nonsequential fetch	4-18a	before L.E., PHI1 T1	4			t_{Cp}
t_{LXPF}	Last operand transfer of an instruction to next $\overline{\text{PFS}}$ clock cycle	4-28	before L.E., PHI1 T1 of first bus cycle of transfer	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...".

4.2.2 Input Signal Requirements: NS32032-4, NS32032-6

Name	Description	Figure	Reference/Conditions	Min	Typ	Max	Unit
t_{PWR}	Power stable to $\overline{RST}$ T.E.	4-24	after V_{CC} reaches 4.5V	50			μs
t_{DIs}	Data in setup (read cycle)	4-4	before T.E., PHI2 T3	20			ns
t_{DIh}	Data in hold (read cycle)	4-4	after T.E., PHI2 T3	10			ns
t_{HLDa}	$\overline{HOLD}$ active (low) setup time (See note)	4-5	before T.E., PHI2 TX1	25			ns
t_{HLDia}	$\overline{HOLD}$ inactive setup time	4-7	before T.E., PHI2 Ti	25			ns
t_{HLDh}	$\overline{HOLD}$ hold time	4-5	after L.E., PHI1 TX2	0			ns
t_{FLTa}	$\overline{FLT}$ active (low) setup time	4-8	before T.E., PHI2 Tmmu	25			ns
t_{FLTia}	$\overline{FLT}$ inactive setup time	4-9	before T.E., PHI2 T2	25			ns
t_{RDYs}	RDY setup time	4-10, 4-11	before T.E., PHI2 T2 or T3	25			ns
t_{RDYh}	RDY hold time	4-10, 4-11	after T.E., PHI1 T3	0			ns
t_{ABTs}	$\overline{ABT}$ setup time ($\overline{FLT}$ inactive)	4-22	before T.E., PHI2 Tmmu	30			ns
t_{ABTs}	$\overline{ABT}$ setup time ($\overline{FLT}$ active)	4-23	before T.E., PHI2 T2	30			ns
t_{ABTh}	$\overline{ABT}$ hold time	4-22	after L.E., PHI1	0			ns
t_{RSTs}	$\overline{RST}$ setup time	4-24, 4-25	before T.E., PHI1	20			ns
t_{RSTw}	$\overline{RST}$ pulse width	4-25	at 0.8V (both edges)	64			t_{Cp}
t_{INTs}	$\overline{INT}$ setup time	4-26	before T.E., PHI1	20			ns
t_{NMIw}	$\overline{NMI}$ pulsewidth	4-27	at 0.8V (both edges)	40			ns
t_{DIs}	Data setup (slave read cycle)	4-13	before T.E., PHI2 T1	20			ns
t_{DIh}	Data hold (slave read cycle)	4-13	after T.E., PHI2 T1	10			ns
t_{SPCw}	$\overline{SPC}$ pulse width (from slave processor)	4-12	at 0.8V (both edges)	30			ns
t_{ATs}	$\overline{AT/SPC}$ setup for address translation strap	4-15	before L.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	1			t_{Cp}
t_{ATh}	$\overline{AT/SPC}$ hold for address translation strap	4-15	after T.E., PHI1 of cycle during which $\overline{RST}$ pulse is removed	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.

4.2.3 Clocking Requirements: NS32032-4

Name	Description	Figure	Reference/Conditions	Min	Typ	Max	Unit
t_{CLr}	PHI1, PHI2 rise time	4-16	to V_{CH} (see page 2)			9	ns
t_{CLf}	PHI1, PHI2 fall time	4-16	from 90-10% of V_{CH} (see page 2)			9	ns
t_{CLh}	PHI1, PHI2 high time	4-16		0.4			t_{Cp}
t_{CLl}	PHI1, PHI2 low time	4-16		0.35			t_{Cp}
t_{Cp}	Clock period	4-16		240		5000	ns
t_{OVL}	Non-overlap time	4-16	at 10% of V_{CH} (see page 2)	0			ns

4.2.4 Clocking Requirements: NS32032-6

Name	Description	Figure	Reference/Conditions	Min	Typ	Max	Unit
t_{CLr}	PHI1, PHI2 rise time	4-16	to V_{CH} (see page 2)			9	ns
t_{CLf}	PHI1, PHI2 fall time	4-16	from 90-10% of V_{CH} (see page 2)			9	ns
t_{CLh}	PHI1, PHI2 high time	4-16		0.4			t_{Cp}
t_{CLl}	PHI1, PHI2 low time	4-16		0.35			t_{Cp}
t_{Cp}	Clock period	4-16		160		5000	ns
t_{OVL}	Non-overlap time	4-16	at 10% of V_{CH} (see page 2)	0			ns

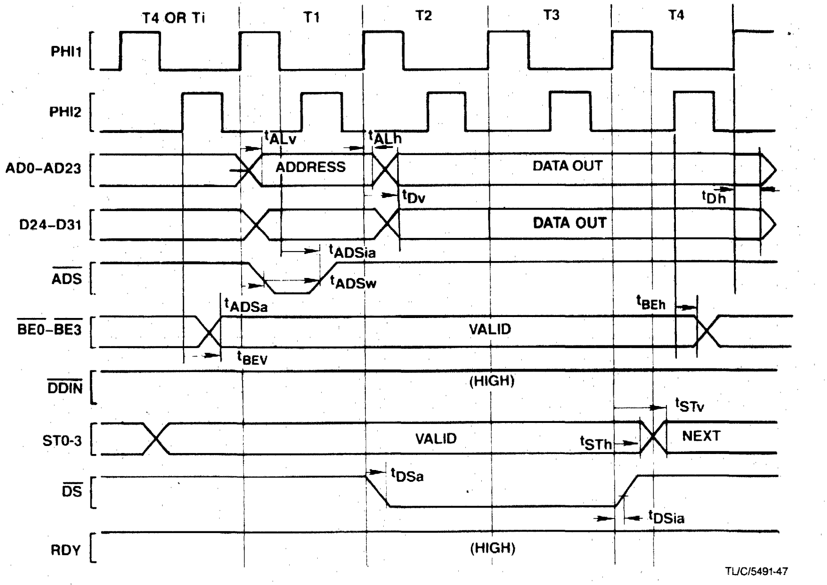


FIGURE 4-3. Write Cycle.

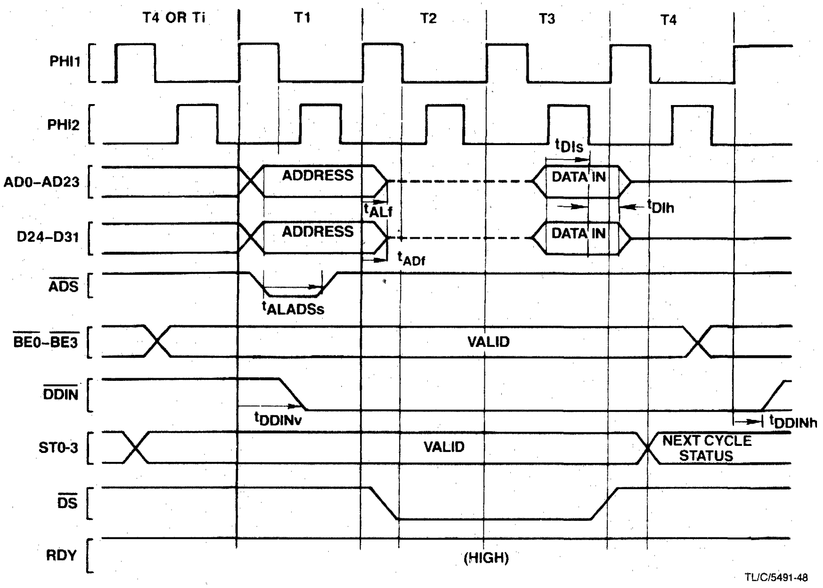
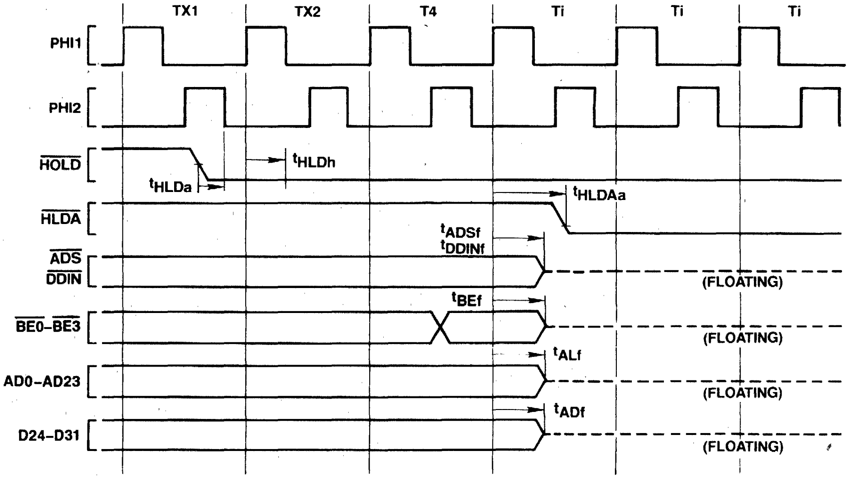


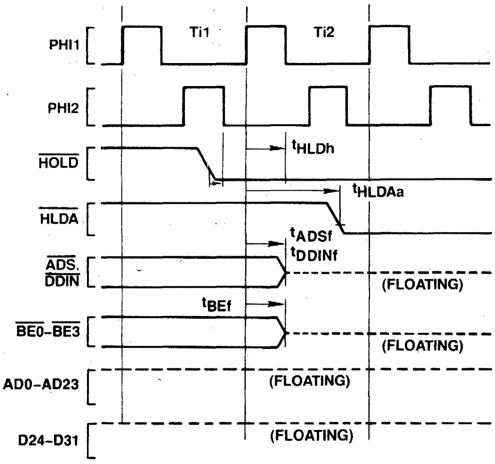
FIGURE 4-4. Read Cycle.



TL/C/5491-49

FIGURE 4-5. Floating by $\overline{\text{HOLD}}$ Timing (CPU Not Idle Initially).

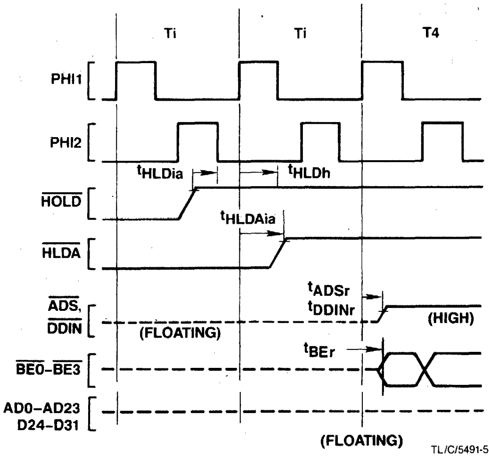
Note that whenever the CPU is not idling (not in Ti), the $\overline{\text{HOLD}}$ request ($\overline{\text{HOLD}}$ low) must be active tHLDa before the trailing edge of PHI2 of the clock cycle that appears two clock cycles before T4 (TX1) and stay low until tHLDh after the leading edge of PHI1 of the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



TL/C/5491-50

FIGURE 4-6. Floating by $\overline{\text{HOLD}}$ Timing (CPU Initially Idle).

Note that during Ti1 the CPU is already idling.



TL/C/5491-51

FIGURE 4-7. Release from $\overline{\text{HOLD}}$.

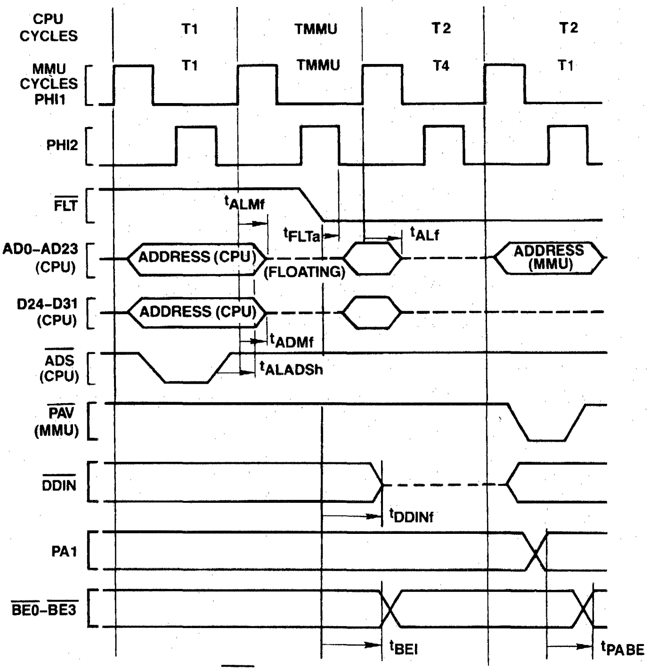
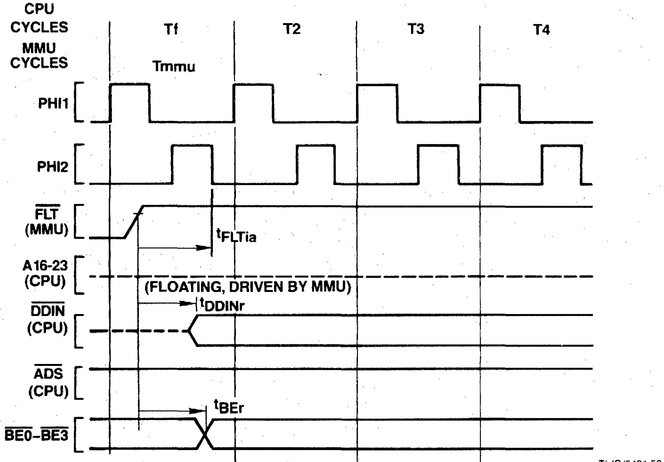


FIGURE 4-8. $\overline{\text{FLT}}$ Initiated Float Cycle Timing.

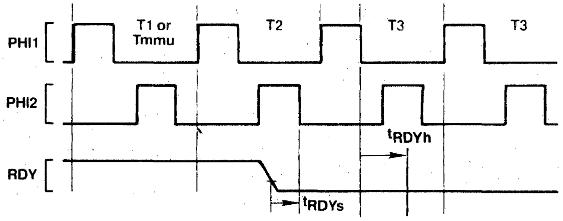
TL/C/5491-52



TL/C/5491-53

FIGURE 4-9. Release from $\overline{\text{FLT}}$ Timing.

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



TL/C/5491-54

FIGURE 4-10. Ready Sampling (CPU Initially READY).

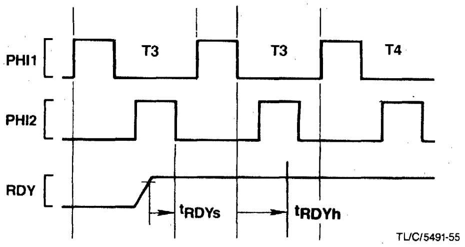


FIGURE 4-11. Ready Sampling (CPU Initially NOT READY).

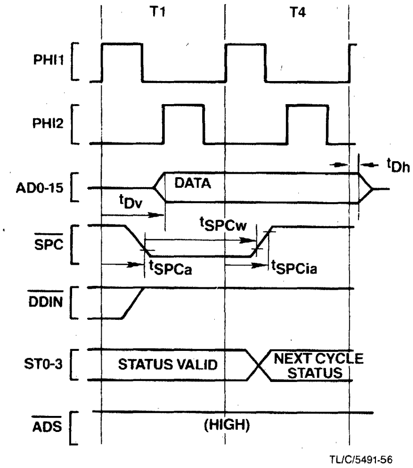


FIGURE 4-12. Slave Processor Write Timing.

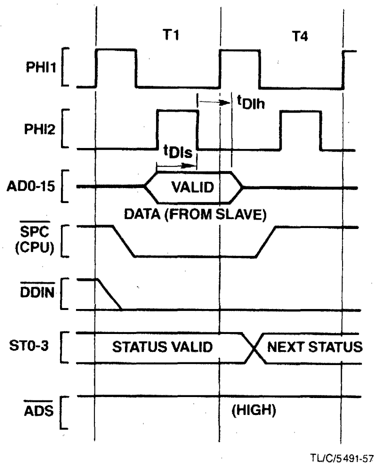


FIGURE 4-13. Slave Processor Read Timing.

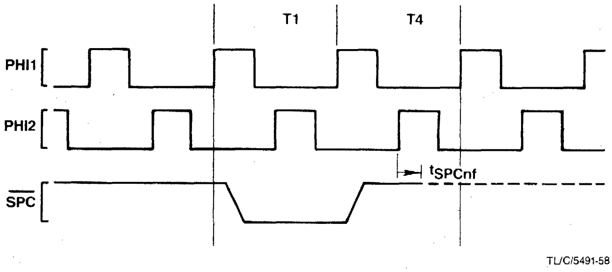


FIGURE 4-14. $\overline{\text{SPC}}$ Non-Forcing Delay.

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

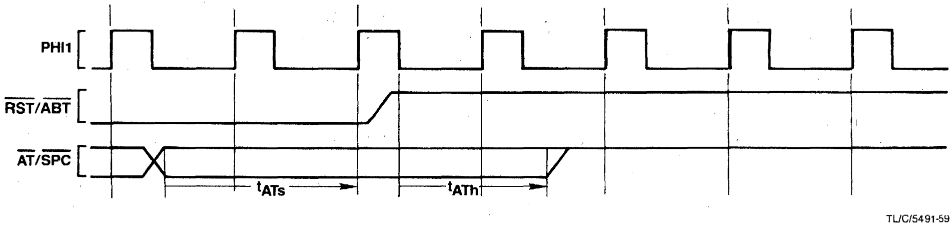
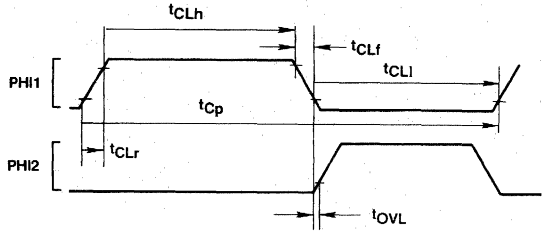
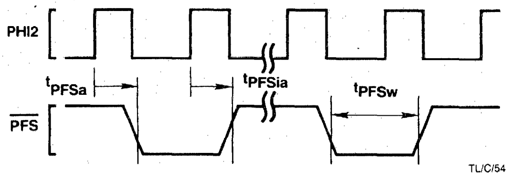


FIGURE 4-15. Reset Configuration Timing.



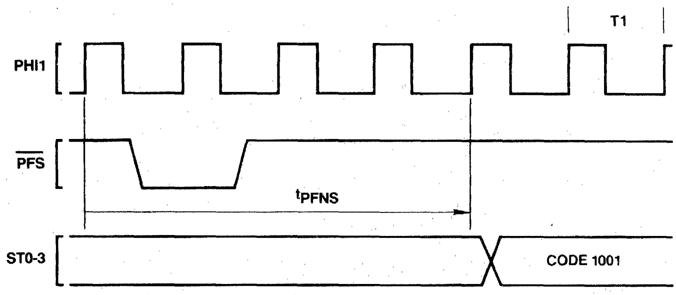
TL/C/5491-60

FIGURE 4-16. Clock Waveforms.



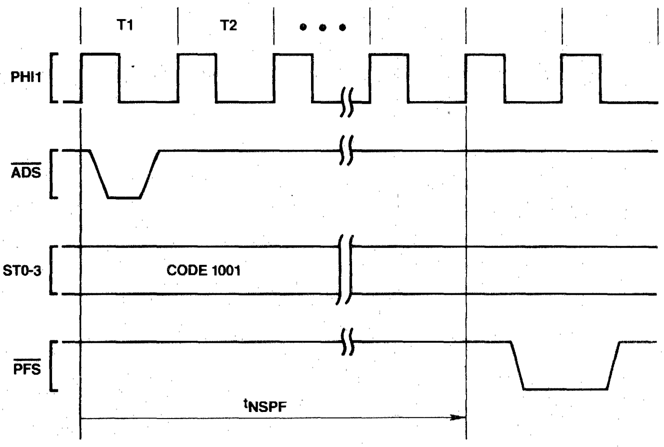
TL/C/5491-61

FIGURE 4-17. Relationship of $\overline{PFS}$ to Clock Cycles.



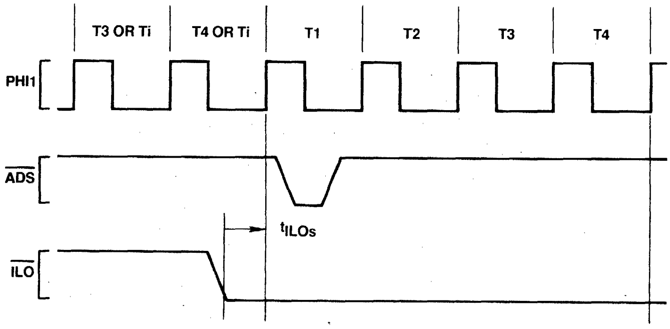
TL/C/5491-62

FIGURE 4-18a. Guaranteed Delay, $\overline{PFS}$ to Non-Sequential Fetch.



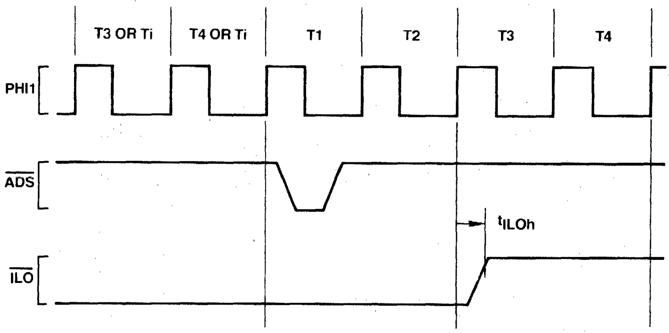
TL/C/5491-63

FIGURE 4-18b. Guaranteed Delay, Non-Sequential Fetch to $\overline{PFS}$.



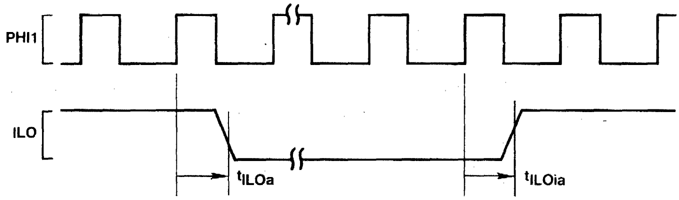
TL/C/5491-64

FIGURE 4-19a. Relationship of $\overline{ILO}$ to First Operand Cycle of an Interlocked Instruction.



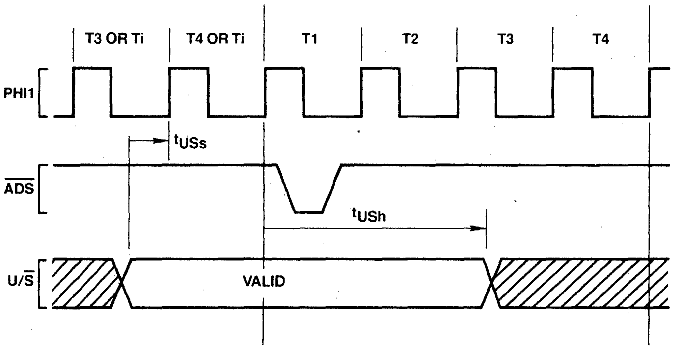
TL/C/5491-65

FIGURE 4-19b. Relationship of $\overline{ILO}$ to Last Operand Cycle of an Interlocked Instruction.



TL/C/5491-66

FIGURE 4-20. Relationship of $\overline{ILO}$ to Any Clock Cycle.



TL/C/5491-67

FIGURE 4-21. U/$\overline{S}$ Relationship to Any Bus Cycle — Guaranteed Valid Interval.

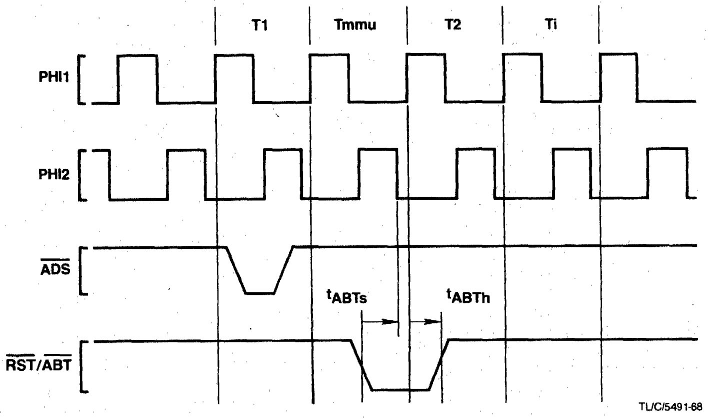


FIGURE 4-22. Abort Timing, $\overline{FLT}$ Not Applied.

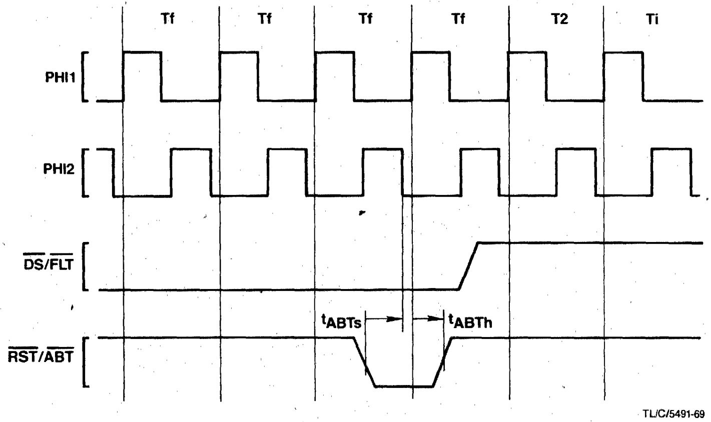

Tf
Tf
Tf
Tf
T2
Ti
PHI1
PHI2
$\overline{DS}/\overline{FLT}$
t_{ABTs}
t_{ABTh}
$\overline{RST}/\overline{ABT}$
TL/C/5491-69

FIGURE 4-23. Abort Timing, $\overline{FLT}$ Applied.

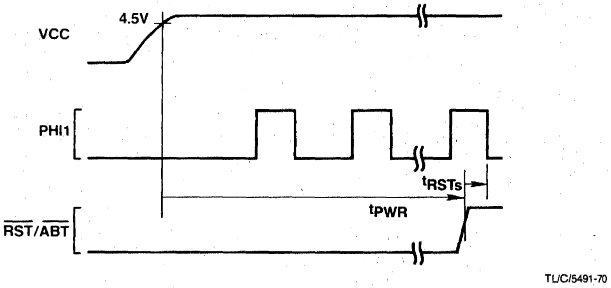


FIGURE 4-24. Power-On Reset.

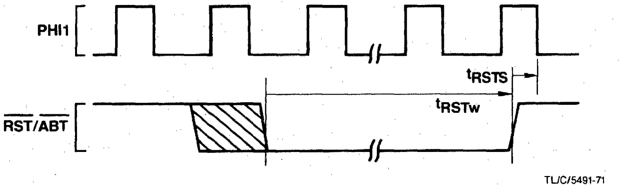


FIGURE 4-25. Non-Power-On Reset.

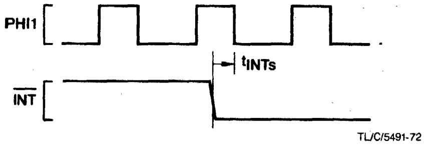


FIGURE 4-26. INT Interrupt Signal Detection.

Violation of tINTs timing is allowed, but detection then occurs one clock cycle later.

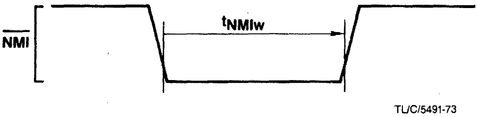


FIGURE 4-27. NMI Interrupt Signal Timing.

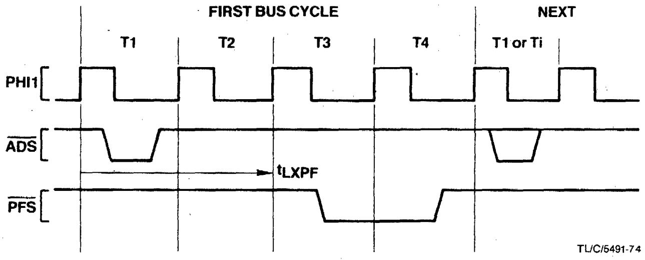


FIGURE 4-28. Relationship Between Last Data Transfer of an Instruction and PFS Pulse of Next Instruction.

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

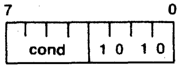
U/W	B	T

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

Configuration bits, in SETCFG:

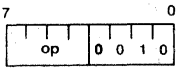
C	M	F	I

mreg: MMU Register number, in LMR, SMR.
0000 = BPR0
0001 = BPR1
0010 = (Reserved)
0011 = (Reserved)
0100 = PF0
0101 = PF1
0110 = (Reserved)
0111 = (Reserved)
1000 = SC
1001 = (Reserved)
1010 = MSR
1011 = BCNT
1100 = PTB0
1101 = PTB1
1110 = (Reserved)
1111 = EIA



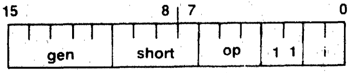
Format 0

Bcond	(BR)



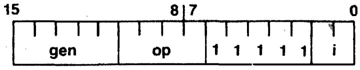
Format 1

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

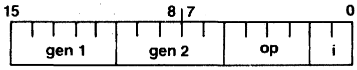
ADDQ	–000	ACB	–100
CMPQ	–001	MOVQ	–101
SPR	–010	LPR	–110
Scond	–011		



Format 3

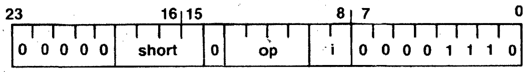
CXPD	−0000	ADJSP	−1010
BICPSR	−0010	JSR	−1100
JUMP	−0100	CASE	−1110
BISPSR	−0110		

Trap (UND) on XXX1, 1000



Format 4

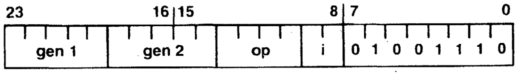
ADD	−0000	SUB	−1000
CMP	−0001	ADDR	−1001
BIC	−0010	AND	−1010
ADDC	−0100	SUBC	−1100
MOV	−0101	TBIT	−1101
OR	−0110	XOR	−1110



Format 5

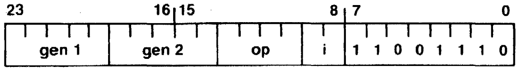
MOVS	−0000	SETCFG	−0010
CMPS	−0001	SKPS	−0011

Trap (UND) on 1XXX, 01XX



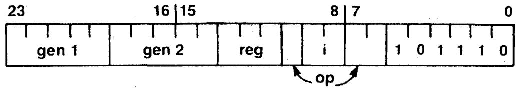
Format 6

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



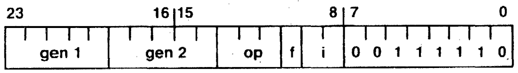
Format 7

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



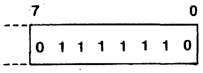
Format 8

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



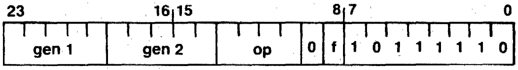
Format 9

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



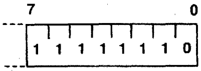
Format 10

Trap (UND) Always



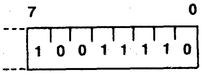
Format 11

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



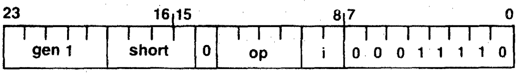
Format 12

Trap (UND) Always



Format 13

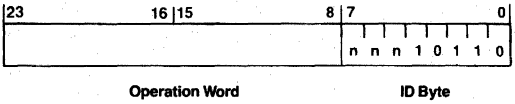
Trap (UND) Always



Format 14

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

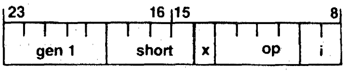
Trap (UND) on 01XX, 1XXX



Format 15
(Custom Slave)

nnn **Operation Word Format**

000

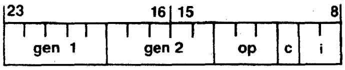


Format 15.0

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

Trap (UND) on all others

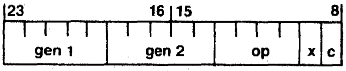
001



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	Trap (UND)	−1010
CCMP	−0010	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

7							0
0	1	0	1	1	1	1	0

Format 16

Trap (UND) Always

7							0
1	1	0	1	1	1	1	0

Format 17

Trap (UND) Always

7							0
1	0	0	0	1	1	1	0

Format 18

Trap (UND) Always

7	
X	X

Format 19

Trap (UND) Always

Implied Immediate Encodings:

7					
r7	r6	r5	r4	r3	r2

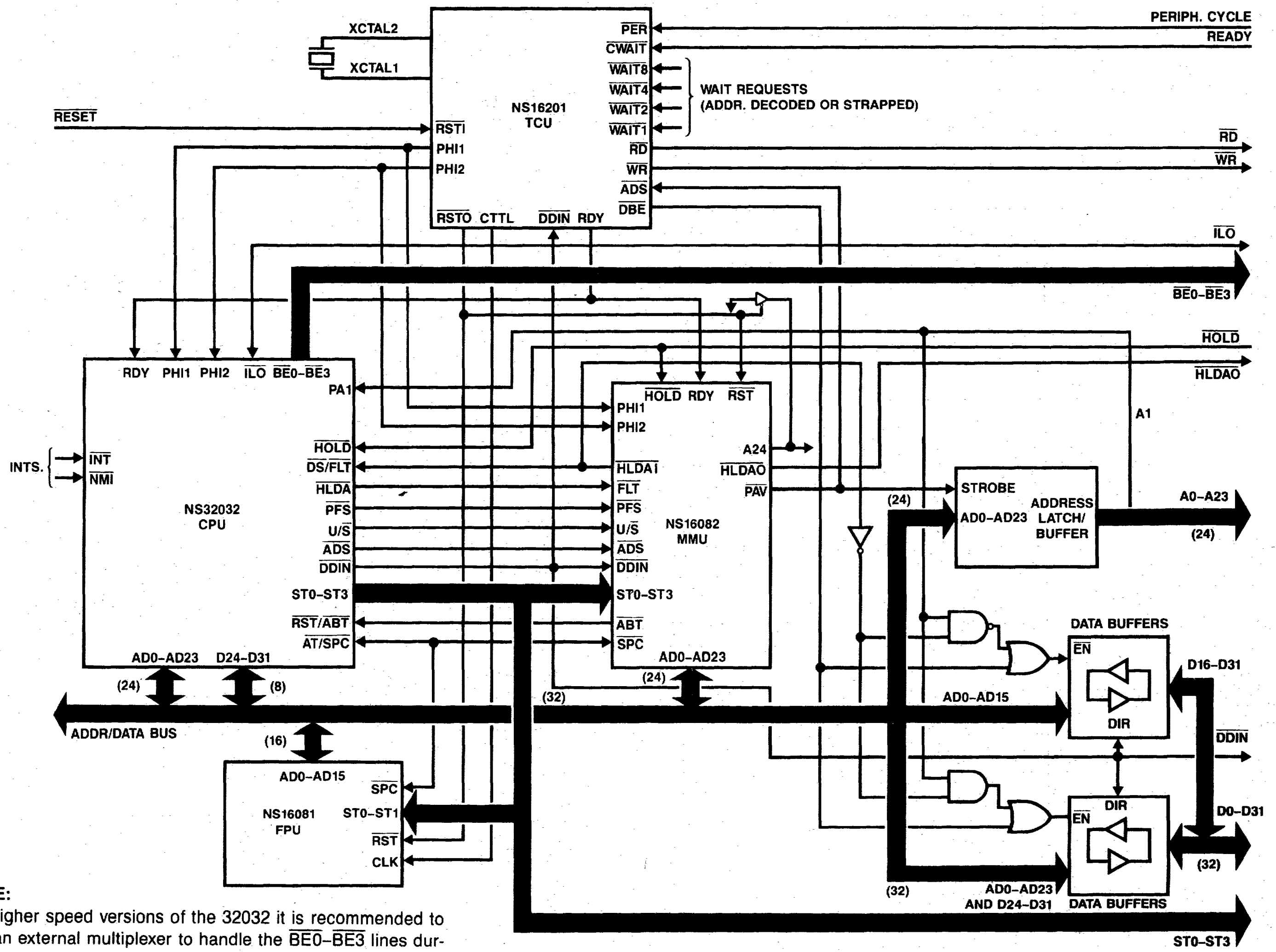
Register Mask, appended to SAVE, I

7					
r0	r1	r2	r3	r4	r5

Register Mask, appended to RESTOI

7	
offset	lengt

Offset/Length Modifier appended to IN

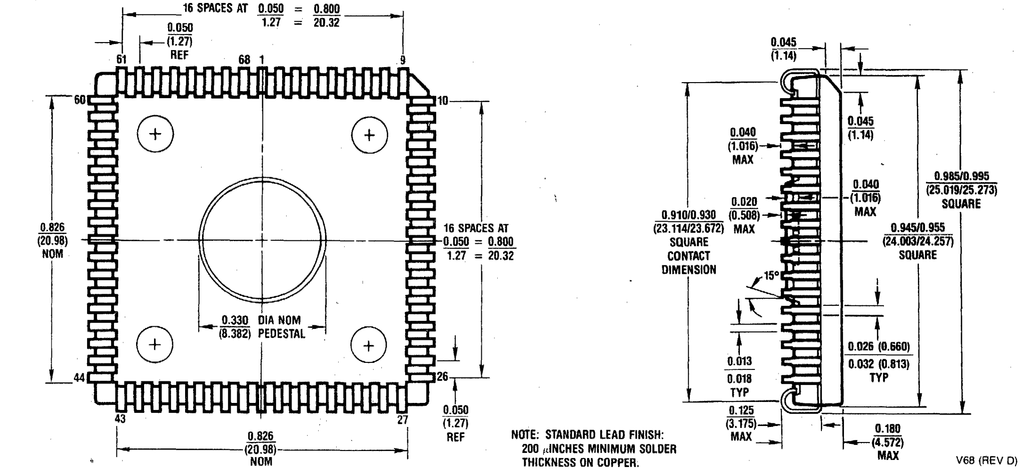


NOTE:
For higher speed versions of the 32032 it is recommended to use an external multiplexer to handle the BE0–BE3 lines during MMU cycles, instead of PA1.

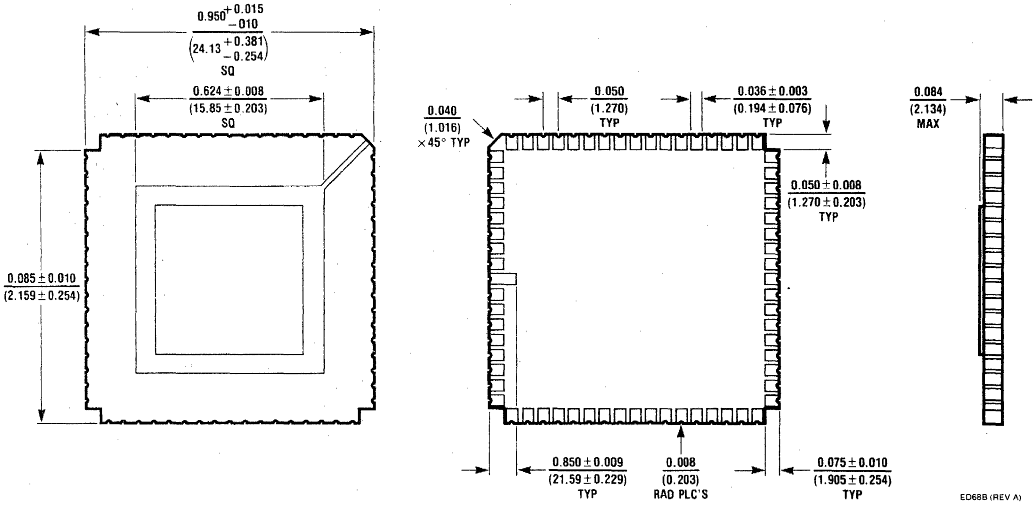
TL/C/5491-75

FIGURE B-1. Single Processor System Connection Diagram.

Physical Dimensions inches (millimeters)


16 SPACES AT 0.050/1.27 = 0.800/20.32
0.050 (1.27) REF
61
68
1
9
60
10
44
26
43
27
0.826 (20.98) NOM
16 SPACES AT 0.050/1.27 = 0.800/20.32
0.330 (8.382) DIA NOM PEDESTAL
0.050 (1.27) REF
0.826 (20.98) NOM
NOTE: STANDARD LEAD FINISH: 200 μINCHES MINIMUM SOLDER THICKNESS ON COPPER.
0.045 (1.14)
0.045 (1.14)
0.040 (1.016) MAX
0.020 (0.508) MAX
0.040 (1.016) MAX
0.910/0.930 (23.114/23.672) SQUARE CONTACT DIMENSION
0.985/0.995 (25.019/25.273) SQUARE
0.945/0.955 (24.003/24.257) SQUARE
15°
0.013 / 0.018 TYP
0.026 (0.660) / 0.032 (0.813) TYP
0.125 (3.175) MAX
0.180 (4.572) MAX
V68 (REV D)

Plastic Chip Carrier
NS Package Number V68


0.950 +0.015 −010
(24.13 +0.381 −0.254) SQ
0.624 ± 0.008 (15.85 ± 0.203) SQ
0.085 ± 0.010 (2.159 ± 0.254)
0.040 (1.016) × 45° TYP
0.050 (1.270) TYP
0.036 ± 0.003 (0.194 ± 0.076) TYP
0.050 ± 0.008 (1.270 ± 0.203) TYP
0.084 (2.134) MAX
0.850 ± 0.009 (21.59 ± 0.229) TYP
0.008 (0.203) RAD PLC'S
0.075 ± 0.010 (1.905 ± 0.254) TYP
ED68B (REV A)

Chip Carrier
NS Package Number ED68B

Slave Processors

National Semiconductor

PRELIMINARY

NS16081-6 Floating-Point Unit

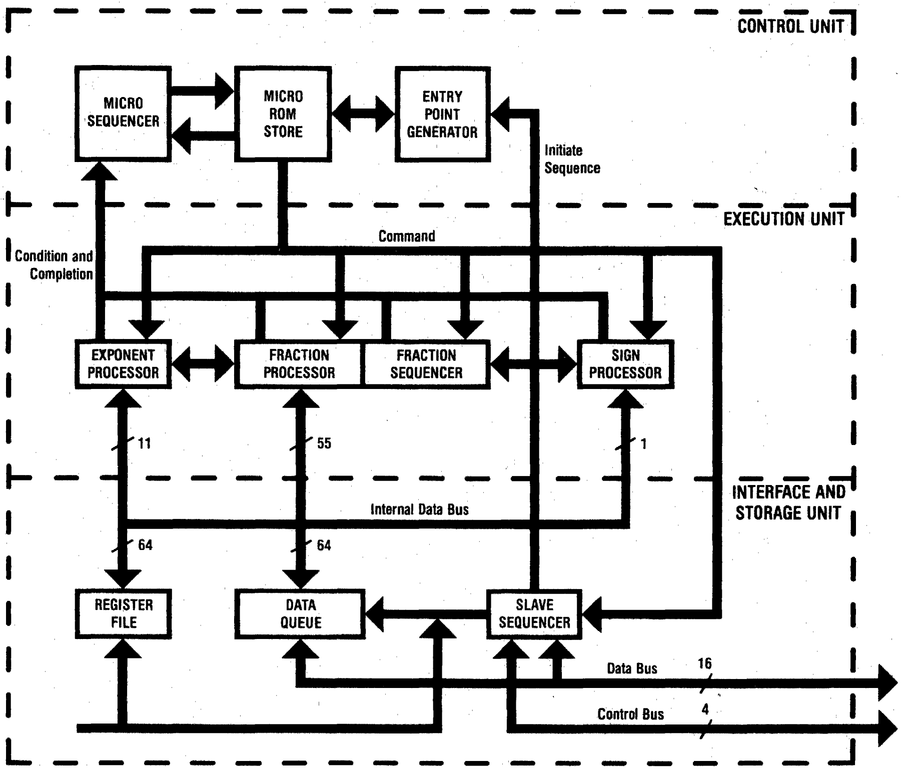
General Description

The NS16081 Floating-Point Unit functions as a slave processor in National Semiconductor's NS16000™ microcomputer family. It provides a high-speed floating-point instruction set for any NS16000 family CPU, while remaining architecturally consistent with the full two-address architecture and powerful addressing modes of the NS16000 microprocessor family.

Features

- Eight on-chip data registers
- Standard (32-bit) and long (64-bit) operations
- Supports proposed IEEE standard for binary floating-point arithmetic
- Directly compatible with NS16032, NS16008 and NS32032 CPUs
- High-speed XMOS™ technology
- Single 5V supply
- 24-pin dual-in-line package

Block Diagram



TL/C/5234-1

Absolute Maximum Ratings

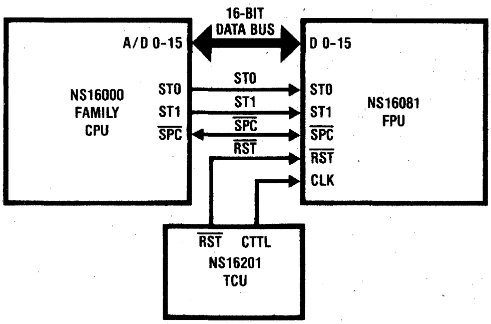
Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to GND	−0.5V to +7.0V
Power Dissipation	1.5W

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 DC Electrical Characteristics.*

DC Electrical Characteristics $T_A = 0°C$ to $70°C$, $V_{CC} = 5V \pm 5\%$, $GND = 0V$

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	Logical 1 Input Voltage		2.0		$V_{CC} + 0.5$	V
V_{IL}	Logical 0 Input Voltage		−0.5		0.8	V
V_{OH}	Logical 1 Output Voltage	$I_{OH} = -400\ \mu A$	2.4			V
V_{OL}	Logical 0 Output Voltage	$I_{OL} = 2$ mA			0.45	V
I_I	Input Leakage Current	$0 \leq V_{IN} \leq V_{CC}$	−10.0		10.0	μA
$I_{O(OFF)}$	Output Leakage Current	$0.45 \leq V_{IN} \leq 2.4V$	−20.0		20.0	μA
I_{CC}	Active Supply Current	$I_{OUT} = 0$, $T_A = 0°C$			300	mA

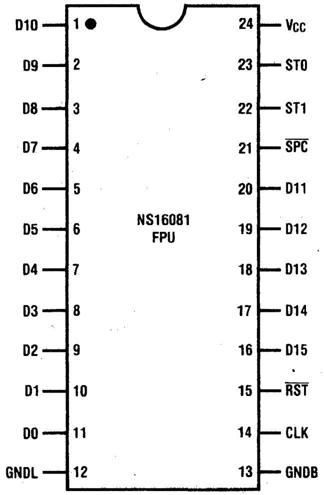
System Connections



TL/C/5234-2

Connection Diagram

Dual-In-Line Package



TOP VIEW

TL/C/5234-3

1. NS16081 FPU Pin Descriptions

The following are brief descriptions of all NS16081 FPU pins. The descriptions reference the relevant portions of the Functional Description, Section 3.

1.1. SUPPLIES

Power (V_{CC}): +5V positive supply. 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.

1.2. INPUT SIGNALS

Clock (CLK): TTL-level clock signal.

Reset ($\overline{RST}$): Active low. Initiates a Reset, Section 3.3.

Status (ST0, ST1): Active high. Input from CPU, Section 3.4. ST0 is the least significant bit. The status codes are:

00—(Reserved)
01—Transferring Operation Word or Operand
10—Transferring Status Word
11—Broadcasting Slave ID

1.3. INPUT/OUTPUT SIGNALS

Slave Processor Control ($\overline{SPC}$): Active low. Driven by the CPU as the data strobe for bus transfers to and from the NS16081 FPU, Section 3.4. Driven by the FPU to signal completion of an operation, Section 3.5.1.

Data Bus (D0–D15): Active high. 16-bit bus for data transfer. D0 is the least significant bit. Section 3.4.

2. Architectural Description

2.1. OPERAND FORMATS

The NS16081 FPU operates on two floating-point data types—single precision (32 bits) and double precision (64 bits). Floating-point instruction mnemonics use the suffix F (Floating) to select the single precision data type, and the suffix L (Long Floating) to select the double precision data type.

A floating-point number is divided into three fields, as shown in *Figure 2-1.*

The F field is the fractional portion of the represented number. In Normalized numbers (Section 2.1.1), the binary point is assumed to be immediately to the left of the most significant bit of the F field, with an implied 1 bit to the left of the binary point. Thus, the F field represents values from 1.0 (inclusive) to 2.0 (exclusive) as shown in Table 2-1.

TABLE 2-1. SAMPLE F FIELDS

F Field	Binary Value	Decimal Value
000...0	1.000...0	1.000...0
010...0	1.010...0	1.250...0
100...0	1.100...0	1.500...0
110...0	1.110...0	1.750...0

↑
Implied Bit

The E field is an unsigned number which gives the binary exponent of the represented number. The value in the E field is biased; that is, a constant bias value must be subtracted from the E field value in order to obtain the true exponent. The bias value is $011\ldots11_2$, which is either the value 127 (single precision) or 1023 (double precision). Thus, the true exponent can be either positive or negative, as shown in Table 2-2.

TABLE 2-2. SAMPLE E FIELDS

E Field	F Field	Represented Value
011...110	100...0	$1.5 \times 2^{-1} = 0.75$
011...111	100...0	$1.5 \times 2^{0} = 1.50$
100...000	100...0	$1.5 \times 2^{1} = 3.00$

Two forms of the E field represent special values, and are not available for use as exponents. 11...11 represents a value which is a reserved operand (Section 2.1.3). 00...00 represents the number zero if the F field is also all zeroes, otherwise the represented value is a reserved operand.

The S bit indicates the sign of the operand—0 for positive and 1 for negative. Floating-point numbers are in sign-magnitude form, such that only the S bit is complemented in order to change the sign of the represented number.

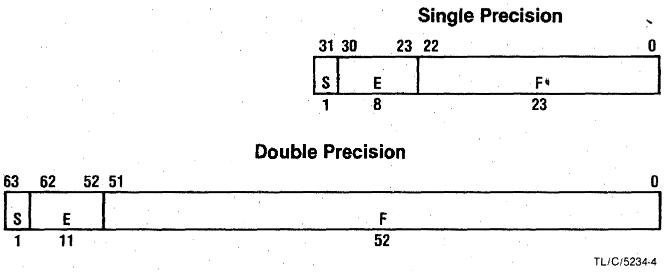


FIGURE 2-1. Floating-Point Operand Formats

2.1.1. Normalized Numbers

Normalized numbers are numbers which can be expressed as floating-point operands, as described above, where the E field is neither all zeroes nor all ones.

The value of a Normalized number can be derived by the formula:

$$(-1)^S \times 2^{(E\text{-Bias})} \times 1.F$$

The range of Normalized numbers is given in Table 2-3.

2.1.2. Zero

There are two representations for zero—positive and negative. Positive zero has all-zero F and E fields, and the S bit is zero. Negative zero also has all-zero F and E fields, but its S bit is one.

2.1.3. Reserved Operands

The proposed IEEE Standard for Binary Floating-Point Arithmetic (Task P754) provides for certain exceptional forms of floating-point operands. The NS16081 FPU treats these forms as reserved operands. The reserved operands are:

- Positive and negative infinity
- Not-a-Number (NaN) values
- Denormalized numbers

Both Infinity and NaN values have all ones in their E fields. Denormalized numbers have all zeroes in their E fields and non-zero values in their F fields.

The NS16081 FPU causes an Invalid Operation trap (Section 2.2.2.2) if it receives a reserved operand, unless the operation is simply a move (without conversion). The FPU does not generate reserved operands as results.

2.1.4. Integers

In addition to performing floating-point arithmetic, the NS16081 FPU performs conversions between integer and floating-point data types. Integers are accepted and generated by the FPU as two's complement values of byte (8 bits), word (16 bits) or double word (32 bits) length.

2.1.5. Memory Representations

The NS16081 FPU does not directly access memory. However, it is cooperatively involved in the execution of a set of two-address instructions with its NS16000 Family CPU. The CPU determines the representation of operands in memory.

In the NS16000 family of CPUs, operands are stored in memory with the least significant byte at the lowest byte address. The only exception to this rule is the Immediate addressing mode, where the operand is held (within the instruction format) with the most significant byte at the lowest address.

2.2. PROGRAMMING MODEL

The NS16000 architecture includes nine registers which are implemented on the NS16081 Floating-Point Unit (FPU).

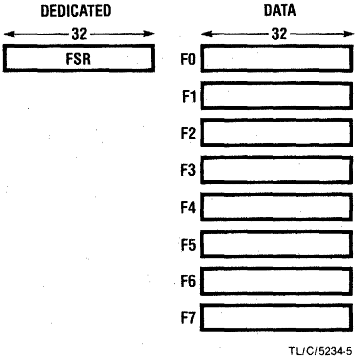


FIGURE 2-2. Register Set

2.2.1. Floating-Point Registers

There are eight registers (F0–F7) on the NS16081 FPU for providing high-speed access to floating-point operands. Each is 32 bits long. A floating-point register is referenced whenever a floating-point instruction uses the Register addressing mode (Section 2.3.2) for a floating-point operand. All other Register mode usages (i.e., integer operands) refer to the General Purpose Registers (R0–R7) on the CPU. When the Register addressing mode is specified for a double precision (64-bit) operand, a pair of registers holds the operand. The programmer must specify the even register of the pair. The even register contains the least significant half of the operand and the next consecutive register contains the most significant half.

TABLE 2-3. NORMALIZED NUMBER RANGES

	Single Precision	Double Precision
Most Positive	$2^{127} \times (2 - 2^{-23})$ $= 3.4028235 \times 10^{38}$	$2^{1023} \times (2 - 2^{-52})$ $= 1.797693134862316 \times 10^{308}$
Least Positive	2^{-126} $= 1.1754943 \times 10^{-38}$	2^{-1022} $= 2.225073858507201 \times 10^{-308}$
Least Negative	$-(2^{-126})$ $= -1.1754943 \times 10^{-38}$	$-(2^{-1022})$ $= -2.225073858507201 \times 10^{-308}$
Most Negative	$-2^{127} \times (2 - 2^{-23})$ $= -3.4028235 \times 10^{38}$	$-2^{1023} \times (2 - 2^{-52})$ $= -1.797693134862316 \times 10^{308}$

2.2.2. Floating-Point Status Register (FSR)

The Floating-Point Status Register (FSR) selects operating modes and records any exceptional conditions encountered during execution of a floating-point operation. *Figure 2-3* shows the format of the FSR.

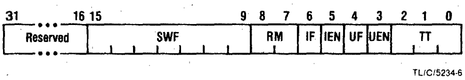


FIGURE 2-3. The Floating-Point Status Register

2.2.2.1. FSR Mode Control Fields

The FSR mode control fields select FPU operation modes. The meanings of the FSR mode control bits are given below.

Rounding Mode (RM): Bits 7 and 8. This field selects the rounding method. Floating-point results are rounded whenever they cannot be exactly represented. The rounding modes are:

00 Round to nearest value. The value which is nearest to the exact result is returned. If the result is exactly halfway between the two nearest values the even value (LSB = 0) is returned.

01 Round toward zero. The nearest value which is closer to zero or equal to the exact result is returned.

10 Round toward positive infinity. The nearest value which is greater than or equal to the exact result is returned.

11 Round toward negative infinity. The nearest value which is less than or equal to the exact result is returned.

Underflow Trap Enable (UEN): Bit 3. If this bit is set, the FPU requests a trap whenever a result is too small in absolute value to be represented as a Normalized number. If it is not set, any underflow condition returns a result of exactly zero.

Inexact Result Trap Enable (IEN): Bit 5. If this bit is set, the FPU requests a trap whenever the result of an operation cannot be represented exactly in the operand format of the destination. If it is not set, the result is rounded according to the selected rounding mode.

2.2.2.2. FSR Status Fields

The FSR Status Fields record exceptional conditions encountered during floating-point data processing. The meanings of the FSR status bits are given below:

Trap Type (TT): Bits 0–2. This 3-bit field indicates the reason for any trap requested by the FPU. It is cleared by writing zeroes into it with the Load FSR instruction or by a hardware reset.

000 No trap requested.

001 Underflow. If the UEN bit is set, this trap occurs whenever a result is too close to zero to be represented as a Normalized number.

010 Overflow. This trap occurs whenever a result is too large in absolute value to be represented.

011 Divide by zero. An attempt was made to divide a non-zero value by zero.

100 Illegal instruction. An undefined Floating-Point instruction was passed to the FPU.

101 Invalid operation. This trap occurs if either:
1. A Reserved operand is used as a floating-point operand by any instruction except MOVf (move without conversion), or
2. Both operands of the DIVf (Divide) instruction are zero.

110 Inexact Result. If the IEN bit is set, this trap occurs whenever the result of an operation cannot be represented exactly in the operand format of the destination. It occurs only if no other error has occurred.

Underflow Flag (UF): Bit 4. This bit is set by the FPU whenever a result is too small in absolute value to be represented as a Normalized number. Its function is not affected by the state of the UEN bit. The UF bit is cleared only by writing a zero into it with the Load FSR instruction or by a hardware reset.

Inexact Result Flag (IF): Bit 6. This bit is set by the FPU whenever the result of an operation must be rounded to fit within the destination format. This situation applies both to floating-point and integer destinations. The IF bit is set only if no other error has occurred. It is cleared only by writing a zero into it with the Load FSR instruction or by a hardware reset.

2.2.2.3. FSR Software Field (SWF)

Bits 9–15 of the FSR hold and display any information written to them (using the LFSR and SFSR instructions), but are not otherwise used by FPU hardware. They are reserved for use with NSC floating-point extension software.

2.3. INSTRUCTION SET

2.3.1. General Instruction Format

Figure 2-4 shows the general format of an NS16000 instruction. The Basic Instruction is one to three bytes long and contains the opcode and up to two 5-bit General Addressing Mode (Gen) fields. Following the Basic Instruction field is a set of optional extensions, which may appear depending on the instruction and the addressing modes selected.

The only form of extension issued to the NS16081 FPU is an Immediate operand. Other extensions are used only by the CPU to reference memory operands needed by the FPU.

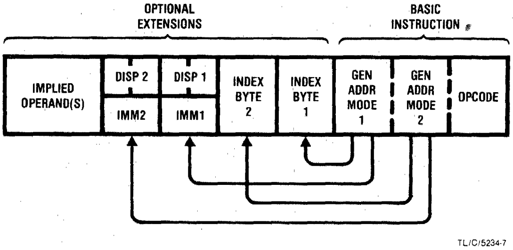


FIGURE 2-4. General Instruction Format

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 2-5.*

Following Index Bytes come any displacements (addressing constants) or immediate values associated with the selected addressing modes. Each Disp/Imm field may contain one or 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 2-6,* 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.

Some non-FPU 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.

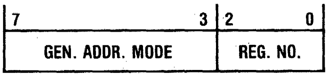
2.3.2. Addressing Modes

The NS16000 Family CPUs generally access 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 NS16000 family 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 which acts upon that variable. Extraneous data movement is therefore minimized.

NS16000 Addressing Modes fall into nine basic types:

Register: In floating-point instructions, these addressing modes refer to a Floating-Point Register (F0–F7) if the operand is of a floating-point type. Otherwise, a CPU General Purpose Register (R0–R7) is referenced. See Section 2.2.1.



TL/C/5234-8

FIGURE 2-5. Index Byte Format

Register Relative: A CPU 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 CPU 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 CPU SP, SB or FP register. A displacement is added to that pointer to generate the Effective Address of the operand.

Immediate: The operand is encoded within the instruction. This addressing mode is not allowed if the operand is to be written. Floating-point operands as well as integer operands may be specified using Immediate mode.

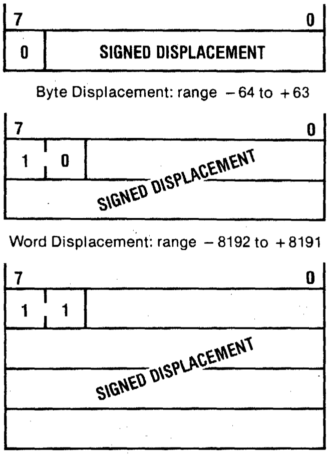
Absolute: The address of the operand is specified by a Displacement field in the instruction.

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

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

The following table, Table 2-4, is a brief summary of the addressing modes. For a complete description of their actions, see the NS16000 Family Programmer's Reference Manual.



Byte Displacement: range −64 to +63

Word Displacement: range −8192 to +8191

Double Word Displacement: range (entire addressing space)

TL/C/5234-9

FIGURE 2-6. Displacement Encodings

TABLE 2-4. NS16000 FAMILY ADDRESSING MODES

ENCODING	MODE	ASSEMBLER SYNTAX	EFFECTIVE ADDRESS
Register			
00000	Register 0	R0 or F0	None: Operand is in the specified register.
00001	Register 1	R1 or F1	
00010	Register 2	R2 or F2	
00011	Register 3	R3 or F3	
00100	Register 4	R4 or F4	
00101	Register 5	R5 or F5	
00110	Register 6	R6 or F6	
00111	Register 7	R7 or F7	
Register Relative			
01000	Register 0 relative	disp(R0)	Disp + Register.
01001	Register 1 relative	disp(R1)	
01010	Register 2 relative	disp(R2)	
01011	Register 3 relative	disp(R3)	
01100	Register 4 relative	disp(R4)	
01101	Register 5 relative	disp(R5)	
01110	Register 6 relative	disp(R6)	
01111	Register 7 relative	disp(R7)	
Memory 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	
Memory Relative			
10000	Frame memory relative	disp2(disp1(FP))	Disp2 + Pointer; Pointer found at address Disp1 + Register. "SP" is either SP0 or SP1, as selected in PSR.
10001	Stack memory relative	disp2(disp1(SP))	
10010	Static memory relative	disp2(disp1(SB))	
Immediate			
10100	Immediate	value	None: Operand is issued from CPU 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.
Scaled Index			
11100	Index, bytes	mode[Rn:B]	Mode + Rn.
11101	Index, words	mode[Rn:W]	Mode + 2 × Rn.
11110	Index, double words	mode[Rn:D]	Mode + 4 × Rn.
11111	Index, quad words	mode[Rn:Q]	Mode + 8 × Rn. "Mode" and "n" are contained within the Index Byte.
10011	(Reserved for Future Use)		

2.3.3. Floating-Point Instruction Set

The NS16081 FPU instructions occupy formats 9 and 11 of the NS16000 Family instruction set *(Figure 2-7)*. A list of all NS16000 family instruction formats is found in the applicable CPU data sheet.

Certain notations in the following instruction description tables serve to relate the assembly language form of each instruction to its binary format in *Figure 2-7*.

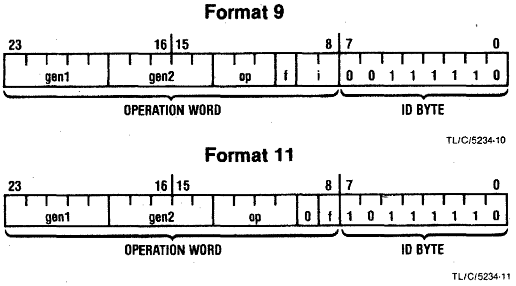


FIGURE 2-7. Floating-Point Instruction Formats

The Format column indicates which of the two formats in *Figure 2-7* represents each instruction.

The Op column indicates the binary pattern for the field called "op" in the applicable format.

The Instruction column gives the form of each instruction as it appears in assembly language. The form consists of an instruction mnemonic in upper case, with one or more suffixes (i or f) indicating data types, followed by a list of operands (gen1, gen2).

An i suffix on an instruction mnemonic indicates a choice of integer data types. This choice affects the binary pattern in the i field of the corresponding instruction format *(Figure 2-7)* as follows:

Suffix i	Data Type	i Field
B	Byte	00
W	Word	01
D	Double Word	11

An f suffix on an instruction mnemonic indicates a choice of floating-point data types. This choice affects the setting of the f bit of the corresponding instruction format *(Figure 2-7)* as follows:

Suffix f	Data Type	f Bit
F	Single Precision	1
L	Double Precision (Long)	0

An operand designation (gen1, gen2) indicates a choice of addressing mode expressions. This choice affects the binary pattern in the corresponding gen1 or gen2 field of the instruction format *(Figure 2-7)*. Refer to Table 2-4 for the options available and their patterns.

Further details of the exact operations performed by each instruction are found in the NS16000 Family Programmer's Reference Manual.

Movement and Conversion

The following instructions move the gen1 operand to the gen2 operand, leaving the gen1 operand intact.

Format	Op	Instruction		Description
11	0001	MOVf	gen1, gen2	Move without conversion.
9	010	MOVLF	gen1, gen2	Move, converting from double precision to single precision.
9	011	MOVFL	gen1, gen2	Move, converting from single precision to double precision.
9	000	MOVif	gen1, gen2	Move, converting from any integer type to any floating-point type.
9	100	ROUNDfi	gen1, gen2	Move, converting from floating-point to the nearest integer.
9	101	TRUNCfi	gen1, gen2	Move, converting from floating-point to the nearest integer closer to zero.
9	111	FLOORfi	gen1, gen2	Move, converting from floating-point to the largest integer less than or equal to its value.

Note: The MOVLF instruction f bit must be 1 and the i field must be 10.
The MOVFL instruction f bit must be 0 and the i field must be 11.

Arithmetic Operations

The following instructions perform floating-point arithmetic operations on the gen1 and gen2 operands, leaving the result in the gen2 operand.

Format	Op	Instruction		Description
11	0000	ADDf	gen1, gen2	Add gen1 to gen2.
11	0100	SUBf	gen1, gen2	Subtract gen1 from gen2.
11	1100	MULf	gen1, gen2	Multiply gen2 by gen1.
11	1000	DIVf	gen1, gen2	Divide gen2 by gen1.
11	0101	NEGf	gen1, gen2	Move negative of gen1 to gen2.
11	1101	ABSf	gen1, gen2	Move absolute value of gen1 to gen2.

Comparison

The Compare instruction compares two floating-point values, sending the result to the CPU PSR Z and N bits for use as condition codes. The Z bit is set if the gen1 and gen2 operands are equal; it is cleared otherwise. The N bit is set if the gen1 operand is greater than the gen2 operand; it is cleared otherwise. The CPU PSR L bit is unconditionally cleared. Positive and negative zero are considered equal.

Format	Op	Instruction		Description
11	0010	CMPf	gen1, gen2	Compare gen1 to gen2.

Floating-Point Status Register Access

The following instructions load and store the FSR as a 32-bit integer.

Format	Op	Instruction		Description
9	001	LFSR	gen1	Load FSR
9	110	SFSR	gen2	Store FSR

2.4. TRAPS

Upon detecting an exceptional condition in executing a floating-point instruction, the NS16081 FPU requests a trap by setting the Q bit of the status word transferred during the slave protocol (Section 3.5). The CPU responds by performing a trap using a default vector value of 3. See the Programmer's Reference Manual and the applicable CPU data sheet for trap service details.

A trapped floating-point instruction returns no result, and does not affect the CPU Processor Status Register (PSR). The FPU displays the reason for the trap in the Trap Type (TT) field of the FSR (Section 2.2.2.2).

3. Functional Description

3.1. POWER AND GROUNDING

The NS16081 requires a single 5V power supply, applied on pin 24 (V_{CC}). See DC Electrical Characteristics table.

Grounding connections are made on two pins. Logic Ground (GNDL, pin 12) is the common pin for on-chip logic, and Buffer Ground (GNDB, pin 13) is the common pin for the output drivers. For optimal noise immunity, it is recommended that GNDL be attached through a single conductor directly to GNDB, and that all other grounding connections be made only to GNDB, as shown below *(Figure 3-1).*

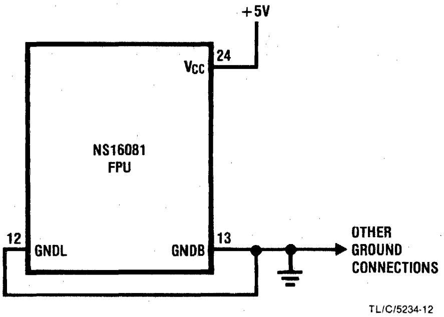


FIGURE 3-1. Recommended Supply Connections

3.2. CLOCKING

The NS16081 FPU requires a single-phase TTL clock input on its CLK pin (pin 14). The CLK signal is asynchronous to bus transfers and can come from any source, but the CTTL signal, provided by the NS16201 Timing Control Unit, can be used for this purpose.

3.3. RESETTING

The $\overline{\text{RST}}$ pin serves as a reset for on-chip logic. The FPU may be reset at any time by pulling the $\overline{\text{RST}}$ pin low for at least 64 clock cycles. Upon detecting a reset, the FPU terminates instruction processing, resets its internal logic, and clears the FSR to all zeroes.

On application of power, $\overline{\text{RST}}$ must be held low for at least 50 μs after V_{CC} is stable. This ensures that all on-chip voltages are completely stable before operation. See *Figures 3-2 and 3-3.*

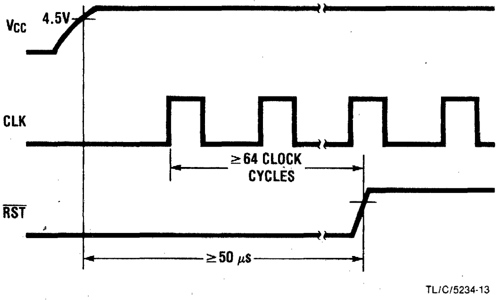


FIGURE 3-2. Power-On Reset Requirements

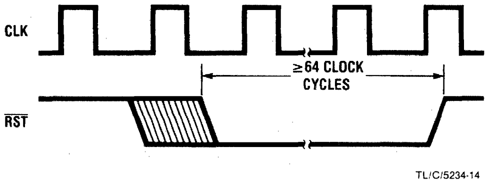


FIGURE 3-3. General Reset Timing

3.4. BUS OPERATION

Instructions and operands are passed to the NS16081 FPU with slave processor bus cycles. Each bus cycle transfers either one byte (8 bits) or one word (16 bits) to or from the FPU. During all bus cycles, the $\overline{\text{SPC}}$ line is driven by the CPU as an active low data strobe, and the FPU monitors pins ST0 and ST1 to keep track of the sequence (protocol) established for the instruction being executed. This is especially necessary in a virtual memory environment, allowing the FPU to retry an aborted instruction.

3.4.1. Bus Cycles

A bus cycle is initiated by the CPU, which asserts the proper status on ST0 and ST1 and pulses $\overline{\text{SPC}}$ low. ST0 and ST1 are sampled by the FPU on the leading (falling) edge of the $\overline{\text{SPC}}$ pulse. If the transfer is from the FPU (a slave processor read cycle), the FPU asserts data on the data bus for the duration of the $\overline{\text{SPC}}$ pulse. If the transfer is to the FPU (a slave processor write cycle), the FPU latches data from the data bus on the trailing (rising) edge of the $\overline{\text{SPC}}$ pulse. *Figures 3-4 and 3-5* illustrate these sequences.

The direction of the transfer and the role of the bidirectional $\overline{\text{SPC}}$ line are determined by the instruction protocol being performed. $\overline{\text{SPC}}$ is always driven by the CPU during slave processor bus cycles. Protocol sequences for each instruction are given in Section 3.5.

3.4.2. Operand Transfer Sequences

An operand is transferred in one or more bus cycles. A 1-byte operand is transferred on the least significant byte of the data bus (D0–D7). A 2-byte operand is transferred on the entire bus. A 4-byte or 8-byte operand is transferred in consecutive bus cycles, least significant word first.

3.5. INSTRUCTION PROTOCOLS

3.5.1. General Protocol Sequence

Slave Processor instructions have a three-byte Basic Instruction field, consisting of an ID Byte followed by an Operation Word. See *Figure 2-7* for FPU instruction encodings. The ID Byte has three functions:

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

2) It specifies which Slave Processor will execute it.

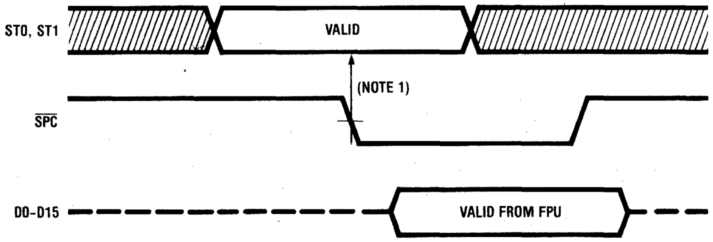
3) It determines the format of the following Operation Word of the instruction.

Upon receiving a Slave Processor instruction, the CPU initiates the sequence outlined in Table 3-2. While applying Status Code 11 (Broadcast ID, Table 3-1), the CPU transfers the ID Byte on the least significant half of the Data Bus (D0-D7). 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 01 (Transfer Slave Operand, Table 3-1). 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 D8–D15 respectively, and bits 8–15 appear on pins D0–D7 respectively.

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 01 (Transfer Slave Processor Operand, Table 3-1).

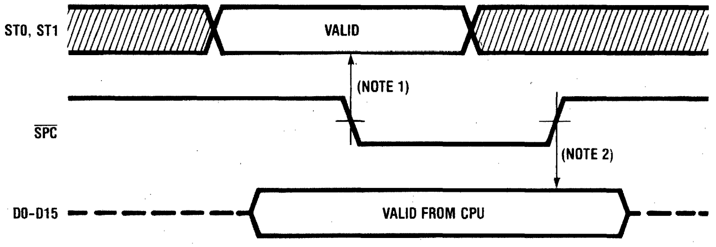
After the CPU has issued the last operand, the Slave Processor starts the actual execution of the instruction. Upon completion, it will signal the CPU by pulsing $\overline{\text{SPC}}$ low. To allow for this, $\overline{\text{SPC}}$ is normally held high only by a pull-up device of approximately 5 kΩ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



TL/C/5234-15

Note 1: FPU samples CPU status here.

FIGURE 3-4. Slave Processor Read Cycle



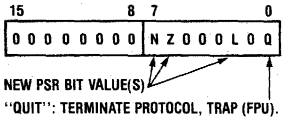
TL/C/5234-16

Note 1: FPU samples CPU status here.

Note 2: FPU samples data bus here.

FIGURE 3-5. Slave Processor Write Cycle

Upon receiving the pulse on $\overline{\text{SPC}}$, the CPU uses $\overline{\text{SPC}}$ to read a Status Word from the Slave Processor, applying Status Code 10 (Read Slave Status, Table 3-1). This word has the format shown in *Figure 3-6*. If the Q bit ("Quit", Bit 0) is set, this indicates that an error has been detected by the Slave Processor. The CPU will not continue the protocol, but will immediately trap through the FPU vector in the Interrupt Table. If the instruction being performed is CMPf (Section 2.3.3) and the Q bit is not set, the CPU loads Processor Status Register (PSR) bits N, Z and L from the corresponding bits in the Status Word. The NS16081 FPU always sets the L bit to zero.



TL/C/5234-17

FIGURE 3-6. FPU Protocol Status Word Format

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 01 (Transfer Slave Operand, Table 3-1).

TABLE 3-1. BUS STATUS COMBINATIONS

ST1	ST0	CPU Function
0	0	(Reserved)
0	1	Transferring Operation Word or Operand
1	0	Reading Status Word
1	1	Broadcasting ID Byte

TABLE 3-2. GENERAL INSTRUCTION PROTOCOL

Step	Status	Action
1	11	CPU sends ID Byte.
2	01	CPU sends Operation Word.
3	01	CPU sends required operands.
4	XX	FPU starts execution.
5	XX	FPU pulses $\overline{\text{SPC}}$ low.
6	10	CPU reads Status Word.
7	01	CPU reads result (if any).

3.5.2. Floating-Point Protocols

Table 3-3 gives the protocols followed for each floating-point instruction. The instructions are referenced by their mnemonics. For the bit encodings of each instruction, see Section 2.3.3.

The Operand Class columns give the Access Classes for each general operand, defining how the addressing modes are interpreted by the CPU (see NS16000 Family Programmer's 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 3-6)*.

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.

TABLE 3-3. 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

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.

4. AC Electrical Characteristics

4.1. OUTPUT SIGNAL PROPAGATION DELAYS

Maximum times assume capacitive loading of 100 pF.

Name	Figure	Description	Min	Typ	Max	Units
t_{Dv}	4-5	$\overline{\text{SPC}}$ Low to Data Valid	0		65	ns
t_{Df}	4-5	$\overline{\text{SPC}}$ High to Data Bus Invalid (Floating)	0		50	ns
t_{SPCFw}	4-6	$\overline{\text{SPC}}$ Pulse Width from FPU (at 0.8V)	$t_{CLKp}-50$		$t_{CLKp}+50$	ns
t_{SPCFl}	4-6	CLK High to $\overline{\text{SPC}}$ Low from FPU	0		120	ns
t_{SPCFh}	4-6	CLK High to $\overline{\text{SPC}}$ High from FPU	0		120	ns
t_{SPCFnf}	4-6	CLK Low to FPU not Forcing $\overline{\text{SPC}}$ High	0		75	ns

4.2. INPUT SIGNAL REQUIREMENTS

Name	Figure	Description	Min	Typ	Max	Units
t_{PWR}	4-2	Power-On Reset Duration	50			μs
t_{RSTw}	4-3	Reset Pulse Width (at 0.8V)	64			t_{CLKp}
t_{Ss}	4-4	Status Set-Up to $\overline{\text{SPC}}$ Low	75			ns
t_{Sh}	4-4	Status Hold from $\overline{\text{SPC}}$ Low	100			ns
t_{Ds}	4-4	Data Set-Up to $\overline{\text{SPC}}$ High	75			ns
t_{Dh}	4-4	Data Hold from $\overline{\text{SPC}}$ High	100			ns
t_{SPCw}	4-4	$\overline{\text{SPC}}$ Pulse Width from CPU (at 0.8V)	100			ns

4.3. CLOCKING REQUIREMENTS

Name	Figure	Description	Min	Typ	Max	Units
t_{CLKh}	4-1	Clock High Time (above 2.0V)	60		1000	ns
t_{CLKl}	4-1	Clock Low Time (below 0.8V)	60		1000	ns
t_{CLKp}	4-1	Clock Period	160		2000	ns

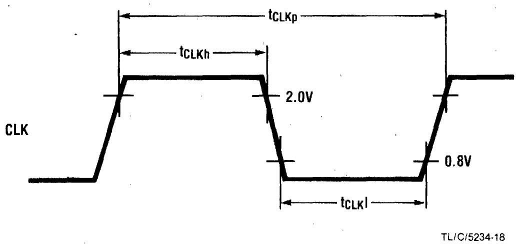


FIGURE 4-1. Clock Timing

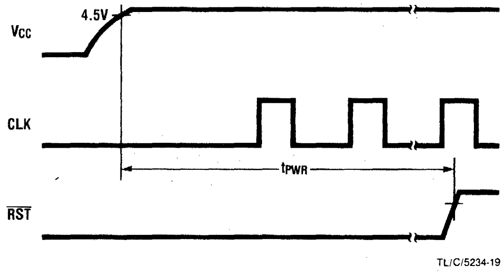
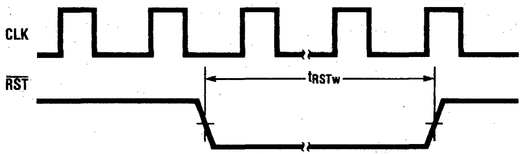
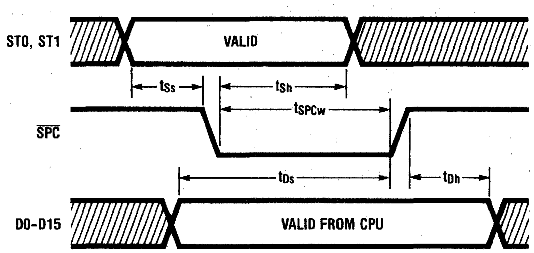


FIGURE 4-2. Power-On Reset



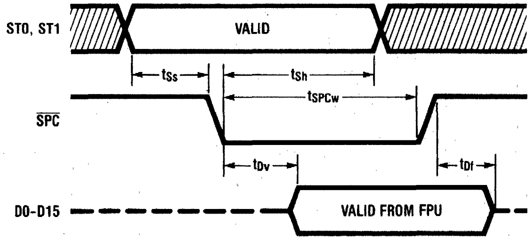
TL/C/5234-20

FIGURE 4-3. Non-Power-On Reset



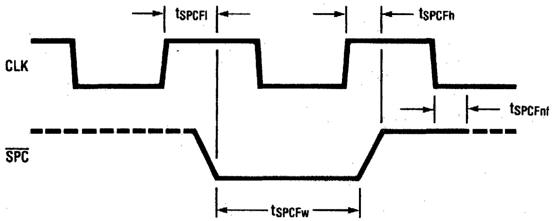
TL/C/5234-21

FIGURE 4-4. Write Cycle to FPU



TL/C/5234-22

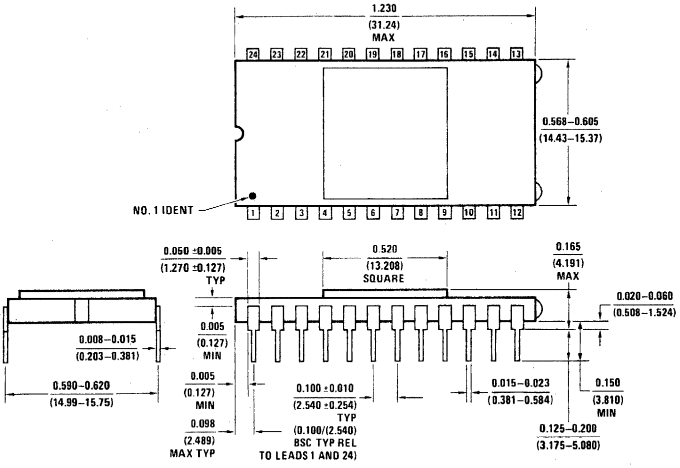
FIGURE 4-5. Read Cycle from FPU



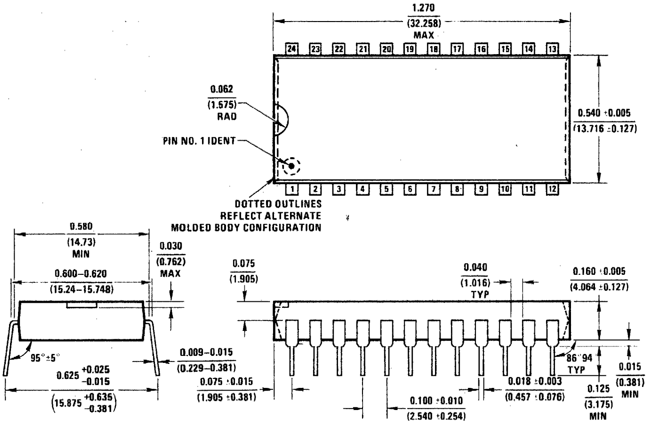
TL/C/5234-23

FIGURE 4-6. $\overline{\text{SPC}}$ Pulse from FPU

Physical Dimensions inches (millimeters)



Ceramic Dual-In-Line Package (D)
Order Number NS16081D-6
NS Package Number D24C



Molded Dual-in-Line Package (N)
Order Number NS16081N-6
NS Package Number N24A

National Semiconductor

PRELIMINARY

NS16082 Memory Management Unit (MMU)

General Description

The NS16082 Memory Management Unit (MMU) provides hardware support for demand-paged virtual memory management. Its specific capabilities include fast dynamic address translation, protection on individual 512-byte pages, and detailed status to assist an operating system in efficiently managing up to 16MB of physical memory. Support for virtual machine implementations is provided, along with comprehensive software debugging features.

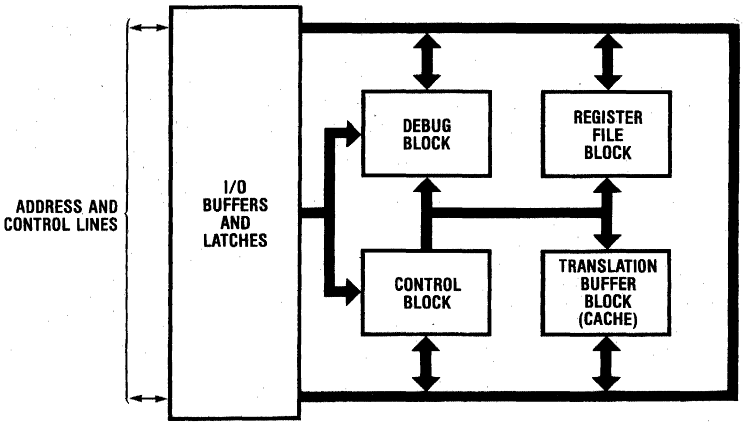
High-speed address translation is performed on-chip through a 32-entry associative cache, which maintains itself from tables in memory with no software intervention. Protection violations and page faults (references to nonresident pages) are automatically detected by the MMU, invoking the instruction abort with retry feature of the CPU. This fault handling mechanism provides the necessary hooks for virtual memory and virtual machines.

Additional features for program debugging include two hardware breakpoint registers and a program flow traceback facility, which provide the programmer with powerful stand-alone debugging capability even without expensive test equipment.

Features

- Dynamic address translation
- 32-entry on-chip translation cache, updated automatically from page tables in memory
- Full hardware support for virtual memory and virtual machines
- Security mechanisms implemented via access level checking and dual-space mapping
- Program debugging support:
  - — Two breakpoint registers
  - — "Break on Branch" mode
  - — Program flow traceback
- High-speed XMOS technology
- 48-pin dual-in-line package
- Single 5V supply

NS16082 MMU Block Diagram



Absolute Maximum Ratings

Temperature Under Bias	0°C to 70°C
Storage Temperature	−65°C to 150°C
All Input or Output Voltages with respect to GND	−0.5V to +7.0V
Power Dissipation	1.5Watt

Note: Absolute maximum ratings are those values beyond which the safety of the device cannot be guaranteed. Continuous operation at these limits is not intended. Therefore, operation should be limited to those conditions specified under DC Electrical Characteristics (see below).

DC Electrical Characteristics $V_{CC} = 5V \pm 5\%$

Symbol	Parameter	Test	Min.	Max.	Unit
V_{IL}	Input Low Voltage		−0.5	+0.8	V
V_{IH}	Input High Voltage		2.0	$V_{CC}+0.5$	V
V_{OL}	Output Low Voltage	$I_{OL}=2.0mA$		0.45	V
V_{OH}	Output High Voltage	$I_{OH}=-400\mu A$	2.4		V
I_{CC}	Power Supply Current	$T_A=0°C$		300	mA
I_{IL}	Input Leakage Current	$0 < V_{IN} < V_{CC}$		10	μA
I_{OL}	Output Leakage Current	$0.45 < V_O < V_{CC}$		10	μA

NS16082 MMU Bit Maps

(Reserved)	NT	UT	FT	AI	UB	BEN	AO	DS	TS	TU	BST	EST	BD	ED	☒	BN	TET	ERC
31		24	23							16	15			8	7			0

Memory Management Status Register (MSR)

M	(Reserved)	Address Bits 10-23	0	0	0	0	0	0	0	0	0	0
31	24	23 10	9									0

Page Table Base Registers (PTB0, PTB1)

AS	(Reserved)	Virtual Address
31	24	23 0

Error/Invalidate Address Register (EIA)
Program Flow Registers (PF0, PF1)

AS	VP	BE	BR	BW	CE	☒	☒	Address
31							24	23 0

Breakpoint Registers (BPR0, BPR1)

0	0	0	0	0	0	0	0	Count
31							24	23 0

Breakpoint Counter Register (BCNT)

SC1	SC0
31 16	15 0

Sequential Count Register (SC)

BS	(Reserved)	Page Frame Number	(Reserved)	M	R	PL	V
31	24	23 9	8 5	4			0

Page Table Entry (PTE) In Memory

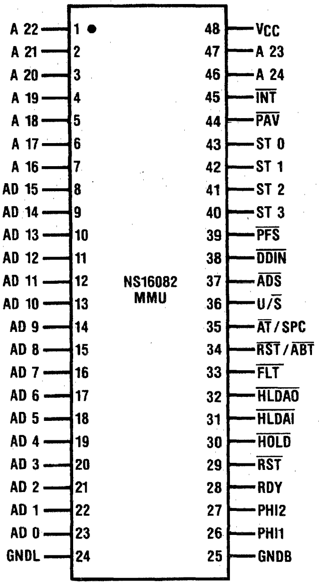
gen	short	0	opcode	d	0	0	0	1	1	1	1	0
23	Operation Word			8	7			ID Code				0

Slave Instruction Format

Index 1	Index 2	Offset
23 16	15 9	8 0

Virtual Address Format

1 Pin Descriptions



Pin Configuration

1 Pin Descriptions

The following is a brief description of all NS16082 pins. The descriptions reference portions of the functional descriptions, Section 3.

1.1 Supplies

Power (V_{CC}): +4.75 to 5.25 V DC supply. (Sec. 3.1)

Logical Ground (GNDL): Internal Logic Common Ground. (Sec. 3.1)

Buffer Ground (GNDB): Signal Ground and Power Supply return. (Sec. 3.1)

1.2 Input Signals

Clock (PHI1, PHI2): Two-phase clocking signals from 0.5 to 10 MHz MOS Level clocks from NS16201 clock generator. See NS16032.

Ready (RDY): Used by slow memories to extend the memory cycle for more than four clock periods. It is synchronized externally and sampled at the beginning of T3.

Hold Request ($\overline{\text{HOLD}}$), ($\overline{\text{HLDAI}}$), ($\overline{\text{HLDAO}}$): Hold/Hold Acknowledge Input/Hold Acknowledge Output—three pins for active low signals used in DMA transfers. A DMA device requests the bus pulling the $\overline{\text{HOLD}}$ line low. $\overline{\text{HLDAI}}$ is connected to the $\overline{\text{HLDA}}$ output line of the CPU and $\overline{\text{HLDAO}}$ is output to the DMA device.

Reset ($\overline{\text{RST}}$): System reset, active low.

Status Lines (ST0-3): Status lines output by the CPU. Active from T4 of previous bus transfer through T3 of present bus transfer. See NS16032 Data Sheet for assignments.

Program Flow Status ($\overline{\text{PFS}}$): Active low pulse issued by the CPU at the beginning of each instruction.

User/Supervisor Modes (U/$\overline{\text{S}}$): This signal is provided by the CPU, from its U-bit in Program Status Register (PSR). It is used by the MMU for memory protection and for selection of the Address Space (in Dual Space mode only).

Address Strobe Input ($\overline{\text{ADS}}$): Active low pulse received from the CPU during T1. Used as a strobe to latch the virtual address from the multiplexed bus.

1.3 Output Signals

Reset Output or Abort ($\overline{\text{RST}}$/$\overline{\text{ABT}}$): Active low pulse accepted by the CPU during TMMU or T2. When active, the CPU is interrupted. The MMU status register (MSR) holds the information about the cause of the abort. When held longer than one clock cycle it is interpreted by the CPU as a reset signal. Upon receiving the $\overline{\text{RST}}$ signal on its own pin, the MMU will activate the $\overline{\text{RST}}$/$\overline{\text{ABT}}$ pin to reset the CPU.

Interrupt Output ($\overline{\text{INT}}$): Active low pulse used by the debug functions to inform the CPU (when connected to its $\overline{\text{NMI}}$ input) or external hardware that a break condition has occurred. The MMU status register (MSR) holds the information about the cause of the interrupt.

Float Output ($\overline{\text{FLT}}$): An active low signal that floats the CPU from the bus, when the MMU has to get Page Table Entries from memory. It is sampled by the CPU during TMMU. It is held high during reset.

Physical Address Valid ($\overline{\text{PAV}}$): Active low pulse generated during TMMU. Used as a strobe pulse by the memory address latches. It floats during HOLD ACKNOWLEDGE and it is also pulsed when $\overline{\text{FLT}}$ is active to access page tables.

Most significant bit of Physical Address (A24): Valid from TMMU to T4.

Hold Acknowledge Output ($\overline{\text{HLDAO}}$): See Hold Request. (Sec. 1.2)

1.4 Input-Output Signals

Multiplexed Address/Data Bus (AD0-AD15): During clock period T1, contains the Virtual Address (output by the CPU). During period TMMU contains the Physical Address (output by the MMU). During periods T2–T4 contains Data (output by the CPU, memory or MMU).

Virtual/Physical Address Multiplexed Bus (A16-A23): High-order bits of address. During T1, the bus contains the Virtual Address (output by the CPU). During TMMU-T4 it contains the Physical Address (output by the MMU).

Data Direction In ($\overline{\text{DDIN}}$): As an input indicates to the MMU the type of memory cycle: Low for Read and high for Write. When the CPU tri-states this line, the line is driven by the MMU allowing it to read/write in the memory pages independently of the CPU.

Address Translation or Slave Processor Control ($\overline{\text{AT}}$/SPC): A bidirectional control line used in slave instructions. For a description of the characteristics of this line refer to the NS16032 CPU Data Sheet. During reset it is held low by the MMU to set the CPU for the Address Translation mode.

2 General Description

For purposes of address translation, memory is divided into 512-byte pages. A virtual address for the MMU is composed of two fields: a virtual page frame number and a 9-bit offset. The offset is unchanged by the translation algorithm. The MMU translates the virtual page number to a physical page number, according to page tables stored in memory.

The Operating System and MMU exchange information on the status of the memory pages through a Page Table in physical memory. The entries track both the presence of a page in the physical memory and the protection level of that page.

By manipulating the page tables, an Operating System dynamically controls the mapping of virtual to physical addresses. In particular, the Operating System may specify that references to certain pages should generate translation error aborts. This mechanism implements virtual memory management and protection.

The virtual address output from the NS16032 CPU is 24 bits wide while the physical address output from the MMU is 25 bits wide. The extra bit (bit 25) can partition memory, making it especially useful in an In-System Emulation (ISE™) environment.

The MMU has an internal cache memory which contains direct virtual-to-physical address mappings of the 32 most recently used pages. Thus, most address translations take only one additional clock cycle. The "hit rate" of the cache memory is usually better than 98%, so that the time overhead involved in dynamic translation is minimal.

The MMU is also capable of debugging support. The MMU's ability to perform program flow tracing and address breakpoints aids debugging.

Program flow tracing allows software to reconstruct the sequence of instructions executed prior to an exception or a breakpoint. The address of a nonsequential instruction is stored and a count is kept of the number of instructions following until the next nonsequential instruction is reached. The MMU enables retracing of two such branchings with no effect on execution time. By enabling a special "Break on Nonsequential Fetch" feature, a table of arbitrary length can be maintained.

Up to two breakpoint addresses, virtual or physical, may be activated in the MMU. A counter may be attached to one of these, enabling "break on N occurances" capability.

2.1 Internal Organization

Internal organization of the NS16032 MMU consists of five functional blocks and their respective addressable registers. These are shown in *Figure 2-1*. Both internal and external MMU connections are shown in the block diagram. Detailed block and register operation is described in the following subsections.

2.1.1 Hardware Debug Block

This block contains the registers, counters, and logic which allow the execution of program breakpoints. The circuits also permit program flow tracing, which, in turn, enables the software to reconstruct the sequence of instructions executed before breakpoint.

Program flow tracing information is recorded in the two 24-bit Program Flow registers (PF0 and PF1). The 32-bit Sequential Counter (SC) contains the number of sequentially executed instructions following the last two program flow changes. The PF registers store the virtual addresses of the last two nonsequentially executed instructions. If such an address is obtained, it is entered into PF0. The old PF0 contents are shifted into PF1; the previous contents of PF1 are lost. The corresponding counts in the halves of SC are transferred similarly. For the user these are read-only registers read by means of the Store Memory Management Register (SMR) slave instruction. A Load Memory Management Register (LMR) instruction addressing any of the PF registers will reset the PF and SC registers.

The debug block includes the following registers. Each is described below. Particular emphasis is placed on the the MMU Status Register (MSR).

- MMU Status Register (MSR)
- Program Flow Registers (PF0 and PF1)
- Sequential Counter (SC)
- Breakpoint Registers (BPR0 and BPR1)
- Breakpoint Counter Register (BCNT)

Memory Management Status Register

The Memory Management Status Register (MSR) specifies the operational mode and current processing status of the MMU. The register permits user control of address translation, breakpoints, and program tracing. The MMU Status Register is 32 bits in length. The MSR format is shown on page 2.

Bits 0 to 25 are the various control bits and flags of the MMU. Bits 26 to 31 are not used. The following describes the control bits and flags:

ERC is the Error Class flag. The 3-bit flag specifies the cause of the current MMU exception.

Bit 0 is set to 1 on an address translation error.

Bit 1 is not used.

Bit 2 is set to 1 on a break and set to 0 on a nonsequential trace interrupt.

TET is the Translation Error Trace flag. The 3-bit flag specifies the cause of the current address translation error.

Bit 3 is set to 1 on a protection level error.

Bit 4 is set to 1 on an invalid Level 1 Page Table Entry.

Bit 5 is set to 1 on an invalid Level 2 Page Table Entry.

BN is the Breakpoint Number bit. BN is set to indicate the breakpoint address of the current break. If BN is 1, the breakpoint address is contained in BPR1. If BN is 0, the breakpoint address is in BPR0.

ED is the error Data Direction bit. If ED is 1, a read operation or the first part of a read-modify-write operation caused an address translation error. If

ED is 0, a write or the last part of a read-modify-write operation caused the error.

BD is the Breakpoint Direction bit. If BD is 1, a read operation or the first part of a read-modify-write operation caused the current break. If BD is 0, a write operation on the last part of a read-modify-write operation caused the break.

EST is the Error Status flag. The 3-bit flag is set on an address translation error to the low order three bits of the system status bus. (See CPU Hardware Specs.)

BST is the Breakpoint Status flag. The 3-bit flag is set on a break to the low order three bits of the system status bus. (See CPU Hardware Specs.)

TU is the Translate User bit. If TU is 1, the MMU translates all adddresses specified in the User mode. If TU is 0, the MMU interprets addresses specified in the User mode as physical address.

TS is the Translate ·Supervisor bit. If TS is 1, the MMU translates all addresses specified in the Supervisor mode. If TS is 0, the MMU interprets addresses specified in the Supervisor mode as physical addresses.

DS is the Dual Space bit. If DS is 1, the PTB1 register contains the Level 1 Page Table Base address of all addresses specified in the User mode. If DS is 0, the PTB0 register contains the Level 1 Page Table Base address of all addresses specified in both User and Supervisor modes.

AO is the Access Override bit. If AO is 1, the MMU overrides the protection level of all addresses. This permits a program to access memory which is normally accessible only to the supervisor while the system is in the User mode. If A0 is 0, the MMU does not override protection level.

BEN is the Breakpoint Enable bit. If BEN is 1, the MMU enables the BPR0 and BPR1 registers and breaks program execution whenever a breakpoint is encountered. If BEN is 0, the MMU disables the BPR0 and BPR1 registers.

UB is the User Break bit. If UB is 1, the MMU enables the BPR0 and BPR1 registers for User mode operation only. If UB is 0, the MMU enables the registers for both User and Supervisor mode. The UB bit is ignored if breakpoints are disabled (BE = 0).

AI is the Abort or Interrupt bit.

FT is the Flow Trace bit. If FT is 1, the MMU enables the PF0, PF1, SC0, and SC1 registers and traces program execution. If FT is 0, the MMU disables the registers.

UT is the User Trace bit. IF UT is 1, the MMU enables the PF0, PF1, SC0, and SC1 registers for User mode operation only. IF UT is 0, the MMU enables the registers for both User and Supervisor mode. The UT bit is ignored if flow trace is disabled (FT = 0).

NT is the Nonsequential Trace bit. If NT is 1, the MMU enables the Nonsequential Trace interrupt. The MMU stops execution of any branch, jump, call, or return instruction and sends a non-maskable interrupt to the system CPU. If NT is 0, the MMU disables the interrupt.

The MSR control bits and flags may be read or modified by executing the SMR and LMR instructions. The NT, FT, TS, TU bits and the ERC flag are set to 0 whenever the system is reset. The NT, FT, and BEN bits are set to 0 whenever the MMU generates a break on a breakpoint, a flow trace interrupt, or an instruction abort on an address translation error.

After writing to the MSR, the MMU automatically suppresses the generation of breaks and flow tracing until a branch, jump, call, or return instruction has been executed. This permits a routine to set the MSR and

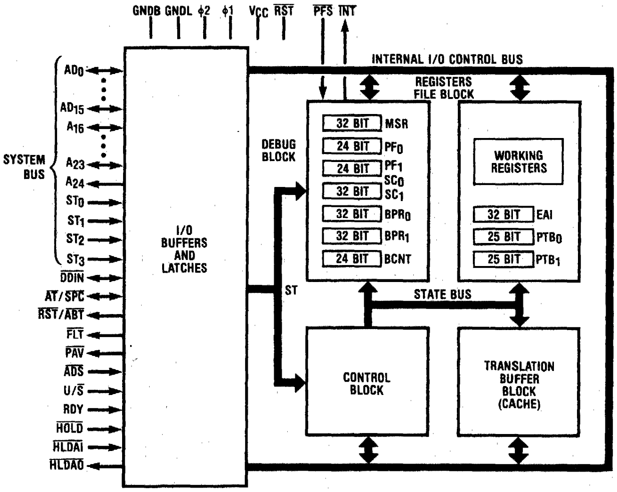


Figure 2-1. MMU Block Diagram

then pass execution to the program being debugged without generating a premature break. The Error Memory Cycle Type (EMCT) is the combination of the BST, EST, BD and ED fields.

Program Flow Registers

The Program Flow registers PF0 and PF1 record the addresses of the two most recently executed nonsequential instructions. Nonsequential instructions are instructions to which execution control is passed by a program exception or a branch, jump, call, or return instruction.

The PF0 and PF1 registers are each 24 bits in length. PF0 contains the address of the last nonsequential instruction to be executed, and PF1 contains the address of the next to last nonsequential instruction to be executed.

The MMU records program flow by copying the address of the current nonsequential instruction to PF0 and copying the previous contents of PF0 to PF1. The MMU copies new addresses to the PF0 and PF1 registers each time a nonsequential instruction is executed by the system.

The FT bit in the MSR enables/disables the registers. If FT is 1, the registers record the program flow. If FT is 0, the registers are disabled.

The contents of the PF0 and PF1 registers may be read by executing the SMR instruction and cleared by executing the LMR instruction to any of the program flow registers.

Sequential Count Registers

The Sequential Count Registers SC0 and SC1 record the number of sequential instructions the system executes after each of the two most recent non-sequential instructions. The SC0 and SC1 registers occupy one 32-bit register and have the format shown on page 2.

SC0 contains the number of sequential instructions executed after the most recent nonsequential instruction. SC1 specifies the number of sequential instructions executed between the nonsequential instruction specified by PF1 and the instruction specified by PF0.

The MMU records the sequential count by incrementing by 1 the SC0 register for each sequential instruction executed. On execution of a nonsequential instruction, the MMU copies the contents of SC0 to SC1 and clears SC0 to 0 to begin a new count. The MMU continues to increment SC0 until another nonsequential instruction is executed or until SC0 is incremented to 65535. SC0 count cannot exceed 65535.

The FT bit in the MSR enables/disables the SC0 and SC1 registers. If FT is 1, the registers record the sequential instruction count. If FT is 0, the registers are disabled.

The contents of the SC0 and SC1 registers may be read by executing the SMR instructions and cleared by executing the LMR instruction to any of the program flow registers. The instructions treat the registers as a single 32-bit register with SC0 at the low order word, SC1 at the high order word.

Breakpoint Registers

The Breakpoint Registers BPR0 and BPR1 provide the breakpoint addresses and breakpoint conditions for system breaks. The registers are each 32 bits in length and have the format shown on page 2.

Bits 0 to 23 specify the breakpoint address. The MMU compares the breakpoint address with addresses referred to by the program. If a match is found and breakpoint conditions are met, the MMU sends a Nonmaskable interrupt to the system CPU and breaks program execution.

Bits 26 to 31 specify the breakpoint conditions (bits 24 and 25 are not used). Breakpoint conditions define how the MMU compares the breakpoint address and which conditions permit the MMU to generate breaks.

AS is the Address Space bit. If AS is 0, the MMU compares the breakpoint address with virtual addresses whose Level 1 Page Table is specified by the PTB0 register. If AS is 1, the MMU compares the breakpoint address with virtual addresses whose Level 1 Page Table is specified by the PTB1 register. If the VP bit is 1, the MMU takes the AS bit as bit 24 of the physical address.

VP is the Virtual/Physical bit. If VP is 0, the MMU compares the breakpoint address with virtual addresses only. If VP is 1, the MMU compares the breakpoint address with translated virtual addresses (i.e., final physical addresses) or physical addresses only.

BE is the Breakpoint Execution bit. If BE is 1, the MMU breaks program execution when the instruction at the breakpoint address is executed. The instruction must start at the breakpoint address for the break to occur. If BE is 0, no break occurs.

BR is the Breakpoint Read bit. If BR is 1, the MMU breaks execution when data is read from the breakpoint address. If BR is 0, no break occurs.

BW is the Breakpoint Write bit. If BW is 1, the MMU breaks execution when data is written to the breakpoint address or when data is read from the breakpoint address in the first part of a read-modify-write operation. If BW is 0, no break occurs.

CE is the Counter Enable bit (BPR0 only). If CE is 1, the Breakpoint Count register is enabled. If CE is 0, the register is disabled. The Breakpoint Count register is described in the next section.

Breakpoint Count Register

The Breakpoint Count Register (BCNT) controls the generation of the MMU interrupt signal to the CPU. It permits the user to specify the number of breakpoints the MMU should ignore before generating a break. The BCNT register is 24 bits in length.

The BCNT register affects system breaks only when it is enabled. The CE bit in the BPR0 register enables/disables the register. When the MMU encounters a breakpoint, it checks the CE bit in the register containing the breakpoint address. If CE is 1, the MMU decrements the contents of BCNT by 1, compares the new contents with zero. If the new contents are not equal to zero, the MMU ignores the breakpoint, i.e., it permits program execution to continue. If the contents are zero, the MMU breaks execution. If CE is 0, the MMU ignores the BCNT register and breaks program execution.

The user may set the register to any value within the range 0 to $2^{24}-1$ by executing an LMR instruction. If the register is not given a new value after a break, the next breakpoint decrements the register contents by 1.

2.1.2 Register File Block

This block contains a number of working registers, with no external access, used to execute the address translation algorithm. In addition, it has three addressable registers (PTB_0, PTB_1, and EAI) used in performing dynamic address translations.

Page Table Base Registers

The Page Table Base registers PTB0 and PTB1 specify the base addresses of the Level 1 Page Tables used in address translation. The PTB0 and PTB1 registers are each 32 bits in length and have the format shown on page 2.

Bits 0 to 23 specify the Page Table Base address. When a virtual address is translated, the MMU reads the base address from the register and accesses the specified Page Table. Bits 0 to 9 must be zeroes. Bits 24 to 30 are not used. Bit 31 is the Memory Space bit. It is intended to be used in system emulation.

The MMU accesses only one Page Table Base register for any given address translation. The current mode of system operation (User or Supervisor) and the Dual Space bit (DS) in the MSR specify which register is read. If the DS bit is 0, the MMU reads the base address from the PTB0 register when in either User or Supervisor mode. If the DS bit is 1, the MMU reads the base address from PTB1 when in User mode and reads the base address from PTB0 when in Supervisor mode.

The contents of the registers may be read or modified at any time by executing an SMR and LMR instruction.

Error/Invalidate Address Register

The Error/Invalidate Address register (EIA) is a dual purpose register that (1) holds a virtual address that has generated an MMU exception, and (2) when written to, removes page table entries from the MMU's Translation Buffer. The EIA is 32 bits in length.

The EIA permits examination of the virtual address that caused the current MMU exception. On an exception (such as a protection level error), the MMU copies the virtual address that generated the error to the EIA. The MMU sets bit 31 in the EIA to 1 if the address's Level 1 Page Table is specified by PTB1 and to 0 if the Level 1 Page Table is specified by PTB0. The error address may be read by executing an SMR instruction. The cause of the error is specified by the ERC and TET flags in the MSR.

The EIA also permits removal of invalid Page Table Entries from the MMU's Translation Buffer. The Translation Buffer contains a copy of the Level 2 Page Table Entries of recently accessed virtual addresses. A virtual address written to the EIA causes the MMU to remove the Page Table Entries of that virtual address from the Translation Buffer. Bit 31 of the EIA must be set to 1 if the address' Level 1 Page Table is specified by PTB1 and set to 0 if the Level 1 Page Table is specified by PTB0. Page Table Entries must be removed whenever the user modifies the corresponding entries in the page tables. The user may write to the EIA register using an LMR instruction.

2.1.3 Translation Buffer Block

The Translation Buffer is the cache memory of the chip. It provides direct virtual to physical address mapping for the most recently used pages in memory. Entries in the Translation Buffer are allocated and replaced by the MMU; the programmer is not involved in the process.

The Translation Buffer is a content-addressable memory. The virtual page frame number (the 15 high order bits of the virtual address) and the address space bit are compared to the entries in the buffer. If the virtual page frame number is present in the buffer, the mapped physical address is output immediately. If not, a control line is set, indicating to the Control Block that the memory page tables should be referenced. When this occurs, the MMU gets the corresponding mapping from memory and replaces the least recently used entry in the Translation Buffer with the new mapping.

Each entry in the Translation Buffer has, besides the virtual and physical page frame numbers and the address space bit, a copy of the protection level field (PL) and the modified bit (M) of the corresponding Page Table Entry. These bits are used by the MMU to implement the translation and error handling algorithms described in the Functional Description. The protection level field contains the most restrictive combination of the Level 1 and Level 2 PTE's.

2.1.4 Control Block

This block is made up of state machines and combinatorial logic. Each machine controls the sequence of operations taking place during the different MMU operations. A State Bus carries the operation code; the different blocks decode appropriate signals from the State Bus.

2.1.5 Input/Output Block

The Input/Output block consists of I/O buffers and internal buffers.

The I/O buffers provide the communication between the MMU and the outside system bus. The internal buffers between the I/O buses which transfer the address offset and the complete address in no-translation mode are also part of this block.

2.2 Memory Management Instructions

Format	Instruction		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. (Priv.)
8	MOVUSi	gen,gen	Move a value from User Space to Supervisor Space. (Priv.)

The MOVSUi and MOVUSi instructions are intended for memory management. Instruction format detail can be found in the NS16032 Data Sheet, Appendix A.

3 Functional Description

3.1 Power and Grounding

The NS16082 requires a single 5V power supply applied to pin 48 (V_{CC}). See DC Electrical Characteristics.

Grounding connections are made on pins 24 and 25, Logic Ground Pin (GNDL) and Buffer Ground Pin (GNDB), respectively. GNDL is the common pin for on-chip logic, and GNDB is the common pin for the output drivers. As shown in *Figure 3-1,* GNDL is directly connected to GNDB with a single conductor.

All other grounding connections should be made only to GNDB (pin 25) to ensure optimum noise immunity.

3.2 MMU Operation

The MMU operation incorporates the following:

1. Bus Operation—related to Address Translation, DMA Transfers, Breakpoints on Physical Address, and Slave Operation
2. Slave Instruction Execution
3. Address Translation
4. Hardware Debugging
5. Error Handling

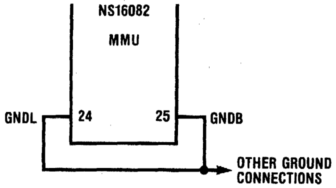


Figure 3-1. Grounding Connections

3.2.1 Bus Operation

Address Translation (see *Figures 3-2* to *3-5*): The MMU time-shares the address/data bus with the CPU. During a memory access cycle, the MMU reads the virtual address, performs the virtual to physical translation, and places the physical address on the bus. A typical memory cycle has five clock periods: T1, TMMU (time of physical address on the bus), T2, T3, and T4. The 16 A/D bus drivers of the MMU are in high impedance state at all times except during TMMU or when the $\overline{\text{FLT}}$ signal is active. The bus drivers of lines A16 to A24 drive the bus from TMMU through T4.

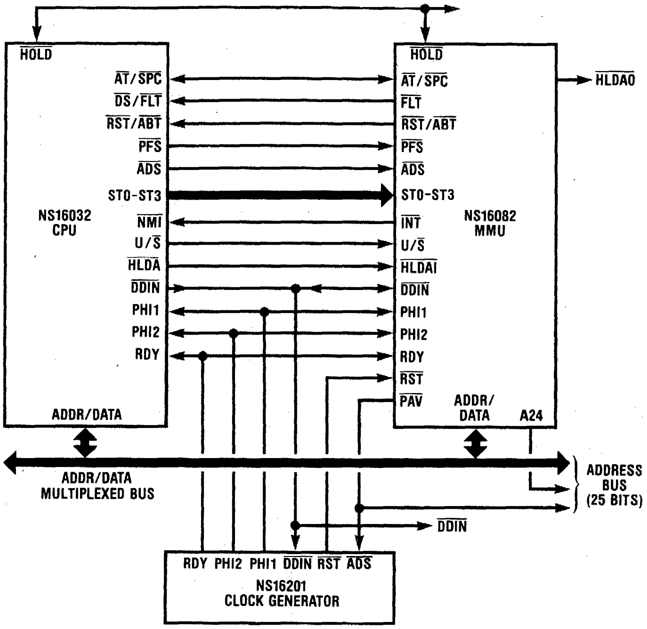


Figure 3-2. CPU, MMU Interconnections

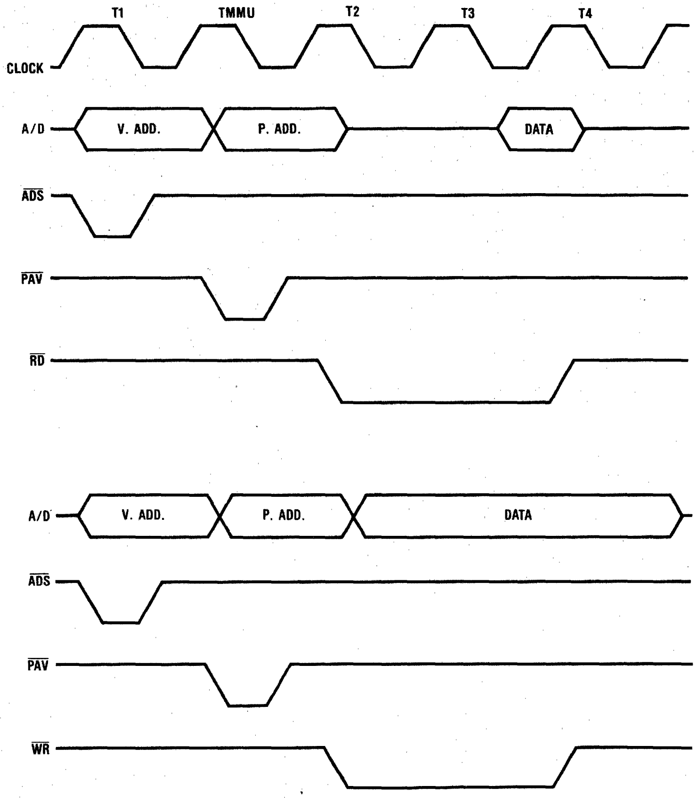


Figure 3-3. Bus Operation Timing: Virtual Address in Translation Buffer

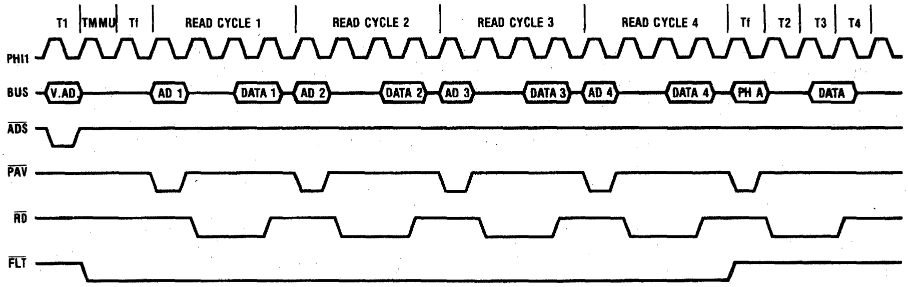


Figure 3-4. Bus Operation: Read Cycle When Virtual Address Is Not In Translation Buffer

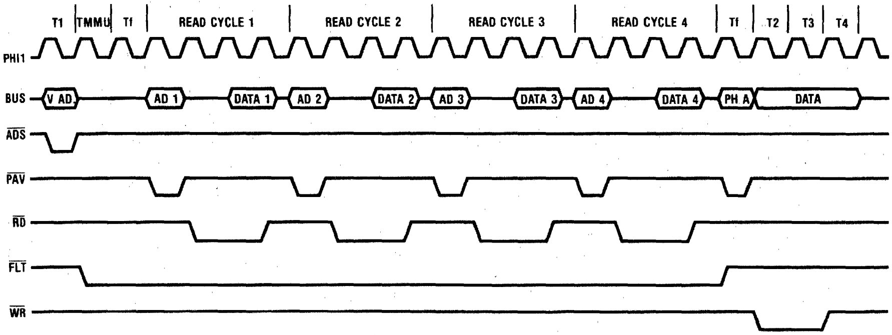


Figure 3-5. Bus Translation Write Cycle When Virtual Address Is Not In Translation Buffer

During period T1, the CPU places on the bus the virtual address to be translated; this address is strobed into the MMU with the $\overline{\mathrm{ADS}}$ pulse. During period TMMU, the CPU places the bus in high impedance and the MMU does one of two things. If the address to be translated is in the Translation Buffer, the MMU sends the physical address on the bus with a $\overline{\mathrm{PAV}}$ timing pulse; if not, it takes the bus from the CPU with the $\overline{\mathrm{FLT}}$ signal and executes four memory read cycles, to get the two double words needed to perform the translation algorithm. When necessary, the MMU executes two memory write cycles to update the referenced and modified bits in the Page Table Entry. It then releases control of the bus and sends the physical address on the bus. The memory cycle initiated by the CPU is resumed from the point it was stopped.

Between periods T2 and T4, there is data on the AD0-AD15 bus lines, output either by the CPU or memory. Bus lines A16 to A24 continue to hold the physical address.

DMA Transfers: The Hold and Hold Acknowledge lines are connected as shown in *Figure 3-6.*

The DMA device pulls the $\overline{\mathrm{HOLD}}$ line low to request the bus; this line is input to both the CPU and the MMU. If the MMU is not floating the CPU (through the $\overline{\mathrm{FLT}}$ line), the MMU transfers the $\overline{\mathrm{HLDA}}$ CPU output directly to the $\overline{\mathrm{HLDAO}}$ MMU output. If the MMU (when accessing the Memory Page Tables) is floating the CPU, the CPU cannot respond to $\overline{\mathrm{HOLD}}$ request, $\overline{\mathrm{HLDAI}}$ remains high, and the MMU grants the bus by pulling low $\overline{\mathrm{HLDAO}}$ at the end of the present memory cycle. When the DMA device releases $\overline{\mathrm{HOLD}}$, the MMU releases $\overline{\mathrm{HLDAO}}$ and regains control of the bus.

Breakpoint on Physical Address: During debug, if a breakpoint is specified to occur on a physical address (VP is set in any BPR), an additional clock period is needed in the bus cycle. The additional clock period is required to make the address comparison after getting the physical address from the cache or page table. In this case the MMU floats the CPU for one clock period. This gives the memory cycles six periods: T1, TMMU, Tf, T2, T3, and T4. The corresponding waveforms are illustrated in the Timing Characteristics, *Figure 3-7.*

Slave Instruction Bus Operation: For slave instructions, the bus operation follows a different protocol. The bus cycle has only two periods (T1 and T4) and the timing is done by a one-clock-wide pulse on the Slave Processor Control ($\overline{\mathrm{SPC}}$) bidirectional line. All bus transfers are illustrated in Timing Diagrams, *Figures 3-8* and *3-9.*

3.2.2 Slave Instructions

Introduction to Slave Instructions

The MMU Slave Instructions serve two purposes. First, slave instructions set up the different registers and check their contents (LMR and SMR instructions) in order to control the MMU mode of operation. Second, a slave instruction can request the MMU to return a flag indicating whether a specified access to a given address would generate a protection fault in user mode.

The general format for slave instructions appears in the NS16032 Microprocessor Data Sheet. The formats for the MMU slave instructions are described below.

Note: All MMU instructions are privileged. While in the User Mode, the CPU will trap on any MMU instruction.

MMU Slave Instruction Format

The 3-byte format of the MMU slave instruction is shown on page 2.

The format corresponds to the instructions as they are stored in memory; the CPU sends the operation word to the MMU with its bytes swapped, i.e., high byte in low bus byte and vice versa.

The short code assignments for the registers are shown below:

Code Value	Register
0000	BPR0
0001	BPR1
0100	PF0
0101	PF1
1000	SC
1010	MSR
1011	BCNT
1100	PTB0
1101	PTB1
1111	EAI/INVALIDATE

Note: All other short codes are illegal.

Address Translation Validation Instructions

The two instructions used to validate an address are: RDVAL address and WRVAL address. Both instructions consist of mnemonics and address type operands.

Upon receipt of a RDVAL or WRVAL instruction, the MMU checks if the address operand can be translated without protection violations in user mode (user space). If the address can be translated without violations, the MMU sends status word zero. If not, the MMU sends status word 32.

A trap is generated with error class 1 and error translation type 2 if the first Page Table Entry is invalid. No trap is generated if the second PTE is invalid or if protection violation errors occur.

A Validate instruction generates a status word which sets or resets a flag bit (F) in the CPU PSR register. This flag is positioned in bit 5. The remaining bits are all zero. Slave Instruction operation is shown in the following charts.

RDVAL/WRVAL Instruction (Validate Read/Write Address)

CPU		MMU	
Execution Unit	**Bus Interface Unit**	**Status Pins**	**Action**
Sends ID Code in low byte	Sends ID Code with SPC timing pulse	1111	Recognizes ID Code
Sends Opcode in two bytes	Sends Opcode with SPC timing pulse	1101	Latches Opcode
Sends Address to be validated in two words (bits 24-31 set to zero)	Sends Address in two Write Slave cycles with SPC timing pulse	1101	
Generates Dummy Read with address to be validated	Starts a Read cycle with address to be validated	1010	Performs validation
	Detects MMU completion	0011	Signals completion SPC pulse
Reads MMU status	Reads MMU status word with SPC strobe	1110	Sends status word

LMR Instruction (Load MMU Register)

LMR short, read.d (See NS16000 Programmers Reference Manual, Document No. 420306565-001.)

The MMU register specified by first operand is loaded with the contents of the second operand. The instruction executes as follows:

CPU		MMU	
Execution Unit	**Bus Interface Unit**	**Status Pins**	**Action**
Sends ID Code in low byte	Sends ID Code with SPC timing pulse	1111	Recognizes ID Code
Sends Opcode in two bytes	Sends Opcode with SPC timing pulse	1101	Latches Opcode
Sends low word of operand	Sends low word of operand with SPC timing pulse	1101	Stores operand in low word of addressed register
Sends high word of operand	Sends high word of operand with SPC timing pulse	1101	Stores operand in high word of addressed register

SMR Instruction (Store MMU Register)

SMR short, write.d

The MMU register specified by first operand is stored in the second operand. The instruction executes as follows:

CPU		MMU	
Execution Unit	**Bus Interface Unit**	**Status Pins**	**Action**
Sends ID Code in low byte	Sends ID Code with SPC timing pulse	1111	Recognizes ID Code
Sends Opcode in two bytes	Sends Opcode with SPC timing pulse	1101 (See Note 1)	Latches Opcode
	Detects MMU completion	0011	Signals completion with SPC pulse
	Reads status with SPC strobe	1110	Sends zero status
	Strobes operand with the SPC pulse	1101	Sends low word of addressed register
	Strobes operand with SPC pulse	1101	Sends high word of addressed register

Notes:

1. The CPU may prefetch more code before this step.
2. After CPU reads the operand, the contents are stored in second operand according to the second operand addressing mode.
3. If addressed register is less than 32 bits, then the high order bits are reset to zero.

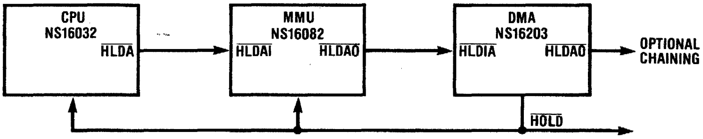


Figure 3-6. Hold Connections

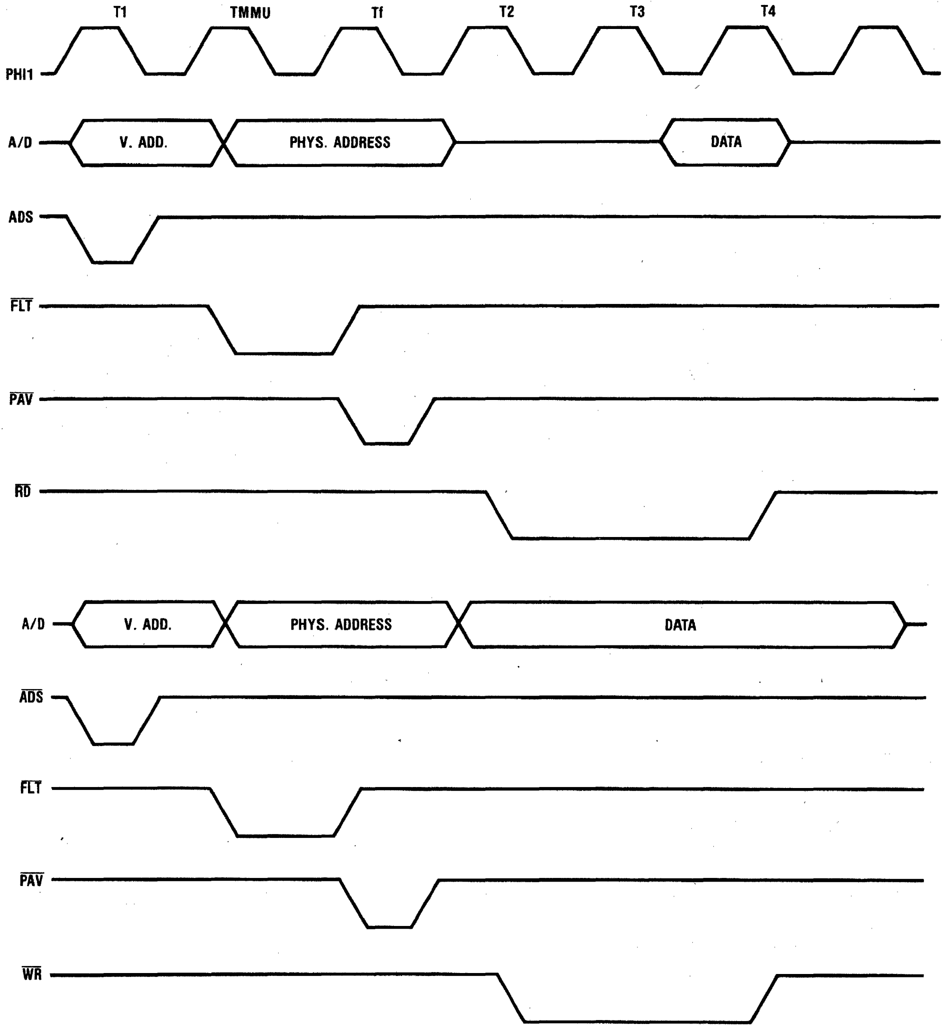


Figure 3-7. Bus Operation in Breakpoints on Physical Address

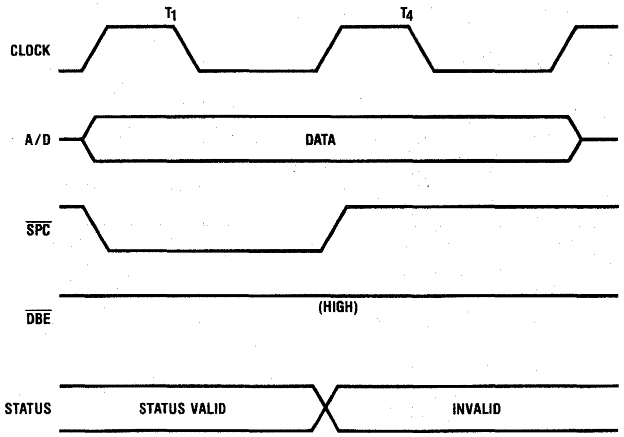


Figure 3-8. Slave Instruction Timing: Get ID/Opcode/ Data from CPU

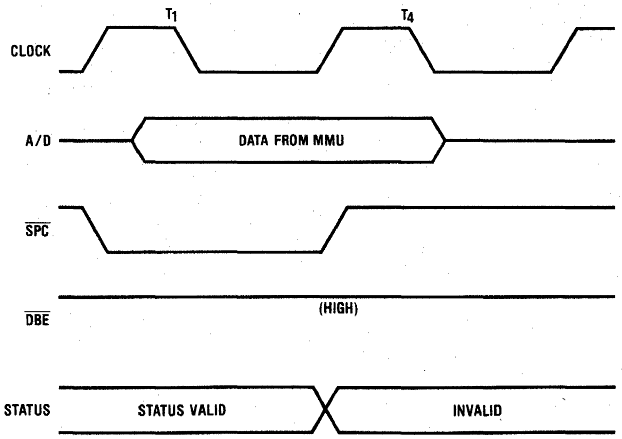


Figure 3-9. Slave Instruction Timing: MMU Sends Status/Data to CPU

3.2.3 Address Translations

Page Table Entry (PTE) Format

Address translation is controlled by page tables contained in memory. A page table is a linear array of Page Table Entries. Each PTE defines the access characteristics of one page (512 bytes) of virtual storage. The PTE bit format is shown on page 2.

BS Bank select; most significant bit of PFN field.

PFN Page Frame Number; when the V bit is set, the PFN low field, together with the BS bit, contains the high order 16 bits of a physical page address which is used by the address translation algorithm.

M Modified; used only in index 2 (bits 9 to 15 of virtual address) PTE's and set when page mapped is modified.

R Referenced; set when page mapped by PTE is referenced.

PL Protection Level; Index 1 PTE and Index 2 PTE's control access to pages mapped by the PTE. The table below shows the relationship between User, Supervisor and protection level bit:

Mode	PSR bit 8	Protection Level Bits 00	01	10	11
User	1	no access	no access	read only	full access
Supervisor	0	read only	full access	full access	full access

V Valid bit; when set indicates that the corresponding page is resident in physical memory. When cleared, any attempted reference to the page will cause the MMU to abort the reference. If the V-bit is cleared, the PTE may be used by the Operating System for any desired function.

Note: Bits 7 and 8 are reserved for the user and are not affected by the MMU.

Address Translation Algorithm

The MMU translates the 24-bit virtual address generated by the CPU to either a 25-bit physical address or a translation error. This process is described below. See *Figure 3-10.*

The virtual address is divided into three components as shown on page 2.

The access level of a reference is a two-bit number whose logical expressions are:

bit 1 = U and AO
  where AO = Access Override bit in MSR

bit 0 = 1 for write, read/modify/write, (RMW)
bit 0 = 0 for read

The detailed description of the translation algorithm follows. (See NS16000 Programmers Reference Manual 420306565-001.)

If TU = 0 and U = 1 or TS = 0 and U = 0, then PA = virtual address, else

```
1. Select first PTE:
     If DS (in MSR) = 1 and U (in PSR) = 1, then PTEP =
     PTB1 or Index 1 * 4
   else
     PTEP = PTB0 or Index 1 * 4
   end.
   Validate PTE:
     If access level is greater than (PTEP).PL or if
     (PTEP).V = 0, then abort CPU
   else
     Set (PTEP).R = 1
2. Select second PTE:
     PTEP = (PTEP).PFN * 512 or Index 2 * 4
   Validate PTE:
     If access level is greater than (PTEP).PL or if
     (PTEP).V = 0, then abort CPU
   else
     Set (PTEP).R = 1
     If writing, then set (PTEP).M = 1
3. Generate physical address:
     PA = (PTEP).PFN * 512 or Offset
```

Legend:

PA	-	Physical Address
TU, DS, TS	-	MSR bits
U	-	Program Status Register bit (sent to MMU via the U/S pin)
PTEP	-	PTE pointer
(PTEP).PL	-	represents protection level in Page Table Entry
(PTEP).V	-	represents valid bit in Page Table Entry
(PTEP).M	-	represents modified bit in Page Table Entry
PFN	-	Page Frame Number

The MMU marks bits R and M of the PTE for subsequent use by the operating system. If a physical page is written upon, it is assumed that the user intends for this modification to be permanent in his storage system. The M bit indicates whether a page needs to be written to mass storage when it is deallocated from physical memory. The R bit is tested and cleared periodically by the operating system in order to compile statistics of the frequency of references to each page currently in memory. It will use this information to deallocate the least frequently used pages when new pages must be called in.

Page tables that refer to physical pages are themselves referenced by two page tables of double length. Selection of the PTB0 or PTB1 register depends on the Dual Space (DS) and User/Supervisor (U/S) modes as shown below:

		U/$\overline{S}$ 0	1
DS	0	PTB0	PTB0
	1	PTB0	PTB1

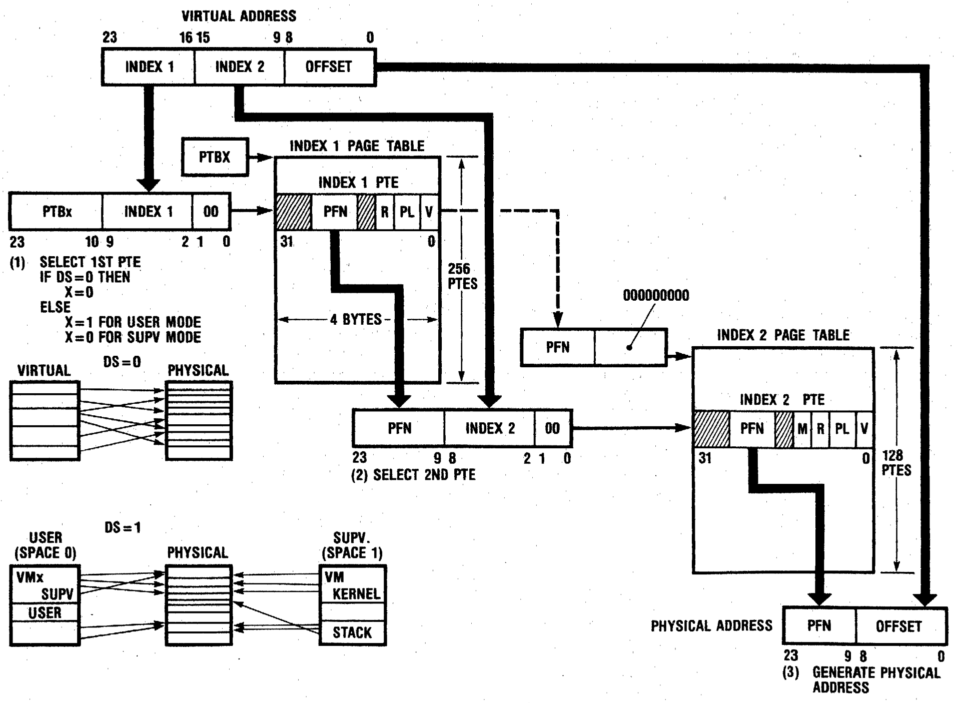


Figure 3-10. Virtual to Physical Address Translation

Page Table Base (PTB) Registers

PTB0 and PTB1 registers are specified as double words. The BS bit is used by the MMU to produce the 25th bit of the physical addresses pointing to the entries in the first translation table. Their format is given on page 2.

3.2.4 Hardware Debugging

The MS16082 MMU incorporates two special debugging facilities: program flow tracing and breakpointing.

Program Flow Tracing: Program Flow Tracing allows the software to reconstruct the sequence of exception executed prior to a certain instruction or breakpoint. Registers PF0, PF1 and SC are used to record program flow information. The SC register is divided into two 16-bit fields (SC0 and SC1). SC0 is also a 16-bit counter. For the user, these are read-only registers which can be read via an SMR slave instruction. An LMR instruction addressing any of the PF registers resets all of them.

When a sequential instruction is executed, a PFS pulse is received form the CPU for acknowledgement. This pulse is also used to increment the SC0 counter.

When a nonsequential instruction is fetched, the content of register PF0 is copied into PF1 and that of SC0 into SC1; then the virtual address of this instruction is stored in PF0 and the SC0 counter is reset. This happens only if before the fetch the SC0 content was not zero (to prevent multiple tracing for instructions which cause more than one queue flush, e.g., ACBI), and bits FT and UT in the MSR register were set.

Note: When the SC0 counter reaches 64k minus 1, it stops counting.

Entry To and Exit From a Debugged Program: When the MSR is written to, debugging traps are disabled. At the end of the second nonsequential fetch cycle they are enabled. This feature enables entry to a debugged program from a monitor or debugger after the flushing of the queue and a nonsequential fetch. This will occur normally without getting an immediate trap if bit NT is set.

After a debugging trap, the MSR bits which enable this trap (NT, BEN) are cleared, thus inhibiting further debug traps until the MSR is rewritten by the monitor or debugger program. Bit FT in the MSR is also cleared thus freezing the program tracing. This last feature inhibits the tracing of useless information (like the program flow inside the monitor or debugger), until the program tracing registers are read.

Breakpoints: A breakpoint generates an abort or interrupt pulse when a software specified address is referenced under software controlled conditions. It also updates the ERC and BN fields in MSR. Breakpoints are controlled by the BEN and UB bits (in MSR) and the BPR registers which have the format shown on page 2.

Breakpoint on Execution Fetch mechanism: When a sequential instruction is fetched by the CPU, the instruction is placed in the queue. Unless the queue is empty, aborts on queue fetches are not received and so a breakpoint could be missed. The proper operation of breakpoint execution requires flushing the queue, as described below.

When the BE bit is set and the location specified in the BPR is accessed in a nonsequential fetch, an Abort or INT pulse is generated.

When the BE bit is set and the location specified in the breakpoint register is accessed in a sequential fetch (or in a nonsequential fetch from an even-numbered address (2n) and the location specified in BPR is (2n + 1), the MMU returns a DIA instruction instead of the memory byte at the breakpoint location. This is preceded by a read cycle in order to return the other original byte from memory. This causes the CPU to flush the queue and to fetch the instruction a second time, now with a nonsequential fetch status.

The BPR bit functions are tabulated below:

AS	Address Space; virtual address when VP = 0, bank select bit of physical address when VP = 1.
VP	Virtual or Physical address; if VP is set, address field is matched against physical address. If VP is reset, address field is matched against virtual address.
BE	Breakpoint on Execution; if BE is set, a breakpoint occurs when the location specified in ADDRESS field is referenced in an instruction fetch cycle (instruction execution detailed below).
BR	Breakpoint on Read operand; if BR is set, a breakpoint occurs when the location specified in ADDRESS field referenced in a read operand cycle.
BW	Breakpoint on Write operands; if BW is set, a breakpoint occurs when the location specified in ADDRESS field is referenced in a write or RMW operand cycle.
CE	Counter Enable (BPR0 only); the 24-bit BCNT counter decrements when Counter Enable bit (CE) is set and the conditions for a breakpoint in register BPR0 are obtained. When this counter reaches zero, an "abort" or INT pulse is generated by the MMU.

Note: An erroneous count will result if both the CE and BW bits are set.

3.2.5 Error Handling

Traps are serviced according to class and type (c, t) as follows:

In the MSR register, the appropriate bit in the ERC field is set due to the fact that RMW accesses are counted twice.

For Address Translation Error, the following bits are set in the TET field:

If access level is greater than (PTEP).PL	bit 0 set
If (PTEP).V = 0 in Index 1 PTE	bit 1 set
If (PTEP).V = 0 in Index 2 PTE	bit 2 set

In the EMCT field set the CPU status and DDIN bits.

In the EAI register, set AS bit to designate the address space PTB0/PTB1 of virtual address being translated and set the address field to the value of the virtual address being translated, as shown in the register format shown on page 2.

For Breakpoint Error, the following bits are set in the MSR register:

BN field — the number of the appropriate breakpoint register

EMCT field — CPU status and DDIN bits

4 AC Electrical Characteristics

4.1 Definitions

All the timing specifications given in this chapter refer to 50% of the leading or trailing edges of the appropriate clock phase and 0.8V or 2.0V on the appropriate signal as illustrated in *Figures 4-1* and *4-2*, unless specifically stated otherwise.

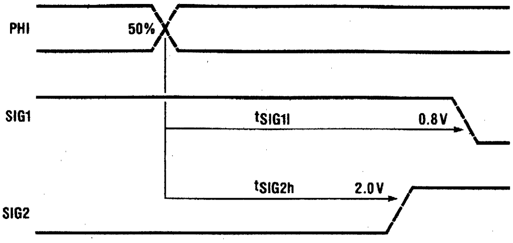


Figure 4-1. Timing Specification Standard (Signal Valid After Edge)

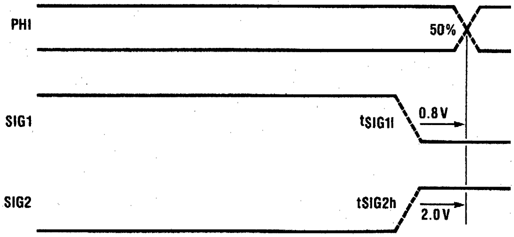


Figure 4-2. Timing Specification Standard (Signal Valid After Edge)

ıe *Figure 4-3* through *4-10*.)

0°C to 70°C
5V ±5%
100pF max.

Description	Duration (ns)
high pulse width	Note 3
low pulse width	Note 3
fall time	5 max.
rise time	5 max.
d	100 min./ 2000 max.
dress valid	50 max.
dress float	25 max.
7 active	35 max.
7 inactive	95 max.
	40 min.
t up time before	
	20 min.
ld time after $\overline{\text{PAV}}$ TE	10 min.
$\overline{\text{IN}}$ active	50 max.
$\overline{\text{IN}}$ inactive	50 max.
ıp time before TE of	
	30 min.

Timing	Description	Duration (ns)
tRDYh	Ready hold time after LE of PHI1 (T3)	0 min.
tSPCa	PHI1 to $\overline{\text{SPC}}$ active	35 max.
tSPCia	PHI1 to $\overline{\text{SPC}}$ inactive	35 max.
tDvSPCia	Data hold time after $\overline{\text{SPC}}$ TE	15 min.
tDV	PHI1 to Data valid	50 max.
tDiv	PHI1 to Data invalid	25 max.
tDs	Data set-up time before TE of PHI2 (T4)	15 min.
tDh	Data hold time after LE of PHI1 (T4)	0 min.
tFLTa	PHI1 to $\overline{\text{FLT}}$ active	45 max.
tFLTia	PHI1 to $\overline{\text{FLT}}$ inactive	45 max.
tFLTw	$\overline{\text{FLT}}$ width	40 min.
tABTa	PHI1 to $\overline{\text{ABT}}$ active	55 max.
tABTia	PHI1 to $\overline{\text{ABT}}$ inactive	45 max.
tABTw	$\overline{\text{ABT}}$ width	70 min.
tHLDd	delay from $\overline{\text{HLDAI}}$ to $\overline{\text{HLDAO}}$	30 max.

ıinst PHI are relative to PHI leading
d.
railing edge.
ı to 0.35 x tCp.

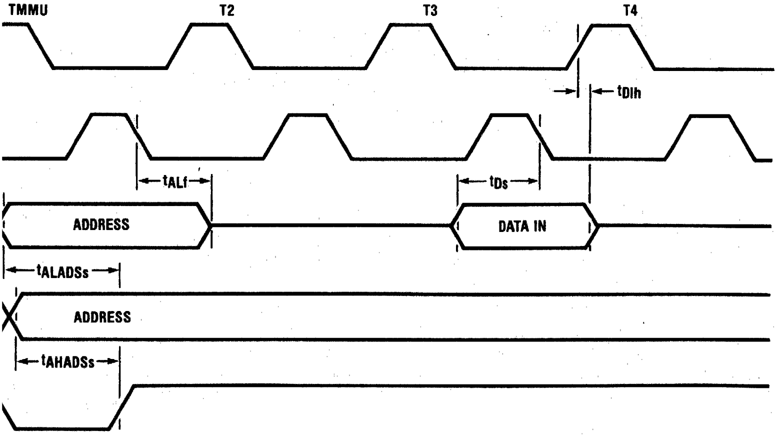


Figure 4-3. Write Cycle

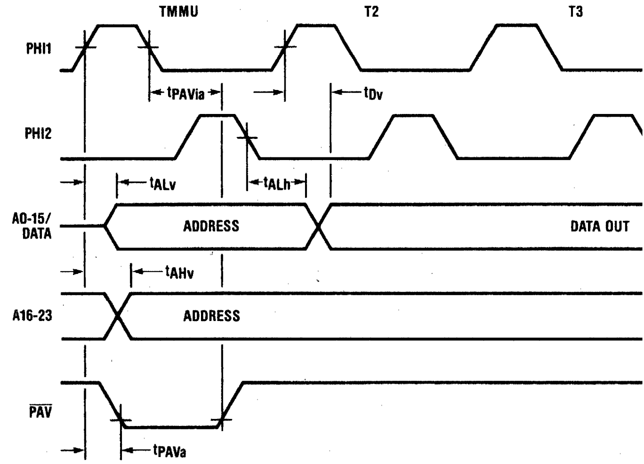


Figure 4-4. Read Cycle

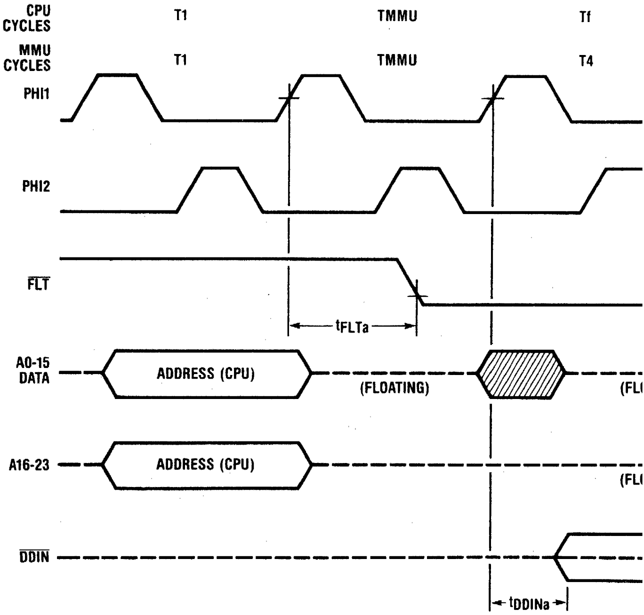


Figure 4-5. FLT Initiated Float Cycle Tim

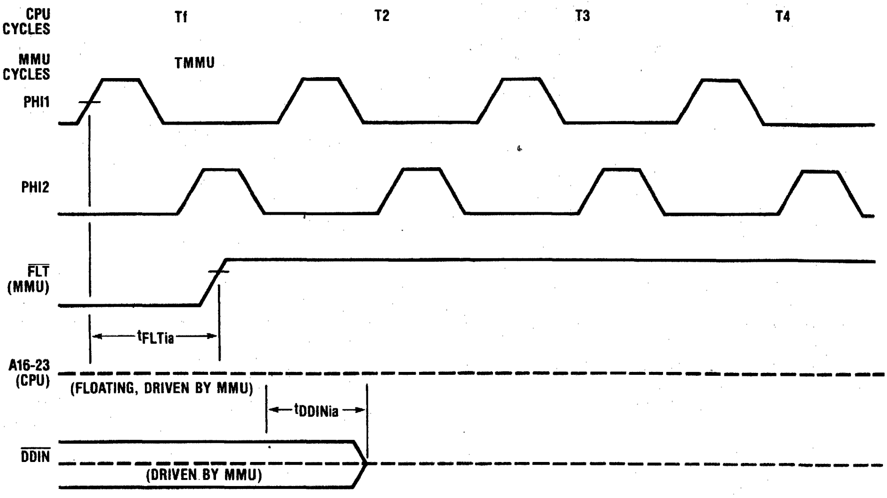
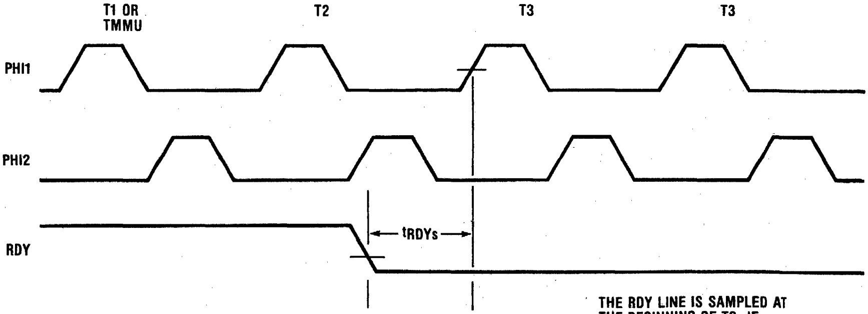


Figure 4-6. Release from FLT Timing



THE RDY LINE IS SAMPLED AT THE BEGINNING OF T3. IF RDY IS LOW, T3 WILL BE REPEATED. IF RDY IS HIGH, T4 WILL BE THE NEXT T STATE.

Figure 4-7. Ready Sampling (MMU Initially Ready)

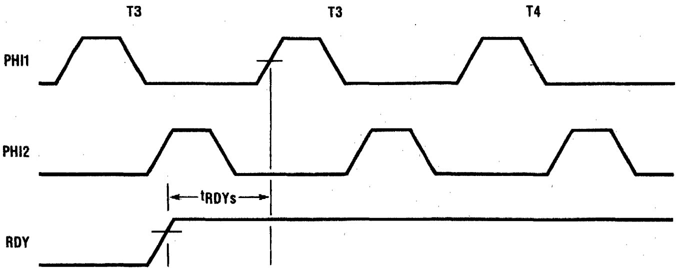


Figure 4-8. Ready Sampling (MMU Initially Not Ready)

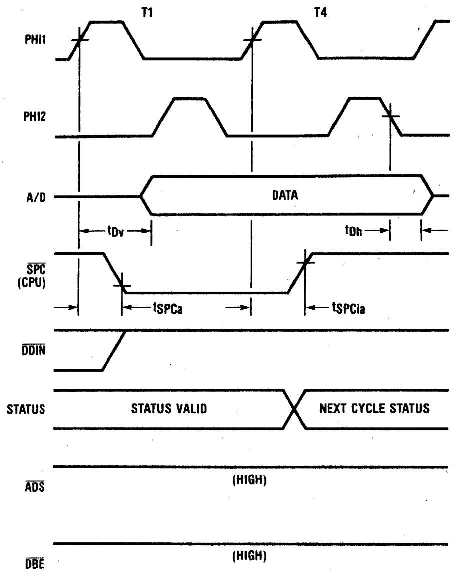


Figure 4-9. Write Slave Processor Timing

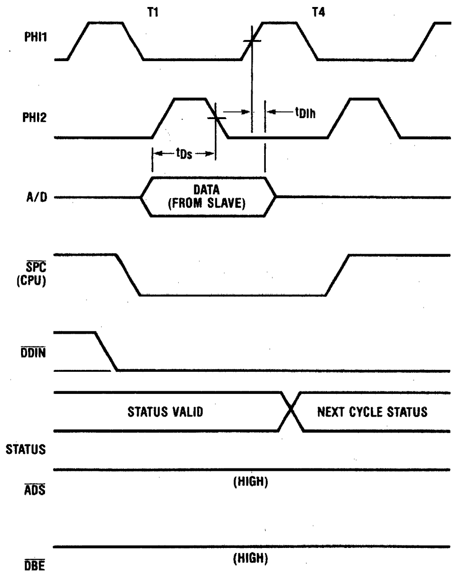
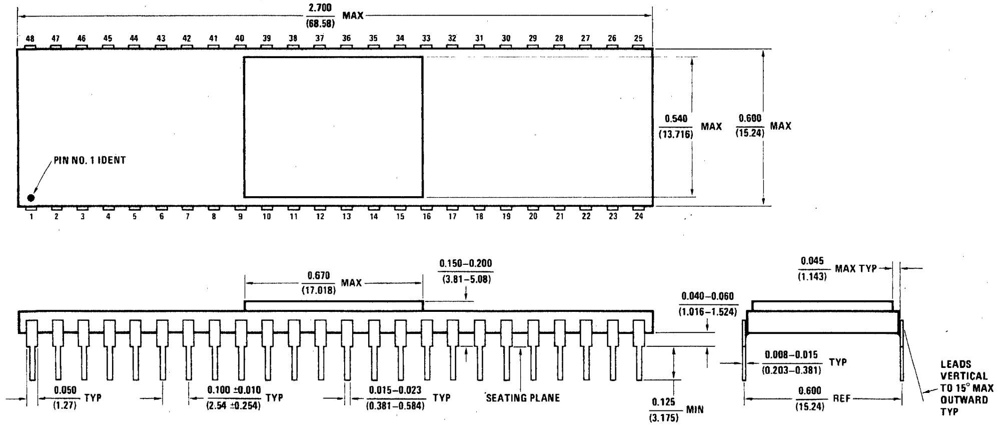


Figure 4-10. Read Slave Processor Timing

Physical Dimensions inches (millimeters)



48-Lead Dual-In-Line Package (D)
Order No. NS16082D
NS Package D48A

Peripherals

National
Semiconductor

DP8350 Series CRT Controllers

General Description

The DP8350 Series of CRT Controllers are single-chip bipolar (I^{2}L technology) circuits in a 40-pin package. They are designed to be dedicated CRT display refresh circuits. Three standard products are available, designated DP8350, DP8352, DP8353. Custom devices, however, are available in a broad range of mask programmable options.

The CRT Controller (CRTC) provides an internal dot rate crystal controlled oscillator for ease of system design. For systems where a dot rate clock is already provided, an external clock may be inputted to the CRTC. In either case system synchronization is made possible with the use of the buffered Dot Rate Clock Output.

The DP8350 Series has 11 character generation related timing outputs. These outputs are compatible for systems with or without line buffers, using character ROMs, or DM86S64-type latch/ROM/shift register circuits.

12 bits (4k) of bidirectional TRI-STATE® character memory addresses are provided by the CRTC for direct interface to character memory.

Three on-chip registers provide for external loading of the row starting address, cursor address, and top-of-page address.

A complete set of video outputs is available including cursor enable, vertical blanking, horizontal sync, and vertical sync.

The DP8350 Series CRTC provides for a wide range of programmablility using internal mask programmable ROMs:

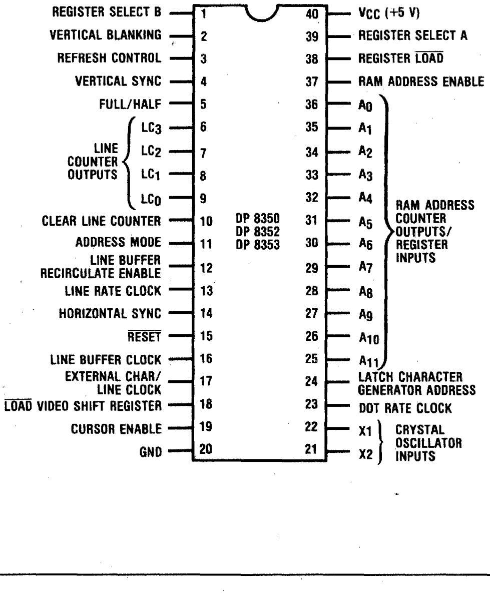
- Character Field (both number of dots/character and number of scan lines/character)
- Characters per Row
- Character Rows per Video Frame
- Format of Video Outputs

The CRTC also provides system sync and program inputs including Refresh Control, Reset, and Address Mode.

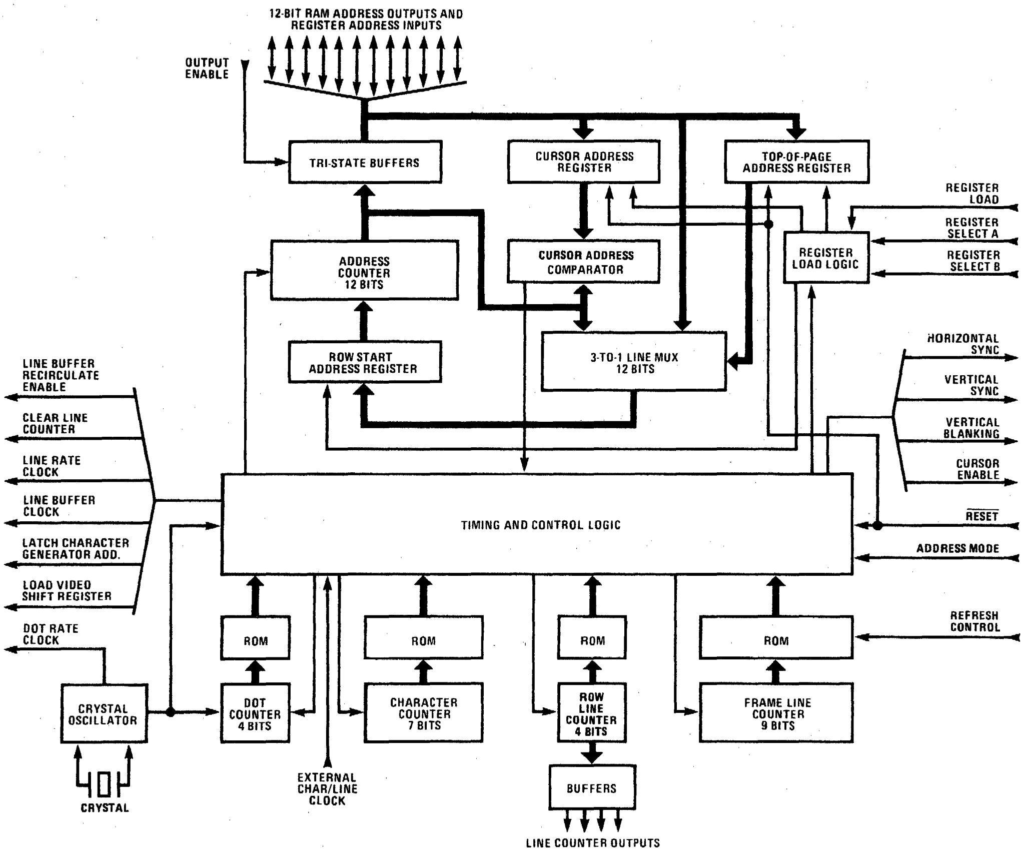
Features

- Internal crystal controlled dot rate oscillator
- External dot rate clock input
- Buffered dot rate clock output
- Timing pulses for character generation
- Character memory address outputs (12 bits)
- Internal cursor address register
- Internal row starting address register
- Internal top-of-page address register (for scrolling)
- Programmable horizontal and vertical sync outputs
- Programmable cursor enable output
- Programmable vertical blanking output
- 2 programmable refresh rates, pin selectable
- Programmable characters/row (128 max.)
- Progammable character field size (up to 16 dots × 16 scan line field size)
- Programmable scan lines/frame (512 max.)
- Programmable character rows/frame
- Single +5V power supply
- Inputs and outputs TTL compatible
- Direct interface with DM86S64 character generator
- Ease of system design/application

Connection Diagram



Block Diagram



The Video Display

Discussion of the CRT Controller necessitates an understanding of the video display as presented by a raster scan monitor. The resolution of the data displayed on the monitor screen is a function of the dot size. As shown in Figure 1, the dot size is determined by the frequency of the system dot clock. The visible size of the dot can be modified to less than 100% by external gating of the serial video data. The CRT Controller organizes the dots into cell groupings that define video rows. These cells are accessed by a specific horizontal address output (4096 maximum) and are resolved by a row scan-line-counter output (16 maximum) as shown in Figure 2. The relation of the video portion of a frame to the horizontal blanking and vertical blanking intervals is shown in Figure 3 in a two-dimensional format.

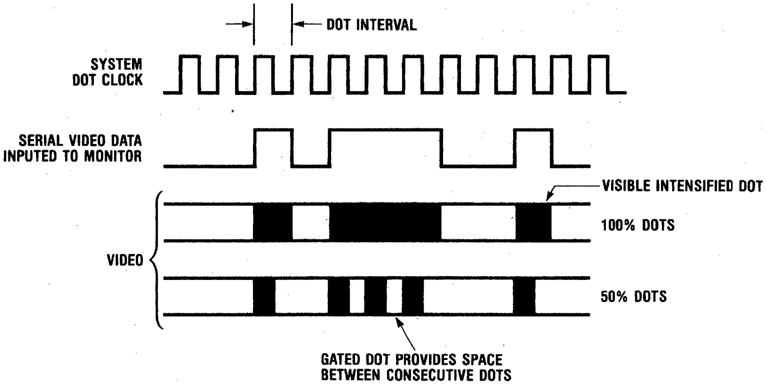

DOT INTERVAL
SYSTEM DOT CLOCK
SERIAL VIDEO DATA INPUTED TO MONITOR
VISIBLE INTENSIFIED DOT
100% DOTS
VIDEO
50% DOTS
GATED DOT PROVIDES SPACE BETWEEN CONSECUTIVE DOTS

Figure 1. Dot Definition

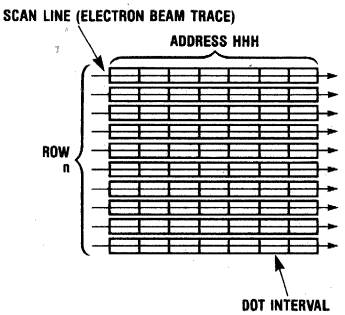


Figure 2. Character Cell Definition (Example Shown is a 7 x 10 Character Cell)

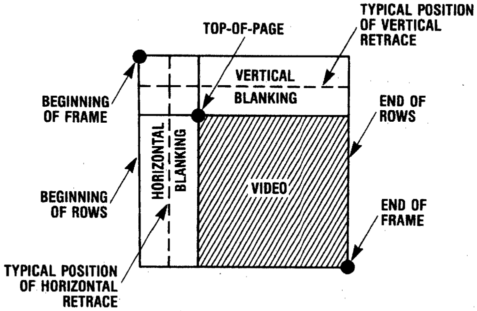


Figure 3. Frame Format Definition

Character Generation/Timing Outputs

The CRT controller provides 11 interface timing outputs for line buffers, character generator ROMs, DM86S64-type latch/ROM/shift register combination character generators, and system status timing. All outputs are buffered to provide TTL compatible direct interface to popular system circuits such as:

- DM86S64 Series Character Generators
- MM52116 Series Character ROMs
- DM74166 Dot Shift Register
- MM5034, MM5035 Octal 80-Bit Shift Registers (Line Buffers)

Dot Rate Clock: This output is provided for use in system synchronization and interface to the dot shift register used in character generation. This output is non-inverting with respect to an external clock applied to the X1 oscillator input (see Figure 6). The dot rate clock output exhibits a 50% duty cycle. All CRTC output logic transitions are synchronous with the rising edge of the Dot Rate Clock output.

Latch Character Generator Address (Character Rate Clock): This output provides an active clock pulse at character rate frequency which is active at all times. The rising edge of this pulse is synchronous with the beginning of each character cell. This output is intended for direct interface to character/video generation data latch registers.

Line Rate Clock: This output provides an active clock pulse at scan-line rate frequency (horizontal frequency), which is active at all times. The falling edge of this pulse is synchronous with the beginning of horizontal blanking. This output is intended for direct interface to character generation scan line counters.

Load Video Shift Register: This output provides a character rate signal intended for direct interface to the video dot shift register used in character generation. Active low pulses are outputted only during video time. As a result of the inactive time, horizontal and vertical video blanking can be derived from this output signal.

Clear Line Counter: This output signal is active only during the first scan line of all rows. It exhibits an active low pulse identical and synchronous to the Line Rate Clock and is provided for direct interface to character generation scan line counters.

Line Counter Outputs (LC_0 to LC_3): These outputs clock at line rate frequency, synchronous with the falling edge of the line rate clock, and provide a consecutive binary count for each scan line within a row. These outputs are provided for system designs that require decoded information indicating the present scan line position within a row. These outputs are always active, however, the next to the last row during vertical blanking will exhibit an invalid line count as a function of internal frame synchronization.

Line Buffer Clock: This output directly interfaces to data shift registers when they are incorporated as line buffers in a system design (see Figure 16). This signal is active at character rate frequency and is intended for shift registers that shift on a falling edge clock. This output is inactive during all horizontal blanking intervals yielding the number of active clocks per scan line equal to the number of video characters per row. For custom requirements, the duty cycle of this output is mask programmable.

Line Buffer Recirculate Enable: This output is provided to control the input loading mode of the data shift register (line buffer) when used in a system design. The format of this output is intended for shift registers that load external data into the input with the mode control in the low state, and load output data into the input (recirculate) with the mode control in the high state. This output will transition to the low state, synchronous with the line rate clock falling edge, for one complete scan line of each row. The position of this scan line will either be the first scan line of the addressed row, or the last scan line of the previous row depending upon the logic level of the address mode input (pin 11), tabulated in Table 3.

Memory Address Outputs/Inputs and Registers

Address Outputs (A_0-A_{11}): These 12 address bits (4k) are bi-directional TRI-STATE® outputs that directly interface to the system RAM memory address bus.

In the output mode (enabled), these outputs will exhibit a specific 12-bit address for each video character cell to be displayed on the CRT screen. This 12-bit address increments sequentially at character rate frequency and is valid at the address bus 2 character times prior to the addressed character appearing as video on the CRT screen. This pipelining by 2 characters is provided to allow sufficient time for first, accessing the RAM memory, and second, accessing the character generation memory with the RAM memory data. Since a character cell is comprised of several scan lines of the CRT beam, the sequential address output string for a given video row is identically repeated for each scan line within the row. The starting address for each video scan line is stored within an internal 12-bit register called the Row Start Register. At the beginning of each video scan line, the internal address counter logic is preset with the contents of the Row Start Register (see Figure 4). To accomplish row by row sequential addressing, internal logic updates the Row Start Register at the beginning of the first scan line of a video row with the last address + 1 of the last scan line of the previous video row. Since the number of address locations on the video screen display is typically much less than the 4k dimension of the 12-bit address bus, an internal 12-bit register called the Top Of Page Register, contains the starting address of the first video row. Internal logic loads the contents of this top of page register into the Row Start Register at the beginning of the first scan line of the first video row. The Top Of Page Register is loaded with address zero whenever the Reset input is pulsed to the logic "0" state.

In the input mode (disabled), external addresses can be loaded into the internal 12-bit registers by external control of the register select A, register select B, and register $\overline{\text{load}}$ inputs (see Table 1). As a result of specific external loading of the contents of the Row Start Register, Top Of Page Register, and the Cursor Register, row by row page scrolling, non-sequential row control, and cursor location control, can easily be accomplished.

During the non-video intervals, the address output operation is modified. During all horizontal blanking intervals, the incrementing of the address counter is inhibited and the address count is held constant at the last video address + 1. For example, if a video row has an 80 character cell format and addressing for the video portion of a given scan line starts at address 1, the address counter will increment up through address 81. Address 81 is held constant during the horizontal blanking interval until 3 character times before the next video scan line. At this point, the address counter is internally loaded with the contents of the Row Start Register which may contain address 1 or 81 as a function of internal control, or a new address that was loaded from the external bus. During vertical blanking, however, this loading of the internal address counter with the contents of the Row Start Register is inhibited providing scan line by scan line sequential address incrementing. This allows minimum access time to the CRTC when the address counter outputs are being used for dynamic RAM refresh.

RAM Address Enable Input: At all times the status of the bi-directional address outputs is controlled externally by the logic level of the enable input. A 'low' logic level at this input places the address outputs in the TRI-STATE® (disabled) input mode. A 'high' logic level at this input places the address outputs in the active (enabled) output mode.

Register $\overline{\text{Load}}$/Select Inputs: When the Register $\overline{\text{Load}}$ input is pulsed to the logic 'low' state, the Top Of Page, Row Start, or Cursor Register will be loaded with a 12-bit address which originates from either the internal address counter or the external address bus (refer to discussion on register loading constraints). The destination register is selected prior to the load pulse by setting the register select inputs to the appropriate state as defined in Table 1.

Table 1. Register Load Truth Table

Register Select A (Pin 39)	Register Select B (Pin 1)	Register Load Input (Pin 38)	Register Loading Destination
0	0	0	No Select
0	1	0	Top-of-Page
1	0	0	Row-Start*
1	1	0	Cursor
X	X	1	No Load

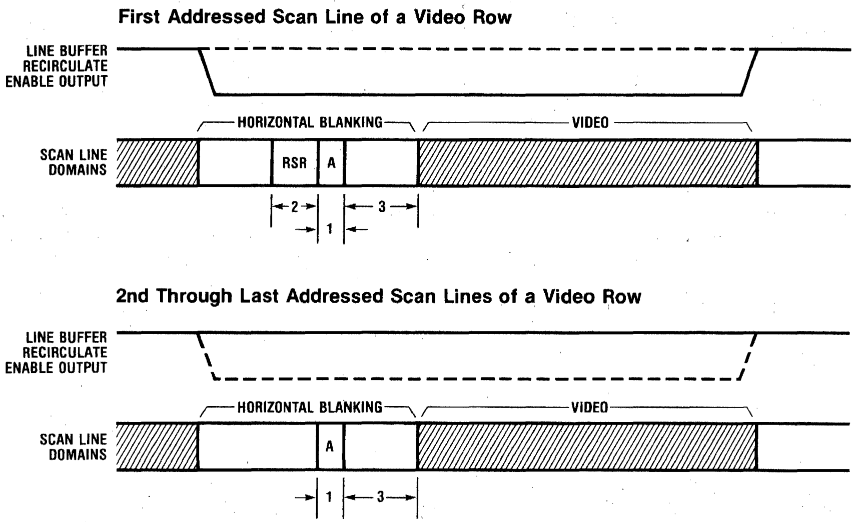
*During the vertical blanking interval, a load to this register is internally routed to the Top-Of-Page register.

Internal Registers and Loading Constraints: There are 3 internal 12-bit registers that facilitate video screen management with respect to row-by-row page scrolling, non-sequential row control and cursor location. These registers can be loaded with addresses from the external address bus while the address outputs are disabled (RAM address enable inut in the low state), by controlling the register select and load inputs within the constraints of each register.

The Row-Start Register (RSR) holds the starting address for each scan line of the video portion of a frame. The video addressing format is completely determined by the contents of this register. With no external loading, the RSR is automatically loaded by internal control such that row-by-row sequential addressing is achieved. Referring to Figure 4, the RSR is loaded automatically once for each video row during the first addressed scan line. The source of the loaded address is internally controlled such that the RSR load for the first video row comes from the Top-Of-Page Register. The RSR load for all subsequent video rows comes from the address counter which holds the last displayed address + 1. If non-sequential row formatting is desired, the RSR can be loaded externally with a 12-bit address. However, this external load must be made prior to the internal automatic load. Generally speaking, the external load to the RSR should be made during the video domain of the last addressed scan line of the previous row. Figure 4 indicates the internal automatic loading intervals which must be avoided, if the load must be made during the horizontal blanking interval. Once an external address has been loaded to the RSR, the next occurring internal automatic RSR load will be inhibited by internal detection logic. If an external load is made to the RSR during the vertical blanking interval, the 12-bit address is loaded into the Top-Of-Page Register instead of the RSR as a result of internal control. This internal function is performed due to the fact that the address loaded into the RSR for the first video row can only come from the Top-Of-Page Register.

The Top-Of-Page register (TOPR) holds the address of the first character of the first video row. As a function of internal control the contents of this register are loaded into the RSR at the beginning of the first addressed scan line of the first video row (see Figure 4). This loading operation is strictly a function of internal control and cannot be overridden by an external load to the RSR. For this reason, any external load to the RSR during the vertical blanking interval is interpreted internally as a TOPR load. When the Reset input is pulsed to the logic "0" state, the TOPR register is loaded with address zero by internal control. This yields a video page display with the first row of sequential addressing beginning at zero. Page scrolling can be accomplished by externally loading a new address into the TOPR. This loading operation can be performed at any time during the frame prior to the interval where the TOPR is loaded automatically into the RSR (see Figure 4). Once the TOPR has been loaded, it does not have to be accessed again until the contents are to be modified.

The Cursor Register (CR) holds the present address of the cursor location. A true comparison of the address counter outputs and the contents of the CR results in a Cursor Enable output signal delayed by two character times. When the $\overline{\text{Reset}}$ input is pulsed to the logic "0" state, the contents of the CR are set to address zero by internal control. Modifying the contents of the CR is accomplished by external loading at any time during this frame. Typically, loading is performed only during intervals when the address outputs are not actively controlling the video display. Once the CR has been loaded, it does not have to be accessed again until the contents are to be modified.



First Addressed Scan Line of a Video Row
LINE BUFFER RECIRCULATE ENABLE OUTPUT
HORIZONTAL BLANKING
VIDEO
SCAN LINE DOMAINS
RSR
A
2
1
3
2nd Through Last Addressed Scan Lines of a Video Row
LINE BUFFER RECIRCULATE ENABLE OUTPUT
HORIZONTAL BLANKING
VIDEO
SCAN LINE DOMAINS
A
1
3

Note 1: Dimensions are in character time intervals.

Note 2: "A" denotes the interval that the address counter is preset with the contents of the Row Start Register.

Note 3: "RSR" denotes the interval that the Row Start Register is internally loaded with either the contents of the Top-Of-Page Register (1st video row) or the last video address + 1 from the address counter.

Figure 4. Automatic Internal Loading Intervals

Video-Related Outputs

Horizontal Sync: This output provides the necessary scan line rate sync signal for direct interface to either three-terminal or composite sync monitors. The pulse width, position, and logic polarity are mask programmable, in character time increments, for custom requirements. This output may also be mask programmed to have RS-170 compatible serration pulses during the vertical sync interval (refer to DP8352 format and Figure 15).

Vertical Sync: This output provides the necessary frame rate sync signal for direct interface to either three-terminal or composite sync monitors. The pulse width, position, and logic polarity are mask programmable, in scan line increments, for custom requirements.

Cursor Enable: This output provides a signal that is intended to be combined with the video signal to display a cursor attribute which serves as a visual pointer for video RAM location. Internally, the 12-bit address count is continuously being compared with the 12-bit address stored in the Cursor Register. When a true compare is detected, an active high level signal will be present at the Cursor Enable output, delayed by 2 character times after the corresponding address bus output. The signal is delayed by 2 character times so that it will be coincident with the video information resulting from the corresponding address. Mask programmability allows the cursor enable output signal to be formatted such that a signal will be outputted for all addressed scan lines of a video character cell or any single scan line of that cell. The cursor enable output signal is inhibited during the horizontal and vertical blanking intervals so that video blanking is maintained. When the addressing is advanced by setting the address mode input (pin 11) in the logic "0" state, the cursor enable signal will also be shifted with respect to the scan line count. Specifically, for a character cell with the cursor output active on all addressed scan lines of the cell, the first scan line of the cursor signal will occur at the last scan line count of the previous video row, and the last scan line count of the addressed character cell will have no cursor output signal. This mode of operation gives rise to a unique situation for the first video row where the first addressed scan line of a character cell has no cursor output signal since its advanced scan line position is inhibited by the vertical blanking interval.

CRT System Control Functions

Refresh Control Input: This input provides a logic level selectable CRT system refresh rate. Typically, this input will select either a 60 Hz or 50 Hz refresh rate to provide geographical marketing flexibility. However, mask programmability provides the capability of a wide range of frequencies for custom requirements. For definition of the input logic truth table and the refresh rate format, refer to Table 2 and the standard device type format tables.

Table 2. Refresh Rate Select Truth Table

Refresh Control (Pin 3) Logic Level	Frame Refresh Rate			
	Symbol	DP8350	DP8352	DP8353
1	f1	60 Hz	60 Hz	60 Hz
0	f0	50 Hz	50 Hz	50 Hz

Vertical Blanking Output: This output provides a signal that transitions at the end of the last video scan line of the last video row and indicates the beginning of the vertical blanking interval. This signal transitions back to the inactive state during the row of scan lines just prior to the first video row. The transition position within this last row of vertical blanking, as well as the active logic polarity, is a function of the particular device format (item 21 of the format tables) or is mask programmable for custom requirements.

Address Mode: When a system utilizes a line buffer shift register, the first scan line of addressing for a row is used to load the shift register. As a result of this loading operation, addressing for a particular row will not begin accessing the video RAM until the second scan line of addressing for the row. It also follows that the first scan line of a row can only exhibit addressed data for the previous video row that is in the shift register. This offset in addressing becomes a problem for character generation designs that output video on the first scan line of a row (with respect to the line counter outputs). The result is invalid data being displayed for the first scan line. One solution would be to utilize a character generation design that began outputting video on the second scan line of a row. However, since most single chip character generators begin video on the first scan line, the DP8350 series CRT controller provides a pin selectable advanced addressing mode which will compensate for addressing shifts resulting from shift register loading. Referring to Table 3, a high logic level at this input will cause addressing to be coincident with the scan line counter positions of a row, and a low logic level at this input will cause addressing to start on the last scan line counter position of the previous row. This shifted alignment of the addressing, with respect to the designated scan lines of a row, is diagrammed in Figure 5. Characteristically, it follows that, when addressing is advanced by one scan line, the Line Buffer Recirculate Enable output and the Cursor Enable output are also advanced by one scan line. This advanced position of the Cursor Enable output may deserve special consideration depending upon the system design.

Table 3. Address Mode Truth Table

Address Mode Input (Pin 11) (Logic Level)	New Row Addressing At Address Outputs and Line Buffer Recirculate Enable Logic Low Level (Scan Line Position)
0	Last scan line of previous row
1	First scan line of row

Full/Half Row Control: This control input is provided for applications that require the option of half-page addressing. As an example, if the normal video page format is 80 characters/row by 24 rows, setting this input to the logic "0" state will cause the video format to become evenly spaced at 80 characters/row by 12 rows. Specifically, when this input is in the logic "0" state, row addressing is repeated for every other row. This yields successive groups of two rows of identical addressing. The second row of addressing, however, has the Load Video Shift Register output and the Cursor Enable output internally inhibited to provide the necessary video blanking. Setting this input to the logic "1" state yields normal frame addressing.

External Character/Line Rate Clock: This input is intended to aid testing of the CRTC and is not meant to be used as an active input in a CRT system. When this input is left open, it is guaranteed not to interfere with normal operation.

Reset Input: This input is provided for power-up synchronization. When brought to the logic "0" state, device operation is halted. Internal logic is set at the beginning of vertical blanking, and the Top-Of-Page Register and the Cursor Register are loaded with address zero. When this input returns to the logic "1" state, device operation resumes at the vertical blanking interval followed by video addressing which begins at zero. This input has hysteresis and may be connected through a resistor to V_{CC} and through a capacitor to ground to accomplish a power-up Reset. The logic "0" state should be maintained for a minimum of 250 ns.

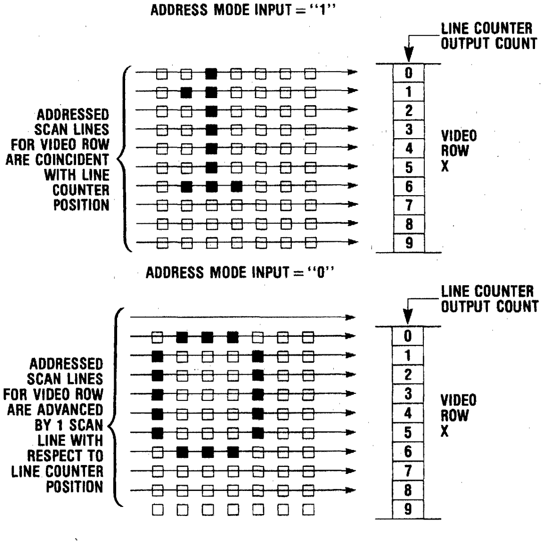

ADDRESS MODE INPUT = "1"
LINE COUNTER OUTPUT COUNT
ADDRESSED SCAN LINES FOR VIDEO ROW ARE COINCIDENT WITH LINE COUNTER POSITION
0
1
2
3
4
5
6
7
8
9
VIDEO ROW X
ADDRESS MODE INPUT = "0"
LINE COUNTER OUTPUT COUNT
ADDRESSED SCAN LINES FOR VIDEO ROW ARE ADVANCED BY 1 SCAN LINE WITH RESPECT TO LINE COUNTER POSITION
0
1
2
3
4
5
6
7
8
9
VIDEO ROW X

Figure 5. Address Mode Functionality

Crystal Inputs X1 and X2: The "Pierce"-type oscillator is controlled by an external crystal providing parallel resonant operation. Connection of external bias components is made to pin 22 (X1) and pin 21 (X2) as shown in Figure 6. It is important that the crystal be mounted in close proximity to the X1 and X2 pins to ensure that printed circuit trace lengths are kept to an absolute minimum. Typical specifications for the crystal are shown in Table 4 for each of the standard products, DP8350, DP8352, and DP8353. When customer mask options require higher frequencies, it may be necessary to change the crystal specifications and biasing components. If the CRTC is to be clocked by an external system dot clock, pin 22 (X1) should be driven directly by Schottky family logic while pin 21 (X2) is left open. The typical threshold for pin 22 (X1) is $V_{CC}/2$.

Table 4. Typical Crystal Specifications

Parameter	Specification		
	DP8350	DP8352	DP8353
Type	At-Cut		
Frequency	10.92 MHz	7.02 MHz	17.6256 MHz
Tolerance	0.005% at 25°C		
Stability	0.01% from 0°C to +70°C		
Resonance	Fundamental, Parallel		
Maximum Series Resistance	50Ω		
Load Capacitance	20 pF		

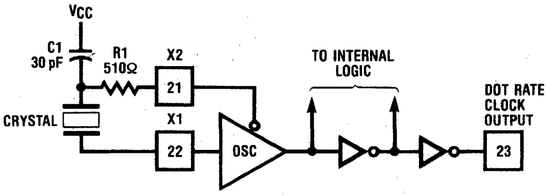


Figure 6. Dot Clock Oscillator Configuration with Typical External Bias Circuitry Shown

Custom Order Mask Programmability: The DP8350 Series CRT controller is available in three standard options designated DP8350, DP8352, and DP8353. The functional format of these devices was selected to meet the typical needs of CRT terminal designs. In order to accommodate specific customer formats, the DP8350 series CRT controller is mask programmable with a diverse range of options available. The items listed in the program table worksheet indicate the available options, while Table 5 tabulates the programming constraints.

Table 5. Mask Programming Limitations

Designation	Parameter	Min. Value	Max. Value
f_{DOT}	Dot Rate Frequency	DC	30 MHz
f_{CHAR}	Character Rate Frequency	DC	2.5 MHz
—	Line Buffer Clock Logic "0" Width (Item 20 x Item 24)	200 ns	
Item 3	Dots per Character Field Width	4	16
Item 4	Scan Lines per Character Field	2	16
Item 12	Scan Lines per Frame		512
Item 14	Character Times per Row Video	5	122
	Blanking	6	123
Item 11	Scan Lines per Vertical Blanking	(Item 4) +2	

If the cursor enable output, Item 22, is active on only one line of a character row, then Item 21 value must be either "1" or "0" or equivalent to the line selected for the cursor enable output.

DP8350 Series Custom Order Format Table

This table is provided as a worksheet to aid in determining the programmed configuration for custom mask options. Refer to Table 5 for a list of programming limitations.

Item No.	Parameter		Value	
1	Character Font Size (Reference Only)	Dots per Character (Width)		
2		Scan Lines per Character (Height)		
3	Character Field Block Size	Dots per Character (Width)		
4		Scan Line per Character (Height)		
5	Number of Video Characters per Row			
6	Number of Video Character Rows per Frame			
7	Number of Video Scan Lines (Item 4 × Item 6)			
8	Frame Refresh Rate (Hz) (two pin selectable frequencies allowed) (Item 13 ÷ Item 12)		f1=	f0=
9	Delay after Vertical Blank start to start of Vertical Sync (Number of Scan Lines)			
10	Vertical Sync Width (Number of Scan Lines)			
11	Interval between Vertical Blank start and start of Video (Number of Scan Lines of Video Blanking)			
12	Total Scan Lines per Frame (Item 7 + Item 11)			
13	Horizontal Scan Frequency (Line Rate) (kHz) (Item 8 × Item 12)			
14	Number of Character Times per Scan Line			
15	Character Clock Rate (MHz) (Item 13 × Item 14)			
16	Character Time (ns) (1 ÷ Item 15)			
17	Delay after Horizontal Blank start to Horizontal Sync start (Character Times)			
18	Horizontal Sync Width (Character Times)			
19	Dot Frequency (MHz) (Item 3 × Item 15)			
20	Dot Time (ns) (1 ÷ Item 19)			
21	Vertical Blanking Output Stop before start of Video (Number of Scan Lines) (Range = Item 4 − 1 line to 0 lines)			
22	Cursor Enable on all Scan Lines of a Row? (Yes or No) If not, which Line?			
23	Does the Horizontal Sync Pulse have Serrations during Vertical Sync? (Yes or No)			
24	Width of Line Buffer Clock logic "0" state within a Character Time (Number of Dot Time increments) (Typically ½ Item 3 rounded up)			
25	Serration Pulse Width, if used (Character Times) (See Figure 13)			
26	Horizontal Sync Pulse Active state logic level (1 or 0)			
27	Vertical Sync Pulse Active state logic level (1 or 0)			
28	Vertical Blanking Pulse Active state logic level (1 or 0)			

Video Monitor: Manufacturer and Model No. (For Engineering Reference)

Absolute Maximum Ratings (Note 1)

Supply Voltage, V_{CC}	7.0V
Input Voltage	5.5V
Output Voltage	5.5V
Storage Temperature Range	−65°C to +150°C
Lead Temperature (soldering, 10 seconds)	300°C

Operating Conditions (Note 6)

	Min.	Max.	Units
V_{CC}, Supply Voltage	4.75	5.25	V
T_A, Ambient Temperature	0	+70	°C

Electrical Characteristics $V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to $+70°C$ (Notes 2, 3, and 5)

Parameter		Conditions	Min.	Typ.	Max.	Units
V_{IH}	Logic "1" Input Voltage All Inputs Except X1, X2 $\overline{\text{RESET}}$ $\overline{\text{RESET}}$		2.0 2.6			V V
V_{IL}	Logic "0" Input Voltage All Inputs Except X1, X2				0.8	V
V_{HYS}	$\overline{\text{RESET}}$ Input Hysteresis			0.4		V
V_{clamp}	Input Clamp Voltage All Inputs Except X1, X2	$I_{IN} = -12\,mA$		−0.8	−1.2	V
I_{IH}	Logic "1" Input Current A_0–A_{11}	Enable Input = 0V, $V_{CC} = 5.25V$, $V_{IN} = 5.25V$		10	100	μA
	All Other Inputs Except X1, X2	$V_{CC} = 5.25V$, $V_{IN} = 5.25V$		2.0	20	μA
I_{IL}	Logic "0" Input Current A_0–A_{11}	Enable Input = 0V, $V_{CC} = 5.25V$, $V_{IN} = 0.5V$		−20	−100	μA
	All Other Inputs Except X1, X2	$V_{CC} = 5.25V$, $V_{IN} = 0.5V$		−20	−100	μA
V_{OH}	Logic "1" Output Voltage	$I_{OH} = -100\,\mu A$	3.2	4.1		V
		$I_{OH} = -1\,mA$	2.5	3.3		V
V_{OL}	Logic "0" Output Voltage	$I_{OL} = 5\,mA$		0.35	0.5	V
I_{OS}	Output Short Circuit Current	$V_{CC} = 5V$, $V_{OUT} = 0V$ (Note 4)	10	40	100	mA
I_{CC}	Power Supply Current (Note 10)	$V_{CC} = 5.25V$		220	300	mA

Note 1: "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: Unless otherwise specified, min./max. limits apply across the 0°C to +70°C temperature range and the 4.75V to 5.25V power supply range. All typical values are for $T_A = 25°C$ and $V_{CC} = 5.0V$ and are intended for reference only.

Note 3: All currents into device pins are shown as positive; all currents out of device pins are shown as negative; all voltages are referenced to ground, unless otherwise specified. All values shown as max. or min. are so classified on absolute value basis.

Note 4: Only one output at a time should be shorted.

Note 5: Electrical specifications do not apply to pin 17, external char/line clock, as this pin is used for production testing only.

Note 6: Functional operation of device is not guaranteed when operated beyond specified operating condition limits.

Switching Characteristics $V_{CC} = 5.0V \pm 5\%$, $T_A = 25°C$ (Note 7)

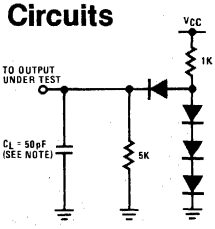
Parameter		Load Circuit	Notes	Min.	Typ.	Max.	Units
Symmetry	Dot Rate Clock Output High Symmetry With Crystal Control	1		50% − 4	50% − 2	50% + 1	ns
t_{pd1}	XI Input to Dot Rate Clock Output Positive Edge	1			17	22	ns
t_{pd0}	XI Input to Dot Rate Clock Output Negative Edge	1			21	26	ns
t_{D1}	Dot Clock to Load Video Shift Register Negative Edge	1			6.0	10	ns
t_{D2}	Dot Clock to Load Video Shift Register Positive Edge	1			11	15	ns
t_{D3}	Dot Clock to Latch Character Generator Positive Edge	1			8.0	13	ns
t_{D4}	Dot Clock to Latch Character Generator Negative Edge	1			6.0	10	ns

Switching Characteristics (Cont'd.) $V_{CC} = 5.0V \pm 5\%$, $T_A = 25°C$ (Note 7)

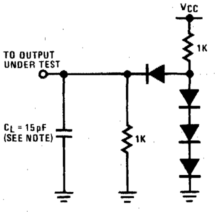
Parameter		Load Circuit	Notes	Min.	Typ.	Max.	Units
t_{D2}-t_{D3}	Latch Character Generator Positive Edge to Load Video Shift Register Positive Edge	1		0	3.0		ns
t_{D5}	Dot Clock to Line Buffer Clock Negative Edge	1			23	35	ns
t_{PW1}	Line Buffer Clock Pulse Width	1	8,9	N(DT)	N(DT)+8	N(DT)+12	ns
t_{D6}	Dot Clock to Cursor Enable Output Transition	1			24	36	ns
t_{D7}	Dot Clock to Valid Address Output	1			15	25	ns
t_{D8_0}	Latch Character Generator to Line Rate Clock Neg. Transition	1	8,10		425 + DT	500 + DT	ns
t_{D8_1}	Latch Character Generator to Line Rate Clock Pos. Transition	1	8,10		300 + DT	400 + DT	ns
t_{D9_0}	Latch Character Generator to Clear Line Counter Neg. Transition	1	8,10		525 + DT	700 + DT	ns
t_{D9_1}	Latch Character Generator to Clear Line Counter Pos. Transition	1	8,10		290 + DT	400 + DT	ns
t_{D8_1}-t_{D9_1}	Clear Line Counter Pos. Transition to Line Rate Clock Pos. Transition	1	10		10	60	ns
t_{D10}	Line Rate Clock to Line Counter Output Transition	1			60	120	ns
t_{D11}	Line Rate Clock to Line Buffer Recirculate Enable Transition	1			195	300	ns
t_{D12}	Line Rate Clock to Vertical Blanking Transition	1			160	300	ns
t_{D13}	Line Rate Clock to Vertical Sync Transition	1			220	300	ns
t_{D14}	Latch Character Generator to Horizontal Sync Transition	1			96	150	ns
t_{S1}	Register Select Set-up Before Register Load Negative Edge			0			ns
t_{H1}	Register Select Hold After Register Load Positive Edge			0			ns
t_{S2}	Valid Address Input Set-Up Before Register Load Positive Edge			250			ns
t_{H2}	Valid Address Hold Time After Register Load Positive Edge			0			ns
t_{PW2}	Register Load Required Pulse Width			150	65		ns
t_{LZ}, t_{HZ}	Delay from Enable Input to Address Output High Impedance State from Logic "0" and Logic "1"	2			15	30	ns
t_{ZL}, t_{ZH}	Delay from Enable Input to Logic "0" and Logic "1" from Address Output High Impedance State	2			17	30	ns

Note 7: Typical values are for $V_{CC} = 5.0V$ and $T_A = 25°C$ and are meant for reference only.
Note 8: "DT" denotes dot rate clock period time, item 20 from option format table.
Note 9: "N" denotes value of item 24 from option format table.
Note 10: Revised since last issue.

Switching Load Circuits



Load Circuit 1



Load Circuit 2

Note: C_L includes probe and jig capacitance. All diodes are 1N914 or equivalent.

Switching Waveforms

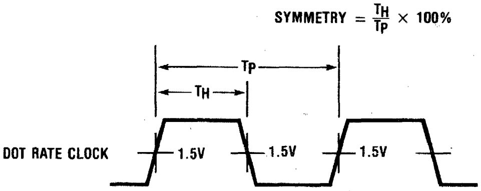


Figure 7. Dot Rate Clock Output Waveform Symmetry with Crystal Control

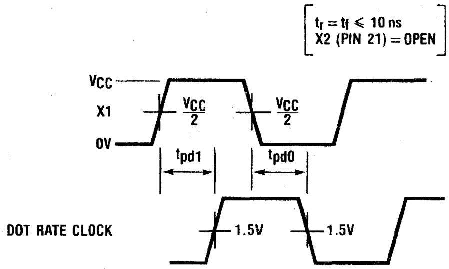
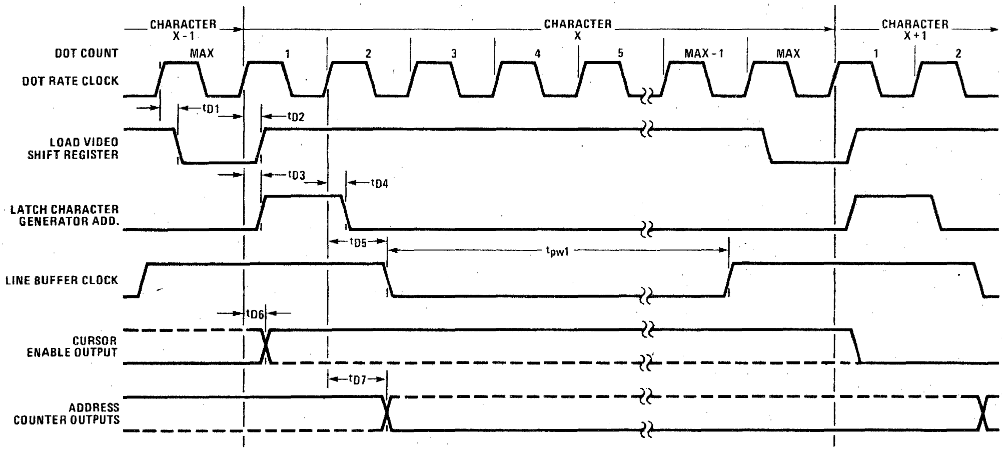
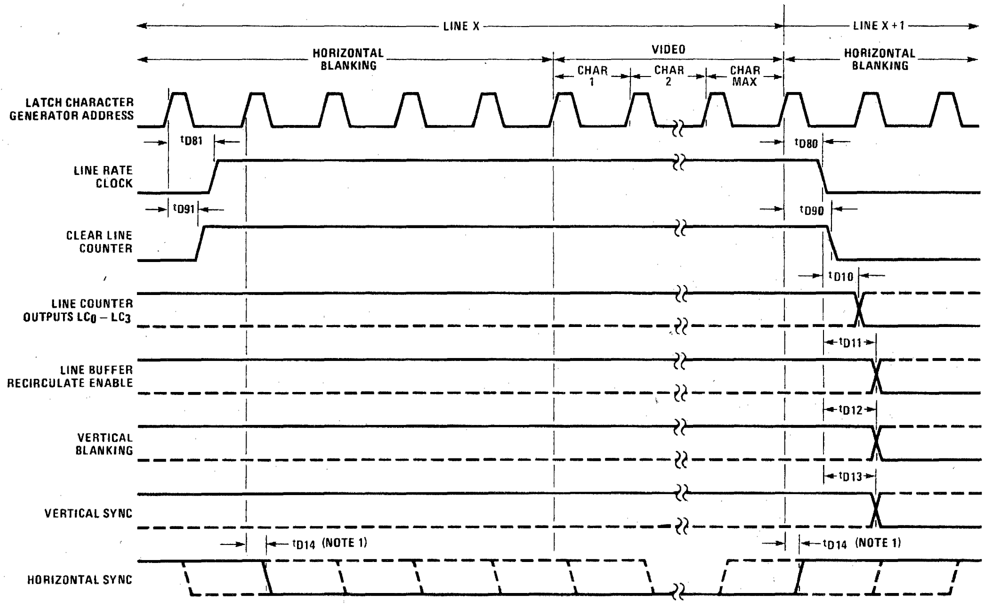


Figure 8. X1 Input to Dot Rate Clock Output Propagation Delay



Note 1: All measurement points are 1.5V

Figure 9. Dot/Character Rate Timing

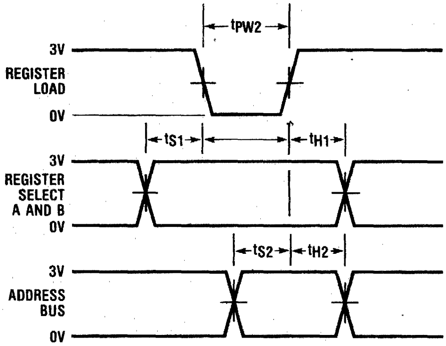


Note 1: Actual polarity and position of the horizontal sync start and stop points is a function of the particular device format.

Note 2: All measurement points are 1.5V.

Figure 10. Character/Line Rate Timing

Switching Waveforms (cont'd)



Note 1: All measurement points are 1.5V.

Note 2: $t_r = t_f \leqslant 10\text{ns}$.

Note 3: Address enable (pin 37) = 0V.

Figure 11. Register Select and Load Waveforms

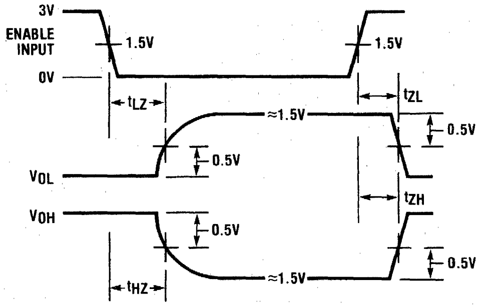
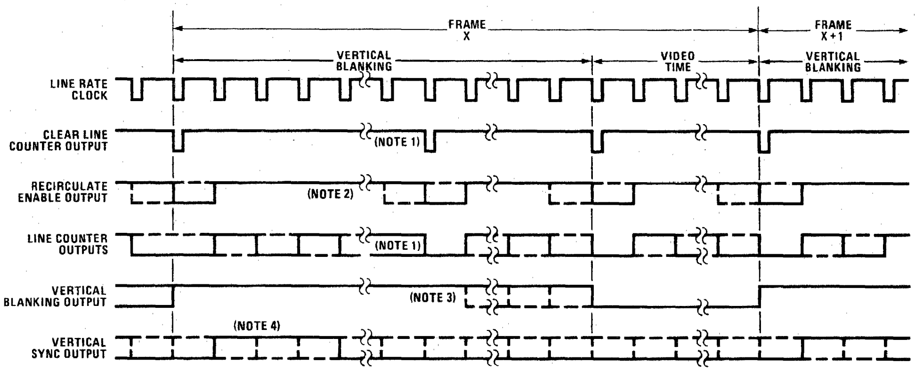


Figure 12. Address Output Enable/Disable Waveforms

Timing Diagrams



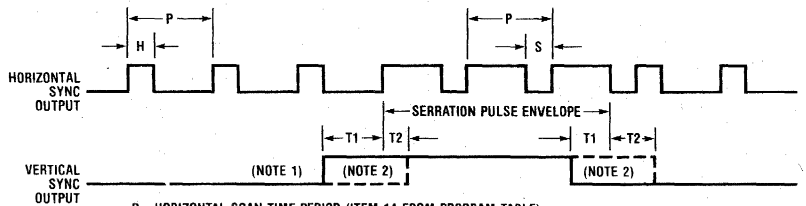
Note 1: One full row before start of video the line counter is set to zero state — this provides line counter synchronization in cases where the number of lines in vertical blanking are not even multiples of the number of lines per row.

Note 2: The position of the line buffer recirculate enable logic low level is a function of the logic level of the address mode input (see Table 3).

Note 3: The stop point of the vertical blanking output active signal is a function of device type or custom option, and will always be within one row prior to video.

Note 4: The transition start and stop points of the vertical sync output signal are a function of device type or custom option.

Figure 14. Line/Frame Rate Functional Diagram



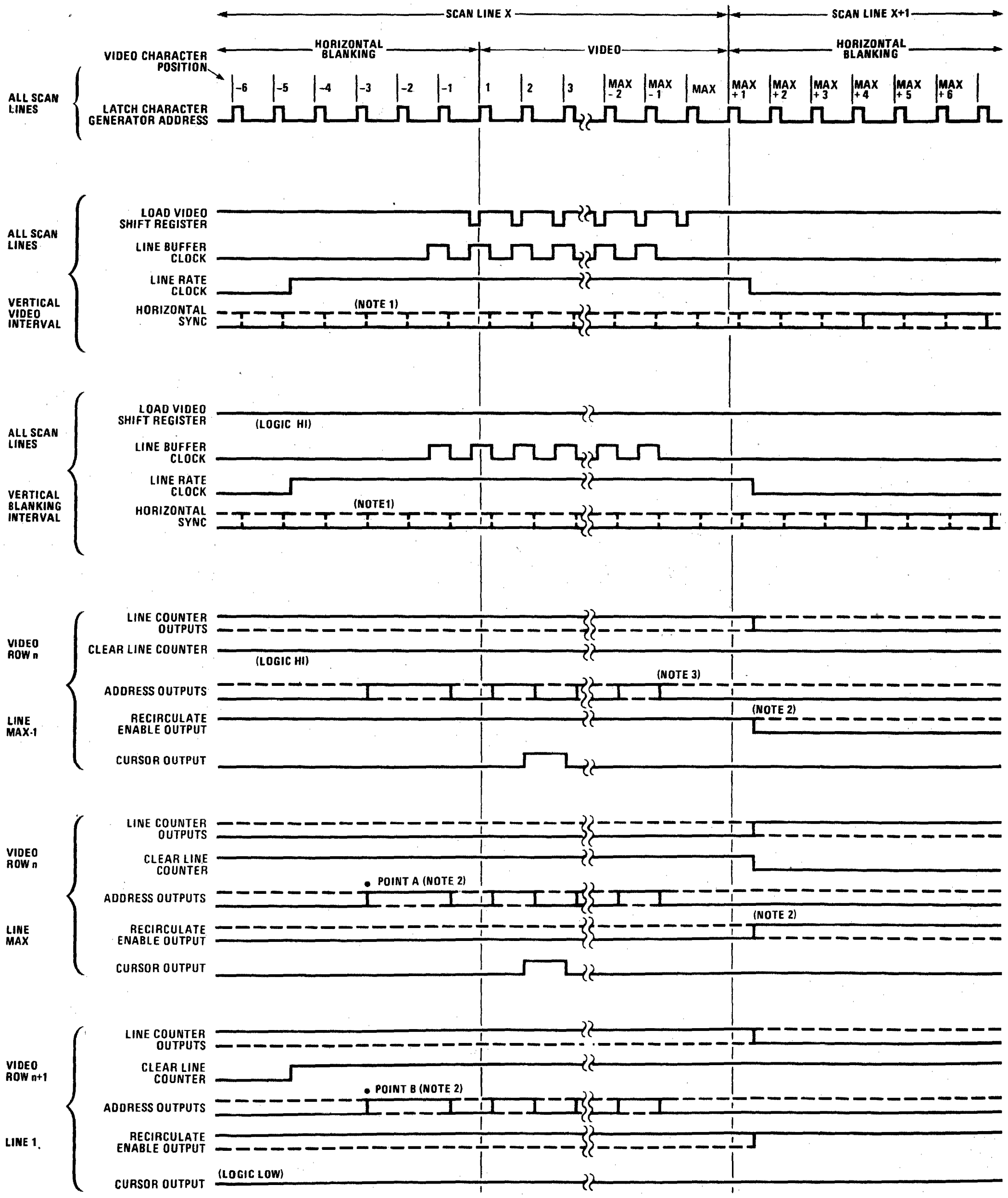
P = HORIZONTAL SCAN TIME PERIOD (ITEM 14 FROM PROGRAM TABLE)
H = HORIZONTAL SYNC WIDTH (ITEM 18 FROM PROGRAM TABLE)
S = SERRATION PULSE WIDTH (ITEM 25 FROM PROGRAM TABLE)
T1 = P-H (MAX)
T2 = H-1 CHARACTER TIME (MAX)

Note 1: The vertical sync transition point is always coincident with the beginning of horizontal blanking.

Note 2: T1 and T2 intervals represent the range of alignment offset between the vertical sync pulse and the serration pulse envelope and is a function of the horizontal sync position with respect to the beginning of horizontal blanking.

Figure 15. Serration Pulse Format

Timing Diagrams (cont'd)



Note 1: The horizontal sync output start and stop point positions are a function of device type or custom option.

Note 2: The position of the recirculate enable output logic "0" level is dependent on the state of the address mode input. When address mode = "0", recirculate enable occurs on the max. line of a character row (solid line) and the address counter outputs roll over to the new row address at point A. When address mode = "1", recirculate enable occurs on the first line of a character row (dashed line) and the address counter outputs roll over to the new row address at point B.

Note 3: The address counter outputs clock to the address of the last character of a video row plus 1. This address is then held during the horizontal blanking interval until video minus three character times. At this point the outputs are modified to the contents of the Row Start Register (RSR).

Figure 13. Character/Line Rate Functional Diagram

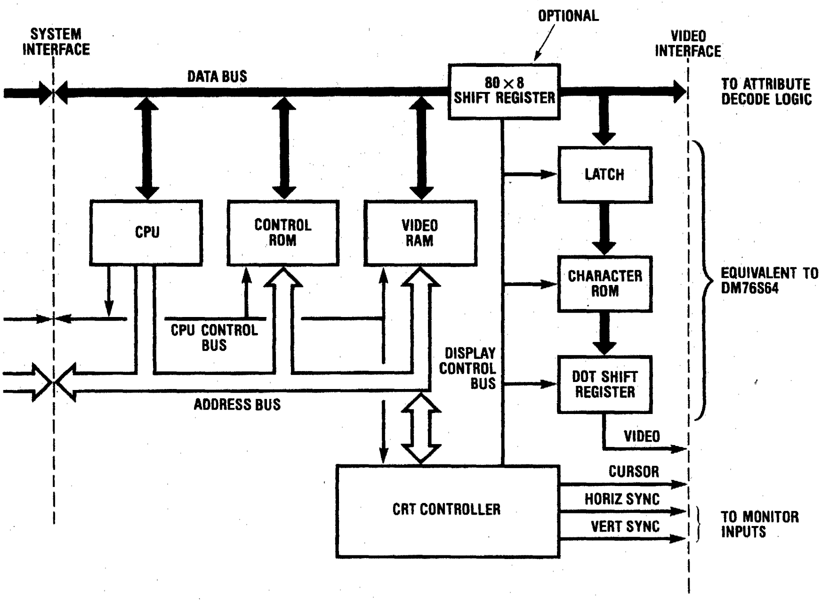


Figure 16. General System Block Diagram

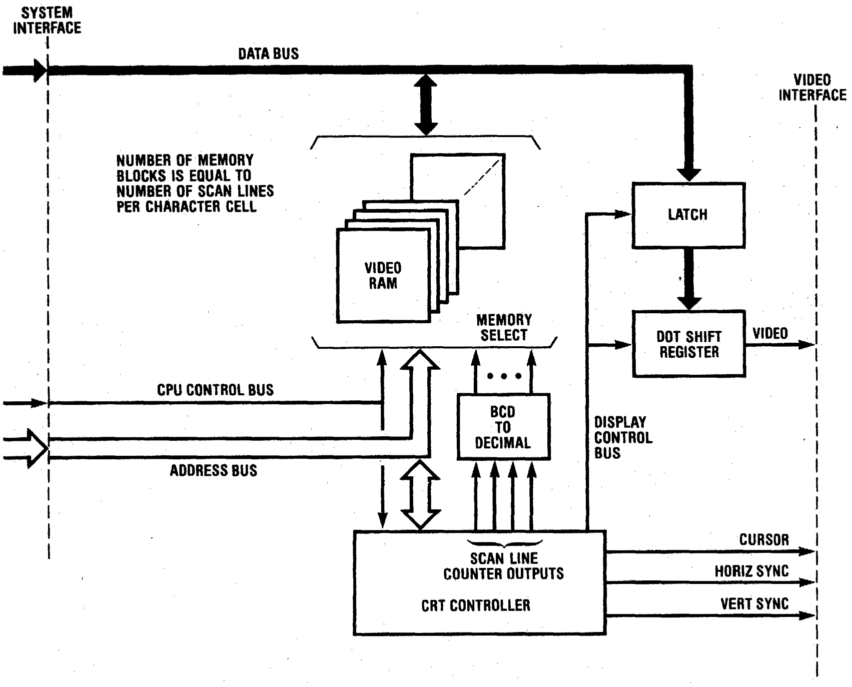


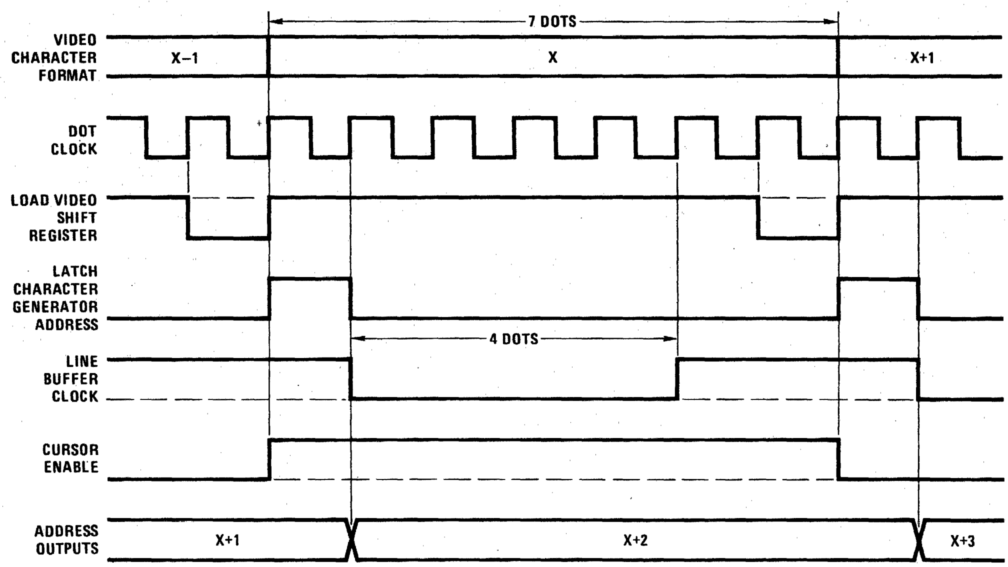
Figure 17. Dot-By-Dot Graphics Block Diagram

DP8350 CRT Controller

Table 6. Characteristic Format

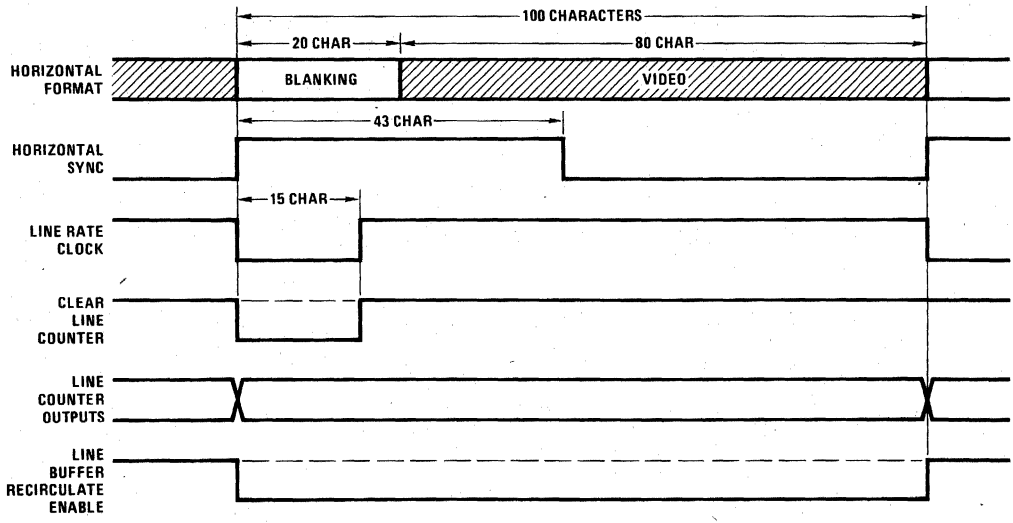
Item No.	Parameter		Value	
1	Character Font Size (Reference Only)	Dots per Character (Width)	(5)	
2		Scan Lines per Character (Height)	(7)	
3	Character Field Cell Size	Dots per Character (Width)	7	
4		Scan Line per Character (Height)	10	
5	Number of Video Characters per Row		80	
6	Number of Video Character Rows per Frame		24	
7	Number of Video Scan Lines (Item 4 × Item 6)		240	
8	Frame Refresh Rate (Hz)		f1 = 60	f0 = 50
9	Delay after Vertical Blank start to start of Vertical Sync (Number of Scan Lines)		4	30
10	Vertical Sync Width (Number of Scan Lines)		10	10
11	Interval between Vertical Blank start and start of Video (Number of Scan Lines of Video Blanking)		20	72
12	Total Scan Lines per Frame (Item 7 + Item 11)		260	312
13	Horizontal Scan Frequency (Line Rate) (Item 8 × Item 12)		15.6kHz	
14	Number of Character Times per Scan Line		100	
15	Character Clock Rate (Item 13 × Item 14)		1.56MHz	
16	Character Time (1 ÷ Item 15)		641ns	
17	Delay after Horizontal Blank start to Horizontal Sync start (Character Times)		0	
18	Horizontal Sync Width (Character Times)		43	
19	Dot Frequency (Item 3 × Item 15)		10.92MHz	
20	Dot Time (1 ÷ Item 19)		91.6ns	
21	Vertical Blanking Output Stop before start of Video (Number of Scan Lines)		1	
22	Cursor Enable on all Scan Lines of a Row? (Yes or No)		Yes	
23	Does the Horizontal Sync Pulse have Serrations during Vertical Sync? (Yes or No)		No	
24	Width of Line Buffer Clock logic "0" state within a Character Time (Number of Dot Time increments)		4	
25	Serration Pulse Width, if used (Character Times)		—	
.26	Horizontal Sync Pulse Active state logic level (1 or 0)		1	
27	Vertical Sync Pulse Active state logic level (1 or 0)		0	
28	Vertical Blanking Pulse Active state logic level (1 or 0)		1	

Video Monitor Format: Ball Brothers TV-12, TV-120 or Equivalent.



NOTE: DASHED LINES IN WAVEFORMS DENOTE INACTIVE STATE LOGIC LEVELS.

Figure 18. DP8350 Video Character Signals



NOTE: DASHED LINES IN WAVEFORMS DENOTE INACTIVE STATE LOGIC LEVELS.

Figure 19. DP8350 Scan Line Signals

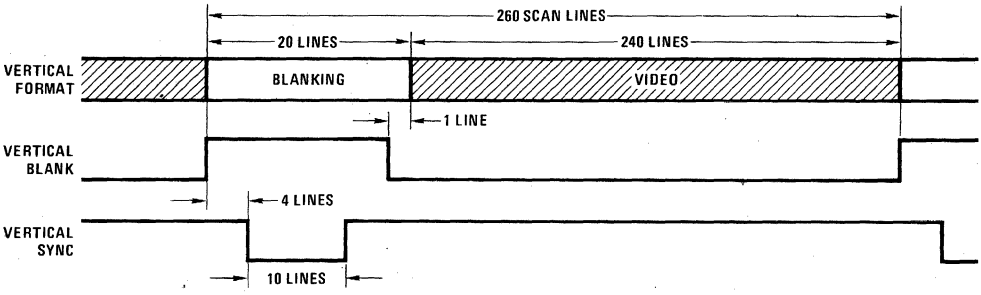


Figure 20. DP8350 60 Hz Refresh Rate Frame Signals

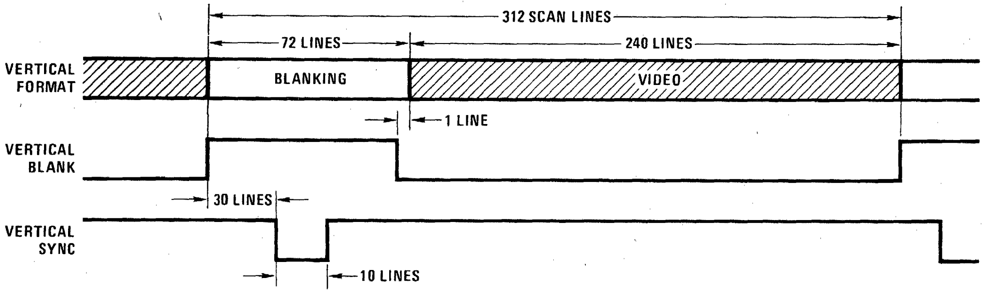


Figure 21. DP8350 50 Hz Refresh Rate Frame Signals

Table 7. Characteristic Format

Item No.	Parameter		Value	
1	Character Font Size (Reference Only)	Dots per Character (Width)	(7)	
2		Scan Lines per Character (Height)	(9)	
3	Character Field Cell Size	Dots per Character (Width)	9	
4		Scan Line per Character (Height)	12	
5	Number of Video Characters per Row		32	
6	Number of Video Character Rows per Frame		16	
7	Number of Video Scan Lines (Item 4 x Item 6)		192	
8	Frame Refresh Rate (Hz)		f1 = 60	f0 = 50
9	Delay after Vertical Blank start to start of Vertical Sync (Number of Scan Lines)		27	53
10	Vertical Sync Width (Number of Scan Lines)		3	3
11	Interval between Vertical Blank start and start of Video (Number of Scan Lines of Video Blanking)		68	120
12	Total Scan Lines per Frame (Item 7 + Item 11)		260	312
13	Horizontal Scan Frequency (Line Rate) (Item 8 x Item 12)		15.6 kHz	
14	Number of Character Times per Scan Line		50	
15	Character Clock Rate (Item 13 x Item 14)		0.78 MHz	
16	Character Time (1 ÷ Item 15)		1282 ns	
17	Delay after Horizontal Blank start to Horizontal Sync start (Character Times)		6	
18	Horizontal Sync Width (Character Times)		4	
19	Dot Frequency (Item 3 x Item 15)		7.02 MHz	
20	Dot Time (1 ÷ Item 19)		142.4 ns	
21	Vertical Blanking Output Stop before start of Video (Number of Scan Lines)		0	
22	Cursor Enable on all Scan Lines of a Row? (Yes or No)		Yes	
23	Does the Horizontal Sync Pulse have Serrations during Vertical Sync? (Yes or No)		Yes	
24	Width of Line Buffer Clock logic "0" state within a Character Time (Number of Dot Time increments)		5	
25	Serration Pulse Width, if used (Character Times)		4	
26	Horizontal Sync Pulse Active state logic level (1 or 0)		0	
27	Vertical Sync Pulse Active state logic level (1 or 0)		0	
28	Vertical Blanking Pulse Active state logic level (1 or 0)		1	

Video Monitor Format: RS-170-Compatible (Standard American TV).

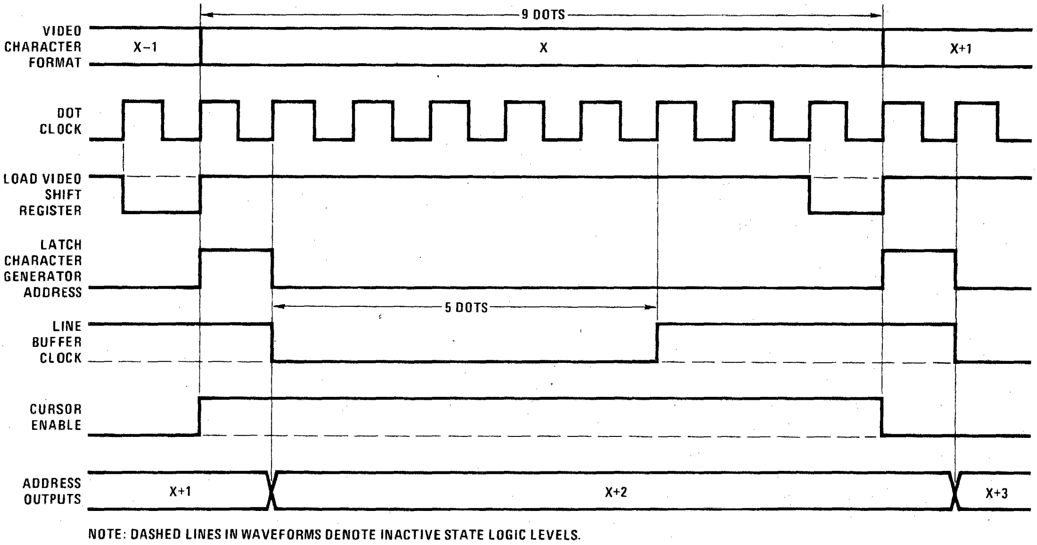

9 DOTS
VIDEO CHARACTER FORMAT
X−1
X
X+1
DOT CLOCK
LOAD VIDEO SHIFT REGISTER
LATCH CHARACTER GENERATOR ADDRESS
5 DOTS
LINE BUFFER CLOCK
CURSOR ENABLE
ADDRESS OUTPUTS
X+1
X+2
X+3
NOTE: DASHED LINES IN WAVEFORMS DENOTE INACTIVE STATE LOGIC LEVELS.

Figure 22. DP8352 Video Character Signals

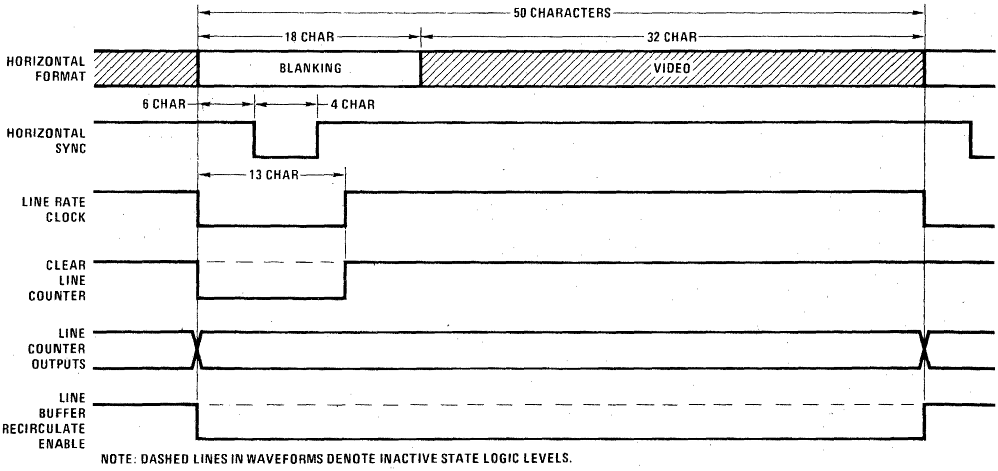


Figure 23. DP8352 Scan Line Signals

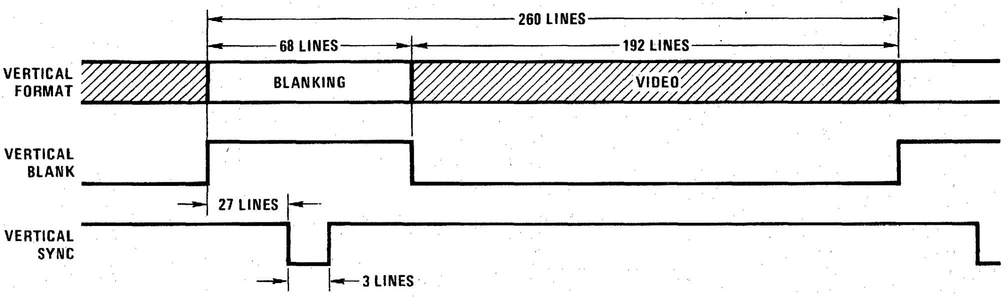


Figure 24. DP8352 60 Hz Refresh Rate Frame Signals

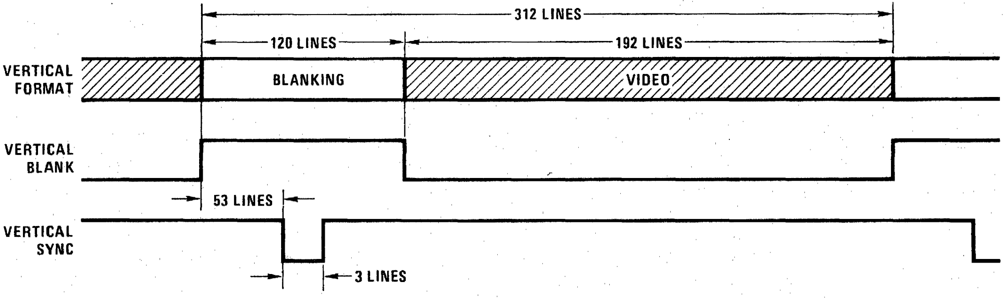


Figure 25. DP8352 50 Hz Refresh Rate Frame Signals

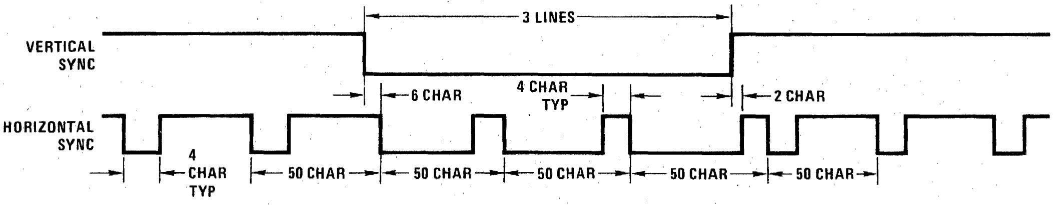


Figure 26. DP8352 Serration Pulse Format

DP8353 CRT Controller

Table 8. Characteristic Format

Item No.	Parameter		Value	
1	Character Font Size (Reference Only)	Dots per Character (Width)	(7)	
2		Scan Lines per Character (Height)	(9)	
3	Character Field Cell Size	Dots per Character (Width)	9	
4		Scan Line per Character (Height)	12	
5	Number of Video Characters per Row		80	
6	Number of Video Character Rows per Frame		25	
7	Number of Video Scan Lines (Item 4 x Item 6)		300	
8	Frame Refresh Rate (Hz)		f1 = 60	f0 = 50
9	Delay after Vertical Blank start to start of Vertical Sync (Number of Scan Lines)		0	32
10	Vertical Sync Width (Number of Scan Lines)		3	3
11	Interval between Vertical Blank start and start of Video (Number of Scan Lines of Video Blanking)		20	84
12	Total Scan Lines per Frame (Item 7 + Item 11)		320	384
13	Horizontal Scan Frequency (Line Rate) (Item 8 x Item 12)		19.20 kHz	
14	Number of Character Times per Scan Line		102	
15	Character Clock Rate (Item 13 x Item 14)		1.9584 MHz	
16	Character Time (1 ÷ Item 15)		510.6 ns	
17	Delay after Horizontal Blank start to Horizontal Sync start (Character Times)		5	
18	Horizontal Sync Width (Character Times)		9	
19	Dot Frequency (Item 3 x Item 15)		17.6256 MHz	
20	Dot Time (1 ÷ Item 19)		56.7 ns	
21	Vertical Blanking Output Stop before start of Video (Number of Scan Lines)		1	
22	Cursor Enable on all Scan Lines of a Row? (Yes or No)		Yes	
23	Does the Horizontal Sync Pulse have Serrations during Vertical Sync? (Yes or No)		No	
24	Width of Line Buffer Clock logic "0" state within a Character Time (Number of Dot Time increments)		5	
25	Serration Pulse Width, if used (Character Times)		—	
26	Horizontal Sync Pulse Active state logic level (1 or 0)		1	
27	Vertical Sync Pulse Active state logic level (1 or 0)		1	
28	Vertical Blanking Pulse Active state logic level (1 or 0)		1	

Video Monitor Format: Motorola M3003 or Equivalent.

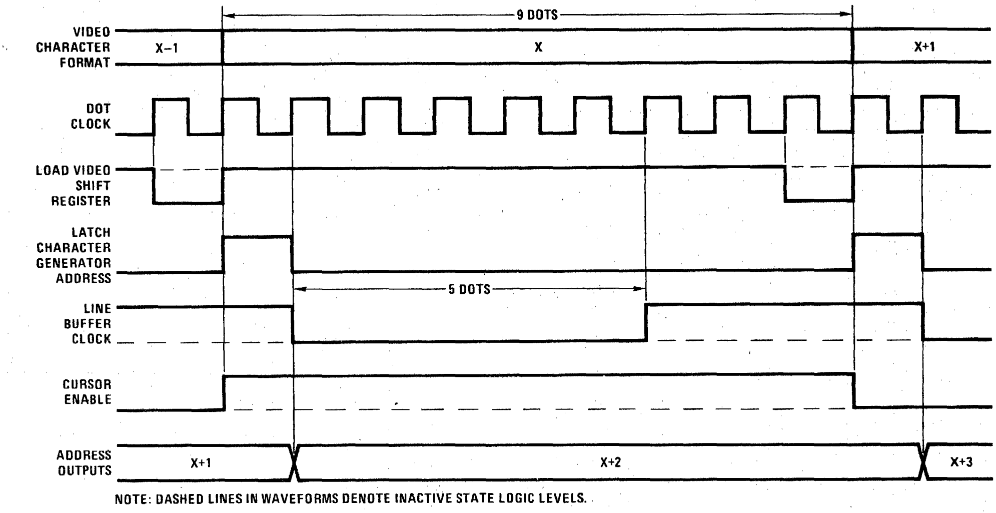

9 DOTS
VIDEO CHARACTER FORMAT
X−1
X
X+1
DOT CLOCK
LOAD VIDEO SHIFT REGISTER
LATCH CHARACTER GENERATOR ADDRESS
5 DOTS
LINE BUFFER CLOCK
CURSOR ENABLE
ADDRESS OUTPUTS
X+1
X+2
X+3
NOTE: DASHED LINES IN WAVEFORMS DENOTE INACTIVE STATE LOGIC LEVELS.

Figure 27. DP8353 Video Character Signals

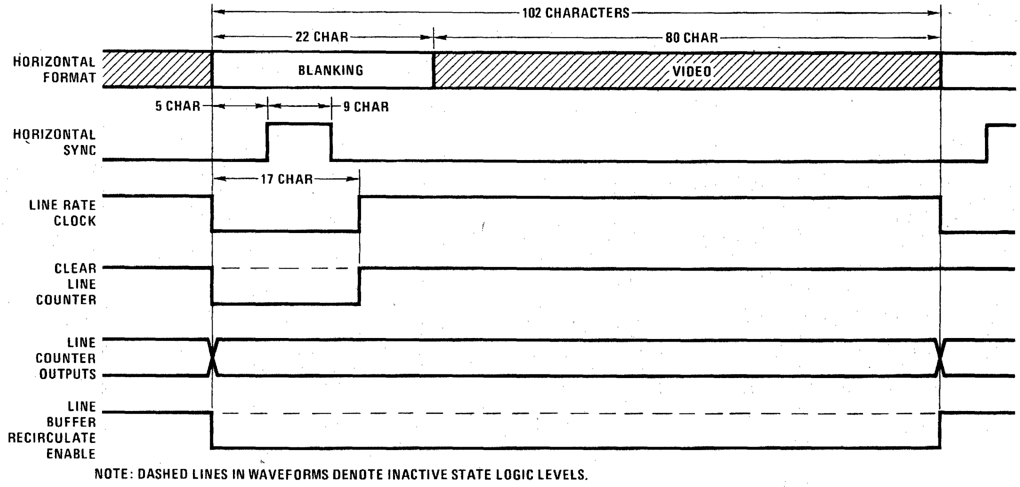


Figure 28. DP8353 Scan Line Signals

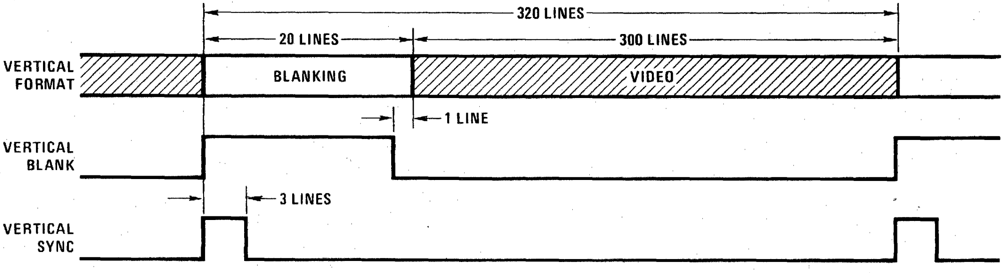


Figure 29. DP8353 60 Hz Refresh Rate Frame Signals

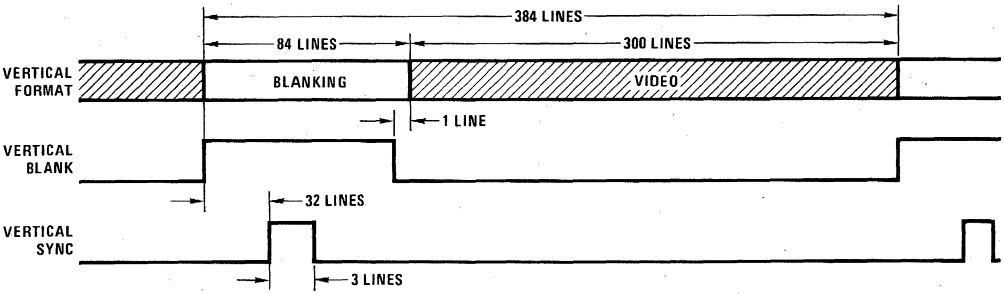
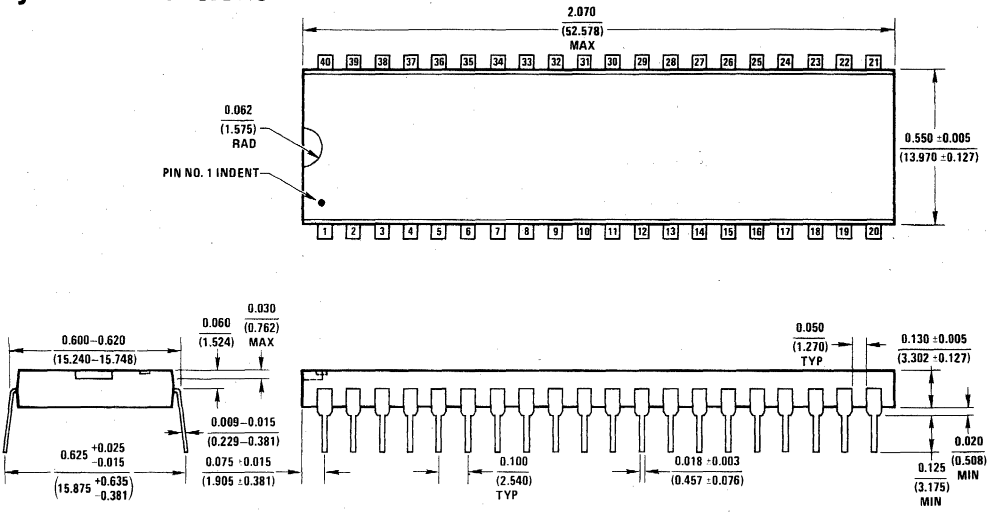


Figure 30. DP8353 50 Hz Refresh Rate Frame Signals

Physical Dimensions



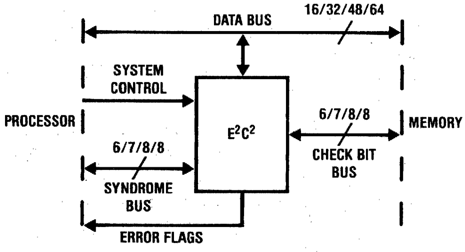
40-Lead Molded DIP (N)
NS Package Number N40A

National Semiconductor

DP8400 — E^2C^2 Expandable Error Checker and Corrector

General Description

The DP8400 Expandable Error Checker and Corrector (E^2C^2) aids system reliability and integrity by detecting errors in memory data and correcting single or double-bit errors. The E^2C^2 data I/O port sits across the processor-memory data bus as shown, and the check bit I/O port connects to the memory check bits. Error flags are provided, and a syndrome I/O port is available. Fabricated using high speed Schottky technology in a 48-pin dual-in-line package, the DP8400 has been designed such that its internal delay times are minimal, maintaining maximum memory performance.



For a 16-bit word, the DP8400 monitors data between the processor and memory, with its 16-bit bidirectional data bus connected to the memory data bus. The DP8400 uses an encoding matrix to generate 6 check bits from the 16 bits of data. In a WRITE cycle, the data word and the corresponding check bits are written into memory. When the same location of memory is subsequently read, the E^2C^2 generates 6 new check bits from the memory data and compares them with the 6 check bits read from memory to create 6 syndrome bits. If there is a difference (causing some syndrome bits to go high), then that memory location contains an error and the DP8400 indicates the type of error with 3 error flags. If the error is a single-bit error, the DP8400 will automatically correct it.

The DP8400 is easily expandable to other data configurations. For a 32-bit data bus with 7 check bits, two DP8400s can be used in cascade with no other ICs. Three DP8400s can be used for 48 bits, and four DP8400s for 64 data bits, both with 8 check bits. In all these configurations, single and double-error detection and single-error correction are easy to implement.

When the memory is more unreliable, or better system integrity is preferred, then in any of these configurations, double-error correction can be performed. One approach requires a further memory WRITE-READ cycle using complemented data and check bits from the DP8400. If at least one of the two errors is a hard error, the DP8400 will correct both errors. This implementation requires no more memory check bits or DP8400s than the single-error correct configurations.

The DP8400 has a separate syndrome I/O bus which can be used for error logging or error management. In addition, the DP8400 can be used in BYTE-WRITE applications (for up to 72 data bits) because it has separate byte controls for the data buffers. In 16 or 32-bit systems, the DP8400 will generate and check system byte parity, if required, for integrity of the data supplied from or to the processor. There are three latch controls to enable latching of data in various modes and configurations.

Operational Features

- Fast single and double-error detection
- Fast single-error correction
- Double-error correction after catastrophic failure with no additional ICs or check bits
- Functionally expandable to 100% double-error correct capability
- Functionally expandable to triple-error detect
- Directly expandable to 32 bits using 2 DP8400s only
- Directly expandable to 48 bits using 3 DP8400s only
- Directly expandable to 64 bits using 4 DP8400s only
- Expandable to and beyond 64 bits in fast configuration with extra ICs
- 3 error flags for complete error recording
- 3 latch enable inputs for versatile control
- Byte parity generating and checking
- Separate byte controls for outputting data in BYTE-WRITE operation
- Separate syndrome I/O port accessible for error logging and management
- On-chip input and output latches for data bus, check bit bus and syndrome bus
- Diagnostic capability for simulating check bits
- Memory check bit bus, syndrome bus, error flags and internally generated syndromes available on the data bus
- Self-test of E^2C^2 on the memory card under processor control
- Full diagnostic check of memory with the E^2C^2
- Complete memory failure detectable
- Power-on clears data and syndrome input latches

Timing Features

16-BIT CONFIGURATION

WRITE Time: 35 ns from data-in to check bits valid
DETECT Time: 35 ns from data-in to Any Error (AE) flag set
CORRECT Time: 70 ns from data-in to correct data out

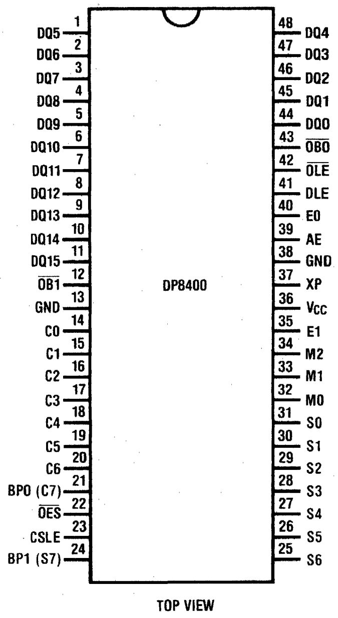
Timing Features (Continued)

32-BIT CONFIGURATION

WRITE Time: 65 ns from data-in to check bits valid
DETECT Time: 60 ns from data-in to Any Error (AE) flag set
CORRECT Time: 125 ns from data-in to correct data out

DP8400 Connection Diagram

Dual-In-Line Package



Order Number DP8400N-4 or DP8400D-4
See NS Package N48A or D48A

Pin Definitions See *Figure 1* for abbreviations

V_{CC}, GND, GND: 5.0V ± 5%. The 3 supply pins have been assigned to the center of the package to reduce voltage drops, both DC and AC. Also there are two ground pins to reduce the low-level noise. The second ground pin is located two pins from V_{CC}, so that decoupling capacitors can be inserted directly next to these pins. It is important to adequately decouple this device, due to the high switching currents that will occur when all 16 data bits change in the same direction simultaneously. A recommended solution would be a 1 μF multilayer ceramic capacitor in parallel with a low-voltage tantalum capacitor, both connected close to pins 36 and 38 to reduce lead inductance.

DQ0–DQ15: Data I/O port. 16-bit bidirectional data bus which is connected to the input of DIL0 and DIL1 and the output of DOB0 and DOB1, with DQ8–DQ15 also to CIL.

C0–C6: Check-bit I/O port. 7-bit bidirectional bus which is connected to the input of the CIL and the output of the COB. COB is enabled whenever M2 is low.

S0–S6: Syndrome I/O port. 7-bit bidirectional bus which is connected to the input of the SIL and the output of the SOB.

DLE: Input data latch enable. When high, DIL0 and DIL1 outputs follow the input data bus. When low, DIL0 and DIL1 latch the input data.

CSLE: Input check bit and syndrome latch enable. When high, CIL and SIL follow the input check and syndrome bits. When low, CIL and SIL latch the input check and syndrome bits. If $\overline{\text{OES}}$ is low, SIL remains latched.

$\overline{\text{OLE}}$: Output latch enable. $\overline{\text{OLE}}$ enables the internally generated data to DOL0, and DOL1, COL and SOL when low, and latches when high.

XP: Multi-expansion, which feeds into a three-level comparator. With XP at 0V, only 6 or 7 check bits are available for expansion up to 40 bits, allowing byte parity capability. With XP open or at V_{CC}, expansion beyond 40 bits is possible, but byte parity capability is no longer available. When XP is at V_{CC}, CG6 and CG7, the internally generated upper two check bits, are set low. When XP is open, CG6 and CG7 are set to word parity.

BP0 (C7): When XP is at 0V, this pin is byte-0 parity I/O. In the Normal WRITE mode, BP0 receives system byte-0 parity, and in the Normal READ mode outputs system byte-0 parity. When XP is open or at V_{CC}, this pin becomes C7 I/O, the eighth check bit for the memory check bits, for 48-bit expansion and beyond.

BP1 (S7): When XP is at 0V, this pin is byte-1 parity I/O. In the Normal WRITE mode, BP1 receives system byte-1 parity, and in the Normal READ mode outputs system byte-1 parity. When XP is open or at V_{CC}, this pin becomes S7 I/O, the eighth syndrome bit for 48-bit expansion and beyond.

AE: Any error. In the Normal READ mode, when low, AE indicates no error and when high, indicates that an error has occurred. In any WRITE mode, AE is permanently low.

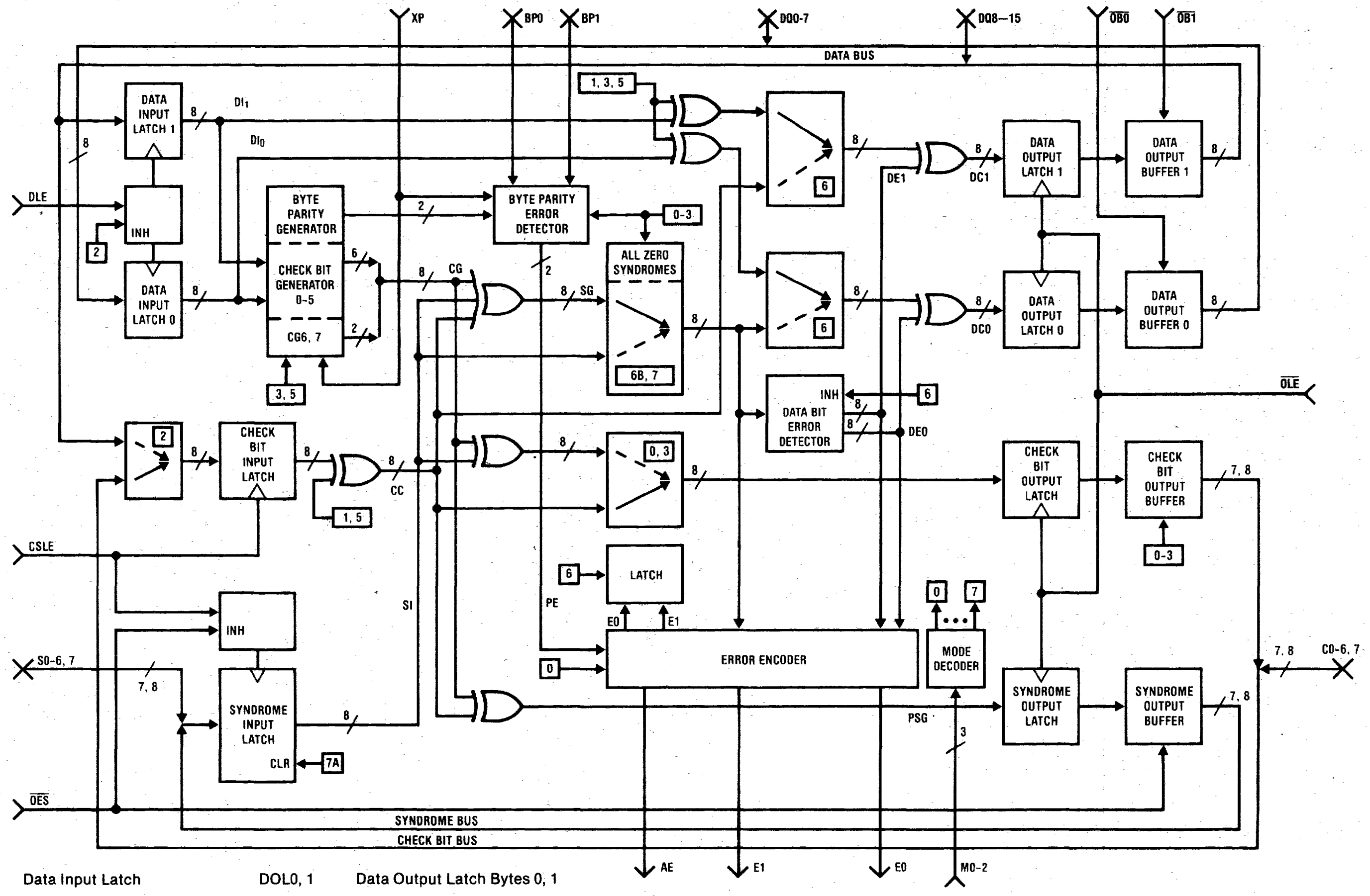
E0: In the Normal READ mode, E0 is high for a single-data error, and low for other conditions. In the Normal WRITE mode, E0 becomes $\overline{\text{PE0}}$ and is low if a parity error exists in byte-0 as transmitted from the processor.

E1: In the Normal READ mode, E1 is high for a single-data error or a single check bit error, and low for no error and double-error. In the Normal WRITE mode, E1 becomes $\overline{\text{PE1}}$ and is low if a parity error exists in byte-1 as transmitted from the processor.

$\overline{\text{OB0}}$, $\overline{\text{OB1}}$: Output byte-0 and output byte-1 enables. These inputs, when low, enable DOL0 and DOL1 through DOB0 and DOB1 onto the data bus pins DQ0–DQ7 and DQ8–DQ15. When $\overline{\text{OB0}}$ and $\overline{\text{OB1}}$ are high the DOB0, DOB1 outputs are TRI-STATE®.

$\overline{\text{OES}}$: Output enable syndromes. I/O control of the syndrome latches. When high, SOB is TRI-STATED and external syndromes pass through the syndrome input latch with CSLE high. When $\overline{\text{OES}}$ is low, SOB is enabled and the generated syndromes appear on the syndrome bus, also CSLE is inhibited internally to SIL.

M0, M1, M2: Mode control inputs. These three controls define the eight major operational modes of the DP8400. Table III depicts the modes.



DIL	Data Input Latch	DOL0, 1	Data Output Latch Bytes 0, 1
CG	Check Bit Generator	COL	Check Bit Output Latch
CIL	Check Bit Input Latch	SOL	Syndrome Output Latch
CC	Check Bit Complementor	DOB0, 1	Data Output Buffer Bytes 0, 1
SIL	Syndrome Input Latch	COB	Check Bit Output Buffer
PSG	Partial Syndrome Generator	SOB	Syndrome Output Buffer
SG	Syndrome Generator	EE	Error Encoder
DED	Data Error Detector	DC0, 1	Data Corrector Bytes 0, 1
DE0, 1	Data Error Bytes 0, 1		
PE	Parity Error		

$\boxed{n}$ mode of operation signifies active signal

FIGURE 1. DP8400 Block Diagram

SYSTEM WRITE *(Figure 2a)*

The Normal WRITE mode is mode 0 of Table III. Referring to the block diagram in *Figure 9a* and the timing diagram of *Figure 9b,* the 16 bits of data from the processor are enabled into the data input latches, DIL0 and DIL1, when the input data latch enable (DLE) is high. When this goes low, the input data is latched. The check bit generator (CG) then produces 6 parity bits, called check bits. Each parity bit monitors different combinations of the input data-bits. In the 16-bit configuration, assuming no syndrome bits are being fed in from the syndrome bus into the syndrome input latch, the 6 check bits enter the check bit output latch (COL), when the output latch enable $\overline{\text{OLE}}$ is low, and are latched in when $\overline{\text{OLE}}$ goes high. Whenever M2 (READ/$\overline{\text{WRITE}}$) is low, the check bit output buffer COB always enables the COL contents onto the external check bit bus. Also the data error decoder (DED) is inhibited during $\overline{\text{WRITE}}$ so no correction can take place. Data output latches DOL0 and DOL1, when enabled with $\overline{\text{OLE}}$, will therefore see the contents of DIL0 and DIL1. If valid system data is still on the data bus, a memory WRITE will write to memory the data on the data bus and the check bits output from COB. If the system has vacated the data bus, output enables ($\overline{\text{OB0}}$ and $\overline{\text{OB1}}$) must be set low so that the original data word with its 6 check bits can be written to memory.

SYSTEM READ

There are two methods of reading data: the error monitoring method *(Figure 2b)*, and the always correct method *(Figure 2c)*. Both require fast error detection, and the second, fast correction. With the first method, the memory data is only monitored by the E^2C^2, and is assumed to be correct. If there is an error, the Any Error flag (AE) goes high, requiring further action from the system to correct the data. With the always correct method, the memory data is assumed to be possibly in error. Memory data is removed and the corrected, or already correct, data is output from the E^2C^2 by enabling $\overline{\text{OB1}}$ and $\overline{\text{OB0}}$. To detect an error (referring to *Figures 10a and 10b*) first DLE and CSLE

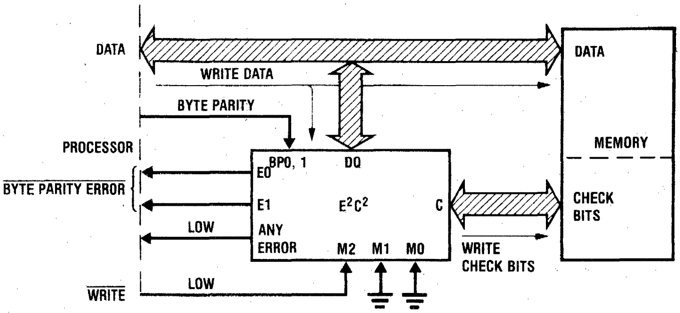


FIGURE 2a. Normal WRITE Mode with E^2C^2

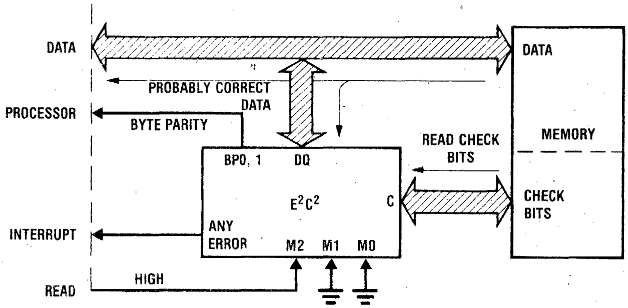


FIGURE 2b. Normal READ Mode, Error Monitoring Method with E^2C^2

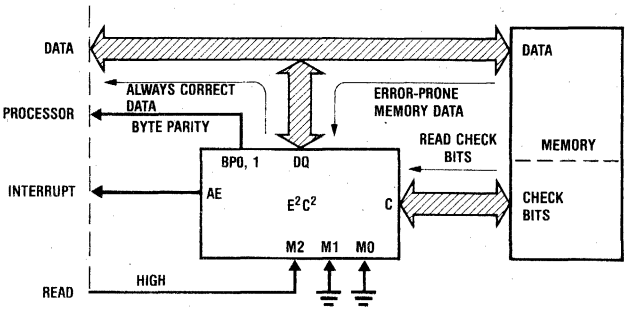


FIGURE 2c. Normal READ Mode, Always Correct Method with E^2C^2

go high to enter data bits and check bits from memory into DILO, DOL1 and CIL. The 6 check bits generated in CG from DILO and DOL1 are then compared with CIL to generate syndromes on the internal syndrome bus (SG). Any bit or bits of SG that go high indicate an error to the error encoder (EE).

If data correction is required $\overline{\text{OB0}}$ and $\overline{\text{OB1}}$ must be set low (after memory data has been disabled) to enable data output buffers DOB0 and DOB1. The location of any data bit error is determined by the data error decoder (DED), from the syndrome bits. The bit in error is complemented in the DOL for correction. The other 15 bits from DED pass the DIL contents directly to the DOL, so that DOL now contains corrected data.

ERROR DETERMINATION

The three error flags, for a 16-bit example, are decoded from the internally generated syndromes as shown in *Figure 3*. First, if any error has occurred, the generated check bits will be different from the memory check bits, causing some of the syndrome bits to go high. By OR-ing the syndrome bits, the output will be an indication of any error.

If there is a single-data error, then (from the matrix in Table IV) it can be seen that any data error causes either 3 or 5 syndrome bits to go high. 16 AND gates decode which bit is in error and the bit in error is XOR-ed with the corresponding bit of the DIL to correct it, whereas the other 15 decoder outputs are low, causing the corresponding 15 bits in DIL to transfer to DOL directly. DOL now contains corrected data. The 16 AND gate outputs are OR-ed together causing E0 to go high, so that E0 is the single-data-error indication. If the error is a double-error, then either 2, 4 or 6 of the syndrome bits will be high. The syndromes for two errors (including one or two check bit errors) are the two sets of syndromes for each individual error bit, XOR-ed together. By performing a parity check on the syndrome bits, flag E1 will indicate $\overline{\text{even}}$/odd parity. If there is still an error, but it is not one of these errors, then it is a detectable triple-bit error. Some triple-bit errors are not detectable as such and may be interpreted as single-bit errors and falsely corrected as single-data errors. This is true for all standard ECC circuits using a Modified Hamming-code matrix. The DP8400 is capable, with its Rotational Syndrome Word Generator matrix, of determining all triple-bit errors using twice as many DP8400s and twice as many check bits.

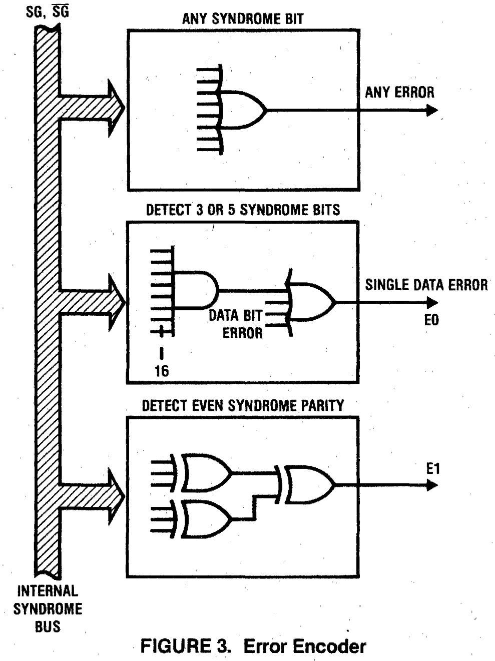


FIGURE 3. Error Encoder

ERROR FLAGS

Three error flags are provided to allow full error determination. Table I shows the error flag outputs for the different error types in Normal READ mode. If there is an error, then ANY ERROR will go high, at a time t_{DEV} *(Figure 10b)* after data and check bits are presented to the DP8400. The other two error flags E0 and E1 become valid t_{DE0} and t_{DE1} later.

The error flags differentiate between no error single check bit error, single data-bit error, double-bit error. Because the DP8400 can correct double errors, it is important to know that two errors have occurred, and not just a multiple-error indication. The error flags will remain valid as long as DLE and CSLE are low, or if DLE is high, and data and check bits remain valid.

BYTE PARITY SUPPORT

Some systems require extra integrity for transmission of data between the different cards. To achieve this, individual byte parity bits are transmitted with the data bits in both directions. The DP8400 offers byte parity support for up to 40 data bits. If the processor generates byte parity when transferring information to the memory, during the WRITE cycle, then each byte parity bit can be connected to the corresponding byte parity I/O pin on the DP8400, either BP0 or BP1. The DP8400 develops its own internal byte parity bits from the two bytes of data from the processor, and compares them with BP0 and BP1 using an exclusive-OR for both parities. The output of each exclusive-OR is fed to the error flags E0 and E1 as $\overline{\text{PE0}}$ and $\overline{\text{PE1}}$, so that a byte parity error forces its respective error flag low, as in Table II. These flags are only valid for the Normal WRITE (mode 0) and XP at 0V. The DP8400 checks and generates even byte parity.

When transferring information from the memory to the processor, the DP8400 receives the memory data, and outputs the corresponding byte parity bits on BP0 and BP1 to the processor. The processor block can then check data integrity with its own byte parity generator. If in fact memory data was in error, the DP8400 derives BP0 and BP1 from the memory input data, and not the corrected data, so when corrected data is output from the DP8400, the processor will detect a byte parity error.

If correct byte parity is required, transfer of corrected output data in the DOL to DIL will result in correct byte parity at BP0 and BP1. This can be part of a normal memory reWRITE cycle once an error has occurred.

TABLE I. ERROR FLAGS AFTER NORMAL READ (MODE 4)

AE	E1	E0	Error Type
0	0	0	No error
1	1	0	Single check bit error
1	1	1	Single-data error
1	0	0	Double-bit error
All Others			Invalid conditions

TABLE II. ERROR FLAGS AFTER NORMAL WRITE (MODE 0)

AE	E1 ($\overline{\text{PE1}}$)	E0 ($\overline{\text{PE0}}$)	Error Type
0	1	1	No parity error
0	1	0	Parity error, byte 0
0	0	1	Parity error, byte 1
0	0	0	Parity error, bytes 0, 1

TABLE III. DP8400 MODES OF OPERATION

Mode	M2 (R/W)	M1	M0	$\overline{\text{OES}}$	Operation
0	0	0	0	X	Normal WRITE DIL → DOL, CG → COL → COB
1	0	0	1	X	Complement WRITE $\overline{\text{DIL}}$ → DOL, $\overline{\text{CIL}}$ → COL → COB
2	0	1	0	X	Diagnostic WRITE, DLE inhibited DQ8-DQ15 ⊕ CG → SOL → SOB DQ8-DQ15 → CIL → COL → COB
3	0	1	1	X	Complement data-only WRITE $\overline{\text{DIL}}$ → DOL, (CG0, 1, 4, 5, $\overline{\text{CG2}}$, $\overline{\text{CG3}}$) → COL → COB
4	1	0	0	X	Normal READ DIL ⊕ DE → DOL, CIL → COL
5	1	0	1	X	Complement READ $\overline{\text{DIL}}$ ⊕ DE → DOL, $\overline{\text{CIL}}$ → COL
6A	1	1	0	0	READ generated syndromes, check bit bus, error flags, SG0-SG6 → DQ0-DQ6, CIL0-CIL6 → DQ8-DQ14, E1 → DQ7, E0 → DQ15
6B	1	1	0	1	READ syndrome bus, check bit bus, error flags, SIL0-SIL6 → DQ0-DQ6, CIL0-CIL6 → DQ8-DQ14, E1 → DQ7, E0 → DQ15
7A	1	1	1	0	Generated syndromes replace with zero 0 → SIL → SG, CIL → COL, DIL ⊕ DE → DOL
7B	1	1	1	1	Generated syndromes replace SIL → SG, CIL → COL, DIL ⊕ DE → DOL

TABLE IV. DATA-IN TO CHECK BIT GENERATE, OR DATA BIT ERROR TO SYNDROME-GENERATE MATRIX (16-BIT CONFIGURATION)

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		DQ0–15
		GENERATE CHECK BITS →																	
GENERATED SYNDROMES	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	GENERATED CHECK BITS
	1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1	
	2	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	2*	
	3	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	3*	
	4	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	4	
	5	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	5	
		4	8	9	7	5	1	3	9	E	B	D	3	C	7	F	F	0	
		3	3	2	0	2	3	2	1	3	0	0	1	2	3	2	1	1	

HEXADECIMAL EQUIVALENT OF SYNDROME BITS

* C2, C3 generate odd parity

MODES OF OPERATION

There are three mode-control pins, M2, M1 and M0, offering 8 major modes of operation, according to Table III.

M2 is the READ/$\overline{\text{WRITE}}$ control. In normal operation, mode 0 is Normal WRITE and mode 4 is Normal READ. By clamping M0 and M1 low, and setting M2 low during WRITE and high during READ, the DP8400 is very easy to use for normal operation. The other modes will be covered in later sections.

16-Bit Configuration

The first two rows on top of the check bit generate matrix (Table IV) indicate the data position of DQ0 to DQ15. The left side of the matrix, listed 0 to 5, corresponds to syndromes S0 to S5. S0 is the least significant syndrome bit. There are two rows of hexadecimal numbers below the matrix. They are the hex equivalent of the syndrome patterns. For example, syndrome pattern in the first column of the matrix is 001011. Its least significant four bits (0010) equal hexadecimal 4, and the remaining two bits (11) equal hexadecimal 3.

Check bit generation is done by selecting different combinations of data bits and generating parities from them. Each row of the check bit generate matrix corresponds to the generation of a check bit numbered on the right hand side of the matrix, and the ones in that row indicate the selection of data bits.

The following are the check bit generate equations for 16-bit wide data words:

$$CG0 = DQ2 \oplus DQ3 \oplus DQ4 \oplus DQ5 \oplus DQ6 \oplus DQ7 \oplus DQ9 \oplus DQ10 \oplus DQ11 \oplus DQ13 \oplus DQ14 \oplus DQ15$$

$$CG1 = DQ3 \oplus DQ6 \oplus DQ8 \oplus DQ9 \oplus DQ11 \oplus DQ13 \oplus DQ14 \oplus DQ15$$

$${}^{*}CG2 = DQ0 \oplus DQ3 \oplus DQ4 \oplus DQ8 \oplus DQ10 \oplus DQ12 \oplus DQ13 \oplus DQ14 \oplus DQ15 \oplus 1$$

$${}^{*}CG3 = DQ1 \oplus DQ2 \oplus DQ7 \oplus DQ8 \oplus DQ9 \oplus DQ10 \oplus DQ12 \oplus DQ14 \oplus DQ15 \oplus 1$$

$$CG4 = DQ0 \oplus DQ1 \oplus DQ5 \oplus DQ7 \oplus DQ8 \oplus DQ11 \oplus DQ13 \oplus DQ15$$

$$CG5 = DQ0 \oplus DQ1 \oplus DQ2 \oplus DQ4 \oplus DQ5 \oplus DQ6 \oplus DQ8 \oplus DQ12 \oplus DQ13 \oplus DQ14$$

*CG2 and CG3 are odd parities.

The following error map (Table V) depicts the relationship between all possible error conditions and their associated syndrome patterns. For example, if a syndrome pattern is S0 – 5 = 111101, data bit 14 is in error.

Figure 4 shows how to connect one DP8400 in a 16-bit configuration, in order to detect and correct single or double-bit errors. For a Normal WRITE, processor data is presented to the DP8400, where it is fed through DIL0 and DIL1 to the check bit generator. This generates 6 parity bits from different combinations of data bits, according to Table IV. The numbers in the row below the table are the hexadecimal equivalent of the column bits (with bits 6, 7 low). A '1' in any row indicates that the data bit in that column is connected to the parity generator for that row. For example, check bit 1 generates parity from data bits 3, 6, 8, 9, 11, 13, 14, and 15.

Check bits 0, 1, 4, 5, and 6 generate even parity, and check bits 2 and 3 generate odd parity. This is done to insure that a total memory failure is detected. If all check bits were even parity, then all zeros in the data word would generate all check bits zero and a total memory failure would not be detected when a memory READ was performed. Now all-zero-data bits produce C2 and C3 high and a total memory failure will be detected. When reading back from the same location, the memory data bits (possibly in error) are fed to the same check bit generator, where they are compared to the memory check bits (also possibly in error) using 6 exclusive-OR gates. The outputs of the XORs are the syndrome bits, and these can be determined according to Table IV for one data bit error. For example, an error in bit 2 will produce the syndrome word 101001 (for S5 to S0 respectively). The syndrome word is decoded by the error encoder to the error flags, and the data-error decoder to correct a single data bit error. Assuming the memory data has been latched in the DIL, by making DLE go low, memory data can be disabled. Then by setting $\overline{\text{OB0}}$ and $\overline{\text{OB1}}$ low, corrected data will appear on the data bus. The syndromes are available as outputs on pins S0–5 when $\overline{\text{OES}}$ is low. It is also possible to feed in syndromes to SIL when $\overline{\text{OES}}$ is high and CSLE goes high. This can be useful when using the Error Management Unit shown in *Figure 4.* C6 and S6 are not used for 16 bits. It is safe therefore to make C6 appear low, through a 2.7 kΩ resistor to ground. The same applies for S6 if syndromes are input to the DP8400. If $\overline{\text{OES}}$ is permanently low, S6 may be left open.

Any 16-bit memory correct system using the DP8400 without syndrome inputs must keep the $\overline{\text{OES}}$ pin grounded, then all the syndrome I/O pins may be left open. The reason for this is that the DP8400 resets the syndrome input latch at power up. If the $\overline{\text{OES}}$ pin is grounded, the syndrome input latch will remain reset for normal operations.

The parameter t_{NMR} (see *Figure 10b*), new mode recognized time, is measured from M2 (changing from READ to WRITE) to the valid check bits appearing on the check bit bus, provided the $\overline{\text{OLE}}$ was held low.

TABLE V. SYNDROME DECODE TO BIT IN ERROR FOR 16-BIT DATA WORD

	S0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Syndrome	**S1**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Bits	**S2**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	S3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S5	**S4**																
0	0	NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D
0	1	C4	D	D	11	D	T	T	D	D	7	T	D	T	D	D	15
1	0	C5	D	D	6	D	4	T	D	D	2	T	D	12	D	D	14
1	1	D	5	T	D	0	D	D	13	1	D	D	T	D	T	8	D

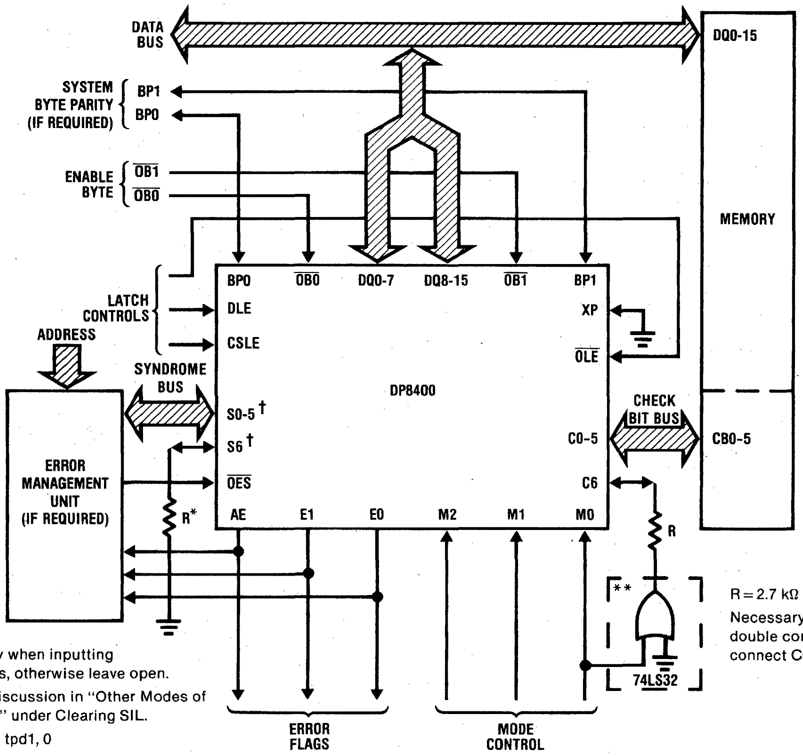
NE = no error

Number = single data bit in error

Cn = check bit n in error

D = two bits in error

T = three errors detected



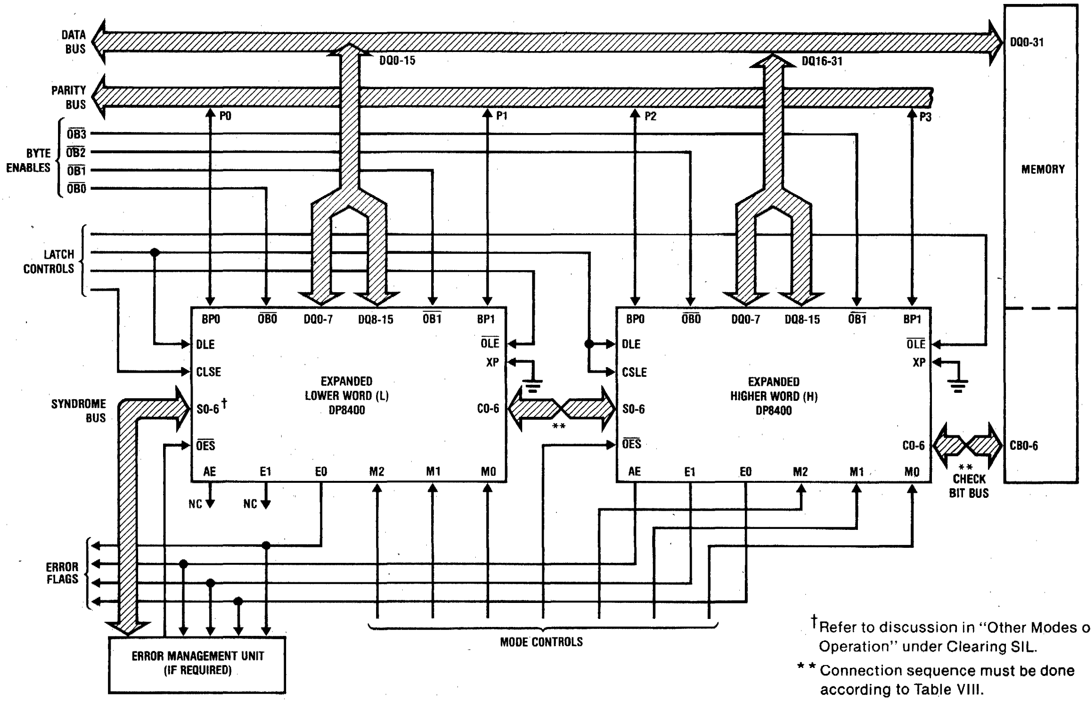
R = 2.7 kΩ
Necessary only when incorporating double complement correct. Otherwise connect C6 to GND through R.

* Necessary when inputting syndromes, otherwise leave open.

† Refer to discussion in "Other Modes of Operation" under Clearing SIL.

** 20 ns max tpd1, 0

FIGURE 4. 16-Bit Configuration Using One DP8400



† Refer to discussion in "Other Modes of Operation" under Clearing SIL.

** Connection sequence must be done according to Table VIII.

FIGURE 5. 32-Bit Error Detection and Correction

The parameter t_{MCR} (see *Figure 10b*), mode change recognized time, is measured from M2 (changing from WRITE to READ) when both E1 and E2 become invalid. This is required when a memory correcting system employs the DP8400 with byte parity checking. The E1 and E2 pins flag the byte parity error in a memory WRITE cycle. When the DP8400 switches to a subsequent memory READ cycle, it requires t_{MCR} for E1 and E2 to be switched to flag any READ error(s).

EXPANDED OPERATION

32-Bit Configuration

Figure 5 shows how to connect two DP8400s in cascade to detect single and double-bit errors, and to correct single-data errors. The same circuit will also correct double-bit errors once a double-error has been detected, provided at least one error is a hard error. The lower chip L is in effect a slave to the higher chip H, which controls the memory check bits and error reporting. The check bit bus of L is re-ordered and connected to the syndrome bus of H, as shown in *Figure 5.*

In a Normal WRITE mode, referring to *Figures 13a, 13b, and 13c*, the 6 check bits generated from the lower 16 bits (CGL) are transferred via the COL to the COB of L, provided $\overline{OLE}$ is high and M2 (R/$\overline{W}$) of L is low. These partial check bits from L then appear at SIL of H, so that with CSLE high, they combine with the 6 check bits generated in H with an overlap of one bit, to produce 7 check bits. With M2 (R/$\overline{W}$) of H low, these 7 check bits are output from COB to memory.

A READ cycle may consist of DETECT ONLY or DETECT THEN CORRECT, depending on the system approach. In both approaches, L writes its partial check bits, CGL, to H as in WRITE mode. H develops the syndrome bits from CGL, CGH and the 7 check bits read from memory in CIL. H then outputs from its error encoder (EE) if there is an error. If corrected data is required, H already knows if it has a single-data error from its syndrome bits, but if not, it must transfer partial syndromes back to L. These partial syndromes PSH, (CGH XOR-ed with CIL), are stored in SOL of H. L must therefore change modes from WRITE to READ, while H outputs the partial syndromes from its SOB by setting $\overline{OES}$ low. The partial syndromes are fed into CIL of L and XOR-ed with CGL to produce syndrome bits at SGL. The data error decoder, DED, then corrects the error in L. The DED of H will already have corrected an error in the higher 16 bits. Only one error in 32 bits can be corrected as a single-data error, the chip with no error does not change the contents of its DIL when it is enabled in DOL. Table VI shows the 3 error flags of H, which become valid during the DETECT cycle. E0 of L becomes valid during the CORRECT cycle, so that the 4 flags provide complete error reporting.

TABLE VI. ERROR FLAGS AFTER NORMAL READ (32-BIT CONFIGURATION)

AE (H)	E1 (H)	E0 (H)	E0 (L)*	Error Type
0	0	0	0	No error
1	1	0	0	Single-check bit error
1	1	1	0	Single-data bit error (H)
1	1	0	1	Single-data bit error (L)
1	0	0	0	Double-bit error
All Others				Invalid conditions

* E0 (L) is valid after transfer of partial syndromes from higher to lower

Equations for 32-bit expansion:

$$t_{DCB32} = t_{DCB16} + t_{SCB16}$$
$$t_{DEV32} = t_{DCB16} + t_{SEV16}$$
$$t_{DCD32}\text{ (High Chip)} = t_{DCB16} + t_{SCD16}$$
$$t_{DCD32}\text{ (Low Chip)} = t_{DCB16} + t_{BR}^{*} + t_{CCD16}$$

$^{*}t_{BR}$: Bus reversing time (25 ns)

32-Bit Matrix

Table VII shows a 32-bit matrix using two DP8400s in cascade as in *Figure 5*. This is one of 12 matrices that work for 32 bits. The matrix for bits 0 to 15 (lower chip) is the matrix of Table IV for 16-bit configuration, with row 6 always '0'. The matrix for bits 16 to 31 (higher chip) uses the same row combinations but interchanged, for example, the 3rd row (row 2) of L matrix is the same as the 6th row (row 5) of the H matrix. This means row 5 of H is in fact check bit 2 of H. Thus, the 6th row (row 5) combines generated check bit 5 (CG5) of L and generated check bit 2 of H. Check bit 5 of L therefore connects to the syndrome bit 2 (CG2) of H, and the composite generated check bit is written to check bit 2 of memory. Thus C2 performs a parity check on bits 0, 1, 2, 4, 5, 6, 8, 12, 13, 14, of L, and bits 16, 19, 20, 24, 26, 28, 29, 30, 31, of H. CG2 and CG3 generate odd parity, so that CG5 of L generates even parity which combines with CG2 of H generating odd parity. CG3 of L and CG3 of H both generate odd parity causing C3 to memory to represent even parity. Only 6 check bits are generated in each chip, the 7th (CG6) is always zero with XP grounded. Thus CG6 of L combines with CG0 of H so that C0 to memory is the parity of bits 18, 19, 20, 21, 22, 23, 25, 26, 27, 29, 30, 31. Similarly C6 to memory is only CG2 of L. The 7 composite generated check bits of H can now be written to memory.

When reading data and check bits from memory, CG6–CG0 of L are combined with CG6–CG0 of H in the same combination as WRITE. Memory check bits are fed into C6–C0 of H and compared with the 7 combined parity bits

TABLE VII. DATA BIT ERROR TO SYNDROME-GENERATE MATRIX (32-BIT CONFIGURATION)

		L																H																	
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31		DQ0-31
SYNDROMES	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1	GENERATED CHECK BITS
	1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	5	
	*2	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	
	*3	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	3	
	4	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	4	
	5	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	2	
	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	
		4	8	9	7	5	1	3	9	E	B	D	3	C	7	F	F	2	A	A	1	2	2	3	8	B	9	8	1	A	3	B	9	0	HEX
		3	3	2	0	2	3	2	1	3	0	0	1	2	3	2	1	3	1	4	6	6	5	4	5	3	4	6	5	2	7	6	7	1	

* CG2, CG3 generate odd parity

TABLE VIII. CHECK BIT PORT TO SYNDROME PORT INTERCONNECTIONS FOR EXPANSION TO 32 BITS

		L S		L C	H S		H C		
Syndrome I/O to Management	S0	0		0	1		1	C0	Check Bit I/O to Memory
	S1	1		1	5		5	C1	
	S2	2		2	6		6	C2	
	S3	3		3	3		3	C3	
	S4	4		4	4		4	C4	
	S5	5		5	2		2	C5	
	S6	6		6	0		0	C6	

TABLE IX. SYNDROME DECODE TO BIT IN ERROR FOR 32-BIT DATA WORD

		S0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Syndrome		S1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Bits		S2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
		S3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S6	S5	S4																
0	0	0	NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D
0	0	1	C4	D	D	11	D	T	T	D	D	7	17	D	T	D	D	15
0	1	0	C5	D	D	6	D	4	T	D	D	2	28	D	12	D	D	14
0	1	1	D	5	16	D	0	D	D	13	1	D	D	24	D	T	8	D
1	0	0	C6	D	D	22	D	T	T	D	D	25	18	D	T	D	D	T
1	0	1	D	27	21	D	T	D	D	T	23	D	D	T	D	T	T	D
1	1	0	D	19	20	D	T	D	D	T	26	D	D	30	D	T	T	D
1	1	1	T	D	D	29	D	T	T	D	D	31	T	D	T	D	D	T

NE = no error Cn = check bit n in error T = three errors detected

Number = single data bit in error D = two bits in error

in H, to produce 7 syndrome bits S6–S0. H can now determine if there is any error, and if it has a single-data error, it can locate it and correct it without transferring partial syndromes to L. As an example of a DETECT cycle, CG5 of L combines with CG2 of H and is compared in H with memory check bit 2.

If L is now set to mode 4, Normal READ, and $\overline{\text{OES}}$ of H is set low, the partial syndromes of H (CG6–CG0 of H XOR-ed with C6–C0 of H) are transferred and shifted to L. L receives these partial syndromes (S6–S0 of H) as check bit inputs C2, C1, C4, C3, C5, C0, C6 respectively, and compares them with CG6–CG0 respectively, to produce syndrome bits S6–S0. L now decodes these syndromes to correct any single-data error in data bits 0 to 15. For example, partial syndrome bit 2 of H combines with generated check bit 5 of L to produce syndrome bit 5 in L. An error in data bit 10 will create syndrome bits in L as 0001101 from S6–S0, and these will appear on S6–S0 of L with $\overline{\text{OES}}$ low. An error in H will appear as per the H matrix. For example, an error in bit 16 will cause S6–S0 of L to be 0110010.

If $\overline{\text{OES}}$ of L is set low, this syndrome combination appears on pins S6 to S0. For errors in bits 0 to 15, the syndrome outputs will be according to Table VII. For errors in bits 16 to 31, the syndrome outputs from L will still be according to Table VII due to the shifting of partial syndrome bits from H to L. The syndrome outputs from L are unique for each of the possible 32 bits in error.

If there is a check bit error, only one syndrome bit will be high. For example, if C5 is in error, then S1 of L will be high. For double-errors, an even number of syndrome bits will be high, derived from XOR-ing the two single-bit error syndromes. As mentioned previously, this is only one of the 12 approaches to connecting two chips for 32 bits, 6 of which are mirror images.

Table VIII depicts the exact connection for 32-bit expansion. LS equals syndrome bits of L. LC equals check bits of L. HS equals syndrome bits of H. HC equals check bits of H. Syndrome bits S0 to S6 of L are connected to system syndrome bits S0 to S6. LC and HS columns are lined together showing the check bit port of L connected to the syndrome port of H in the exact sequence as shown in Table VIII. For example, check bit C0 of L is connected to the syndrome bit S1 of H, and check bit C6 of L is connected to the syndrome bit S0 of H. Check bits of H are connected to the system check bits in the order shown. Check bit C1 of H is connected to the system check bit C0.

Expansion for Data Words Requiring 8 Check Bits

For 16-bit and 32-bit configurations, XP is set permanently low. In 48-bit or 64-bit configurations, XP is either set permanently to V_{CC} or left open, according to Table X, to provide 8 check bits and syndrome bits.

TABLE X. XP: EXPANSION STATUS

XP	Status	Data Bus
0V	BP0 and BP1 are byte parity I/O CG6 = 0	< 40 Bits
Open	No byte parity I/O, CG6 and CG7 = word parity	≥ 40 Bits
V_{CC}	No byte parity I/O, CG6 and CG7 = 0	≥ 40 Bits

48-Bit Expansion

Three DP8400s are required for 48 bits, with the higher chip using all 8 of its check bits to the memory. No byte parity is available for 48 or 64 bits. XP of all three chips must be at V_{CC}. The three chips are connected in cascade

TABLE XI. CHECK BIT PORT TO SYNDROME PORT INTERCONNECTIONS FOR EXPANSION TO 48 BITS

		LL S		LL C	LH S		LH C	HL S		HL C		
Syndrome I/O to Management	S0	0		0	1		1	6		6	C0	Check Bit I/O to Memory
	S1	1		1	5		5	1		1	C1	
	S2	2		2	6		6	4		4	C2	
	S3	3		3	3		3	7		7	C3	
	S4	4		4	4		4	2		2	C4	
	S5	5		5	2		2	3		3	C5	
	S6	6		6	0		0	5		5	C6	
	S7	7		7	7		7	0		0	C7	

For example: S0 of LL is connected to system syndrome S0. C0 of LL is connected to S1 of LH. C1 of LH is connected to S6 of HL. C6 of HL is connected to system check bit C0.

TABLE XII. SYNDROME DECODE TO BIT IN ERROR FOR 48-BIT DATA WORD

Syndrome Bits																
S0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
S1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
S2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
S3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S7 S6 S5 S4																
0 0 0 0	NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D
0 0 0 1	C4	D	D	11	D	T	T	D	D	7	17	D	T	D	D	15
0 0 1 0	C5	D	D	6	D	4	T	D	D	2	28	D	12	D	D	14
0 0 1 1	D	5	16	D	0	D	D	13	1	D	D	24	D	T	8	D
0 1 0 0	C6	D	D	22	D	T	T	D	D	25	18	D	T	D	D	T
0 1 0 1	D	27	21	D	32	D	D	T	23	D	D	T	D	T	T	D
0 1 1 0	D	19	20	D	33	D	D	T	26	D	D	30	D	T	T	D
0 1 1 1	44	D	D	29	D	T	40	D	D	31	T	D	T	D	D	T
1 0 0 0	C7	D	D	T	D	T	43	D	D	T	T	D	T	D	D	T
1 0 0 1	D	T	35	D	T	D	D	T	T	D	D	T	D	T	T	D
1 0 1 0	D	T	41	D	39	D	D	T	T	D	D	T	D	T	T	D
1 0 1 1	42	D	D	T	D	T	47	D	D	T	T	D	T	D	D	T
1 1 0 0	D	T	38	D	37	D	D	T	T	D	D	T	D	T	T	D
1 1 0 1	36	D	D	T	D	T	45	D	D	T	T	D	T	D	D	T
1 1 1 0	34	D	D	T	D	T	T	D	D	T	T	D	T	D	D	T
1 1 1 1	D	T	46	D	T	D	D	T	T	D	D	T	D	T	T	D

NE = no error	Cn = check bit n in error	T = three errors detected

Number = single data bit in error	D = two bits in error

as in *Figure 6,* but with the HH chip removed. The error flags are as Table XV, but with AE (HH) and E1 (HH) becoming AE (HL) and E1 (HL), and E0 (HH) removed.

48-Bit Matrix

The matrix for 48 bits is that for 64 bits shown (in Table XVI) but only using bits 0 to 47. This is one of many matrices for 48-bit expansion using the basic 16-bit matrix. The matrix shown uses 2 zeroes for CG6 and CG7, for all three chips, with XP set to V_{CC}. Other matrices may use CG6 and CG7 as word parity with XP open.

64-Bit Expansion

There are two basic methods of expansion to 64 bits, both requiring 8 check bits to memory, and four DP8400s. One is the cascade method of *Figure 6*, requiring no extra ICs. With this method partial check bits have to be transferred through three chips in the WRITE or DETECT mode, and partial syndrome bits transferred back through three chips in CORRECT mode. This method is similar to *Figure 5*, 32-bit approach. The connections between the check bit bus and syndrome bus for each of the chip pairs are shown in Table XIII.

The error flags of HH are valid during the DETECT cycle as in Table XV, and the other error flags are valid during the CORRECT cycle.

A faster method of 64-bit expansion shown in *Figure 7* requires a few extra ICs, but can WRITE in 57 ns, DETECT in 57 ns or DETECT THEN CORRECT in 116 ns. In the WRITE mode, all four sets of check bits are combined externally in the 8 74S280 parity generators. These generate 8 composite check bits from the system data, which are then enabled to memory. In the DETECT mode, again 8 composite check bits are generated, from the memory data this time, and compared with the memory check bits to produce 8 external syndrome bits. These syndrome bits may be OR-ed to determine if there is any error. By making the 74S280 outputs $\overline{\text{SYNDROMES}}$, then any bit low causes the 74S30 NAND gate to go high, giving any error indication. To correct the error, these syndrome bits are fed re-ordered into SIL of each DP8400 now set to mode 7B. This enables the syndromes directly to SG and then

DED of each chip. One chip will output corrected data, while the other three output non-modified data (but still correct).

Equations for fast 64-bit expansion:

$$t_{DCB64} = t_{DCB16} + t_{pd}\,(74S280) + t_{pd}\,(74S240)$$
$$t_{DEV64} = t_{DCB16} + t_{pd}\,(74S280) + t_{pd}\,(74S30)$$
$$t_{DCD64} = t_{DCB16} + t_{pd}\,(74S280) + t_{pd}\,(74ALS533) + t_{SCD16}$$

64-Bit Matrix

With the 64-bit matrix shown in Table XVI, it is necessary to set at least one chip with CG6, CG7 non-zero. The highest chip, connected to data bits 48 to 63, has XP set open, so that its CG6 and CG7 are word parity. The syndrome word of the highest chip will now have either 5 or 7 syndrome bits high, but inside the chip CG6 and CG7 remove two of these in a READ so that the chip sees the normal 3 or 5 syndrome bits.

TABLE XIII. CHECK BIT PORT TO SYNDROME PORT INTERCONNECTIONS FOR EXPANSION TO 64 BITS

		LL S		LL C	LH S		LH C	HL S		HL C	HH S		HH C		
Syndrome I/O to Management	S0	0		0	1		1	6		6	7		7	C0	Check Bit I/O to Memory
	S1	1		1	5		5	1		1	0		0	C1	
	S2	2		2	6		6	4		4	1		1	C2	
	S3	3		3	3		3	7		7	2		2	C3	
	S4	4		4	4		4	2		2	3		3	C4	
	S5	5		5	2		2	3		3	4		4	C5	
	S6	6		6	0		0	5		5	5		5	C6	
	S7	7		7	7		7	0		0	6		6	C7	

For example: S0 of LL is connected to system syndrome S0. C0 of LL is connected to S1 of LH. C1 of LH is connected to S6 of HL. C6 of HL is connected to S7 of HH. C7 of HH is connected to system check bit C0.

TABLE XIV. SYNDROME DECODE TO BIT IN ERROR FOR 64-BIT DATA WORD

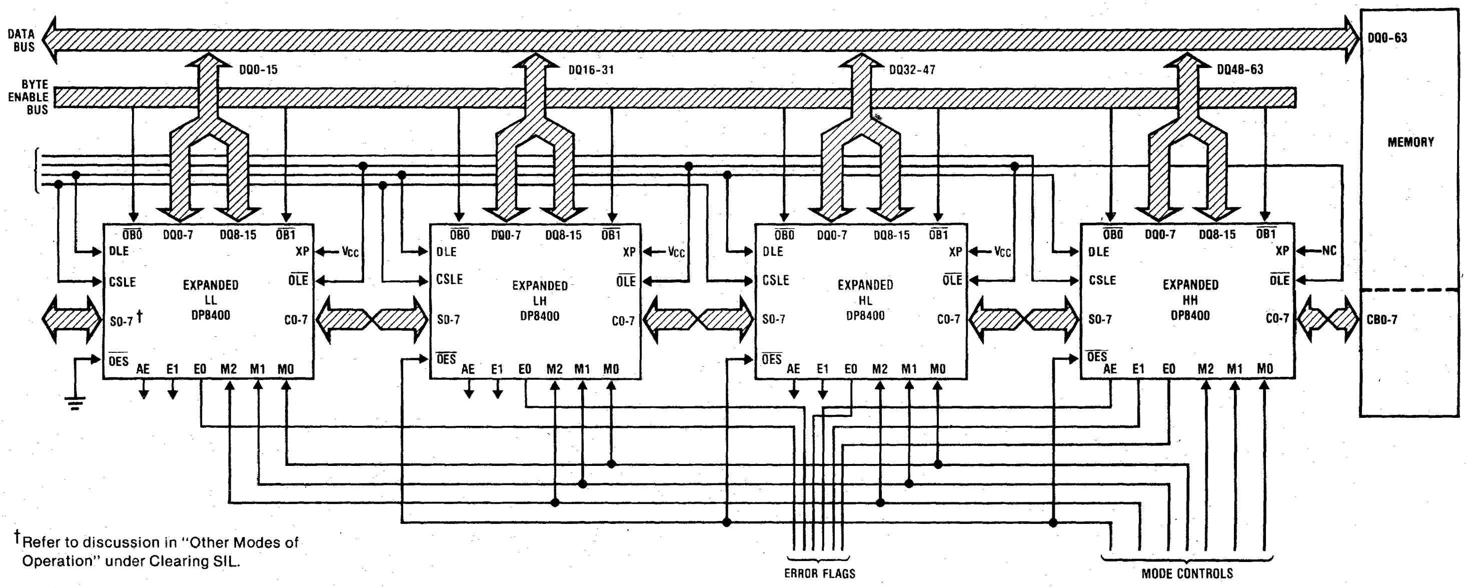
S0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Syndrome S1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
Bits S2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
S3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
S7 S6 S5 S4																
0 0 0 0	NE	C0	C1	D	C2	D	D	3	C3	D	D	9	D	10	T	D
0 0 0 1	C4	D	D	11	D	T	T	D	D	7	17	D	T	D	D	15
0 0 1 0	C5	D	D	6	D	4	T	D	D	2	28	D	12	D	D	14
0 0 1 1	D	5	16	D	0	D	D	13	1	D	D	24	D	T	8	D
0 1 0 0	C6	D	D	22	D	T	T	D	D	25	18	D	T	D	D	T
0 1 0 1	D	27	21	D	32	D	D	T	23	D	D	T	D	T	T	D
0 1 1 0	D	19	20	D	33	D	D	T	26	D	D	30	D	T	T	D
0 1 1 1	44	D	D	29	D	T	40	D	D	31	T	D	T	D	D	T
1 0 0 0	C7	D	D	T	D	T	43	D	D	T	T	D	T	D	D	51
1 0 0 1	D	T	35	D	T	D	D	57	T	D	D	58	D	T	T	D
1 0 1 0	D	T	41	D	39	D	D	59	T	D	D	T	D	T	T	D
1 0 1 1	42	D	D	55	D	T	47	D	D	T	T	D	T	D	D	63
1 1 0 0	D	T	38	D	37	D	D	54	T	D	D	52	D	T	T	D
1 1 0 1	36	D	D	50	D	T	45	D	D	60	T	D	T	D	D	62
1 1 1 0	34	D	D	53	D	T	T	D	D	48	T	D	T	D	D	61
1 1 1 1	D	49	46	D	T	D	D	T	T	D	D	T	D	56	T	D

NE = no error Cn = check bit n in error T = three errors detected

Number = single data bit in error D = two bits in error

TABLE XV. ERROR FLAGS AFTER NORMAL READ (ANY 64-BIT CONFIGURATION)

AE (HH)	E1 (HH)	E0 (HH)	E0 (HL)	E0 (LH)	E0 (LL)	Error Type
0	0	0	0	0	0	No error
1	1	0	0	0	0	Single-check bit error
1	1	1	0	0	0	Single-data bit error in HH
1	1	0	1	0	0	Single-data bit error in HL
1	1	0	0	1	0	Single-data bit error in LH
1	1	0	0	0	1	Single-data bit error in LL
1	0	0	0	0	0	Double-error



†Refer to discussion in "Other Modes of Operation" under Clearing SIL.

FIGURE 6. Cascade Expansion Using No Extra ICs (64-Bit Configuration)

TABLE XVI. DATA BIT ERROR TO SYNDROME-GENERATE MATRIX (64-BIT CONFIGURATION)

		LL																	
												1	1	1	1	1	1		DQ0-63
		0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5		
SYNDROMES	0	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	7	GENERATED CHECK BITS
	1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	0	
	2	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	1	
	3	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	2	
	4	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	3	
	5	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	4	
	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	
	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	
		4	8	9	7	5	1	3	9	E	B	D	3	C	7	F	F	0	HEX
		3	3	2	0	2	3	2	1	3	0	0	1	2	3	2	1	1	

		LH																	
		1	1	1	1	2	2	2	2	2	2	2	2	2	2	3	3		DQ0-63
		6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1		
SYNDROMES	0	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	7	GENERATED CHECK BITS
	1	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	0	
	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	
	3	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	2	
	4	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	3	
	5	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	4	
	6	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	5	
	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	
		2	A	A	1	2	2	3	8	B	9	8	1	A	3	B	9	0	HEX
		3	1	4	6	6	5	4	5	3	4	6	5	2	7	6	7	1	

		HL																	
		3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4		DQ0-63
		2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7		
SYNDROMES	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	GENERATED CHECK BITS
	1	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	0	
	2	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	1	
	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	
	4	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	3	
	5	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	4	
	6	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	5	
	7	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	6	
		4	4	0	2	0	4	2	4	6	2	0	6	0	6	2	6	0	HEX
		5	6	E	9	D	C	C	A	7	A	B	8	7	D	F	B	1	

		HH																	
		4	4	5	5	5	5	5	5	5	5	5	5	6	6	6	6		DQ0-63
		8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3		
SYNDROMES	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7	GENERATED CHECK BITS
	1	0	0	1	1	1	1	1	1	0	1	1	1	0	1	1	1	0	
	2	0	0	0	1	0	0	1	0	1	1	0	1	0	1	1	1	1	
	3	1	0	0	1	1	0	0	0	1	0	1	0	1	1	1	1	2	
	4	0	1	1	0	0	0	0	1	1	1	1	0	1	0	1	1	3	
	5	1	1	0	0	0	1	0	1	1	0	0	1	0	1	0	1	4	
	6	1	1	1	0	1	1	1	0	1	0	0	0	1	1	1	0	5	
	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	6	
		9	1	3	F	B	3	7	3	D	7	B	7	9	F	F	F	0	HEX
		E	F	D	8	C	E	C	B	F	9	9	A	D	E	D	B	1	

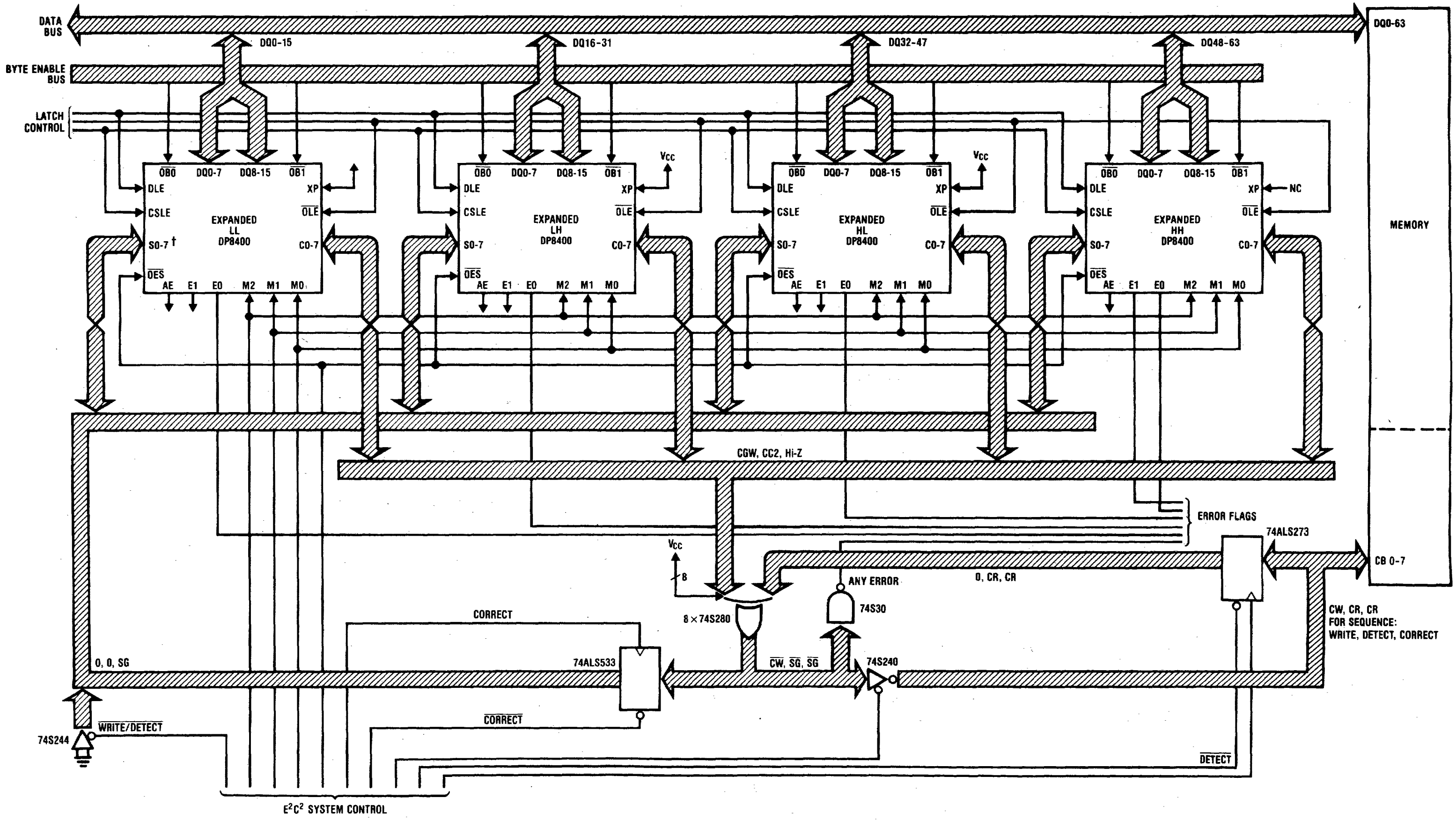


FIGURE 7. Parallel Expansion (Fast 64-Bit Configuration)

OTHER MODES OF OPERATION

Double Error Correction, using the Double-Complement Approach

The DP8400 can be made to correct two errors, using no extra ICs or check bits, if at least one of the two errors detected is a hard error. This does require an extra memory WRITE and READ. Nevertheless, if a permanent failure exists, and an additional error occurs (creating two errors), both errors can be corrected, thereby saving a system crash.

Once a double error has been detected, the system puts the DP8400 in COMPLEMENT mode by setting M0 high. First a WRITE cycle is required and M2 is set low, putting the chip in mode 1, Table III, (COMPLEMENT WRITE), so that the contents of DIL are complemented into DOL, and the contents of CIL complemented into COL. OB0 and OB1 are set low so that complemented data and check bits can be written back to the same location of memory. Writing back complemented data to a location with a hard error forces the error to repeat itself. For example, if cell N of a particular location is jammed permanently high, and a low is written to it, a high will be read. However, when the data is complemented a low is again written, so that a high is read back for the second time. After a second READ (this second READ is a COMPLEMENT READ) of the location, data and check bits from the memory are re-complemented, so that bit N now contains a low. In other words, the error in bit N has corrected itself, while the other bits are true again. If there are two hard errors in a location, both are automatically corrected and the DP8400 detects no error on COMPLEMENT READ, as in *Figure 8a. Figure 8b* also shows that if one error is soft, the hard error will disappear on the second READ and the DP8400 corrects the soft error as a single-error. Therefore, in both cases, the DOL contains corrected data, ready to be enabled by OB0 and OB1. A WRITE to memory at this stage removes the complemented data written at the start of the sequence.

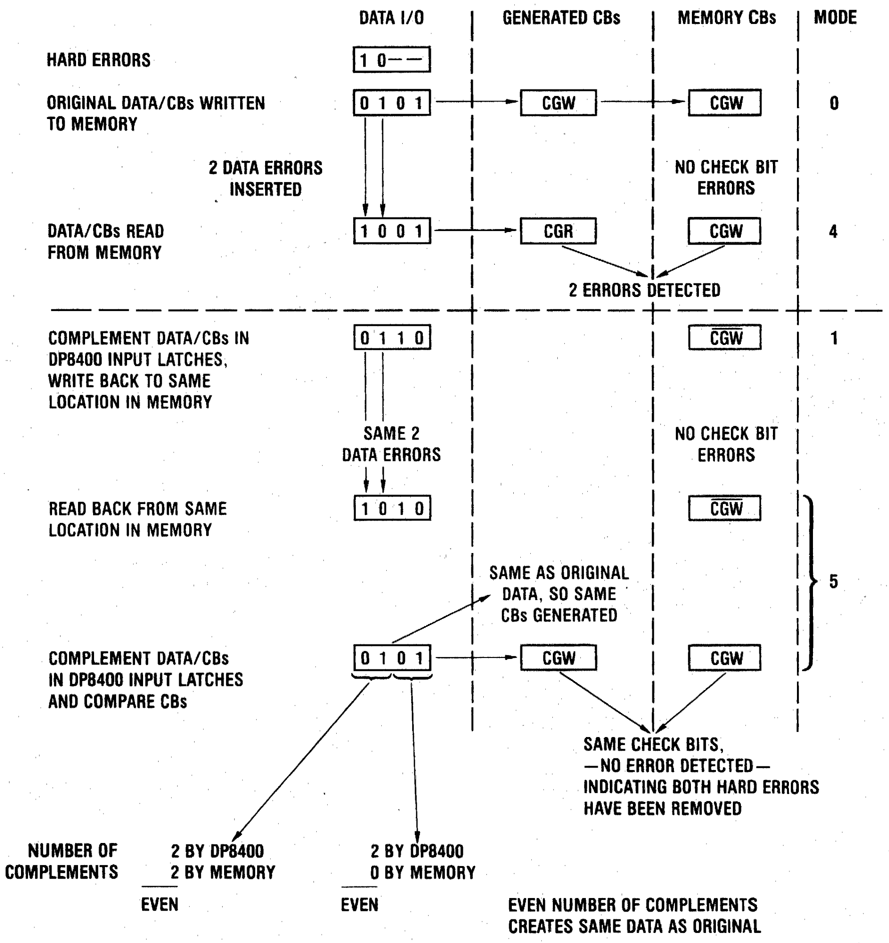

DATA I/O
GENERATED CBs
MEMORY CBs
MODE
HARD ERRORS
1 0 – –
ORIGINAL DATA/CBs WRITTEN TO MEMORY
0 1 0 1
CGW
CGW
0
2 DATA ERRORS INSERTED
NO CHECK BIT ERRORS
DATA/CBs READ FROM MEMORY
1 0 0 1
CGR
CGW
4
2 ERRORS DETECTED
COMPLEMENT DATA/CBs IN DP8400 INPUT LATCHES, WRITE BACK TO SAME LOCATION IN MEMORY
0 1 1 0
$\overline{CGW}$
1
SAME 2 DATA ERRORS
NO CHECK BIT ERRORS
READ BACK FROM SAME LOCATION IN MEMORY
1 0 1 0
$\overline{CGW}$
5
SAME AS ORIGINAL DATA, SO SAME CBs GENERATED
COMPLEMENT DATA/CBs IN DP8400 INPUT LATCHES AND COMPARE CBs
0 1 0 1
CGW
CGW
SAME CHECK BITS, —NO ERROR DETECTED— INDICATING BOTH HARD ERRORS HAVE BEEN REMOVED
NUMBER OF COMPLEMENTS
2 BY DP8400
2 BY MEMORY
EVEN
2 BY DP8400
0 BY MEMORY
EVEN
EVEN NUMBER OF COMPLEMENTS CREATES SAME DATA AS ORIGINAL

FIGURE 8a. Double Error Correct Complement Hard Error Method—2 Hard Errors in Data Bits

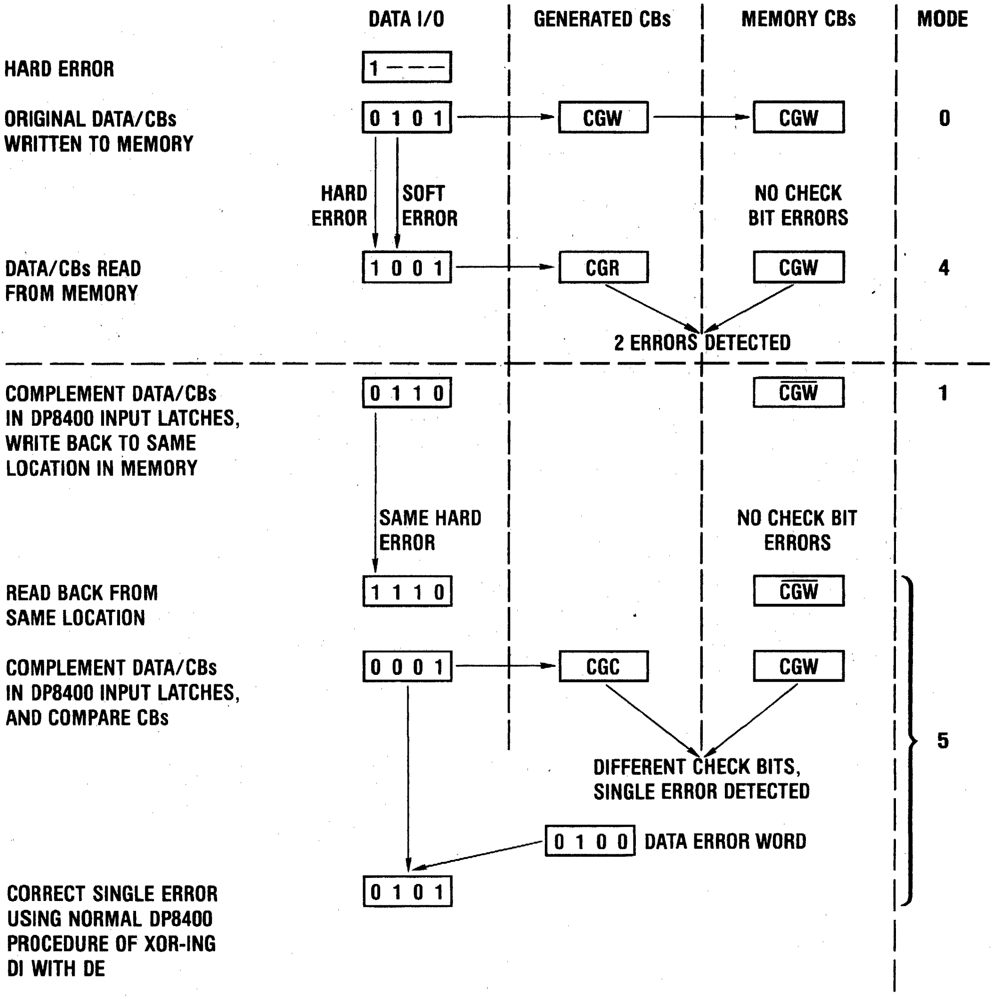

DATA I/O
GENERATED CBs
MEMORY CBs
MODE
HARD ERROR
1 – – –
ORIGINAL DATA/CBs WRITTEN TO MEMORY
0 1 0 1
CGW
CGW
0
HARD ERROR
SOFT ERROR
NO CHECK BIT ERRORS
DATA/CBs READ FROM MEMORY
1 0 0 1
CGR
CGW
4
2 ERRORS DETECTED
COMPLEMENT DATA/CBs IN DP8400 INPUT LATCHES, WRITE BACK TO SAME LOCATION IN MEMORY
0 1 1 0
$\overline{\text{CGW}}$
1
SAME HARD ERROR
NO CHECK BIT ERRORS
READ BACK FROM SAME LOCATION
1 1 1 0
$\overline{\text{CGW}}$
COMPLEMENT DATA/CBs IN DP8400 INPUT LATCHES, AND COMPARE CBs
0 0 0 1
CGC
CGW
5
DIFFERENT CHECK BITS, SINGLE ERROR DETECTED
0 1 0 0
DATA ERROR WORD
CORRECT SINGLE ERROR USING NORMAL DP8400 PROCEDURE OF XOR-ING DI WITH DE
0 1 0 1

FIGURE 8b. Double Error Correct Complement Hard Error Method—1 Hard Error, 1 Soft Error In Data Bits

The examples shown in *Figures 8a and 8b* are for 4 data bits. This approach will work for any number of data bits, but for simplicity these examples show how complementing twice corrects two errors in the data bits. The double COMPLEMENT approach also works for any two errors providing at least one is hard. In other words, one data-bit error and one check bit error, or two check bit errors are also corrected if one or both are hard. At the end of the COMPLEMENT READ cycle, the error flags indicate whether the data was correctable or not, as shown in Table XVII. If both the errors were soft, then the data was not correctable and the error flags indicate this.

This approach is ideal where double errors are rare but may occur. To avoid a system crash, a double-error detect now causes the system to enter a subroutine to set the DP8400 in COMPLEMENT mode. This method is also useful in bulk-memory applications, where RAMs are used with known cell failures, and is applicable in 16, 32, 48 or 64-bit configurations. In the 16-bit configuration, modes 1 and 5 of Table III are used. In the 32-bit expanded configuration, modes 1, 5 and 5 are used for the highest chip, and modes 3, 3 and 4 for the lower chip for WRITE, DETECT, and CORRECT. With the lower chip it is necessary to wrap around DOL (after latching its contents in mode 3), back to DIL and perform a Normal READ in mode 4 in the lower chip.

TABLE XVII. ERROR FLAGS AFTER COMPLEMENT READ (MODE 5)

AE	E1	E0	Error Type
0	0	0	Two hard errors
1	1	0	One hard error, one soft check bit error
1	1	1	One hard error, one soft data bit error
1	0	0	Two soft errors, not corrected

Double-Error Correct with Error Logging

Figures 4 and 5 show the E^2C^2 syndrome port connected to an error management unit (EMU). This scheme stores syndromes and the address of locations that fail, thereby logging the errors. Subsequent errors in a memory location that has already stored syndromes in the EMU, can then be removed by injecting the stored syndromes of the first error. To save the addresses and syndromes when power to the EMU is removed, it is necessary to be able to transfer information via the E^2C^2 syndrome port to the processor data bus. This is also useful for logging the errors in the processor. Transfer in the opposite direction is also necessary.

Data Bus to Syndrome Bus Transfer

This is necessary when transferring syndrome information to the error management unit, which is connected to the external syndrome bus. First, data to make CG = 0 (all data bits high) must be latched in DIL. Then in mode 2, data is fed to CIL, XOR-ed with 0, and output via SOL with $\overline{\text{OES}}$ low to the syndrome bus. Data is therefore fed directly from the system to the syndrome bus, and this cycle may be repeated as long as DLE is kept low, forcing CG to remain zero.

Syndrome Bus to Data Bus Transfer

This is important when information in the error logger or error management unit has to be read. The DP8400 is set to mode 6B with $\overline{\text{OES}}$ high, and with $\overline{\text{OB0}}$, $\overline{\text{OB1}}$ and $\overline{\text{OLE}}$ low. If CSLE is high, the syndrome bus and check bit bus data appear on the lower and upper bytes of the data bus to be read by the system. Also E1 and E0 values that were valid when mode 6 was entered, appear on DQ7 and DQ15.

Full Diagnostic Check of Memory

Using mode 2, it is possible to transfer the upper byte of the data bus directly to the CIL, with CSLE high, without affecting DIL. These simulated check bits then appear on the check bit bus with $\overline{\text{OLE}}$ low, which also causes the previously latched contents of DIL to transfer to DOL. By enabling $\overline{\text{OB0}}$ and $\overline{\text{OB1}}$ data can be written to memory with the simulated check bits. A Normal READ cycle can then aid the system in determining that the memory bits are functioning correctly, since the processor knows the check bits and data it sent to the E^2C^2. Another solution is to put the E^2C^2 in mode 6 and read the memory check bits directly back to the processor.

Self-Test of the E^2C^2 On-Card

Again using mode 2, data written from the processor data bus upper byte to CIL may be stored in CIL, by taking CSLE low. Data can now be fed into DIL from the processor, with DLE set high, as in a Normal READ mode (mode 4). Providing CSLE is kept low, the DP8400 will use the simulated check bits in CIL to perform a diagnostic READ, with valid error reporting and correcting. This may be repeated with new data provided CSLE is kept low. In this way memory is not used, thus by reading corrected data in mode 4, and by reading the generated syndromes, and error flags E0 and E1, the DP8400 can be tested completely on-card without involving memory.

Monitoring Generated Syndromes and Memory Check Bits

Mode 6A enables SG0-SG6 onto DQ0-DQ6, and CIL0-CIL6 onto DQ8-DQ14, provided $\overline{\text{OLE}}$, $\overline{\text{OB0}}$ and $\overline{\text{OB1}}$ are low. Also the two error flags, E1 and E0 (latched from the previous READ mode), appear on DQ7 and DQ15. This may be used for checking the internal syndromes, for reading the memory check bits, or for diagnostics by checking the latched error flags.

Clearing SIL

In the 16-bit only configuration, or the lower chip of expanded configurations, and in various modes of operation in the higher expanded chips, it is required that SIL be maintained at zero. At power-up initialization, both SIL and DIL are reset to all low. If $\overline{\text{OES}}$ is kept low, SIL will remain reset because CSLE is inhibited to SIL. Another method is to keep $\overline{\text{OLE}}$ always high and the syndrome bus externally set low, or set low whenever CSLE can be used to clear SIL.

Mode 7A also forces the SIL to be cleared whenever CSLE occurs, and also these zero syndromes go to the internal syndrome bus SG. This puts the DP8400 in a PASS-THROUGH mode where the DIL contents pass to DOL and CIL contents to COL, if $\overline{\text{OLE}}$ is low.

Power-Up Initialization of Memory

Both SIL and DIL are reset low at power-up initialization. This facilitates writing all zeroes to the memory data bits to set up the memory. The check bits corresponding to all-zero data will appear on the check bit bus if the DP8400 is set to mode 0 and $\overline{\text{OLE}}$ is set low. All-zero data appears on the data bus when $\overline{\text{OB0}}$ and $\overline{\text{OB1}}$ are also set low. The system can now write zero-data and corresponding check bits to every memory location.

Byte Writing

Figure 14a shows the block diagram of a 16-bit memory correction system consisting of a DP8400 error correction chip and a DP8409 DRAM controller chip. There are 12 control signals associated with the interface. Six of the signals are standard DP8400 input signals, three are standard DP8409 input signals, and three are buffer control signals. The buffer control signals, $\overline{\text{PBUF0}}$ and $\overline{\text{PBUF1}}$, control when data words or bytes from the DP8400/memory data bus are gated to the processor bus and when data words or bytes from the processor are gated to the DP8400/memory data bus.

When the processor is reading or writing bytes to memory, words will always be read or written by the DP8400 and DP8409 error correction and DRAM controller section. The High Byte Enable and Address Data Bit Zero signals from the processor should control the byte transfers via the ocal bus transceiver signals $\overline{\text{PBUF0}}$ and $\overline{\text{PBUF1}}$. The buffer control signal, $\overline{\text{DOUTB}}$, controls when data from memory is gated onto the DP8400/memory data bus.

Figure 14b shows the timing relationships of the 12 control signals, along with the DP8400/memory data bus and some of the DRAM control signals ($\overline{\text{RAS}}$ and $\overline{\text{CAS}}$). RGCK is the $\overline{\text{RAS}}$ generator clock of the DP8409 which is used in Mode 1 (Auto Refresh mode), along with being the system clock.

Having two separate byte enable pins, $\overline{\text{OB0}}$ and $\overline{\text{OB1}}$, it is fairly easy to implement byte writing using the DP8400. First it is necessary to read from the location to which the byte is to be written. To do this the DP8400 is put in normal Read mode (Mode 4), which will detect and correct a single bit error. $\overline{\text{WIN}}$ is kept high and $\overline{\text{RASIN}}$ is pulled low, causing the DP8409, now in Mode 5 (Auto Access mode), to start a read memory cycle. The data word and check bits from memory are then enabled onto the DP8400/memory data bus by pulling $\overline{\text{DOUTB}}$ low. The data and check bits are valid on the bus after the $\overline{\text{RASIN}}$ to $\overline{\text{CAS}}$ time (t_{RAC}) plus the column access time (t_{CAC}) of the particular memories used. DLE,CSLE can then be pulled low in order to latch the memory data into the input latches of the DP8400. Next $\overline{\text{OLE}}$ can be pulled low to enable the corrected memory word, or the original memory word if no error was present, into the data output latches. The corrected memory word will be available at the data output latches "t_{DCD16}" after the memory word was available at the data input latches. Once the corrected data is available at the output latches $\overline{\text{OLE}}$ can be pulled high to latch the corrected data. After this DLE,CSLE can be pulled high in order to enable the input data latches again and $\overline{\text{DOUTB}}$ can be pulled high to disable the memory data from the DP8400/memory data bus.

There is no reason to use the data or check bit input latches (DLE,CSLE) of the DP8400 during the read cycle time period if the memory data and checkbits are valid throughout the cycle.

Now the DP8400 can be put into a write cycle (Mode 0 = M2 = Low). At this time the byte to be written to memory and the other byte from memory can be enabled onto the DP8400/memory data bus ($\overline{\text{OB0}}$, $\overline{\text{PBUF1}}$ or $\overline{\text{OB1}}$, $\overline{\text{PBUF0}}$ go low). DLE,CSLE can now transition low to latch the new memory word into the data input latch. Next $\overline{\text{OLE}}$ is pulled low to enable the output latches. When the new checkbits are valid, t_{DCB16} after the data word is valid on the DP8400/memory data bus, $\overline{\text{OLE}}$ and $\overline{\text{DLE}}$ can be pulled high to latch the new memory word into the output latches, and then $\overline{\text{WIN}}$ can be pulled low to write the data into memory. $\overline{\text{RASIN}}$ should be held low long enough to cause the new data and check bits to be stored into memory ($\overline{\text{WIN}}$ data hold time).

DLE,CSLE and $\overline{\text{OLE}}$ could transition high and low simultaneously instead of being sequenced as was done in this example.

Also a READ-MODIFY-WRITE cycle was performed, taking approximately 30% longer than a normal memory WRITE cycle. A READ and then a WRITE memory cycle could have been used in the above example but it would have taken longer.

Because data from the processor was valid at the same time as data from memory, memory buffers were used ($\overline{\text{PBUF0}}$,$\overline{\text{PBUF1}}$,$\overline{\text{DOUTB}}$).

A byte READ from memory is no different from a normal READ. This approach may be used for a 16-bit processor using byte writing, or an 8-bit processor using a 16-bit memory to reduce the memory percentage of check bits, or with memory word sizes greater than two bytes.

Beyond Single-Error Correct

With the advent of larger semiconductor memories, the frequency of the soft errors will increase. Also some memory system designers may prefer to buy less expensive memories with known cell, row or column failures, thus, more hard errors. All this means that double-error correct, triple-error detect capability, and beyond will become increasingly important. The DP8400 can correct two errors, provided one or both are hard errors, with no extra components, using the double complement approach. There are two other approaches to enhance reliability and integrity. One is to use the error management unit to log errors using the syndrome bus, and then to output these syndromes, when required, back to the DP8400.

Double Syndrome Decoding

The other approach takes advantage of the Rotational Syndrome Word Generator matrix. This matrix is an improvement of the Modified Hamming-code, so that if, on a second DP8400, the data bus is shifted or rotated by one bit, and 2 errors occur, the syndromes for this second chip will be different from the first for any 2 bits in error. Both chips together output a unique set of syndromes for any 2 bits in error. This can be decoded to correct any 2-bit error. This is not possible with other Modified Hamming-code matrices.

Absolute Maximum Ratings (Note 1)

Storage Temperature Range	−65°C to +150°C
Supply Voltage, V_{CC}	7V
Input Voltage	5.5V
Output Sink Current	50 mA
Maximum Power Dissipation* at 25°C	
Molded Package	3269mW
Lead Temperature (Soldering, 10 seconds)	300°C

*Derate molded package 26.2mW/°C above 25°C.

Operating Conditions

	Min	Max	Unit
V_{CC}, Supply Voltage	4.75	5.25	V
T_A, Ambient Temperature	0	70	°C

Electrical Characteristics (Note 2) $V_{CC} = 5V \pm 5\%$, $T_A = 0°C$ to 70°C unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IL}	Input Low Threshold				0.8	V
V_{IH}	Input High Threshold		2.0			V
V_C	Input Clamp Voltage	V_{CC} = Min, $I_C = -18$ mA		−0.8	−1.5	V
I_{IH}	Input High Current	V_{IN} = 2.7V		1	160	μA
I_{IH} (XP)	Input High Current	V_{CC} = Max, XP = 5.25V		2.5	3.6	mA
I_{IL} (XP)	Input Low Current	V_{CC} = Max, XP = 0V		−2.5	−3.6	mA
I_{IL} (BP0/C7)	Input Low Current	V_{CC} = Max, V_{IN} = 0.5V		−100.0	−500	μA
I_{IL} (BP1/S7)	Input Low Current	V_{CC} = Max, V_{IN} = 0.5V		−100.0	−500	μA
I_{IL} (CSLE)	Input Low Current	V_{CC} = Max, V_{IN} = 0.5V		−150.0	−750	μA
I_{IL} (DLE)	Input Low Current	V_{CC} = Max, V_{IN} = 0.5V		−200.0	−1000	μA
I_{IL}	Input Low Current	V_{CC} = Max, V_{IN} = 0.5V		−50.0	−250	μA
I_I	Input High Current (Max)	V_{IN} = 5.5V (Except XP Pin)			1.0	mA
V_{OL}	Output Low Voltage	I_{OL} = 8 mA (Except BP0, BP1) I_{OL} = 4 mA (BP0, BP1 Only)		0.3 0.3	0.5 0.5	V V
V_{OH}	Output High Voltage	$I_{OH} = -100$ μA $I_{OH} = -1$ mA	2.7 2.4	3.2 3.0		V V
I_{OS}	Output Short Current (Note 3)	V_{CC} = Max		−55	−100	mA
I_{CC}	Supply Current	V_{CC} = Max		340	410	mA
C_{IN} (I/O)	Input Capacitance All Bidirectional Pins	Note 4		8.0		pF
C_{IN}	Input Capacitance All Unidirectional Input Pins	Note 4		5.0		pF

Note 1: "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All typical values are for $T_A = 25°C$ and $V_{CC} = 5.0V$.

Note 3: Only one output at a time should be shorted.

Note 4: Input capacitance is guaranteed by periodic testing. F test = 10 kHz at 300 mV, $T_A = 25°C$.

Note 5: All switching parameters measured from 1.5V of input to 1.5V of output. Input pulse amplitude 0V to 3V, $t_r = t_f = 2.5$ ns.

DP8400-4 Switching Characteristics (Note 5)

$V_{CC} = 5.0V \pm 5\%$, $T_A = 0°C$ to $70°C$, $C_L = 50$ pF, unless otherwise noted.

Symbol	Parameter	Conditions		Min	Typ	Max	Units
t_{DCB16}	Data Input Valid to Check Bit Valid	*Figure 9b*			35	55	ns
t_{DEV16}	Data Input to Any Error Valid	*Figures 10b, 11b*			35	45	ns
t_{DCD16}	Data Input Valid to Corrected Data Valid	*Figure 10b*, $\overline{OB0}$, $\overline{OB1}$ Low			70	85	ns
t_{DSI}	Data Input Set-Up Time Before DLE, CSLE H to L	*Figures 10b, 13d*			−40	−10	ns
t_{DHI}	Data Input Hold Time After DLE, CSLE H to L	*Figures 10b, 13d*		16	10		ns
t_{DSO}	Data Input Set-Up Time Before $\overline{OLE}$ L to H	*Figure 10b*		20	12		ns
t_{DHO}	Data Input Hold Time After $\overline{OLE}$ L to H	*Figure 10b*		20	12		ns
t_{DE0}	E0 Valid After AE Valid	*Figures 9b, 10b, 13d*			20	30	ns
t_{DE1}	E1 Valid After AE Valid	*Figures 9b, 10b, 13d*			12	30	ns
t_{IEV}	DLE, CSLE High to Any Error Flag Valid (Input Data Previously Valid)	*Figure 10b*			60	80	ns
t_{IEX}	DLE, CSLE High to Any Error Flag Invalid	*Figures 9b, 10b*			60	77	ns
t_{ILE}	DLE, CSLE High Width to Guarantee Valid Data Latched	*Figures 10b, 13d*	DLE	25			ns
			CSLE	50			ns
t_{OLE}	$\overline{OLE}$ Low Width to Guarantee Valid Data Latched	*Figure 13d*		25			ns
t_{ZH}	High Impedance to Logic 1 from $\overline{OB0}$, $\overline{OB1}$, $\overline{OES}$	*Figures 9b, 10b*			32	50	ns
	M2 H to L	*Figure 13d*			70	85	ns
t_{HZ}	Logic 1 to High Impedance from $\overline{OB0}$, $\overline{OB1}$, $\overline{OES}$, M2 L to H	*Figures 9b, 10b, 13d*, $C_L = 15$ pF			25	40	ns
t_{ZL}	High Impedance to Logic 0 from $\overline{OB0}$, $\overline{OB1}$, $\overline{OES}$	*Figures 9b, 10b*			30	45	ns
	M2 H to L	*Figure 13d*			70	85	ns
t_{LZ}	Logic 0 to High Impedance from $\overline{OB0}$, $\overline{OB1}$, $\overline{OES}$, M2 H to L	*Figures 9b, 10b, 13d* $C_L = 15$ pF			25	40	ns
t_{PPE}	Byte Parity Input Valid to Parity Error Flags Valid	*Figure 9b*			40	55	ns
t_{DPE}	Data In Valid to Parity Error Flags Valid	*Figures 9b, 13d*			60	75	ns
t_{DBP}	Data in Valid to Byte Parity Output Valid	*Figure 9b*			36	50	ns
t_{MCR}	Mode Change Recognize Time	*Figures 9b, 10b*			60	100	ns

DP8400-4 Switching Characteristics (Continued) (Note 5)

$V_{CC} = 5.0V \pm 5\%$, $T_A = 0°C$ to $70°C$, $C_L = 50$ pF, unless otherwise noted.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{NMR}	New Mode Recognize Time	*Figure 10b*		60	100	ns
t_{CDV}	Mode Valid to Complement Data Valid	*Figure 11b*		55	72	ns
t_{CCV}	Mode Valid to Complement Check Bit Valid	*Figure 11b*		55	72	ns
t_{SCB}	Syndrome Input Valid to Check Bit Valid	*Figure 13d*		28	41	ns
t_{SEV}	Syndrome Input Valid (CGL) to Any Error Valid	*Figure 13d*		25	39	ns
t_{SCD}	Syndrome Inputs Valid to Corrected Data Valid	*Figure 13d*		55	75	ns
t_{DSB}	Data Input Valid to Syndrome Bus Valid	*Figure 13d*, $\overline{OES}$ Low		45	58	ns
t_{CSB}	Check Bit Inputs Valid to Syndrome Bus Valid	*Figure 13d*, $\overline{OES}$ Low		40	51	ns
t_{CEV}	Check Bit Inputs Valid (PSH) to Any Error Valid	*Figure 13d*		35	45	ns
t_{CCD}	Check Bit Input Valid (PSH) to Corrected Data Valid	*Figure 13d*		70	82	ns
t_{DCB32}	Data Input Valid to Check Bit Valid	*Figure 13d*		63	96	ns
t_{DEV32}	Data Input Valid to Any Error Valid	*Figure 13d*		60	94	ns
t_{DCD32}	Data Input Valid to Corrected Data Out	*Figure 13d*, $\overline{OB0}$, $\overline{OB1}$ Low		125	157	ns

Note 1: "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All typical values are for $T_A = 25°C$ and $V_{CC} = 5.0V$.

Note 3: Only one output at a time should be shorted.

Note 4: Input capacitance is guaranteed by periodic testing. F test = 10 kHz at 300 mV, $T_A = 25°C$.

Note 5: All switching parameters measured from 1.5V of input to 1.5V of output. Input pulse amplitude 0V to 3V, $t_r = t_f = 2.5$ ns.

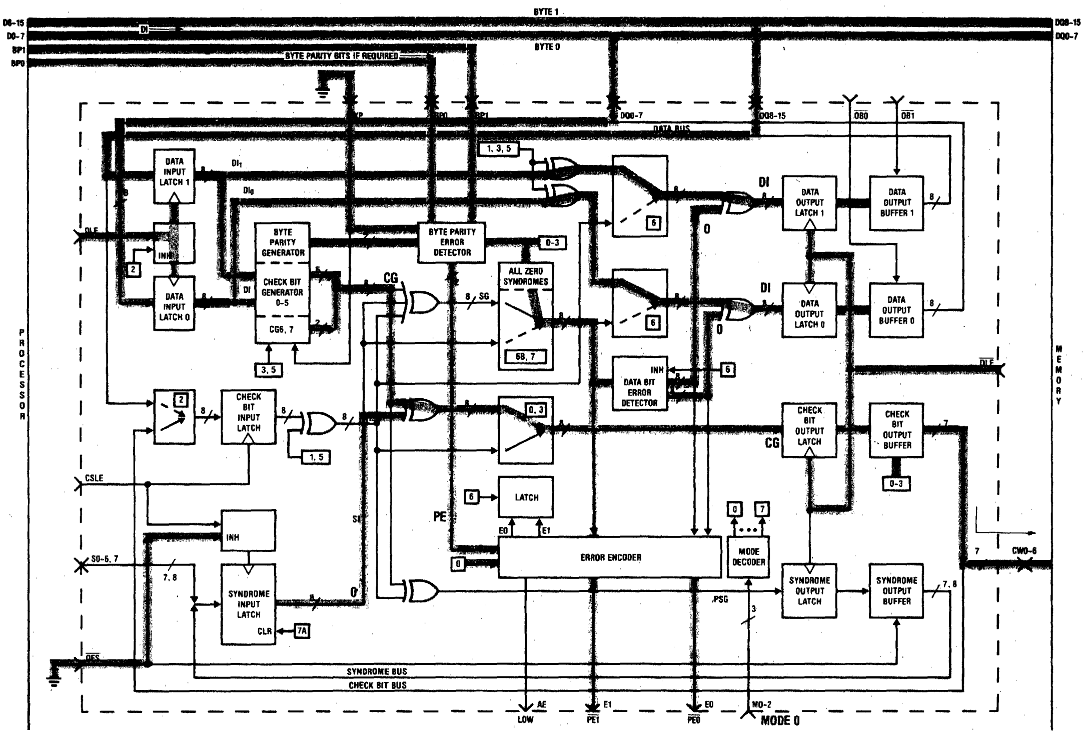


FIGURE 9a. DP8400 16-Bit Configuration, Normal WRITE with Byte Parity Error Detect If Required

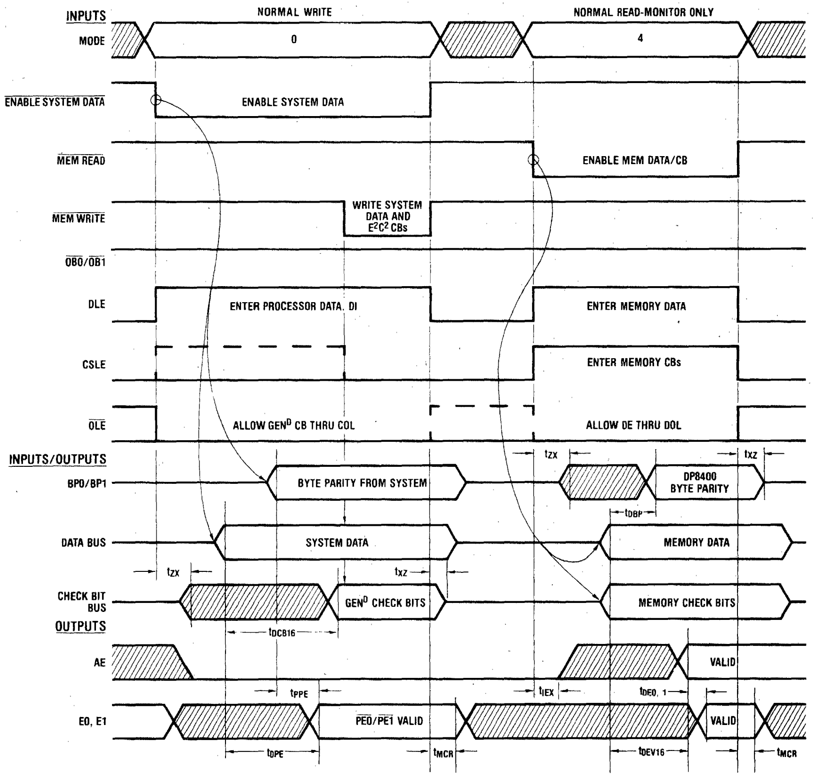


FIGURE 9b. DP8400 16-Bit Configuration, Normal WRITE and Normal READ Timing Diagram

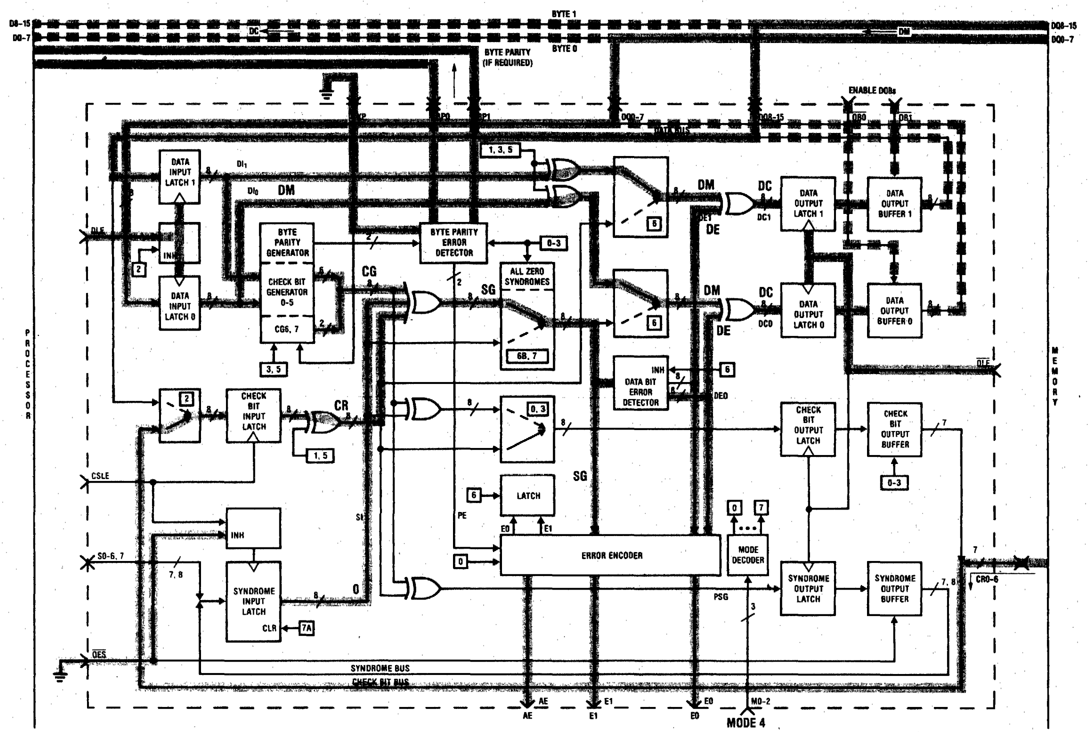
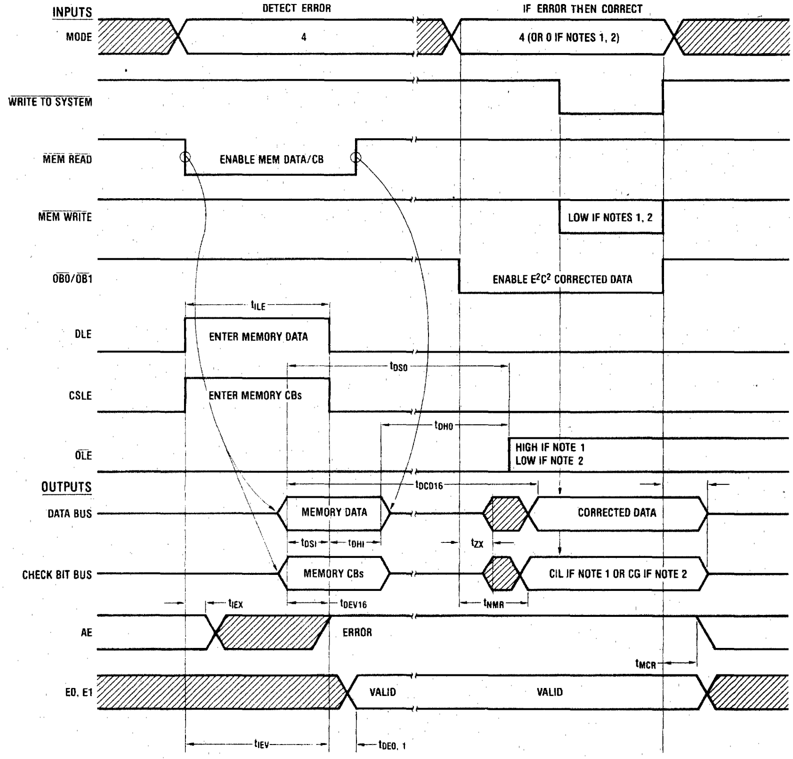


FIGURE 10a. DP8400 16-Bit Configuration, Normal READ — Detect Error (And Correct if Required---)



Note 1: If rewriting correct data and CBs to same location and single data error was detected.

Note 2: If rewriting correct data and CBs to same location and single check bit error was detected.

FIGURE 10b. DP8400 16-Bit Configuration, DETECT THEN CORRECT Timing Diagram

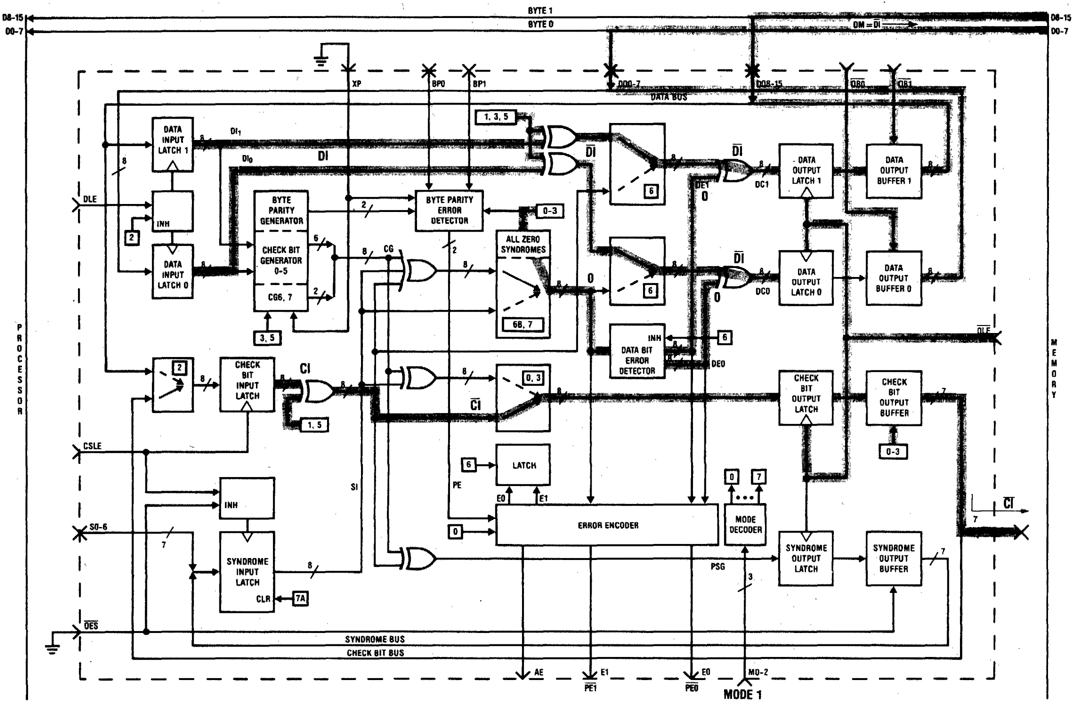
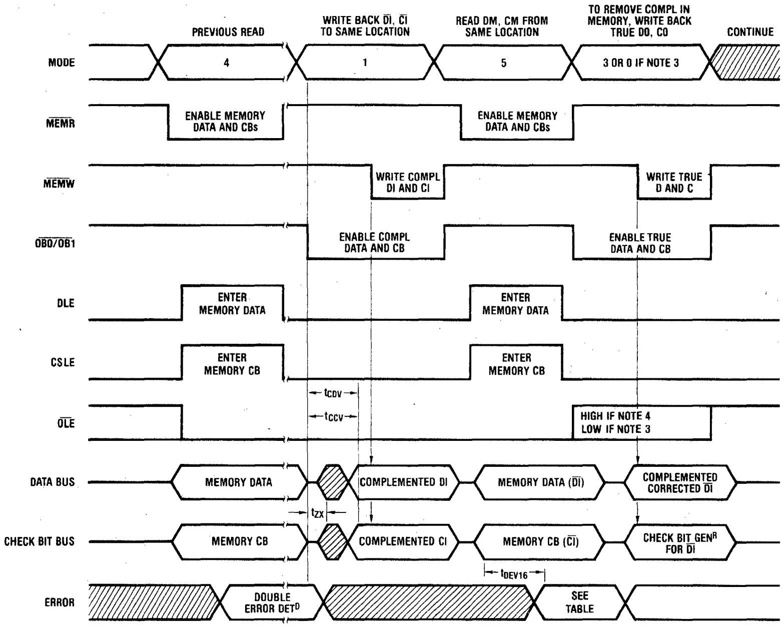


FIGURE 11a. DP8400 16-Bit Configuration, COMPLEMENT WRITE



Note 3: If rewriting corrected data and CBs back to same location and 1 soft data bit error was detected.

Note 4: If rewriting corrected data and CBs back to same location and 2 hard errors or 1 soft check bit was detected.

FIGURE 11b. DP8400 16-Bit Configuration, Detect 2 Errors, COMPLEMENT WRITE, COMPLEMENT READ, Output Corrected Data Timing Diagram

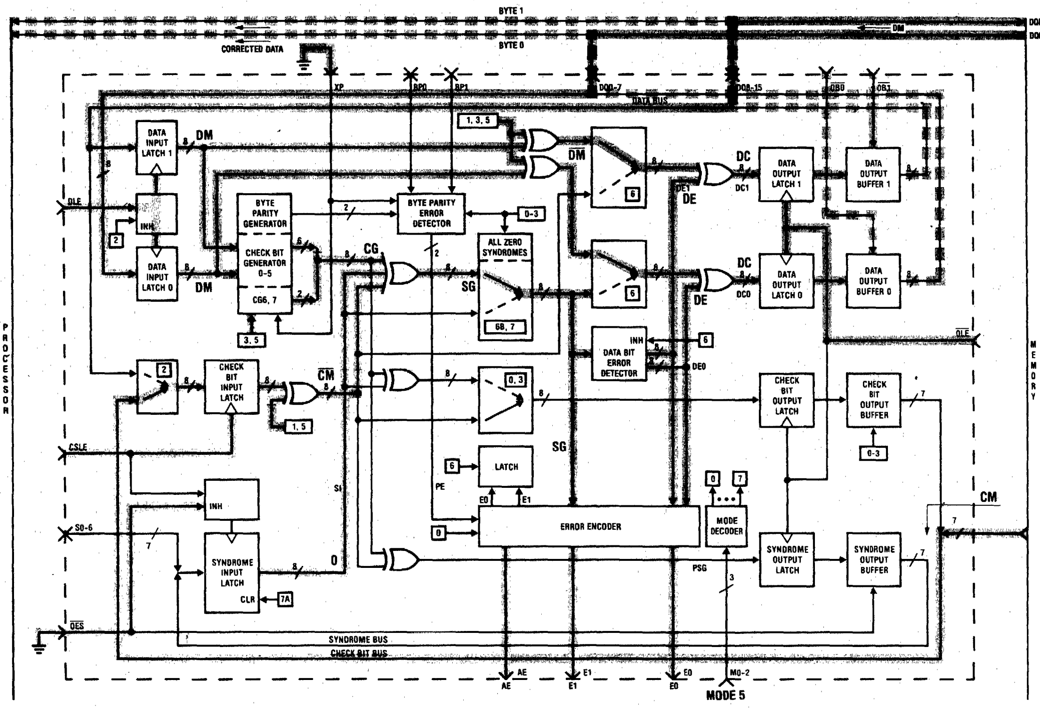


FIGURE 11c. DP8400 16-Bit Configuration, COMPLEMENT READ and Output Corrected if One or Two Hard Errors (---)

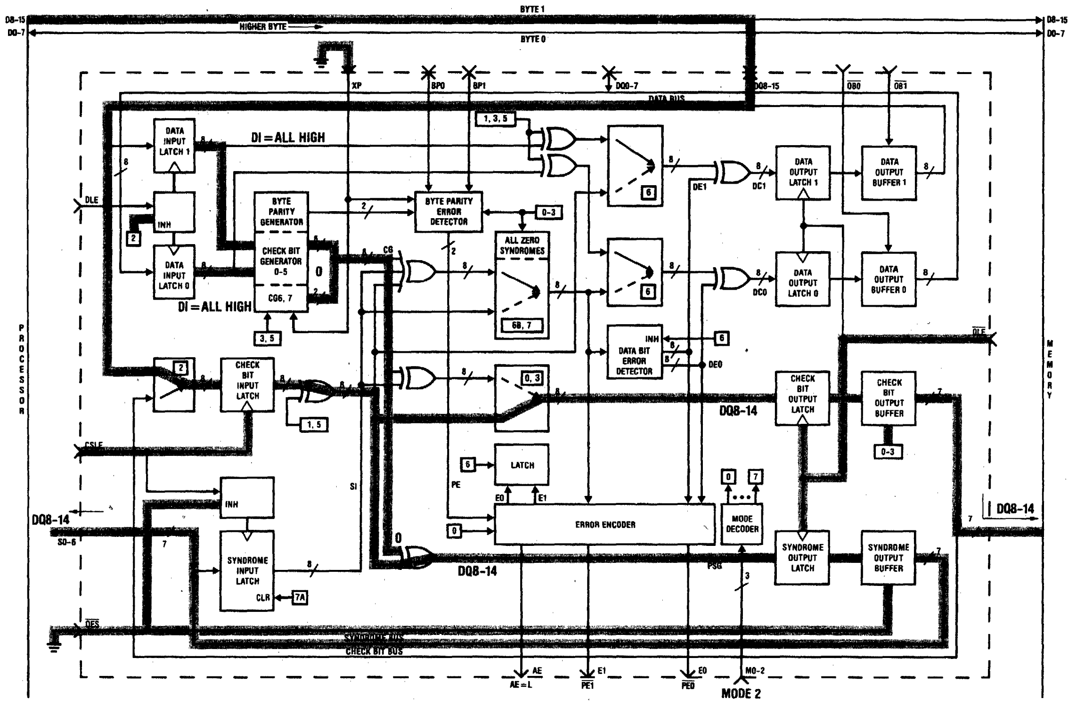


FIGURE 12a. DP8400 16-Bit Configuration, Diagnostic WRITE, READ. Data Bus to Check Bit Bus or Syndrome Bus (Providing DI = HIGH in Previous Cycle to Set CG = All Zero For Transfer to S).

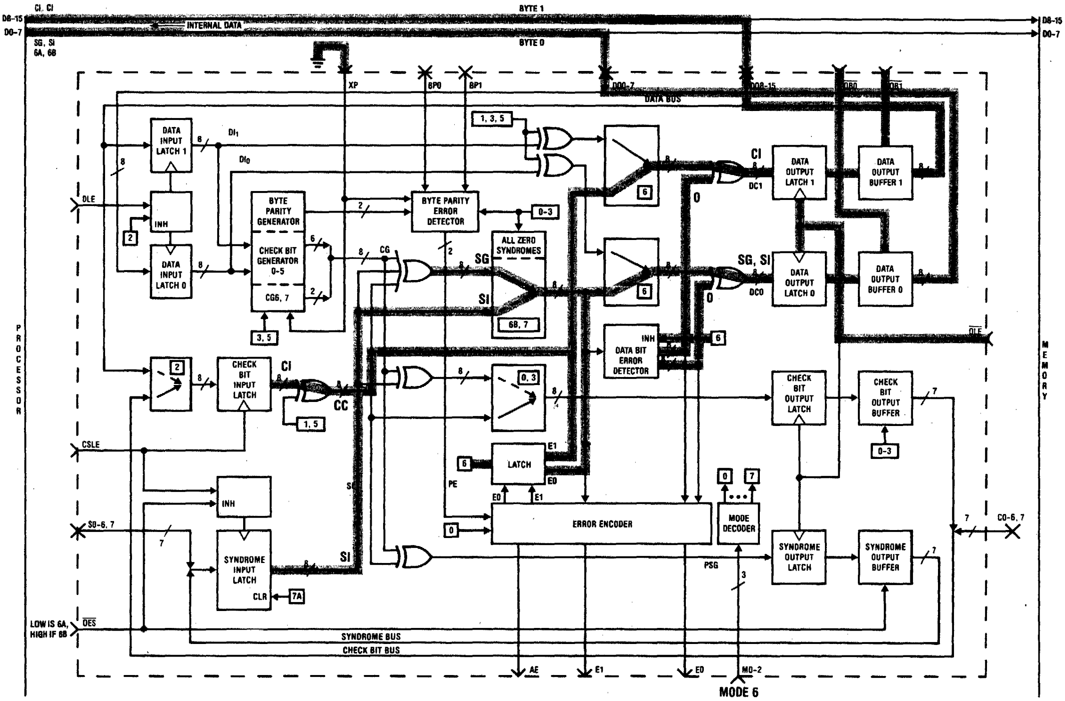


FIGURE 12b. DP8400 16-Bit Configuration, Monitor on Data Bus — Memory Check Bits

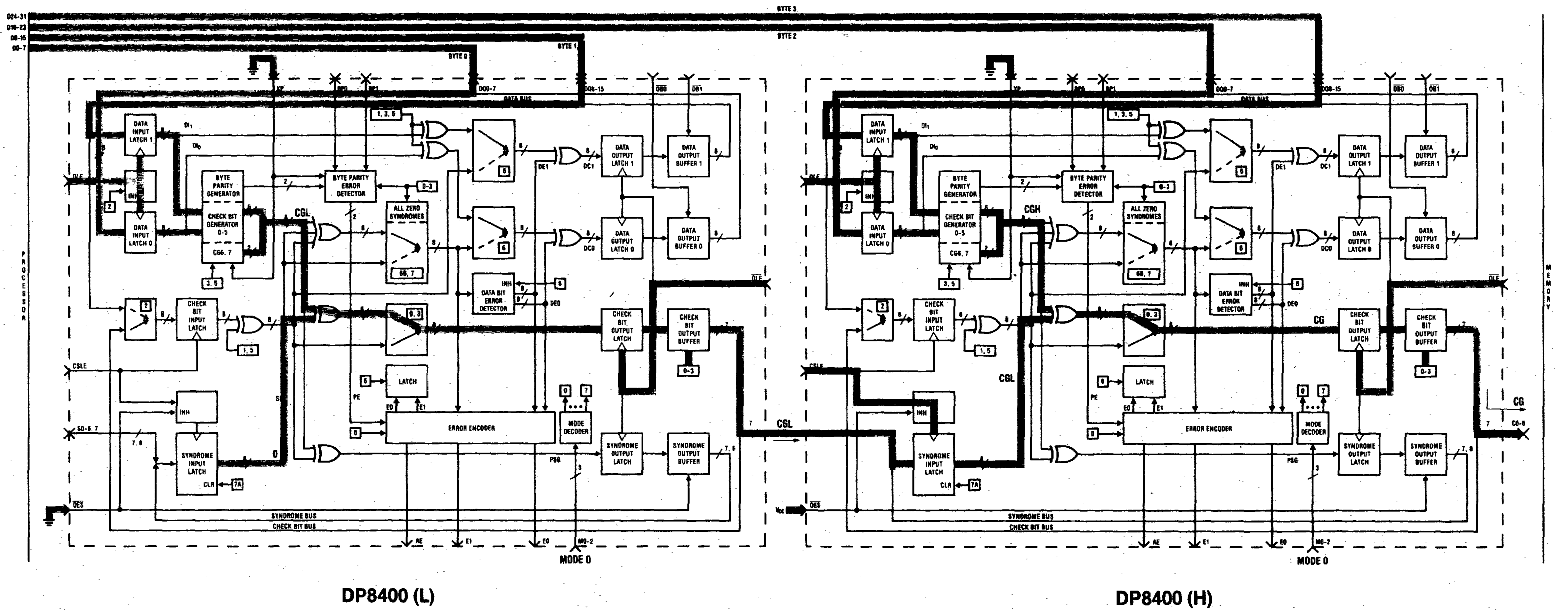


FIGURE 13a. DP8400 32-Bit Configuration, WRITE

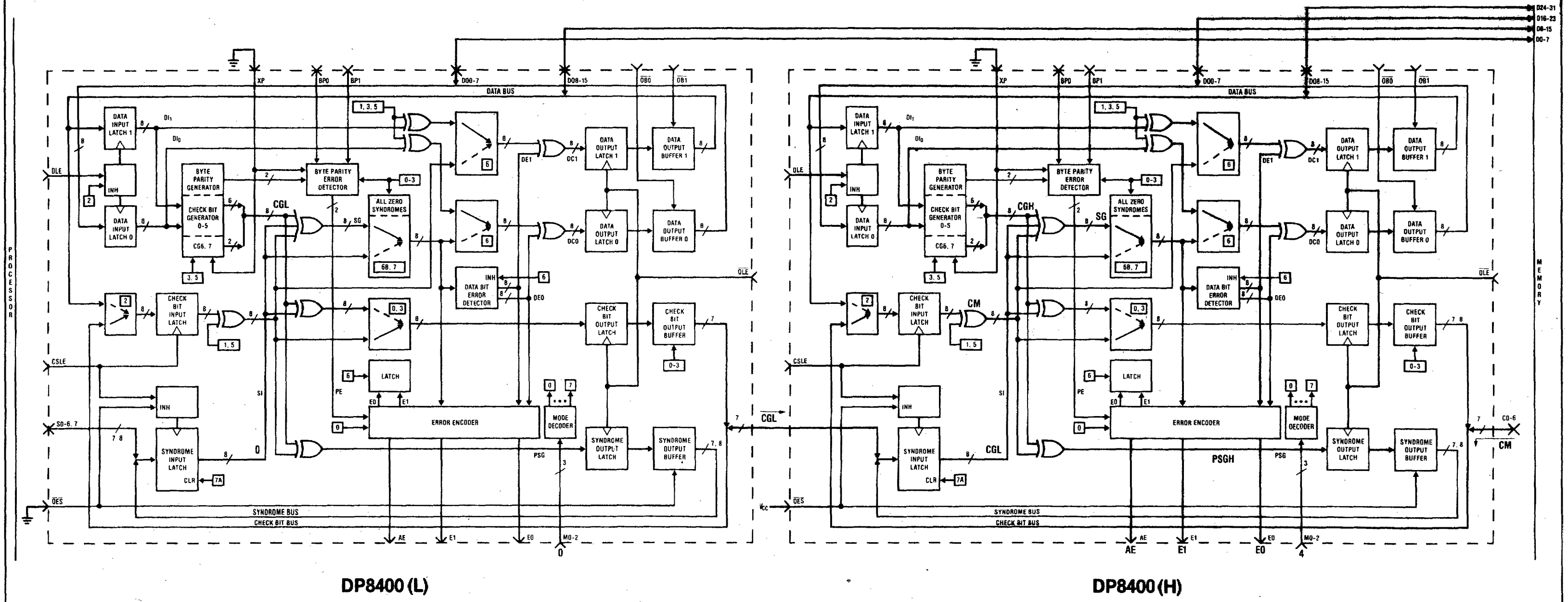
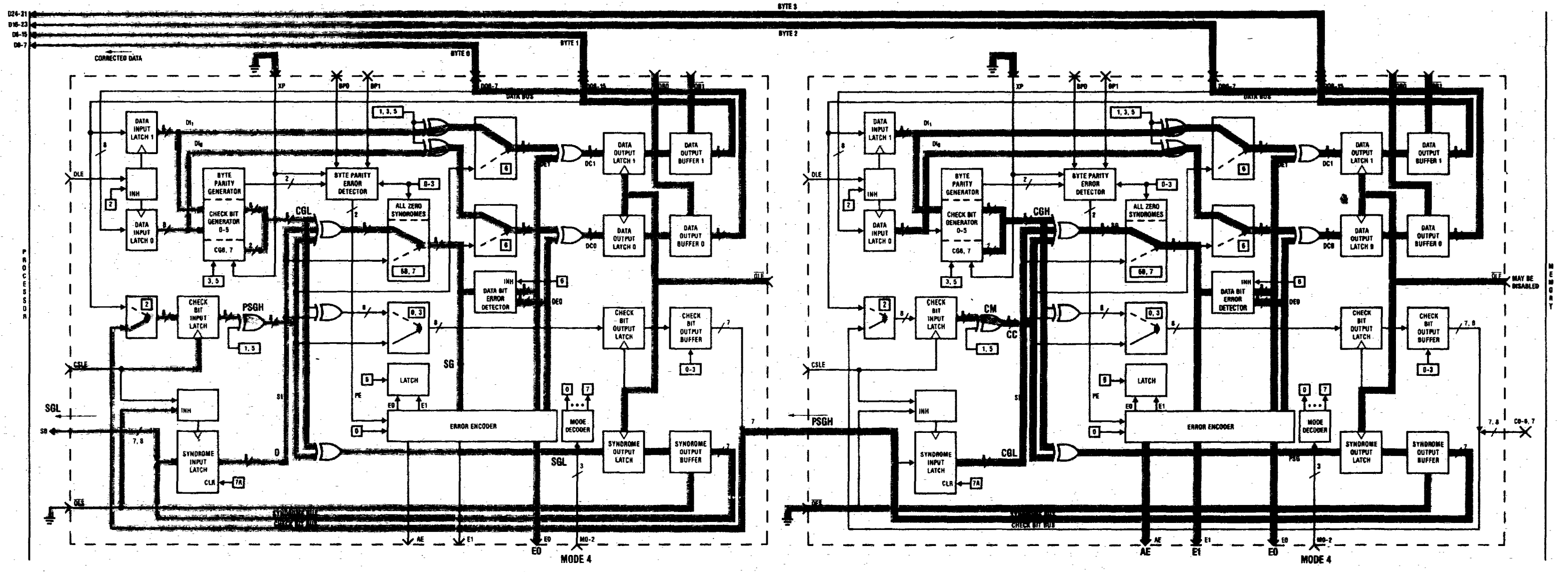


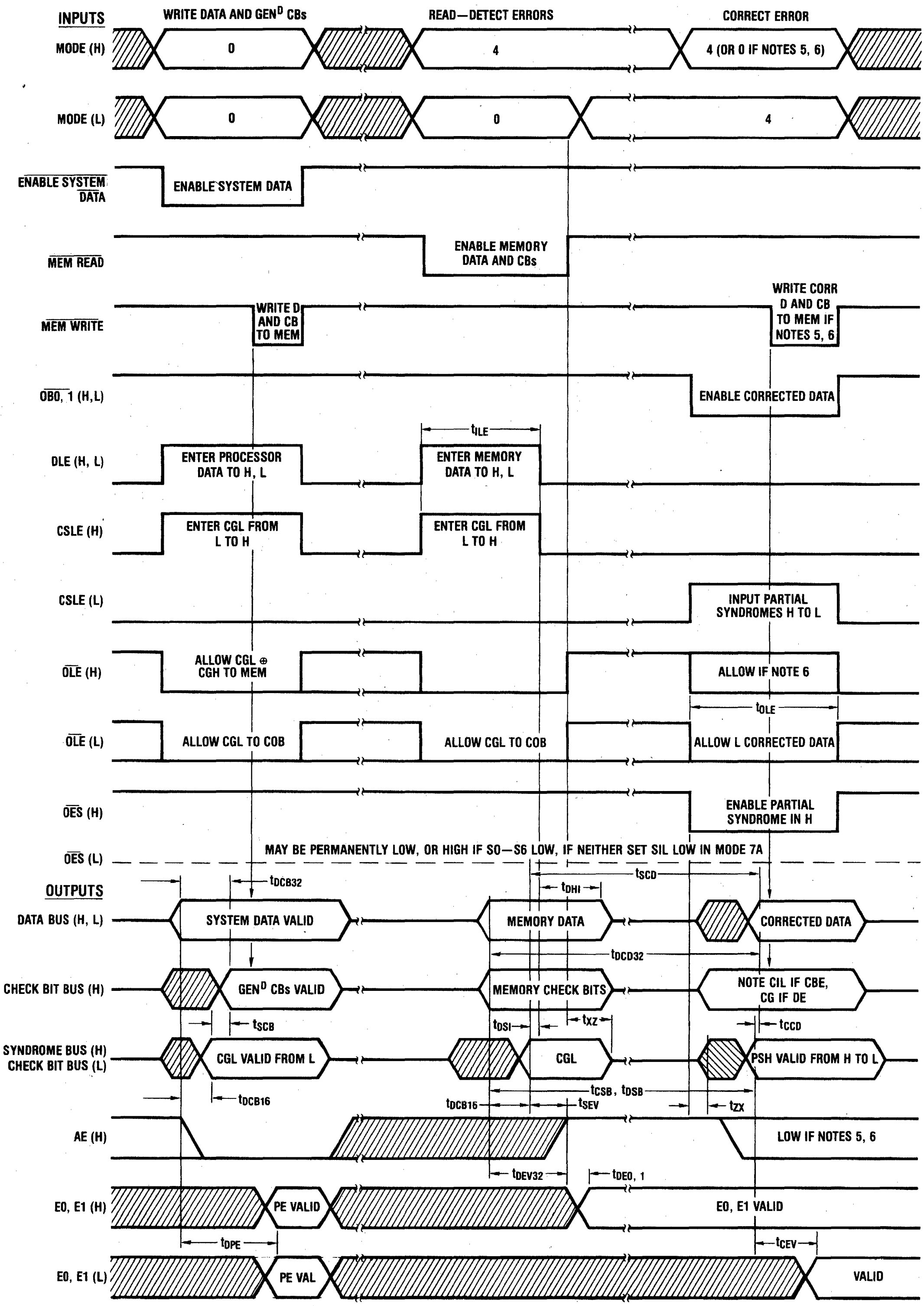
FIGURE 13b. DP8400 32-Bit Configuration, READ Detect Error Only



DP8400 (L)

DP8400 (H)

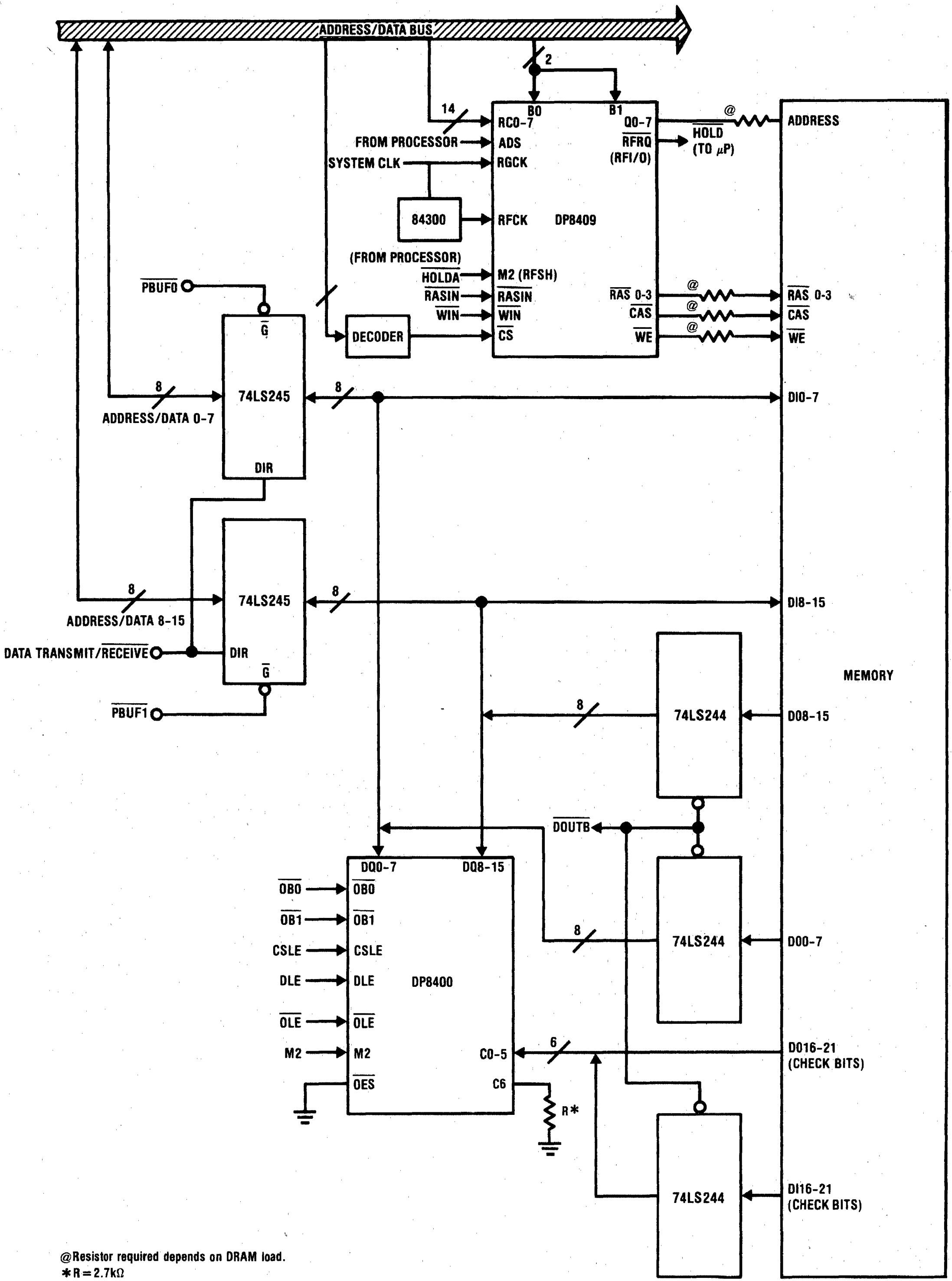
FIGURE 13c. DP8400 32-Bit Configuration, READ Correct Data



Note 5: If rewriting corrected data and CBs back to same location and single data error was detected.

Note 6: If rewriting corrected data and CBs back to same location and single check bit error was detected.

FIGURE 13d. DP8400 32-Bit Configuration, WRITE, DETECT and CORRECT Timing Diagram



@Resistor required depends on DRAM load.
*R = 2.7kΩ

FIGURE 14a. DP8400/8409 System Interface Block Diagram (See Figure 14b for Byte Write Control Timing)

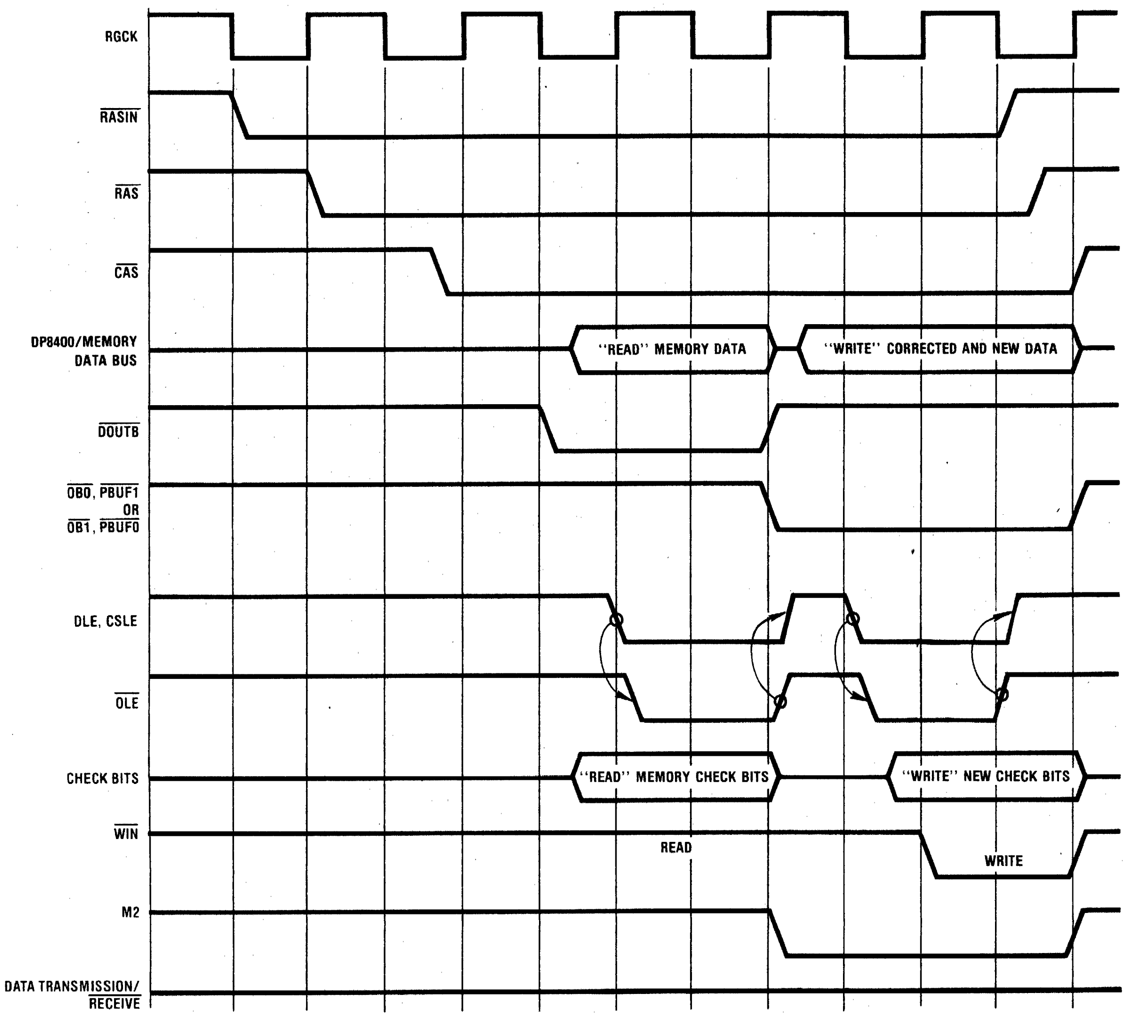
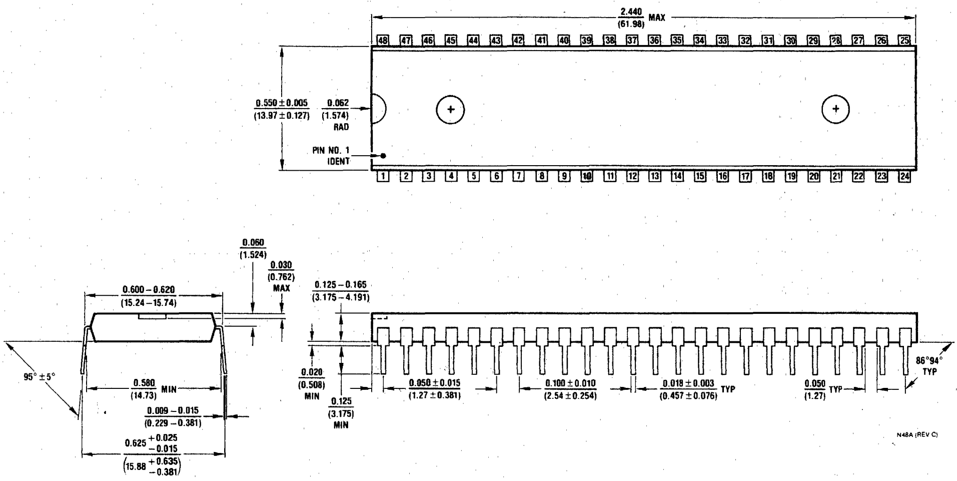


FIGURE 14b. DP8400 16-Bit Configuration, Byte Write Timing

Physical Dimensions inches (millimeters)



Molded Dual-In-Line Package (N)
Order Number DP8400N-4
NS Package Number N48A

National Semiconductor

DP8408 Dynamic RAM Controller/Driver

General Description

Dynamic memory system designs, which formerly required several support chips to drive the memory array, can now be implemented with a single IC. . . the DP8408 Dynamic RAM Controller/Driver. The DP8408 is capable of driving all 16k and 64k Dynamic RAMs (DRAMs). Since the DP8408 is a one-chip solution (including capacitive-load drivers), it minimizes propagation delay skews, the major performance disadvantage of multiple-chip memory drive and control.

The DP8408's 6 modes of operation offer a wide selection of DRAM control capabilities. Memory access may be controlled externally or on-chip automatically; an on-chip refresh counter makes refreshing less complicated.

The DP8408 is a 48-pin DRAM Controller/Driver with 8 multiplexed address outputs and control signals. It consists of two 8-bit address latches, an 8-bit refresh counter, and control logic. All output drivers are capable of driving 500 pF loads with propagation delays of 25 ns. The DP8408 timing parameters are specified driving the typical load capacitance of 88 DRAMs, including trace capacitance.

The DP8408 has 3 mode-control pins: M2, M1, and M0, where M2 is in general $\overline{REFRESH}$. These 3 pins select 6 modes of operation. Inputs B1 and B0 in the memory access modes (M2 = 1), are select inputs which select one of four $\overline{RAS}$ outputs. During normal access, the 8 address outputs can be selected from the Row Address Latch or the Column Address Latch. During refresh, the 8-bit on-chip refresh counter is enabled onto the address bus and in this mode all $\overline{RAS}$ outputs are selected, while $\overline{CAS}$ is inhibited.

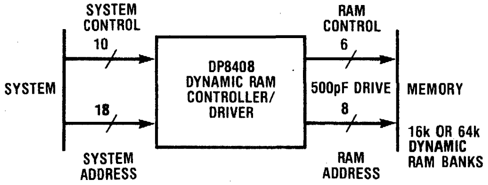
The DP8408 can drive up to 4 banks of DRAMs, with each bank comprised of 16k's, or 64k's. Control signal outputs $\overline{RAS}$, $\overline{CAS}$, and $\overline{WE}$ are provided with the same drive capability. Each $\overline{RAS}$ output drives one bank of DRAMs so that the four $\overline{RAS}$ outputs are used to select the banks, while $\overline{CAS}$, $\overline{WE}$, and the multiplexed addresses can be connected to all of the banks of DRAMs. This leaves the non-selected banks in the standby mode (less than one tenth of the operating power) with the data outputs in TRI-STATE®. Only the bank with its associated $\overline{RAS}$ low will be written to or read from.

Operational Features

- All DRAM drive functions on one chip — minimizes skew on outputs, maximizes AC performance
- On-chip capacitive-load drives (specified to drive up to 88 DRAMs)
- Drives directly all 16k and 64k DRAMs
- Capable of addressing 64k and 256k words
- Propagation delays of 25 ns typical at 500 pF load
- $\overline{CAS}$ goes low automatically after column addresses are valid if desired
- Auto Access mode provides $\overline{RAS}$, Row to Column, select, then $\overline{CAS}$ automatically and fast
- $\overline{WE}$ follows $\overline{WIN}$ unconditionally—offering READ, WRITE or READ-MODIFY-WRITE cycles
- On-chip 8-bit refresh counter with selectable End-of-Count (127 or 255)
- End-of-Count indicated by RF I/O pin going low at 127 or 255
- Low input on RF I/O resets 8-bit refresh counter
- $\overline{CAS}$ inhibited during refresh cycle
- Fall-through latches on address inputs controlled by ADS
- TRI-STATE outputs allow multi-controller addressing of memory
- Control output signals go high-impedance logic "1" when disabled for memory sharing
- Power-up: counter reset, control signals high, address outputs TRI-STATE, and End-of-Count set to 127

Mode Features

- 6 modes of operation: 3 access, 1 refresh, and 2 set-up
- 2 externally controlled modes: 1 access (Mode 4) and 1 refresh (Modes 0, 1, 2)
- 2 auto-access modes $\overline{RAS} \rightarrow R/\overline{C} \rightarrow \overline{CAS}$ automatic, with $t_{RAH} = 20$ or 30 ns minimum (Modes 5, 6)
- Externally controlled All-$\overline{RAS}$ Access modes for memory initialization (Mode 3)
- End-of-Count value of Refresh Counter set by B1 and B0 (Mode 7)



DP8408 Interface Between System & DRAM Banks

Block Diagram

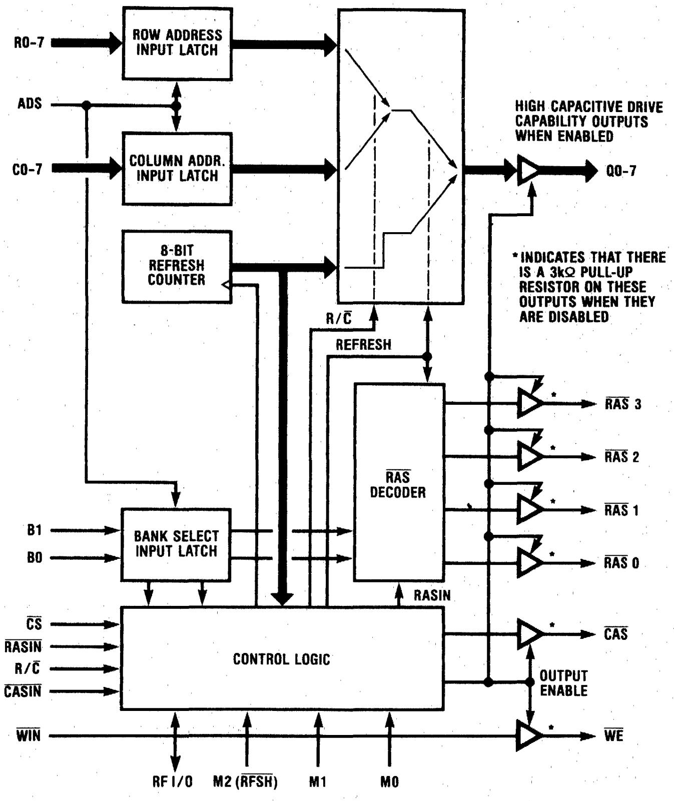
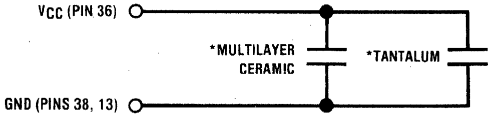


Table 1. DP8408 Mode Select Options

Mode	(RFSH) M2	M1	M0	Mode of Operation	Conditions
0	0	0	0	Externally Controlled Refresh	RF I/O = EOC
1	0	0	1		
2	0	1	0		
3	0	1	1	Externally Controlled All-RAS Write	All-RAS Active
4	1	0	0	Externally Controlled Access	Active RAS defined by Table 2
5	1	0	1	Auto Access, Slow t_{RAH}	Active RAS defined by Table 2
6	1	1	0	Auto Access, Fast t_{RAH}	Active RAS defined by Table 2
7	1	1	1	Set End of Count	See Table 3 for Mode 7

Pin Definitions

V_{CC}, GND, GND — V_{CC} = 5V ± 5%. The three supply pins have been assigned to the center of the package to reduce voltage drops, both DC and AC. There are also two ground pins to reduce the low level noise. The second ground pin is located two pins from V_{CC}, so that decoupling capacitors can be inserted directly next to these pins. It is important to adequately decouple this device, due to the high switching currents that will occur when all 8 address bits change in the same direction simultaneously. A recommended solution would be a 1µF multilayer ceramic capacitor in parallel with a low-voltage tantalum capacitor, both connected as close as possible to pins 36 and 38 to reduce lead inductance. See Figure below.



*Capacitor values should be chosen depending on the particular application.

R0-R7: Row Address Inputs.

C0-C7: Column Address Inputs.

Q0-Q7: Multiplexed Address Outputs — Selected from the Row Address Input Latch, the Column Address Input Latch, or the Refresh Counter.*

$\overline{\text{RASIN}}$: Row Address Strobe Input — Enables selected $\overline{\text{RAS}}_n$ output when M2 ($\overline{\text{RFSH}}$) is high, or all $\overline{\text{RAS}}_n$ outputs when $\overline{\text{RFSH}}$ is low.

R/$\overline{\text{C}}$: Row/Column Select Input — Selects either the row or column address input latch onto the output bus.

$\overline{\text{CASIN}}$: Column Address Strobe Input — Inhibits $\overline{\text{CAS}}$ output when high in Modes 4 and 3. In Mode 6 it can be used to prolong $\overline{\text{CAS}}$ output.

ADS: Address (Latch) Strobe Input — Row Address, Column Address, and Bank Select Latches are fall-through with ADS high; Latches on high-to-low transition.

$\overline{\text{CS}}$: Chip Select Input — TRI-STATE's the Address Outputs and puts the control signal into a high-impedance logic "1" state when high (except in Mode 0); enables all outputs when low.

M0, M1, M2: Mode Control Inputs — These 3 control pins determine the 6 major modes of operation of the DP8408 as depicted in Table 1.

RF I/O — The I/O pin functions as a Reset Counter Input when set low from an external open-collector gate, or as a flag output. The flag goes active-low when M2 = 0 and the End-of-Count output is at 127 or 255 (see Table 3).

$\overline{\text{WIN}}$: Write Enable Input.

$\overline{\text{WE}}$: Write Enable Output — Buffered output from $\overline{\text{WIN}}$.*

$\overline{\text{CAS}}$: Column Address Strobe Output — In Modes 5 and 6, $\overline{\text{CAS}}$ goes low following valid column address. In Modes 3 and 4, it transitions low after R/$\overline{\text{C}}$ goes low, or follows $\overline{\text{CASIN}}$ going low if R/$\overline{\text{C}}$ is already low. $\overline{\text{CAS}}$ is high during refresh.*

$\overline{\text{RAS}}$ 0-3: Row Address Strobe Outputs — Selects a memory bank decoded from B1 and B0 (see Table 2), if $\overline{\text{RFSH}}$ is high. If $\overline{\text{RFSH}}$ is low, all banks are selected.*

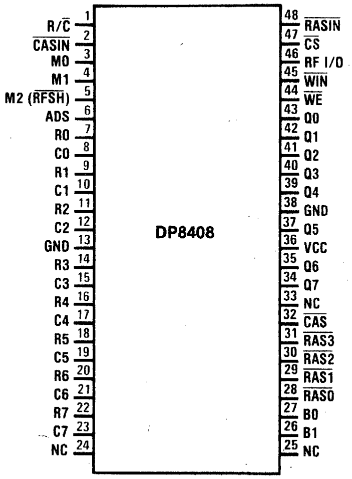
B0, B1: Bank Select Inputs — Strobed by ADS. Decoded to enable one of the $\overline{\text{RAS}}$ outputs when $\overline{\text{RASIN}}$ goes low. Also used to define End-of-Count in Mode 7 (Table 3).

*These outputs may need damping resistors to prevent overshoot, undershoot. See AN-305 "Precautions to Take When Driving Memories."

Table 2. Memory Bank Decode

Bank Select (Strobed by ADS)		Enabled $\overline{\text{RAS}}_n$
B1	**B0**	
0	0	$\overline{\text{RAS}}_0$
0	1	$\overline{\text{RAS}}_1$
1	0	$\overline{\text{RAS}}_2$
1	1	$\overline{\text{RAS}}_3$

Connection Diagram



NC = NO CONNECTION

Conditions for all Modes

Input Addressing

The address block consists of a row-address latch, a column-address latch, and a resettable refresh counter. The address latches are fall-through when ADS is high and latch when ADS goes low. If the address bus contains valid addresses until after the valid address time, ADS can be permanently high. Otherwise ADS must go low while the addresses are still valid.

In normal memory access operation, $\overline{RASIN}$ and R/$\overline{C}$ are initially high. When the address inputs are enabled into the address latches, the row addresses appear on the Q outputs. The address strobe also inputs the bank-select address, (B0 and B1). If $\overline{CS}$ is low, all outputs are enabled. When $\overline{CS}$ is transitioned high, the address outputs go TRI-STATE and the control outputs first go high through a low impedance, and then are held by an on-chip high impedance. This allows output paralleling with other DP8408s for multi-addressing. All outputs go active about 50 ns after the chip is selected again. If $\overline{CS}$ is high, and a refresh cycle begins, all the outputs become active until the end of the refresh cycle.

Drive Capability

The DP8408 has timing parameters that are specified with up to 600 pF loads. In a typical memory system this is equivalent to about 88, 5V-only DRAMs, with trace lengths kept to a minimum. Therefore, the chip can drive four banks each of 16 or 22 bits, or two banks of 32 or 39 bits, or one bank of 64 or 72 bits.

Less loading will slightly reduce the timing parameters, and more loading will increase the timing parameters, according to the graph of Figure 6. The AC performance parameters are specified with the typical load capacitance of 88 DRAMs. This graph can be used to extrapolate the variations expected with other loading.

Because of distributed trace capacitance and inductance and DRAM input capacitance, current spikes can be created, causing overshoots and undershoots at the DRAM inputs that can change the contents of the DRAMs or even destroy them. To remove these spikes, a damping resistor (low inductance, carbon) can be inserted between the DP8408 driver outputs and the DRAMs, as close as possible to the DP8408. The values of the damping resistors may differ between the different control outputs; $\overline{RAS}$'s $\overline{CAS}$, Q's and $\overline{WE}$. The damping resistors should be determined by the first prototypes (not wire-wrapped due to larger distributed capacitance and inductance). The best values for the damping resistors are the critical values giving a critically damped transition on the control outputs. Typical values for the damping resistors will be between 15Ω and 100Ω, the lower the loading the higher the value. (For more information, see AN-305 "Precautions to Take When Driving Memories.")

DP8408 Driving any 16k or 64k DRAMs

The DP8408 can drive any 16k or 64k DRAMs. All 16k DRAMs are basically the same configuration, including the newer 5V-only version. Hence, in most applications, different manufacturers' DRAMs are interchangeable (for the same supply-rail chips), and the DP8408 can drive all 16k DRAMs (see Figure 1a).

There are three basic configurations for the 5V-only 64k DRAMs: a 128-row by 512-column array with an on-RAM refresh counter, a 128-row by 512-column array with no on-RAM refresh counter, and a 256-row by 256-column array with no on-RAM refresh counter. The DP8408 can drive all three configurations, and at the same time allows them all to be interchangeable (as shown in Figures 1b and 1c), providing maximum flexibility in the choice of DRAMs. Since the 8-bit on-chip refresh counter can be used as a 7-bit refresh counter for the 128-row configuration, or as an 8-bit refresh counter for the 256-row configuration, the on-RAM refresh counter (if present) is never used. As long as 128 rows are refreshed every 2 ms (i.e. 256 rows in 4 ms) all DRAM types are correctly refreshed.

When the DP8408 is in a refresh mode, the RF I/O pin indicates that the on-chip refresh counter has reached its end-of-count. This end-of-count is selectable as 127 or 255 to accommodate 16k or 64k DRAMs, respectively. Although the end-of-count may be chosen to be either of these values, the counter is not reset and always counts to 255 before rolling over to zero.

Read, Write, and Read-Modify-Write Cycles

The output signal, $\overline{WE}$, determines what type of memory access cycle the memory will perform. If $\overline{WE}$ is kept high while $\overline{CAS}$ goes low, a read cycle occurs. If $\overline{WE}$ goes low before $\overline{CAS}$ goes low, a write cycle occurs and data at DI (DRAM input data) is written into the DRAM as $\overline{CAS}$ goes low. If $\overline{WE}$ goes low later than t_{CWD} after $\overline{CAS}$ goes low, first a read occurs and DO (DRAM output data) becomes valid; then data DI is written into the same address in the DRAM when $\overline{WE}$ goes low. In this read-modify-write case, DI and DO cannot be linked together. The type of cycle is therefore controlled by $\overline{WE}$, which follows $\overline{WIN}$.

Power-Up Initialize

When V_{CC} is first applied to the DP8408, an initialize pulse clears the refresh counter, the internal control flip-flops, and sets the End-of-Count of the refresh counter to 127 (which may be changed via Mode 7). As V_{CC} increases to about 2.3 volts, it holds the output control signals at a level of one Schottky diode-drop below V_{CC}, and the output address to TRI-STATE. As V_{CC} increases above 2.3 volts, control of these outputs is granted to the system.

DP8408 Driving any 16k or 64k Dynamic RAMs

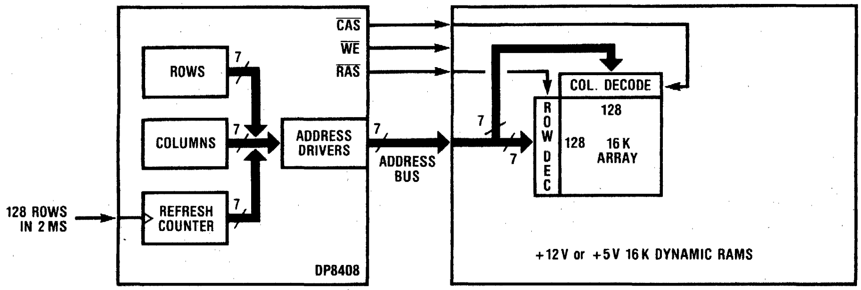
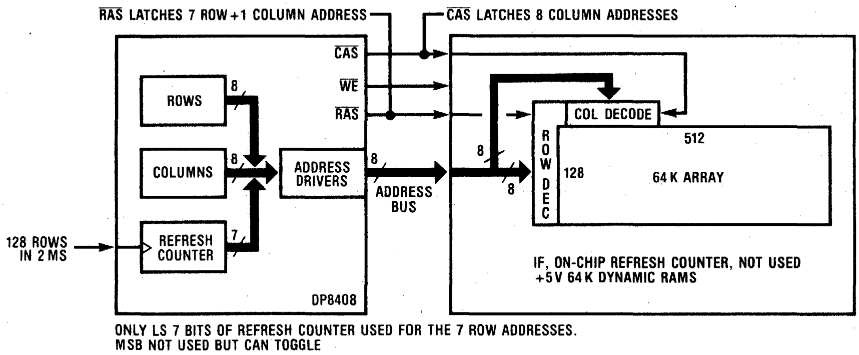


FIGURE 1a. DP8408 with any 16k DRAMS



RAS LATCHES 7 ROW+1 COLUMN ADDRESS
CAS LATCHES 8 COLUMN ADDRESSES
CAS
WE
RAS
ROWS
8
COLUMNS
8
REFRESH COUNTER
7
ADDRESS DRIVERS
8
ADDRESS BUS
128 ROWS IN 2 MS
DP8408
8
8
ROW DEC
COL DECODE
512
128
64 K ARRAY
IF, ON-CHIP REFRESH COUNTER, NOT USED
+5V 64 K DYNAMIC RAMS
ONLY LS 7 BITS OF REFRESH COUNTER USED FOR THE 7 ROW ADDRESSES.
MSB NOT USED BUT CAN TOGGLE

FIGURE 1b. DP8408 with 128 Row × 512 Column 64k DRAM

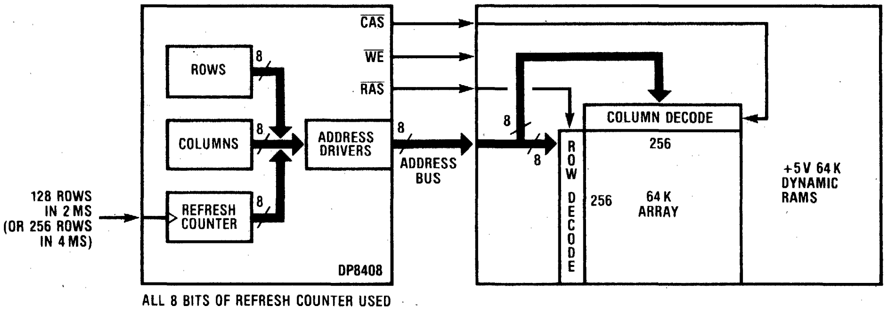


FIGURE 1c. DP8408 with 256 × 256 Column 64k DRAM

DP8408 Functional Mode Descriptions

Note: All delay parameters stated in text refer to the DP8408. Substitute the respective delay numbers for the DP8408-2 or DP8408-3 when using these devices.

Modes 0,1,2 — Externally Controlled Refresh

In this mode, the input address latches are disabled from the address outputs and the refresh counter is enabled. When RAS occurs, the enabled row in the DRAM is refreshed. In the Externally Controlled Refresh mode, all RAS outputs are enabled following RASIN, and CAS is inhibited. This refreshes the same row in all four banks. The refresh counter increments when either RASIN or RFSH goes low-to-high after a refresh. RF I/O goes low when the count is 127 or 255, as set by End-of-Count (see Table 3), with RASIN and RFSH low. To reset the counter to all zeroes, RF I/O is set low through an external open-collector driver.

During refresh, RASIN and RFSH must be skewed transitioning low such that the refresh address is valid on the address outputs of the controller before the RAS outputs go low. The amount of time that RFSH should go low before RASIN does depends on the capacitive loading of the address and RAS lines. For the load specified in the switching characteristics of this data sheet, 10ns is sufficient. Refer to Figure 2.

To perform externally controlled burst refresh, RASIN is toggled while RFSH is held low. The refresh counter increments with RASIN going low to high, so that the DRAM rows are refreshed in succession by RASIN going high to low.

Mode 3 — Externally Controlled All-RAS Write

This mode is useful at system initialization. The memory address is provided by the processor, which also performs the incrementing. All four RAS outputs follow RASIN (supplied by the processor), strobing the row address into the DRAMs. R/C can now go low, while CASIN may be used to control CAS (as in the Externally Controlled Access mode), so that CAS strobes the column address contents into the DRAMs. At this time WE should be low, causing the data to be written into all four banks of DRAMs. At the end of the write cycle, the input address is incremented and latched by the DP8408 for the next write cycle.

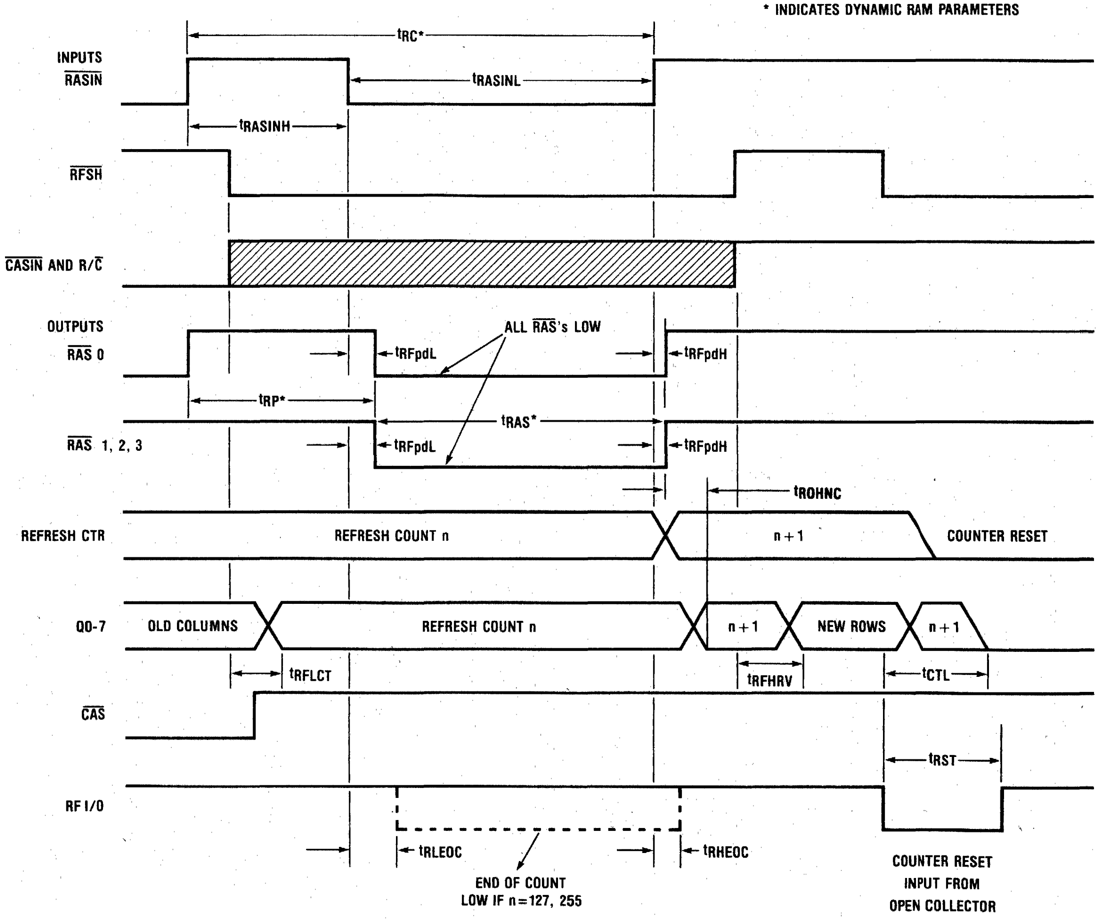


FIGURE 2. External Control Refresh Cycle (MODES 0,1,2)

Mode 4 — Externally Controlled Access

This mode facilitates externally controlling all access-timing parameters associated with the DRAMs. The application of modes 0 and 4 are shown in Figure 3.

Output Address Selection

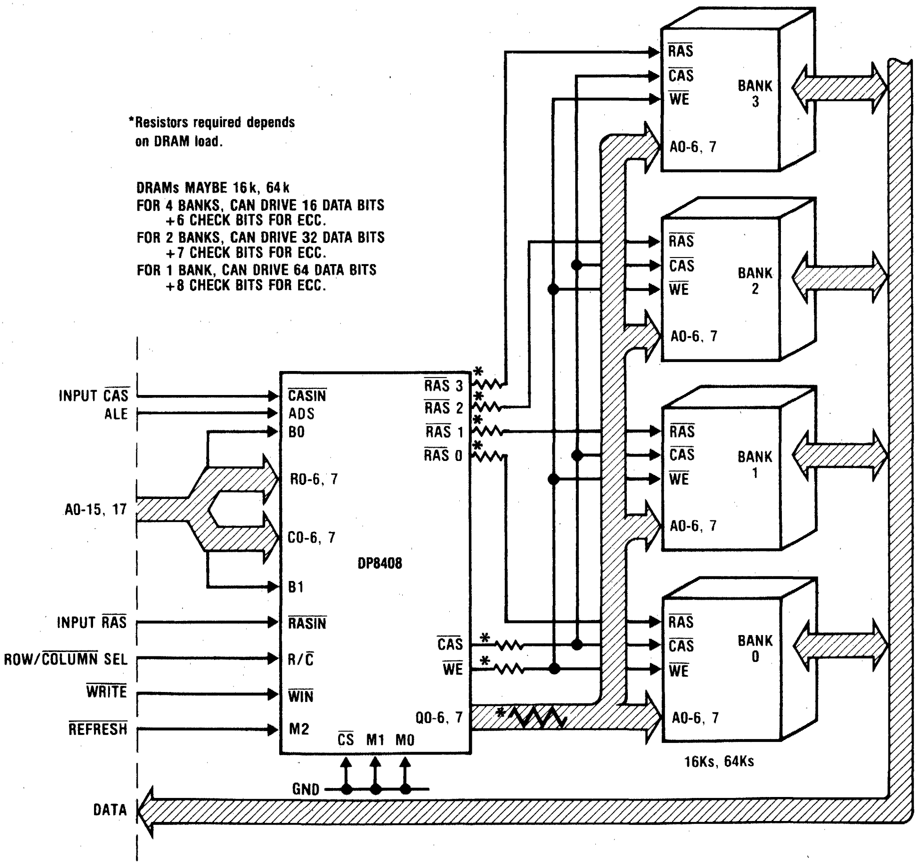
Refer to Figure 4a. With M2 ($\overline{\text{RFSH}}$) and R/$\overline{\text{C}}$ high, the row address latch contents are transferred to the multiplexed address bus output Q0–Q7, provided $\overline{\text{CS}}$ is set low. The column address latch contents are output after R/$\overline{\text{C}}$ goes low. $\overline{\text{RASIN}}$ can go low after the row addresses have been set up on Q0–Q7. This selects one of the $\overline{\text{RAS}}$ outputs, strobing the row address on the Q outputs into the desired bank of memory. After the row-address hold-time of the DRAMs, R/$\overline{\text{C}}$ can go low so that about 40ns later column addresses appear on the Q outputs.

Automatic $\overline{\text{CAS}}$ Generation

In a normal memory access cycle $\overline{\text{CAS}}$ can be derived from inputs $\overline{\text{CASIN}}$ or R/$\overline{\text{C}}$. If $\overline{\text{CASIN}}$ is high, then R/$\overline{\text{C}}$ going low switches the address output drivers from rows to columns. $\overline{\text{CASIN}}$ then going low causes $\overline{\text{CAS}}$ to go low approximately 40 ns later, allowing $\overline{\text{CAS}}$ to occur at a predictable time (see Figure 4b). If $\overline{\text{CASIN}}$ is low when R/$\overline{\text{C}}$ goes low, $\overline{\text{CAS}}$ will be automatically generated, following the row to column transition by about 20 ns (see Figure 4a). Most DRAMs have a column address set-up time before $\overline{\text{CAS}}$ (t_{ASC}) of 0 ns or – 10 ns. In other words, a t_{ASC} greater than 0 ns is safe. This feature reduces timing-skew problems, thereby improving access time of the system.

Fast Memory Access

AC parameters t_{DIF1}, t_{DIF2} may be used to determine the minimum delays required between $\overline{\text{RASIN}}$, R/$\overline{\text{C}}$, and $\overline{\text{CASIN}}$ (see Application Brief 9; "Fastest DRAM Access Mode").


*Resistors required depends on DRAM load.

DRAMs MAYBE 16k, 64k
FOR 4 BANKS, CAN DRIVE 16 DATA BITS
+6 CHECK BITS FOR ECC.
FOR 2 BANKS, CAN DRIVE 32 DATA BITS
+7 CHECK BITS FOR ECC.
FOR 1 BANK, CAN DRIVE 64 DATA BITS
+8 CHECK BITS FOR ECC.

INPUT $\overline{\text{CAS}}$
ALE
A0-15, 17
INPUT $\overline{\text{RAS}}$
ROW/$\overline{\text{COLUMN}}$ SEL
$\overline{\text{WRITE}}$
$\overline{\text{REFRESH}}$
DATA
$\overline{\text{CASIN}}$
ADS
B0
R0-6, 7
C0-6, 7
DP8408
B1
$\overline{\text{RASIN}}$
R/$\overline{\text{C}}$
$\overline{\text{WIN}}$
M2
$\overline{\text{CS}}$ M1 M0
GND
$\overline{\text{RAS}}$ 3
$\overline{\text{RAS}}$ 2
$\overline{\text{RAS}}$ 1
$\overline{\text{RAS}}$ 0
$\overline{\text{CAS}}$
$\overline{\text{WE}}$
Q0-6, 7
*
$\overline{\text{RAS}}$
$\overline{\text{CAS}}$
$\overline{\text{WE}}$
A0-6, 7
BANK 3
BANK 2
BANK 1
BANK 0
16Ks, 64Ks

FIGURE 3. Typical Application of DP8408 Using Externally Controlled Access and Refresh in Modes 0 and 4

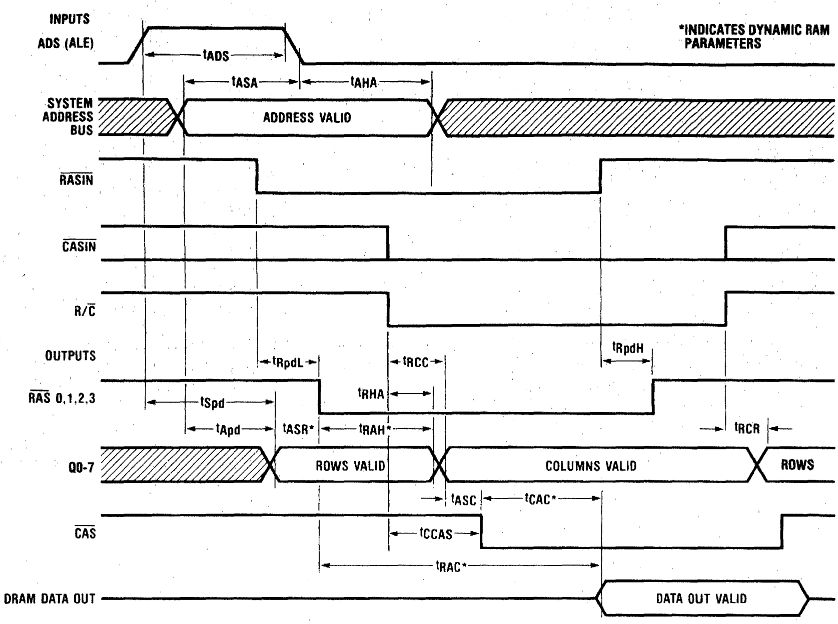


FIGURE 4a. Read Cycle Timing (Mode 4)

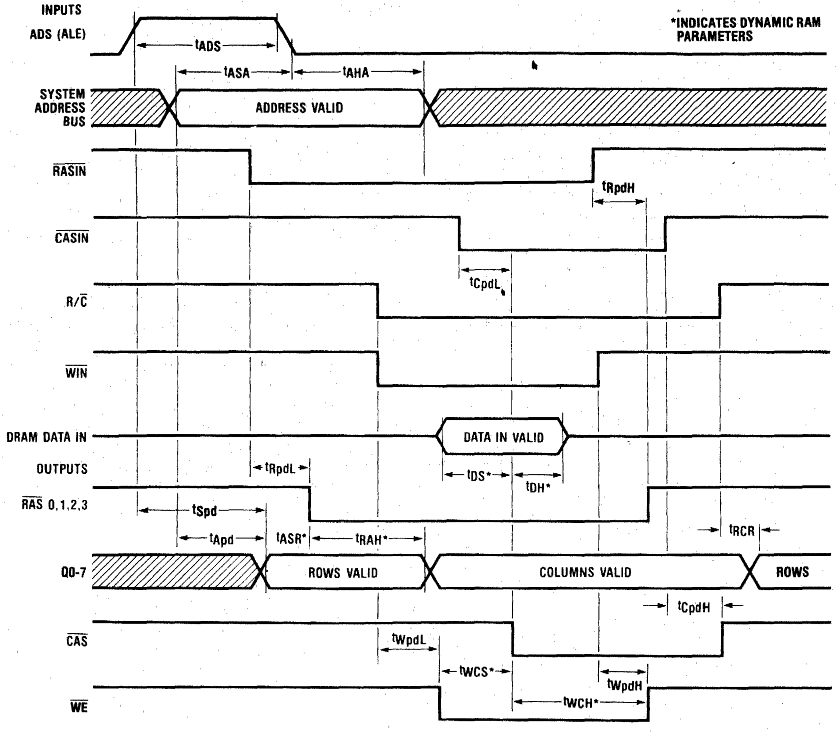


FIGURE 4b. Write Cycle Timing (Mode 4)

Mode 5 — Automatic Access

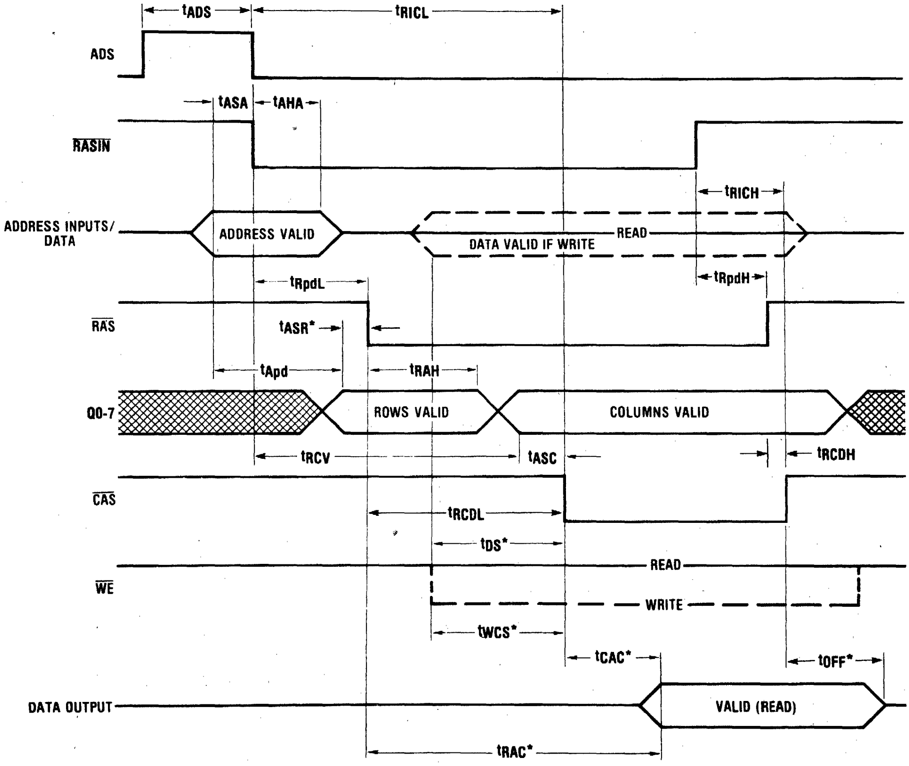
The Auto Access mode has two advantages over the Externally Controlled Access mode, due to the fact that all outputs except $\overline{\text{WE}}$ are initiated from $\overline{\text{RASIN}}$. First, inputs R/$\overline{\text{C}}$ and $\overline{\text{CASIN}}$ are unnecessary. Secondly, because the output control signals are derived internally from one input signal ($\overline{\text{RASIN}}$), timing-skew problems are reduced, thereby reducing memory access time substantially or allowing use of slower DRAMs. The automatic access features of Mode 5 (and Mode 6) of the DP8408 make DRAM accessing appear essentially "static".

Automatic Access Control

The major disadvantage of DRAMs compared to static RAMs is the complex timing involved. First, a $\overline{\text{RAS}}$ must occur with the row address previously set up on the multiplexed address bus. After the row address has been held for t_{RAH}, (the Row-Address hold-time of the DRAM), the column address is set up and then $\overline{\text{CAS}}$ occurs. This is all performed automatically by the DP8408 in this mode.

Provided the input address is valid as ADS goes low, $\overline{\text{RASIN}}$ can go low any time after ADS. This is because the selected $\overline{\text{RAS}}$ occurs typically 27ns later, by which time the row address is already valid on the address output of the DP8408. The Address Set-Up time (t_{ASR}), is 0ns on most DRAMs. The DP8408 in this mode (with ADS and $\overline{\text{RASIN}}$ edges simultaneously applied) produces a minimum t_{ASR} of 0ns. This is true provided the input address was valid t_{ASA} before ADS went low (see Figure 5a).

Next, the row address is disabled after t_{RAH} (30ns minimum); in most DRAMs, t_{RAH} minimum is less than 30ns. The column address is then set up and t_{ASC} later, $\overline{\text{CAS}}$



*INDICATES DYNAMIC RAM PARAMETERS

FIGURE 5a. Modes 5, 6 Timing ($\overline{\text{CASIN}}$ High in Mode 6

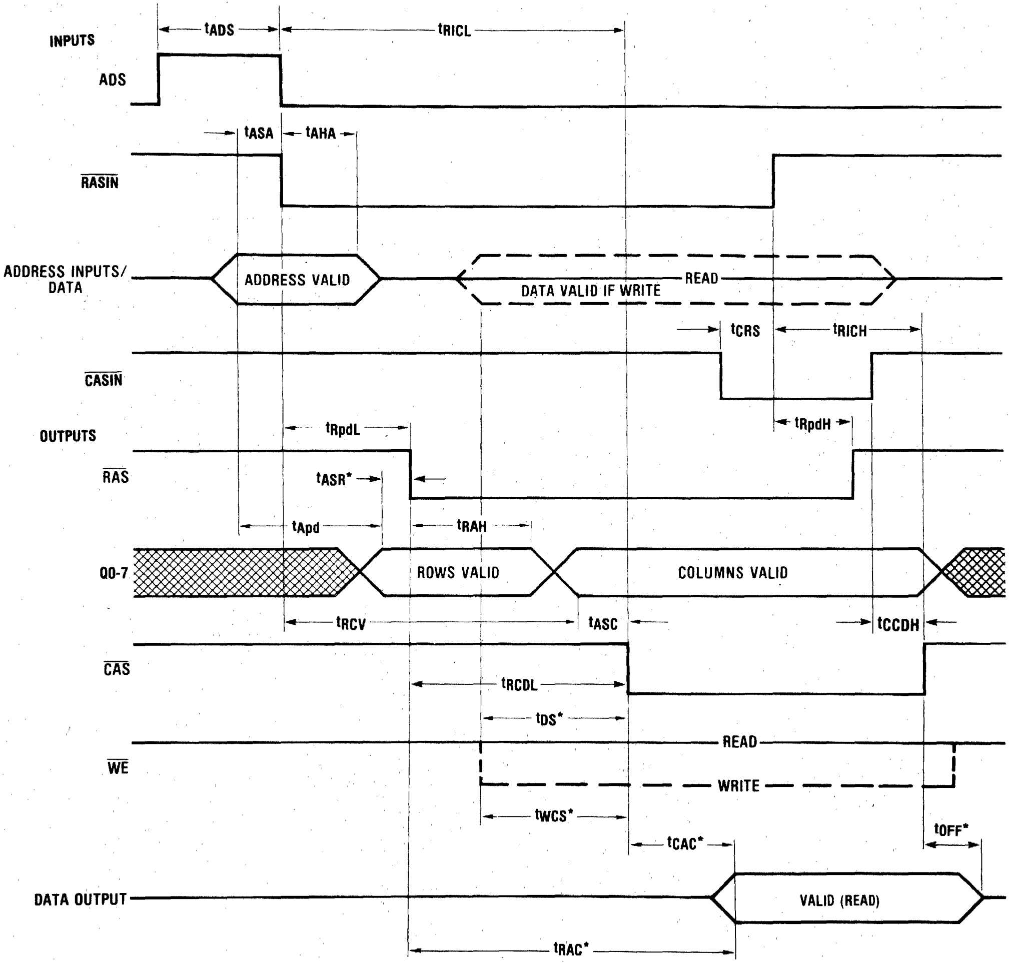
occurs. The only other control input required is $\overline{\text{WIN}}$. When a write cycle is required, $\overline{\text{WIN}}$ must go low at least 30ns before $\overline{\text{CAS}}$ is output low.

This gives a total typical delay from: input address valid to $\overline{\text{RASIN}}$ (15ns); to $\overline{\text{RAS}}$ (27ns); to rows held (50ns); to columns valid (25ns); to $\overline{\text{CAS}}$ (23ns) = 140ns (that is, 125ns from $\overline{\text{RASIN}}$). All of these typical figures are for heavy capacitive loading, of approximately 88 DRAMs. This mode is therefore extremely fast. The external timing is greatly simplified for the memory system designer: the only system signal required is $\overline{\text{RASIN}}$.

Mode 6 — Fast Automatic Access

The Fast Access mode is similar to Mode 5, but has a faster t_{RAH} of 20ns, minimum. It therefore can only be used with fast 16k or 64k DRAMs (which have a t_{RAH} of 10ns to 15ns) in applications requiring fast access times; $\overline{\text{RASIN}}$ to $\overline{\text{CAS}}$ is typically 105ns.

In this mode, the R/$\overline{\text{C}}$ pin is not used, but $\overline{\text{CASIN}}$ is used to allow an extended $\overline{\text{CAS}}$ after $\overline{\text{RAS}}$ has already terminated. Refer to Figure 5b. This is desirable with fast cycle-times where $\overline{\text{RAS}}$ has to be terminated as soon as possible before the next $\overline{\text{RAS}}$ begins (to meet the precharge time, or t_{RP}, requirements of the DRAM). $\overline{\text{CAS}}$ may then be held low by $\overline{\text{CASIN}}$ to extend the data output valid time from the DRAM to allow the system to read the data. $\overline{\text{CASIN}}$ subsequently going high ends $\overline{\text{CAS}}$. If this extended $\overline{\text{CAS}}$ is not required, $\overline{\text{CASIN}}$ should be set high in Mode 6.



*INDICATES DYNAMIC RAM PARAMETERS

FIGURE 5b. Mode 6 Timing, Extended $\overline{\text{CAS}}$

Mode 7 — Set End-of-Count

The End-of-Count can be externally selected in Mode 7, using ADS to strobe in the respective value of B1 and B0 (see Table 3). With B1 and B0 the same EOC is 127; with B1 = 0 and B0 = 1, EOC is 255; and with B1 = 1 and B0 = 0, EOC is 127. This selected value of EOC will be used until the next Mode 7 selection. At power-up the EOC is automatically set to 127 (B1 and B0 set to 11).

Table 3. Mode 7

Bank Select (Strobed by ADS)		End of Count Selected
B1	B0	
0	0	127
0	1	255
1	0	127
1	1	127

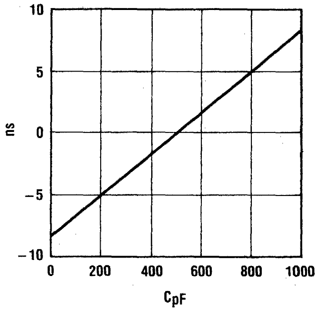


FIGURE 6. Change in Propagation Delay vs. Loading Capacitance Relative to a 500pF Load

Absolute Maximum Ratings (Note 1)

Supply Voltage, V_{CC}	7.0V
Storage Temperature Range	−65°C to +150°C
Input Voltage	5.5V
Output Current	150 mA
Lead Temperature (Soldering, 10 seconds)	300°C

Maximum Power Dissipation* at 25°C	
Cavity Package	3542mW
Molded Package	2833mW

*Derate cavity package 23.6mW/°C above 25°C; derate molded package 22.7mW/°C above 25°C.

Operating Conditions

	Min	Max	Units
V_{CC} Supply Voltage	4.75	5.25	V
T_A Ambient Temperature	0	+70	°C

Electrical Characteristics $V_{CC} = 5.0V \pm 5\%$, $0°C \leqslant T_A \leqslant 70°C$ unless otherwise noted (Notes 2, 6)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_C	Input Clamp Voltage	V_{CC} = Min., $I_C = -12$ mA		−0.8	−1.2	V
I_{IH1}	Input High Current for ADS, R/$\overline{C}$ only	$V_{IN} = 2.5V$		2.0	100	μA
I_{IH2}	Input High Current for All Other Inputs*	$V_{IN} = 2.5V$		1.0	50	μA
I_I RSI	Output Load Current for RF I/O	$V_{IN} = 0.5V$, Output High		−1.5	−2.5	mA
I_I CTL	Output Load Current for $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$	$V_{IN} = 0.5V$, Chip Deselect		−1.5	−2.5	mA
I_{IL1}	Input Low Current for ADS, R/$\overline{C}$ only	$V_{IN} = 0.5V$		−0.1	−1.0	mA
I_{IL2}	Input Low Current for All Other Inputs*	$V_{IN} = 0.5V$		−0.05	−0.5	mA
V_{IL}	Input Low Threshold				0.8	V
V_{IH}	Input High Threshold		2.0			V
V_{OL1}	Output Low Voltage*	$I_{OL} = 20$ mA		0.3	0.5	V
V_{OL2}	Output Low Voltage for RF I/O	$I_{OL} = 10$ mA		0.3	0.5	V
V_{OH1}	Output High Voltage*	$I_{OH} = -1$ mA	2.4	3.5		V
V_{OH2}	Output High Voltage for RF I/O	$I_{OH} = -100\mu A$	2.4	3.5		V
I_{1D}	Output High Drive Current*	$V_{OUT} = 0.8V$ (Note 3)		−200		mA
I_{0D}	Output Low Drive Current*	$V_{OUT} = 2.7V$ (Note 3)		200		mA
I_{OZ}	TRI-STATE Output Current (Address Outputs)	$0.4V \leqslant V_{OUT} \leqslant 2.7V$, $\overline{CS} = 2.0V$, Mode 4	−50	1.0	50	μA
I_{CC}	Supply Current	V_{CC} = Max.		210	285	mA

*Except RF I/O Output.

Switching Characteristics: DP8408/DP8408-3 $V_{CC} = 5.0V \pm 5\%$, $0°C \leqslant T_A \leqslant 70°C$ unless otherwise noted (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0-Q7, $C_L = 500pF$; $\overline{RAS0}$-$\overline{RAS3}$, $C_L = 150pF$; $\overline{WE}$, $C_L = 500pF$; $\overline{CAS}$, $C_L = 600pF$, unless otherwise noted. See Figure 7 for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Access Parameter	Conditions	8408			8408 − 3			Units
			Min	Typ	Max	Min	Typ	Max	
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 5a	95	125	160	95	125	185	ns
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 5a, 5b	80	105	140	80	105	160	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 5a	40	48	60	40	48	70	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 5a, 5b	50	63	80	50	63	95	ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 5a		98	125		98	145	ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 5a, 5b		78	105		78	120	ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 5a		27	40		27	40	ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figure 5a		40	65		40	65	ns
t_{CCDH}	$\overline{CASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figure 5b	40	54	70	40	54	80	ns
t_{RAH}	Row Address Hold Time (Mode 5)	Figure 5a	30			30			ns
t_{RAH}	Row Address Hold Time (Mode 6)	Figures 5a, 5b	20			20			ns
t_{ASC}	Column Address Setup Time (Mode 5)	Figure 5a	8			8			ns
t_{ASC}	Column Address Setup Time (Mode 6)	Figures 5a, 5b	6			6			ns
t_{RCV}	$\overline{RASIN}$ to Column Address Valid (Mode 5)	Figure 5a		90	120		90	140	ns

Switching Characteristics (Cont'd)

Symbol	Access Parameter	Conditions	8408			8408 – 3			Units
			Min	Typ	Max	Min	Typ	Max	
t_{RCV}	$\overline{\text{RASIN}}$ to Column Address Valid (Mode 6)	Figures 5a, 5b		75	105		75	120	ns
t_{RPDL}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay	Figures 4a, 4b, 5a, 5b	20	27	35	20	27	40	ns
t_{RPDH}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay	Figures 4a, 4b, 5a, 5b	15	23	32	15	23	37	ns
t_{APDL}	Address Input to Output Low Delay	Figures 4a, 4b, 5a, 5b		25	40		25	46	ns
t_{APDH}	Address Input to Output High Delay	Figures 4a, 4b, 5a, 5b		25	40		25	46	ns
t_{SPDL}	Address Strobe to Address Output Low	Figures 4a, 4b		40	60		40	70	ns
t_{SPDH}	Address Strobe to Address Output High	Figures 4a, 4b		40	60		40	70	ns
t_{ASA}	Address Setup Time to ADS	Figures 4a, 4b, 5a, 5b	15			15			ns
t_{AHA}	Address Hold Time from ADS	Figures 4a, 4b, 5a, 5b	15			15			ns
t_{ADS}	Address Strobe Pulse Width	Figures 4a, 4b, 5a, 5b	30			30			ns
t_{WPDL}	$\overline{\text{WIN}}$ to $\overline{\text{WE}}$ Output Delay	Figure 4b	15	25	30	15	25	35	ns
t_{WPDH}	$\overline{\text{WIN}}$ to $\overline{\text{WE}}$ Output Delay	Figure 4b	15	30	60	15	30	70	ns
t_{CRS}	$\overline{\text{CASIN}}$ Setup Time to $\overline{\text{RASIN}}$ High (Mode 6)	Figure 5b	35			35			ns
t_{CPDL}	$\overline{\text{CASIN}}$ to $\overline{\text{CAS}}$ Delay (R/$\overline{\text{C}}$ low in Mode 4)	Figure 4b	32	41	68	32	41	77	ns
t_{CPDH}	$\overline{\text{CASIN}}$ to $\overline{\text{CAS}}$ Delay	Figure 4b	25	39	50	25	39	60	ns
t_{RCC}	Column Select to Column Address Valid	Figure 4a		40	58		40	67	ns
t_{RCR}	Row Select to Row Address Valid	Figures 4a, 4b		40	58		40	67	ns
t_{RHA}	Row Address Held from Column Select	Figure 4a	10			10			ns
t_{CCAS}	R/$\overline{\text{C}}$ Low to $\overline{\text{CAS}}$ Low (Mode 4 Auto $\overline{\text{CAS}}$)	Figure 7a		65	90				ns
t_{DIF1}	Maximum ($t_{RPDL} - t_{RHA}$)	See Mode 4 description			13			18	ns
t_{DIF2}	Maximum ($t_{RCC} - t_{CPDL}$)	See Mode 4 description			13			18	ns
	Refresh Parameter								
t_{RC}	Refresh Cycle Period	Figure 2	100			100			ns
$t_{RASINL,H}$	Pulse Width of $\overline{\text{RASIN}}$ during Refresh	Figure 2	50			50			ns
t_{RFPDL}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay during Refresh	Figure 2	35	50	70	35	50	80	ns
t_{RFPDH}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay during Refresh	Figure 2	30	40	55	30	40	65	ns
t_{RFLCT}	$\overline{\text{RFSH}}$ Low to Counter Address Valid	$\overline{\text{CS}}$ = X, Figure 2		47	60		47	70	ns
t_{RFHRV}	$\overline{\text{RFSH}}$ High to Row Address Valid	Figure 2		45	60		45	70	ns
t_{ROHNC}	$\overline{\text{RAS}}$ High to New Count Valid	Figure 2		30	55		30	55	ns
t_{RLEOC}	$\overline{\text{RASIN}}$ Low to End-of-Count Low	$C_L = 50pF$, Figure 2			80			80	ns
t_{RHEOC}	$\overline{\text{RASIN}}$ High to End-of-Count High	$C_L = 50pF$, Figure 2			80			80	ns
t_{RST}	Counter Reset Pulse Width	Figure 2	70			70			ns
t_{CTL}	RF I/O Low to Counter Outputs All Low	Figure 2			100			100	ns
	TRI-STATE Parameter								
t_{ZH}	$\overline{\text{CS}}$ Low to Address Output High from Hi-Z	Figure 8 R1 = 3.5k, R2 = 1.5k		35	60		35	60	ns
t_{HZ}	$\overline{\text{CS}}$ High to Address Output Hi-Z from High	$C_L = 15pF$, Figure 8 R2 = 1k, S1 open		20	40		20	40	ns
t_{ZL}	$\overline{\text{CS}}$ Low to Address Output Low from Hi-Z	Figure 8 R1 = 3.5k, R2 = 1.5k		35	60		35	60	ns
t_{LZ}	$\overline{\text{CS}}$ High to Address Output Hi-Z from Low	$C_L = 15pF$, Figure 8, R1 = 1k, S2 open		25	50		25	50	ns

Switching Characteristics (Cont'd)

Symbol	TRI-STATE Parameter	Conditions	8408			8408 – 3			Units
			Min	Typ	Max	Min	Typ	Max	
t_{HZH}	$\overline{CS}$ Low to Control Output High from Hi-Z High	Figure 8 R2 = 750Ω, S1 open		50	80		50	80	ns
t_{HHZ}	$\overline{CS}$ High to Control Output Hi-Z High from High	C_L = 15 pF, Figure 8 R2 = 750Ω, S1 open		40	75		40	75	ns
t_{HZL}	$\overline{CS}$ Low to Control Output Low from Hi-Z High	Figure 8 S1, S2 open		45	75		45	75	ns
t_{LHZ}	$\overline{CS}$ High to Control Output Hi-Z High from Low	C_L = 15 pF, Figure 8, R2 = 750Ω, S1 open		50	80		50	80	ns

Switching Characteristics: DP8408-2

V_{CC} = 5.0V ± 5%, 0°C ⩽ T_A ⩽ 70°C unless otherwise noted (Notes 2, 4, 5, 7). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0-Q7, C_L = 500 pF; $\overline{RAS0}$-$\overline{RAS3}$, C_L = 150 pF; $\overline{WE}$, C_L = 500 pF; $\overline{CAS}$, C_L = 600 pF, unless otherwise noted. See Figure 7 for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Access Parameter	Conditions	8408 – 2						Units
			Min	Typ	Max	Min	Typ	Max	
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 5a	75	100	130				ns
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 5a, 5b	65	90	115				ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 5a	40	48	60				ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 5a, 8b	50	63	80				ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 5a		75	100				ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 5a, 5b		65	85				ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 5a		27	40				ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figure 5a		40	65				ns
t_{CCDH}	$\overline{CASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figure 5b	40	54	70				ns
t_{RAH}	Row Address Hold Time (Mode 5) (Note 7)	Figure 5a	20						ns
t_{RAH}	Row Address Hold Time (Mode 6) (Note 7)	Figures 5a, 5b	12						ns
t_{ASC}	Column Address Setup Time (Mode 5)	Figure 5a	3						ns
t_{ASC}	Column Address Setup Time (Mode 6)	Figures 5a, 8b	3						ns
t_{RCV}	$\overline{RASIN}$ to Column Address Valid (Mode 5)	Figure 5a		80	105				ns
t_{RCV}	$\overline{RASIN}$ to Column Address Valid (Mode 6)	Figures 5a, 5b		70	90				ns
t_{RPDL}	$\overline{RASIN}$ to $\overline{RAS}$ Delay	Figures 4a, 4b, 5a, 5b	20	27	35				ns
t_{RPDH}	$\overline{RASIN}$ to $\overline{RAS}$ Delay	Figures 4a, 4b, 5a, 5b	15	23	32				ns
t_{APDL}	Address Input to Output Low Delay	Figures 4a, 4b, 5a, 5b		25	40				ns
t_{APDH}	Address Input to Output High Delay	Figures 4a, 4b, 5a, 5b		25	40				ns
t_{SPDL}	Address Strobe to Address Output Low	Figures 4a, 4b		40	60				ns
t_{SPDH}	Address Strobe to Address Output High	Figures 4a, 4b		40	60				ns
t_{ASA}	Address Set-up Time to ADS	Figures 4a, 4b, 5a, 5b	15						ns
t_{AHA}	Address Hold Time from ADS	Figures 4a, 4b, 5a, 5b	15						ns
t_{ADS}	Address Strobe Pulse Width	Figures 4a, 4b, 5a, 5b	30						ns
t_{WPDL}	$\overline{WIN}$ to $\overline{WE}$ Output Delay	Figure 4b	15	25	30				ns
t_{WPDH}	$\overline{WIN}$ to $\overline{WE}$ Output Delay	Figure 4b	15	30	60				ns
t_{CRS}	$\overline{CASIN}$ Set-up Time to $\overline{RASIN}$ High (Mode 6)	Figure 5b	35						ns
t_{CPDL}	$\overline{CASIN}$ to $\overline{CAS}$ Delay (R/$\overline{C}$ low in Mode 4)	Figure 4b	32	41	58				ns

Switching Characteristics (Cont'd)

Symbol	Access Parameter	Conditions	8408 – 2						Units
			Min	Typ	Max	Min	Typ	Max	
t_{CPDH}	$\overline{CASIN}$ to $\overline{CAS}$ Delay (R/$\overline{C}$ low in Mode 4)	Figure 4b	25	39	50				ns
t_{RCC}	Column Select to Column Address Valid	Figure 4a		40	58				ns
t_{RCR}	Row Select to Row Address Valid	Figures 4a, 4b		40	58				ns
t_{RHA}	Row Address Held from Column Select	Figure 4a	10						ns
t_{CCAS}	R/$\overline{C}$ Low to $\overline{CAS}$ Low (Mode 4 Auto $\overline{CAS}$)	Figure 7a		55	75				ns
t_{DIF1}	Maximum ($t_{RPDL} - t_{RHA}$)	See Mode 4 description			13				ns
t_{DIF2}	Maximum ($t_{RCC} - t_{CPDL}$)	See Mode 4 description			13				ns
	Refresh Parameter								
t_{RC}	Refresh Cycle Period	Figure 2	100						ns
$t_{RASINL,H}$	Pulse Width of $\overline{RASIN}$ during Refresh	Figure 2	50						ns
t_{RFPDL}	$\overline{RASIN}$ to $\overline{RAS}$ Delay during Refresh	Figure 2	35	50	70				ns
t_{RFPDH}	$\overline{RASIN}$ to $\overline{RAS}$ Delay during Refresh	Figure 2	30	40	55				ns
t_{RFLCT}	$\overline{RFSH}$ Low to Counter Address Valid	$\overline{CS}$ = X, Figure 2		47	60				ns
t_{RFHRV}	$\overline{RFSH}$ High to Row Address Valid	Figure 2		45	60				ns
t_{ROHNC}	$\overline{RAS}$ High to New Count Valid	Figure 2		30	55				ns
t_{RLEOC}	$\overline{RASIN}$ Low to End-of-Count Low	C_L = 50pF, Figure 2			80				ns
t_{RHEOC}	$\overline{RASIN}$ High to End-of-Count High	C_L = 50pF, Figure 2			80				ns
t_{RST}	Counter Reset Pulse Width	Figure 2	70						ns
t_{CTL}	RF I/O Low to Counter Outputs All Low	Figure 2			100				ns
	TRI-STATE Parameter								
t_{ZH}	$\overline{CS}$ Low to Address Output High from Hi-Z	Figures 9, 12 R1 = 3.5k, R2 = 1.5k		35	60				ns
t_{HZ}	$\overline{CS}$ High to Address Output Hi-Z from High	C_L = 15pF, Figures 9, 12 R2 = 1k, S1 open		20	40				ns
t_{ZL}	$\overline{CS}$ Low to Address Output Low from Hi-Z	Figures 9, 12 R1 = 3.5k, R2 = 1.5k		35	60				ns
t_{LZ}	$\overline{CS}$ High to Address Output Hi-Z from Low	C_L = 15pF, Figures 9, 12 R1 = 1k, S2 open		25	50				ns
t_{HZH}	$\overline{CS}$ Low to Control Output High from Hi-Z High	Figures 9, 12 R2 = 750Ω, S1 open		50	80				ns
t_{HHZ}	$\overline{CS}$ High to Control Output Hi-Z High from High	C_L = 15pF, Figures 9, 12 R2 = 750Ω, S1 open		40	75				ns
t_{HZL}	$\overline{CS}$ Low to Control Output Low from Hi-Z High	Figure 12, S1, S2 open		45	75				ns
t_{LHZ}	$\overline{CS}$ High to Control Output Hi-Z High from Low	C_L = 15pF, Figure 12, R2 = 750Ω, S1 open		50	80				ns

Input Capacitance $T_A = 25°C$ (Notes 2, 6)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{IN}	Input Capacitance ADS, R/$\overline{C}$			8		pF
C_{IN}	Input Capacitance All Other Inputs			5		pF

Note 1: "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All typical values are for $T_A = 25°C$ and $V_{CC} = 5.0V$.

Note 3: This test is provided as a monitor of Driver output source and sink current capability. Caution should be exercised in testing this parameter. In testing these parameters, a 15Ω resistor should be placed in series with each output under test. One output should be tested at a time and test time should not exceed 1 second.

Note 4: Input pulse 0V to 3.0V, $t_R = t_F = 2.5$ ns, f = 2.5 MHz, $t_{PW} = 200$ ns. Input reference point on AC measurements is 1.5V. Output reference points are 2.7V for High and 0.8V for Low.

Note 5: The load capacitance on RF I/O should not exceed 50 pF.

Note 6: Applies to all DP8408 versions unless otherwise specified.

Note 7: The DP8408-2 device can only be used with memory devices that meet the t_{RAH} specification indicated.

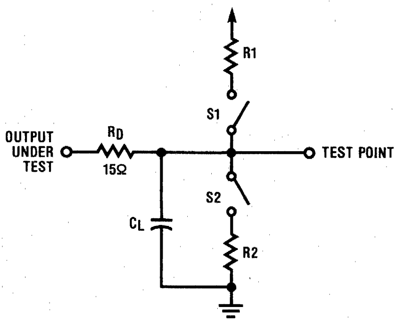


FIGURE 7. Output Load Circuit

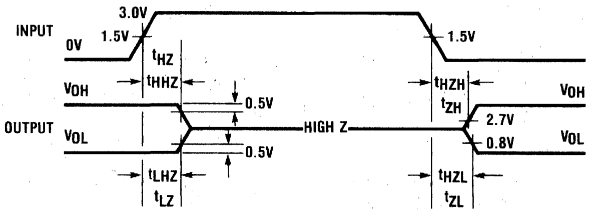


FIGURE 8. Waveform

Applications

If external control is preferred, the DP8408 may be used in Modes 0 or 4, as in Figure 3.

If basic auto access and refresh are required, then in cases where the user requires the minimum of external complexity, Modes 0 and 5 are ideal, as shown in Figure 9a. The DP843X2 is used to provide proper arbitration between memory access and refresh. This chip supplies all the necessary control signals to the processor as well as the DP8408. Furthermore, two separate $\overline{\text{CAS}}$ outputs are also included for systems using byte-writing. The refresh clock RFCK may be divided down from either RGCK using an IC counter such as the DM74LS393 or better still, the DP84300 Programmable Refresh Timer. The DP84300 can provide RFCK periods ranging from 15.4μs to 15.6μs based on the input clock of 2 to 10 MHz. Figure 9b shows the general timing diagram for interfacing the DP8408 to different microprocessors using the interface controller DP843X2.

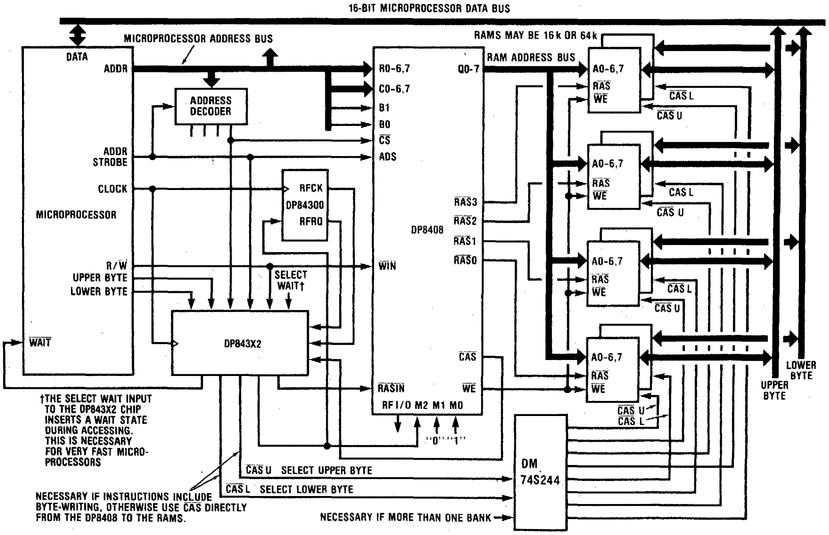


FIGURE 9a. Connecting the DP8408 Between the 16-Bit Microprocessor and Memory

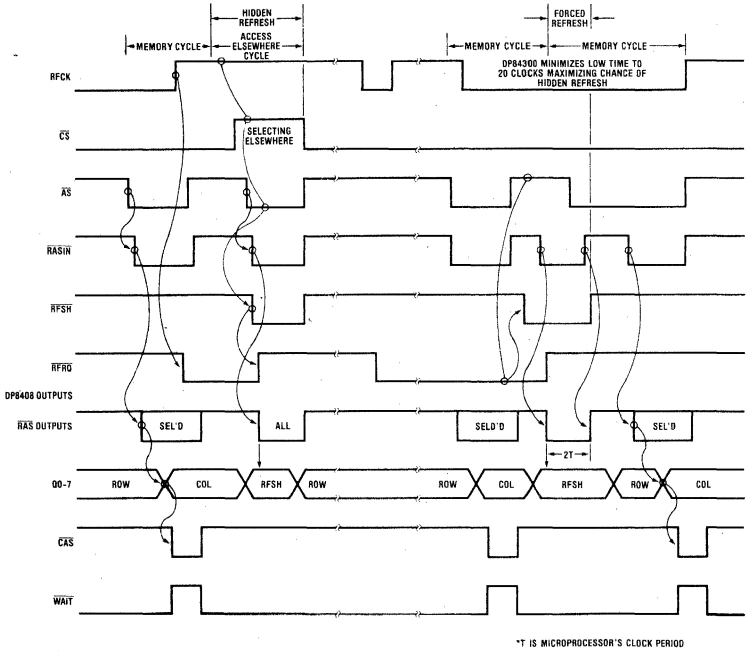
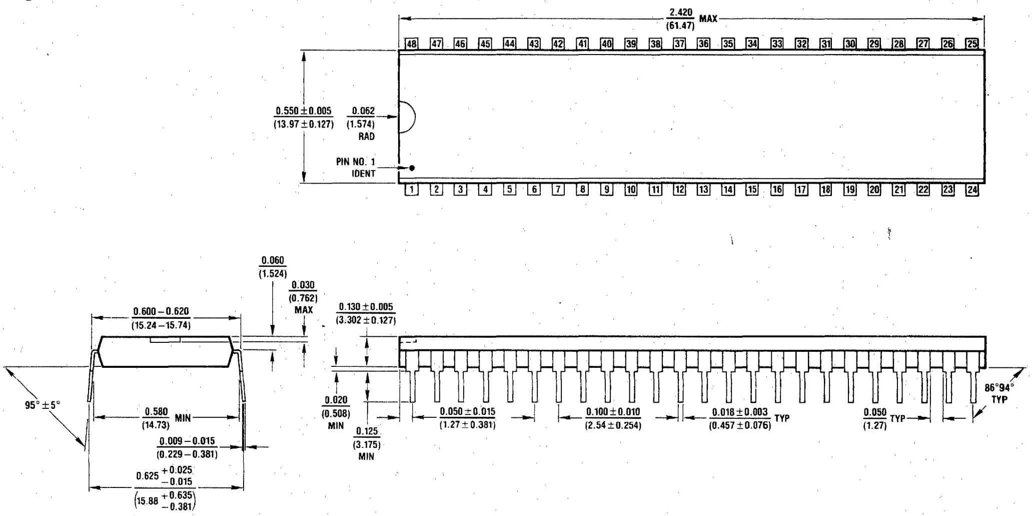
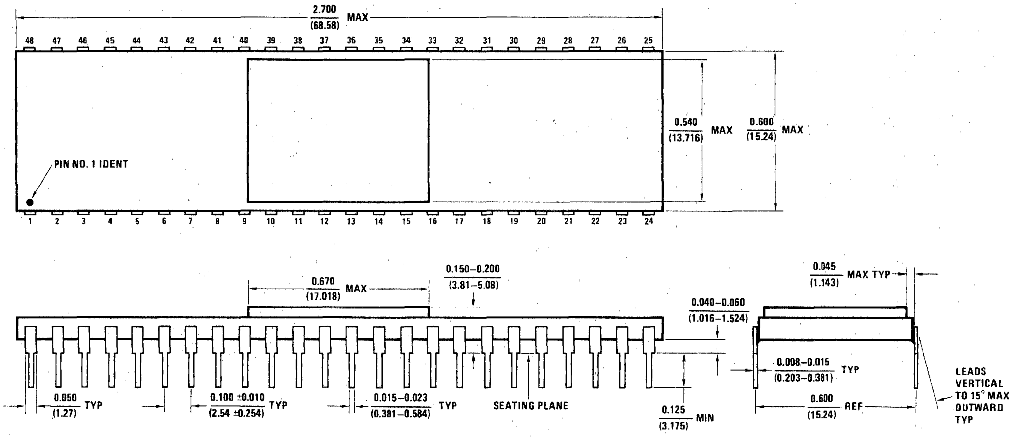


FIGURE 9b. DP8408 Auto Refresh

Physical Dimensions (inches, millimeters)



48-Lead Molded Dual-In-Line Package (N)
Order Number DP8408N
or Order Number DP8408N-3
NS Package N48A



48-Lead Molded Dual-In-Line Package (D)
Order Number DP8408D
NS Package D48A

National Semiconductor

DP8409 Multi-Mode Dynamic RAM Controller/Driver

General Description

Dynamic memory system designs, which formerly required several support chips to drive the memory array, can now be implemented with a single IC. . . the DP8409 Multi-Mode Dynamic RAM Controller/Driver. The DP8409 is capable of driving all 16k and 64k Dynamic RAMs (DRAMs) as well as 256k DRAMs. Since the DP8409 is a one-chip solution (including capacitive-load drivers), it minimizes propagation delay skews, the major performance disadvantage of multiple-chip memory drive and control.

The DP8409's 8 modes of operation offer a wide selection of DRAM control capabilities. Memory access may be controlled externally or on-chip automatically; an on-chip refresh counter makes refreshing (either externally or automatically controlled) less complicated; and automatic memory initialization is both simple and fast.

The DP8409 is a 48-pin DRAM Controller/Driver with 9 multiplexed address outputs and 6 control signals. It consists of two 9-bit address latches, a 9-bit refresh counter, and control logic. All output drivers are capable of driving 500pF loads with propagation delays of 25ns. The DP8409 timing parameters are specified driving the typical load capacitance of 88 DRAMs, including trace capacitance.

The DP8409 has 3 mode-control pins: M2, M1, and M0, where M2 is in general $\overline{\text{REFRESH}}$. These 3 pins select 8 modes of operation. Inputs B1 and B0 in the memory access modes (M2 = 1), are select inputs which select one of four $\overline{\text{RAS}}$ outputs. During normal access, the 9 address outputs can be selected from the Row Address Latch or the Column Address Latch. During refresh, the 9-bit on-chip refresh counter is enabled onto the address bus and in this mode all $\overline{\text{RAS}}$ outputs are selected, while $\overline{\text{CAS}}$ is inhibited.

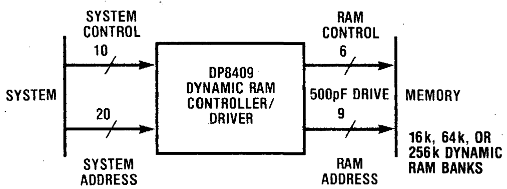
The DP8409 can drive up to 4 banks of DRAMs, with each bank comprised of 16k's, 64k's, or 256k's. Control signal outputs $\overline{\text{RAS}}$, $\overline{\text{CAS}}$, and $\overline{\text{WE}}$ are provided with the same drive capability. Each $\overline{\text{RAS}}$ output drives one bank of DRAMs so that the four $\overline{\text{RAS}}$ outputs are used to select the banks, while $\overline{\text{CAS}}$, $\overline{\text{WE}}$, and the multiplexed addresses can be connected to all of the banks of DRAMs. This leaves the non-selected banks in the standby mode (less than one tenth of the operating power) with the data outputs in TRI-STATE®. Only the bank with its associated $\overline{\text{RAS}}$ low will be written to or read from.

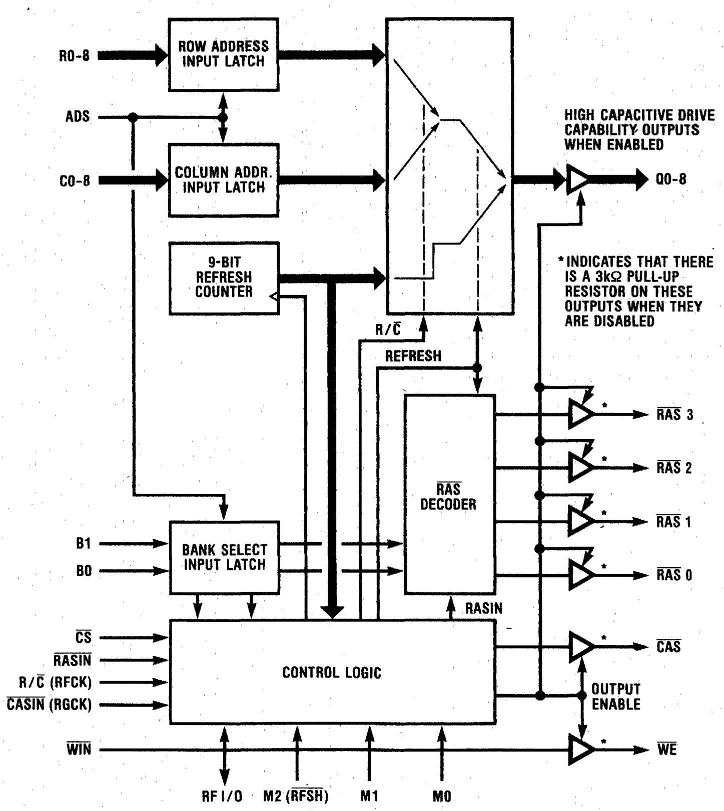
Operational Features

- All DRAM drive functions on one chip — minimizes skew on outputs, maximizes AC performance
- On-chip capacitive-load drives (specified to drive up to 88 DRAMs)
- Drives directly all 16k, 64k, and 256k DRAMs
- Capable of addressing 64k, 256k, or 1M words
- Propagation delays of 25ns typical at 500pF load
- $\overline{\text{CAS}}$ goes low automatically after column addresses are valid if desired
- Auto Access mode provides $\overline{\text{RAS}}$, row to column select, then $\overline{\text{CAS}}$ automatically and fast
- $\overline{\text{WE}}$ follows $\overline{\text{WIN}}$ unconditionally—offering READ, WRITE or READ-MODIFY-WRITE cycles
- On-chip 9-bit refresh counter with selectable End-of-Count (127, 255, or 511)
- End-of-Count indicated by RF I/O pin going low at 127, 255, or 511
- Low input on RF I/O resets 9-bit refresh counter
- $\overline{\text{CAS}}$ inhibited during refresh cycle
- Fall-through latches on address inputs controlled by ADS
- TRI-STATE outputs allow multi-controller addressing of memory
- Control output signals go high-impedance logic "1" when disabled for memory sharing
- Power-up: counter reset, control signals high, address outputs TRI-STATE, and End-of-Count set to 127

Mode Features

- 8 modes of operation: 3 access, 3 refresh, and 2 set-up
- 2 externally controlled modes: 1 access and 1 refresh (Modes 0, 4)
- 2 auto-access modes $\overline{\text{RAS}} \rightarrow \text{R}/\overline{\text{C}} \rightarrow \overline{\text{CAS}}$ automatic, with $t_{RAH} = 20$ or 30ns minimum (Modes 5, 6)
- Auto-access mode allows Hidden Refreshing (Mode 5)
- Forced Refresh requested on RF I/O if no Hidden Refresh (Mode 5)
- Forced Refresh performed after system acknowledge of request (Mode 1)
- Automatic Burst Refresh mode stops at End-of-Count of 127, 255, or 511 (Mode 2)
- 2 All-$\overline{\text{RAS}}$ Access modes externally or automatically controlled for memory initialization (Modes 3a, 3b)
- Automatic All-$\overline{\text{RAS}}$ mode with external 8-bit counter frees system for other set-up routines (Mode 3a)
- End-of-Count value of Refresh Counter set by B1 and B0 (Mode 7)

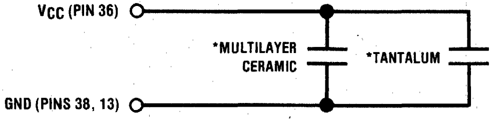




DP8409 Functional Block Diagram

Pin Definitions

V_{CC}, GND, GND — $V_{CC} = 5V \pm 5\%$. The three supply pins have been assigned to the center of the package to reduce voltage drops, both DC and AC. There are also two ground pins to reduce the low level noise. The second ground pin is located two pins from V_{CC}, so that decoupling capacitors can be inserted directly next to these pins. It is important to adequately decouple this device, due to the high switching currents that will occur when all 9 address bits change in the same direction simultaneously. A recommended solution would be a $1\mu F$ multilayer ceramic capacitor in parallel with a low-voltage tantalum capacitor, both connected as close as possible to pins 36 and 38 to reduce lead inductance. See Figure below.



*Capacitor values should be chosen depending on the particular application.

R0-R8: Row Address Inputs.

C0-C8: Column Address Inputs.

Q0-Q8: Multiplexed Address Outputs — Selected from the Row Address Input Latch, the Column Address Input Latch, or the Refresh Counter.*

$\overline{\text{RASIN}}$: Row Address Strobe Input — Enables selected $\overline{\text{RAS}}_n$ output when M2 ($\overline{\text{RFSH}}$) is high, or all $\overline{\text{RAS}}_n$ outputs when $\overline{\text{RFSH}}$ is low.

R/$\overline{\text{C}}$ (RFCK) — In Auto-Refresh Mode this pin is the external Refresh Clock Input: one refresh cycle has to be performed each clock period. In all other modes it is Row/Column Select Input: selects either the row or column address input latch onto the output bus.

$\overline{\text{CASIN}}$ (RGCK) — In Auto-Refresh Mode, Auto Burst Mode, and All-$\overline{\text{RAS}}$ Auto-Write Mode, this pin is the $\overline{\text{RAS}}$ Generator Clock input. In all other modes it is $\overline{\text{CASIN}}$ (Column Address Strobe Input), which inhibits $\overline{\text{CAS}}$ output when high in Modes 4 and 3b. In Mode 6 it can be used to prolong $\overline{\text{CAS}}$ output.

ADS: Address (Latch) Strobe Input — Row Address, Column Address, and Bank Select Latches are fall-through with ADS high; Latches on high-to-low transition.

$\overline{\text{CS}}$: Chip Select Input — TRI-STATE's the Address Outputs and puts the control signal into a high-impedance logic "1" state when high (unless refreshing in one of the Refresh Modes). Enables all outputs when low.

M0, M1, M2: Mode Control Inputs — These 3 control pins determine the 8 major modes of operation of the DP8409 as depicted in Table 1.

Table 1. DP8409 Mode Select Options

Mode	($\overline{RFSH}$) M2	M1	M0	Mode of Operation	Conditions
0	0	0	0	Externally Controlled Refresh	RF I/O = $\overline{EOC}$
1	0	0	1	Auto Refresh — forced	RF I/O = Refresh Request ($\overline{RFRQ}$)
2	0	1	0	Internal Auto Burst Refresh	RF I/O = $\overline{EOC}$
3a	0	1	1	All $\overline{RAS}$ Auto Write	RF I/O = $\overline{EOC}$; All $\overline{RAS}$ Active
3b	0	1	1	Externally Controlled All $\overline{RAS}$ Access	All $\overline{RAS}$ Active
4	1	0	0	Externally Controlled Access	Active $\overline{RAS}$ defined by Table 2
5	1	0	1	Auto Access, Slow t_{RAH}, Hidden Refresh	Active $\overline{RAS}$ defined by Table 2
6	1	1	0	Auto Access, Fast t_{RAH}	Active $\overline{RAS}$ defined by Table 2
7	1	1	1	Set End of Count	See Table 3 for Mode 7

RF I/O — The I/O pin functions as a Reset Counter Input when set low from an external open-collector gate, or as a flag output. The flag goes active-low in Modes 0 and 2 when the End-of-Count output is at 127, 255, or 511 (see Table 3). in Auto-Refresh Mode it is the Refresh Request output.

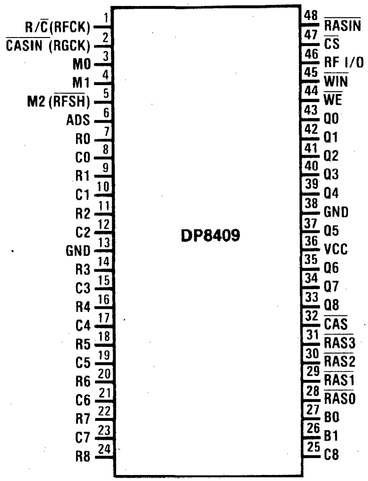
$\overline{WIN}$: Write Enable Input.

$\overline{WE}$: Write Enable Output — Buffered output from $\overline{WIN}$.*

$\overline{CAS}$: Column Address Strobe Output — In Modes 3a, 5, and 6, $\overline{CAS}$ transitions low following valid column address. In Modes 3b and 4, it goes low after R/$\overline{C}$ goes low, or follows $\overline{CASIN}$ going low if R/$\overline{C}$ is already low. $\overline{CAS}$ is high during refresh.*

$\overline{RAS}$ 0-3: Row Address Strobe Outputs — Selects a memory bank decoded from B1 and B0 (see Table 2), if $\overline{RFSH}$ is high. If $\overline{RFSH}$ is low, all banks are selected.*

B0, B1: Bank Select Inputs — Strobed by ADS. Decoded to enable one of the $\overline{RAS}$ outputs when $\overline{RASIN}$ goes low. Also used to define End-of-Count in Mode 7 (Table 3).



Pin Configuration

Conditions for all Modes

Input Addressing

The address block consists of a row-address latch, a column-address latch, and a resettable refresh counter. The address latches are fall-through when ADS is high and latch when ADS goes low. If the address bus contains valid addresses until after the valid address time, ADS can be permanently high. Otherwise ADS must go low while the addresses are still valid.

In normal memory access operation, $\overline{RASIN}$ and R/$\overline{C}$ are initially high. When the address inputs are enabled into the address latches, the row addresses appear on the Q outputs. The address strobe also inputs the bank-select address, (B0 and B1). If $\overline{CS}$ is low, all outputs are enabled. When $\overline{CS}$ is transitioned high, the address outputs go TRI-STATE and the control outputs first go high through a low impedance, and then are held by an on-chip high impedance. This allows output paralleling with other DP8409s for multi-addressing. All outputs go active about 50ns after the chip is selected again. If $\overline{CS}$ is high, and a refresh cycle begins, all the outputs become active until the end of the refresh cycle.

Drive Capability

The DP8409 has timing parameters that are specified with up to 600pF loads. In a typical memory system this is equivalent to about 88, 5V-only DRAMs, with trace lengths kept to a minimum. Therefore, the chip can drive four banks each of 16 or 22 bits, or two banks of 32 or 39 bits, or one bank of 64 or 72 bits.

Less loading will slightly reduce the timing parameters, and more loading will increase the timing parameters, according to the graph of Figure 10. The AC performance parameters are specified with the typical load capacitance of 88 DRAMs. This graph can be used to extrapolate the variations expected with other loading.

Because of distributed trace capacitance and inductance and DRAM input capacitance, current spikes can be created, causing overshoots and undershoots at the DRAM inputs that can change the contents of the DRAMs or even destroy them. To remove these spikes, a damping resistor (low inductance, carbon) can be inserted between the DP8409 driver outputs and the DRAMs, as close as possible to the DP8409. The values of the damping resistors may differ between the different control outputs; $\overline{RAS}$'s, $\overline{CAS}$, Q's, and $\overline{WE}$. The damping resistors should be determined by the first prototypes (not wire-wrapped due to the larger distributed capacitance and inductance). The best values for the damping resistors are the critical values giving a critically damped transition on the control outputs. Typical values for the damping resistors will be between 15Ω and 100Ω, the lower the loading the higher the value. (For more information, see AN-305 "Precautions to Take When Driving Memories.")

DP8409 Driving any 16k or 64k DRAMs

The DP8409 can drive any 16k or 64k DRAMs. All 16k DRAMs are basically the same configuration, including the newer 5V-only version. Hence, in most applications, different manufacturers' DRAMs are interchangeable (for the same supply-rail chips), and the DP8409 can drive all 16k DRAMs (see Figure 1a).

There are three basic configurations for the 5V-only 64k DRAMs: a 128-row by 512-column array with an on-RAM refresh counter, a 128-row by 512-column array with no on-RAM refresh counter, and a 256-row by 256-column array with no on-RAM refresh counter. The DP8409 can drive all three configurations, and at the same time allows them all to be interchangeable (as shown in Figures 1b and 1c), providing maximum flexibility in the choice of DRAMs. Since the 9-bit on-chip refresh counter can be used as a 7-bit refresh counter for the 128-row configuration, or as an 8-bit refresh counter for the 256-row configuration, the on-RAM refresh counter (if present) is never used. As long as 128 rows are refreshed every 2ms (i.e. 256 rows in 4ms) all DRAM types are correctly refreshed.

DP8409 Interface Between System & DRAM Banks

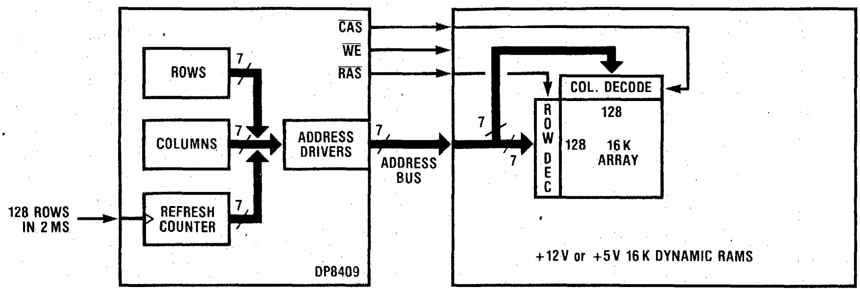
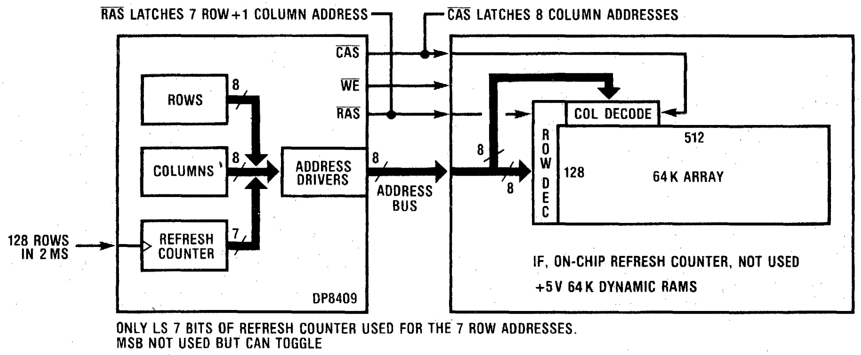


FIGURE 1a. DP8409 with any 16k DRAMS



RAS LATCHES 7 ROW+1 COLUMN ADDRESS
CAS LATCHES 8 COLUMN ADDRESSES
CAS
WE
RAS
ROWS
COLUMNS
REFRESH COUNTER
ADDRESS DRIVERS
8
7
ADDRESS BUS
128 ROWS IN 2 MS
DP8409
COL DECODE
512
ROW DEC
128
64K ARRAY
IF, ON-CHIP REFRESH COUNTER, NOT USED
+5V 64K DYNAMIC RAMS
ONLY LS 7 BITS OF REFRESH COUNTER USED FOR THE 7 ROW ADDRESSES.
MSB NOT USED BUT CAN TOGGLE

FIGURE 1b. DP8409 with 128 Row × 512 Column 64k DRAM

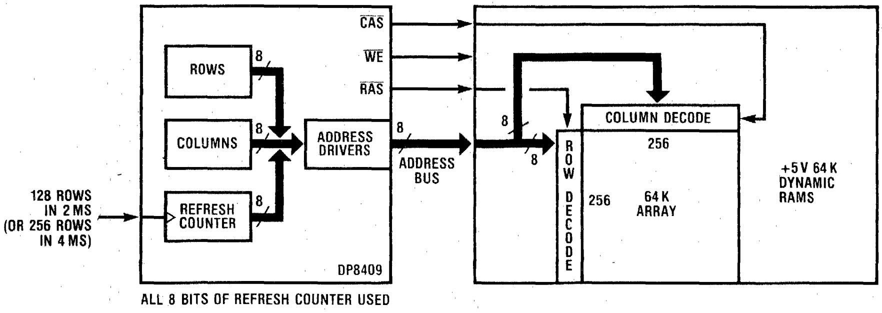


FIGURE 1c. DP8409 with 256 × 256 Column 64k DRAM

When the DP8409 is in a refresh mode, the RF I/O pin indicates that the on-chip refresh counter has reached its end-of-count. This end-of-count is selectable as 127, 255 or 512 to accommodate 16k, 64k, or 256k DRAMs. Although the end-of-count may be chosen to be any of these, the counter always counts to 511 before rolling over to zero.

Read, Write, and Read-Modify-Write Cycles

The output signal, WE, determines what type of memory access cycle the memory will perform. If WE is kept high while CAS goes low, a read cycle occurs. If WE goes low before CAS goes low, a write cycle occurs and data at DI (DRAM input data) is written into the DRAM as CAS goes low. If WE goes low later than t_{CWD} after CAS goes low, first a read occurs and DO (DRAM output data) becomes valid; then data DI is written into the same address in the DRAM when WE goes low. In this read-modify-write case, DI and DO cannot be linked together. The type of cycle is therefore controlled by WE, which follows WIN.

Power-Up Initialize

When V_{CC} is first applied to the DP8409, an initialize pulse clears the refresh counter, the internal control flip-flops, and sets the End-of-Count of the refresh counter to 127 (which may be changed via Mode 7). As V_{CC} increases to about 2.3 volts, it holds the output control signals at a level of one Schottky diode-drop below V_{CC}, and the output address to TRI-STATE. As V_{CC} increases above 2.3 volts, control of these outputs is granted to the system.

DP8409 Functional Mode Descriptions

Note: All delay parameters stated in text refer to the DP8409. Substitute the respective delay numbers for the DP8409-2 or DP8409-3 when using these devices.

Mode 0 — Externally Controlled Refresh

Figure 2 is the Externally Controlled Refresh Timing. In this mode, the input address latches are disabled from the address outputs and the refresh counter is enabled. When RAS occurs, the enabled row in the DRAM is refreshed. In the Externally Controlled Refresh mode, all RAS outputs are enabled following RASIN, and CAS is inhibited. This refreshes the same row in all four banks. The refresh counter increments when either RASIN or RFSH goes low-to-high after a refresh. RF I/O goes low when the count is 127, 255, or 511, as set by End-of-Count (see Table 3), with RASIN and RFSH low. To reset the counter to all zeroes, RF I/O is set low through an external open-collector driver.

During refresh, RASIN and RFSH must be skewed transitioning low such that the refresh address is valid on the address outputs of the controller before the RAS outputs go low. The amount of time that RFSH should go low before RASIN does depends on the capacitive loading of the address and RAS lines. For the load specified in the switching characteristics of this data sheet, 10ns is sufficient. Refer to Figure 2.

To perform externally controlled burst refresh, RASIN is toggled while RFSH is held low. The refresh counter increments with RASIN going low to high, so that the DRAM rows are refreshed in succession by RASIN going high to low.

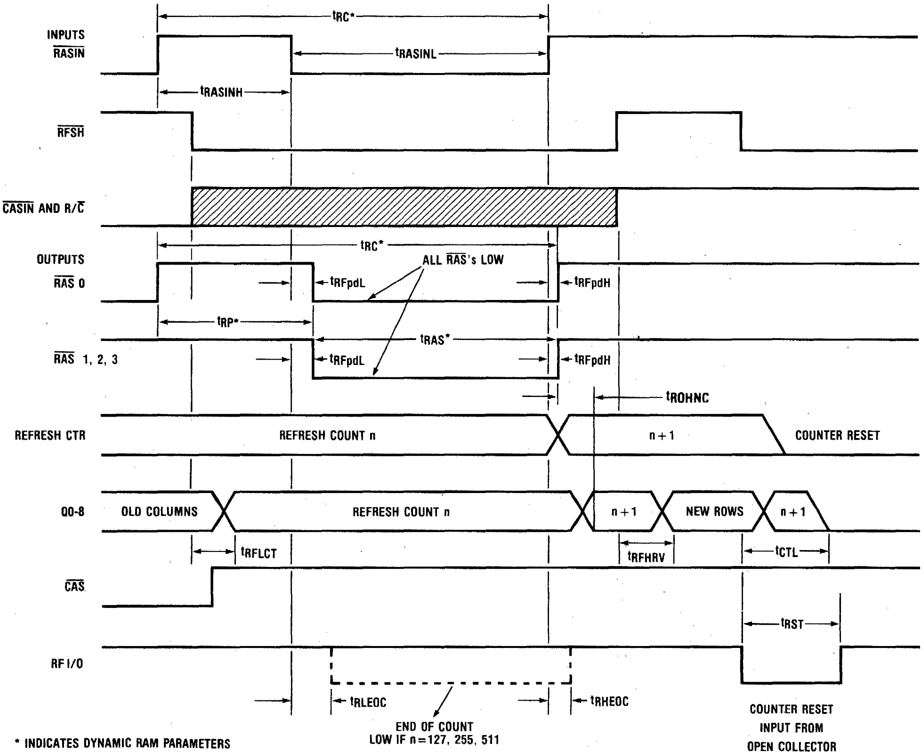


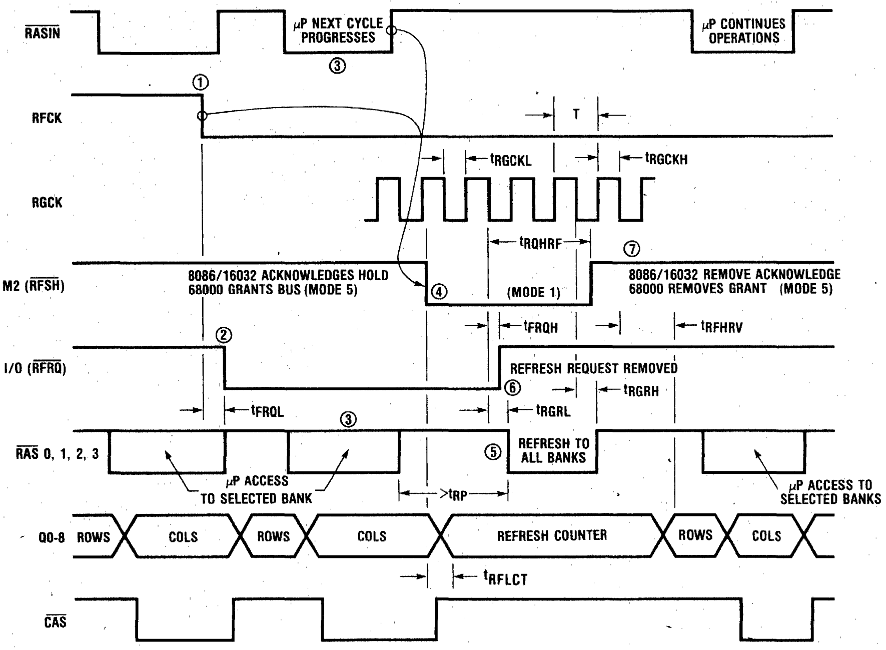
FIGURE 2. External Control Refresh Cycle (Mode 0)

Mode 1 — Automatic Forced Refresh

In Mode 1, the R/C (RFCK) pin becomes RFCK (refresh cycle clock), instead of R/C, and CAS remains high. If RFCK is kept permanently high, then whenever M2 (RFSH) goes low, an externally controlled refresh will occur and all RAS outputs will follow RASIN, strobing the refresh counter contents to the DRAMs. The RF I/O pin will always output high, but when set low externally through an open-collector driver, the refresh counter resets as normal. This externally controlled method may be preferred when operating in the Automatic Access mode (Mode 5), where hidden or forced refreshing is undesirable, but refreshing is still necessary.

If RFCK is an input clock signal, one (and only one) refresh cycle must take place every RFCK cycle. Refer to Figure 9. If a hidden refresh does not occur while RFCK is high, in Mode 5, then RF I/O (Refresh Request) goes low immediately after RFCK goes low, indicating to the system that a forced refresh is requested. The system must allow a forced refresh to take place while RFCK is low (refer to Figure 3). The Refresh Request signal on RF I/O may be connected to a Hold or Bus Request input to the system. The system acknowledges the Hold or Bus Request when ready, and outputs Hold Acknowledge or Bus Request Acknowledge. If this is connected to the M2 (RFSH) pin, a forced-refresh cycle will be initiated by the DP8409, and RAS will be internally generated on all four RAS outputs, to strobe the refresh counter contents on the address outputs into all the DRAMs. An external RAS Generator Clock (RGCK) is required for this function. It is fed to the CASIN (RGCK) pin, and may be up to 10MHz. Whenever M2 goes low (inducing a forced refresh), RAS remains high for one to two periods of RGCK, depending on when M2 goes low relative to the high-to-low triggering edge of RGCK; RAS then goes low for two periods, performing a refresh on all banks. In order to obtain the minimum delay from M2 going low to RAS going low, M2 should go low t_{RFSRG} before the next falling edge of RGCK. The Refresh Request on RF I/O is terminated as RAS begins, so that by the time the system has acknowledged the removal of the request and disabled its Acknowledge, (i.e., M2 goes high), Refresh RAS will have ended, and normal operations can begin again in the Automatic Access mode (Mode 5). If it is desired that Refresh RAS end in less than 2 periods of RGCK from the time RAS went low, then M2 may be high earlier than t_{RQHRF} after RGCK goes low and RAS will go high t_{RFRH} after M2, if CS is low. If CS is high, the RAS will go high impedance high after 25ns after M2 goes high.

To allow the forced refresh, the system will have been inactive for about 4 periods of RGCK, which can be as fast as 400ns every RFCK cycle. To guarantee a refresh of 128 rows every 2ms, a period of up to 16μs is required for RFCK. In other words, the system may be down for as little as 400ns every 16μs, or 2.5% of the time. Although this is not excessive, it may be preferable to perform a Hidden Refresh each RFCK cycle, which is allowed while still in the Auto-Access mode, (Mode 5).



① RFCK GOES LOW
② RFRQ GOES LOW IF NO HIDDEN REFRESH OCCURED WHILE RFCK WAS HIGH
③ NEXT RASIN STARTS NEXT ACCESS
④ μP ACKNOWLEDGES REFRESH REQUEST
⑤ FORCED REFRESH RAS STARTS AFTER >T (>t_{RP})
⑥ FORCED REFRESH RAS ENDS RFRQ
⑦ μP REMOVES REFRESH ACKNOWLEDGE

FIGURE 3. DP8409 Performing a Forced Refresh (Mode 5→1→5) with Various Microprocessors

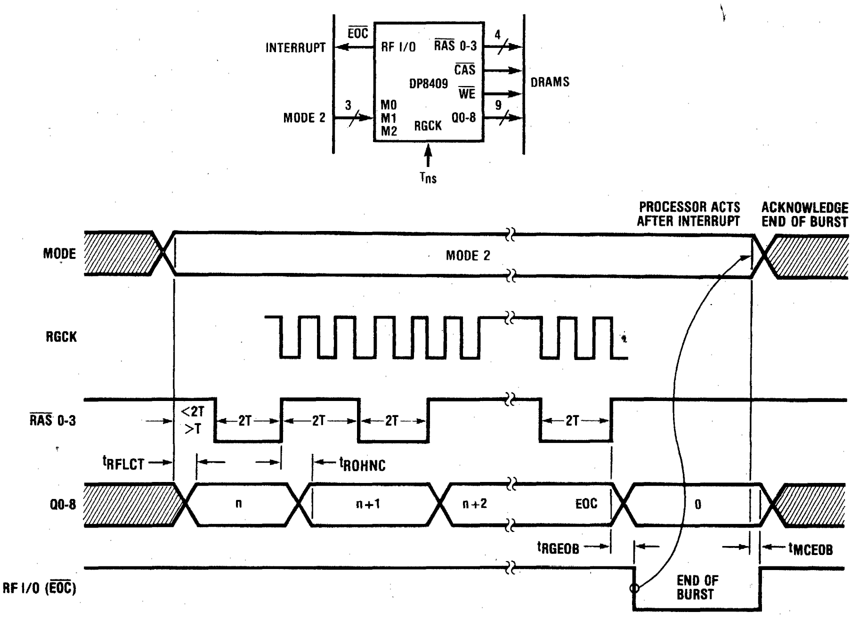


FIGURE 4. Auto-Burst Mode, Mode 2

Mode 2 — Automatic Burst Refresh

This mode is normally used before and/or after a DMA operation to ensure that all rows remain refreshed, provided the DMA transfer takes less than 2 ms (see Figure 4). When the DP8409 enters this mode, $\overline{\text{CASIN}}$ (RGCK) becomes the $\overline{\text{RAS}}$ Generator Clock (RGCK), and $\overline{\text{RASIN}}$ is disabled. $\overline{\text{CAS}}$ remains high, and RF I/O goes low when the refresh counter has reached the selected End-of-Count and the last $\overline{\text{RAS}}$ has ended. RF I/O then remains low until the Auto-Burst Refresh mode is terminated. RF I/O can therefore be used as an interrupt to indicate the End-of-Burst condition.

The signal on all four $\overline{\text{RAS}}$ outputs is just a divide-by-four of RGCK; in other words, if RGCK has a 100 ns period, $\overline{\text{RAS}}$ is high and low for 200 ns each cycle. The refresh counter increments at the end of each $\overline{\text{RAS}}$, starting from the count it contained when the mode was entered. If this was zero, then for a RGCK with a 100 ns period with End-of Count set to 127, RF I/O will go low after 128 x 0.4 μs, or 51.2μs. During this time, the system may be performing operations that do not involve DRAM. If all rows need to be burst refreshed, the refresh counter may be cleared by setting RF I/O low externally before the burst begins.

Burst-mode refreshing is also useful when powering down systems for long periods of time, but with data retention still required while the DRAMs are in standby. To maintain valid refreshing, power can be applied to the DP8409 (set to Mode 2), causing it to perform a complete burst refresh. When end-of-burst occurs (after 26 μs), power can then be removed from the DP8409 for 2 ms, consuming an average power of 1.3% of normal operating power. No control signal glitches occur when switching power to the DP8409.

Mode 3a — All-$\overline{\text{RAS}}$ Automatic Write

Mode 3a is useful at system initialization, when the memory is being cleared (i.e., with all-zeroes in the data field and the corresponding check bits for error detection and correction). This requires writing the same data to each location of memory (every row of each column of each bank). All $\overline{\text{RAS}}$ outputs are activated, as in refresh, and so are $\overline{\text{CAS}}$ and $\overline{\text{WE}}$. To write to all four banks simultaneously, every row is strobed in each column, in sequence, until data has been written to all locations.

To select this mode, B1 and B0 must have previously been set to 00, 01, or 10 in Mode 7, depending on the DRAM size. For example, for 16k DRAMs, B1 and B0 are 00. For 64k DRAMs, B1 and B0 are 01, so that for the configuration of Figure 1b, the 8 refresh counter bits are strobed by $\overline{\text{RAS}}$ into the 7 row addresses and the ninth column address. After this Automatic-Write process, B1 and B0 must be set again in Mode 7 to 00 to set End-of-Count to 127. For the configuration of Figure 1c, B1 and B0 set to 01 will work for Automatic-Write and End-of-Count equals 255.

In this mode, R/$\overline{\text{C}}$ is disabled, $\overline{\text{WE}}$ is permanently enabled low, and $\overline{\text{CASIN}}$ (RGCK) becomes RGCK. RF I/O goes low whenever the refresh counter is 127, 255, or 511 (as set by End-of-Count in Mode 7), and the $\overline{\text{RAS}}$ outputs are active.

Referring to Figure 5a, an external 8-bit counter (for 64k DRAMs) with TRI-STATE outputs is required and must be connected to the column address inputs. It is enabled only during this mode, and is clocked from RF I/O. The DP8409 refresh counter is used to address the rows, and the column address is supplied by the external counter. Every row for each column address is written to in all four banks. At the End-of-Count RF I/O goes low, which clocks the external counter.

Therefore, for each column address, the refresh counter first outputs row-0 to the address bus and all four $\overline{RAS}$ outputs strobe this row address into the DRAMs (see Figure 5b). A minimum of 30ns after $\overline{RAS}$ goes low (t_{RAH} = 30ns), the refresh counter is disabled and the column address input latch is enabled onto the address bus. About 14ns after the column address is valid, $\overline{CAS}$ goes low, (t_{ASC} = +14ns), strobing the column address into the DRAMs. When $\overline{RAS}$ and $\overline{CAS}$ go high the refresh counter increments to the next row and the cycle repeats. Since $\overline{WE}$ is kept low in this mode, the data at DI (input data) of the DRAMs is written into each row of the latched column. During each cycle $\overline{RAS}$ is high for two periods of RGCK and low for two periods, giving a total write-cycle time of 400ns minimum, which is adequate for most 16k and 64k DRAMs. On the last row of a column, RF I/O increments the external counter to the next column address.

At the end of the last column address, an interrupt is generated from the external counter to let the system know that initialization has been completed. During the entire initialization time, the system can be performing other initialization functions. This approach to memory initialization is both automatic and fast. For instance, if four banks of 64k DRAMs are used, and RGCK is 100ns, a write cycle to the same location in all four banks takes 400ns, so the total time taken in initializing the 64k DRAMs is 65k × 400ns or 26ms. When the system receives the interrupt, the external counter must be permanently disabled. ADS and $\overline{CS}$ are interfaced by the system, and the DP8409 mode is changed. The interrupt must then be disabled.

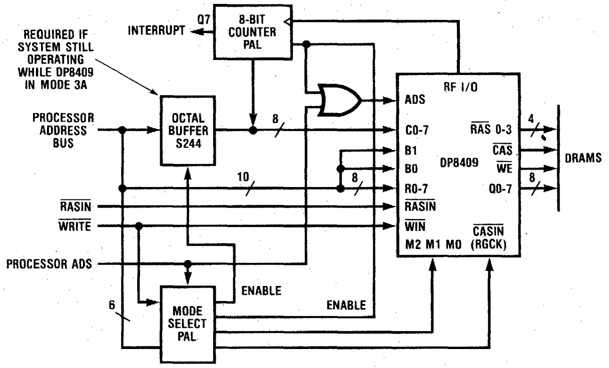


FIGURE 5a. DP8409 Extra Circuitry Required for All-$\overline{RAS}$ Auto Write Mode, Mode 3a

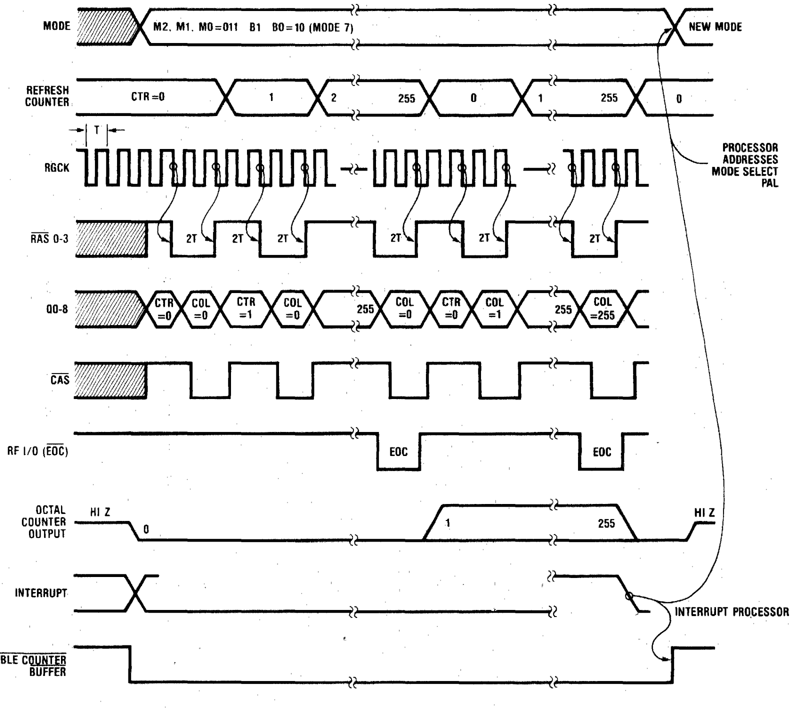


FIGURE 5b. DP8409 All-$\overline{RAS}$ Auto Write Mode, Mode 3a, Timing Waveform

Mode 3b — Externally Controlled All-RAS Write

To select this mode, B1 and B0 must first have been set to 11 in Mode 7. This mode is useful at system initialization, but under processor control. The memory address is provided by the processor, which also performs the incrementing. All four RAS outputs follow RASIN (supplied by the processor), strobing the row address into the DRAMs. R/C can now go low, while CASIN may be used to control CAS (as in the Externally Controlled Access mode), so that CAS strobes the column address contents into the DRAMs. At this time WE should be low, causing the data to be written into all four banks of DRAMs. At the end of the write cycle, the input address is incremented and latched by the DP8409 for the next write cycle. This method is slower than Mode 3a since the processor must perform the incrementing and accessing. Thus the processor is occupied during RAM initialization, and is not free for other initialization operations. However, initialization sequence timing is under system control, which may provide some system advantage.

Mode 4 — Externally Controlled Access

This mode facilitates externally controlling all access-timing parameters associated with the DRAMs. The application of modes 0 and 4 are shown in Figure 6.

Output Address Selection

Refer to Figure 7a. With M2 (RFSH) and R/C high, the row address latch contents are transferred to the multiplexed address bus output Q0–Q8, provided CS is set low. The column address latch contents are output after R/C goes low. RASIN can go low after the row addresses have been set up on Q0–Q8. This selects one of the RAS outputs, strobing the row address on the Q outputs into the desired bank of memory. After the row-address hold-time of the DRAMs, R/C can go low so that about 40 ns later column addresses appear on the Q outputs.

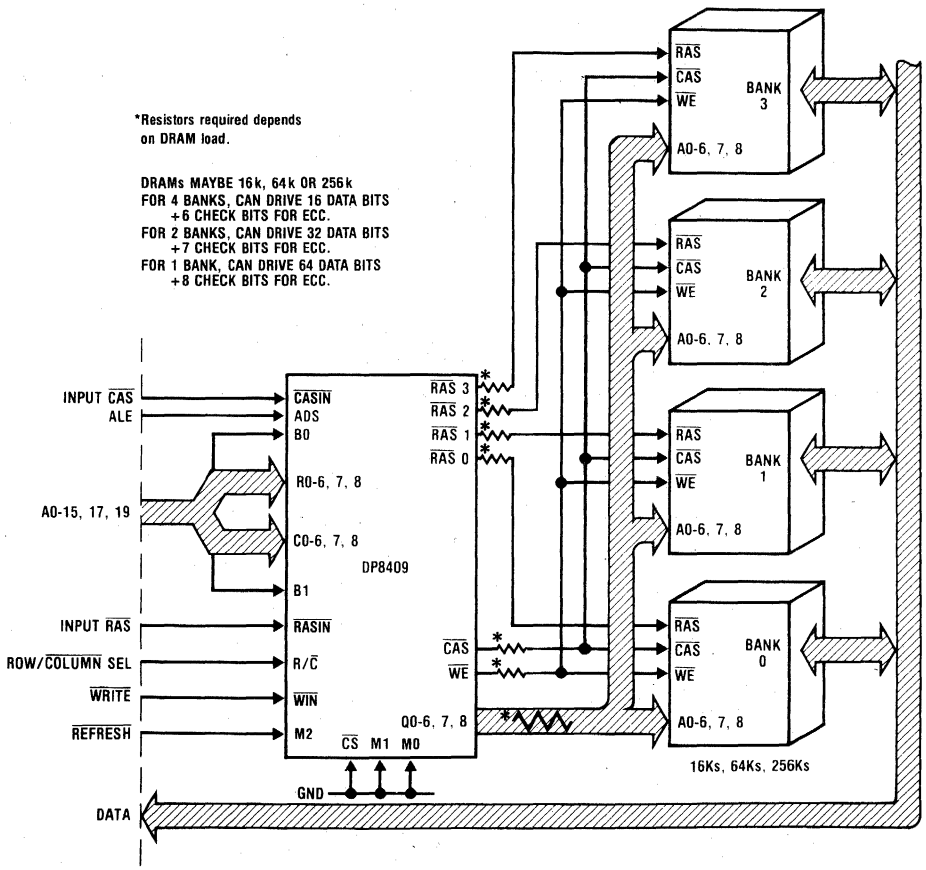


FIGURE 6. Typical Application of DP8409 Using External Control Access and Refresh in Modes 0 and 4

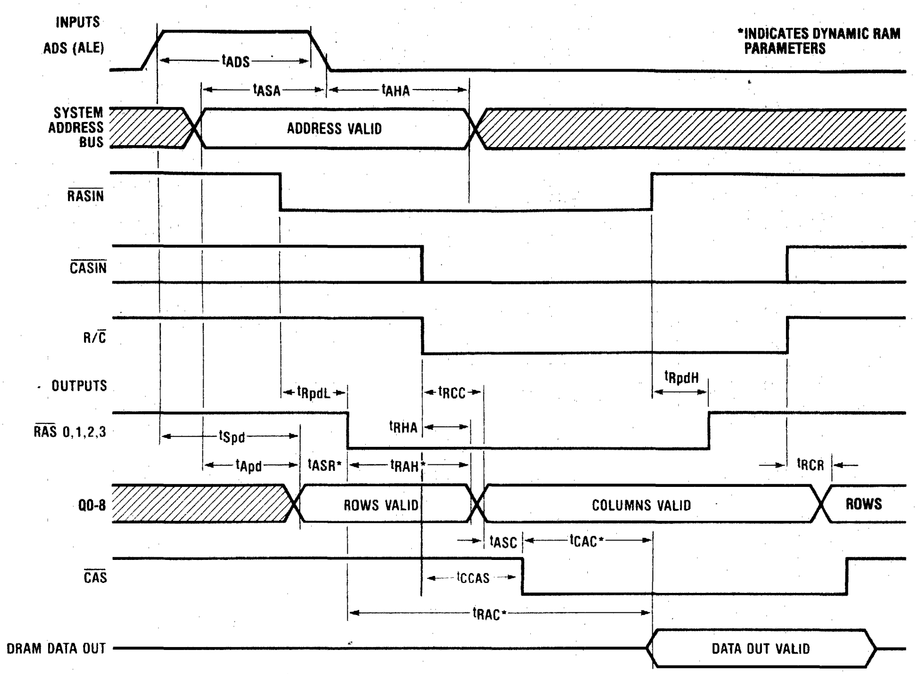


FIGURE 7a. Read Cycle Timing (MODE 4)

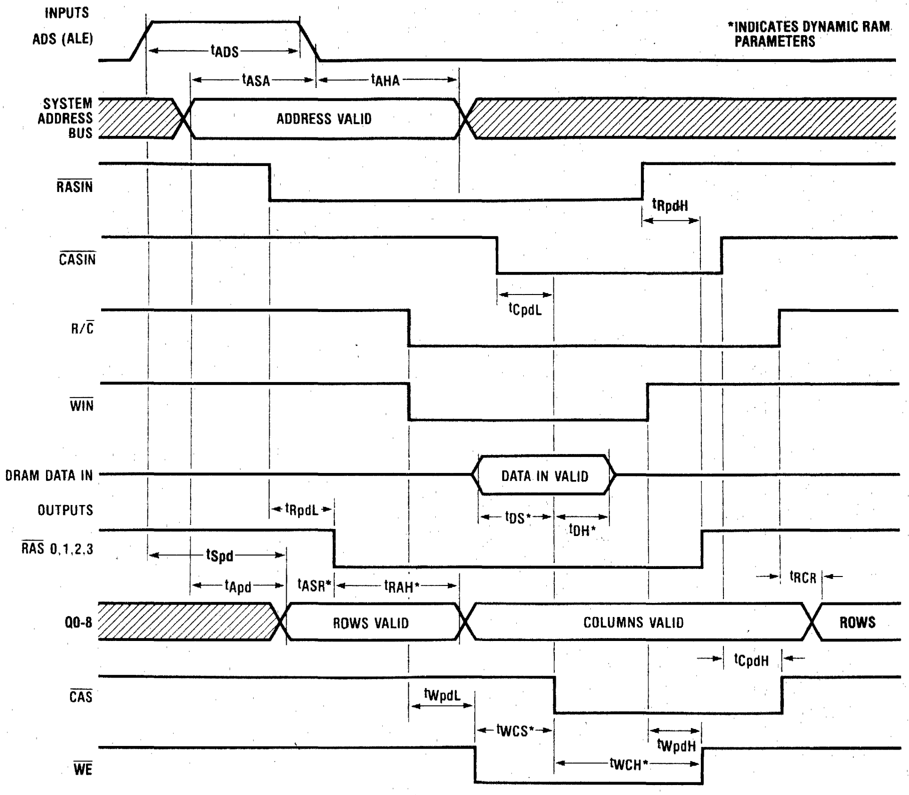


FIGURE 7b. Write Cycle Timing (Mode 4)

Automatic CAS Generation

In a normal memory access cycle CAS can be derived from inputs CASIN or R/C. If CASIN is high, then R/C going low switches the address output drivers from rows to columns. CASIN then going low causes CAS to go low approximately 40 ns later, allowing CAS to occur at a predictable time (see Figure 7b). If CASIN is low when R/C goes low, CAS will be automatically generated, following the row to column transition by about 20 ns (see Figure 7a). Most DRAMs have a column address set-up time before CAS (t_{ASC}) of 0 ns or – 10 ns. In other words, a t_{ASC} greater than 0 ns is safe.

Fast Memory Access

AC parameters t_{DIF1}, t_{DIF2} may be used to determine the minimum delays required between RASIN, R/C, and CASIN (see Application Brief 9; "Fastest DRAM Access Mode").

Mode 5 — Automatic Access with Hidden Refresh

The Auto Access with Hidden Refresh mode has two advantages over the Externally Controlled Access mode, due to the fact that all outputs except WE are initiated from RASIN. First, inputs R/C and CASIN are unnecessary and can be used for other functions (see Refreshing, below). Secondly, because the output control signals are derived internally from one input signal (RASIN), timing-skew problems are reduced, thereby reducing memory access time substantially or allowing use of slower DRAMs. The automatic access features of Mode 5 (and Mode 6) of the DP8409 make DRAM accessing appear essentially "static".

Automatic Access Control

The major disadvantage of DRAMs compared to static RAMs is the complex timing involved. First, a RAS must occur with the row address previously set up on the multiplexed address bus. After the row address has been held for t_{RAH}, (the Row-Address hold-time of the DRAM), the column address is set up and then CAS occurs. This is all performed automatically by the DP8409 in this mode.

Provided the input address is valid as ADS goes low, RASIN can go low any time after ADS. This is because the selected RAS occurs typically 27ns later, by which time the row address is already valid on the address output of the DP8409. The Address Set-Up time (t_{ASR}), is 0ns on most DRAMs. The DP8409 in this mode (with ADS and RASIN edges simultaneously applied) produces a minimum t_{ASR} of 0ns. This is true provided the input address was valid t_{ASA} before ADS went low (see Figure 8a).

Next, the row address is disabled after t_{RAH} (30ns minimum); in most DRAMs, t_{RAH} minimum is less than 30ns. The column address is then set up and t_{ASC} later, CAS occurs. The only other control input required is WIN. When a write cycle is required, WIN must go low at least 30ns before CAS is output low.

This gives a total typical delay from: input address valid to RASIN (15ns); to RAS (27ns); to rows held (50ns); to columns valid (25ns); to CAS (23ns) = 140ns (that is, 125ns from RASIN). All of these typical figures are for heavy capacitive loading, of approximately 88 DRAMs. This mode is therefore extremely fast. The external timing is greatly simplified for the memory system designer: the only system signal required is RASIN.

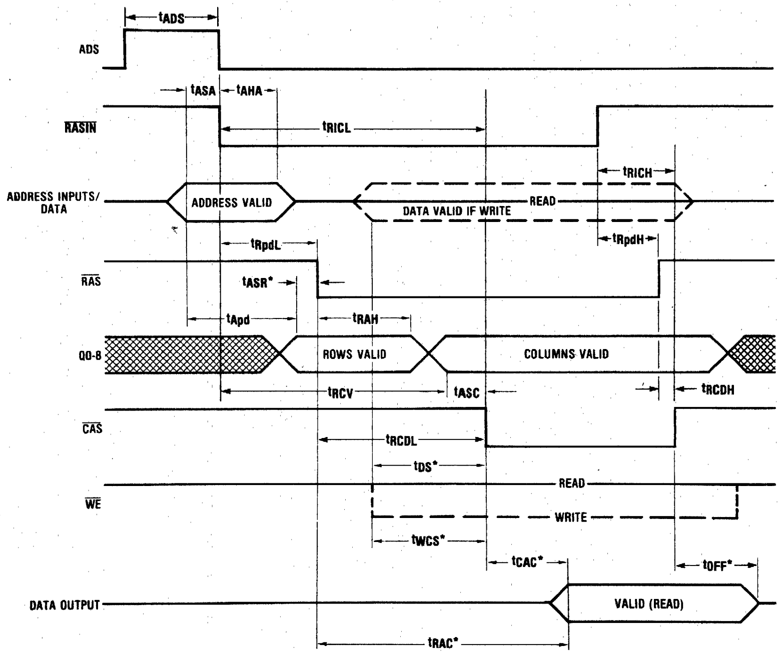
Refreshing

Because R/C and CASIN are not used in this mode, R/C becomes RFCK (refresh clock) and CASIN becomes RGCK (RAS generator clock). With these two signals it is possible to perform refreshing without extra ICs, and without holding up the processor.

One refresh cycle must occur during each refresh clock period and then the refresh address must be incremented to the next refresh cycle. As long as 128 rows are refreshed every 2ms (one row every 16μs), all 16k and 64k DRAMs will be correctly refreshed. The cycle time of RFCK must, therefore, be less than 16μs. RFCK going high sets an internal refresh-request flip-flop. First the DP8409 will attempt to perform a hidden refresh so that the system throughput will not be affected. If, during the time RFCK is high, CS on the DP8409 goes high and RASIN occurs, a hidden refresh will occur. In this case, RASIN should be considered a common read/write strobe. In other words, if the processor is accessing elsewhere (other than the DRAMs) while RFCK is high, the DP8409 will perform a refresh. The refresh counter is enabled to the address outputs whenever CS goes high with RFCK high, and all RAS outputs follow RASIN. If a hidden refresh is taking place as RFCK goes low, the refresh continues. At the start of the hidden refresh, the refresh-request flip-flop is reset so no further refresh can occur until the next RFCK period starts with the positive-going edge of RFCK. Refer to Figure 9.

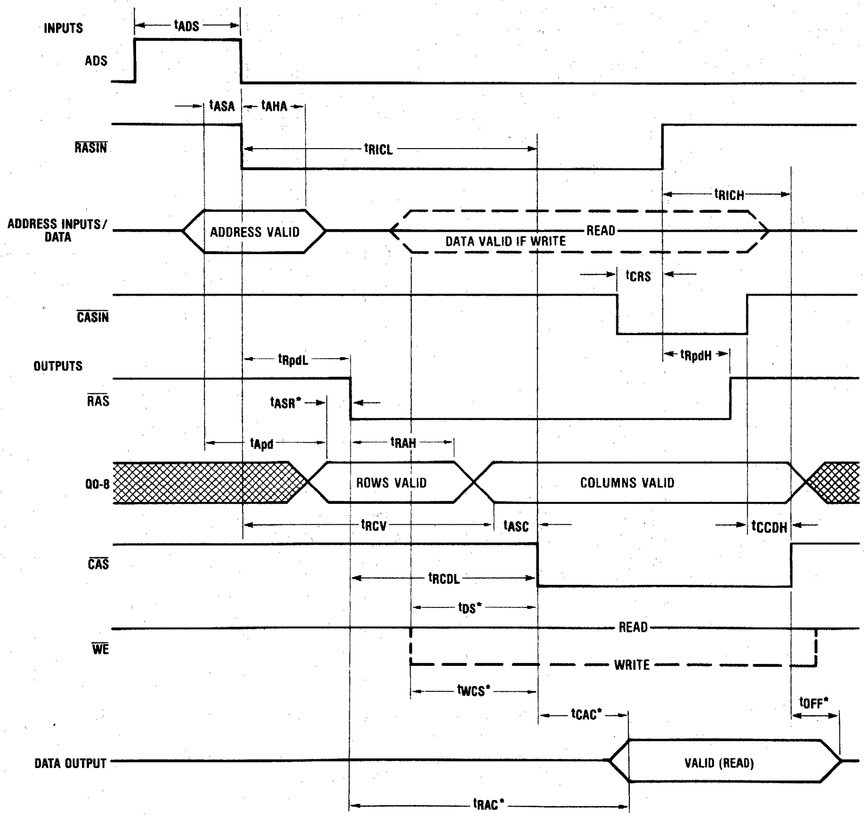
To determine the probability of a Hidden Refresh occurring, assume each system cycle takes 400ns and RFCK is high for 8μs, then the system has 20 chances to not select the DP8409. If during this time a hidden refresh did not occur, then the DP8409 forces a refresh while RFCK is low, but the system chooses when the refresh takes place. After RFCK goes low, (and the internal-request flip-flop has not been reset), RF I/O goes low indicating that a refresh is requested to the system. Only when the system acknowledges this request by setting M2 (RFSH) low does the DP8409 initiate a forced refresh (which is performed automatically). Refer to Mode 1, and Figure 3. The internal refresh request flip-flop is then reset.

Figure 9 illustrates the refresh alternatives in Mode 5. If a hidden refresh has occurred and CS again goes high before RFCK goes low, the chip is deselected. All the control signals go high-impedance high (logic "1") and the address outputs go TRI-STATE until CS again goes low. This mode (combined with Mode 1) allows very fast access, and automatic refreshing (possibly not even slowing down the system), with no extra ICs. Careful system design can, and should, provide a higher probability of hidden refresh occurring. The duty cycle of RFCK need not be 50-percent; in fact, the low-time should be designed to be a minimum. This is determined by the worst-case time (required by the system) to respond to the DP8409's forced-refresh request.



*INDICATES DYNAMIC RAM PARAMETERS

FIGURE 8a. Modes 5, 6 Timing ($\overline{\text{CASIN}}$ High in Mode 6)



*INDICATES DYNAMIC RAM PARAMETERS

FIGURE 8b. Mode 6 Timing, Extended $\overline{\text{CAS}}$

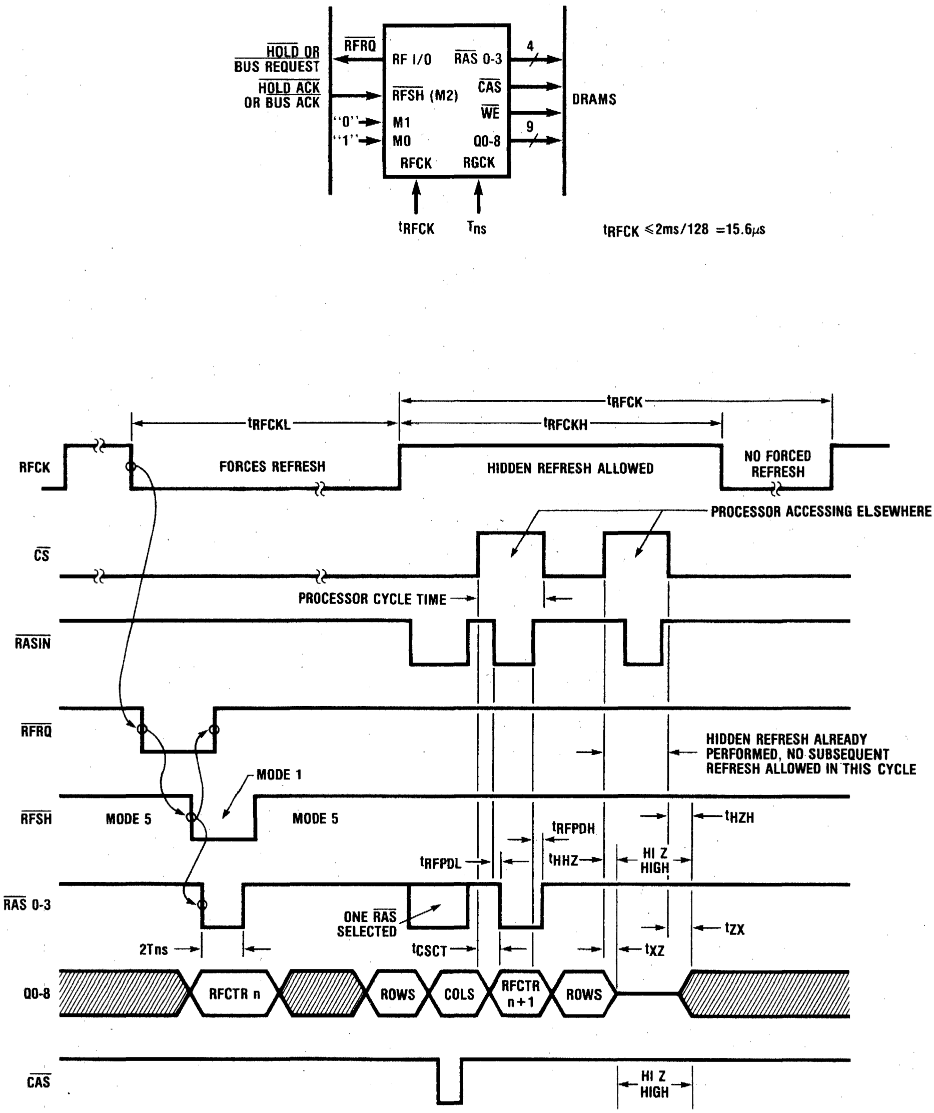


FIGURE 9. Hidden Refreshing (Mode 5) and Forced Refreshing (Mode 1) Timing

Table 2. Memory Bank Decode

Bank Select (Strobed by ADS)		Enabled $\overline{\text{RAS}}_n$
B1	B0	
0	0	$\overline{\text{RAS}}_0$
0	1	$\overline{\text{RAS}}_1$
1	0	$\overline{\text{RAS}}_2$
1	1	$\overline{\text{RAS}}_3$

Note that $\overline{\text{RASIN}}$ going low earlier than t_{CSRL} after $\overline{\text{CS}}$ goes low may result in the DP8409 interpreting the $\overline{\text{RASIN}}$ as a hidden refresh $\overline{\text{RASIN}}$ if no hidden refresh has occurred in the current RFCK cycle. In this case, all $\overline{\text{RAS}}$ outputs would go low for a short time. Thus, it is suggested that when using Mode 5, $\overline{\text{RASIN}}$ should be held high until t_{CSRL} after $\overline{\text{CS}}$ goes low if a refresh is not intended. Similarly, $\overline{\text{CS}}$ should be held low for a minimum of t_{CSRL} after $\overline{\text{RASIN}}$ returns high when ending the access in Mode 5.

Mode 6 — Fast Automatic Access

The Fast Access mode is similar to Mode 5, but has a faster t_{RAH} of 20ns, minimum. It therefore can only be used with fast 16k or 64k DRAMs (which have a t_{RAH} of 10ns to 15ns) in applications requiring fast access times; $\overline{\text{RASIN}}$ to $\overline{\text{CAS}}$ is typically 105ns.

In this mode, the R/$\overline{\text{C}}$ (RFCK) pin is not used, but $\overline{\text{CASIN}}$ (RGCK) is used as $\overline{\text{CASIN}}$ to allow an extended $\overline{\text{CAS}}$ after $\overline{\text{RAS}}$ has already terminated. Refer to Figure 8b. This is desirable with fast cycle-times where $\overline{\text{RAS}}$ has to be terminated as soon as possible before the next $\overline{\text{RAS}}$ begins (to meet the precharge time, or t_{RP}, requirements of the DRAM). $\overline{\text{CAS}}$ may then be held low by $\overline{\text{CASIN}}$ to extend the data output valid time from the DRAM to allow the system to read the data. $\overline{\text{CASIN}}$ subsequently going high ends $\overline{\text{CAS}}$. If this extended $\overline{\text{CAS}}$ is not required, $\overline{\text{CASIN}}$ should be set high in Mode 6.

There is no internal refresh-request flip-flop in this mode, so any refreshing required must be done by entering Mode 0 or Mode 2.

Mode 7 — Set End-of-Count

The End-of-Count can be externally selected in Mode 7, using ADS to strobe in the respective value of B1 and B0 (see Table 3). With B1 and B0 the same $\overline{\text{EOC}}$ is 127; with B1 = 0 and B0 = 1, $\overline{\text{EOC}}$ is 255; and with B1 = 1 and B0 = 0, $\overline{\text{EOC}}$ is 511. This selected value of $\overline{\text{EOC}}$ will be used until the next Mode 7 selection. At power-up the $\overline{\text{EOC}}$ is automatically set to 127 (B1 and B0 set to 11).

Table 3. Mode 7

Bank Select (Strobed by ADS)		End of Count Selected
B1	B0	
0	0	127
0	1	255
1	0	511
1	1	127

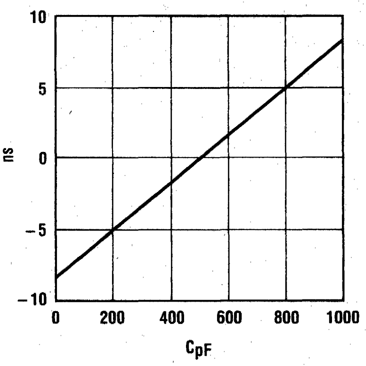


FIGURE 10. Change in Propagation Delay vs. Loading Capacitance Relative to a 500pF Load

Absolute Maximum Ratings (Note 1)

Supply Voltage, V_{CC}	7.0V
Storage Temperature Range	−65°C to +150°C
Input Voltage	5.5V
Output Current	150mA
Lead Temperature (Soldering, 10 seconds)	300°C

*Derate cavity package 23.6mW/°C above 25°C; derate molded package 22.7mW/°C above 25°C.

Maximum Power Dissipation* at 25°C	
Cavity Package	3542mW
Molded Package	2833mW

Operating Conditions

		Min	Max	Units
V_{CC}	Supply Voltage	4.75	5.25	V
T_A	Ambient Temperature	0	+70	°C

Electrical Characteristics $V_{CC} = 5.0V \pm 5\%$, $0°C \leqslant T_A \leqslant 70°C$ unless otherwise noted (Notes 2, 6)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_C	Input Clamp Voltage	V_{CC} = Min., $I_C = -12mA$		−0.8	−1.2	V
I_{IH1}	Input High Current for ADS, R/$\overline{C}$ only	$V_{IN} = 2.5V$		2.0	100	μA
I_{IH2}	Input High Current for All Other Inputs*	$V_{IN} = 2.5V$		1.0	50	μA
I_I RSI	Output Load Current for RF I/O	$V_{IN} = 0.5V$, Output High		−1.5	−2.5	mA
I_I CTL	Output Load Current for $\overline{RAS}$, $\overline{CAS}$, $\overline{WE}$	$V_{IN} = 0.5V$, Chip Deselect		−1.5	−2.5	mA
I_{IL1}	Input Low Current for ADS, R/$\overline{C}$ only	$V_{IN} = 0.5V$		−0.1	−1.0	mA
I_{IL2}	Input Low Current for All Other Inputs*	$V_{IN} = 0.5V$		−0.05	−0.5	mA
V_{IL}	Input Low Threshold				0.8	V
V_{IH}	Input High Threshold		2.0			V
V_{OL1}	Output Low Voltage*	$I_{OL} = 20mA$		0.3	0.5	V
V_{OL2}	Output Low Voltage for RF I/O	$I_{OL} = 10mA$		0.3	0.5	V
V_{OH1}	Output High Voltage*	$I_{OH} = -1mA$	2.4	3.5		V
V_{OH2}	Output High Voltage for RF I/O	$I_{OH} = -100\mu A$	2.4	3.5		V
I_{1D}	Output High Drive Current*	$V_{OUT} = 0.8V$ (Note 3)		−200		mA
I_{0D}	Output Low Drive Current*	$V_{OUT} = 2.7V$ (Note 3)		200		mA
I_{OZ}	TRI-STATE Output Current (Address Outputs)	$0.4V \leqslant V_{OUT} \leqslant 2.7V$, $\overline{CS} = 2.0V$, Mode 4	−50	1.0	50	μA
I_{CC}	Supply Current	V_{CC} = Max.		250	325	mA

*Except RF I/O Output.

Switching Characteristics: DP8409/DP8409-3 $V_{CC} = 5.0V \pm 5\%$, $0°C \leqslant T_A \leqslant 70°C$ unless otherwise noted (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0–Q8, $C_L = 500pF$; $\overline{RAS0}$–$\overline{RAS3}$, $C_L = 150pF$; $\overline{WE}$, $C_L = 500pF$; $\overline{CAS}$, $C_L = 600pF$, unless otherwise noted. See Figure 11 for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Access Parameter	Conditions	8409			8409 − 3			Units
			Min	Typ	Max	Min	Typ	Max	
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a	95	125	160	95	125	185	ns
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 8a, 8b	80	105	140	80	105	160	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a	40	48	60	40	48	70	ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 8a, 8b	50	63	80	50	63	95	ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a		98	125		98	145	ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 8a, 8b		78	105		78	120	ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a		27	40		27	40	ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figure 8a		40	65		40	65	ns
t_{CCDH}	$\overline{CASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figure 8b	40	54	70	40	54	80	ns
t_{RAH}	Row Address Hold Time (Mode 5)	Figure 8a	30			30			ns
t_{RAH}	Row Address Hold Time (Mode 6)	Figures 8a, 8b	20			20			ns
t_{ASC}	Column Address Setup Time (Mode 5)	Figure 8a	8			8			ns
t_{ASC}	Column Address Setup Time (Mode 6)	Figures 8a, 8b	6			6			ns
t_{RCV}	$\overline{RASIN}$ to Column Address Valid (Mode 5)	Figure 8a		90	120		90	140	ns

Switching Characteristics (Cont'd)

Symbol	Access Parameter	Conditions	8409			8409 – 3			Units
			Min	Typ	Max	Min	Typ	Max	
t_{RCV}	$\overline{\text{RASIN}}$ to Column Address Valid (Mode 6)	Figures 8a, 8b		75	105		75	120	ns
t_{RPDL}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay	Figures 7a, 7b, 8a, 8b	20	27	35	20	27	40	ns
t_{RPDH}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay	Figures 7a, 7b, 8a, 8b	15	23	32	15	23	37	ns
t_{APDL}	Address Input to Output Low Delay	Figures 7a, 7b, 8a, 8b		25	40		25	46	ns
t_{APDH}	Address Input to Output High Delay	Figures 7a, 7b, 8a, 8b		25	40		25	46	ns
t_{SPDL}	Address Strobe to Address Output Low	Figures 7a, 7b		40	60		40	70	ns
t_{SPDH}	Address Strobe to Address Output High	Figures 7a, 7b		40	60		40	70	ns
t_{ASA}	Address Set-up Time to ADS	Figures 7a, 7b, 8a, 8b	15			15			ns
t_{AHA}	Address Hold Time from ADS	Figures 7a, 7b, 8a, 8b	15			15			ns
t_{ADS}	Address Strobe Pulse Width	Figures 7a, 7b, 8a, 8b	30			30			ns
t_{WPDL}	$\overline{\text{WIN}}$ to $\overline{\text{WE}}$ Output Delay	Figure 7b	15	25	30	15	25	35	ns
t_{WPDH}	$\overline{\text{WIN}}$ to $\overline{\text{WE}}$ Output Delay	Figure 7b	15	30	60	15	30	70	ns
t_{CRS}	$\overline{\text{CASIN}}$ Set-up Time to $\overline{\text{RASIN}}$ High (Mode 6)	Figure 8b	35			35			ns
t_{CPDL}	$\overline{\text{CASIN}}$ to $\overline{\text{CAS}}$ Delay (R/$\overline{\text{C}}$ low in Mode 4)	Figure 7b	32	41	68	32	41	77	ns
t_{CPDH}	$\overline{\text{CASIN}}$ to $\overline{\text{CAS}}$ Delay (R/$\overline{\text{C}}$ low in Mode 4)	Figure 7b	25	39	50	25	39	60	ns
t_{RCC}	Column Select to Column Address Valid	Figure 7a		40	58		40	67	ns
t_{RCR}	Row Select to Row Address Valid	Figures 7a, 7b		40	58		40	67	ns
t_{RHA}	Row Address Held from Column Select	Figure 7a	10			10			ns
t_{CCAS}	R/$\overline{\text{C}}$ Low to $\overline{\text{CAS}}$ Low (Mode 4 Auto $\overline{\text{CAS}}$)	Figure 7a		65	90				ns
t_{DIF1}	Maximum ($t_{RPDL} - t_{RHA}$)	See Mode 4 description			13			18	ns
t_{DIF2}	Maximum ($t_{RCC} - t_{CPDL}$)	See Mode 4 description			13			18	ns
	Refresh Parameter								
t_{RC}	Refresh Cycle Period	Figure 2	100			100			ns
$t_{RASINL,H}$	Pulse Width of $\overline{\text{RASIN}}$ during Refresh	Figure 2	50			50			ns
t_{RFPDL}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay during Refresh	Figures 2, 9	35	50	70	35	50	80	ns
t_{RFPDH}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay during Refresh	Figures 2, 9	30	40	55	30	40	65	ns
t_{RFLCT}	$\overline{\text{RFSH}}$ Low to Counter Address Valid	$\overline{\text{CS}}$ = X, Figures 2,3,4		47	60		47	70	ns
t_{RFHRV}	$\overline{\text{RFSH}}$ High to Row Address Valid	Figures 2, 3		45	60		45	70	ns
t_{ROHNC}	$\overline{\text{RAS}}$ High to New Count Valid	Figures 2, 4		30	55		30	55	ns
t_{RLEOC}	$\overline{\text{RASIN}}$ Low to End-of-Count Low	$C_L = 50pF$, Figure 2			80			80	ns
t_{RHEOC}	$\overline{\text{RASIN}}$ High to End-of-Count High	$C_L = 50pF$, Figure 2			80			80	ns
t_{RGEOB}	RGCK Low to End-of-Burst Low	$C_L = 50pF$, Figure 4			95			95	ns
t_{MCEOB}	Mode Change to End-of-Burst High	$C_L = 50pF$, Figure 4			75			75	ns
t_{RST}	Counter Reset Pulse Width	Figure 2	70			70			ns
t_{CTL}	RF I/O Low to Counter Outputs All Low	Figure 2			100			100	ns
$t_{RFCKL,H}$	Minimum Pulse Width of RFCK	Figure 9	100			100			ns
T	Period of $\overline{\text{RAS}}$ Generator Clock	Figure 3	100			100			ns
t_{RGCKL}	Minimum Pulse Width Low of RGCK	Figure 3	35			40			ns
t_{RGCKH}	Minimum Pulse Width High of RGCK	Figure 3	35			40			ns
t_{FRQL}	RFCK Low to Forced $\overline{\text{RFRQ}}$ Low	$C_L = 50pF$, Figure 3		20	30		20	30	ns

Switching Characteristics (Cont'd)

Symbol	Refresh Parameter	Conditions	8409			8409 – 3			Units
			Min	Typ	Max	Min	Typ	Max	
t_{FRQH}	RGCK Low to Forced $\overline{RFRQ}$ High	$C_L = 50pF$, Figure 3		50	75		50	75	ns
t_{RGRL}	RGCK Low to $\overline{RAS}$ Low	Figure 3	50	65	95	50	65	95	ns
t_{RGRH}	RGCK Low to $\overline{RAS}$ High	Figure 3	40	60	85	40	60	85	ns
t_{RQHRF}	$\overline{RFSH}$ Hold Time from $\overline{RFSH\ RQST}$ (RF I/O)	Figure 3	2T			2T			ns
t_{RFRH}	$\overline{RFSH}$ High to $\overline{RAS}$ High (ending forced RFSH)	See Mode 1 Descrip.	55	80	110	55	80	125	ns
t_{RFSRG}	$\overline{RFSH}$ Low Set-up to RGCK Low (Mode 1)	See Mode 1 Descrip.	35			40			ns
t_{CSCT}	$\overline{CS}$ High to RFSH Counter Valid	Figure 9		55	70		55	75	ns
t_{CSRL}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low	See Mode 5 Descrip.	10			15			ns
	TRI-STATE Parameter								
t_{ZH}	$\overline{CS}$ Low to Address Output High from Hi-Z	Figures 9, 12 R1 = 3.5k, R2 = 1.5k		35	60		35	60	ns
t_{HZ}	$\overline{CS}$ High to Address Output Hi-Z from High	$C_L = 15pF$, Figures 9, 12 R2 = 1k, S1 open		20	40		20	40	ns
t_{ZL}	$\overline{CS}$ Low to Address Output Low from Hi-Z	Figures 9, 12 R1 = 3.5k, R2 = 1.5k		35	60		35	60	ns
t_{LZ}	$\overline{CS}$ High to Address Output Hi-Z from Low	$C_L = 15pF$, Figures 9, 12, R1 = 1k, S2 open		25	50		25	50	ns
t_{HZH}	$\overline{CS}$ Low to Control Output High from Hi-Z High	Figures 9, 12 R2 = 750Ω, S1 open		50	80		50	80	ns
t_{HHZ}	$\overline{CS}$ High to Control Output Hi-Z High from High	$C_L = 15pF$, Figures 9, 12 R2 = 750Ω, S1 open		40	75		40	75	ns
t_{HZL}	$\overline{CS}$ Low to Control Output Low from Hi-Z High	Figure 12, S1, S2 open		45	75		45	75	ns
t_{LHZ}	$\overline{CS}$ High to Control Output Hi-Z High from Low	$C_L = 15pF$, Figure 12, R2 = 750Ω, S1 open		50	80		50	80	ns

Switching Characteristics: DP8409-2

$V_{CC} = 5.0V \pm 5\%$, $0°C \leq T_A \leq 70°C$ unless otherwise noted (Notes 2, 4, 5). The output load capacitance is typical for 4 banks of 22 DRAMs each or 88 DRAMs including trace capacitance. These values are: Q0-Q8, $C_L = 500pF$; $\overline{RAS0}$-$\overline{RAS3}$, $C_L = 150pF$; $\overline{WE}$, $C_L = 500pF$; $\overline{CAS}$, $C_L = 600pF$, unless otherwise noted. See Figure 11 for test load. Switches S1 and S2 are closed unless otherwise noted, and R1 and R2 are 4.7kΩ unless otherwise noted. Maximum propagation delays are specified with all outputs switching.

Symbol	Access Parameter	Conditions	8409 – 2						Units
			Min	Typ	Max	Min	Typ	Max	
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a	75	100	130				ns
t_{RICL}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 8a, 8b	65	90	115				ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a	40	48	60				ns
t_{RICH}	$\overline{RASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 8a, 8b	50	63	80				ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a		75	100				ns
t_{RCDL}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figures 8a, 8b		65	85				ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 5)	Figure 8a		27	40				ns
t_{RCDH}	$\overline{RAS}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figure 8a		40	65				ns
t_{CCDH}	$\overline{CASIN}$ to $\overline{CAS}$ Output Delay (Mode 6)	Figure 8b	40	54	70				ns
t_{RAH}	Row Address Hold Time (Mode 5) (Note 7)	Figure 8a	20						ns

Switching Characteristics (Cont'd)

Symbol	Access Parameter	Conditions	8409 – 2						Units
			Min	Typ	Max	Min	Typ	Max	
t_{RAH}	Row Address Hold Time (Mode 6) (Note 7)	Figures 8a, 8b	12						ns
t_{ASC}	Column Address Setup Time (Mode 5)	Figure 8a	3						ns
t_{ASC}	Column Address Setup Time (Mode 6)	Figures 8a, 8b	3						ns
t_{RCV}	$\overline{\text{RASIN}}$ to Column Address Valid (Mode 5)	Figure 8a		80	105				ns
t_{RCV}	$\overline{\text{RASIN}}$ to Column Address Valid (Mode 6)	Figures 8a, 8b		70	90				ns
t_{RPDL}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay	Figures 7a, 7b, 8a, 8b	20	27	35				ns
t_{RPDH}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay	Figures 7a, 7b, 8a, 8b	15	23	32				ns
t_{APDL}	Address Input to Output Low Delay	Figures 7a, 7b, 8a, 8b		25	40				ns
t_{APDH}	Address Input to Output High Delay	Figures 7a, 7b, 8a, 8b		25	40				ns
t_{SPDL}	Address Strobe to Address Output Low	Figures 7a, 7b		40	60				ns
t_{SPDH}	Address Strobe to Address Output High	Figures 7a, 7b		40	60				ns
t_{ASA}	Address Set-up Time to ADS	Figures 7a, 7b, 8a, 8b	15						ns
t_{AHA}	Address Hold Time from ADS	Figures 7a, 7b, 8a, 8b	15						ns
t_{ADS}	Address Strobe Pulse Width	Figures 7a, 7b, 8a, 8b	30						ns
t_{WPDL}	$\overline{\text{WIN}}$ to $\overline{\text{WE}}$ Output Delay	Figure 7b	15	25	30				ns
t_{WPDH}	$\overline{\text{WIN}}$ to $\overline{\text{WE}}$ Output Delay	Figure 7b	15	30	60				ns
t_{CRS}	$\overline{\text{CASIN}}$ Set-up Time to $\overline{\text{RASIN}}$ High (Mode 6)	Figure 8b	35						ns
t_{CPDL}	$\overline{\text{CASIN}}$ to $\overline{\text{CAS}}$ Delay (R/$\overline{\text{C}}$ low in Mode 4)	Figure 7b	32	41	58				ns
t_{CPDH}	$\overline{\text{CASIN}}$ to $\overline{\text{CAS}}$ Delay (R/$\overline{\text{C}}$ low in Mode 4)	Figure 7b	25	39	50				ns
t_{RCC}	Column Select to Column Address Valid	Figure 7a		40	58				ns
t_{RCR}	Row Select to Row Address Valid	Figures 7a, 7b		40	58				ns
t_{RHA}	Row Address Held from Column Select	Figure 7a	10						ns
t_{CCAS}	R/$\overline{\text{C}}$ Low to $\overline{\text{CAS}}$ Low (Mode 4 Auto $\overline{\text{CAS}}$)	Figure 7a		55	75				ns
t_{DIF1}	Maximum ($t_{RPDL} - t_{RHA}$)	See Mode 4 description			13				ns
t_{DIF2}	Maximum ($t_{RCC} - t_{CPDL}$)	See Mode 4 description			13				ns
	Refresh Parameter								
t_{RC}	Refresh Cycle Period	Figure 2	100						ns
$t_{RASINL,H}$	Pulse Width of $\overline{\text{RASIN}}$ during Refresh	Figure 2	50						ns
t_{RFPDL}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay during Refresh	Figures 2, 9	35	50	70				ns
t_{RFPDH}	$\overline{\text{RASIN}}$ to $\overline{\text{RAS}}$ Delay during Refresh	Figures 2, 9	30	40	55				ns
t_{RFLCT}	$\overline{\text{RFSH}}$ Low to Counter Address Valid	$\overline{\text{CS}}$ = X, Figures 2,3,4		47	60				ns
t_{RFHRV}	$\overline{\text{RFSH}}$ High to Row Address Valid	Figures 2, 3		45	60				ns
t_{ROHNC}	$\overline{\text{RAS}}$ High to New Count Valid	Figures 2, 4		30	55				ns
t_{RLEOC}	$\overline{\text{RASIN}}$ Low to End-of-Count Low	$C_L = 50pF$, Figure 2			80				ns
t_{RHEOC}	$\overline{\text{RASIN}}$ High to End-of-Count High	$C_L = 50pF$, Figure 2			80				ns
t_{RGEOB}	RGCK Low to End-of-Burst Low	$C_L = 50pF$, Figure 4			95				ns
t_{MCEOB}	Mode Change to End-of-Burst High	$C_L = 50pF$, Figure 4			75				ns
t_{RST}	Counter Reset Pulse Width	Figure 2	70						ns
t_{CTL}	RF I/O Low to Counter Outputs All Low	Figure 2			100				ns

Switching Characteristics (Cont'd)

Symbol	Access Parameter	Conditions	8409 – 2						Units
			Min	Typ	Max	Min	Typ	Max	
$t_{RFCKL,H}$	Minimum Pulse Width of RFCK	Figure 9	100						ns
T	Period of $\overline{RAS}$ Generator Clock	Figure 3	100						ns
t_{RGCKL}	Minimum Pulse Width Low of RGCK	Figure 3	35						ns
t_{RGCKH}	Minimum Pulse Width High of RGCK	Figure 3	35						ns
t_{FRQL}	RFCK Low to Forced $\overline{RFRQ}$ Low	$C_L = 50pF$, Figure 3		20	30				ns
t_{FRQH}	RGCK Low to Forced $\overline{RFRQ}$ High	$C_L = 50pF$, Figure 3		50	75				ns
t_{RGRL}	RGCK Low to $\overline{RAS}$ Low	Figure 3	50	65	95				ns
t_{RGRH}	RGCK Low to $\overline{RAS}$ High	Figure 3	40	60	85				ns
t_{RQHRF}	$\overline{RFSH}$ Hold Time from $\overline{RFSH}$ $\overline{RQST}$ (RF I/O)	Figure 3	2T						ns
t_{RFRH}	$\overline{RFSH}$ High to $\overline{RAS}$ High (ending forced RFSH)	See Mode 1 Descrip.	55	80	110				ns
t_{RFSRG}	$\overline{RFSH}$ Low Set-up to RGCK Low (Mode 1)	See Mode 1 Descrip.	35						ns
t_{CSCT}	$\overline{CS}$ High to RFSH Counter Valid	Figure 9		55	70				ns
t_{CSRL}	$\overline{CS}$ Low to Access $\overline{RASIN}$ Low	See Mode 5 Descrip.	10						ns
t_{ZH}	$\overline{CS}$ Low to Address Output High from Hi-Z	Figures 9, 12 R1 = 3.5k, R2 = 1.5k		35	60				ns
t_{HZ}	$\overline{CS}$ High to Address Output Hi-Z from High	$C_L = 15pF$, Figures 9, 12 R2 = 1k, S1 open		20	40				ns
t_{ZL}	$\overline{CS}$ Low to Address Output Low from Hi-Z	Figures 9, 12 R1 = 3.5k, R2 = 1.5k		35	60				ns
t_{LZ}	$\overline{CS}$ High to Address Output Hi-Z from Low	$C_L = 15pF$, Figures 9, 12 R1 = 1k, S2 open		25	50				ns
t_{HZH}	$\overline{CS}$ Low to Control Output High from Hi-Z High	Figures 9, 12 R2 = 750Ω, S1 open		50	80				ns
t_{HHZ}	$\overline{CS}$ High to Control Output Hi-Z High from High	$C_L = 15pF$, Figures 9, 12 R2 = 750Ω, S1 open		40	75				ns
t_{HZL}	$\overline{CS}$ Low to Control Output Low from Hi-Z High	Figure 12, S1, S2 open		45	75				ns
t_{LHZ}	$\overline{CS}$ High to Control Output Hi-Z High from Low	$C_L = 15pF$, Figure 12, R2 = 750Ω, S1 open		50	80				ns

Input Capacitance $T_A = 25°C$ (Notes 2, 6)

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{IN}	Input Capacitance ADS, R/$\overline{C}$			8		pF
C_{IN}	Input Capacitance All Other Inputs			5		pF

Note 1: "Absolute Maximum Ratings" are the values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Electrical Characteristics" provides conditions for actual device operation.

Note 2: All typical values are for $T_A = 25°C$ and $V_{CC} = 5.0V$.

Note 3: This test is provided as a monitor of Driver output source and sink current capability. Caution should be exercised in testing this parameter. In testing these parameters, a 15Ω resistor should be placed in series with each output under test. One output should be tested at a time and test time should not exceed 1 second.

Note 4: Input pulse 0V to 3.0V, $t_R = t_F = 2.5ns$, $f = 2.5MHz$, $t_{PW} = 200ns$. Input reference point on AC measurements is 1.5V. Output reference points are 2.7V for High and 0.8V for Low.

Note 5: The load capacitance on RF I/O should not exceed 50pF.

Note 6: Applies to all DP8409 versions unless otherwise specified.

Note 7: The DP8409-2 device can only be used with memory devices that meet the t_{RAH} specification indicated.

Applications

If external control is preferred, the DP8409 may be used in Mode 0 or 4, as in Figure 6.

If basic auto access and refresh are required, then in cases where the user requires the minimum of external complexity, Modes 1 and 5 are ideal, as shown in Figure 13a. The DP843X2 is used to provide proper arbitration between memory access and refresh. This chip supplies all the necessary control signals to the processor as well as the DP8409. Furthermore, two separate $\overline{\text{CAS}}$ outputs are also included for systems using byte-writing. The refresh clock RFCK may be divided down from either RGCK using an IC counter such as the DM74LS393 or better still, the DP84300 Programmable Refresh Timer. The DP84300 can provide RFCK periods ranging from 15.4µs to 15.6µs based on the input clock of 2 to 10MHz. Figure 13b shows the general timing diagram for interfacing the DP8409 to different microprocessors using the interface controller DP843X2.

If the system is complex, requiring automatic access and refresh, burst refresh, and all-banks auto-write, then more circuitry is required to select the mode. This may be accomplished by utilizing a PAL. The PAL has two functions. One as an address comparator, so that when the desired port address occurs (programmed in the PAL), the comparator gates the data into a latch, where it is connected to the mode pins of the DP8409. Hence the mode of the DP8409 can be changed as desired with one PAL chip merely by addressing the PAL location, and then outputting data to the mode-control pins. In this manner, all the automatic modes may be selected, assigning R/C as RFCK always, and $\overline{\text{CASIN}}$ as RGCK always. The output from $\overline{\text{RF I/O}}$ may be used as End-of-Count to an interrupt, or $\overline{\text{Refresh Request}}$ to $\overline{\text{HOLD}}$ or $\overline{\text{BUS REQUEST}}$. A complex system may use Modes 5 and 1 for automatic access and refresh, Modes 3a and 7 for system initialization, and Mode 2 (auto-burst refresh) before and after DMA.

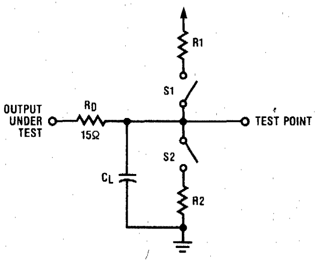


FIGURE 11. Output Load Circuit

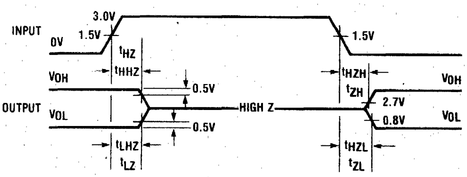
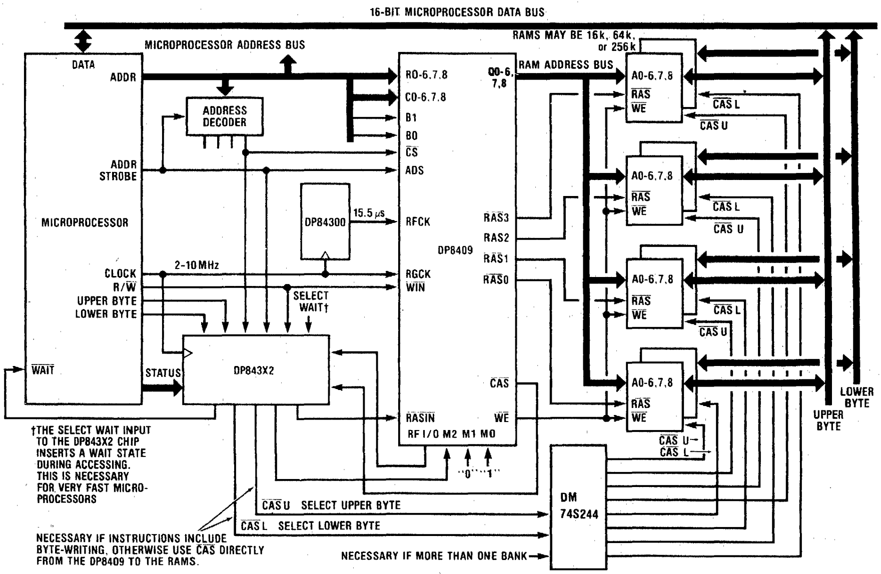


FIGURE 12. Waveform



†THE SELECT WAIT INPUT TO THE DP843X2 CHIP INSERTS A WAIT STATE DURING ACCESSING. THIS IS NECESSARY FOR VERY FAST MICROPROCESSORS

NECESSARY IF INSTRUCTIONS INCLUDE BYTE-WRITING, OTHERWISE USE $\overline{\text{CAS}}$ DIRECTLY FROM THE DP8409 TO THE RAMS.

FIGURE 13a. Connecting the DP8409 Between the 16-bit Microprocessor and Memory

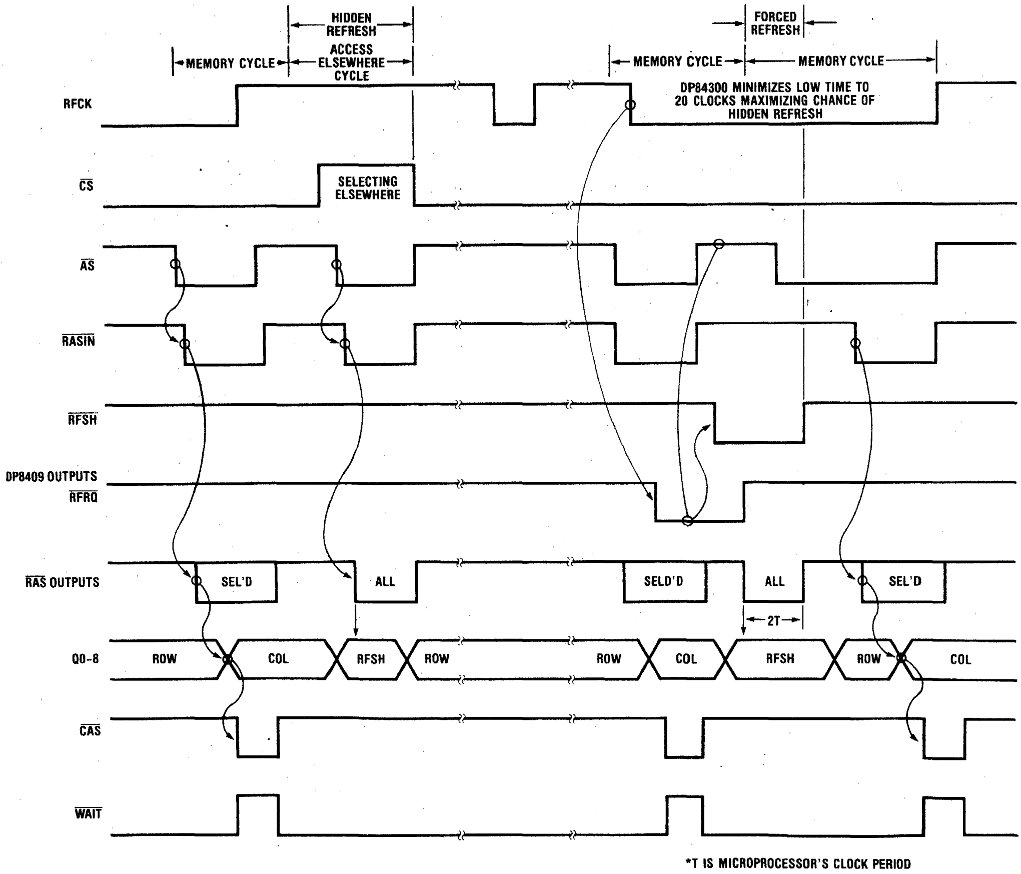
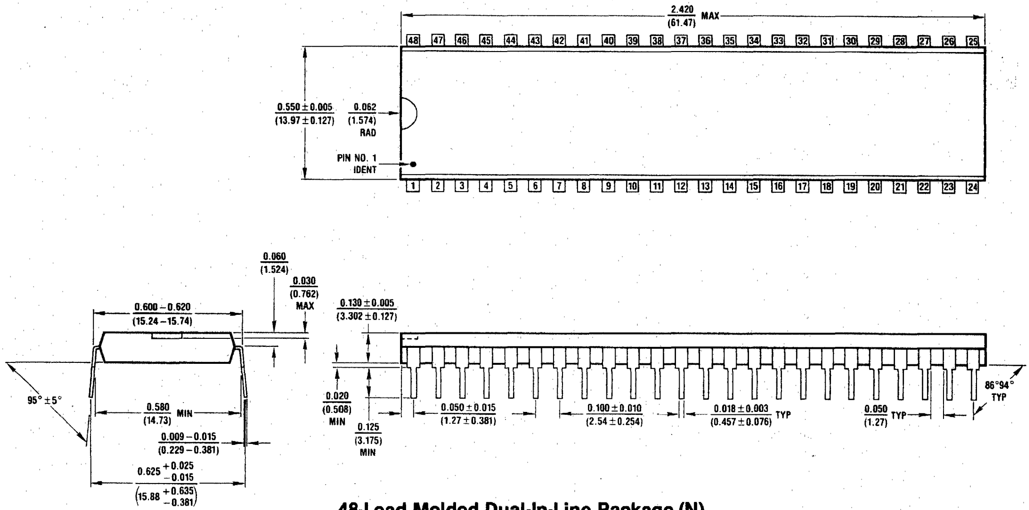
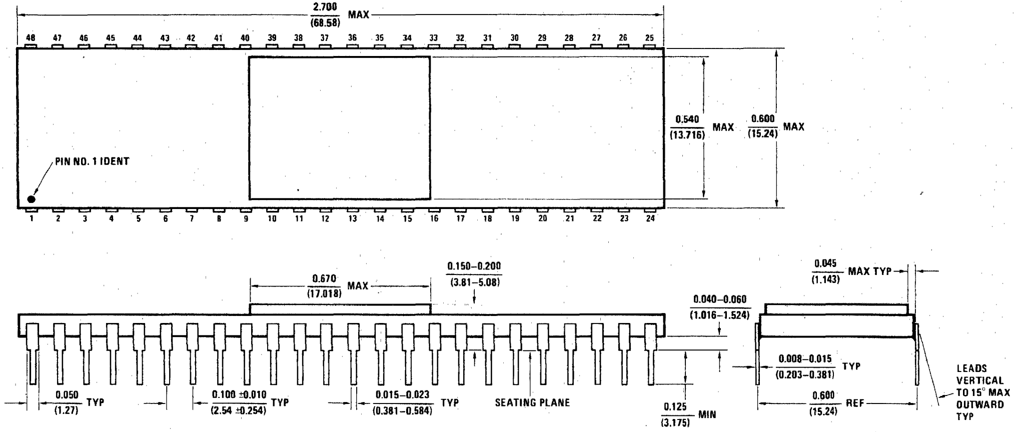


FIGURE 13b. DP8409 Auto Refresh

Physical Dimensions (inches/millimeters)



48-Lead Molded Dual-In-Line Package (N)
Order Number DP8409N
or Order Number DP8409N-3
NS Package N48A



48-Lead Dual-In-Line Package (D)
Order Number DP8409D
NS Package D48A

PRELIMINARY

National Semiconductor

DP8460 Data Separator

General Description

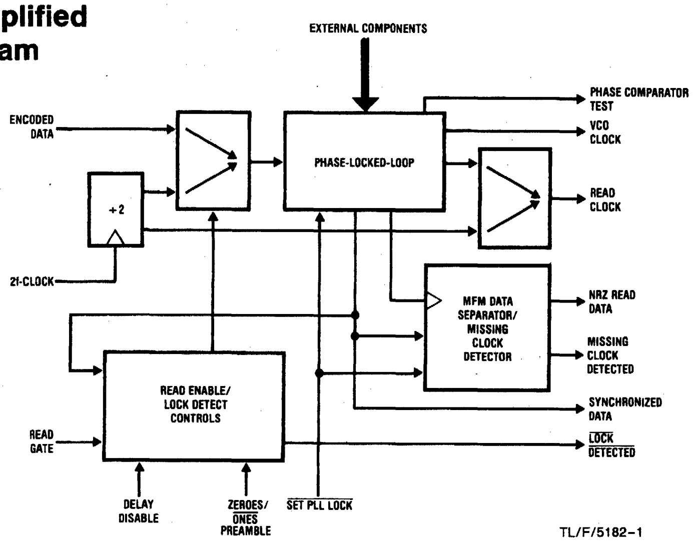
The DP8460 Data Separator is designed for application in disk drive memory systems, and depending on system requirements, may be located either in the drive or in the controller. It receives digital pulses from a pulse detector circuit (such as the DP8464 Disk Pulse Detector), if the DP8460 is situated in the drive, or from the Floppy Extension Interface if it is situated in the controller. After locking on to the frequency of these input pulses, it separates them into synchronized data and clock signals. If the input pulses are MFM encoded data, the data is made available as decoded NRZ data to be deserialized directly by a controller (such as the DP8466 Disk Data Controller). If a run-length-limited code is used, the synchronized data output is available to allow external circuitry to perform the data decoding function. All of the digital input and output signals are TTL compatible and only a single +5V supply is required. The chip is housed in a standard narrow 24-pin dual-in-line package (DIP) and is fabricated using Advanced Schottky bipolar analog and digital circuitry. This high speed I.C. process allows the chip to work with data rates up to 25Mbit/sec. There are three versions of the chip, each having a different decode window error specification. All three versions (−2, −3, −4) will operate from 2 to 25Mbit/sec, with respectively increasing window errors, as specified in the Electrical Characteristics Table.

The DP8460 features a phase-lock-loop (PLL) consisting of a pulse gate, phase comparator, charge pump, buffering amplifier, and voltage-controlled-oscillator (VCO). Pins are provided for the user to select the values of the external filtering components required for the pulse gate and amplifier, the frequency setting components required for the VCO, and two current setting resistors for the charge pump. The DP8460 has been designed to lock on to the incoming preamble data pattern within the first two bytes, using a high rate of charge pump current. Once lock-on has been achieved, the charge pump switches to a lower rate (both rates being determined by the external resistors) to maintain stability for the remainder of the read operation. At this time the READ CLOCK output switches, without glitching, from half the 2f-CLOCK frequency to half the VCO CLOCK frequency. After lock-on, with soft sectored disks, the MISSING CLOCK DETECTED output indicates when a missing clock in an address mark field occurs so the controller can align byte boundaries to begin deserialization of the incoming data.

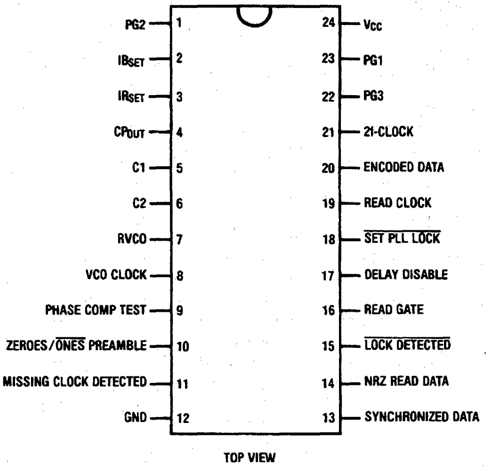
Features

- Operates at data rates up to 25Mbit/sec
- Separates MFM data into read clock and serial NRZ data
- 4 byte preamble-lock indication capability
- Preamble recognition of MFM encoded "0"s or "1"s
- User-determined PLL loop filter network
- PLL charge pump has two user-determined tracking rates
- External control of track rate switchover
- No glitch on READ CLOCK at switchover
- Synchronized data provided as an output (for RLL codes)
- ORed phase comparator outputs for monitoring bit-shift
- Missing clock detected for soft sectored disks
- Less than ½W power consumption
- Standard narrow 24-pin DIP
- Single +5V supply

DP8460 Simplified Block Diagram


EXTERNAL COMPONENTS
ENCODED DATA
÷2
2f-CLOCK
PHASE-LOCKED-LOOP
PHASE COMPARATOR TEST
VCO CLOCK
READ CLOCK
MFM DATA SEPARATOR/ MISSING CLOCK DETECTOR
NRZ READ DATA
MISSING CLOCK DETECTED
SYNCHRONIZED DATA
$\overline{\text{LOCK DETECTED}}$
READ ENABLE/ LOCK DETECT CONTROLS
READ GATE
DELAY DISABLE
$\overline{\text{ZEROES}}$/ ONES PREAMBLE
$\overline{\text{SET PLL LOCK}}$

TL/F/5182-1



TL/F/5182-2

PIN DEFINITIONS:

Power Supply

24 Vcc +5V ±5%

12 Ground

TTL Level Logic Inputs

16 READ GATE: This is an active high input signal that sets the DP8460 Data Separator into the Read Mode.

17 DELAY DISABLE: This input determines the delay from READ GATE going high to the time the DP8460 enters the Read Mode. If DELAY DISABLE is set high, this delay is within one cycle of the 2f-CLOCK signal. If DELAY DISABLE is set low, the delay is thirty two cycles of the 2f-CLOCK, as shown in *Figure 1.*

18 $\overline{\text{SET PLL LOCK}}$: This input allows the user to determine when the on-chip PLL will go into the low track rate. A high level at this input results in the PLL being in the high track rate. If this input is connected to the $\overline{\text{LOCK DETECTED}}$ output, the PLL will go into the low track rate mode immediately after lock is detected.

10 ZEROES/$\overline{\text{ONES}}$ PREAMBLE: A high level on this input enables the circuit to recognize an All Zeros data preamble. A low level results in the recognition of an All Ones data preamble.

20 ENCODED DATA: This input is connected to the output of the head amplifier/pulse-detecting network located in the disk drive. Each positive edge of the ENCODED DATA waveform identifies a change of flux on the disk. In the case of MFM encoded data, the input will be raw MFM.

21 2f-CLOCK: This is a system clock input, which is either a signal generated from the servo track (for systems utilizing servo tracks), or a signal buffered from a crystal.

TTL Level Logic Outputs

8 VCO CLOCK: This is the output of the on-chip VCO, transmitted from an Advanced Schottky-TTL buffer. It is synchronized to the MFM data output and, if needed, it can be used as the 2f-CLOCK for encoding MFM when writing to the disk.

15 $\overline{\text{LOCK DETECTED}}$: This output goes active low only after both PLL Lock has occurred and the preamble pattern has been recognized. It remains low until READ GATE goes inactive.

14 NRZ READ DATA: This is the NRZ decoded data output, whose leading edges coincide with the trailing edge of READ CLOCK.

13 SYNCHRONIZED DATA: This output is the same encoded data that is input to the chip, but is synchronous with the negative edge of the VCO CLOCK.

11 MISSING CLOCK DETECTED: When a missing clock is detected, this output will be a single pulse (of width equal to one cycle of READ CLOCK) occurring as shown in *Figure 2.*

19 READ CLOCK: This is half VCO CLOCK frequency during read mode after PLL Lock; it is half 2f-CLOCK frequency at all other times. A deglitcher is utilized to ensure that no short clock periods occur during either switchover.

9 PHASE COMP TEST: This output is the logical "OR" of the Phase Comparator outputs, and may be used for the testing of the disk media.

Analog Signals

23, 22, PG1, PG3: The external capacitors and resistor of the Pulse Gate filter are connected to these pins.

1 PG2: This is the Pulse Gate current supply.

3 IRSET: The current into the rate set pin (V_{BE}/R_{Rate}) is half the charge pump output current for the slow tracking rate.

2 IBSET: The current into the boost set pin (V_{BE}/R_{Boost}) is half the amount by which the charge pump current is increased for the high tracking rate. ($I_{HIRATE} = I_{RATE}$ Set + I_{BOOST} Set).

4 CPOUT: CHARGE PUMP OUT/BUFFER AMP IN is available for connection of external filter components, for the phase-lock-loop. In addition to being the charge pump output node, this pin is also the noninverting input to the op-amp of the Buffer Amplifier.

7 RVCO: The current into this pin determines the operating currents within the VCO.

5, 6 VCO C1, C2: An external capacitor connected across these pins sets the nominal VCO frequency.

Absolute Maximum Ratings

Supply Voltage	7V
TTL Inputs	7V
Output Voltages	7V
Input Current (CPOUT, IRSET, IBSET, RVCO)	2mA
Storage Temperature	−65°C to 150°C

Operating Conditions

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{CC}	Supply Voltage		4.75	5.00	5.25	V
T_A	Ambient Temperature		0	25	70	°C
I_{OH}	High Logic Level Output Current	V_{CO} Clock Others			−2000 −400	μA
I_{OL}	Low Logic Level Output Current	V_{CO} Clock Others			20 8	mA
f_{DATA}	Input Data Rate		2.0		25	Mbit/sec
t_{WCK}	Width of 2f-CLOCK, High or Low		10			ns
t_{WPD}	Width of ENCODED DATA Pulse, High or Low (Note 2)		0.25t			ns
V_{IH}	High Logic Level Input Voltage		2			V
V_{IL}	Low Logic Level Input Voltage				0.8	V

DC Electrical Characteristics Over Recommended Operating Temperature Range

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IC}	Input Clamp Voltage	V_{CC} = Min., $I_I = -18mA$			−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min., I_{OH} = Max.	V_{CC}−2V	V_{CC}−1.6V		V
V_{OL}	Low Level Output Voltage	V_{CC} = Min., I_{OL} = Max.			0.5	V
I_{IH}	High Level Input Current	V_{CC} = Max., V_I = 2.7V			20	μA
I_{IL}	Low Level Input Current	V_{CC} = Max., V_I = 0.4V			−200	μA
I_O	Output Drive Current	V_{CC} = Max., V_O = 2.125V[1]	−30		−110	mA
I_{CC}	Supply Current	V_{CC} = Max.			100	mA
I_{OUT}	Charge Pump Output Current	$I_{RSET} = V_{BE}/R_{RATE}$ $I_{BSET} = V_{BE}/R_{BOOST}$	−10% −10%	$2 \times I_{RSET}$ $2 \times (I_{RSET}+I_{BSET})$	+10% +10%	mA

[1] This value has been chosen to produce a current that closely approximates one-half of the true short-circuit output current, I_{OS}.

[2] t is defined as the period of the encoded data

AC Electrical Characteristics Over Recommended V_{CC} and Operating Temperature Range.

(All Parts unless stated otherwise) ($t_R = t_F = 2.0ns$, $V_{IH} = 3.0V$, $V_{IL} = 0V$)

Symbol	Parameter	Min	Typ	Max	Units
t_{READ}	Positive READ CLOCK transitions from READ GATE set active until PLL Lock sequence begins (DELAY DISABLE low)		16	17	—
t_{READ}	Positive READ CLOCK transitions from READ GATE set active until PLL Lock sequence begins (DELAY DISABLE high)		1	1	—
$t_{DECODE\ NRZ}$	Number of READ CLOCK cycles required to output each decoded MFM data bit[4]	—	2	3	T-clock
$t_{TRANSMIT\ MFM}$	Positive READ CLOCK transitions required to transmit input MFM to output	1	2	3	—
$t_{READ\ ABORT}$	Number of READ CLOCK cycles after READ GATE set low to read operation abort			2	T-clock
t_{WINDOW}	Variance of center of decode window from nominal[7] DP8460-2 DP8460-3 DP8460-4			$2+0.6\%\tau$ $3+0.8\%\tau$ $4+1.0\%\tau$	ns
$\phi_{LINEARITY}$	Phase range for charge pump output linearity[2]	$-\pi$		$+\pi$	Radians
K_1	Phase Comparator — Charge Pump gain constant[5]		$\frac{V_{BE}}{\pi R}$		Amps/rad
$V_{CONTROL}$	Charge pump output voltage swing from nominal		±100		mV
K_{VCO} (=$A \times K_2$)	VCO gain constant (ω_{VCO} = VCO center frequency in rad/s)[6]	$\frac{1.4\omega_C}{V_{BE}}$	$\frac{1.6\omega_C}{V_{BE}}$	$\frac{1.8\omega_C}{V_{BE}}$	rad/sec. V
f_{VCO}	VCO center frequency variation over temperature and V_{CC}	−5		+5	%
$f_{MAX\ VCO}$	VCO maximum frequency	70			MHz
t_{HOLD}	Time READ CLOCK is held low during changeover after lock detection has occurred[3]			1½	T-clock
$t_{MFMSKEW}$	Output skew between VCO clock and Synchronized Data				ns
$t_{NRZSKEW}$	Output skew between READ CLOCK, NRZ READ DATA and MISSING CLOCK DETECTED				ns

[1] A sample calculation of frequency variation vs. control voltage: $V_{IN} = \pm 0.2V$; $K_{VCO} = \frac{\omega_{OUT}}{V_{IN}} = \frac{0.4\,\omega_C}{0.2V} = \frac{2.0\,\omega_C}{V}\ \frac{(rad/sec)}{(volt)}$

[2] $-\pi$ to $+\pi$ with respect to 2f VCO CLOCK

[3] T-clock is defined as the time required for one period of the READ CLOCK to occur.

[4] This number remains fixed after PLL Lock occurs.

[5] With respect to VCO CLOCK; $I_{PUMP\ OUT} = 2\,I_{SET}$

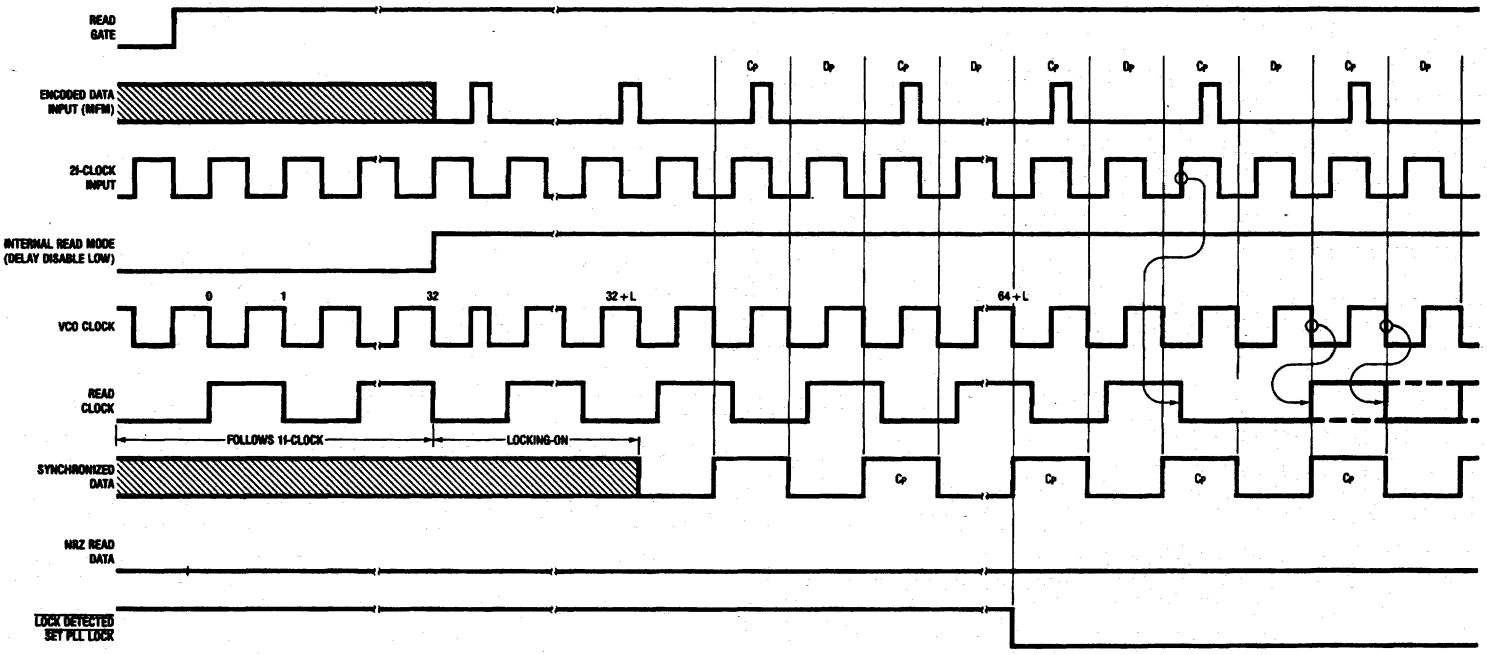
$$I_{SET} = \frac{V_{BE}}{R_{SET}}$$

[6] Although specified as the VCO gain constant, this is the gain from the Buffer Amplifier input to the VCO output.

[7] τ is defined as the period of the incoming data stream

External Component Selection (All Parts)

Symbol	Component	Min	Typ	Max	Unit
R_{VCO}	VCO Frequency Setting Resistor	990		1010	Ω
C_{VCO}	VCO Frequency Setting Capacitor	15			pF
R_{LF}	Loop Filter Resistor	TBD		TBD	Ω
C_{LF1}	Loop Filter Capacitor 1	20		TBD	pF
C_{LF2}	Loop Filter Capacitor 2	20		TBD	pF
R_{RATE}	Charge Pump I_{RATE} Set Resistor	1.2		6.5	kΩ
R_{BOOST}	Charge Pump (High Rate) I_{BOOST} Resistor	0.4		∞	kΩ
R_{PG2}	Delay Time Setting Resistor	0		150	kΩ
R_{PG1}	Pulse Gate Resistor	TBD		TBD	kΩ
C_{PG1}	Pulse Gate Capacitor C1	20		TBD	pF
C_{PG2}	Pulse Gate Capacitor C2	20		TBD	pF
C_R	I_{RATE} Bypass Capacitor	1000			pF
C_B	I_{BOOST} Bypass Capacitor	1000			pF



TL/F/5182-3

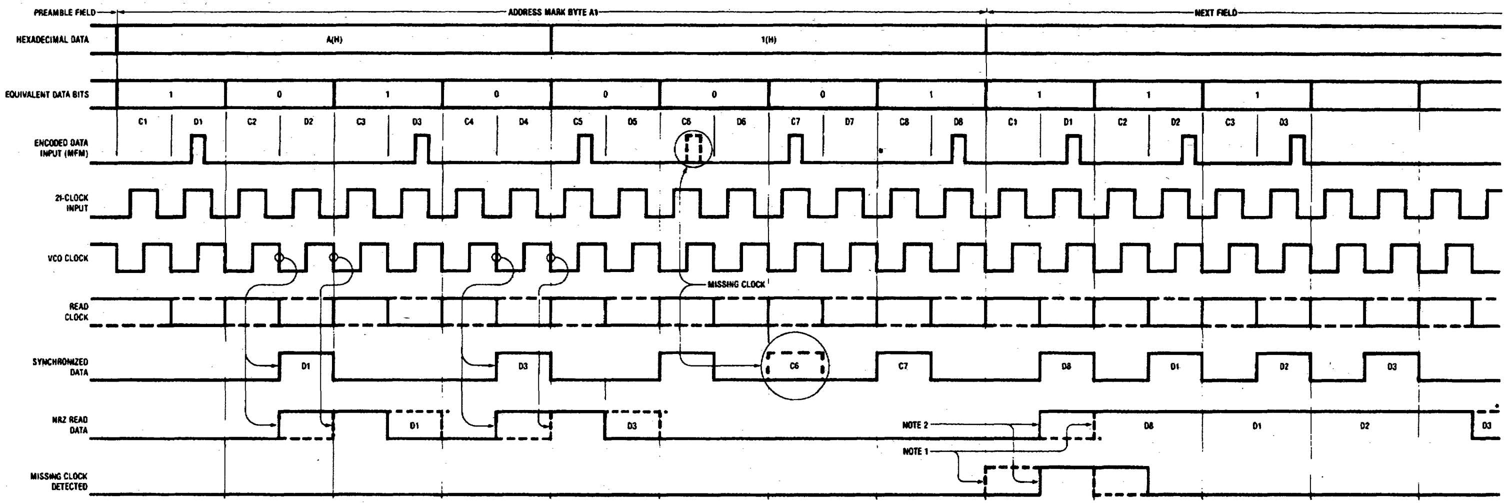
C_P, D_P = preamble clock and preamble data bits respectively.

L = Number of 2f-clock cycles required for VCO to lock (typically ≈ 20 2f-clock cycles), but determined by external component values

At 32 + L, VCO has just locked.

At 64 + L, circuit has confirmed lock (has been in lock for 16 MFM clock bits). This sequence shows the MFM preamble pattern.

FIGURE 1. Lock-on Sequence Waveform Diagram



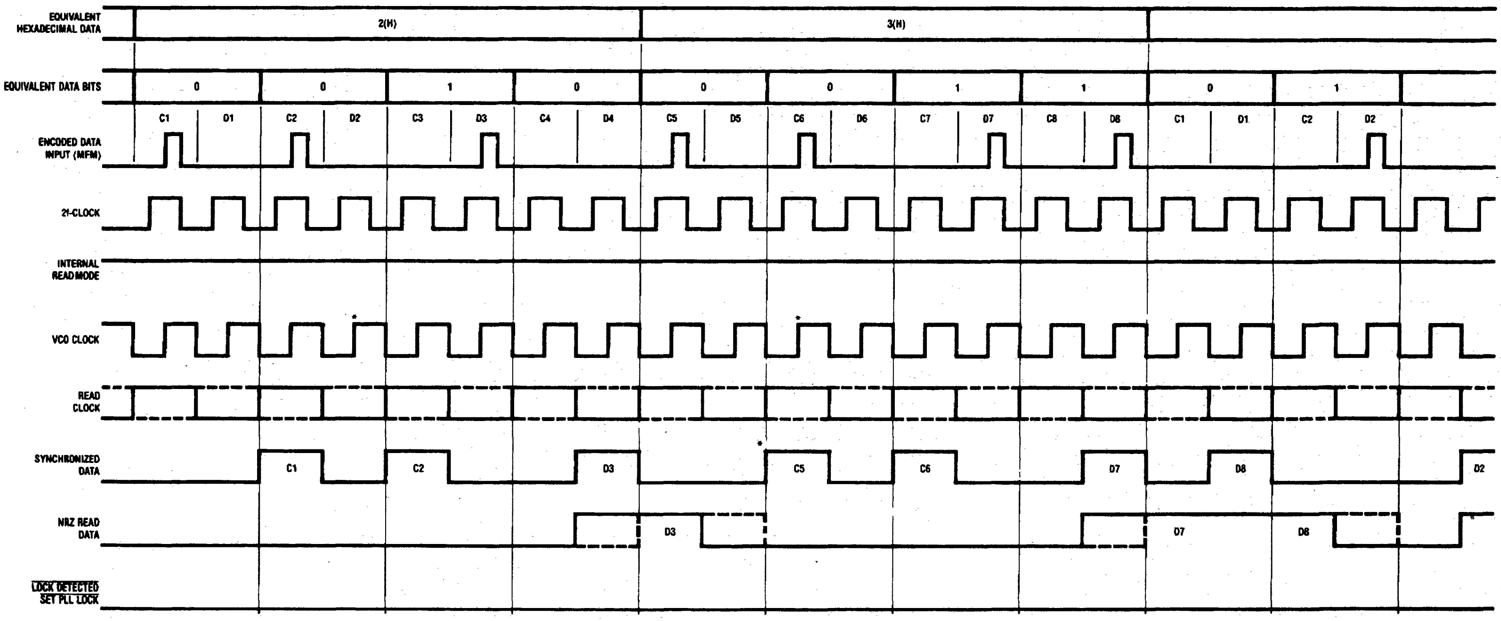
TL/F/5182-4

* READ CLOCK and NRZ READ DATA may be delayed by one VCO clock period depending on the phase of the internal clock at activation of READ GATE input.

① MISSING CLOCK DETECTED is one READ CLOCK period ahead of the chip issuing D8 on the NRZ READ DATA output when READ CLOCK is delayed by one VCO clock period

② MISSING CLOCK DETECTED is synchronous with the chip issuing D8 on the NRZ READ DATA Output when READ CLOCK is not delayed

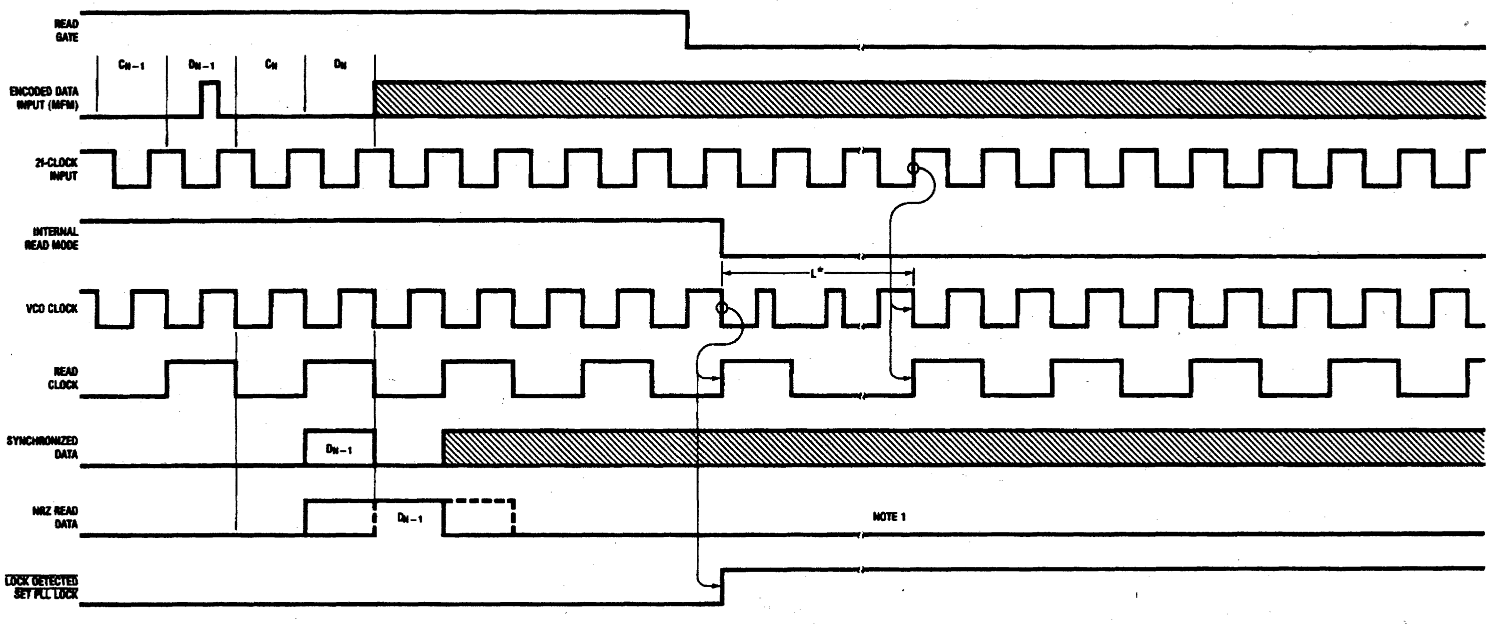
FIGURE 2. Missing Clock Detection Waveform Diagram



TL/F/5182-5

* READ CLOCK and NRZ READ DATA may be delayed by one VCO clock period with respect to Synchronized Data depending on the phase of the internal clock at activation of READ GATE input.

FIGURE 3. Locked-on Waveform Diagram



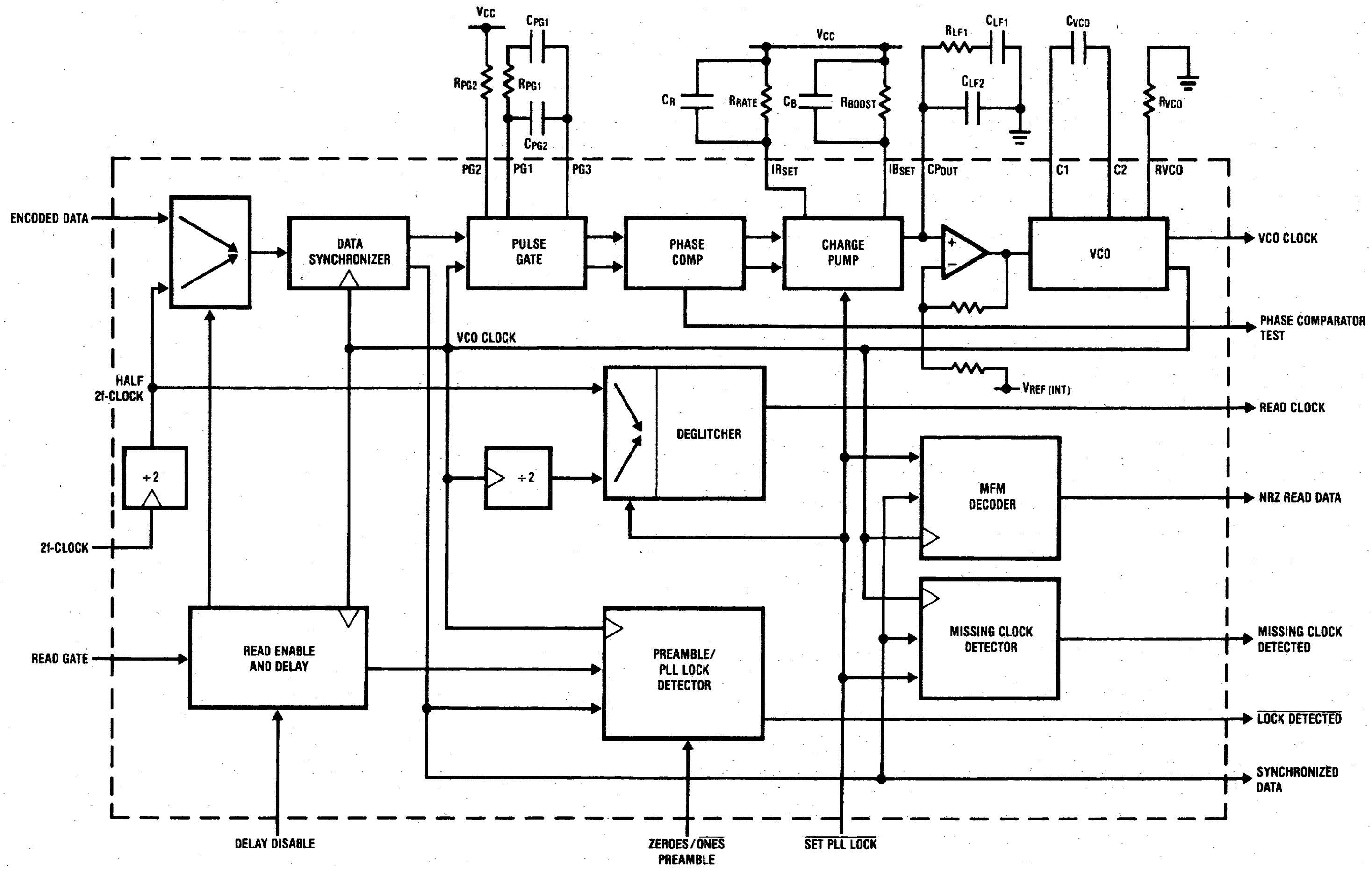
TL/F/5182-6

* L indicates the number of cycles required for the VCO to lock to the 2f-CLOCK

NOTE 1: READ GATE going low will always result in NRZ READ DATA going low regardless of the state of the last bit

FIGURE 4. Lock-Ending Sequence Waveform Diagram

DP8460 Detailed Block Diagram



TL/F/5182-7

CIRCUIT OPERATION

When the READ GATE input goes high, the DP8460 Data Separator enters the read mode after a period determined by the state of the DELAY DISABLE pin. This may be either one or thirty two 2f-CLOCK cycles. Referring to *Figure 1*, once in the read mode, the phase-locked-loop reference signal is switched from 2f-CLOCK input to the ENCODED DATA input. The PLL, initially in the high-tracking rate mode, then attempts to lock onto the incoming encoded data stream. By careful selection of the loop filter components, this can be within 2 bytes. Preamble pattern recognition then can begin. As soon as two bytes of the selected preamble are detected, (the selection is determined by the ZEROES/ONES PREAMBLE pin) the LOCK DETECTED output goes low. In a typical MFM disk drive application, the LOCK DETECTED output is directly connected to the SET PLL LOCK input. With this connection, track rate selection, clock output switchover, and data output enabling will occur after four consecutive preamble bytes have been fed into the chip, from the time the read mode began.

A low level on the SET PLL LOCK causes the PLL Charge Pump to switch from the high to low tracking rate. At the same time the source of the READ CLOCK signal is switched from half the frequency of the 2f-CLOCK to half the VCO clock. The MFM decoder also becomes enabled and begins to output decoded NRZ data. If the preamble is being decoded, and it is a zeroes data preamble, the NRZ READ DATA output will remain low until the end of the preamble. It will then output NRZ data some 2f-CLOCK periods after the preamble field has ended, as shown in *Figures 2* and *3*.

Figure 4 shows the sequence when READ GATE goes low, signifying the end of a read operation. The PLL reference signal is switched back to half the 2f-CLOCK and the LOCK DETECTED output (and therefore the SET PLL LOCK input) goes high. The PLL then returns to the high tracking rate, and the output signals return to their initial conditions.

CIRCUIT DESCRIPTION

1. Read Enable and Delay: If the DELAY DISABLE input is connected low, then thirty two 2f-CLOCK cycles after READ GATE goes active, the DP8460 will go into the read mode. If the DELAY DISABLE input is connected high, the chip will go into the read mode one 2f-CLOCK cycle after READ GATE goes active. This feature allows the user to choose the time at which the PLL Lock Sequence begins and thus accommodates systems with short preambles.

2. Pulse Gate, including Input Multiplexer and Data Synchronizer: The Input Multiplexer selects the input to the phase-lock-loop. While the chip is in the bypassed mode, the PLL is locked on half the 2f-CLOCK frequency, but in the read mode, the Input Multiplexer switches to the ENCODED DATA signal. The VCO CLOCK then begins to synchronize with the ENCODED DATA signal. The Pulse Gate allows a reference signal from the VCO into the Phase Comparator only when a ENCODED DATA bit is valid. The Pulse Gate utilizes a scheme which delays the incoming data by one-half the period of the 2f-CLOCK. This optimizes the position of the decode window and allows input jitter up to ± half the 2f-CLOCK period, assuming no error in the decode window position. The decode window error can be determined from the specification in the Electrical Characteristics Table.

3. Phase Comparator: The Phase Comparator receives its inputs from the Pulse Gate, and is edge-triggered from these inputs to provide charge-up and charge-down outputs.

4. Charge Pump: The high speed charge pump consists of a switchable constant current source and sink. The charge pump constant current is set by connecting external resistors to Vcc from the charge current rate set (IRSET) and current boost set (IBSET) pins. Before lock is indicated, the PLL is in the fast tracking rate and both resistors determine the current. In the slow tracking rate after lock-on, only the IRSET resistor determines the charge pump current. The output of the charge pump feeds into external filter components and the Buffer Amplifier.

5. Buffer Amplifier: The input of the Buffer Amplifier is internally connected to the charge pump's constant current source/sink output as well as the external Loop Filter components. The Buffer Amplifier is configured as a high input impedance amplifier which allows for the connection of external PLL filter components to the Charge Pump output pin CPOUT. The output of the Buffer Amplifier is internally connected to the VCO control input.

6. VCO: The Voltage-Controlled-Oscillator requires a resistor from the RVCO pin to ground and a capacitor between pins C1 and C2, to set the center frequency. The VCO frequency can be varied from nominal by approximately ±20%, as determined by its control input voltage.

7. PLL Lock-on/Preamble Pattern Detector: To recognize preamble, the preamble pattern from the disk must consist exclusively of either data bit zeroes (encoded into ..10.. MFM clock pulses) when the ZEROES/ONES PREAMBLE pin is set high, or data bit ones (encoded into ..01.. MFM clock pulses) when set low. The preamble pattern must be long enough to allow the PLL to lock, and subsequently for the Preamble Pattern Detector circuit to detect two complete bytes.

Once the chip is in the read mode, the VCO proceeds to lock on to the incoming data stream. The Preamble Pattern Detector then searches for a continuous pattern of 10101010101010101010101010101010 (16 consecutive pulses at the data rate) to indicate lock has been achieved. The LOCK DETECTED output then goes low.

Any deviation from the above-mentioned one-zero pattern at any time before PLL Lock is detected will reset the PLL Lock Detector. The lock detection procedure will then start again.

8. MFM Decoder: The MFM Decoder receives synchronized MFM data from the Pulse Gate and converts it to NRZ READ DATA. For run-length-limited codes the MFM Decoder and Missing Clock Detector will not be used.

9. Missing Clock Detector: This block is only required for soft-sectored drives, and is used to detect a missing clock violation of the MFM pattern. The missing clock is inserted when writing to soft-sectored disks to indicate the location of the Address Mark in both the ID and the Data fields of each sector. Once PLL Lock has been indicated, the Missing Clock Detector circuit is enabled. MISSING CLOCK DETECTED will go active only if the incoming data pattern contains one suppressed clock bit framed by two adjacent clock bits. The output signal goes high for one cycle of READ CLOCK.

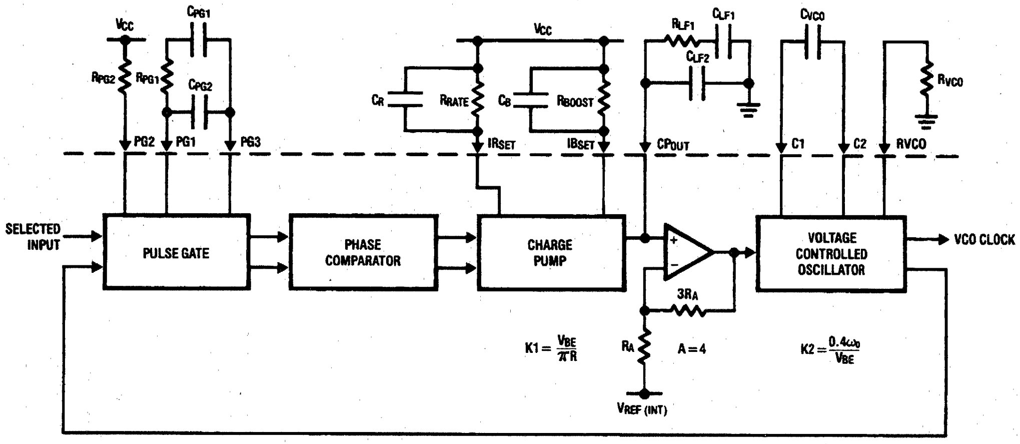
10. Clock Multiplexer and Deglitcher: When the $\overline{\text{SET PLL LOCK}}$ input changes state this circuit switches the source of the READ CLOCK signal between the half 2f-CLOCK frequency and the half VCO CLOCK frequency. A deglitcher circuit is utilized to ensure that no short clock periods occur during either switchover.

BIT-JITTER TOLERANCE

The three options of the DP8460, the −4, −3 and −2 offer decreasing window errors (respectively) so that the parts may be selected for different data rates (up to 25Mbit/sec). The −4 part will be used in most low data rate applications. As an example, at the 5Mbit/sec data rate of most 5¼ inch drives, T = 200ns so that from the Electrical Characteristics Table, t_{WINDOW} = (4 + (1% of 200ns)) or 6ns. The chip therefore contributes up to 6ns of window error, out of the total allowable error of 50ns (half the 2f-clock period of 100ns). This allows the disk drive to have a margin of 44ns of jitter on the transition position before an error will occur.

ANALOG CONNECTIONS TO THE DP8460

External passive components are required for the Pulse Gate, Charge Pump, Loop Filter and VCO as shown in *Figure 5.* The information provided here is for guidelines only. The user should select values according to his own system requirements. Phase-Locked Loops are complex circuits that require detailed knowledge of the specific system. Factors such as loop gain, stability, response to change of signal, lock-on time, etc are all determined by the external components. In many disk systems these factors are critical, and National Semiconductor recommends the designer be knowledgeable of phase-locked-loops, or seek the advice of an expert. Inaccurate design will probably result in excessive disk error rates. The phase-locked-loop in the DP8460 has many advantages over all but the most sophisticated discrete designs, and if the component values are selected correctly, it will offer significant performance advantages. This should result in a reduction of disk error rates over equivalent discrete designs.



TL/F/5182-8

FIGURE 5. Phase-Locked-Loop Section

Pulse Gate

There are four external components connected to the Pulse Gate as shown in *Figure 6* with the associated internal components. The values of R_{PG1}, R_{PG2}, C_{PG1}, and C_{PG2} are dependent on the data rate. R_{PG1} is proportional to the data rate, while R_{PG2}, C_{PG1} and C_{PG2} are inversely proportional. Table I shows component values for the data rates given. For other data rates, use the equations R_{PG1} = (TBD × f_{VCO}) kΩ, R_{PG2} = ((TDB / f_{VCO}) − 0.89) kΩ, C_{PG1} = (TBD / f_{VCO}) pF and C_{PG2} = (TBD / f_{VCO}) pF, where f_{VCO} is the VCO frequency in MHz. As an example, at 5Mbits/sec data rate, f_{VCO} = 10 MHz. This produces R_{PG1} = ... kΩ, R_{PG2} = ... kΩ, C_{PG1} = ... pF and C_{PG2} = ... pF. Components with 5% tolerance will suffice.

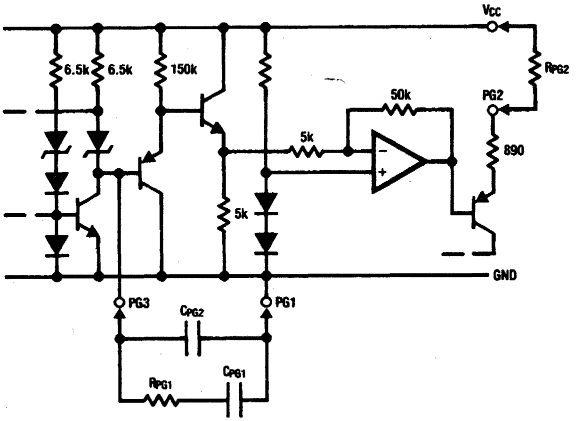
Data Rate	R_{PG2}	R_{PG1}	C_{PG1}	C_{PG2}
2Mbit/sec	16kΩ			
5Mbit/sec	4.7kΩ			
10Mbit/sec	1.9kΩ			
15Mbit/sec	750Ω			
20Mbit/sec	300Ω			
25Mbit/sec	0			

TABLE I. Pulse Gate Component Selection Chart

Charge Pump

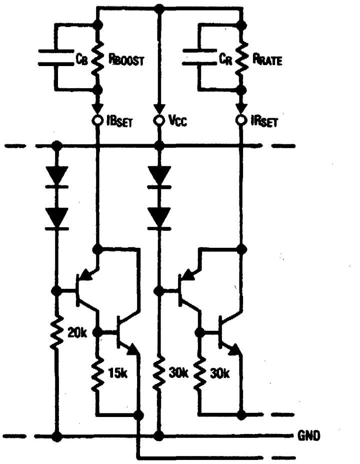
Resistors R_{RATE} and R_{BOOST} determine the charge pump current. The Charge Pump bidirectional output current is approximately (within ±10%) twice the input current. In the high tracking rate with $\overline{\text{SET PLL LOCK}}$ high, the input current is I_{BSET} + I_{RSET}, ie, the sum of the currents through R_{BOOST} and R_{RATE} from Vcc. In the low tracking rate, with $\overline{\text{SET PLL LOCK}}$ low, this input current is I_{RSET} only.

A recommended approach would be to select R_{RATE} first. The External Component Limits table allows R_{RATE} to be 1.2kΩ to 6.5kΩ, so for simplicity select R_{RATE} = 3.3kΩ. A typical loop gain change of 4:1 for high to low tracking rate would require R_{BOOST} = R_{RATE} /3 or 1.1kΩ. Referring to *Figure 7* the input current is effectively V_{BE} / R_{RATE} in the low tracking rate, where V_{BE} is an internal voltage. This means that the current into or out of the loop filter is approximately 2 V_{BE} / R_{RATE}, or in this example approximately 0.4mA. Note that although it would seem the overall gain is dependent on V_{BE}, this is not the case. The loop gain is altered internally by an amount inversely proportional to V_{BE}, as detailed in the section on the Loop Filter. This means that as V_{BE} varies with temperature or device spread, the gain will remain constant for a particular fixed values of R_{RATE} and R_{BOOST}. This alleviates the need for potentiometers to select values for each device. The tolerance required for these two resistors will depend on the total loop gain tolerance allowed, but 5% would be typical. Also Vcc by-pass capacitors are required for these two resistors. A value of 1000pF is suitable for each.



TL/F/5182-9

FIGURE 6. Pulse Gate Controls



TL/F/5182-10

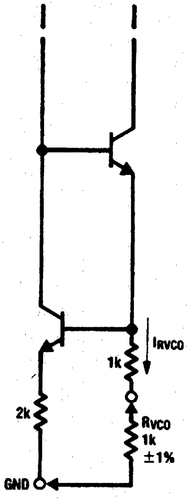
FIGURE 7. I_{RATE} Set and I_{BOOST} Set

VCO

The value of R_{VCO} is fixed at 1 kΩ ±1% in the External Component Limits table. This requires a resistor more accurate than 1% to allow for temperature variations. *Figure 8* shows how R_{VCO} is connected to the internal components of the chip. This value was fixed at 1kΩ to set the VCO operating current such that optimum performance of the VCO is obtained for production device spreads. This means fixed value components will be adequate to set the VCO center frequency for production runs. The value of C_{VCO} can therefore be determined from the VCO frequency f_{VCO}, using the equation: $C_{VCO}= 1 / (R_{VCO}) (f_{VCO})$ where f_{VCO} is twice the input data rate. As an example, for a 5Mbit/sec data rate, f_{VCO} = 10Mhz, requiring that C_{VCO} = 100pF. The capacitor tolerance also should be better than 1%. The capacitor is connected to internal circuitry of the chip as shown in *Figure 9*.

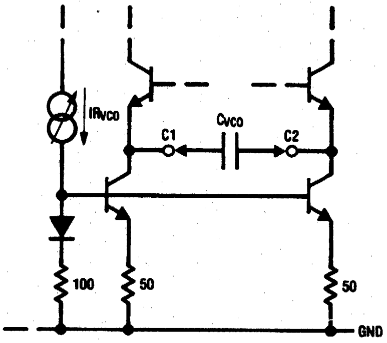
This equation does not cover the whole range of data rates. As the data rate increases and C_{VCO} gets smaller, the effects of unwanted parasitic capacitances influence the frequency. As a guide the graph of *Figure 10* shows approximately the value of C_{VCO} for a given data rate.

The center frequency may be checked by applying pulses at the ENCODED DATA input with READ GATE set high. The input frequency should be varied above and below the chosen center frequency until the VCO stops tracking. Typically this will be 20% either side of the center frequency.



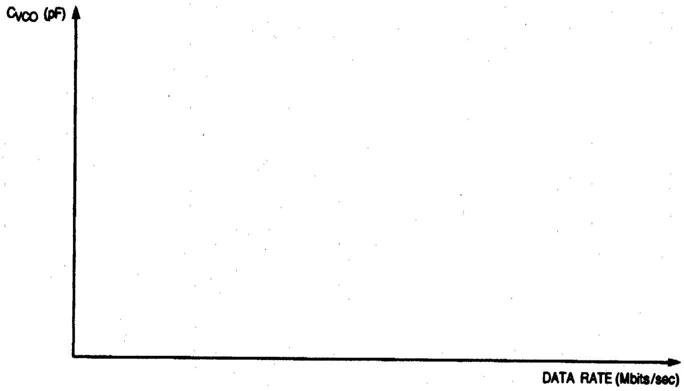
TL/F/5182-11

FIGURE 8. VCO Current Setting Resistor



TL/F/5182-12

FIGURE 9. VCO Capacitor



TL/F/5182-13

FIGURE 10. VCO Capacitor Value for Disk Data Rates

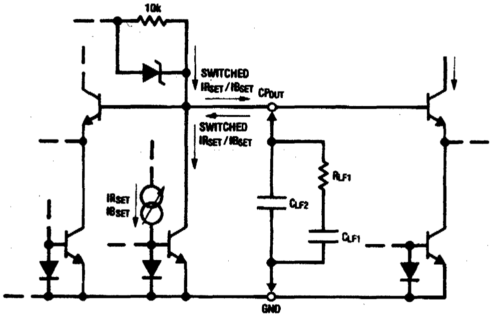
Loop Filter

The input current into the Buffer Amplifier is offset by a matched current out of the Charge Pump, and even so is much less than the switching current in or out of the Charge Pump. It can therefore be assumed that all the Charge Pump switching current goes into the Loop Filter components R_{LF1} and C_{LF1} and C_{LF2}. The tolerance of these components should be the same as R_{RATE} and R_{BOOST}, and will determine the overall loop gain variation. The three components connected to the Charge Pump output are shown in *Figure 11*. Note the return current goes to analog GND, which should be electrically very close to the GND pin itself.

The value of capacitor C_{LF1} basically determines loop stability-the larger the value the longer the loop takes to respond to an input change. If C_{LF1} is too small, the loop will track any jitter on the ENCODED DATA input and the VCO output will follow this jitter, which is undesirable. The value of C_{LF1} should therefore be large enough so that the PLL is fairly immune to phase jitter but not large enough that the loop won't respond to longer term data rate changes that occur on the disk drive.

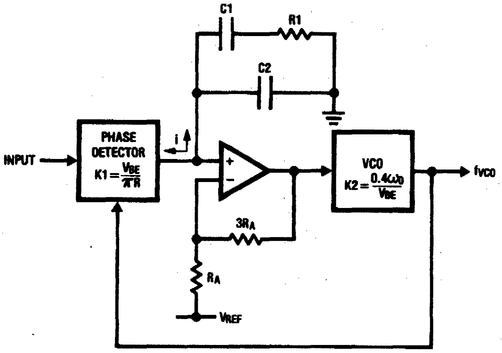
The damping resistor R_{LF1} is required to damp any oscillation on the VCO input that would otherwise occur due to step function changes on the input. A value of R_{LF1} that would give a phase margin of around 45 degrees would be a reasonable starting point.

The main function of the capacitor C_{LF2} is to integrate the effects of the VCO frequency on the VCO input voltage. Typically its value will be about one tenth of C_{LF1}.



TL/F/5182-14

FIGURE 11. Charge Pump Out



TL/F/5182-15

FIGURE 12. Loop Response Components

Figure 12 shows the relevant phase-locked-loop blocks that determine system response, namely the Phase Detector, Filter/Buffer Amplifier, and VCO. The Phase Detector (Phase Comparator and Charge Pump) produces an aggregate output current i which is proportional to the phase difference between the input signal and the VCO signal. The constant (K_1) is $V_{BE}/\pi R$ amps per radian. R is either R_{RATE} or $R_{RATE} \parallel R_{BOOST}$. This aggregate current feeds into or out of the filter impedance (Z), producing a voltage to the VCO that regulates the VCO frequency. The VCO gain constant is 0.4 ω_{VCO}/V_{BE} radians per second per volt. Under steady state conditions, i will be zero and there will be no phase difference between the input signal and the VCO. Any change of input signal will produce a change in VCO frequency that is determined by the loop gain equation. This equation is determined from the gain constants K_1, A and K_2 and the filter v/i response.

The impedance Z of the filter is:

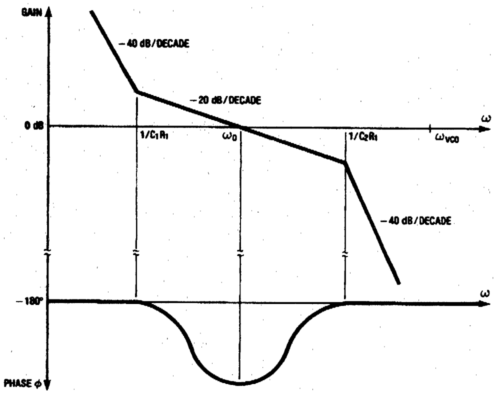
$$\frac{1}{sC_2} \parallel \left(\frac{1}{sC_1} + R_1\right) = \frac{1 + sC_1R_1}{sC_1\left(1 + \frac{C_2}{C_1} + sC_2R_1\right)}$$

If $C_2 \ll C_1$ then the impedance Z approximates to:

$$\frac{1 + sC_1R_1}{sC_1\ (1 + sC_2R_1)}$$

The overall loop gain is then $G(s) = \frac{K_1AK_2}{s} \times \frac{1 + sC_1R_1}{sC_1(1 + sC_2R_1)}$

The desired Bode plot of gain and phase is shown in *Figure 13*, with −20dB/decade slope at ω_O for stability at unity gain.



TL/F/5182–16

FIGURE 13. Bode Plot of Loop Response

If the point of inflexion of the phase curve is at ω_O, (the loop natural frequency and therefore the closed loop bandwidth), then it can be shown that for a phase margin ϕ,

$$C_2R_1 = \frac{1 - \sin\phi}{\omega_O \cos\phi} \text{ sec}$$

$$C_1R_1 = \frac{1}{\omega_O{}^2 C_2R_1} \text{sec}$$

$$\text{and } C_1 = \frac{K_1 A K_2}{\omega_O{}^2} \times \frac{1 + \omega_O C_1 R_1}{1 + \omega_O C_2 R_1} \text{ F}$$

As an example, if we want the PLL to lock-on within two bytes of preamble in the high tracking rate mode, and the disk data rate is 5Mbits/sec, or one bit every 200ns.

(Thus f_{VCO} = 10MHz)

time to lock = 16 × 0.2μs = 3.2μs

Closed loop bandwidth $f_O >$ (0.3/3.2) MHz or about 100kHz

(the factor 0.3 is a rule of thumb guideline derived from the product of rise time and bandwidth).

Select a bandwidth f_O = 200kHz so that $\omega_O = 2\pi \times 200\text{kHz}$ (giving a ratio of $f_{VCO} / f_O = 50$)

Select a phase margin ϕ between 30° and 70° for stability. Choose ϕ = 45° for optimum response

$$\text{Then } C_2R_1 = \frac{(1 - \sin 45°)}{2\pi \times 200 \times 10^3 \times \cos 45°} = 0.33 \times 10^{-6} \text{ sec}$$

$$\text{and } C_1R_1 = \frac{1}{(2\pi \times 200 \times 10^3)^2 \times 0.33 \times 10^{-6}} = 1.92 \times 10^{-6} \text{ sec}$$

To determine C1, we need to know K_1, A and K_2.

$$K_1 = \frac{V_{BE}}{\pi R} \text{ amps per radian}$$

In the high tracking rate, R = 1.1k ‖ 3.3k = 825Ω for our example from the Charge Pump calculations

$$\text{So } K_1 = \frac{V_{BE}}{\pi \times 825} \text{ amps/radian}$$

the Buffer Amplifier gain A is internally set to 4.0

$$K_2 = \frac{0.4\omega_{VCO}}{V_{BE}} \text{ radians per sec per volt}$$

$$K_2 = \frac{0.4 \times 2\pi \times 10^7}{V_{BE}} \text{ radians per sec per volt}$$

$$\text{so } C_1 = \frac{V_{BE}}{\pi \times 825} \times 4.0 \times \frac{0.4 \times 2\pi \times 10^7}{V_{BE} \times (2\pi \times 200 \times 10^3)^2} \times$$

$$\frac{1 + (2\pi \times 200 \times 10^3 \times 1.92 \times 10^{-6})}{1 + (2\pi \times 200 \times 10^3 \times 0.33 \times 10^{-6})} \text{ F}$$

The V_{BE} in the K_1 equation cancels with the V_{BE} in the K_2 equation provided good matching is maintained on the DP8460.

Thus $C_1 = 5.923 \times 10^{-8}$F. Select C_1 to be 0.056μF

$$\therefore R_1 = \frac{1.92 \times 10^{-6}}{56000 \times 10^{-12}} \Omega = 34.28\Omega. \text{ Select } R_1 = 33\Omega$$

$$\therefore C_2 = \frac{0.33 \times 10^{-6}}{33} \text{F} = 10^{-8}\text{F} = 0.01\mu\text{F. Select } C_2 = 0.01\mu\text{F}$$

The calculated values are only a guide, the user should then empirically test the loop and determine stability, lock-on time, jitter tolerance, etc.

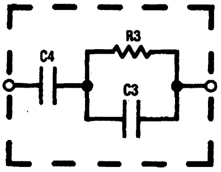
Note that capacitor C_2 affects the amount by which the Charge Pump switching current affects the filter voltage. Obviously as C_2 is increased in value ripple will decrease, but the closer the −40dB/decade slope gets to ω_o on the Bode plot the more unstable the loop will be. Thus if C_2 is made too large the loop will oscillate.

Resistor R_1 determines where the low-frequency end −40dB/decade slope changes into the −20dB/decade slope. The wider the −20dB/decade slope is around unity gain, the more stable the loop becomes. If R_1 is too large it will reduce the impact of C_1, while too small a value will increase instability. The capacitor C_1 strongly effects the response of the loop. Too high a value will slow down the response time, but make it less prone to jitter or frequency shift whereas too low a value will improve response time while tending to react to jitter.

Other filter combinations may be used, other than R_{LF1} in series with C_{LF1}, all in parallel with C_{LF2}. For example the filter shown in *Figure 14* will also perform similarly, and in fact for some systems it will yield superior performance.

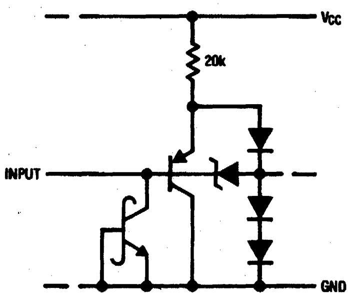
DIGITAL CONNECTIONS TO THE DP8460

Figure 17 shows a connection diagram for the DP8460 in a typical application. All logic inputs and outputs are TTL compatible as shown in *Figures 15* and *16*. The VCO CLOCK output is 74AS compatible and can therefore drive up to 40 74AS (or 74F) inputs, or 10 74S inputs, or 100 74ALS inputs, or 50 of 74LS inputs. All other outputs are 74ALS compatible and so will drive up to 16 74AS inputs, or 4 74S inputs, or 40 74ALS inputs or 20 74LS inputs. All inputs are 74ALS compatible and therefore can be driven easily from any 74 series devices. The raw MFM from the pulse detector in the drive is connected to the ENCODED DATA input. The DELAY DISABLE input determines whether attempting lock-on will begin immediately after READ GATE is set or after 2 bytes. Typically in a hard-sectored drive, READ GATE is set active as the sector pulse appears, meaning a new sector is about to pass under the head. Normally the preamble pattern does not begin immediately, because gap bytes from the preceding sector usually extend just beyond the sector pulse. Allowing 2 bytes to pass after the sector pulse helps ensure that the PLL will begin locking on to preamble, and will not be chasing non-symmetrical gap bits. Attempting to lock-on to a fixed ....1010.... preamble pattern speeds up lock-on, and after another two bytes the PLL will nominally have locked-on. Thus DELAY DISABLE should be set low for this kind of disk drive.



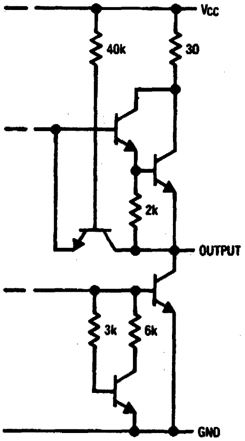
TL/F/5182-17

FIGURE 14. Alternate Loop Filter Configuration



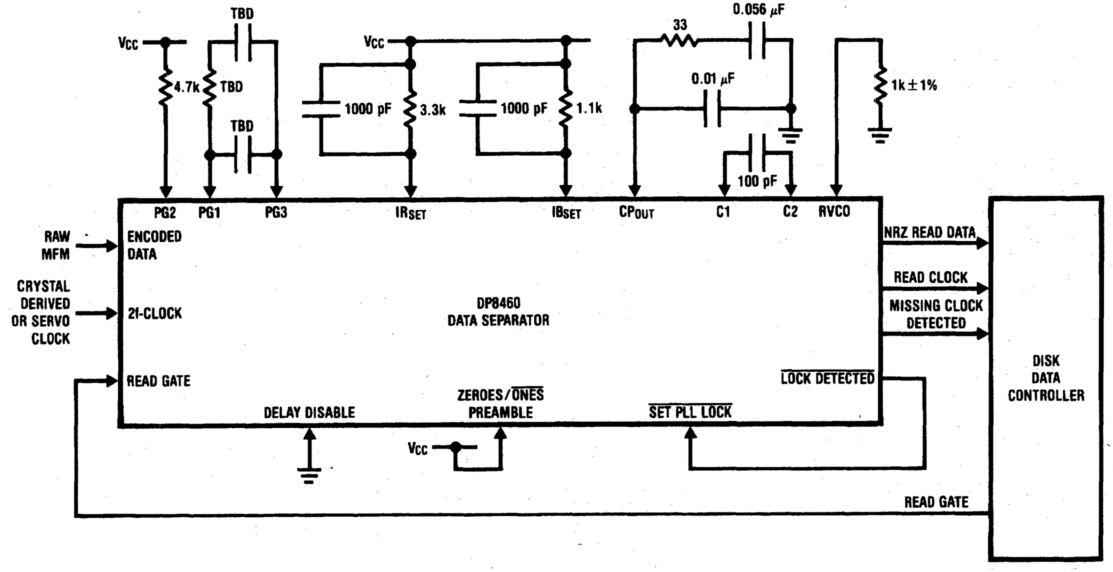
TL/F/5182-18

FIGURE 15. Logic Inputs



TL/F/5182-19

FIGURE 16. Logic Outputs



TL/F/5182-20

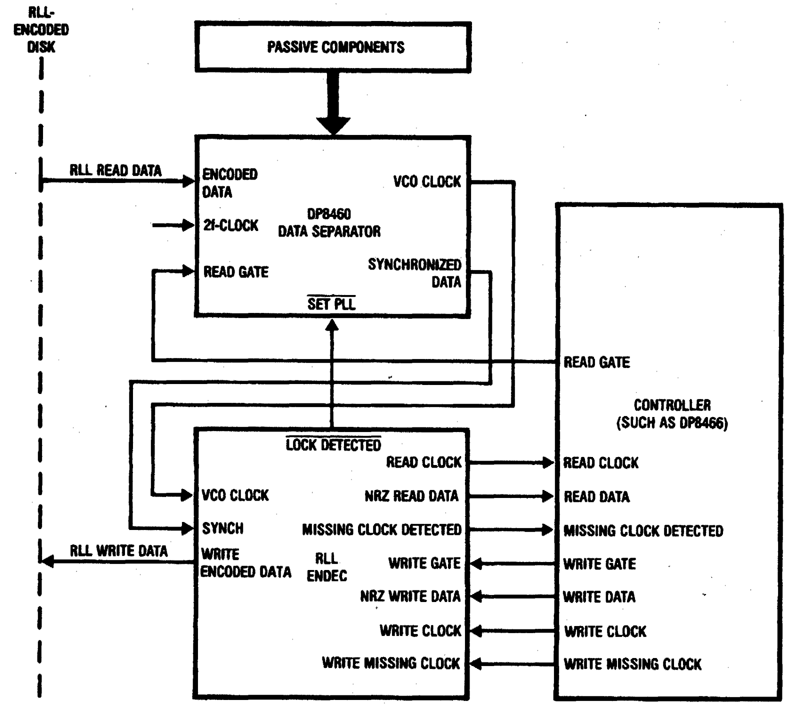
FIGURE 17. Typical Connection to DP8460 for:
1) MFM Data Input, 5Mbit/sec Data Rate
2) 32 Bit Delay to Enable
3) All Zeroes (NRZ) Preamble

For soft sectored drives, the controller normally will not wait for the index pulse before it attempts lock-on, so that READ GATE may go active at any time. Chances are the head will not be over a preamble field and therefore there is no need to wait 2 bytes before attempting lock-on. DELAY DISABLE can therefore be set high. If a non-preamble field is passing by as READ GATE goes active, the DP8460 will not indicate lock, and no data decoding will occur nor will MISSING CLOCK DETECTED go active. Normally, if lock-on has not been achieved after a certain time limit, the controller will de-activate READ GATE and then try again.

For MFM encoded disk drives, the $\overline{\text{LOCK DETECTED}}$ output will be connected back to the $\overline{\text{SET PLL LOCK}}$ input. As the PLL achieves lock-on, the DP8460 will automatically switch to the slower tracking rate and decoded data will appear at the NRZ READ DATA output. Also the READ CLOCK output will switch from half the 2f-clock frequency to the disk data rate frequency. If a delay is required before the changeover occurs, a time delay may be inserted between the two pins.

Some drives have an all-ONES data preamble instead of all-ZEROES and the DP8460 must be set to the type being used before it can properly decode data. The ZEROES/$\overline{\text{ONES}}$ PREAMBLE input selects which preamble type the chip is to lock-on to.

If the drive uses a run-length-limited (RLL) code such as '2, 7', instead of MFM, the phase-locked-loop function of the DP8460 may still be used. *Figure* 18 shows how the DP8460 may be connected to a RLL ENDEC circuit. The RLL ENDEC performs encoding of NRZ data to RLL encoded data, and RLL encoded data back to NRZ data. The RLL ENDEC can use the SYNCHRONIZED DATA output of the DP8460 along with VCO CLOCK to lock-on to the preamble and then decode data. Once lock-on has been detected, the RLL ENDEC can set the $\overline{\text{SET PLL LOCK}}$ input of the DP8460 low so that the tracking rate can be changed.



TL/F/5182-21

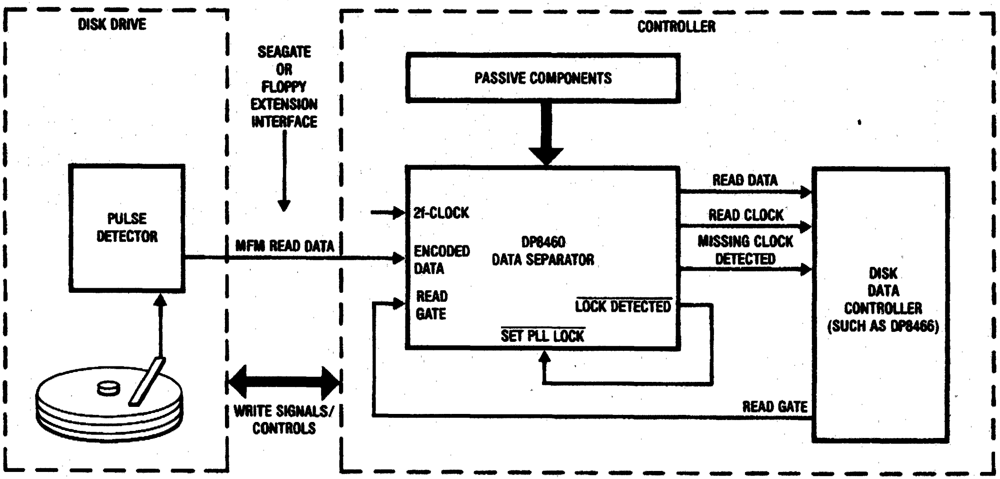
FIGURE 18. DP8460 with Run-Length-Limited (RLL) Codes

APPLICATIONS OF THE DP8460 DATA SEPARATOR

The DP8460 is the first integrated circuit to place on one chip a PLL with features that offer the improved speed and reliability required by the disk industry. Not only does the chip simplify disk system design, but also provides fast lock-on to the incoming preamble. Once locked on, the loop is set into a more stable mode. This inherent loop stability allows for a sizeable amount of jitter on the data stream, such as is encountered in many disk systems. Once in the stable tracking rate, the SYNCHRONIZED DATA output represents the incoming ENCODED DATA and is synchronous with VCO CLOCK. If the disk is MFM encoded, then the chip can decode the synchronized data into NRZ READ DATA and READ CLOCK. These are available as outputs from the chip allowing the NRZ READ DATA to be deserialized using the READ CLOCK.

The DP8460 is capable of operating at up to 25Mbits/sec data rates and so is compatible with a wide assortment of disk drives. The faster data rates of the 8-inch and 14-inch disk drives will mandate the selection of either the DP8460-3 or -2 parts with their narrower window margins on the incoming data stream. This will also be the case when 5¼-inch drives achieve higher data rates. Some 8-inch and 14-inch disk drives incorporate the functions of the DP8460, but use many discrete ICs. In these cases, replacing these components with the DP8460 will offer reduced P.C. board area, lower cost, and improved performance while simplifying circuit testing.

Most 5¼-inch and many 8-inch and 14-inch disk drives manufactured at present do not incorporate any of the functions of the DP8460. This is so primarily because the PLL function is difficult to design and implement, and requires circuitry which covers a large area of the printed circuit card. This is undesirable both from the drive size aspect and from the cost aspect (the cost includes soldering, testing, and adjusting the components). Consequently, most smaller disk drives output MFM encoded data so that the phase-locked-loop and data separation have to be performed by the controller. The DP8460 will therefore replace these functions in controller designs, as shown in *Figure 19.*

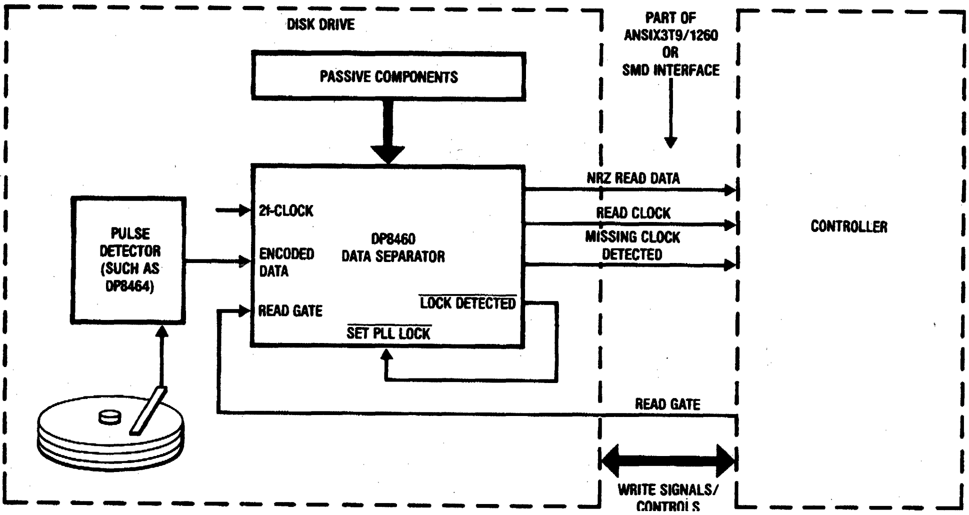


TL/F/5182-22

FIGURE 19. DP8460 In the Controller

System design criteria may now change because the DP8460 is a one-chip solution, requiring only a few external passive components with fixed values. It operates from a +5V supply, consumes about 0.5W, and is housed in a narrow 24-pin package. The circuitry has been designed so that the external resistors and capacitors need not be adjustable; the user chooses the values according to the disk drive requirements. Once selected, they will be fixed for that particular drive type. These features make it possible to transfer these functions to the disk drive, as shown in *Figure 20.* Apart from a slight increase in board area, the advantages outweigh the disadvantages. First, the components selected are fixed for each type of drive and this facilitates the problem of interchangeability of drives. At present, controllers are adjusted to function with each specific drive; with the DP8460 in the drive, component adjustment will no longer be required. Second there is often a problem of reliability of data transfer. Because the MFM data is clock encoded, this signal is susceptible to noise, bit shift, etc. Soft errors will sometimes occur when the incoming disk data bit position is outside the Pulse Gate window as it is being synchronized to the VCO clock in the phase-locked-loop. Obviously, the nearer the PLL is to the MFM source, the less chance there is that errors will occur. Thus placing the DP8460 in the drive will increase the reliability of data transfer within the sysem.

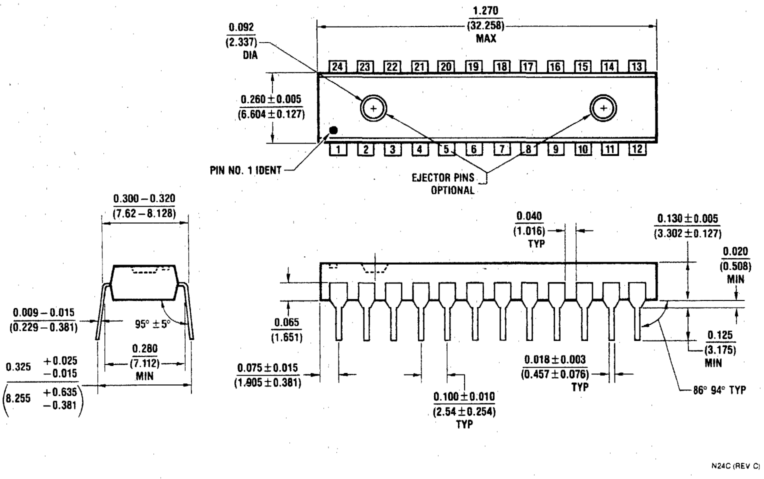
A third advantage is data rate upgrading. Most 5¼-inch drives have 5Mbit/sec data rate because the early drives were made with this data rate. This meant the controllers had to be designed with PLLs which operate at this data rate. It is therefore difficult for drive manufacturers to introduce new drives that are not compatible with existing controllers. Since no new standard data rate has emerged, they must continue to produce drives at this data rate to be compatible with the controllers on the market. With the DP8460 in the drive, and its associated components set for the drive's data rate, it no longer becomes a problem to increase the data rate, assuming the controllers digital circuitry can accommodate the change. This will allow the manufacturers to increase the bit density and therefore the capacity of their drives.



TL/F/5182-23

FIGURE 20. DP8460 in the Disk Drive

Physical Dimensions (Continued) inches (millimeters)



Molded Dual-In-Line Package (N)
Order Number DP8460N
NS Package Number N24C

National Semiconductor

PRELIMINARY

DP84300 Programmable Refresh Timer

General Description

The DP84300 programmable refresh timer is a logic device which produces the desired refresh clock required by all dynamic memory systems.

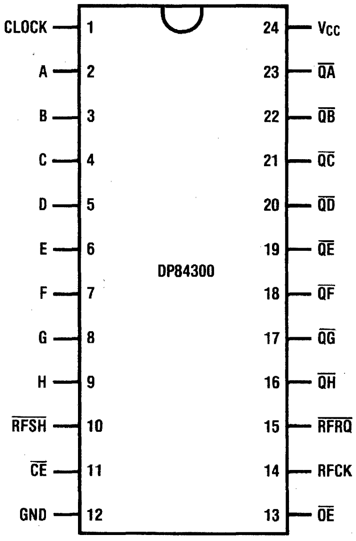
Additional circuitry has been included in the device to minimize logic required by memory systems to perform refresh control.

Features

- One chip solution to produce RFCK timing for the DP8408 and DP8409 dynamic RAM controllers
- Programmable refresh clock timer allows for a maximum refresh period with most system clocks
- Timing is completely synchronous with the input clock, preventing race conditions present in some memory controllers
- Includes a refresh request output, simplifying the design of refresh logic in discrete controllers

Connection Diagram

Dual-In-Line Package



TOP VIEW

Block Diagram

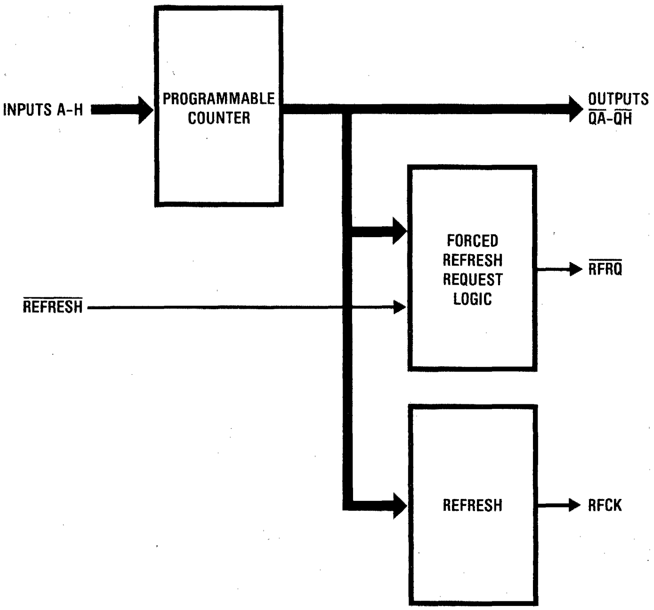


FIGURE 1

Recommended Operating Conditions (Commercial)

	Min	Typ	Max	Units
V_{CC}, Supply Voltage	4.75	5.00	5.25	V
I_{OH}, High Level Output Current			−3.2	mA
I_{OL}, Low Level Output Current			16	mA
T_A, Operating Free Air Temperature	0		75	°C

Electrical Characteristics over recommended operating temperature range

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	High Level Input Voltage		2			V
V_{IL}	Low Level Input Voltage				0.8	V
V_{IC}	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, V_{IH} = 2V, V_{IL} = 0.8V, I_{OH} = Max	2.4			V
V_{OL}	Low Level Output Voltage	V_{CC} = Min, V_{IH} = 2V, V_{IL} = 0.8V, I_{OL} = Max			0.5	V
I_{OZH}	Off-State Output Current High Level Voltage Applied	V_{CC} = Max, V_{IH} = 2V, V_O = 2.4V, V_{IL} = 0.8V			100	μA
I_{OZL}	Off-State Output Current Low Level Voltage Applied	V_{CC} = Max, V_{IH} = 2V, V_O = 0.4V, V_{IL} = 0.8V			−100	μA
I_I	Input Current at Maximum Input Voltage	V_{CC} = Max, V_I = 5.5V			1.0	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.4V			25	μA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V			−250	μA
I_{OS}	Short Circuit Output Current	V_{CC} = Max	−30		−130	mA
I_{CC}	Supply Current	V_{CC} = Max		150	180	mA

DP84300-3 Switching Characteristics over recommended ranges of temperature and V_{CC}

Symbol	Parameter		Conditions R_L = 667Ω	Commercial T_A = 0°C to +75°C V_{CC} = 5.0V ± 5%			Units
				Min	Typ	Max	
t_{PD}	Clock to Output		C_L = 45 pF		15	25	ns
t_{PZX}	Pin 13 to Output Enable				15	25	ns
t_{PXZ}	Pin 13 to Output Disable		C_L = 5 pF		15	25	ns
t_{PZX}	Input to Output Enable		C_L = 45 pF		25	35	ns
t_{PXZ}	Input to Output Disable		C_L = 5 pF		25	35	ns
t_W	Width of Clock	High		25			ns
		Low		25			ns
t_{su}	Set-Up Time			35			ns
t_H	Hold Time			0	−15		ns

Mnemonic Description

INPUT SIGNALS

CLOCK Provides a time base for the programmable divider.

A–H Program inputs A through H. These inputs select the number of clock cycles that will produce one refresh period. These inputs are binary encoded, with input A the LSB, and H the MSB. Additionally, all zeros produce the maximum count of 256, and an input of one will reset the counter to one.

$\overline{\text{REFRESH}}$ This input is used to reset the refresh request output ($\overline{\text{RFRQ}}$).

$\overline{\text{OE}}$ Output enable. Places the outputs in TRI-STATE®.

$\overline{\text{CE}}$ Counter enable. This input, when low, enables the timer clock and, when high, stalls the timer.

OUTPUT SIGNALS

$\overline{\text{QA}}$–$\overline{\text{QH}}$ Refresh timer outputs $\overline{\text{QA}}$ through $\overline{\text{QH}}$. Timer starts at programmed input and counts down to one.

$\overline{\text{RFRQ}}$ Refresh request. This output goes low on the rising edge of the refresh clock (RFCK). The first input clock edge after the $\overline{\text{REFRESH}}$ input is set low clears this output.

RFCK Refresh clock. The period of the clock is determined by setting conditions on input pins A through H. This output is low for 20 clocks, and high for the remainder of the period.

Functional Description

The DP84300 block diagram is shown in *Figure 1*. This circuit is basically an 8-bit programmable counter. The user selects the number of input clock cycles required per refresh period and sets the binary equivalent on inputs A through H. A signal of that period is produced at the refresh clock (RFCK) output. This output stays low for 20 clock cycles, and goes high for the balance of the period. When used with the DP8409 dynamic RAM controller, this duty cycle allows the DP8409 the maximum probability to perform a hidden refresh, while still allowing ample time for the DP8409 to perform a forced refresh when needed.

An additional output is provided to ease the design of systems that don't use the DP8409. This output is called refresh request ($\overline{\text{RFRQ}}$). Refresh request becomes true at the rising edge of refresh clock, and becomes false on the first rising edge of the input clock after a refresh.

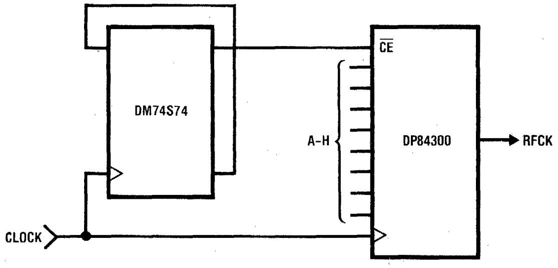
In systems where a divisor of more than 256 is needed, an expansion input ($\overline{\text{CE}}$) has been provided. When this input is high, all counter-related timing is suspended. This excludes actions due to the $\overline{\text{REFRESH}}$ input. The circuits in *Figures 2a and 2b* show how to expand the range of the timer by 2x or by up to 4096 clock cycles. *Figures 3a and 3b* show two typical applications using the DP84300.

By using the clock enable input, it is also possible to change the duty cycle of the refresh clock. The circuits in *Figures 4a and 4b* show how this may be done.

To reset the counter to a known state, select an input divisor of one. On the next clock edge the counter will reset to one. On the next clock edge whatever input divisor that is present on input A–H will be loaded into the counters.

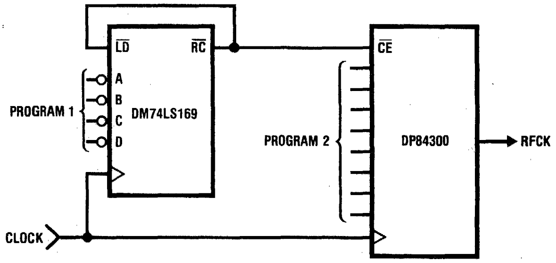
TABLE I. DIVIDER CONSTANTS FOR GENERATION OF A 15.5 μs CLOCK

CPU Clock Frequency	Divisor Input	Actual Period of Output	% Chance of Hidden Refresh
2 MHz	31	15.5 μs	35%
3 MHz	46	15.3 μs	56%
4 MHz	62	15.5 μs	67%
5 MHz	77	15.6 μs	74%
6 MHz	93	15.5 μs	78%
7 MHz	109	15.6 μs	81%
8 MHz	124	15.5 μs	83%
9 MHz	140	15.6 μs	85%
10 MHz	155	15.5 μs	87%



Period of RFCK = 2x program input

FIGURE 2a. Expansion of Clock Divisor by 2x



Period of RFCK 2 = program A x program B

$\overline{\text{RFCK}}$ is low for 20x program 1 clocks

Maximum period of RFCK is 4096 clocks

FIGURE 2b. Typical Expansion for the DP84300

Functional Description (Continued)

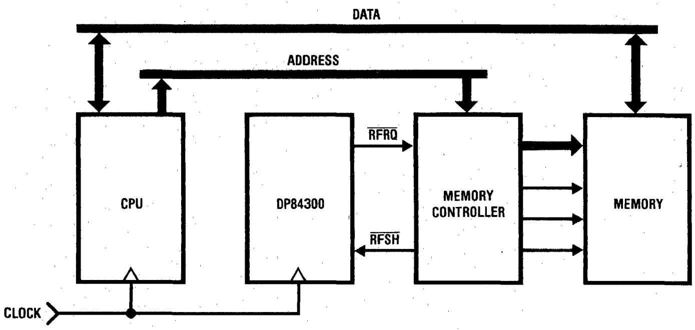


FIGURE 3a. Dynamic Memory System Using DP84300

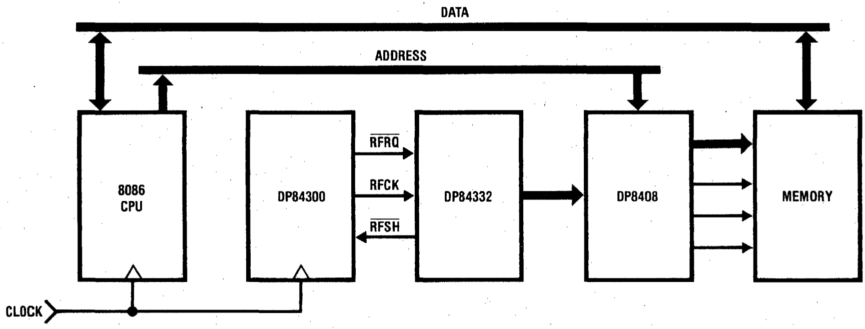


FIGURE 3b. 8086 System Using Dynamic RAMs DP8408, DP84300, and DP84332

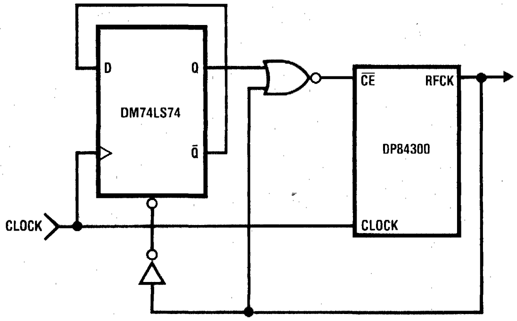


FIGURE 4a. Circuit for Extending RFCK Low to 40 Clocks

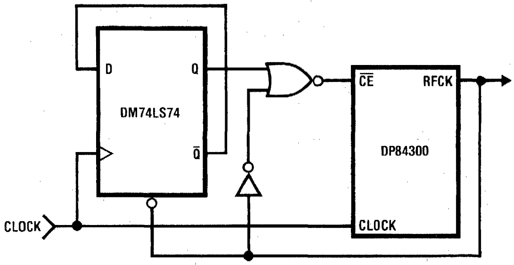
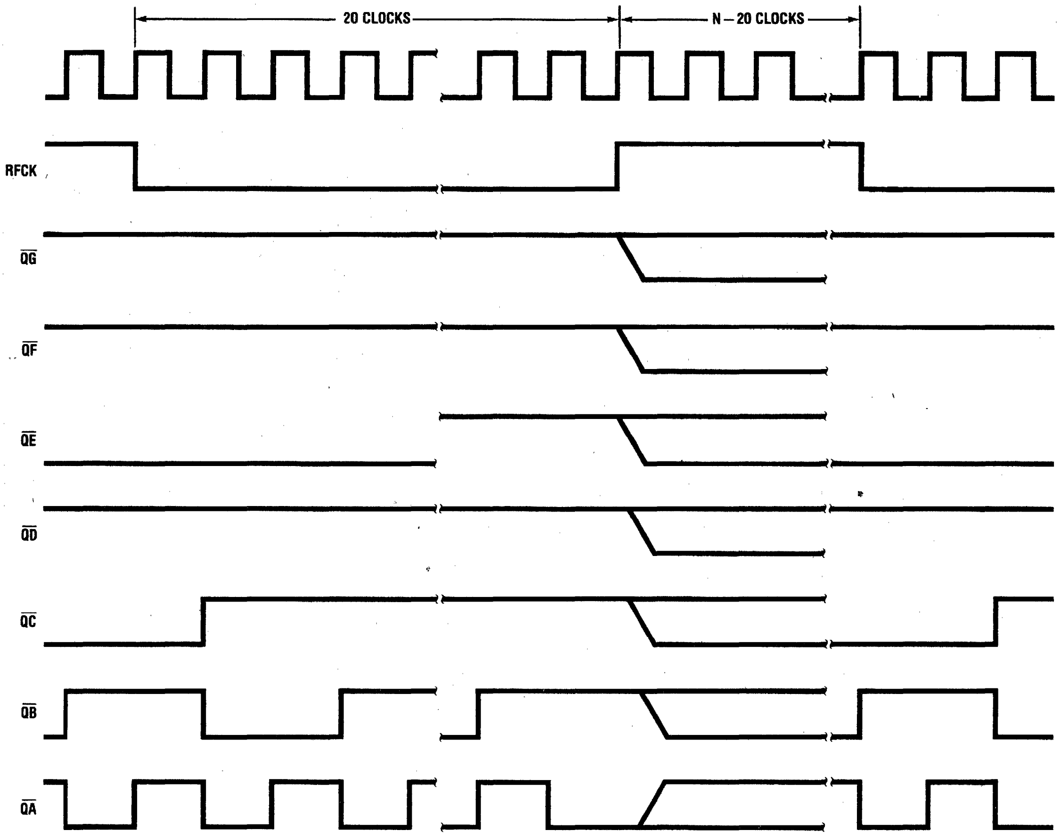


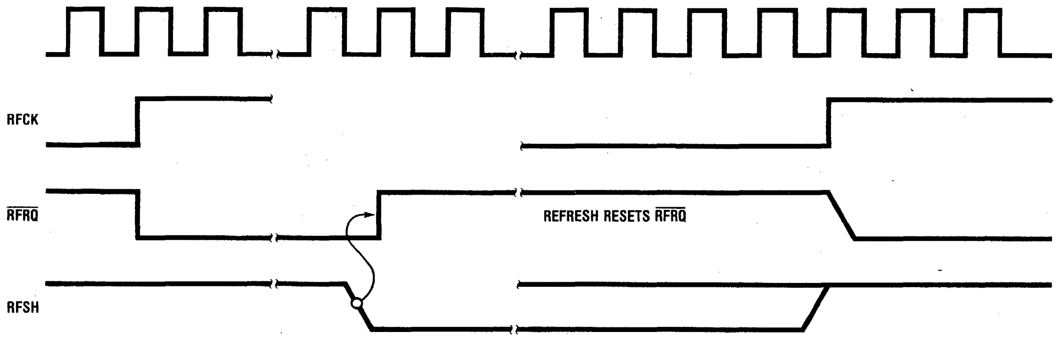
FIGURE 4b. Circuit for Extending RFCK High by 2x

Timing Diagrams

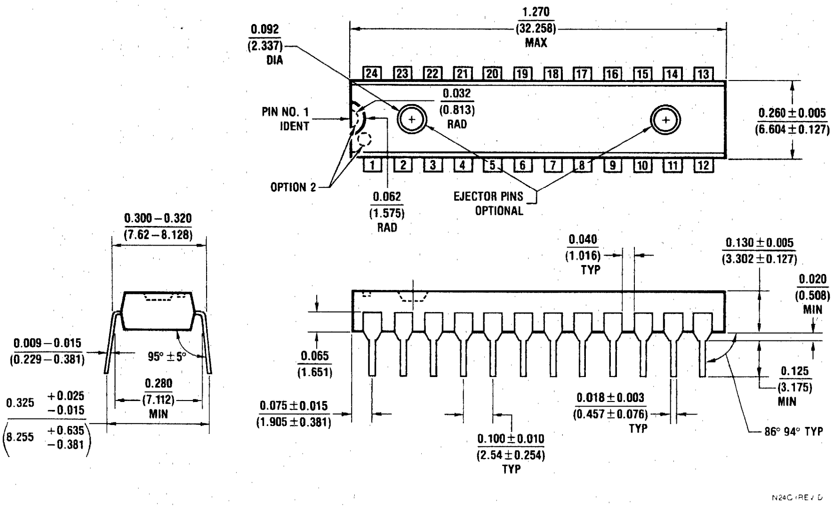
Refresh Timer Outputs



$\overline{\text{REFRESH REQUEST}}$ ($\overline{\text{RFRQ}}$) Output Timing



Physical Dimensions inches (millimeters)



Molded Dual-In-Line Package (N)
Order Number DP84300N-3
NS Package Number N24A

National Semiconductor

PRELIMINARY

DP84312 Dynamic RAM Controller Interface Circuit for the NS16032 CPU

General Description

The DP84312 dynamic RAM controller interface is a Programmable Array Logic (PAL)* device which allows for easy interface between the DP8409 dynamic RAM Controller and the NS16032 microprocessor.

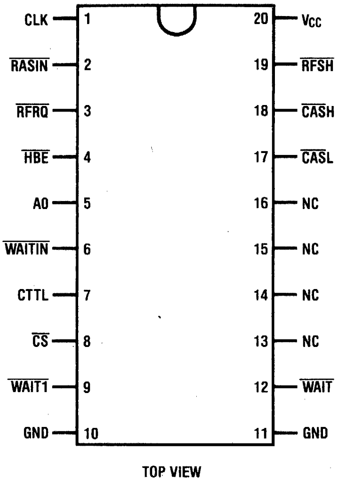
Using timing signals from the NS16201 timing and control unit and the NS16032, the DP84312 supplies all control signals needed to perform memory read, write, byte write, and refresh.

Features

- Low parts count memory system
- Allows the DP8409 to perform hidden refresh
- Allows for the insertion of wait states for slow dynamic RAMs
- Supplies independent $\overline{\text{CAS}}$s for byte writing
- Possibility of operation at 8MHz with no wait states
- 20-pin 0.3 inch wide package
- Standard National Semiconductor PAL part (DMPAL16R6)
- PAL logic equations can be modified by the user for his specific application and programmed into any of the PALs in the National Semiconductor PAL family, including the new high speed PALs.

Connection Diagram

Dual-In-Line Package



Recommended Operating Conditions (Commercial)

	Min	Typ	Max	Units
V_{CC}, Supply Voltage	4.75	5.00	5.25	V
I_{OH}, High Level Output Current			−3.2	mA
I_{OL}, Low Level Output Current			24 (Note 2)	mA
T_A, Operating Free Air Temperature	0		75	°C

Electrical Characteristics over recommended operating temperature range

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{IH}	High Level Input Voltage		2			V
V_{IL}	Low Level Input Voltage				0.8	V
V_{IC}	Input Clamp Voltage	V_{CC} = Min, I_I = −18 mA			−1.5	V
V_{OH}	High Level Output Voltage	V_{CC} = Min, V_{IH} = 2V, V_{IL} = 0.8V, I_{OH} = Max	2.4			V
V_{OL}	Low Level Output Voltage	V_{CC} = Min, V_{IH} = 2V, V_{IL} = 0.8V, I_{OL} = Max			0.5	V
I_{OZH}	Off-State Output Current High Level Voltage Applied	V_{CC} = Max, V_{IH} = 2V, V_O = 2.4V, V_{IL} = 0.8V			100	µA
I_{OZL}	Off-State Output Current Low Level Voltage Applied	V_{CC} = Max, V_{IH} = 2V, V_O = 0.4V, V_{IL} = 0.8V			−100	µA
I_I	Input Current at Maximum Input Voltage	V_{CC} = Max, V_I = 5.5V			1.0	mA
I_{IH}	High Level Input Current	V_{CC} = Max, V_I = 2.4V			25	µA
I_{IL}	Low Level Input Current	V_{CC} = Max, V_I = 0.4V			−250	µA
I_{OS}	Short Circuit Output Current	V_{CC} = Max	−30		−130	mA
I_{CC}	Supply Current	V_{CC} = Max		150	225 (Note 1)	mA

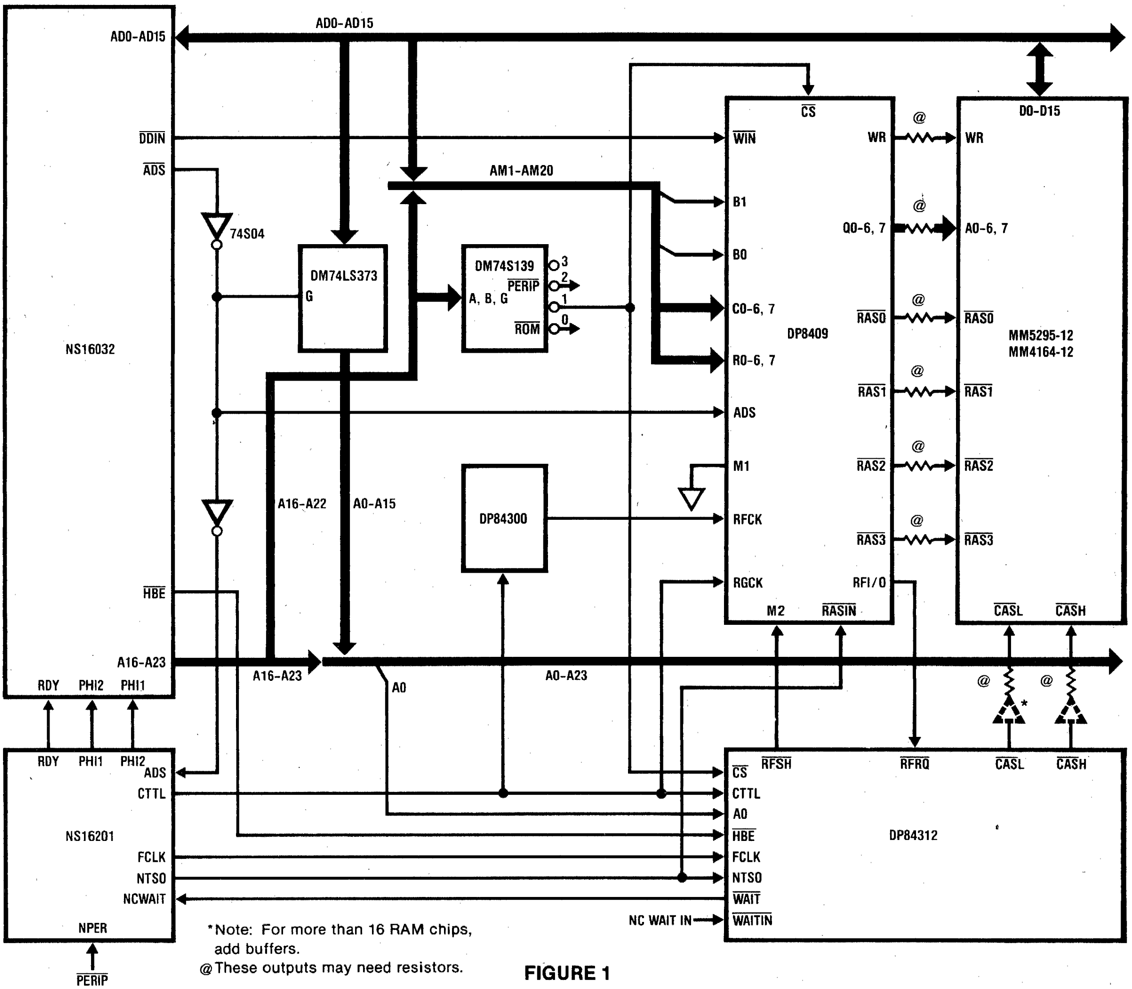
DP84312-3 Switching Characteristics over recommended ranges of temperature and V_{CC}

Symbol	Parameter		Conditions $R_L = 667\Omega$	Commercial T_A = 0°C to +75°C V_{CC} = 5.0V ± 5%			Units
				Min	Typ	Max	
t_{WD}	$\overline{\text{WAITIN}}$ to $\overline{\text{WAIT}}$ Delay		C_L = 45 pF		25	40	ns
t_{PD}	Clock to Output		C_L = 45 pF		15	25	ns
t_{PZX}	Pin 11 to Output Enable		C_L = 45 pF		15	25	ns
t_{PXZ}	Pin 11 to Output Disable		C_L = 5 pF		15	25	ns
t_w	Width of Clock	High		25			ns
		Low		25			ns
t_{su}	Set-Up Time			40			ns
t_h	Hold Time			0	−15		ns

Note 1: I_{CC} = max at minimum temperature.

Note 2: One output at a time; otherwise 16 mA.

System Block Diagram



*Note: For more than 16 RAM chips, add buffers.
@These outputs may need resistors.

FIGURE 1

Mnemonic Description

INPUT SIGNALS

CLK — Clock input. This clock comes from the FCLK output of the NS16201 timing and control unit, and supplies timing for the internal logic.

$\overline{\text{RASIN}}$ — $\overline{\text{RAS}}$ input. This input is connected to the NTSO pin of the NS16201. This signal marks the start of a memory cycle.

$\overline{\text{RFRQ}}$ — Refresh request. The DP8409 requests a forced refresh with this input.

$\overline{\text{HBE}}$, A0 — Address select inputs. These inputs select the type of write during a write cycle, and select their respective $\overline{\text{CAS}}$ outputs. These inputs must remain stable throughout the memory cycle.

$\overline{\text{WAITIN}}$ — This wait input allows other devices to use the NCWAIT line of the NS16201 clock chip.

CTTL — System clock input. This clock is used to synchronize the memory system to the microprocessor clock.

$\overline{\text{CS}}$ — Chip select. This input is used to determine if a memory cycle or a hidden refresh cycle is to be performed.

$\overline{\text{WAIT1}}$ — Insert one wait state. This input allows the use of slow memories with a microprocessor using a fast clock by inserting a wait state in selected memory cycles.

V_{CC}, GND — 5.0V ± 5%.

OUTPUT SIGNALS

$\overline{\text{RFSH}}$ — Refresh. This output switches the DP8409 to a refresh mode.

$\overline{\text{CASH}}$, $\overline{\text{CASL}}$ — $\overline{\text{CAS}}$ outputs. $\overline{\text{CASH}}$ is for controlling the high bank of dynamic RAMs, while $\overline{\text{CASL}}$ controls the $\overline{\text{CAS}}$ line of the lower bank of RAMs. If only eight RAMs are used in each bank, the $\overline{\text{CAS}}$ outputs will directly drive the memories. For larger arrays, these outputs should be buffered with a high current driver, such as the DP84244 MOS driver.

$\overline{\text{WAIT}}$ — This output controls the insertion of wait states. This output is ORed with $\overline{\text{WAITIN}}$ to allow other devices to insert wait states.

Functional Description

The DP84312 detects the start of a memory cycle when NTSO from the NS16032 timing and control unit (TCU) goes low. The NTSO signal is also used to supply $\overline{\text{RASIN}}$ to the DP8409 dynamic RAM controller. After the DP8409 has latched the row address and supplied the column address to the DRAMs, the DP84312 latches the column address. The DP84312 supplies two $\overline{\text{CAS}}$ outputs, one for the high byte of memory, and the other for the low byte. The ability to control the upper and lower bytes of memory separately is important during a memory write cycle where one byte of memory is to be written (byte write).

By connecting $\overline{\text{WAIT1}}$ of the DP84312 to ground, all selected memory cycles will have one wait state inserted. This allows an NS16032 operating at high CPU clock frequencies to use slower dynamic RAMs.

Memory refresh may be achieved in one of two ways: hidden or forced. Hidden refresh is accomplished whenever a refresh is requested (internal to the DP8409) and an unselected memory cycle occurs. With a hidden refresh, the DP84312 does nothing while the DP8409 performs the refresh. If no refresh has occurred before the trailing edge of refresh clock, the DP8409 will request a forced refresh. The DP84312 detects this request, and allows the current memory cycle to finish. It then outputs wait states to the CPU, which will hold the CPU if it requests a memory cycle. During this time the DP84312 has switched the dynamic RAM controller to the auto refresh mode, allowing it to perform a refresh. At the end of the refresh cycle, the DP8409 is switched back to the auto access mode, and the wait is removed after a sufficient $\overline{\text{RAS}}$ precharge time. The total forced refresh takes four CPU clock cycles; of which some, none or all may be actual wait states. If the CPU does not request a memory cycle during this refresh cycle, the refresh will not impact the CPU's performance.

The DP84312 can possibly be operated at 8 MHz with no wait states (WAIT1 = "1") given the following conditions:

T2 + T3 = 250 ns
NTSO generation = 15 ns max.
$\overline{\text{RASIN}}$ to $\overline{\text{CAS}}$ delay DP8409-2 = 130 ns max.
External $\overline{\text{CASH,L}}$ generation using 74S02 and 74S240
7.5 ns (74S02) + 10 ns (74S240) − 7.5 ns (less load on 8409 $\overline{\text{CAS}}$ line) = 10 ns max.
Transceiver delay = 12 ns max.
NS16032 data setup = 20 ns max.
∴ Minimum t_{CAC} = 63 ns
= 250 − 15 − 130 − 10 − 12 − 20
Minimum t_{RAS} = 250 ns
Minimum t_{RP} = 250 ns
Minimum t_{RAH} = 20 ns

The DP84312 is a standard National Semiconductor PAL part (DMPAL16R6). The user can modify the PAL equations to support his particular application. The DP84312 logic equations, function table (functional test), and logic diagram can be seen at the end of this Data Sheet.

Timing Diagrams

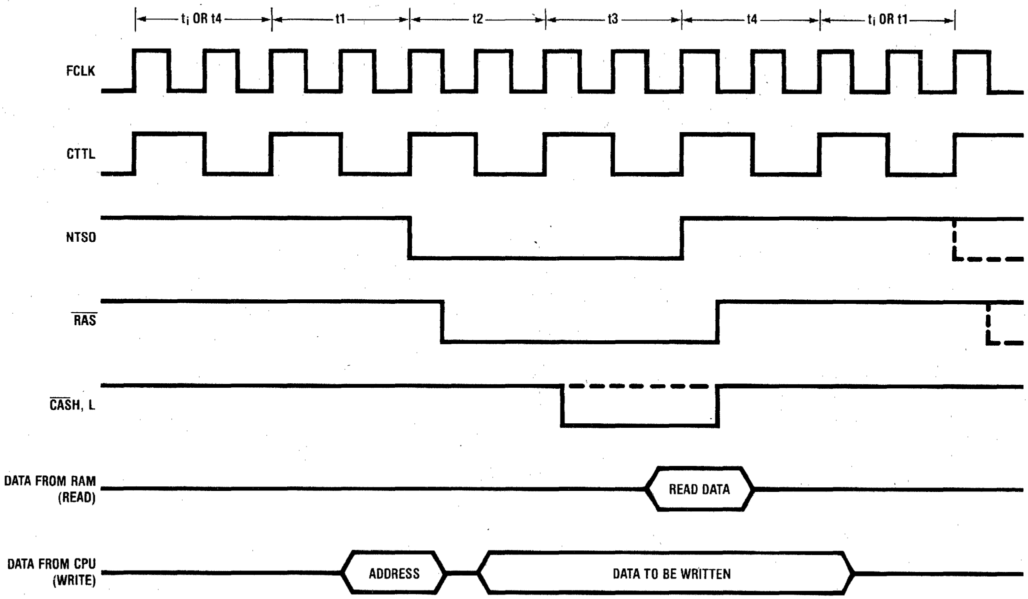


FIGURE 2a. Read, Write, or Hidden Refresh Memory Cycle for the NS16032-DP8409 Interface

Timing Diagrams (Continued)

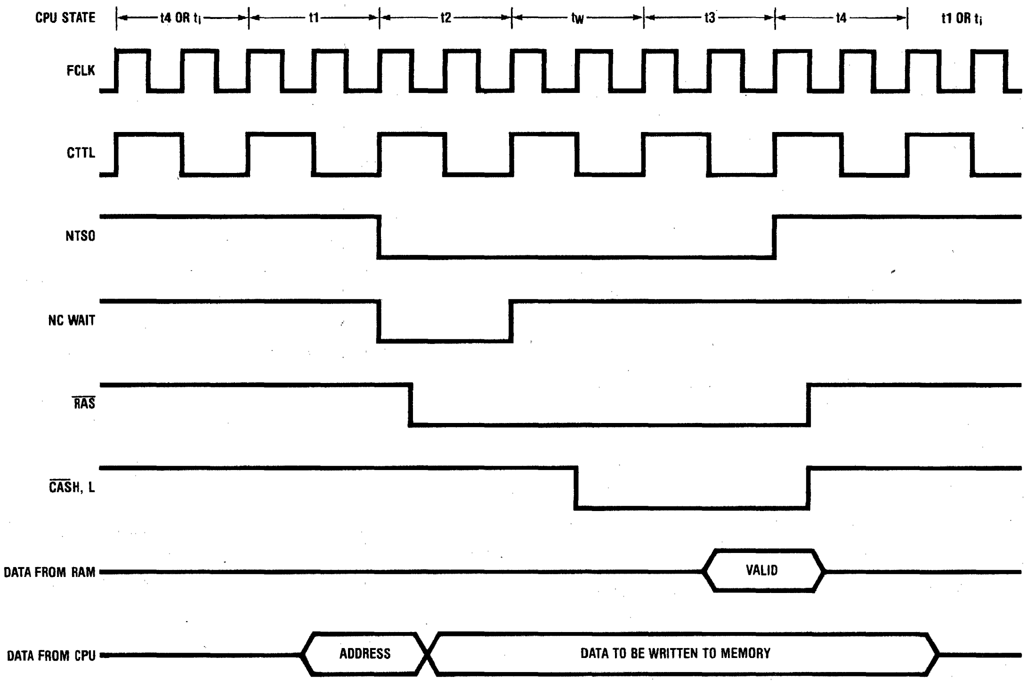


FIGURE 2b. Read or Write Memory Cycle with One Wait

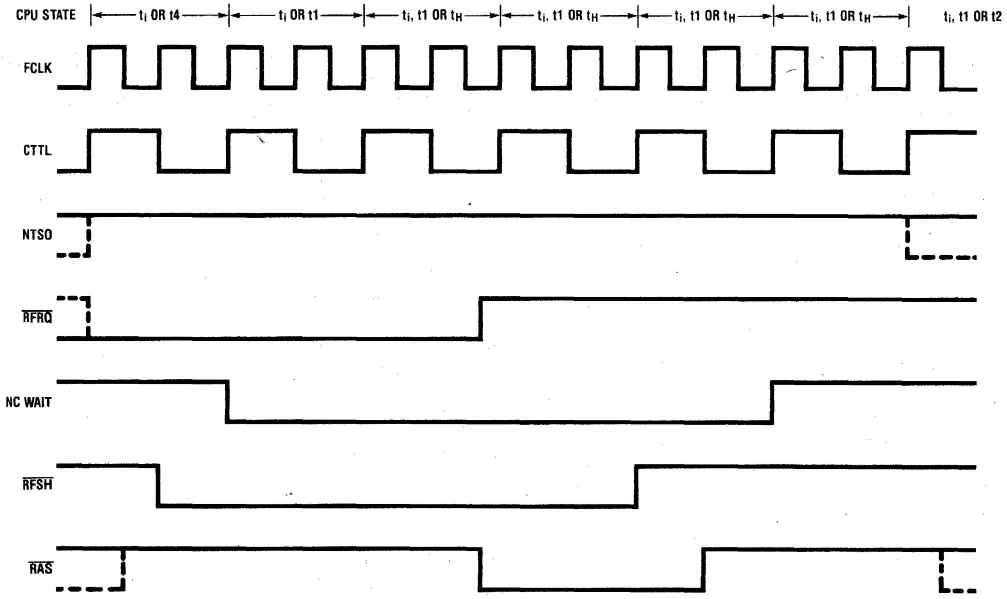


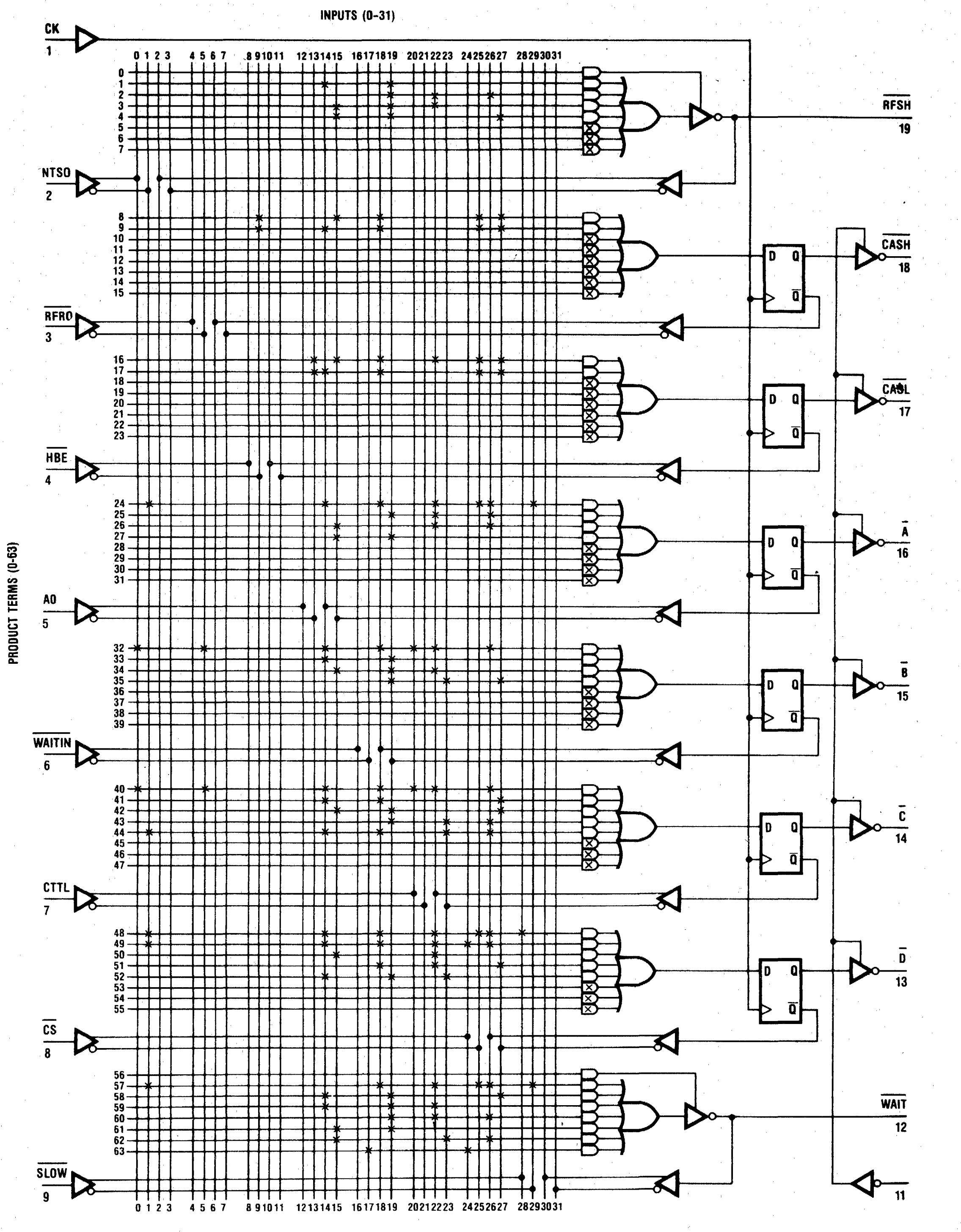
FIGURE 2c. Forced Refresh Cycle

```
PAL16R6
DP84312
Interface Circuit for the NS16032/DP8409
Memory System
CK NTSO /RFRQ /HBE A0 /WAITIN CTTL /CS /SLOW
GND /OE /WAIT /D /C /B /A /CASL /CASH /RFSH VCC
CASH:= A • /B • /C • D •HBE • CS +
        /A • /B • D • HBE • CS
CASL:= A • /B • /C • D • /A0 • CS +
        /A • /B • D /A0 • CS
A     := /A • /B • /C • /D • /NTSO • CS • SLOW +
        B • /C • /D +
        A • /C • /D +
        A • B
B     := /A • /B • /C • /D • NTSO • RFRQ • CTTL +
        /A • B +
        A • B • /C +
        B • C • D
C     := /A • /B • /C • /D • NTSO • RFRQ • CTTL +
        /A • /B • D +
        A • B • D +
        B • C • /D +
        /A • /B • C • /D • /NTSO
D     := /A • /B • /C • /D • /NTSO • CS • /SLOW +
        /A • /B • /C • /D • /NTSO • /CS +
        A • /C +
        /B • /C • D +
        /A • B • C
IF (VCC) WAIT = /B • /C • /D • /NTSO • CS • SLOW +
        /A • B • D +
        B • /C • /D +
        A • B +
        A • C • /D +
        /CS • WAITIN
IF (VCC) RFSH = /A• B +
        B • /C • /D +
        A • B • /C +
        A • B • C
```

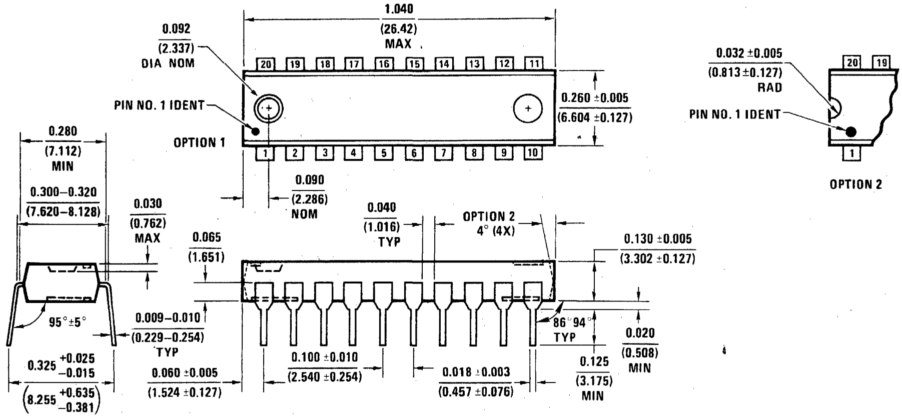
Function Table

CK	NTSO	$\overline{\text{RFRQ}}$	$\overline{\text{HBE}}$	A0	$\overline{\text{WAITIN}}$	CTTL	$\overline{\text{CS}}$	$\overline{\text{SLOW}}$	$\overline{\text{OE}}$	CASH	CASL	A	B	C	D	WAIT	RFSH
C	H	H	L	L	H	H	H	H	L	X	X	X	X	X	X	X	X
C	H	H	L	L	H	H	H	H	L	L	L	L	L	L	L	L	L
C	L	X	L	L	H	X	L	H	L	L	L	L	L	L	H	L	L
C	L	X	L	L	H	X	L	H	L	H	H	L	L	H	H	L	L
C	X	X	L	L	H	X	L	H	L	H	H	L	L	H	L	L	L
C	H	X	L	L	H	X	L	H	L	L	L	L	L	L	L	L	L
C	L	X	L	H	H	X	L	L	L	L	L	H	L	L	L	H	L
C	X	X	L	H	H	X	L	L	L	L	L	H	L	L	H	L	L
C	X	X	L	H	H	X	L	L	L	H	L	L	L	L	H	L	L
C	X	X	L	H	H	X	L	L	L	H	L	L	L	H	H	L	L
C	X	X	L	H	H	X	L	L	L	H	L	L	L	H	L	L	L
C	H	X	L	L	H	X	H	H	L	L	L	L	L	L	L	L	L
C	L	X	L	L	H	X	H	X	L	L	L	L	L	L	H	L	L
C	X	X	L	L	L	X	H	X	L	L	L	L	L	H	H	H	L
C	H	X	L	L	H	X	H	X	L	L	L	L	L	H	L	L	L
C	H	X	L	L	H	X	H	X	L	L	L	L	L	L	L	L	L
C	H	L	X	X	H	H	X	X	L	L	L	L	H	H	L	L	H
C	H	X	X	X	H	L	X	X	L	L	L	L	H	H	H	H	H
C	H	H	X	X	H	H	X	X	L	L	L	L	H	L	H	H	H
C	H	H	X	X	H	L	X	X	L	L	L	L	H	L	L	H	H
C	H	H	X	X	H	H	X	X	L	L	L	H	H	L	L	H	H
C	H	H	X	X	H	L	X	X	L	L	L	H	H	L	H	H	H
C	H	H	X	X	H	H	X	X	L	L	L	H	H	H	H	H	H
C	H	H	X	X	H	L	X	X	L	L	L	H	H	H	L	H	L
C	H	H	X	X	H	H	X	X	L	L	L	H	L	H	L	H	L
C	H	H	X	X	H	X	H	H	L	L	L	L	L	L	L	L	L
C	L	H	X	X	L	X	H	X	L	L	L	L	L	L	H	H	L
C	L	H	X	X	L	X	H	X	L	L	L	L	L	H	H	H	L
C	L	H	X	X	L	X	H	X	L	L	L	L	L	H	L	H	L
C	L	X	X	X	H	X	H	X	L	L	L	L	L	H	L	L	L
C	H	X	X	X	H	X	H	X	L	L	L	L	L	L	L	L	L
C	H	H	H	H	H	H	H	H	H	Z	Z	Z	Z	Z	Z	Z	Z

DP84312 Logic Diagram PAL16R6



Physical Dimensions inches (millimeters)



N20A (REV D)

Molded Dual-In-Line Package (N)
Order Number DP84312N-3
NS Package Number N20A

National Semiconductor

PRELIMINARY

INS8250A Asynchronous Communications Element

General Description

The INS8250A is the enhanced version of the programmable Asynchronous Communications Element (ACE) chip — the 8250. It provides an on-board programmable baud generator, and is contained in a standard 40-pin dual-in-line package. The new ACE is fabricated using National Semiconductor's advanced, scaled N-channel silicon-gate MOS process, XMOS. It functions as a serial data input/output interface in a microcomputer system. The functional configuration of the INS8250A is programmed by the system software via a TRI-STATE® 8-bit bidirectional data bus.

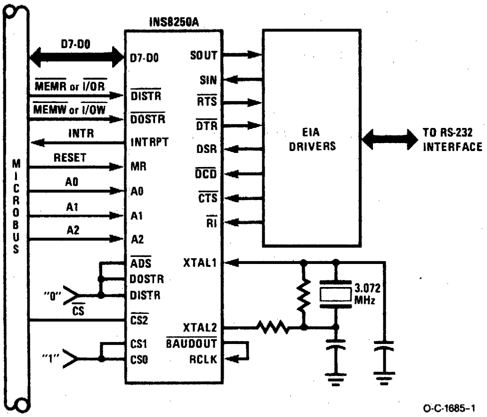
The INS8250A performs serial-to-parallel conversion on data characters received from a peripheral device or a MODEM, and parallel-to-serial conversion on data characters received from the CPU. The CPU can read the complete status of the INS8250A at any time during the functional operation. Status information reported includes the type and condition of the transfer operations being performed by the INS8250A, as well as any error conditions (parity, overrun, framing, or break interrupt).

The INS8250A includes a programmable baud generator that is capable of dividing the timing reference clock input by divisors of 1 to $(2^{16}-1)$, and producing a 16 × clock for driving the internal transmitter logic. Provisions are also included to use this 16 × clock to drive the receiver logic. Also included in the INS8250A is a complete MODEM-control capability, and a processor-interrupt system that may be software tailored to the user's requirements to minimize the computing time required to handle the communications link.

Features

- Easily interfaces to most popular microprocessors.
- Adds or deletes standard asynchronous communication bits (start, stop, and parity) to or from serial data stream.
- Full double buffering eliminates need for precise synchronization.
- Independently controlled transmit, receive, line status, and data set interrupts.
- Programmable baud generator allows division of any input clock by 1 to $(2^{16}-1)$ and generates the internal 16 × clock.
- Independent receiver clock input.
- MODEM control functions (CTS, RTS, DSR, DTR, RI, and DCD).
- Fully programmable serial-interface characteristics:
  - — 5-, 6-, 7-, or 8-bit characters
  - — Even, odd, or no-parity bit generation and detection
  - — 1-, 1½-, or 2-stop bit generation
  - — Baud generation (DC to 56k baud).
- False start bit detection.
- Complete status reporting capabilities.
- TRI-STATE TTL drive capabilities for bidirectional data bus and control bus.
- Line break generation and detection.
- Internal diagnostic capabilities:
  - — Loopback controls for communications link fault isolation
  - — Break, parity, overrun, framing error simulation.
- Full prioritized interrupt system controls.
- Microbus™ compatible.

Microbus Configuration



Absolute Maximum Ratings

Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with Respect to V_{SS}	−0.5V to +7.0V
Power Dissipation	700mW

Note: *Maximum ratings indicate limits beyond which permanent damage may occur. Continuous operation at these limits is not intended and should be limited to those conditions specified under DC electrical characteristics.*

DC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 5\%$, $V_{SS} = 0V$, unless otherwise specified.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{ILX}	Clock Input Low Voltage		−0.5		0.8	V
V_{IHX}	Clock Input High Voltage		2.0		V_{CC}	V
V_{IL}	Input Low Voltage		−0.5		0.8	V
V_{IH}	Input High Voltage		2.0		V_{CC}	V
V_{OL}	Output Low Voltage	$I_{OL} = 1.6mA$ on all			0.4	V
V_{OH}	Output High Voltage	$I_{OH} = -1.0mA$	2.4			V
I_{CC}(AV)	Avg. Power Supply Current (V_{CC})	$V_{CC} = 5.25V$, $T_A = 25°C$ No Loads on output SIN, DSR, RLSD, CTS, RI = 2.0V All other inputs = 0.8V			95	mA
I_{IL}	Input Leakage	$V_{CC} = 5.25V$, $V_{SS} = 0V$ All other pins floating. $V_{IN} = 0V$, 5.25V			±10	μA
I_{CL}	Clock Leakage				±10	μA
I_{OZ}	TRI-STATE Leakage	$V_{CC} = 5.25V$ $V_{SS} = 0V$ $V_{OUT} = 0V$, 5.25V 1) Chip deselected 2) WRITE mode, chip selected			±20	μA
V_{ILMR}	MR Schmitt V_{IL}				0.8	V
V_{IHMR}	MR Schmitt V_{IH}		2.0			V

Capacitance $T_A = 25°C$, $V_{CC} = V_{SS} = 0V$

Symbol	Parameter	Conditions	Min	Typ	Max	Units
C_{XIN}	Clock Input Capacitance	$f_c = 1MHz$ Unmeasured pins returned to V_{SS}		15	20	pF
C_{XOUT}	Clock Output Capacitance			20	30	pF
C_{IN}	Input Capacitance			6	10	pF
C_{OUT}	Output Capacitance			10	20	pF

AC Electrical Characteristics $T_A = 0°C$ to $+70°C$, $V_{CC} = +5V \pm 5\%$.

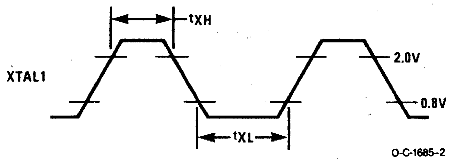
Symbol	Parameter	Conditions	Min	Max	Units
t_{AW}	Address Strobe Width		90		ns
t_{AS}	Address Setup Time		90		ns
t_{AH}	Address Hold Time		0		ns
t_{CS}	Chip Select Setup Time		90		ns
t_{CH}	Chip Select Hold Time		0		ns
t_{DIW}	$\overline{\text{DISTR}}$/DISTR Strobe Width		175		ns
t_{RC}	Read Cycle Delay		500		ns
RC	Read Cycle $= t_{AR}{}^* + t_{DIW} + t_{RC}$		755		ns
t_{DD}	$\overline{\text{DISTR}}$/DISTR to Driver Disable Delay	–@100pF loading		75	ns
t_{DDD}	Delay from $\overline{\text{DISTR}}$/DISTR to Data	–@100pF loading		175	ns
t_{HZ}	$\overline{\text{DISTR}}$/DISTR to Floating Data Delay	–@100pF loading	100		ns
t_{DOW}	$\overline{\text{DOSTR}}$/DOSTR Strobe Width		175		ns
t_{WC}	Write Cycle Delay		500		ns
WC	Write Cycle $= t_{AW}{}^* + t_{DOW} + t_{WC}$		755		ns
t_{DS}	Data Setup Time		90		ns
t_{DH}	Data Hold Time		60		ns
$t_{CSC}{}^*$	Chip Select Output Delay from Select	–@100pF loading		125	ns
$t_{RA}{}^*$	Address Hold Time from $\overline{\text{DISTR}}$/DISTR		20		ns
$t_{RCS}{}^*$	Chip Select Hold Time from $\overline{\text{DISTR}}$/DISTR		20		ns
$t_{AR}{}^*$	$\overline{\text{DISTR}}$/DISTR Delay from Address		80		ns
$t_{CSR}{}^*$	$\overline{\text{DISTR}}$/DISTR Delay from Chip Select		80		ns
$t_{WA}{}^*$	Address Hold Time from $\overline{\text{DOSTR}}$/DOSTR		20		ns
$t_{WCS}{}^*$	Chip Select Hold Time from $\overline{\text{DOSTR}}$/DOSTR		20		ns
$t_{AW}{}^*$	$\overline{\text{DOSTR}}$/DOSTR Delay from Address		80		ns
$t_{CSW}{}^*$	$\overline{\text{DOSTR}}$/DOSTR Delay from Select		80		ns
t_{MRW}	Master Reset Pulse Width		10		μs
t_{XH}	Duration of Clock High Pluse	External Clock (3.1MHz Max.)	140		ns
t_{XL}	Duration of Clock	External Clock (3.1MHz Max.)	140		ns
Baud Generator					
N	Baud Divisor		1	$2^{16}-1$	
t_{BLD}	Baud Output Negative Edge Delay	100pF Load		250	ns
t_{BHD}	Baud Output Positive Edge Delay	100pF Load		250	ns
t_{LW}	Baud Output Down Time	$f_X = 2MHz$, ÷2, 100pF Load	425		ns
t_{HW}	Baud Output Up Time	$f_X = 3MHz$, ÷3, 100pF Load	330		ns
Receiver					
t_{SCD}	Delay from RCLK to Sample Time			2	μs
t_{SINT}	Delay from Stop to Set Interrupt		1	1	$\overline{\text{BAUDOUT}}$ Cycles
t_{RINT}	Delay from $\overline{\text{DISTR}}$/DISTR (RD RBR/RDLSR) to Reset Interrupt	100pF Load		1	μs

*Applicable only when $\overline{\text{ADS}}$ is low.

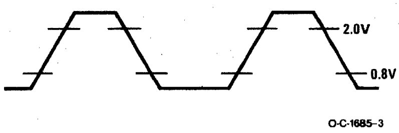
AC Electrical Characteristics (Continued)

Symbol	Parameter	Conditions	Min	Max	Units
Transmitter					
t_{HR}	Delay from $\overline{\text{DOSTR}}$/DOSTR (WR THR) to Reset Interrupt	100pF Load		1	μs
t_{IRS}	Delay from Initial INTR Reset to Transmit Start		8	24	$\overline{\text{BAUDOUT}}$ Cycles
t_{SI}	Delay from Initial Write to Interrupt		16	32	$\overline{\text{BAUDOUT}}$ Cycles
t_{STI}	Delay from Stop to Interrupt (THRE)		8	8	$\overline{\text{BAUDOUT}}$ Cycles
t_{IR}	Delay from $\overline{\text{DISTR}}$/DISTR (RD IIR) to Reset Interrupt (THRE)	100pF Load		1	μs
Modem Control					
t_{MDO}	Delay from $\overline{\text{DOSTR}}$/DOSTR (WR MCR) to Output	100pF Load		1	μs
t_{SIM}	Delay to Set Interrupt from MODEM Input	100pF Load		1	μs
t_{RIM}	Delay to Reset Interrupt from $\overline{\text{DISTR}}$/DISTR (RD MSR)	100pF Load		1	μs

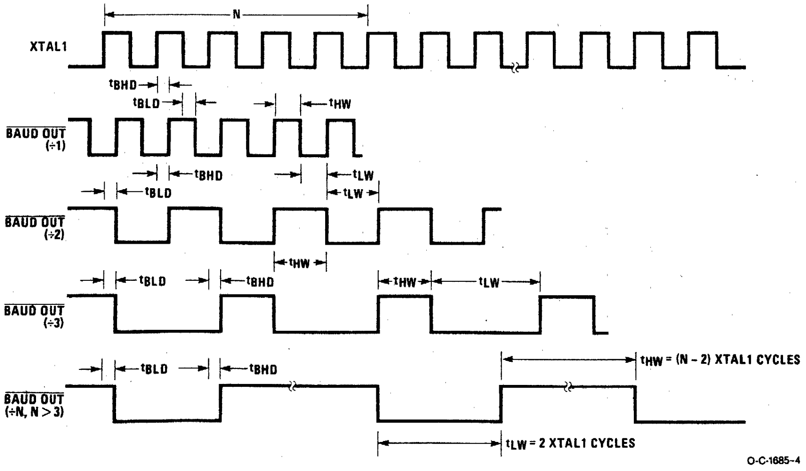
Timing Waveforms



External Clock Input (3.1 MHz Max.)

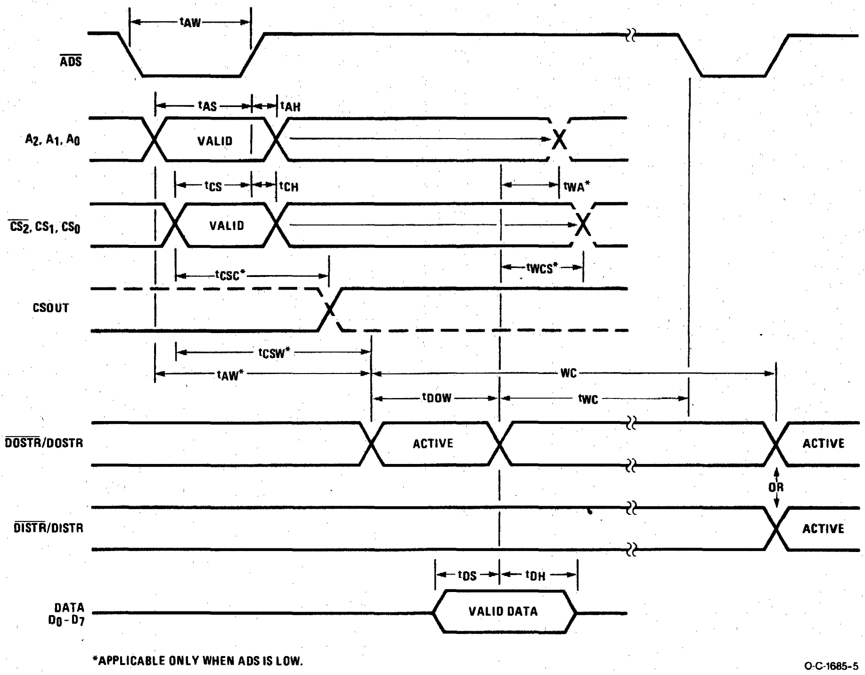


AC Test Points

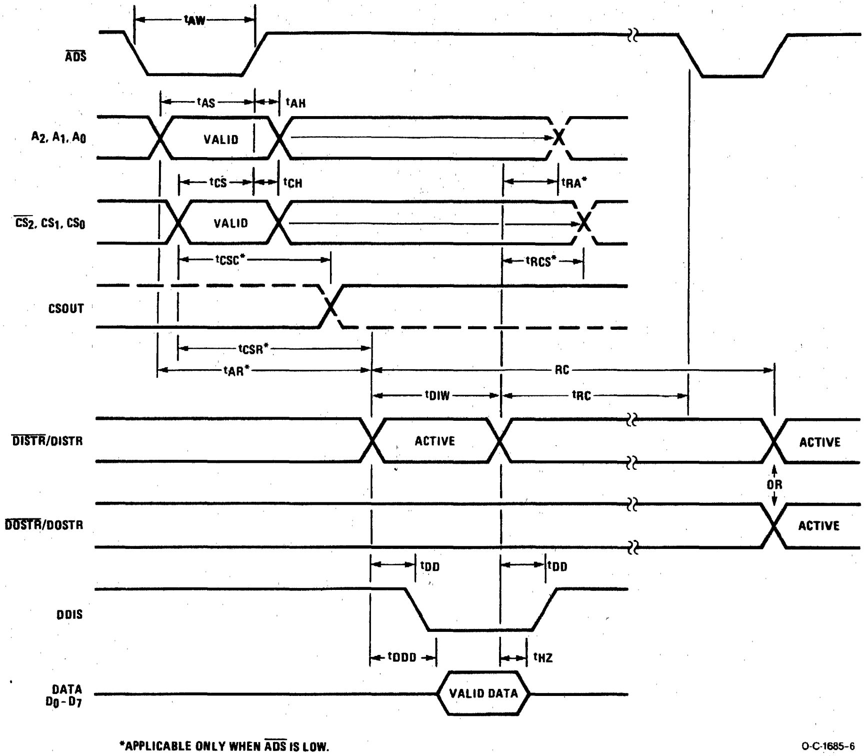


$\overline{\text{BAUDOUT}}$ Timing

Timing Waveforms (Continued)

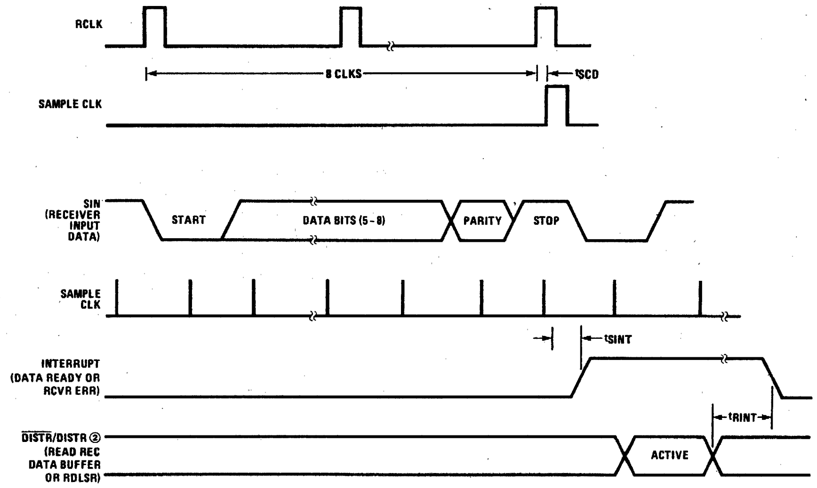


Write Cycle



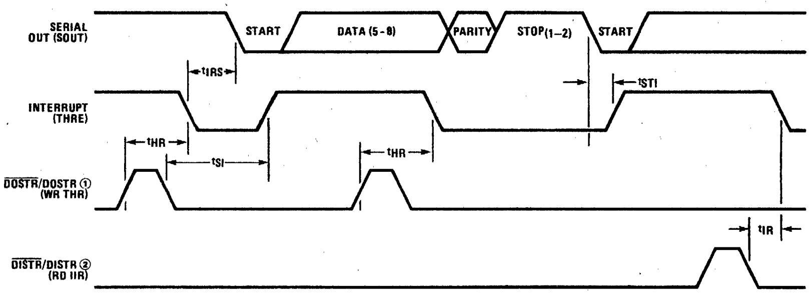
Read Cycle

Timing Waveforms (Continued)

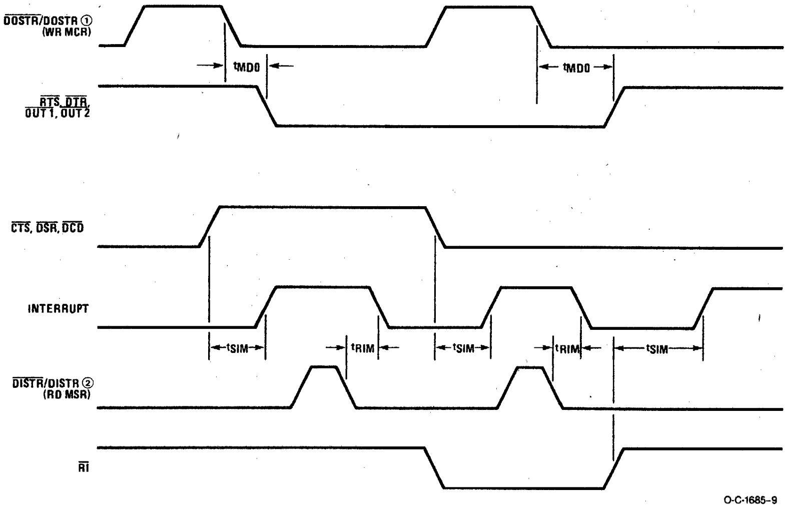


RCLK
8 CLKS
tSCD
SAMPLE CLK
SIN (RECEIVER INPUT DATA)
START
DATA BITS (5-8)
PARITY
STOP
SAMPLE CLK
tSINT
INTERRUPT (DATA READY OR RCVR ERR)
tRINT
DISTR/DISTR ② (READ REC DATA BUFFER OR RDLSR)
ACTIVE
O-C-1685-7

Receiver Timing



Transmitter Timing

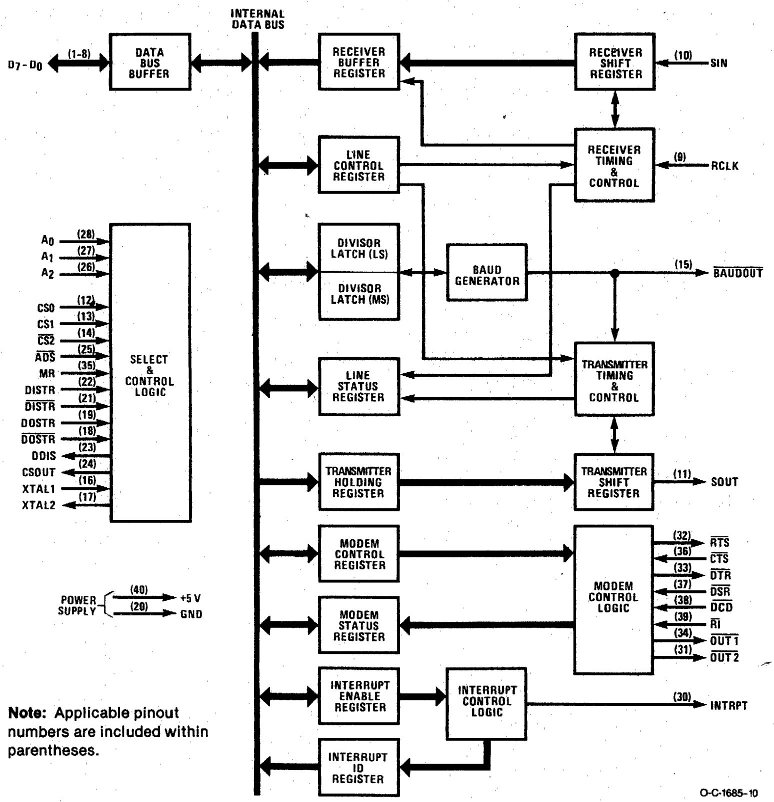


MODEM Controls Timing

Note 1: See Write Cycle Timing

Note 2: See Read Cycle Timing

Block Diagram


INTERNAL DATA BUS
D7-D0
(1-8)
DATA BUS BUFFER
RECEIVER BUFFER REGISTER
RECEIVER SHIFT REGISTER
(10)
SIN
LINE CONTROL REGISTER
RECEIVER TIMING & CONTROL
(9)
RCLK
A0 (28)
A1 (27)
A2 (26)
CS0 (12)
CS1 (13)
$\overline{\text{CS2}}$ (14)
$\overline{\text{ADS}}$ (25)
MR (35)
DISTR (22)
$\overline{\text{DISTR}}$ (21)
DOSTR (19)
$\overline{\text{DOSTR}}$ (18)
DDIS (23)
CSOUT (24)
XTAL1 (16)
XTAL2 (17)
SELECT & CONTROL LOGIC
DIVISOR LATCH (LS)
DIVISOR LATCH (MS)
BAUD GENERATOR
(15)
$\overline{\text{BAUDOUT}}$
LINE STATUS REGISTER
TRANSMITTER TIMING & CONTROL
TRANSMITTER HOLDING REGISTER
TRANSMITTER SHIFT REGISTER
(11)
SOUT
MODEM CONTROL REGISTER
MODEM CONTROL LOGIC
(32) $\overline{\text{RTS}}$
(36) $\overline{\text{CTS}}$
(33) $\overline{\text{DTR}}$
(37) $\overline{\text{DSR}}$
(38) $\overline{\text{DCD}}$
(39) $\overline{\text{RI}}$
(34) $\overline{\text{OUT1}}$
(31) $\overline{\text{OUT2}}$
POWER SUPPLY
(40) +5 V
(20) GND
MODEM STATUS REGISTER
INTERRUPT ENABLE REGISTER
INTERRUPT CONTROL LOGIC
(30)
INTRPT
INTERRUPT ID REGISTER

Note: Applicable pinout numbers are included within parentheses.

O-C-1685-10

Functional Pin Description

The following describes the function of all INS8250A input/output pins. Some of these descriptions reference internal circuits.

Note: In the following descriptions, a low represents a logic 0 (0V nominal) and a high represents a logic 1 (+2.4V nominal).

Input Signals

Chip Select (CS0, CS1, $\overline{\text{CS2}}$), Pins 12-14: When CS0 and CS1 are high and $\overline{\text{CS2}}$ is low, the chip is selected. Chip selection is complete when the decoded chip select signal is latched with an active (low) Address Strobe ($\overline{\text{ADS}}$) input. This enables communication between the INS8250A and the CPU.

Data Input Strobe (DISTR, $\overline{\text{DISTR}}$), Pins 22 and 21: When DISTR is high or $\overline{\text{DISTR}}$ is low while the chip is selected, allows the CPU to read status information or data from a selected register of the INS8250A.

Note: Only an active DISTR or $\overline{\text{DISTR}}$ input is required to transfer data from the INS8250A during a read operation. Therefore, tie either the DISTR input permanently low or the $\overline{\text{DISTR}}$ input permanently high, if not used.

Data Output Strobe (DOSTR, $\overline{\text{DOSTR}}$), Pins 19 and 18: When DOSTR is high or $\overline{\text{DOSTR}}$ is low while the chip is selected, allows the CPU to write data or control words into a selected register of the INS8250A.

Note: Only an active DOSTR or $\overline{\text{DOSTR}}$ input is required to transfer data to the INS8250A during a write operation. Therefore, tie either the DOSTR input permanently low or the $\overline{\text{DOSTR}}$ input permanently high, if not used.

Address Strobe ($\overline{\text{ADS}}$), Pin 25: When low, provides latching for the Register Select (A0, A1, A2) and Chip Select (CS0, CS1, $\overline{\text{CS2}}$) signals.

Note: An active $\overline{\text{ADS}}$ input is required when the Register Select (A0, A1, A2) signals are not stable for the duration of a read or write operation. If not required, tie the $\overline{\text{ADS}}$ input permanently low.

Register Select (A0, A1, A2), Pins 26-28: These three inputs are used during a read or write operation to select an INS8250A register to read from or write into as indicated in the table below. Note that the state of the Divisor Latch Access Bit (DLAB), which is the most significant bit of the Line Control Register, affects the selection of certain INS8250A registers. The DLAB must be set high by the system software to access the Baud Generator Divisor Latches.

Functional Pin Description (Continued)

DLAB	A_2	A_1	A_0	Register
0	0	0	0	Receiver Buffer (read), Transmitter Holding Register (write)
0	0	0	1	Interrupt Enable
X	0	1	0	Interrupt Identification (read only)
X	0	1	1	Line Control
X	1	0	0	MODEM Control
X	1	0	1	Line Status
X	1	1	0	MODEM Status
X	1	1	1	Scratch
1	0	0	0	Divisor Latch (least significant byte)
1	0	0	1	Divisor Latch (most significant byte)

Master Reset (MR), Pin 35: This input is buffered with a TTL-compatible Schmitt Trigger with 0.5V typical hysteresis. When high, it clears all the registers (except the Receiver Buffer, Transmitter Holding, and Divisor Latches), and the control logic of the INS8250A. Also, the state of various output signals (SOUT, INTRPT, $\overline{\text{OUT 1}}$, $\overline{\text{OUT 2}}$, $\overline{\text{RTS}}$, $\overline{\text{DTR}}$) are affected by an active MR input. (Refer to Table 1.)

Receiver Clock (RCLK), Pin 9: This input is the 16 × baud rate clock for the receiver section of the chip.

Serial Input (SIN), Pin 10: Serial data input from the communications link (peripheral device, MODEM, or data set).

Clear to Send ($\overline{\text{CTS}}$), Pin 36: The $\overline{\text{CTS}}$ signal is a MODEM control function input whose conditions can be tested by the CPU by reading bit 4 (CTS) of the MODEM Status Register. Bit 0 (DCTS) of the MODEM Status Register indicates whether the $\overline{\text{CTS}}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{\text{CTS}}$ has no effect on the Transmitter.

Note: Whenever the CTS bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Data Set Ready ($\overline{\text{DSR}}$), Pin 37: When low, this indicates that the MODEM or data set is ready to establish the communications link and transfer data with the INS8250A. The $\overline{\text{DSR}}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 5 (DSR) of the MODEM Status Register. Bit 1 (DDSR) of the MODEM Status Register indicates whether the $\overline{\text{DSR}}$ input has changed state since the previous reading of the MODEM Status Register.

Note: Whenever the DSR bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Data Carrier Detect ($\overline{\text{DCD}}$), Pin 38: When low, indicates that the data carrier has been detected by the MODEM or data set. The $\overline{\text{DCD}}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 7 (DCD) of the MODEM Status Register. Bit 3 (DDCD) of the MODEM Status Register indicates whether the $\overline{\text{DCD}}$ input has changed state since the previous reading of the MODEM Status Register. $\overline{\text{DCD}}$ has no effect on the receiver.

Note: Whenever the DCD bit of the MODEM Status Register changes state, an interrupt is generated if the MODEM Status Interrupt is enabled.

Ring Indicator ($\overline{\text{RI}}$), Pin 39: When low, indicates that a telephone ringing signal has been received by the MODEM or data set. The $\overline{\text{RI}}$ signal is a MODEM-control function input whose condition can be tested by the CPU by reading bit 6 (RI) of the MODEM Status Register. Bit 2 (TERI) of the MODEM Status Register indicates whether the $\overline{\text{RI}}$ input has changed from a low to a high state since the previous reading of the MODEM Status Register.

Note: Whenever the RI bit of the MODEM Status Register changes from a high to a low state, an interrupt is generated if the MODEM Status Register is enabled.

V_{CC}, Pin 40: +5V supply.

V_{SS}, Pin 20: Ground (0V) reference.

Output Signals

Data Terminal Ready ($\overline{\text{DTR}}$), Pin 33: When low, informs the MODEM or data set that the INS8250A is ready to communicate. The $\overline{\text{DTR}}$ output signal can be set to an active low by programming bit 0 (DTR) of the MODEM Control Register to a high level. The $\overline{\text{DTR}}$ signal is set high upon a Master Reset operation. The $\overline{\text{DTR}}$ signal is forced to its inactive state (high) during loop mode operation.

Request to Send ($\overline{\text{RTS}}$), Pin 32: When low, informs the MODEM or data set that the INS8250A is ready to transmit data. The $\overline{\text{RTS}}$ output signal can be set to an active low by programming bit 1 (RTS) of the MODEM Control Register. The $\overline{\text{RTS}}$ signal is set high upon a Master Reset operation. The $\overline{\text{RTS}}$ signal is forced to its inactive state (high) during loop mode operation.

Output 1 ($\overline{\text{OUT 1}}$), Pin 34: User-designated output that can be set to an active low by programming bit 2 (OUT 1) of the MODEM Control Register to a high level. The $\overline{\text{OUT 1}}$ signal is set high upon a Master Reset Operation. The $\overline{\text{OUT 1}}$ signal is forced to its inactive state (high) during loop mode operation.

Output 2 ($\overline{\text{OUT 2}}$), Pin 31: User-designated output that can be set to an active low by programming bit 3 (OUT 2) of the MODEM Control Register to a high level. The $\overline{\text{OUT 2}}$ signal is set high upon a Master Reset Operation. The $\overline{\text{OUT 2}}$ signal is forced to its inactive state (high) during loop mode operation.

Chip Select Out (CSOUT), Pin 24: When high, indicates that the chip has been selected by active, CS0, CS1, and $\overline{\text{CS2}}$ inputs. No data transfer can be initiated until the CSOUT signal is a logic 1.

Driver Disable (DDIS), Pin 23: Goes low whenever the CPU is reading data from the INS8250A. A high-level DDIS output can be used to disable an external transceiver (if used between the CPU and INS8250A on the D_7–D_0 Data Bus) at all times, except when the CPU is reading data.

Functional Pin Description (Continued)

Baud Out ($\overline{\text{BAUDOUT}}$), Pin 15: 16 × clock signal for the transmitter section of the INS8250A. The clock rate is equal to the main reference oscillator frequency divided by the specified divisor in the Baud Generator Divisor Latches. The $\overline{\text{BAUDOUT}}$ may also be used for the receiver section by tying this output to the RCLK input of the chip.

Interrupt (INTRPT), Pin 30: Goes high whenever any one of the following interrupt types has an active high condition and is enabled via the IER: Receiver Error Flag; Received Data Available; Transmitter Holding Register Empty; and MODEM Status. The INTRPT signal is reset low upon the appropriate interrupt service or a Master Reset operation.

Serial Output (SOUT), Pin 11: Composite serial data output to the communications link (peripheral, MODEM or data set). The SOUT signal is set to the Marking (logic 1) state upon a Master Reset operation.

Input/Output Signals

Data (D_7-D_0) Bus, Pins 1-8: This bus comprises eight TRI-STATE input/output lines. The bus provides bidirectional communications between the INS8250A and the CPU. Data, control words, and status information are transferred via the D_7-D_0 Data Bus.

External Clock Input/Output (XTAL 1, XTAL 2) Pins 16 and 17: These two pins connect the main timing reference (crystal or signal clock) to the INS8250A.

Connection Diagram

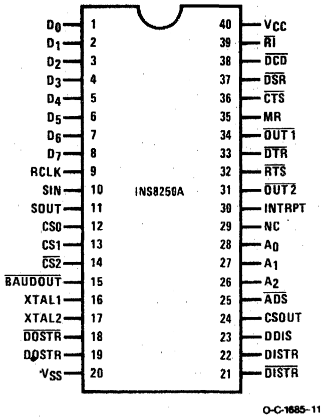

D0 1
D1 2
D2 3
D3 4
D4 5
D5 6
D6 7
D7 8
RCLK 9
SIN 10
SOUT 11
CS0 12
CS1 13
$\overline{\text{CS2}}$ 14
$\overline{\text{BAUDOUT}}$ 15
XTAL1 16
XTAL2 17
$\overline{\text{DOSTR}}$ 18
DOSTR 19
VSS 20
INS8250A
40 VCC
39 $\overline{\text{RI}}$
38 $\overline{\text{DCD}}$
37 $\overline{\text{DSR}}$
36 $\overline{\text{CTS}}$
35 MR
34 $\overline{\text{OUT1}}$
33 $\overline{\text{DTR}}$
32 $\overline{\text{RTS}}$
31 $\overline{\text{OUT2}}$
30 INTRPT
29 NC
28 A0
27 A1
26 A2
25 $\overline{\text{ADS}}$
24 CSOUT
23 DDIS
22 DISTR
21 $\overline{\text{DISTR}}$
0-C-1685-11

Table 1. ACE Reset Functions

Register/Signal	Reset Control	Reset State
Interrupt Enable Register	Master Reset	All Bits Low (0-3 forced and 4-7 permanent)
Interrupt Identification Register	Master Reset	Bit 0 is High, Bits 1 and 2 Low Bits 3-7 are Permanently Low
Line Control Register	Master Reset	All Bits Low
MODEM Control Register	Master Reset	All Bits Low
Line Status Register	Master Reset	All Bits Low, Except Bits 5 and 6 are High
MODEM Status Register	Master Reset	Bits 0-3 Low Bits 4-7 — Input Signal
SOUT	Master Reset	High
INTRPT (RCVR Errs)	Read LSR/MR	Low
INTRPT (RCVR Data Ready)	Read RBR/MR	Low
INTRPT (THRE)	Read IIR/Write THR/MR	Low
INTRPT (Modem Status Changes)	Read MSR/MR	Low
$\overline{\text{OUT 2}}$	Master Reset	High
$\overline{\text{RTS}}$	Master Reset	High
$\overline{\text{DTR}}$	Master Reset	High
$\overline{\text{OUT 1}}$	Master Reset	High

Accessible Registers

The system programmer may access or control any of the INS8250A registers summarized in Table 2 via the CPU. These registers are used to control INS8250A operations and to transmit and receive data.

Line Control Register

The system programmer specifies the format of the asynchronous data communications exchange via the Line Control Register. In addition to controlling the format, the programmer may retrieve the contents of the Line Control Register for inspection. This feature simplifies system programming and eliminates the need for separate storage in system memory of the line characteristics. The contents of the Line Control Register are indicated in Table 2 and are described below.

Bits 0 and 1: These two bits specify the number of bits in each transmitted or received serial character. The encoding of bits 0 and 1 is as follows:

Bit 1	Bit 0	Word Length
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

Bit 2: This bit specifies the number of Stop bits in each transmitted character. If bit 2 is a logic 0, one Stop bit is generated in the transmitted data. If bit 2 is a logic 1 when a 5-bit word length is selected via bits 0 and 1, one

Table 2. Summary of INS8250A Accessible Registers

	Register Address										
	0 DLAB = 0	0 DLAB = 0	1 DLAB = 0	2	3	4	5	6	7	0 DLAB = 1	1 DLAB = 1
Bit No.	Receiver Buffer Register (Read Only)	Transmitter Holding Register (Write Only)	Interrupt Enable Register	Interrupt Ident. Register (Read Only)	Line Control Register	MODEM Control Register	Line Status Register	MODEM Status Register	Scratch Register	Divisor Latch (LS)	Latch (MS)
	RBR	THR	IER	IIR	LCR	MCR	LSR	MSR	SCR	DLL	DLM
0	Data Bit 0*	Data Bit 0	Enable Received Data Available Interrupt (ERBFI)	"0" if Interrupt Pending	Word Length Select Bit 0 (WLS0)	Data Terminal Ready (DTR)	Data Ready (DR)	Delta Clear to Send (DCTS)	Bit 0	Bit 0	Bit 8
1	Data Bit 1	Data Bit 1	Enable Transmitter Holding Register Empty Interrupt (ETBEI)	Interrupt ID Bit (0)	Word Length Select Bit 1 (WLS1)	Request to Send (RTS)	Overrun Error (OE)	Delta Data Set Ready (DDSR)	Bit 1	Bit 1	Bit 9
2	Data Bit 2	Data Bit 2	Enable Receiver Line Status Interrupt (ELSI)	Interrupt ID Bit (1)	Number of Stop Bits (STB)	Out 1	Parity Error (PE)	Trailing Edge Ring Indicator (TERI)	Bit 2	Bit 2	Bit 10
3	Data Bit 3	Data Bit 3	Enable MODEM Status Interrupt (EDSSI)	0	Parity Enable (PEN)	Out 2	Framing Error (FE)	Delta Data Carrier Detect (DDCD)	Bit 3	Bit 3	Bit 11
4	Data Bit 4	Data Bit 4	0	0	Even Parity Select (EPS)	Loop	Break Interrupt (BI)	Clear to Send (CTS)	Bit 4	Bit 4	Bit 12
5	Data Bit 5	Data Bit 5	0	0	Stick Parity	0	Transmitter Holding Register (THRE)	Data Set Ready (DSR)	Bit 5	Bit 5	Bit 13
6	Data Bit 6	Data Bit 6	0	0	Set Break	0	Transmitter Empty (TEMT)	Ring Indicator (RI)	Bit 6	Bit 6	Bit 14
7	Data Bit 7	Data Bit 7	0	0	Divisor Latch Access Bit (DLAB)	0	0	Data Carrier Detect (DCD)	Bit 7	Bit 7	Bit 15

*Bit 0 is the least significant bit. It is the first bit serially transmitted or received.

Accessible Registers (Continued)

and a half Stop bits are generated. If bit 2 is a logic 1 when either a 6-, 7-, or 8-bit word length is selected, two Stop bits are generated. The Receiver checks the first Stop-bit only, regardless of the number of Stop bits selected.

Bit 3: This bit is the Parity Enable bit. When bit 3 is a logic 1, a Parity bit is generated (transmit data) or checked (receive data) between the last data word bit and Stop bit of the serial data. (The Parity bit is used to produce an even or odd number of 1s when the data word bits and the Parity bit are summed.)

Bit 4: This bit is the Even Parity Select bit. When bit 3 is a logic 1 and bit 4 is a logic 0, an odd number of logic 1s is transmitted or checked in the data word bits and Parity bit. When bit 3 is a logic 1 and bit 4 is a logic 1, an even number of logic 1s is transmitted or checked.

Bit 5: This bit is the Stick Parity bit. When bit 3 is a logic 1 and bit 5 is a logic 1, the Parity bit is transmitted and checked by the receiver as a logic 0 if bit 4 is a logic 1 or as a logic 1 if bit 4 is a logic 0.

Bit 6: This bit is the Break Control bit. When it is set to a logic 1, the serial output (SOUT) is forced to the Spacing (logic 0) state. The break is disabled by setting bit 6 to a logic 0. The Break Control bit acts only on SOUT and has no effect on the transmitter logic.

Note: This feature enables the CPU to alert a terminal in a computer communications system. If the following sequence is followed, no erroneous or extraneous characters will be transmitted because of the break.

1. Load an all 0s pad character in response to THRE.
2. Set break in response to the next THRE.
3. Wait for the transmitter to be idle, (TEMT = 1), and clear break when normal transmission has to be restored.

During the break, the Transmitter can be used as a character timer to accurately establish the break duration.

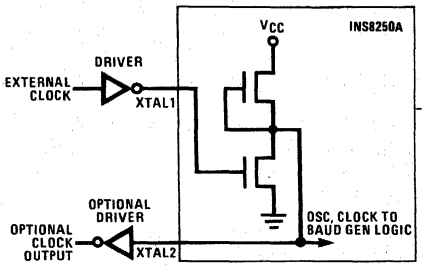
Bit 7: This bit is the Divisor Latch Access Bit (DLAB). It must be set high (logic 1) to access the Divisor Latches of the Baud Generator during a Read or Write operation. It must be set low (logic 0) to access the Receiver Buffer, the Transmitter Holding Register, or the Interrupt Enable Register.

Table 3. Baud Rates Using 1.8432MHz Crystal

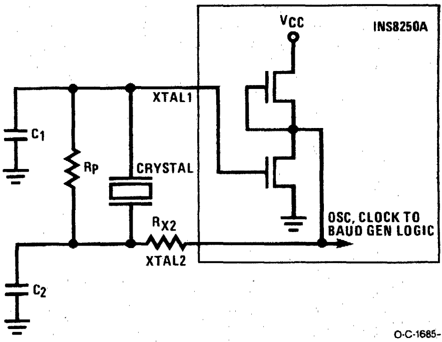
Desired Baud Rate	Divisor Used to Generate 16 × Clock	Percent Error Difference Between Desired and Actual
50	2304	—
75	1536	—
110	1047	0.026
134.5	857	0.058
150	768	—
300	384	—
600	192	—
1200	96	—
1800	64	—
2000	58	0.69
2400	48	—
3600	32	—
4800	24	—
7200	16	—
9600	12	—
19200	6	—
38400	3	—
56000	2	2.86

Note: 1.8432MHz is the standard 8080 frequency divided by 10.

Typical Clock Circuits



O-C-1685-12



O-C-1685-13

CRYSTAL	Rp	RX2	C1	C2
3.1MHz	1MΩ	1.5k	10-30pF	40-60pF
1.8MHz	1MΩ	1.5k	10-30pF	40-60pF

Typical Crystal Oscillator Network

Table 4. Baud Rates Using 3.072MHz Crystal

Desired Baud Rate	Divisor Used to Generate 16 × Clock	Percent Error Difference Between Desired and Actual
50	3840	—
75	2560	—
110	1745	0.026
134.5	1428	0.034
150	1280	—
300	640	—
600	320	—
1200	160	—
1800	107	0.312
2000	96	—
2400	80	—
3600	53	0.628
4800	40	—
7200	27	1.23
9600	20	—
19200	10	—
38400	5	—

Programmable Baud Generator

The INS8250A contains a programmable Baud Generator that is capable of taking any clock input (DC to 3.1 MHz) and dividing it by any divisor from 1 to $2^{16}-1$). The output frequency of the Baud Generator is 16× the Baud [divisor # = (frequency input) ÷ (baud rate × 16)]. Two 8-bit latches store the divisor in a 16-bit binary format. These Divisor Latches must be loaded during initialization in order to ensure desired operation of the Baud Generator. Upon loading either of the Divisor Latches, a 16-bit Baud counter is immediately loaded. This prevents long counts on initial load.

Tables 3 and 4 illustrate the use of the Baud Generator with crystal frequencies of 1.8432MHz and 3.072MHz respectively. For baud rates of 38400 and below, the error obtained is minimal. The accuracy of the desired baud rate is dependent on the crystal frequency chosen.

Note: The maximum operating frequency of the Baud Generator is 3.1MHz. However, when using divisors of 3 and below, the maximum frequency is equal to the divisor in MHz. For example, if the divisor is 1, then the maximum frequency is 1MHz. In no case should the data rate be greater than 56k Baud.

Line Status Register

This 8-bit register provides status information to the CPU concerning the data transfer. The contents of the Line Status Register are indicated in Table 2 and are described below.

Bit 0: This bit is the receiver Data Ready (DR) indicator. Bit 0 is set to a logic 1 whenever a complete incoming character has been received and transferred into the Receiver Buffer Register. Bit 0 is reset to a logic 0 by reading the data in the Receiver Buffer Register.

Bit 1: This bit is the Overrun Error (OE) indicator. Bit 1 indicates that data in the Receiver Buffer Register was not read by the CPU before the next character was transferred into the Receiver Buffer Register, thereby destroying the previous character. The OE indicator is reset whenever the CPU reads the contents of the Line Status Register.

Bit 2: This bit is the Parity Error (PE) indicator. Bit 2 indicates that the received data character does not have the correct even or odd parity, as selected by the even-parity-select bit. The PE bit is set to a logic 1 upon detection of a parity error and is reset to a logic 0 whenever the CPU reads the contents of the Line Status Register.

Bit 3: This bit is the Framing Error (FE) indicator. Bit 3 indicates that the received character did not have a valid Stop bit. Bit 3 is set to a logic 1 whenever the Stop bit following the last data bit or parity bit is detected as a zero bit (Spacing level). The FE indicator is reset whenever the CPU reads the contents of the Line Status indicator.

Bit 4: This bit is the Break Interrupt (BI) indicator. Bit 4 is set to a logic 1 whenever the received data input is held in the Spacing (logic 0) state for longer than a full word transmission time (that is, the total time of Start bit + data bits + Parity + Stop bits). The BI indicator is reset whenever the CPU reads the contents of the Line Status indicator.

Note: Bits 1 through 4 are the error conditions that produce a Receiver Line Status interrupt whenever any of the corresponding conditions are detected.

Bit 5: This bit is the Transmitter Holding Register Empty (THRE) indicator. Bit 5 indicates that the INS8250A is ready to accept a new character for transmission. In addition, this bit causes the INS8250A to issue an interrupt to the CPU when the Transmit Holding Register Empty Interrupt enable is set high. The THRE bit is set to a logic 1 when a character is transferred from the Transmitter Holding Register into the Transmitter Shift Register. The bit is reset to logic 0 concurrently with the loading of the Transmitter Holding Register by the CPU.

Bit 6: This bit is the Transmitter Empty (TEMT) indicator. Bit 6 is set to a logic 1 whenever the Transmitter Holding Register (THR) and the Transmitter Shift Register (TSR) are both empty. It is reset to logic 0 whenever either the THR or TSR contains a data character.

Bit 7: This bit is permanently set to logic 0.

Note: The Line Status Register is intended for read operations only. Writing to this register is not recommended as this operation is used for factory testing.

Table 5. Interrupt Control Functions

Interrupt Identification Register			Interrupt Set and Reset Functions			
Bit 2	**Bit 1**	**Bit 0**	**Priority Level**	**Interrupt Type**	**Interrupt Source**	**Interrupt Reset Control**
0	0	1	—	None	None	—
1	1	0	Highest	Receiver Line Status	Overrun Error or Parity Error or Framing Error or Break Interrupt	Reading the Line Status Register
1	0	0	Second	Received Data Available	Receiver Data Available	Reading the Receiver Buffer Register
0	1	0	Third	Transmitter Holding Register Empty	Transmitter Holding Register Empty	Reading the IIR Register (if source of interrupt) or Writing into the Transmitter Holding Register
0	0	0	Fourth	MODEM Status	Clear to Send or Data Set Ready or Ring Indicator or Data Carrier Detect	Reading the MODEM Status Register

Interrupt Identification Register

The INS8250A has an on-chip interrupt capability that allows for complete flexibility in interfacing to all the popular microprocessors presently available. In order to provide minimum software overhead during data character transfers, the INS8250A prioritizes interrupts into four levels. The four levels of interrupt conditions are as follows: Receiver Line Status (priority 1); Received Data Ready (priority 2); Transmitter Holding Register Empty (priority 3); and MODEM Status (priority 4).

Information indicating that a prioritized interrupt is pending and the type of that interrupt are stored in the Interrupt Identification Register (IIR). When addressed during chip-select time, the IIR freezes the highest priority interrupt pending and no other interrupts are acknowledged until the particular interrupt is serviced by the CPU. The contents of the IIR are indicated in Table 2 and are described below.

Bit 0: This bit can be used in either a hardwired prioritized or polled environment to indicate whether an interrupt is pending. When bit 0 is a logic 0, an interrupt is pending and the IIR contents may be used as a pointer to the appropriate interrupt service routine. When bit 0 is a logic 1, no interrupt is pending and polling (if used) continues.

Bits 1 and 2: These two bits of the IIR are used to identify the highest priority interrupt pending as indicated in Table 5.

Bits 3 through 7: These five bits of the IIR are always logic 0.

Interrupt Enable Register

This 8-bit register enables the four types of interrupts of the INS8250A to separately activate the chip Interrupt (INTRPT) output signal. It is possible to totally disable the interrupt system by resetting bits 0 through 3 of the Interrupt Enable Register. Similarly, by setting the appropriate bits of this register to a logic 1, selected interrupts can be enabled. Disabling the interrupt system inhibits the Interrupt Identification Register and the active (high) INTRPT output from the chip. All other system functions operate in their normal manner, including the setting of the Line Status and MODEM Status Registers. The contents of the Interrupt Enable Register are indicated in Table 2 and are described below.

Bit 0: This bit enables the Received Data Available Interrupt when set to logic 1.

Bit 1: This bit enables the Transmitter Holding Register Empty Interrupt when set to logic 1.

Bit 2: This bit enables the Receiver Line Status Interrupt when set to logic 1.

Bit 3: This bit enables the MODEM Status Interrupt when set to logic 1.

Bits 4 through 7: These four bits are always logic 0.

Modem Control Register

This 8-bit register controls the interface with the MODEM or data set (or a peripheral device emulating a MODEM). The contents of the MODEM Control Register are indicated in Table 2 and are described below.

Bit 0: This bit controls the Data Terminal Ready ($\overline{\text{DTR}}$) output. When bit 0 is set to a logic 1, the $\overline{\text{DTR}}$ output is forced to a logic 0. When bit 0 is reset to a logic 0, the $\overline{\text{DTR}}$ output is forced to a logic 1.

Note: The $\overline{\text{DTR}}$ output of the INS8250A may be applied to an EIA inverting line driver (such as the DS1488) to obtain the proper polarity input at the succeeding MODEM or data set.

Bit 1: This bit controls the Request to Send ($\overline{\text{RTS}}$) output. Bit 1 affects the $\overline{\text{RTS}}$ output in a manner identical to that described above for bit 0.

Bit 2: This bit controls the Output 1 ($\overline{\text{OUT 1}}$) signal, which is an auxiliary user-designated output. Bit 2 affects the $\overline{\text{OUT 1}}$ output in a manner identical to that described above for bit 0.

Bit 3: This bit controls the Output 2 ($\overline{\text{OUT 2}}$) signal, which is an auxiliary user-designated output. Bit 3 affects the $\overline{\text{OUT 2}}$ output in a manner identical to that described above for bit 0.

Bit 4: This bit provides a local loopback feature for diagnostic testing of the INS8250A. When bit 4 is set to logic 1, the following occur: the transmitter Serial Output (SOUT) is set to the Marking (logic 1) state; the receiver Serial Input (SIN) is disconnected; the output of the Transmitter Shift Register is "looped back" into the Receiver Shift Register input; the four MODEM Control inputs ($\overline{\text{CTS}}$, $\overline{\text{DSR}}$, $\overline{\text{DCD}}$, and $\overline{\text{RI}}$) are disconnected; and the four MODEM Control outputs ($\overline{\text{DTR}}$, $\overline{\text{RTS}}$, $\overline{\text{OUT 1}}$, and $\overline{\text{OUT 2}}$) are internally connected to the four MODEM Control inputs, and the MODEM Control output pins are forced to their inactive state (high). In the diagnostic mode, data that is transmitted is immediately received. This feature allows the processor to verify the transmit- and receive-data paths of the INS8250A.

In the diagnostic mode, the receiver and transmitter interrupts are fully operational. The MODEM Control Interrupts are also operational, but the interrupts' sources are now the lower four bits of the MODEM Control Register instead of the four MODEM Control inputs. The interrupts are still controlled by the Interrupt Enable Register.

Bits 5 through 7: These bits are permanently set to logic 0.

Modem Status Register

This 8-bit register provides the current state of the control lines from the MODEM (or peripheral device) to the CPU. In addition to this current-state information, four bits of the MODEM Status Register provide change information. These bits are set to a logic 1 whenever a control input from the MODEM changes state. They are reset to logic 0 whenever the CPU reads the MODEM Status Register.

The contents of the MODEM Status Register are indicated in Table 2 and are described below.

Bit 0: This bit is the Delta Clear to Send (DCTS) indicator. Bit 0 indicates that the $\overline{\text{CTS}}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 1: This bit is the Delta Data Set Ready (DDSR) indicator. Bit 1 indicates that the $\overline{\text{DSR}}$ input to the chip has changed state since the last time it was read by the CPU.

Bit 2: This bit is the Trailing Edge of Ring Indicator (TERI) detector. Bit 2 indicates that the $\overline{\text{RI}}$ input to the chip has changed from an On (logic 1) to an Off (logic 0) condition.

Bit 3: This bit is the Delta Data Carrier Detect (DDCD) indicator. Bit 3 indicates that the $\overline{\text{DCD}}$ input to the chip has changed state.

Note: Whenever bit 0, 1, 2, or 3 is set to logic 1, a MODEM Status interrupt is generated.

Bit 4: This bit is the complement of the Clear to Send ($\overline{\text{CTS}}$) input. If bit 4 (loop) of the MCR is set to a 1, this bit is equivalent to RTS in the MCR.

Bit 5: This bit is the complement of the Data Set Ready ($\overline{\text{DSR}}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent of DTR in the MCR.

Bit 6: This bit is the complement of the Ring Indicator ($\overline{\text{RI}}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 1 in the MCR.

Bit 7: This bit is the complement of the Data Carrier Detect ($\overline{\text{DCD}}$) input. If bit 4 of the MCR is set to a 1, this bit is equivalent to OUT 2 of the MCR.

Scratchpad Register: This 8-bit Read/Write Register does not control the ACE in any way. It is intended as a scratchpad register to be used by the programmer to hold data temporarily.

Typical Applications

Figures 1 and 2 show how to use the INS8250A chip in an INS8080A system and in a microcomputer system with a high-capacity data bus.

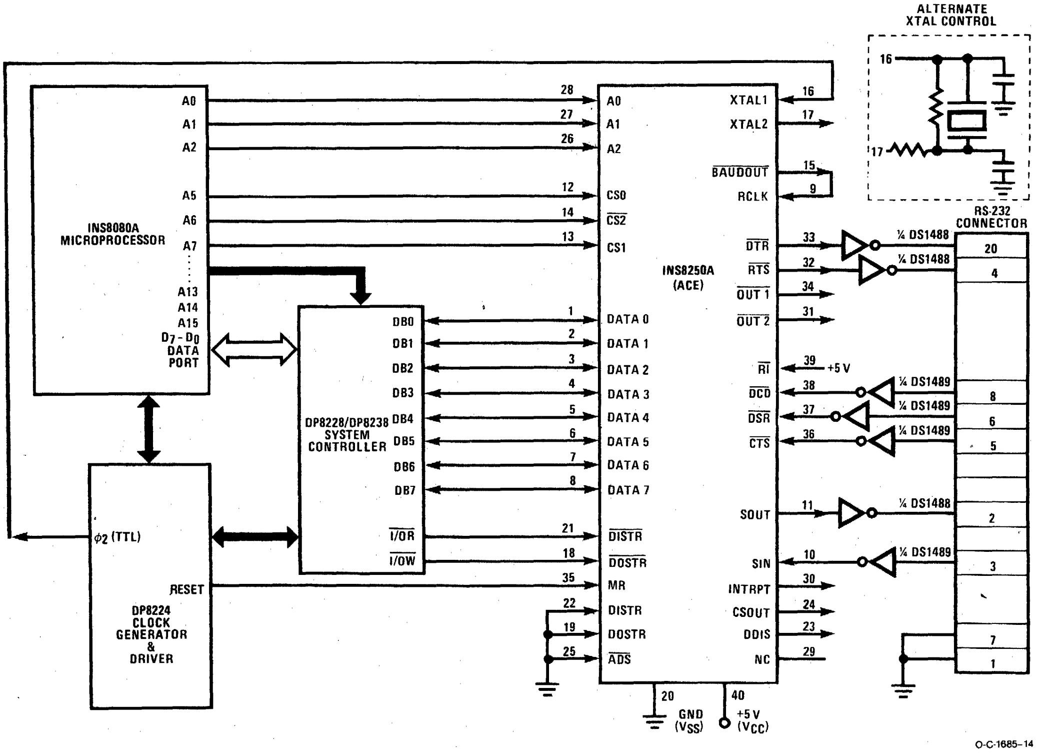


FIGURE 1. Typical INS8080A/INS8250 RS232 Terminal Interface

Typical Applications (Continued)

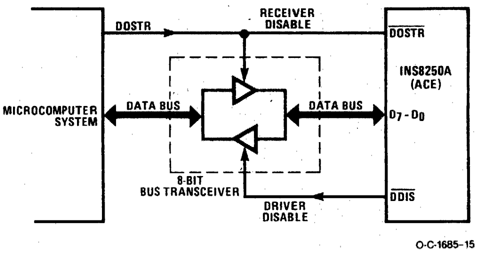


FIGURE 2. Typical Interface for a High-Capacity Data Bus

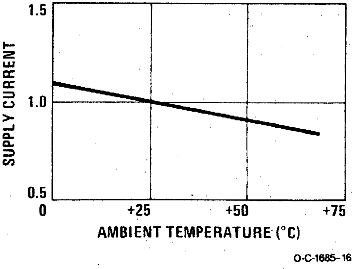
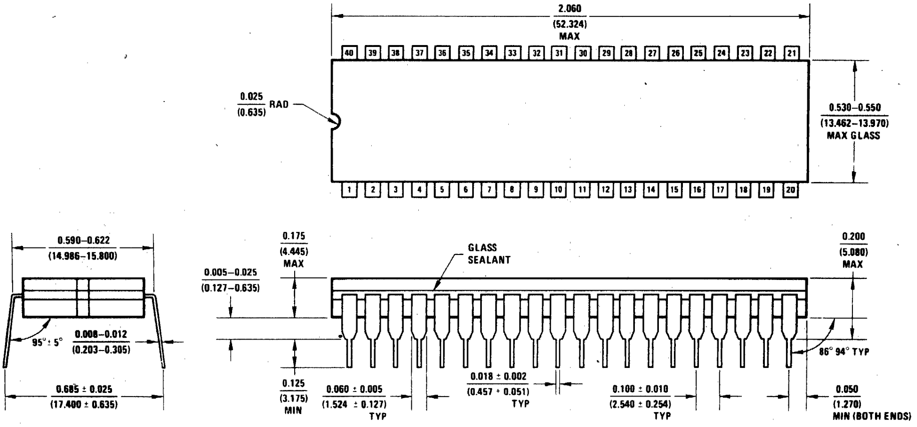
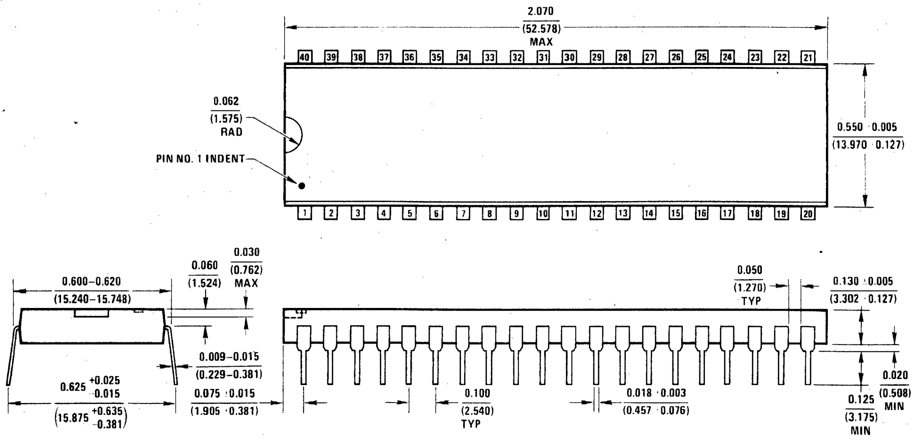


FIGURE 3. Typical Supply Current vs. Temperature, Normalized

Physical Dimensions inches (millimeters)



Ceramic Dual-In-Line Package (J)
Order Number INS8250AJ
NS Package Number J40B



Plastic Dual-In-Line Package (N)
Order Number INS8250AN
NS Package Number N40X

National Semiconductor

PRELIMINARY

NS16201-6 Timing Control Unit

General Description

The NS16201 Timing Control Unit (TCU) is a 24-pin device fabricated on a Schottky bipolar process. It provides the 2 phase MOS clock drivers, system control logic (read, write, and data buffer enable) and cycle extension logic for the NS16000 family.

A crystal or external signal may be used as the 2× frequency source. Besides the 2 phase MPU clock outputs (PHI1 and PHI2), there are two other clock outputs (TTL-compatible) available for system timing use. One of these is a fast clock (FCLK), at twice the MPU clock frequency (i.e, at the crystal frequency). The other is a TTL version of PHI1 (CTTL).

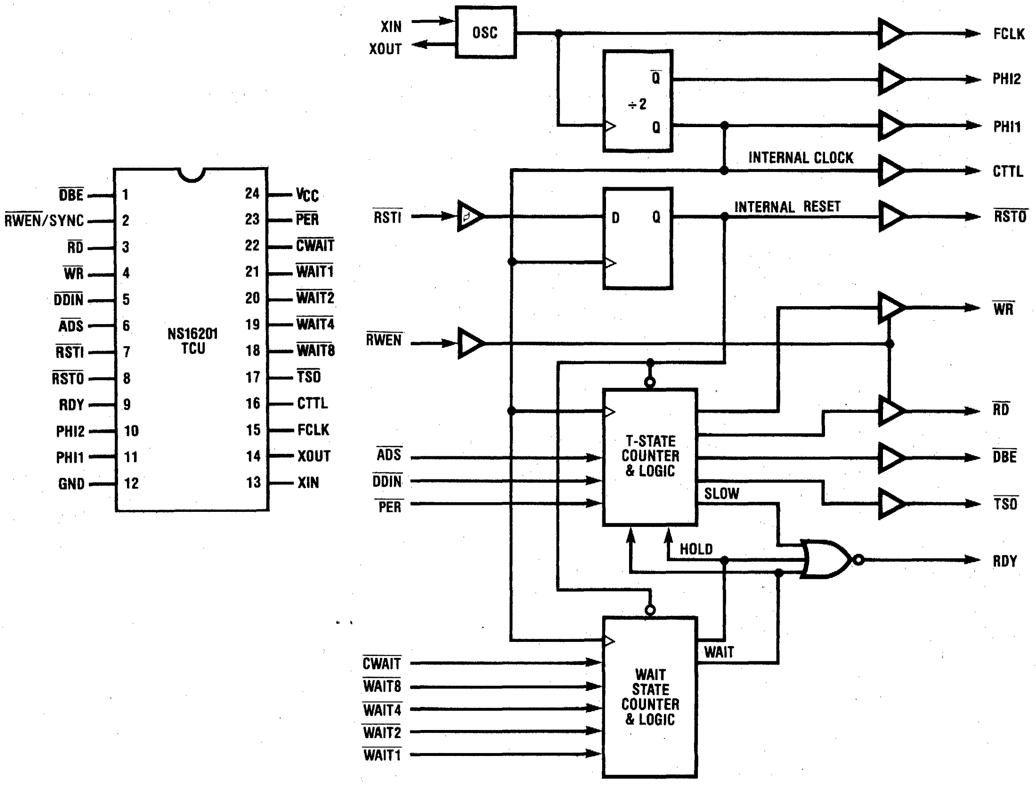
The cycle extension features include:

- Digitally programmable wait state inputs ($\overline{\text{WAITn}}$)
- Peripheral (slow) cycle to accommodate slower MOS peripheral interface ICs (where just adding wait states is not adequate)
- Cycle Hold between the first (T1) and second (T2) timing states to allow additional time for arbitration prior to generating control signals.

Features

- 2 phase full V_{CC} swing high capacitance clock drivers
- 4-bit input ($\overline{\text{WAITn}}$) allowing precise specification of from 0 to 15 wait states
- Cycle Hold for system arbitration and/or memory refresh
- System timing (CTTL and FCLK) and control ($\overline{\text{RD}}$, $\overline{\text{WR}}$, and $\overline{\text{DBE}}$) outputs
- General purpose Timing State Output ($\overline{\text{TSO}}$) that identifies internal states
- Support of slow MOS peripheral interface ICs (e.g., 8080 series)
- Provides "ready" (RDY) output for NS16000 MPUs
- Synchronous system reset generation from Schmitt trigger input
- Single 5V power supply
- 24-pin dual-in-line package

NS16201 TCU Connection and Block Diagrams



NS16201 Functional Pin Descriptions

Supplies

V_{CC} (24): +5V Power Input

GND (12): 0V Power Return

A 0.1μF ceramic decoupling capacitor must be connected across V_{CC} and GND as close to the device as possible (i.e., with least amount of lead and trace resistance). This is not only good layout practice, but necessary due to high current trânsients delivered by the PHI1 and PHI2 outputs.

Clock Pins

XIN (13): Crystal or External Frequency Source Input. The desired MPU clock (PHI1 and PHI2) frequency will be half that of the crystal or external source. A Schottky series gate ($V_{OH(min)}$ = 2.7V) is recommended to drive XIN ($V_{threshold}$ = 2.5V) as the external frequency source.

XOUT (14): Crystal Feedback Output. This output is used in crystal operation only. It must be left open when driving XIN with an external frequency source.

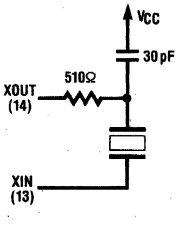
Crystal Oscillator Characteristics

The NS16201 has a "Pierce"-type oscillator that requires a parallel resonant crystal. Connections of the crystal and bias components to XIN and XOUT are shown below. It is important that the crystal be mounted in close proximity to the XIN and XOUT pins to keep printed circuit trace lengths to an absolute minimum.

Typical Crystal Specifications:

Type	At-Cut
Tolerance	0.005% at 25°C
Stability	0.01% from 0 to 70°C
Resonance	Fundamental (parallel)
Capacitance	20pF
Maximum Series Resistance	50Ω

Crystal Connection Diagram



FCLK (15): Fast Clock Output. This is a TTL-level clock output at the same frequency as the crystal or external frequency source. Consequently its frequency is twice that of the MPU clocks.

PHI1(11) and PHI2(10): MPU Clock Outputs. These outputs provide the NS16000 series MPUs with 2-phase non-overlapping MOS (full V_{CC} swing, high capacitive drive) clock signals. Their frequency will be half that of the crystal or external source.

CTTL (16): TTL System Clock. This is a TTL output version of PHI1. Therefore, it operates at the MPU clock frequency.

Control Pins

$\overline{\text{RSTI}}$ (7) and $\overline{\text{RSTO}}$ (8): Reset Input and Reset Output. $\overline{\text{RSTI}}$ is a Schmitt trigger input that will generate the synchronous system reset signal, $\overline{\text{RSTO}}$. $\overline{\text{RSTO}}$'s rising edge is synchronized to PHI1 to satisfy the MPU's requirements. $\overline{\text{RSTI}}$ going low will cause $\overline{\text{RSTO}}$ to go low, signaling a system reset. When the slow ramping rising edge (via external RC combination) of $\overline{\text{RSTI}}$ reaches the Schmitt trigger threshold, $\overline{\text{RSTO}}$ will go high on the next rising edge of PHI1.

$\overline{\text{ADS}}$ (6): Address Strobe Input from MPU/MMU. This input (going low) identifies the first timing state (T1) of a bus cycle for the TCU (see Note (1)).

$\overline{\text{DDIN}}$ (5): Data Direction Input. This signal determines whether a Write (when high) or a Read (when low) cycle will be performed. It is connected to the MPU's $\overline{\text{DDIN}}$ output pin in a normal configuration.

$\overline{\text{RD}}$ (3): Read Output Strobe (TRI-STATE®). This is an active low signal that identifies a Read cycle. It is decoded from $\overline{\text{DDIN}}$ and tri-stated by $\overline{\text{RWEN}}$.

$\overline{\text{WR}}$ (4): Write Output Strobe (TRI-STATE). This is an active low signal that identifies a Write cycle. It is decoded from $\overline{\text{DDIN}}$ and tri-stated by $\overline{\text{RWEN}}$.

$\overline{\text{RD}}$ and $\overline{\text{WR}}$ are mutually exclusive (active low) in any cycle.

$\overline{\text{RWEN}}$ (2): Read/Write Enable Input. This input tri-states the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ outputs when high and enables them when low.

NOTE: For compatibility with future revisions of the TCU, this pin should be held low while reset is active.

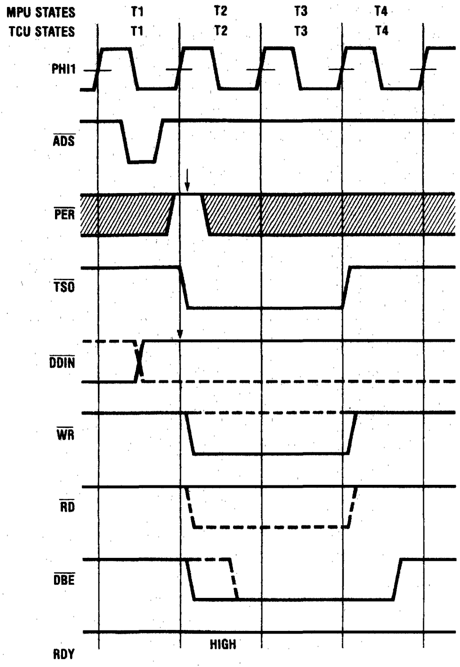
$\overline{\text{DBE}}$ (1): Data Buffer Enable Output. This signal is used to enable or tri-state buffers on the data lines of an NS16000 system. It is low when the buffers are to be enabled.

$\overline{\text{TSO}}$ (17): Timing State Output. The falling edge of $\overline{\text{TSO}}$ signals the start of the second T-state (T2) for the TCU (see Note 1)). The rising edge identifies the start of the last T-state (T4) of a cycle.

$\overline{\text{TSO}}$ is a general purpose signal that may be used by external logic for synchronizing to a bus cycle. One possible application is to use $\overline{\text{TSO}}$ to generate a $\overline{\text{RAS}}$ signal for dynamic RAMs. Another is to gate $\overline{\text{CWAIT}}$ for selection of the Cycle Hold or Wait State functions (see $\overline{\text{CWAIT}}$ description).

$\overline{\text{PER}}$ (23): Peripheral Cycle Input. This input causes the TCU to insert five wait states into a normal bus cycle and reshape the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ signals. This satisfies the $\overline{\text{RD}}$ and $\overline{\text{WR}}$ setup and hold times required by many slower MOS peripheral interface chips. This 9-state Peripheral Cycle will be referred to as a Slow Cycle throughout this data sheet. The normal 4-state bus cycle will be referred to as a Fast Cycle.

The TCU will perform a Slow Cycle when $\overline{\text{PER}}$ is low and a Fast Cycle when high. *Figures 1* and *2* contrast the difference in timing between the Fast and Slow Cycles.



Notes:

1. The MPU and TCU view some timing states (T-states) differently. For clarity, references to T-states will sometimes be followed by (TCU) or (MPU). (MPU) also implies (MMU).
2. Arrows indicate when the TCU samples the input.
3. RWEN is assumed low (RD and WR enabled) unless specified differently.
4. For clarity, T-states for both the TCU and MPU are shown above the diagrams. (See Note 1.)

FIGURE 1. Fast Cycle

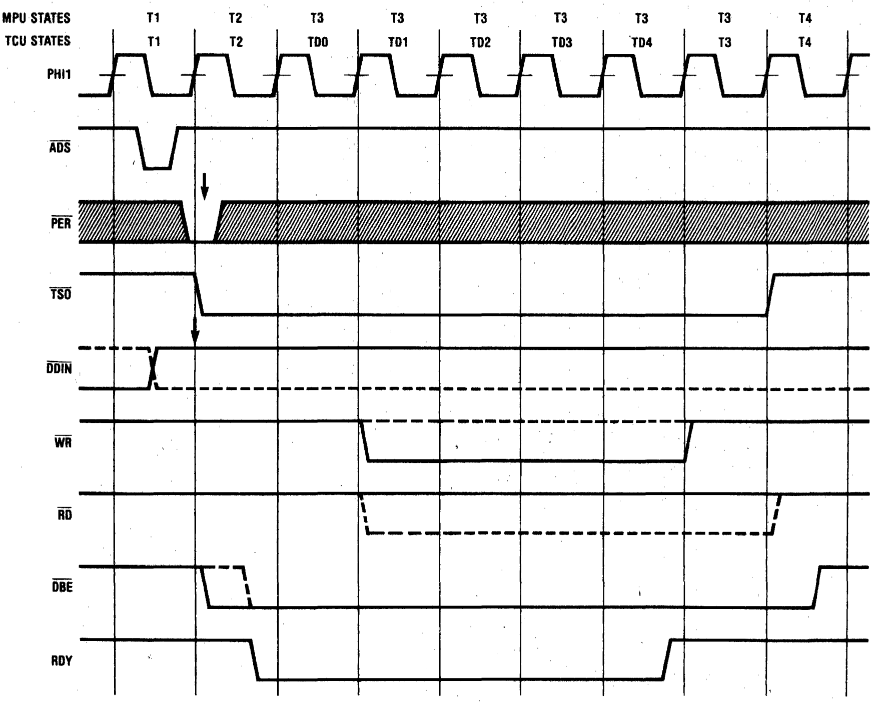


FIGURE 2. Slow Cycle

Cycle Extension Pins

RDY (9): Ready Output. This signal will go low as long as wait states are to be inserted in a bus cycle. It is normally connected to the RDY input of the MPU.

There are three basic cycle extension modes provided by the TCU. These are:

1. Slow Cycle. This mode is also known as the Peripheral Cycle and is used to generated appropriate RD and WR signals for slow peripheral ICs. It inherently adds five wait states identified as TD0-TD4 by the TCU (see PER pin description).

   NOTE: The Slow Cycle will be changed in future revisions of the TCU. The leading edge of both the Read and Write strobes will occur one clock cycle earlier, while the trailing edge of the Write strobe will be delayed by one clock cycle. Thus, the resulting Read and Write strobes will be five and four clock cycles wide, respectively. The Address/Data setup and hold times will also change accordingly.

2. Cycle Hold. This mode is entered when CWAIT is low at the end of T1. It prevents the TCU from entering T2 as long as CWAIT is kept low. TSO, DBE, RD, and WR are held high (inactive) until T2 is entered (see *Figures 3* and *4*).
3. Wait States. This is the normal wait state insertion mode. It is initiated by either the CWAIT (continuous wait states) or the WAIT1-WAIT8 (digitally preset wait states) inputs.

Any combination of these three modes of cycle extension may be used together.

CWAIT (22): Continuous Wait Input. This input is used to initiate the Cycle Hold (sampled at the end of T1) or (continuous) Wait State (sampled in the middle of T2 (TCU) for a Fast Cycle, or in the middle of TD2 (TCU) for a Slow Cycle) modes. In the Wait State mode, it will also cause the WAITn (WAIT1-WAIT8) inputs to be sampled and to possibly be overridden (see Overriding a Waitn Cycle).

WAIT1, WAIT2, WAIT4 and WAIT8 (21, 20, 19 and 18): 4-Bit Wait State Inputs. These inputs (collectively called WAITn) allow from 0 to 15 wait states to be specified. They are binarily weighted (as their names imply), thus if WAIT4 and WAIT1 are low (active), then five (0101_2) wait states will be inserted.

The WAITn inputs are sampled in the middle of T2 (TCU) for a Fast Cycle, in the middle of TD2 (TCU) for a Slow Cycle, or in the middle of a wait state (not Cycle Hold) when CWAIT is sampled low.

Cycle Hold Mode: *Figures 3* and *4* show a Cycle Hold with Fast and Slow Cycles, respectively. Note that the MPU views Cycle Hold as normal wait state insertions, whereas the TCU and any device using TSO, RD, and WR, or DBE view it as delaying the T2 (TCU) state. Also note that since the PER and DDIN inputs are sampled at the beginning of T2, the Cycle Hold allows additional time to set up these signals.

Basic CWAIT and WAITn Wait State Cycles: *Figures 5* and *6* show the basic wait state (Fast) cycles using CWAIT and WAITn inputs, respectively. Note that the WAITn inputs may be held low throughout the cycle since they are only sampled once in the cycle. If in all cycles the number of wait states inserted are the same, the desired WAITn input combination may be permanently fixed (hardwired).

CWAIT and WAITn Sampling: *Figure 7* shows a combination of CWAIT and WAITn inputs to illustrate when each are sampled. The rules for sampling are as follows:

1. In a Fast Cycle, CWAIT and WAITn inputs are always sampled in the middle of T2 (TCU).
2. In a Slow Cycle, they are always sampled in the middle of TD2 (TCU).
3. CWAIT is additionally always sampled in the middle of a wait state (TCW or TWn, not TH).
4. WAITn inputs are also sampled when CWAIT is sampled low during the middle of a wait state.

Note that any number of sequences of CWAIT and WAITn inputs may be exerted to extend a cycle indefinitely.

Overriding a WAITn Wait State Cycle: *Figure 8* shows how to end a WAIT11 cycle prematurely using the CWAIT input. This is accomplished by resampling a WAIT0 (all WAITn inputs high) when CWAIT is sampled low in TW3. Since the last wait state (TCW) is unavoidable, at least one wait state will be inserted using this technique.

Cycle Hold with Wait States: *Figure 9* shows a Cycle Hold with three wait states inserted using either CWAIT or WAITn inputs. Any combination of CWAIT and WAITn inputs may also be used.

Slow Cycle with Wait States: *Figure 10* shows a Slow Cycle with six wait states (one CWAIT and WAIT5) added. Note that wait states in a Slow Cycle are handled in exactly the same fashion as in a Fast Cycle with TD2 and TD3 replacing T2 and T3.

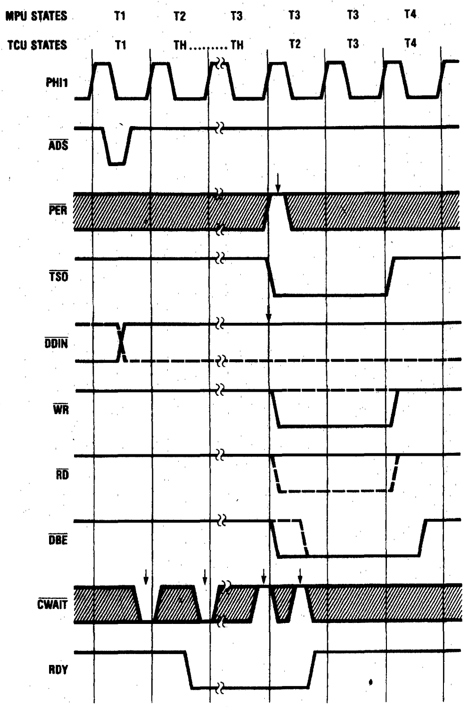


FIGURE 3. Cycle Hold of a Fast Cycle

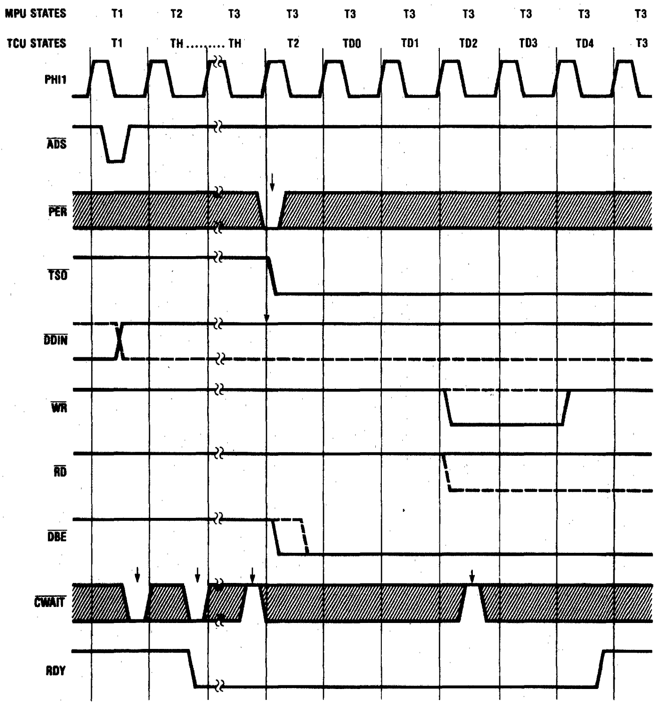


FIGURE 4. Cycle Hold of a Slow Cycle

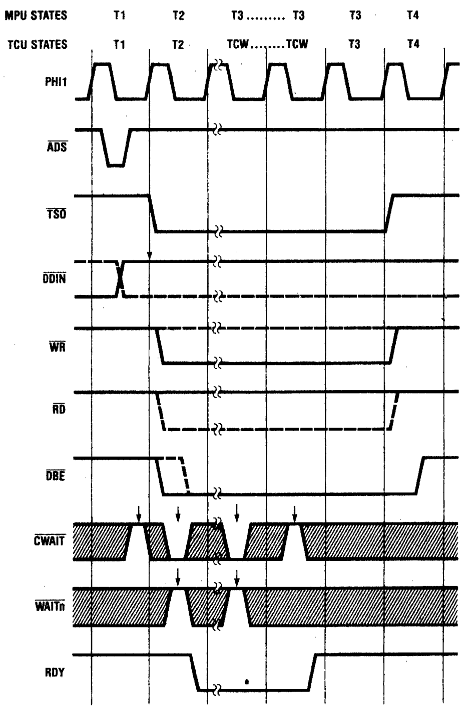


FIGURE 5. Wait State Insertion Using $\overline{\text{CWAIT}}$ Input (Fast Cycle)

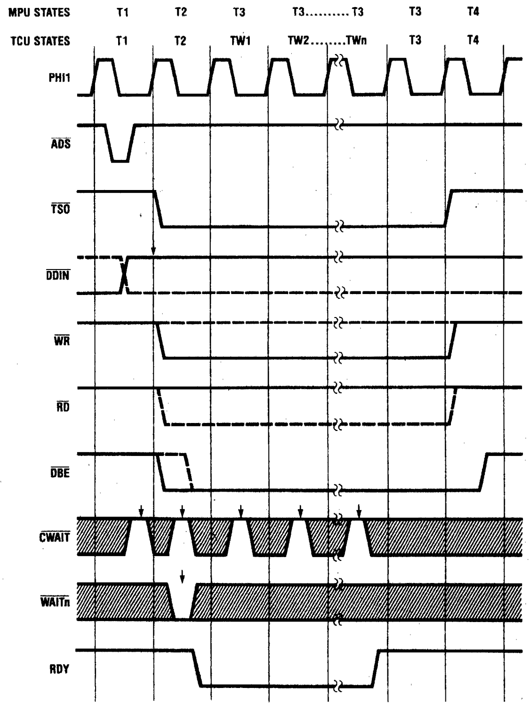


FIGURE 6. Wait State Insertion Using $\overline{\text{WAITn}}$ Inputs (Fast Cycle)

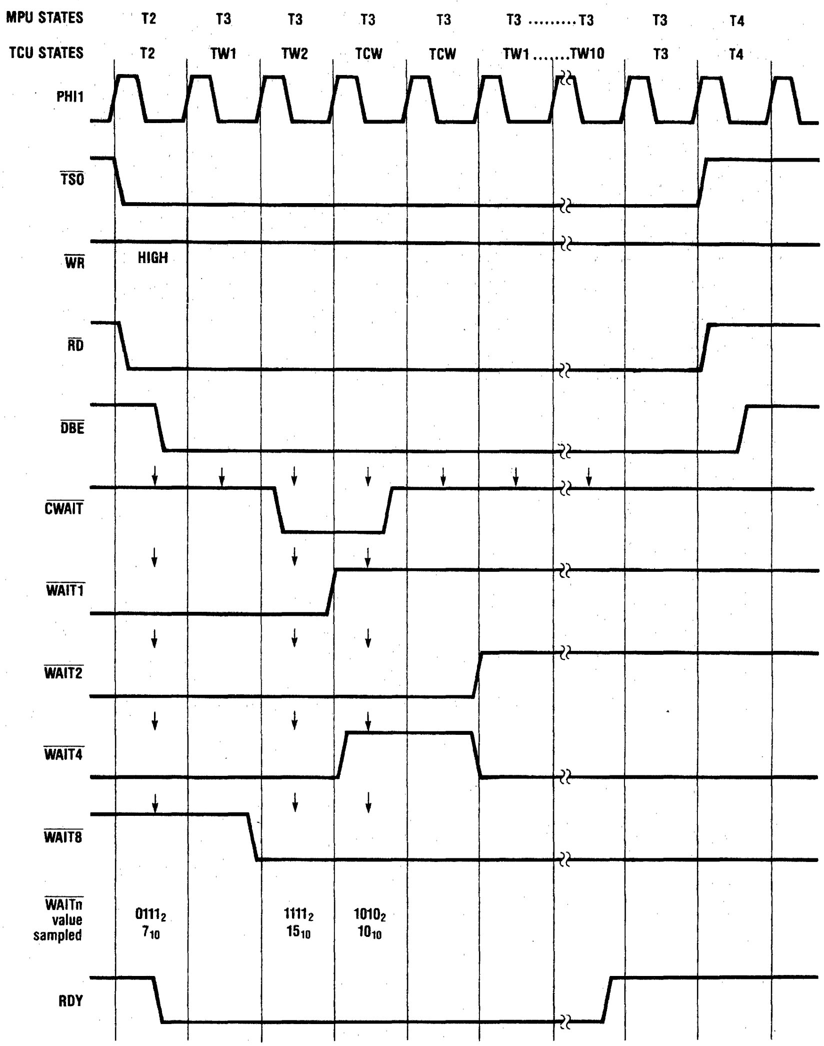


FIGURE 7. Sampling of $\overline{\text{CWAIT}}$ and $\overline{\text{WAITn}}$ Inputs (Fast Read Cycle)

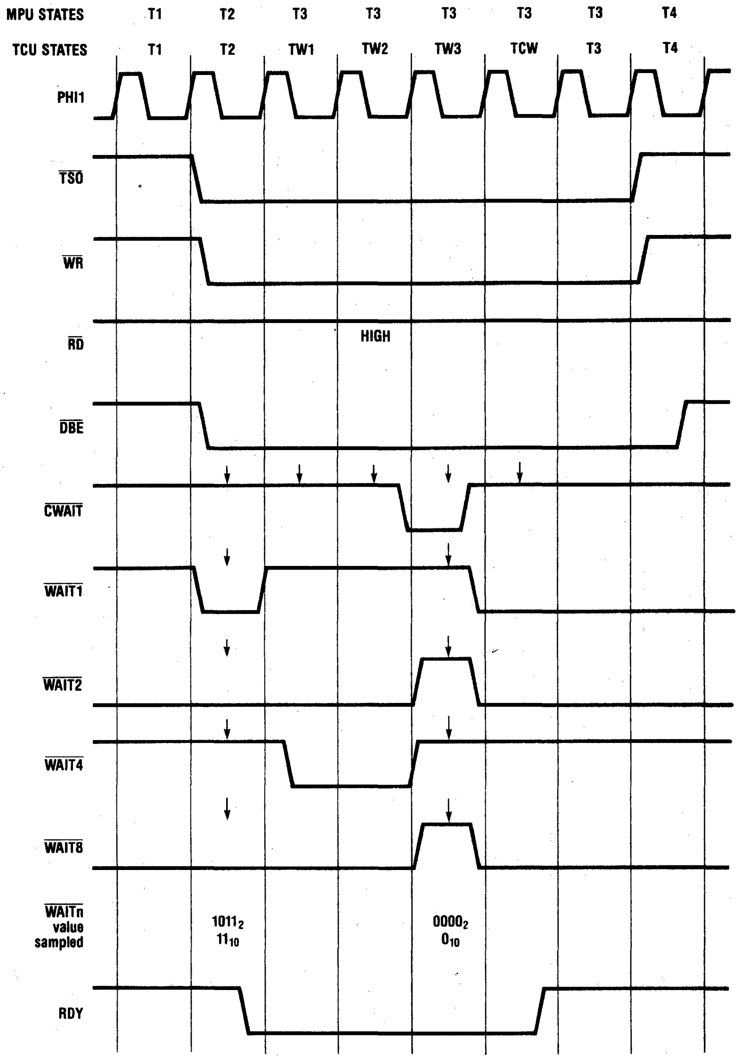


FIGURE 8. Overriding a $\overline{\text{WAIT11}}$ Cycle Using $\overline{\text{CWAIT}}$ (Fast Write Cycle)

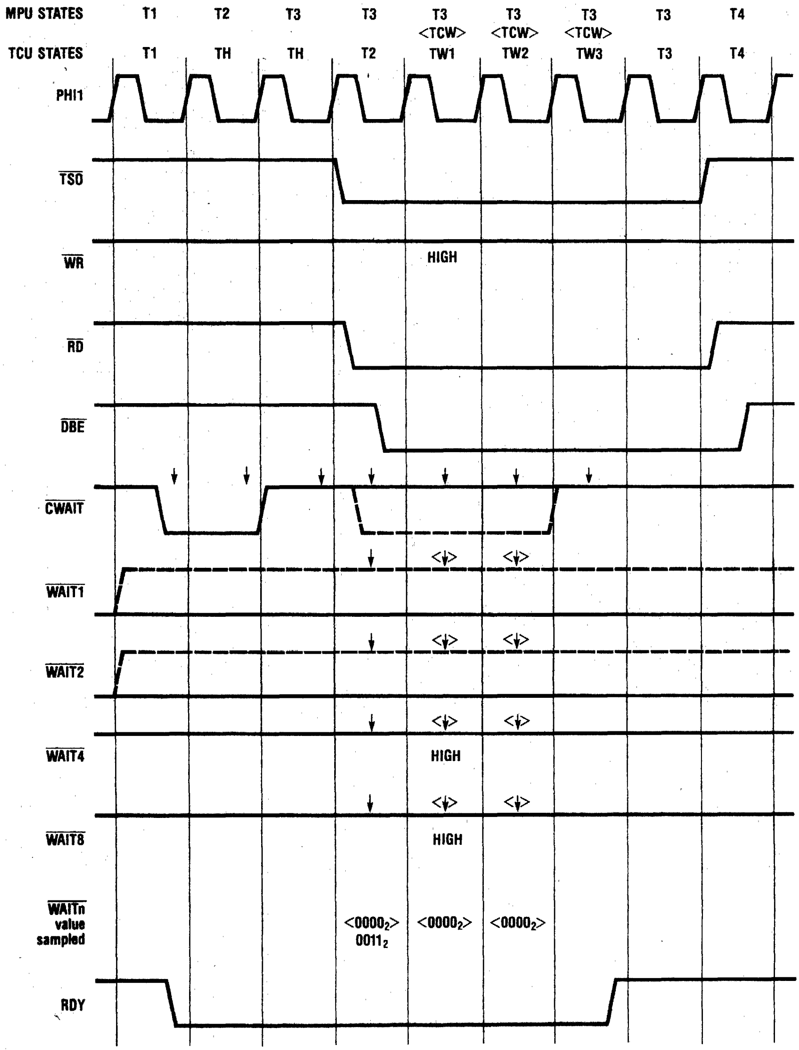


FIGURE 9. Cycle Hold with Three Wait States (either $\overline{\text{WAIT3}}$ or $\overline{\text{CWAIT}}$) (Fast Read Cycle)

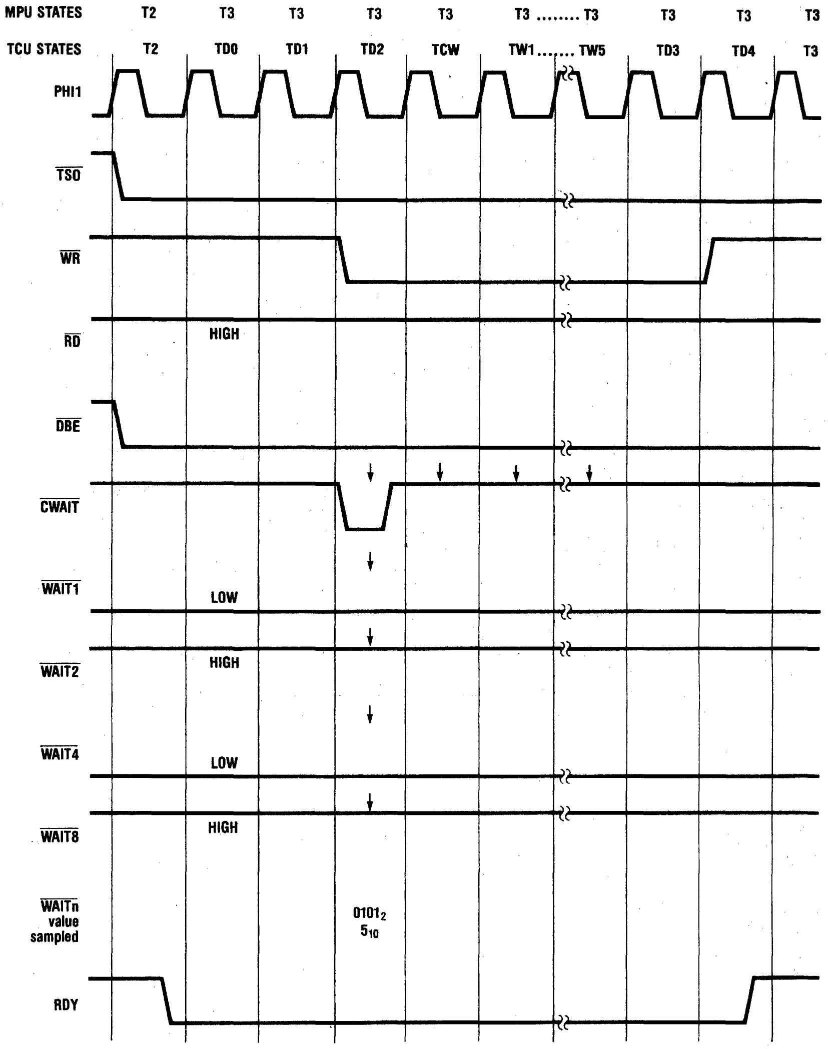


FIGURE 10. Slow Cycle with Six Wait States (1 $\overline{\text{CWAIT}}$ and $\overline{\text{WAIT5}}$) (Write Cycle)

Absolute Maximum Ratings (Note 1)

Supply Voltage	7V
Input Voltages	−1 to +5.5V
Output Voltages	−1 to +5.5V
Storage Temperature	−65°C to +150°C
Lead Temperature (Soldering, 10 sec.)	300°C
Power Dissipation	1.5W

Recommended Operating Conditions

V_{CC}	Supply Voltage	4.75V to 5.25V
T_A	Ambient Temperature	0 to 70°C

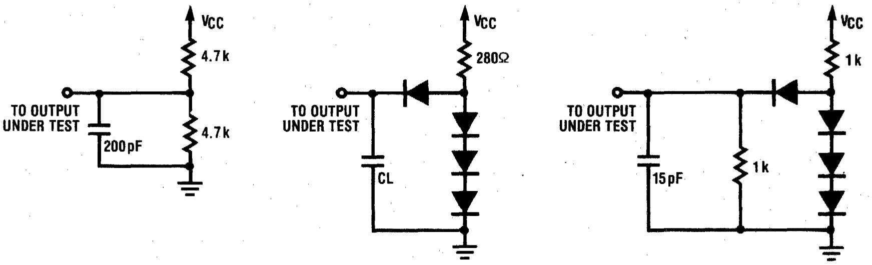
DC Electrical Characteristics: NS16201-6 (Notes 2, 3)

Symbol	Parameter	Conditions	Min.	Typ.	Max.	Units
V_{IL}	Input Low Voltage	All Inputs Except $\overline{\text{RSTI}}$ & XIN			0.8	V
V_{IH}	Input High Voltage	All Inputs Except $\overline{\text{RSTI}}$ & XIN	2			V
V_{IR}	$\overline{\text{RSTI}}$ Rising Threshold Voltage	$V_{CC} = 5V$	1.4	2.1		V
V_{HYS}	$\overline{\text{RSTI}}$ Hysteresis Voltage	$V_{CC} = 5V$	0.3	0.5		V
V_{IX}	XIN Input Threshold Voltage		0.8* Typ.	$0.5V_{CC}$	1.2* Typ.	V
I_{IL}	Input Low Current	$V_{IN} = 0.5V$ Except XIN			−500	μA
I_{IH}	Input High Current	$V_{IN} = 5.25V$ Except XIN			50	μA
V_{OL}	Output Low Voltage	PHI1 & PHI2 I = 1mA			0.3	V
		All Other Outputs Except XOUT I = 20mA			0.5	
V_{OH}	Output High Voltage	PHI1 & PHI2 I = −1mA	$V_{CC} - 0.45$			V
		All Other Outputs Except XOUT I = −1mA	2.4			
V_{CLAMP}	Input Clamp Voltage	$I_{IN} = -18mA$ Except XIN		−0.7	−1.2	V
I_{CC}	Supply Current	All Outputs High		200	280	mA

Notes:

1. "Absolute Maximum Ratings" are those values beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The table of "Recommended Operating Conditions" provides the conditions for actual device operation.
2. All currents into device pins are positive. All voltages are referenced to ground unless otherwise specified.
3. Unless otherwise specified, minimum/maximum limits apply across the supply and temperature range listed in the table of "Recommended Operating Conditions." All typical values are for $V_{CC} = 5V$ and $T_A = 25°C$.

AC Load Circuits (Notes 1, 2, 3)



LOAD 1 LOAD 2 LOAD 3

Notes:

1. Unless otherwise specified, the AC measurements are taken with the output pins in the following conditions (see AC Load Circuits):

Load 1		PHI1 and PHI2
Load 2	CL = 50pF	all TTL output except CTTL
	CL = 100pF	only CTTL
Load 3		$\overline{\text{RD}}$ and $\overline{\text{WR}}$ for TRI-STATE measurements only.

2. Load Capacitance includes probe and jig capacitance.
3. All diodes are 1N914 or equivalent.

AC Electrical Characteristics: NS16201-6 (Notes 1,2)

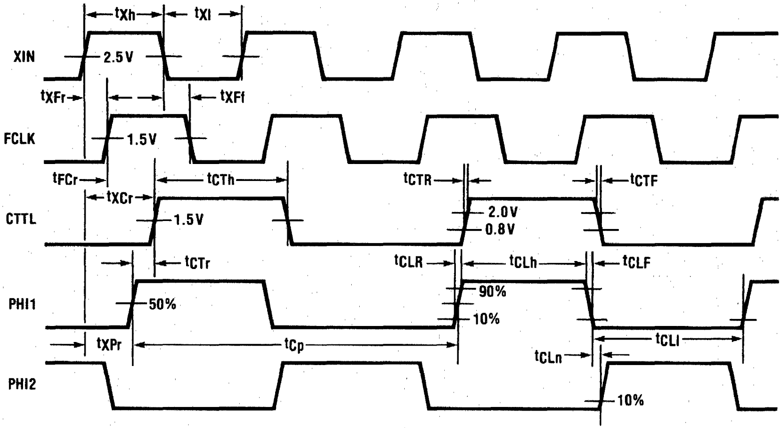
Symbol	Parameter	Definitions	Min.	Typ.	Max.	Units
Clock (XIN, FCLK, PHI1 and PHI2) Timing						
t_{Cp}	Clock Period	50% PHI1 r_E to 50% PHI1 r_E	160			ns
t_{CLh}	Clock High Time	90% PHI1 r_E to 90% PHI1 f_E	$0.5t_{Cp}$ −15ns		$0.5t_{Cp}$ −5ns	
t_{CLl}	Clock Low Time	10% PHI1 f_E to 10% PHI1 r_E	$0.5t_{Cp}$		$0.5t_{CP}$ +10ns	
t_{CLR}	Clock Rise Time	10% PHI1 r_E to V_{OH} (PHI1)	2	5	9	ns
t_{CLF}	Clock Fall Time	90% PHI1 f_E to 10% PHI1 f_E	2	4	7	ns
t_{CLn}	Clock Non-overlap Time	10% PHI1 f_E to 10% PHI2 r_E	0	2	5	ns
t_{Xh}	XIN High Time (External Input)	2.5V XIN r_E to 2.5V XIN f_E	30			ns
t_{Xl}	XIN Low Time (External Input)	2.5V XIN f_E to 2.5V XIN r_E	30			ns
t_{XFr}	XIN to FCLK r_E Delay	2.5V XIN r_E to 1.5V FCLK r_E	13	20	27	ns
t_{XFf}	XIN to FCLK f_E Delay	2.5V XIN f_E to 1.5V FCLK f_E	21	28	35	ns
t_{XCr}	XIN to CTTL r_E Delay	2.5V XIN r_E to 1.5V CTTL r_E	30	37	44	ns
t_{XPr}	XIN to PHI1 r_E Delay	2.5V XIN r_E to 50% PHI1 r_E	30	37	44	ns
t_{FCr}	FCLK to CTTL r_E Delay	1.5V FCLK r_E to 1.5V CTTL r_E	3	9	15	ns
CTTL Timing (CL = 50pF)						
t_{CTr}	PHI1 to CTTL r_E Delay	50% PHI1 r_E to 1.5V CTTL r_E	−5	−2	2	ns
t_{CTh}	CTTL High Time	1.5V CTTL r_E to 1.5V CTTL f_E	$0.5t_{Cp}$ −10ns	$0.5t_{Cp}$ −5ns	$0.5t_{Cp}$	
t_{CTR}	CTTL Rise Time	0.8V CTTL r_E to 2V CTTL r_E	2	3	5	ns
t_{CTF}	CTTL Fall Time	2V CTTL f_E to 0.8V CTTL f_E	2	3	5	ns
CTTL Timing (CL = 100pF)						
t_{CTr}	PHI1 to CTTL r_E Delay	50% PHI1 r_E to 1.5V CTTL r_E	−3	0	4	ns
t_{CTh}	CTTL High Time	1.5V CTTL r_E to 1.5V CTTL f_E	$0.5t_{Cp}$ −10ns	$0.5t_{Cp}$ −5ns	$0.5t_{Cp}$	
t_{CTR}	CTTL Rise Time	0.8V CTTL r_E to 2V CTTL r_E	3	5	7	ns
t_{CTF}	CTTL Fall Time	2V CTTL f_E to 0.8V CTTL f_E	3	5	7	ns
CTTL Timing (CL = 150pF)						
t_{CTr}	PHI1 to CTTL r_E Delay	50% PHI1 r_E to 1.5V CTTL r_E	0	3	7	ns
t_{CTh}	CTTL High Time	1.5 CTTL r_E to 1.5V CTTL f_E	$0.5t_{Cp}$ −10ns	$0.5t_{Cp}$ −5ns	$0.5t_{Cp}$	
t_{CTR}	CTTL Rise Time	0.8V CTTL r_E to 2V CTTL r_E	5	7	9	ns
t_{CTF}	CTTL Fall Time	2V CTTL f_E to 0.8V CTTL f_E	5	7	9	ns
Control Inputs ($\overline{RSTI}$, $\overline{RSTO}$, $\overline{ADS}$, $\overline{DDIN}$ and $\overline{PER}$) Timing						
t_{RSTr}	$\overline{RSTO}$ r_E Delay	50% PHI1 r_E to 1.5V $\overline{RSTO}$ r_E		10	30	ns
t_{ADs}	$\overline{ADS}$ Setup Time	1.5V $\overline{ADS}$ f_E to 50% PHI1 r_E	50			ns
t_{ADw}	$\overline{ADS}$ Pulse Width	1.5V $\overline{ADS}$ f_E to 1.5V $\overline{ADS}$ r_E	40			ns
t_{DDs}	$\overline{DDIN}$ Setup Time	1.5V $\overline{DDIN}$ to 50% PHI1 r_E	40			ns
t_{Ps}	$\overline{PER}$ Setup Time	1.5V $\overline{PER}$ to 50% PHI1 r_E	10			ns
t_{Ph}	$\overline{PER}$ Hold Time	50% PHI1 r_E to 1.5V $\overline{PER}$	20			ns

Notes:

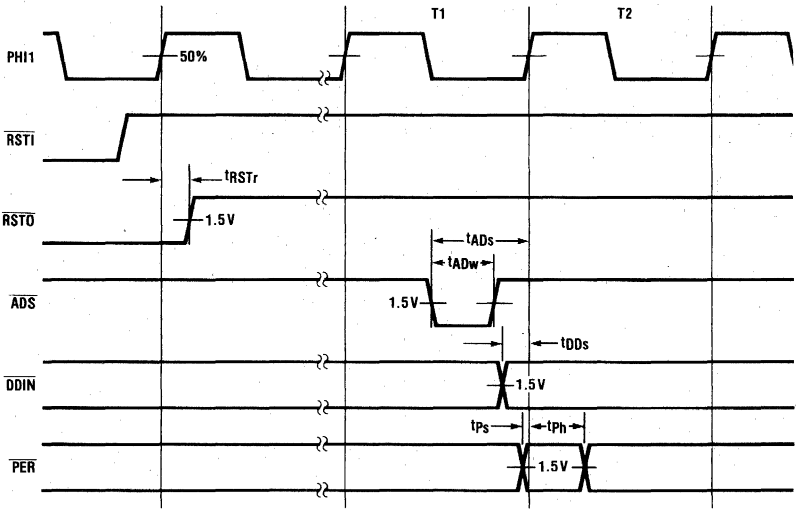
1. Unless otherwise specified, minimum/maximum limits apply across the supply and temperature range listed in the table of "Recommended Operating Conditions." All typical values are for V_{CC} = 5V and T_A = 25°C.
2. Unless otherwise specified, the AC measurements are taken with the output pins in the following conditions (see AC Load Circuits):

Load 1		PHI1 and PHI2
Load 2	CL = 50pF	all TTL output except CTTL
	CL = 100pF	only CTTL
Load 3		RD and WR for TRI-STATE measurements only.

Timing Diagrams

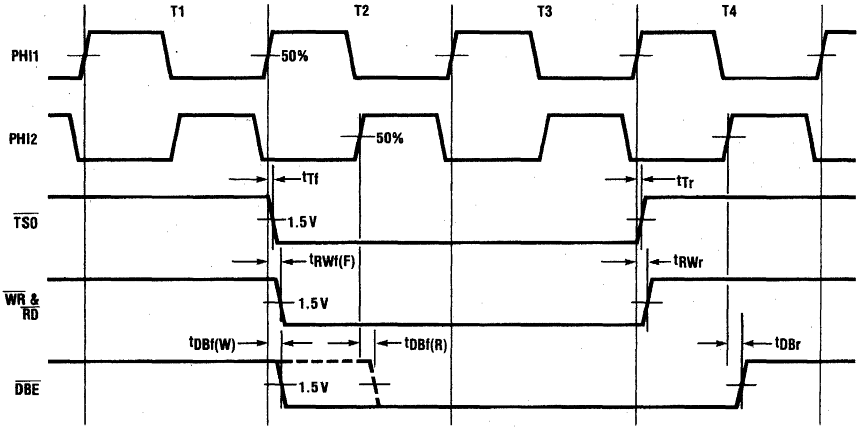


Clock and CTTL Timing Diagram

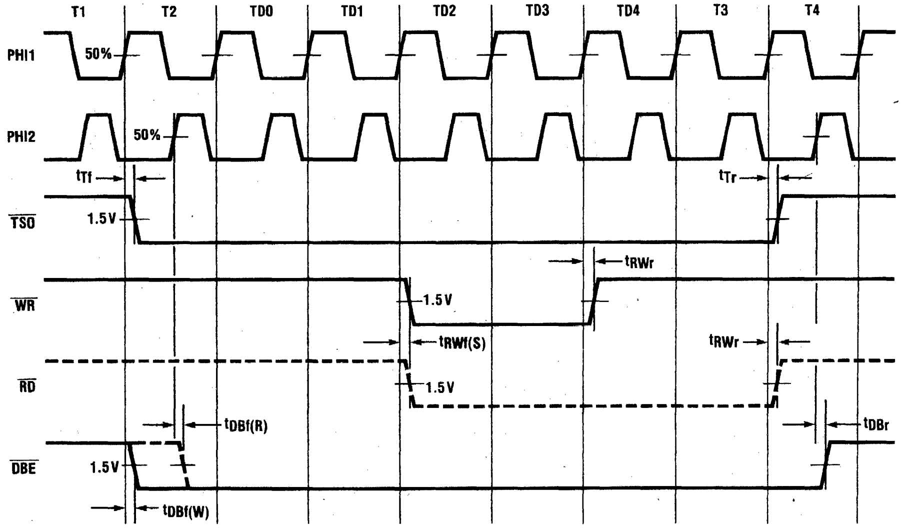


Control Inputs Timing Diagram

Timing Diagrams (Continued)



Control Outputs Timing Diagram (Fast Cycle)

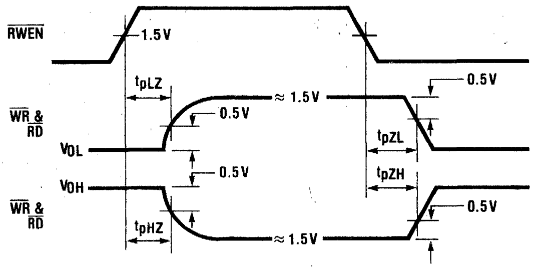


Control Outputs Timing Diagram (Slow Cycle)

AC Electrical Characteristics: NS16201-6 (continued)

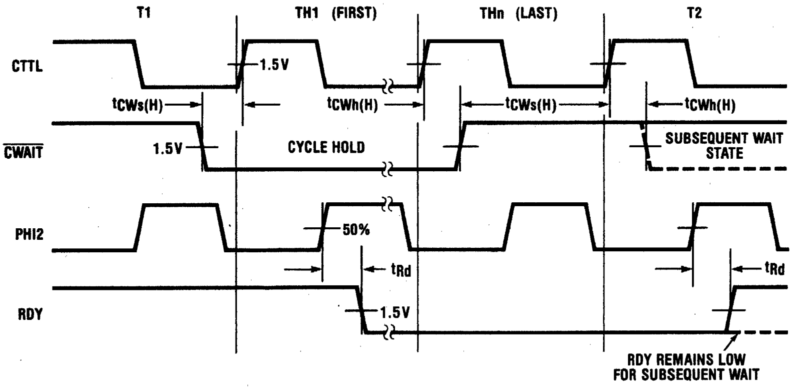
Symbol	Parameter	Definitions	Min.	Typ.	Max.	Units
Control Outputs ($\overline{\text{TSO}}$, $\overline{\text{RD}}$, $\overline{\text{WR}}$, $\overline{\text{DBE}}$ and $\overline{\text{RWEN}}$) Timing						
t_{Tf}	$\overline{\text{TSO}}$ f_E Delay	50% PHI1 r_E to 1.5V $\overline{\text{TSO}}$ f_E	0	7	15	ns
t_{Tr}	$\overline{\text{TSO}}$ r_E Delay	50% PHI1 r_E to 1.5V $\overline{\text{TSO}}$ r_E	5	12	20	ns
$t_{RWf(F)}$	$\overline{\text{RD}}/\overline{\text{WR}}$ f_E Delay (Fast Cycle)	50% PHI1 r_E to 1.5V $\overline{\text{RD}}$, $\overline{\text{WR}}$ f_E	25	35	50	ns
$t_{RWf(S)}$	$\overline{\text{RD}}/\overline{\text{WR}}$ f_E Delay (Slow Cycle)	50% PHI1 r_E to 1.5V $\overline{\text{RD}}$, $\overline{\text{WR}}$ f_E	15	25	35	ns
t_{RWr}	$\overline{\text{RD}}/\overline{\text{WR}}$ r_E Delay	50% PHI1 r_E to 1.5V $\overline{\text{RD}}$, $\overline{\text{WR}}$ r_E	10	20	30	ns
$t_{DBf(W)}$	$\overline{\text{DBE}}$ f_E Delay (Write Cycle)	50% PHI1 r_E to 1.5V $\overline{\text{DBE}}$ f_E		25	35	ns
$t_{DBf(R)}$	$\overline{\text{DBE}}$ f_E Delay (Read Cycle)	50% PHI2 r_E to 1.5V $\overline{\text{DBE}}$ f_E		25	35	ns
t_{DBr}	$\overline{\text{DBE}}$ r_E Delay	50% PHI2 r_E to 1.5V $\overline{\text{DBE}}$ r_E	10	20	30	ns
t_{PLZ}	$\overline{\text{RWEN}}$ r_E Delay	1.5V $\overline{\text{RWEN}}$ r_E to $\overline{\text{RD}}$, $\overline{\text{WR}}$ Low to HI-Z			20	ns
t_{PHZ}	$\overline{\text{RWEN}}$ r_E Delay	1.5V $\overline{\text{RWEN}}$ r_E to $\overline{\text{RD}}$, $\overline{\text{WR}}$ High to HI-Z			20	ns
t_{PZL}	$\overline{\text{RWEN}}$ f_E Delay	1.5V $\overline{\text{RWEN}}$ f_E to $\overline{\text{RD}}$, $\overline{\text{WR}}$ HI-Z to Low			35	ns
t_{PZH}	$\overline{\text{RWEN}}$ f_E Delay	1.5V $\overline{\text{RWEN}}$ f_E to $\overline{\text{RD}}$, $\overline{\text{WR}}$ HI-Z to High			35	ns
Wait State and Cycle Hold ($\overline{\text{CWAIT}}$, $\overline{\text{WAITn}}$ and RDY) Timing						
$t_{CWs(H)}$	$\overline{\text{CWAIT}}$ Setup Time (Cycle Hold)	1.5V $\overline{\text{CWAIT}}$ to 1.5V CTTL r_E	35			ns
$t_{CWh(H)}$	$\overline{\text{CWAIT}}$ Hold Time (Cycle Hold)	1.5V CTTL r_E to 1.5V $\overline{\text{CWAIT}}$	0			ns
$t_{CWs(W)}$	$\overline{\text{CWAIT}}$ Setup Time (Wait State)	1.5V $\overline{\text{CWAIT}}$ to 1.5V CTTL f_E	10			ns
$t_{CWh(W)}$	$\overline{\text{CWAIT}}$ Hold Time (Wait State)	1.5V CTTL f_E to 1.5V $\overline{\text{CWAIT}}$	20			ns
t_{Ws}	$\overline{\text{WAITn}}$ Setup Time	1.5V $\overline{\text{WAITn}}$ to 50% PHI1 f_E	5			ns
t_{Wh}	$\overline{\text{WAITn}}$ Hold Time	50% PHI1 f_E to 1.5V $\overline{\text{WAITn}}$	25			ns
t_{Rd}	RDY Delay	50% PHI2 r_E to 1.5V RDY		20	40	ns

Timing Diagrams (Continued)

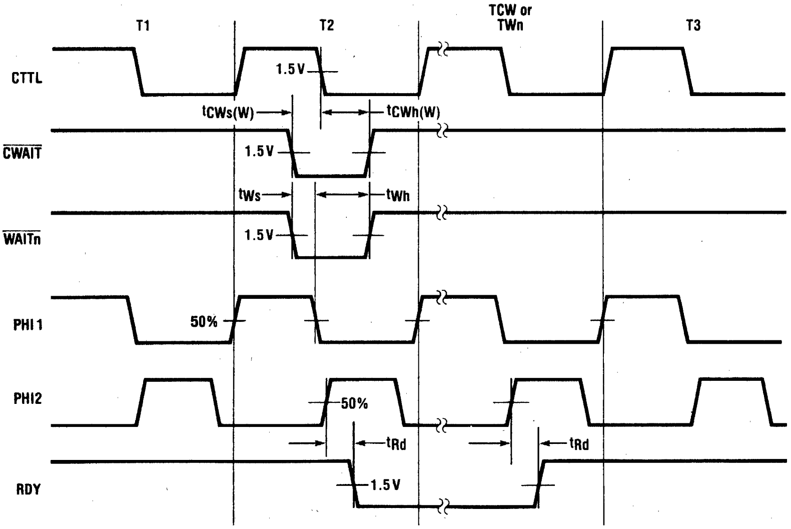


Control Outputs Timing Diagram (TRI-STATE Timing)

Timing Diagrams (Continued)

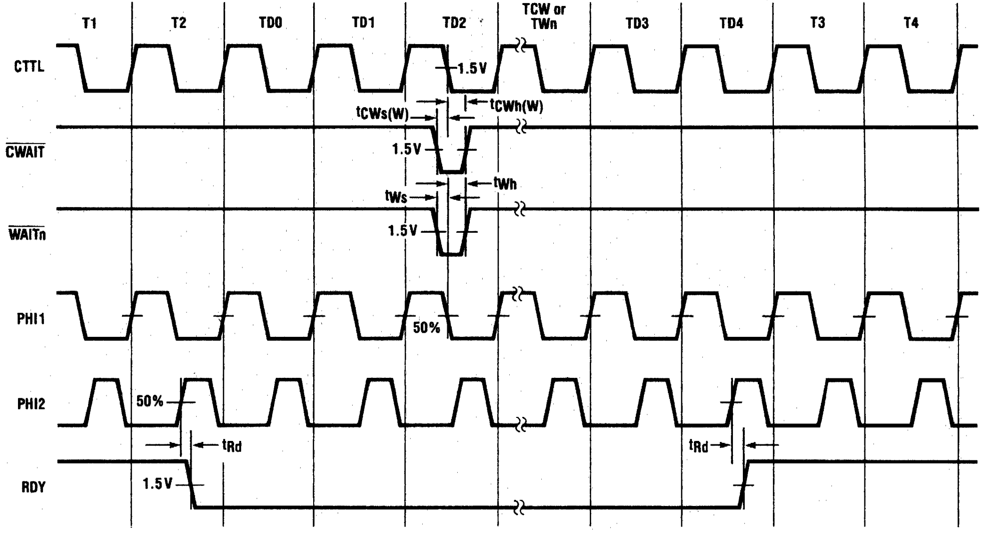


Cycle Hold Timing Diagram



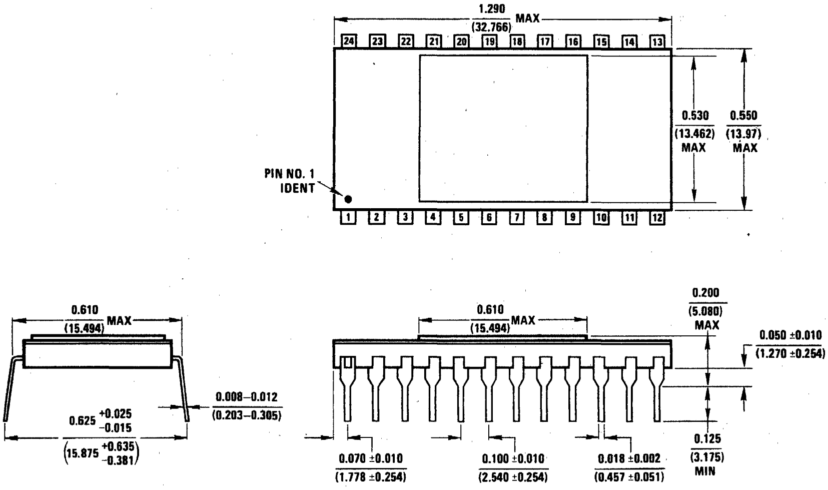
Wait State Timing Diagram (Fast Cycle)

Timing Diagrams (Continued)

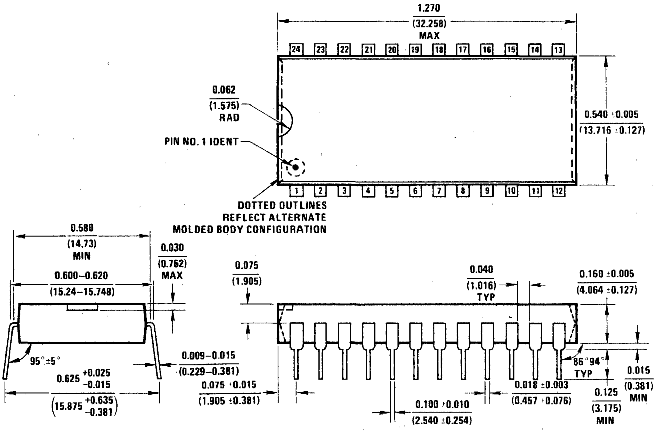


Wait State Timing Diagram (Slow Cycle)

Physical Dimensions inches (millimeters)



24-Lead Cavity Dual-In-Line Ceramic Package (D)
Order Number NS16201D-6
NS Package D24A



24-Lead Dual-In-Line Molded Package (N)
Order Number NS16201N-6
NS Package N24A

National Semiconductor

PRELIMINARY

NS16202 Interrupt Control Unit

General Description

The NS16202 Interrupt Control Unit (ICU) is the interrupt controller for the NS16000 microprocessor family. It is a support circuit that minimizes the software and real-time overhead required to handle multi-level, prioritized interrupts. A single NS16202 manages up to 16 interrupt sources, resolves interrupt priorities, and supplies a single-byte interrupt vector to the CPU.

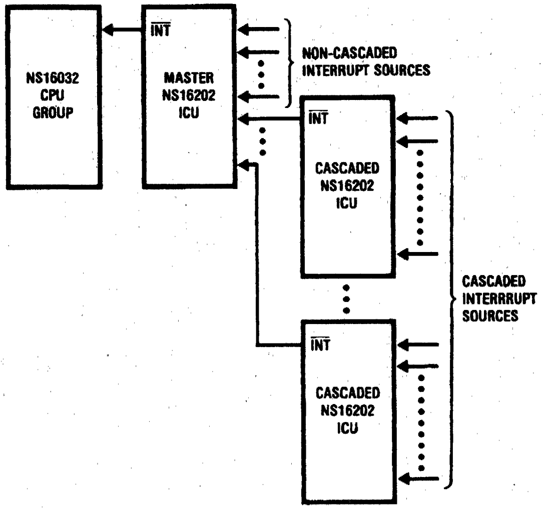
The NS16202 can operate in either of two data bus modes: 8-bit and 16-bit. In the 8-bit mode, up to 16 hardware interrupts with programmable priorities can be handled. In the 16-bit mode, 8 hardware and 8 software interrupts are possible. In either mode, up to 16 additional ICUs may be cascaded to handle a maximum of 256 interrupts.

Two 16-bit counters, which may be concatenated under program control into a single 32-bit counter, are also available for real time applications.

Features

- 16 maskable interrupt sources, cascadable to 256
- Programmable 8- or 16-bit data bus mode
- Edge or level triggering for each hardware interrupt with individually selectable polarities
- 8 software interrupts
- Fixed or rotating priority modes
- Two 16-bit, DC to 10MHz counters, that may be concatenated into a single 32-bit counter
- Optional 8-bit I/O port available in 8-bit data bus mode
- High-speed XMOS technology
- Single, +5V supply
- 40-pin, dual in-line package

NS16202 Basic System Configuration



TL/C/5117-1

Absolute Maximum Ratings

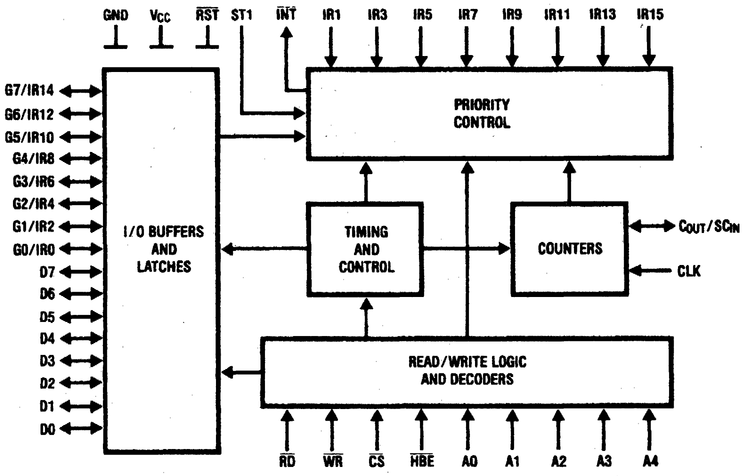
Temperature Under Bias	0°C to +70°C
Storage Temperature	−65°C to +150°C
All Input or Output Voltages with respect to GND	−0.5V to +7.0V
Power Dissipation	1.5 Watt

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 DC Electrical Characteristics.

DC Electrical Characteristics

T_a = 0° to 70°C, V_{CC} = +5V ± 5%, GND = 0V

Symbol	Parameter	Conditions	Min	Typ	Max	Units
V_{il}	Input Low Voltage				0.8	V
V_{ih}	Input High Voltage		2.0			V
V_{ol}	Output Low Voltage	I_{ol} = 2 mA			.45	V
V_{oh}	Output High Voltage	I_{oh} = −400 μA	2.4			V
I_{li}	Input Leakage Current	$V_{in} = V_{CC}$			±10	μA
I_{lo}	Output Leakage Current	$V_{out} = V_{CC}$			±10	μA
I_{il}	Input Load Current	V_{in} = 0 to V_{CC}			±10	μA
I_{CC}	Power Supply Current	I_{out} = 0, T = 0°C			300	mA

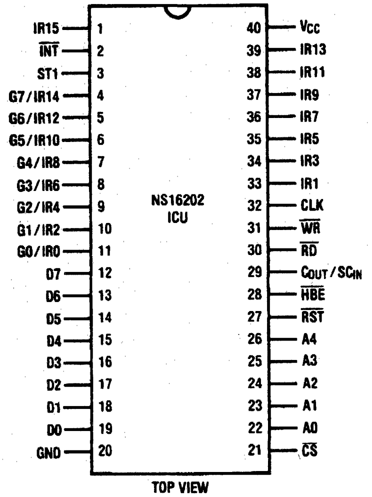


TL/C/5117-3

FIGURE 2-1. NS16202 ICU Block Diagram

Connection Diagram

Dual In-Line Package



TL/C/5117-2

1 NS16202 Pin Descriptions

1.1 Power Supply

Power (V_{CC}): +5V DC Supply

Ground (GND): Power Supply Return

1.2 Input Signals

Reset ($\overline{RST}$): Active low. This signal initializes the ICU. (The ICU initializes to the 8-bit bus mode.)

Chip Select ($\overline{CS}$): Active low. This signal enables the ICU to respond to address, data, and control signals from the CPU.

Addresses (A0 through A4): Address lines used to select the ICU internal registers for read/write operations.

High Byte Enable ($\overline{HBE}$): Active low. Enables data transfers on the most-significant byte of the Data Bus. If the ICU is in the 8-bit Bus Mode, this signal is not used and should be connected to either GND or V_{CC}.

Read ($\overline{RD}$): Active low. Enables data to be read from the ICU's internal registers.

Write ($\overline{WR}$): Active low. Enables data to be written into the ICU's internal registers.

Status (ST1): Status signal from the CPU. When the Hardware Vector Register is read, this signal differentiates an INTA cycle from an RETI cycle. If ST1=0 the ICU initiates an INTA cycle. If ST1=1 an RETI cycle will result.

Interrupt Requests (IR1, IR3, . . . ,IR15); This eight inputs are used for hardware interrupts. Each may be individually triggered in one of four modes: Rising Edge, Falling Edge, Low Level, or High Level.

Counter Clock (CKL): External clock signal to drive the ICU internal counters.

1.3 Output Signals

Interrupt Output ($\overline{INT}$): Active low. This signal indicates that an interrupt is pending.

1.4 Input/Output Signals

Data Bus 0–7 (D0 through D7): Eight low-order data bus lines used in both 8-bit and 16-bit bus modes.

General Purpose I/O Lines (G0/IR0, G1/IR2, . . . ,G7/IR14): These pins are the high-order data bits when the ICU is in the 16-bit bus mode. When the ICU is in the 8-bit bus mode, each pin may be individually assigned one of the following functions:

- Additional Hardware Interrupt Input (IR0 through IR14)
- General Purpose Data Input
- General Purpose Data Output
- Clock Output from H-Counter (Pins G0/IR0 through G3/IR6 only)

It should be noted that, for maximum flexibility in assigning interrupt priorities, the interrupt positions corresponding to pins G0/IR0, . . . ,G7/IR14 and IR1, . . . ,IR15 are interleaved.

Counter or Oscillator Output/Sampling Clock Input (COUT/SCIN): As an output, this pin provides either a clock signal generated by the ICU internal oscillator, or a zero detect signal from one or both of the ICU counters. As an input, it is used for an external clock, to override the internal oscillator used for interrupt sampling. This is done only for testing purposes.

2 Architectural Description

The NS16202 ICU functions as an overall manager in an interrupt-oriented system environment. Its many features and options permit the design of sophisticated interrupt systems.

Figure 2-1 shows the internal organization of the NS16202. As shown, the NS16202 is divided into five functional blocks. These are described in the following paragraphs:

2.1 I/O Buffers And Latches

The I/O Buffers and Latches is the interface with the system data bus. It contains bidirectional buffers for the data I/O pins. It also contains registers and logic circuits that control the operation of pins G0/IR0, . . . ,G7/IR14 when the ICU is in the 8-bit bus mode.

2.2 Read/Write Logic and Decoders

The Read/Write Logic and Decoders manage all internal and external data transfers for the ICU. These include Data, Control, and Status Word Transfers. This circuit accepts inputs from the CPU address and control buses. In turn, it issues commands to access the internal registers of the ICU.

2.3 Timing and Control

The Timing and Control Block contains status elements that select the ICU operating mode. It also contains state machines that generate all the necessary sequencing and control signals.

2.4 Priority Control

The Priority Control Block contains 16 units, one for each interrupt position. These units provide the following functions.

- Sensing the various forms of hardware interrupt signals e.g. level (high/low) or edge (rising/falling)
- Resolving priorities and generating an interrupt request to the CPU
- Handling cascaded arrangements
- Enabling software interrupts
- Providing for an automatic return from interrupt
- Enabling the assignment of any interrupt position to the internal counters
- Providing for rearrangement of priorities by assigning the first priority to any interrupt position
- Enabling automatic rotation of priorities

2.5 Counters

This block contains two 16-bit counters, called the H-counter and the L-counter. These are down counters that count from an initial value to zero. Both counters have a 16-bit register (designated HCSV and LCSV) for loading their restarting values. They also have registers containing the current count values (HCCV and LCCV). Both sets of registers are fully described in Chapter 5.

The counters are under program control and can be used to generate interrupts. When the count reaches zero, either counter can generate an interrupt request to any of the 16 interrupt positions. The counter then reloads the start value from the appropriate registers and resumes counting. Figure 2–2 shows typical counter output signals available from the NS16202.

The maximum input clock frequency is 2.5MHz.

A divide-by-four prescaler is also provided. When the prescaler is used, the input clock frequency can be up to 10MHz.

When intervals longer than those provided by a 16-bit counter are needed, the L- and H-counters can be concatenated to form a 32-bit counter. In this case, both counters are controlled by the H-counter control bits. Refer to the discussion of the Counter Control Register in Chapter 5 for additional information. Figure 2–3 summarizes counter read/write operations.

3 Functional Description

3.1 Reset

The ICU is reset when a logic low signal is present on the $\overline{\text{RST}}$ pin. An external reset signal is always required because the ICU does not have internal, power-up reset circuitry. At reset, most internal ICU registers are affected, and the ICU becomes inactive.

3.2 Initialization

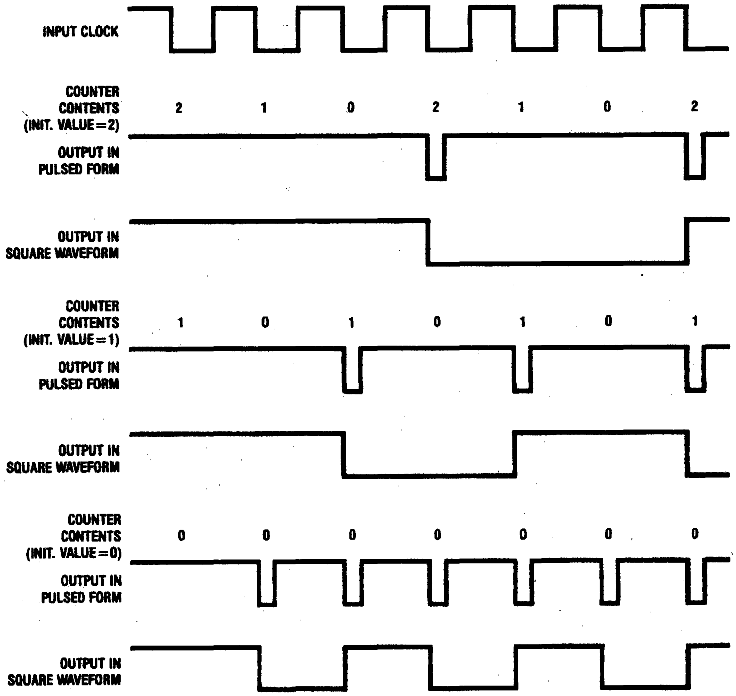
After reset, the CPU must initialize the NS16202 to establish its configuration. Proper initialization requires knowledge of the ICU register's formats. Therefore, a flowchart of a recommended initialization sequence is shown in Chapter 5 (Figure 5–2) after the discussion of the ICU registers.

The operation sequence shown in Figure 5–2 ensures that all counter output pins remain inactive until the counters are completely initialized.

3.3 Vectored Interrupt Handling

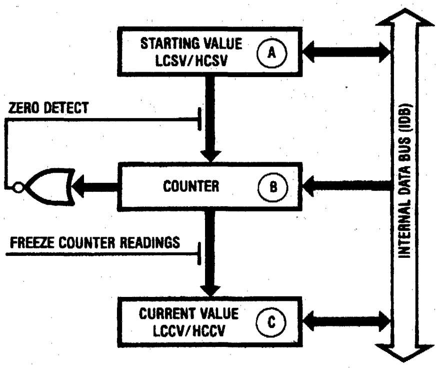
For details on the operation of the vectored interrupt mode for a particular NS16000 CPU, refer to the data sheet for that CPU. In this discussion, it is assumed that the NS16202 is working with a CPU in the vectored interrupt mode. Several ICU applications are discussed, including non-cascaded and cascaded operation. Figures 3–1, 3–2, and 3–3 show typical configurations of the ICU used with the NS16032 CPU.

A peripheral device issues an interrupt request by sending the proper signal to one of the NS16202 interrupt inputs. If the interrupt input is not masked, the ICU activates its Interrupt Output ($\overline{\text{INT}}$) pin and generates an interrupt vector byte. The interrupt vector byte identifies the interrupt source in its four least significant bits. When the CPU detects a low level on its Interrupt Input pin, it performs one or two interrupt acknowledge cycles depending on whether the interrupt request is from the master ICU or a cascaded ICU. Figure 3–4 shows a flowchart of a typical CPU Interrupt Acknowledge sequence.


INPUT CLOCK
COUNTER CONTENTS (INIT. VALUE=2)
2 1 0 2 1 0 2
OUTPUT IN PULSED FORM
OUTPUT IN SQUARE WAVEFORM
COUNTER CONTENTS (INIT. VALUE=1)
1 0 1 0 1 0 1
OUTPUT IN PULSED FORM
OUTPUT IN SQUARE WAVEFORM
COUNTER CONTENTS (INIT. VALUE=0)
0 0 0 0 0 0 0
OUTPUT IN PULSED FORM
OUTPUT IN SQUARE WAVEFORM

TL/C/5117-4

FIGURE 2–2. Counter Output Signals in Pulsed Form and Square Waveform for Three Different Initial Values.

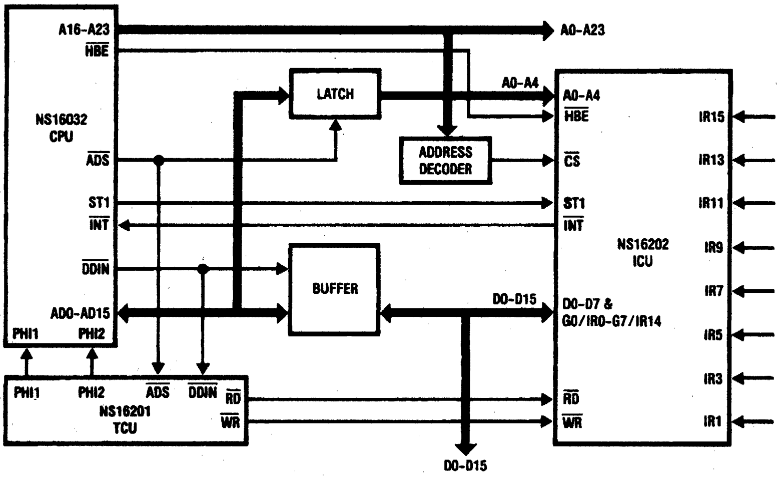


TL/C/5117-5

BASIC OPERATIONS:

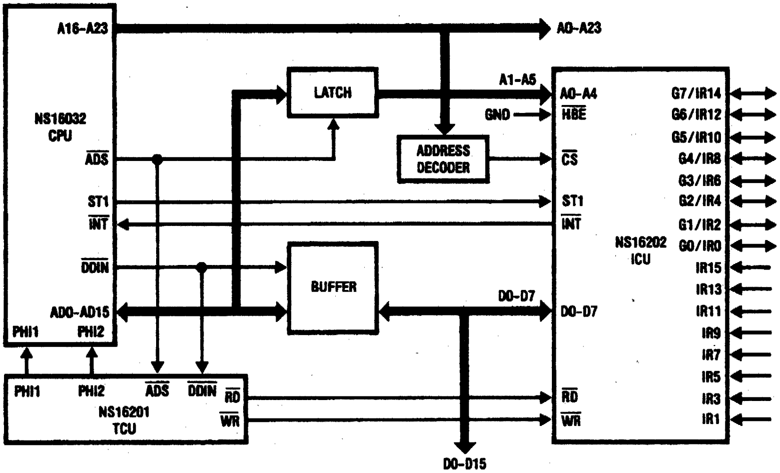
WRITING TO LCSV/HCSV	Ⓐ←(IDB)
READING LCSV/HCSV	Ⓐ→(IDB)
WRITING TO LCCV/HCCV	Ⓑ←(IDB)
(only possible when counters are halted)	Ⓒ←(IDB)
READING LCCV/HCCV	Ⓒ→(IDB)
(recommended only while counter readings are frozen)	
COUNTER COUNTS AND READINGS ARE NOT FROZEN	Ⓒ←Ⓑ
COUNTER RELOADS STARTING VALUE	Ⓑ←Ⓐ
(occurs on the clock cycle following the one where it reaches zero)	

FIGURE 2–3. Counter Configuration and Basic Operations.



TL/C/5117-6

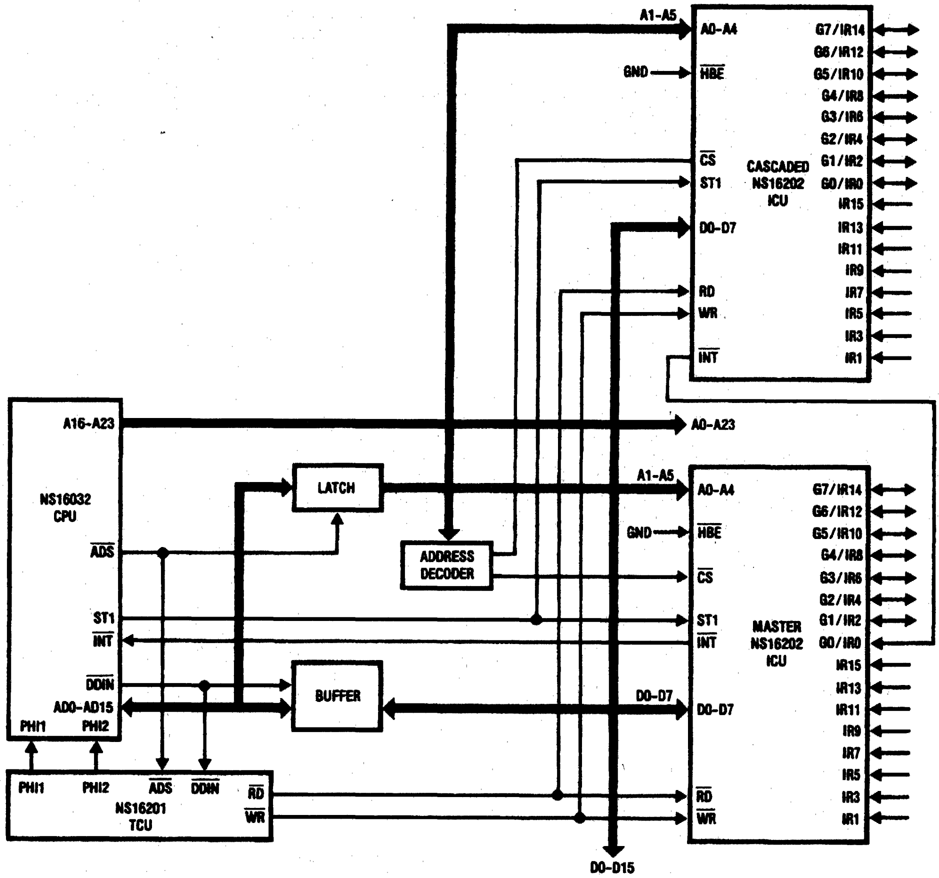
FIGURE 3-1. Interrupt Control Unit Connections in 16-Bit Bus Mode.



TL/C/5117-7

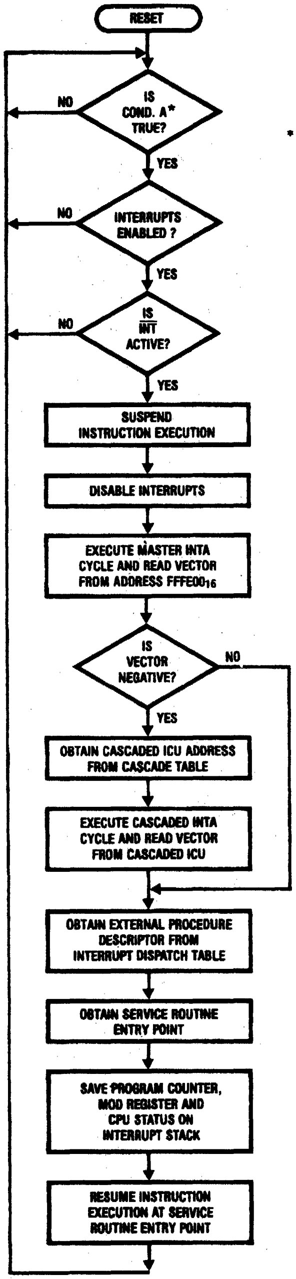
NOTE: In the 8-Bit Bus Mode the Master ICU Registers appear at even addresses (A0 = 0) since the ICU communicates with the least significant byte of the CPU data bus.

FIGURE 3-2. Interrupt Control Unit Connections in 8-Bit Bus Mode.



TL/C/5117-8

FIGURE 3-3. Cascaded Interrupt Control Unit Connections in 8-Bit Bus Mode.


RESET
IS COND. A* TRUE?
NO
YES
INTERRUPTS ENABLED ?
NO
YES
IS INT ACTIVE?
NO
YES
SUSPEND INSTRUCTION EXECUTION
DISABLE INTERRUPTS
EXECUTE MASTER INTA CYCLE AND READ VECTOR FROM ADDRESS FFFE00$_{16}$
IS VECTOR NEGATIVE?
NO
YES
OBTAIN CASCADED ICU ADDRESS FROM CASCADE TABLE
EXECUTE CASCADED INTA CYCLE AND READ VECTOR FROM CASCADED ICU
OBTAIN EXTERNAL PROCEDURE DESCRIPTOR FROM INTERRUPT DISPATCH TABLE
OBTAIN SERVICE ROUTINE ENTRY POINT
SAVE PROGRAM COUNTER, MOD REGISTER AND CPU STATUS ON INTERRUPT STACK
RESUME INSTRUCTION EXECUTION AT SERVICE ROUTINE ENTRY POINT

* Cond. A is true if current instruction is terminated or an interruptible point in a string instruction is reached.

TL/C/5117-9

FIGURE 3-4. CPU Interrupt Acknowledge Sequence.

In general, vectored interrupts are serviced by interrupt routines stored in system memory. The Dispatch Table stores up to 256 external procedure descriptors for the various service procedures. The CPU INTBASE register points to the top of the Dispatch Table. Figure 3-5 shows the layout of the Dispatch Table. This figure also shows the layout of the Cascade Table, which is discussed with ICU cascaded operation.

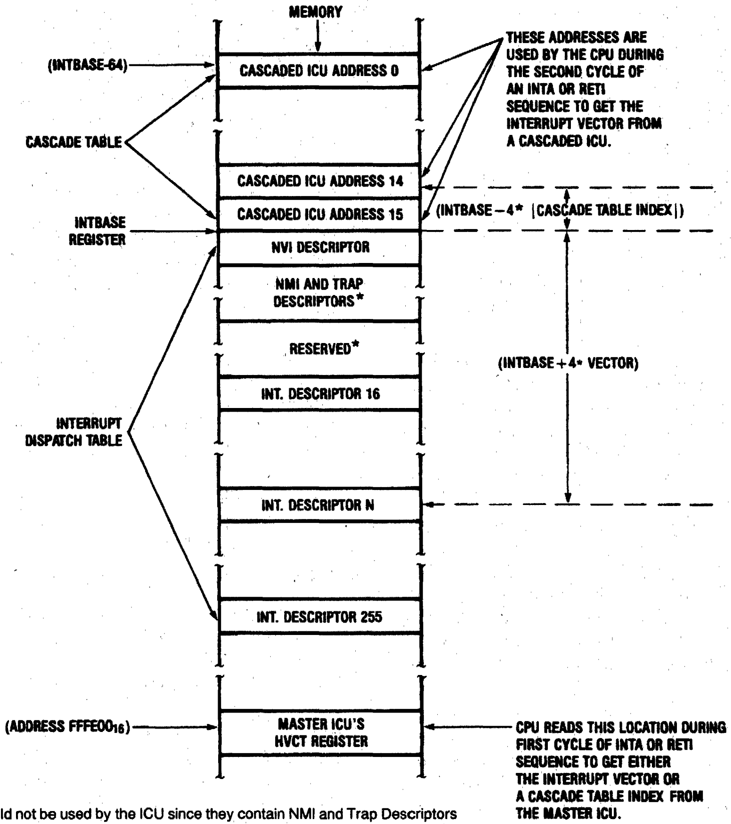
Non-Cascaded Operation. Whenever an interrupt request from a peripheral device is issued directly to the master ICU, a non-cascaded interrupt request to the CPU results. In a system using a single NS16202, up to 16 interrupt requests can be prioritized. Upon receipt of an interrupt request on the $\overline{\text{INT}}$ pin, the CPU performs a Master Interrupt-Acknowledge bus cycle, reading a vector byte from address FFFE00_{16}. 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 a Return-from-Interrupt bus cycle, informing the ICU that it may re-prioritize any interrupt requests still pending. Figure 3-6 shows a typical CPU RETI sequence. In a system with only one ICU, 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 value, the master ICU flags the interrupt source as a cascaded ICU (see below).

Cascaded Operation. In cascaded operation, one or more of the interrupt inputs of the master ICU are connected to the Interrupt Output pin of one or more cascaded ICUs. Up to 16 cascaded ICUs may be used, giving a system total of 256 interrupts.

NOTE: The number of cascaded ICUs is practically limited to 15 because the Dispatch Table for the NS16032 CPU is constructed with entries 1 through 15 either used for NMI and Trap descriptors, or reserved for future use. Interrupt position 0 of the master ICU should not be cascaded, so it can be vectored through Dispatch Table entry 0, reserved for non-vectored interrupts. In this case, the non-vectored interrupt entry (entry 0) is also available for vectored interrupt operation, since the CPU is operating in the vectored interrupt mode.

The address of the master ICU should be FFFE00_{16}. (*) Cascaded ICUs can be located at any system address. A list of cascaded ICU addresses is maintained in the Cascade Table as a series of sixteen 32-bit entries.

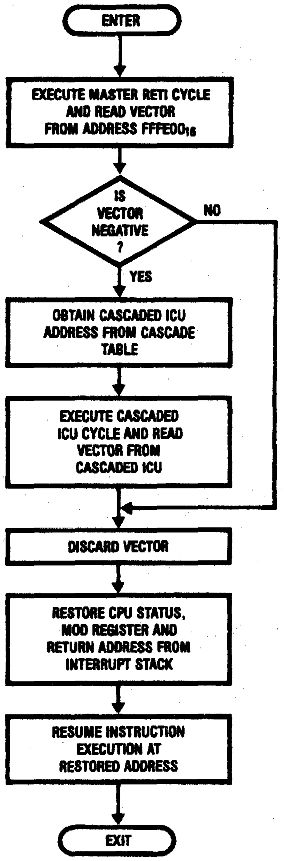
(*)NOTE: The CPU status corresponding to both, master interrupt acknowledge and return from interrupt bus cycles, as well as address bit A8, could be used to generate the chip select ($\overline{\text{CS}}$) signal for accessing the master ICU during one of the above cycles. In this case the master ICU can reside at any system address. The only limitation is that the least significant 5 or 6 address bits (6 in the 8-bit bus mode) must be zero. The address bit A8 must be decoded to prevent an NMI bus cycle from reading the hardware vector register of the ICU. This could happen, since the NS16032 CPU performs a dummy read cycle from address FFFF00_{16}, with the same status as a master INTA cycle, when a non-maskable-interrupt is acknowledged.


MEMORY
(INTBASE-64)
CASCADED ICU ADDRESS 0
CASCADE TABLE
CASCADED ICU ADDRESS 14
CASCADED ICU ADDRESS 15
INTBASE REGISTER
NVI DESCRIPTOR
NMI AND TRAP DESCRIPTORS*
RESERVED*
INT. DESCRIPTOR 16
INTERRUPT DISPATCH TABLE
INT. DESCRIPTOR N
INT. DESCRIPTOR 255
(ADDRESS FFFE00₁₆)
MASTER ICU'S HVCT REGISTER
THESE ADDRESSES ARE USED BY THE CPU DURING THE SECOND CYCLE OF AN INTA OR RETI SEQUENCE TO GET THE INTERRUPT VECTOR FROM A CASCADED ICU.
(INTBASE−4* |CASCADE TABLE INDEX|)
(INTBASE+4* VECTOR)
CPU READS THIS LOCATION DURING FIRST CYCLE OF INTA OR RETI SEQUENCE TO GET EITHER THE INTERRUPT VECTOR OR A CASCADE TABLE INDEX FROM THE MASTER ICU.

* Table entries 1 to 15 should not be used by the ICU since they contain NMI and Trap Descriptors or are reserved for future use. (For more details refer to NS16032 data sheet.)

TL/C/5117-10

FIGURE 3-5. Interrupt Dipatch and Cascade Tables.


ENTER
EXECUTE MASTER RETI CYCLE AND READ VECTOR FROM ADDRESS FFFE00₁₆
IS VECTOR NEGATIVE ?
NO
YES
OBTAIN CASCADED ICU ADDRESS FROM CASCADE TABLE
EXECUTE CASCADED ICU CYCLE AND READ VECTOR FROM CASCADED ICU
DISCARD VECTOR
RESTORE CPU STATUS, MOD REGISTER AND RETURN ADDRESS FROM INTERRUPT STACK
RESUME INSTRUCTION EXECUTION AT RESTORED ADDRESS
EXIT

TL/C/5117-11

FIGURE 3-6. CPU Return from Interrupt Sequence.

The master ICU maintains a list (in the CSRC register pair) of its interrupt positions that are cascaded. When a cascaded interrupt input is active, the master ICU activates its interrupt output and the CPU responds with a Master Interrupt Acknowledge Cycle. However, instead of generating a positive interrupt vector, the master ICU generates a negative Cascade Table index.

The CPU interprets the negative number returned from the master ICU as an index into the Cascade Table. The Cascade Table is located in a negative direction from the Dispatch Table, and it contains the virtual addresses of the hardware vector registers for any cascaded NS16202s in the system. Thus, the Cascade Table index supplied by the master ICU identifies the cascaded ICU that requested the interrupt.

Once the cascaded ICU is identified, the CPU performs a Cascaded Interrupt Acknowledge cycle. During this cycle, the CPU reads the final vector value directly from the cascaded ICU, and uses it to access the Dispatch Table. Each cascaded ICU, of course, has its own set of 16 unique interrupt vectors, one vector for each of its 16 interrupt positions.

The CPU interprets the vector value read during a Cascaded Interrupt Acknowledge cycle as an unsigned number. Thus, this vector can be in the range 0 through 255.

When a cascaded interrupt service routine completes its task, it must return control to the main program with the same RETI instruction used in non-cascaded interrupt service routines. However, when the CPU performs a Master Return From Interrupt cycle, the CPU accesses the master ICU and reads the negative Cascade Table index identifying the cascaded ICU that originally received the interrupt request. Using the cascaded ICU address, the CPU now performs a Cascaded Return From Interrupt cycle, informing the cascaded ICU that the service routine is over. The byte provided by the cascaded ICU during this cycle is ignored.

3.4 Internal ICU Operating Sequence

The NS16202 ICU accepts two interrupt types, software and hardware.

Software interrupts are initiated when the CPU sets the proper bit in the Interrupt Pending (IPND) registers (R6, R7), located in the ICU. Bits are set and reset by writing the proper byte to either R6 or R7. Software interrupts can be masked, by setting the proper bit in the mask registers (R10, R11).

Hardware interrupts can be either internal or external to the ICU. Internal ICU hardware interrupts are initiated by the on-chip counter outputs. External hardware interrupts are initiated by devices extenal to the ICU, that are connected to any of the ICU interrupt input pins.

Hardware interrupts can be masked by setting the proper bit in the mask registers (R10, R11). If the Freeze bit (FRZ), located in the Mode Control Register (MCTL), is set, all incoming hardware interrupts are inhibited from setting their corresponding bits in the IPND registers. This prevents the ICU from recognizing any hardware interrupts.

Once the ICU is initialized, it is enabled to accept interrupts. If an active interrupt is not masked, and has a higher priority than any interrupt currently being serviced, the ICU activates its Interrupt Output ($\overline{\text{INT}}$). Figure 3-7 is a flowchart showing the ICU interrupt acknowledge sequence.

The CPU responds to the active $\overline{\text{INT}}$ line by performing an Interrupt Acknowledge bus cycle. During this cycle, the ICU clears the IPND bit corresponding to the active interrupt position and sets the corresponding bit in the Interrupt In-Service Registers (ISRV). The ISRV bit remains set until the CPU performs a RETI bus cycle triggered by the completion of the interrupt service routine for the active interrupt position. Figure 3-8 is a flowchart showing ICU operation during a RETI bus cycle.

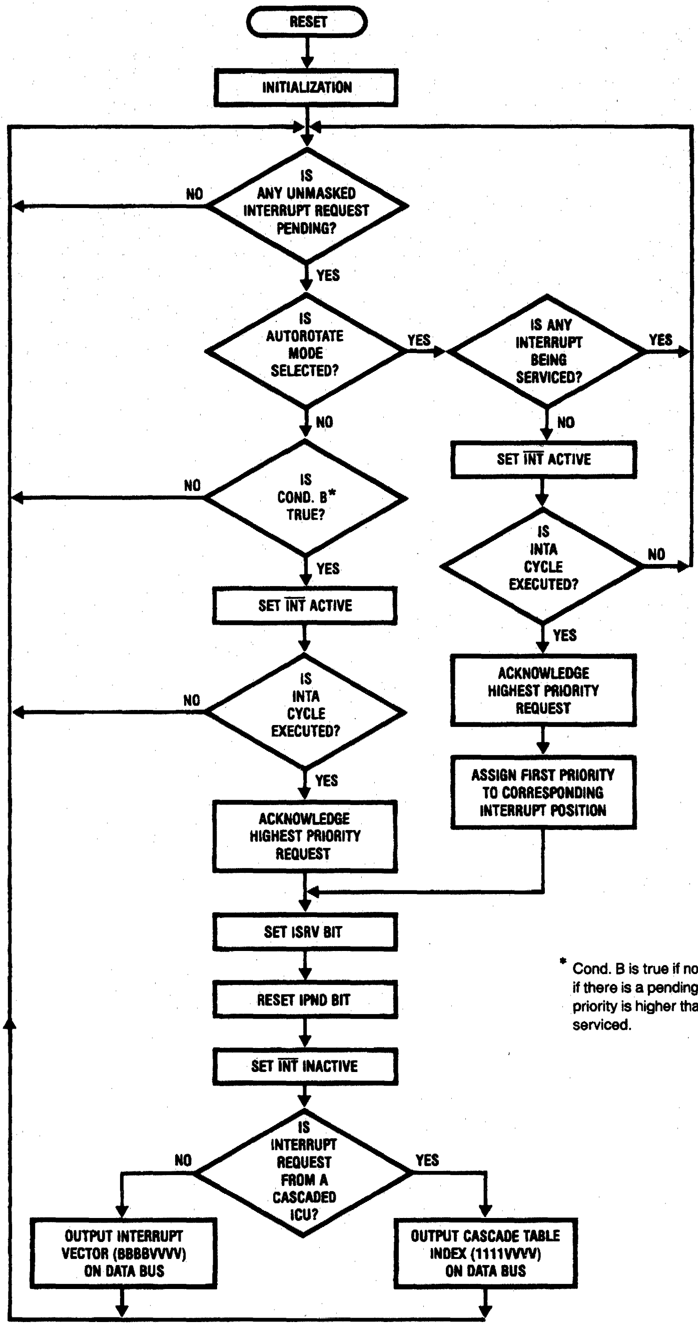
When the ISRV bit is set, the $\overline{\text{INT}}$ output is disabled. This output remains inactive until a higher priority interrupt position becomes active, or the ISRV bit is cleared.

4 Interrupt Priority Modes

The NS16202 ICU can operate in one of four interrupt priority modes: Fixed Priority; Auto-Rotate; Special Mask; and Polling. Each mode is described below.

4.1 Fixed Priority Mode

In the Fixed Priority Mode (also called Fully Nested Mode), each interrupt position is ranked in priority from 0 to 15, with 0 being the highest priority. In this mode, the processing of lower priority interrupts is nested with higher priority interrupts. That is, while an interrupt is being serviced, any other interrupts of the same or lower priority are inhibited. The ICU does, however, recognize higher priority interrupt requests.


RESET
INITIALIZATION
IS ANY UNMASKED INTERRUPT REQUEST PENDING?
NO
YES
IS AUTOROTATE MODE SELECTED?
YES
IS ANY INTERRUPT BEING SERVICED?
YES
NO
NO
SET $\overline{\text{INT}}$ ACTIVE
IS COND. B* TRUE?
NO
YES
IS INTA CYCLE EXECUTED?
NO
YES
SET $\overline{\text{INT}}$ ACTIVE
ACKNOWLEDGE HIGHEST PRIORITY REQUEST
IS INTA CYCLE EXECUTED?
NO
YES
ASSIGN FIRST PRIORITY TO CORRESPONDING INTERRUPT POSITION
ACKNOWLEDGE HIGHEST PRIORITY REQUEST
SET ISRV BIT
RESET IPND BIT
SET $\overline{\text{INT}}$ INACTIVE
IS INTERRUPT REQUEST FROM A CASCADED ICU?
NO
YES
OUTPUT INTERRUPT VECTOR (BBBBVVVV) ON DATA BUS
OUTPUT CASCADE TABLE INDEX (1111VVVV) ON DATA BUS

* Cond. B is true if no interrupt is being serviced, or if there is a pending unmasked interrupt whose priority is higher than that of the interrupt being serviced.

TL/C/5117-12

FIGURE 3-7. ICU Interrupt Acknowledge Sequence.

When the interrupt service routine executes its RETI instruction, the corresponding ISRV bit is cleared. This allows any lower priority interrupt request to be serviced by the CPU.

At reset, the default priority assignment gives interrupt IR0 priority 0 (highest priority), interrupt IR1 priority 1, and so forth. Interrupt IR15 is, of course, assigned priority 15, the lowest priority. The default priority assignment can be altered by writing an appropriate value into register FPRT (L) as explained in Section 5.9.

NOTE: When the ICU generates an interrupt request to the CPU for a higher priority interrupt while a lower priority interrupt is still being serviced by the CPU, the CPU responds to the interrupt request only if its internal interrupt enable flag is set. Normally, this flag is reset at the beginning of an interrupt acknowledge cycle and set during the RETI cycle. If the CPU is to respond to higher priority interrupts during any interrupt service routine, the service routine must set the internal CPU interrupt enable flag, as soon during the service routine as desired.

4.2 Auto-Rotate Mode

The Auto Rotate mode is selected when the NTAR bit is set to 0, and is automatically entered after Reset. In this mode an interrupt source position is automatically assigned lowest priority after a request at that position has been serviced. Highest priority then passes to the next lower priority position. For example, when servicing of the interrupt request at position 3 is completed (ISRV bit 3 is cleared), interrupt position 3 is assigned lowest priority and position 4 assumes highest priority. The nesting of interrupts is inhibited, since the interrupt being serviced always has the highest priority. This mode is used when the interrupting devices have to be assigned equal priority. A device requesting an interrupt, will have to wait, in the worst case, until each of the 15 other devices has been serviced at most once.

4.3 Special Mask Mode

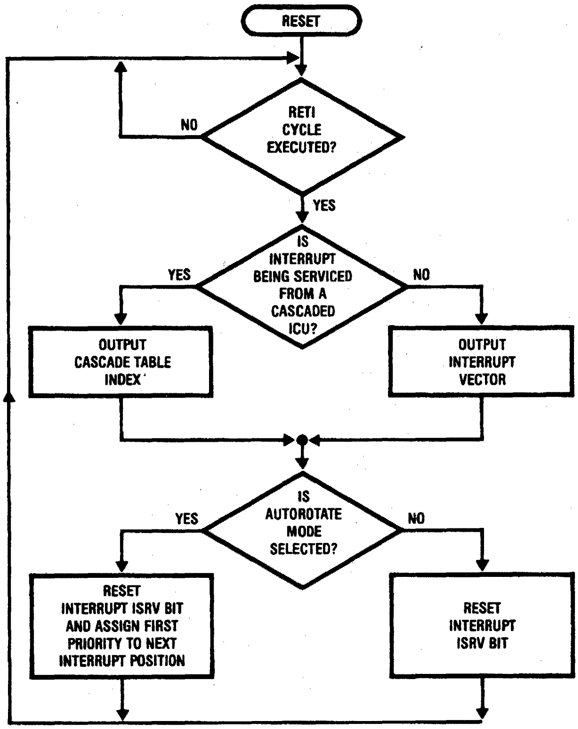
The special mask mode is used when it is necessary to dynamically alter the ICU priority structure while an interrupt is being serviced. For example, it may be desired in a particular interrupt service routine to enable lower priority interrupts during a part of the routine. To do so, the ICU must be programmed in fixed priority mode and the interrupt service routine must control its own in-service bit in the ISRV registers.

The bits of the ISRV registers are changed with either the Set Bit Interlocked or Clear Bit Interlocked instructions (SBITIW or CBITIW). The in-service bit is cleared to enable lower priority interrupts and set to disable them.

NOTE: For proper operation of the ICU, an interrupt service routine must set its ISRV bit before executing the RETI instruction. This prevents the RETI cycle from clearing the wrong ISRV bit.

4.4 Polling Mode

The polling mode gives complete control of interrupt priority to the system software. Either some or all of the interrupt positions can be assigned to the polling mode. To assign all interrupt positions to the polling mode, the CPU interrupt enable flag is reset. To assign only some of the interrupt positions to the polling mode, the desired interrupt positions are masked in the Interrupt Mask registers (IMSK). In either case, the polling operation consists of reading the Interrupt Pending (IPND) registers.


RESET
RETI CYCLE EXECUTED?
NO
YES
IS INTERRUPT BEING SERVICED FROM A CASCADED ICU?
YES
NO
OUTPUT CASCADE TABLE INDEX
OUTPUT INTERRUPT VECTOR
IS AUTOROTATE MODE SELECTED?
YES
NO
RESET INTERRUPT ISRV BIT AND ASSIGN FIRST PRIORITY TO NEXT INTERRUPT POSITION
RESET INTERRUPT ISRV BIT

TL/C/5117-13

FIGURE 3-8. ICU Return from Interrupt Sequence.

If necessary, the IPND read can be synchronized by setting the Freeze (FRZ) bit in the Mode Control register (MCTL). This prevents any change in the IPND registers during the read. The FRZ bit must be reset after the polling operation so the IPND contents can be updated. If an edge-triggered interrupt occurs while the IPND registers are frozen, the interrupt request is latched, and tranferred to the IPND registers as soon as FRZ is reset.

The polling mode is useful when a single routine is used to service several interrupt levels.

5 Register Functions

The NS16202 has thirty-two 8-bit registers that can be accessed either individually or in pairs. In 16-bit data bus mode, register pairs can be accessed with the CPU word or double-word reference instructions. Figure 5-1 shows the ICU internal registers. This figure summarizes the name, function, and offset address for each register.

Because some registers hold similar data, they are grouped into functional pairs and assigned a single name. However, if a single register in a pair is referenced, either an L or an H is appended to the register name. The letters are placed in parentheses and stand for the low order 8 bits (L) and the high order 8 bits (H). For example, register R6, part of the Interrupt Pending (IPND) register pair, is referred to individually as IPND(L).

The following paragraphs give detailed descriptions of the registers shown in Figure 5-1.

5.1 HVCT — Hardware Vector Register (R0)

The HVCT register is a single register that contains the interrupt vector byte supplied to the CPU during an Interrupt Acknowledge (INTA) or Return From Interrupt (RETI) cycle. The HVCT bit map is shown below;

7	6	5	4	3	2	1	0
B	B	B	B	V	V	V	V

The BBBB field is the bias which is programmed by writing $BBBB0000_2$ to the SVCT register (R1). The VVVV field identifies one of the 16 interrupt positions. The contents of the HVCT register provide various information to the CPU, as shown below:

NOTE: The ICU always interprets a read of the HVCT register as either an INTA or RETI cycle. Since these cycles cause internal changes to the ICU, normal programs must never read the ICU HVCT register.

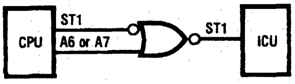
5.2 SVCT. Software Vector Register (R1)

The SVCT register is a copy of the HVCT register. It allows the programmer to read the contents of the HVCT register without initiating a INTA or RETI cycle in the ICU. It also allows a programmer to change the BBBB field of the HVCT register. The bit map of the SVCT register is the same as for the HVCT register.

During a write to SVCT, the four least significant bits are unaffected while the four most significant bits are written into both SVCT and HVCT (R1 and R0).

The SVCT register is updated dynamically by the ICU. The four least significant bits always contain the vector value that would be returned to the CPU if a INTA or RETI cycle were executed. Therefore, when reading the SVCT register, the state of the CPU ST1 pin is used to select either pending interrupt data or in-service interrupt data. For example, if the SVCT register is read with ST1 = 0 (as for an INTA cycle), the VVVV field contains the encoded value of the highest priority pending interrupt. On the other hand, if the SVCT register is read with ST1 = 1, the VVVV field contains the encoded value of the highest priority in-service interrupt.

NOTE: If the CPU ST1 output is connected directly to the ICU ST1 input, the vector read from SVCT is always the RETI vector. If both the INTA and RETI vectors are desired, additional logic must be added to drive the ICU ST1 input. A typical circuit is shown below. In this circuit, the state of the ICU ST1 input is controlled by both the CPU ST1 output and the selected address bit.



TL/C/5117-14

HVCT Register Data Coding

	INTA CYCLE (ST1=0)		RETI CYCLE (ST1=1)	
	Highest priority pending interrupt is from:		Highest priority in-service interrupt was from:	
BBBB	cascaded ICU	any other source	cascaded ICU	any other source
	1111	programmed bias*	1111	programmed bias*
VVVV	encoded value of the highest priority pending interrupt		encoded value of the highest priority in-service interrupt	

* The Programmed bias for the master ICU must range from 0000 to 0111_2 because the CPU interprets a one in the most significant bit position as a Cascade Table Index indicator for a cascaded ICU.

REG. NUMBER AND ADDRESS IN HEX.		REG. NAME	REG. FUNCTION
	R0 (00_{16})	HVCT —	HARDWARE VECTOR
	R1 (01_{16})	SVCT —	SOFTWARE VECTOR
R3 (03_{16})	R2 (02_{16})	ELTG —	EDGE/LEVEL TRIGGERING
R5 (05_{16})	R4 (04_{16})	TPL —	TRIGGERING POLARITY
R7 (07_{16})	R6 (06_{16})	IPND —	INTERRUPTS PENDING
R9 (09_{16})	R8 (08_{16})	ISRV —	INTERRUPTS IN-SERVICE
R11 ($0B_{16}$)	R10 ($0A_{16}$)	IMSK —	INTERRUPT MASK
R13 ($0D_{16}$)	R12 ($0C_{16}$)	CSRC —	CASCADED SOURCE
R15 ($0F_{16}$)	R14 ($0E_{16}$)	FPRT —	FIRST PRIORITY
	R16 (10_{16})	MCTL —	MODE CONTROL
	R17 (11_{16})	OCASN —	OUTPUT CLOCK ASSIGNMENT
	R18 (12_{16})	CIPTR —	COUNTER INTERRUPT POINTER
	R19 (13_{16})	PDAT —	PORT DATA
	R20 (14_{16})	IPS —	INTERRUPT/PORT SELECT
	R21 (15_{16})	PDIR —	PORT DIRECTION
	R22 (16_{16})	CCTL —	COUNTER CONTROL
	R23 (17_{16})	CICTL —	COUNTER INTERRUPT CONTROL
R25 (19_{16})	R24 (18_{16})	LCSV —	L-COUNTER STARTING VALUE
R27 ($1B_{16}$)	R26 ($1A_{16}$)	HCSV —	H-COUNTER STARTING VALUE
R29 ($1D_{16}$)	R28 ($1C_{16}$)	LCCV —	L-COUNTER CURRENT VALUE
R31 ($1F_{16}$)	R30 ($1E_{16}$)	HCCV —	H-COUNTER CURRENT VALUE

FIGURE 5–1. ICU Internal Registers.

5.3 ELTG — Edge/Level Triggering Registers (R2, R3)

The ELTG registers determine the input trigger mode for each of the 16 interrupt inputs. Each input is assigned a bit in this register pair. An interrupt input is level-triggered if its bit in ELTG is set to 1. The input is edge-triggered if its bit is cleared. At reset, all bits in ELTG are set to 1.

Software interrupt positions are not affected by the state of their ELTG bits.

5.4 TPL — Triggering Polarity Registers (R4, R5)

The TPL registers determine the polarity of either the active level or the active edge for each of the 16 interrupt inputs. As with the ELTG registers, each input is assigned a bit. Possible triggering modes for the various combinations of ELTG and TPL bits are shown below.

ELTG BIT	TPL BIT	TRIGGERING MODE
0	0	Falling Edge
0	1	Rising Edge
1	0	Low Level
1	1	High Level

Software interrupt positions are not affected by their TPL bits. At reset, all TPL bits are set to 0.

NOTE: Hardware interrupt inputs connected to cascaded ICUs must have their TPL bits set to 0.

5.5 IPND — Interrupt Pending Registers (R6, R7)

The IPND registers track interrupt requests that are pending but not yet serviced. Each interrupt position is assigned a bit in IPND. When an interrupt is pending, the corresponding bit in IPND is set. The IPND data are used by the ICU to generate interrupts to the CPU. These data are also used in polling operations.

The IPND registers are also used for requesting software interrupts. This is done by writing specially formatted data bytes to either IPND(L) or IPND(H). The formats differ for registers R6 and R7. These formats are shown below:

IPND(L) (R6) — S0000PPP

IPND(H) (R7) — S0001PPP

Where: S = Set (S = 1) or Clear (S = 0)
PPP = is a binary number identifying one of eight bits

NOTE: The data read from either R6 or R7 are different from that written to the register because the ICU returns the register contents, rather than the formatted byte used to set the register bits.

The ICU automatically clears a set IPND bit when the pending interrupt request is serviced. All pending interrupts in a register can be cleared by writing the pattern 'X1XXXXXX' to it (X = don't care). To avoid conflicts with asynchronous hardware interrupt requests, the IPND registers should be frozen before pending interrupts are cleared. Refer to the Mode Control Register description for details on freezing the IPND registers.

At reset, all IPND bits are set to 0.

NOTE: The edge sensing mechanism used for hardware interrupts in the NS16202 ICU is a latching device that can be cleared only by acknowledging the interrupt or by changing the trigger mode to level sensing. Therefore, before clearing pending interrupts in the IPND registers, any edge-triggered interrupt inputs must first be switched to the level-triggered mode. This clears the edge-triggered interrupts; the remaining interrupts can then be cleared in the manner described above. This applies to clearing the interrupts only. Edge-triggered interrupts can be set without changing the trigger mode.

5.6 ISRV — Interrupt In-Service Registers (R8, R9)

The ISRV registers track interrupt requests that are currently being serviced. Each interrupt position is assigned a bit in ISRV. When an interrupt request is serviced by the ICU, its corresponding bit is set in the ISRV registers. Before generating an interrupt to the CPU, the ICU checks the ISRV registers to ensure that no higher priority interrupt is currently being serviced.

Each time the CPU executes an RETI instruction, the ICU clears the ISRV bit corresponding to the highest priority interrupt in service. The ISRV registers can also be written into by the CPU. This is done to implement the special mask priority mode.

At reset, the ISRV registers are set to 0.

5.7 IMSK — Interrupt Mask Registers (R10, R11)

Each NS16202 interrupt position can be individually masked. A masked interrupt source is not acknowledged by the ICU. The IMSK registers store a mask bit for each of the ICU interrupt positions. If an interrupt position's IMSK bit is set to 1, the position is masked.

The IMSK registers are controlled by the system software. At reset, all IMSK bits are set to 1, disabling all interrupts.

5.8 CSRC — Cascaded Source Registers (R12, R13)

The CSRC registers track any cascaded interrupt positions. Each interrupt position is assigned a bit in the CSRC registers. If an interrupt position's CSRC bit is set, that position is connected to the $\overline{\text{INT}}$ output of another NS16202 ICU, i.e., it is a cascaded interrupt.

At reset, the CSRC registers are set to 0.

NOTE: Only the Master ICU should have any CSRC bits set. If CSRC bits are set in a cascaded ICU, incorrect operation results.

5.9 FPRT — First Priority Registers (R14, R15)

The FPRT registers track the ICU interrupt position that currently holds first priority. Only one bit of the FPRT registers is set at one time. The set bit indicates the interrupt position with first (highest) priority.

The FPRT registers are automatically updated when the ICU is in the auto-rotate mode. The first priority interrupt can be determined by reading the FPRT registers. This operation returns a 16-bit word with only one bit set. An interrupt position can be assigned first priority by writing a formatted data byte to the FPRT(L) register. The format is shown below:

7	6	5	4	3	2	1	0
X	X	X	X	F	F	F	F

Where: XXXX = Don't Care
FFFF = A binary number from 0 to 15 indicating the interrupt position assigned first priority.

NOTE: The byte above is written only to the FPRT(L) register. Any data written to FPRT(H) is ignored.

At reset the FFFF field is set to 0, thus giving interrupt position 0 first priority.

5.10 MCTL — Mode Control Register (R16)

The contents of the MCTL set the operating mode of the NS16202 ICU. The MCTL bit map is shown below.

7	6	5	4	3	2	1	0
CFRZ	COUTD	COUTM	CLKM	FRZ	unused	NTAR	T16N8

CFRZ Determines whether or not the NS16202 counter readings are frozen. When frozen, the counters continue counting but the LCCV and HCCV registers are not updated. Reading of the true value of LCCV and HCCV is possible only while they are frozen.

CFRZ = 0 => LCCV and HCCV Not Frozen

CFRZ = 1 => LCCV and HCCV Frozen

COUTD Determines whether the COUT/SCIN pin is an input or an output. COUT/SCIN should be used as an input only for testing purposes. In this case an external sampling clock must be provided otherwise hardware interrupts will not be recognized.

COUTD = 0 => COUT/SCIN is Output

COUTD = 1 => COUT/SCIN is Input

COUTM When the COUT/SCIN pin is programmed as an output (COUTD=0), this bit determines whether the output signal is in pulsed form or in square wave form.

COUTM = 0 => Square Wave Form

COUTM = 1 => Pulsed Form

CLKM Used only in the 8-bit Bus Mode. This bit controls the clock wave form on any of the pins GO/IRO, . . . ,G3/IR6 programmed as counter output.

CLKM = 0 => Square Wave Form

CLKM = 1 => Pulsed Form

FRZ Freeze Bit. In order to allow a synchronous reading of the interrupt pending registers (IPND), their status may be frozen, causing the ICU to ignore incoming requests. This is of special importance if a polling method is used.

FRZ = 0 => IPND Not Fozen

FRZ = 1 => IPND Frozen

NTAR Determines whether the ICU is in the AUTO-ROTATE or FIXED Priority Mode. In AUTO-ROTATE mode, the interrupt source at the highest priority position, after being serviced, is assigned automatically lowest priority. In this mode, the interrupt in service always has highest priority and nesting of interrupts is therefore inhibited.

NTAR = 0 => Auto-Rotate Mode

NTAR = 1 => Fixed Mode

T16N8 Controls the data bus mode of operation.

T16N8 = 0 => 8-Bit Bus Mode

T16N8 = 1 => 16-Bit Bus Mode

At reset, all MCTL bits except COUTD, are reset to 0. COUTD is set to 1.

5.11 OCASN — Output Clock Assignment Register (R17)

Used only in the 8-bit Bus Mode. The four least significant bits of this register control the output clock assignments on pins G0/IR0, . . . ,G3/IR6. If any of these bits is set to 1, the clock generated by either the H-Counter or the H+L-Counter will be output to the corresponding pin. The four most significant bits of OCASN are not used. At Reset the four least significant bits are set to 0.

5.12 CIPTR — Counter Interrupt Pointer Register (R18)

The CIPTR register tracks the assignment of counter outputs to interrupt positions. A bit map of this register is shown below.

7	6	5	4	3	2	1	0
H	H	H	H	L	L	L	L

Where: HHHH = A 4-bit binary number identifying the interrupt position assigned to the H-Counter (or the H+L-counter if the counters are concatenated).

LLLL = A 4-bit binary number identifying the interrupt position assigned to the L-counter.

NOTE: Assignment of a counter output to an interrupt position also requires control bits to be set in the CICTL register. If a counter output is assigned to an interrupt position, external hardware interrupts at that position are ignored.

At reset, all bits in the CIPTR are set to 1. (This means both counters are assigned to interrupt position 15.)

5.13 PDAT — Port Data Register (R19)

Used only in the 8-bit Bus Mode. This register is used to input or output data through any of the pins G0/IR0, . . . ,G7/IR14 programmed as I/O ports by the IPS register. Any pin programmed as an output delivers the data written into PDAT. The input pins ignore it. Reading PDAT provides the logical value of all I/O pins, INPUT and OUTPUT.

5.14 IPS — Interrupt/Port Select Register (R20)

Used only in the 8-bit Bus Mode. This register controls the function of the pins G0/IR0, . . . ,G7/IR14. Each of these pins is individually programmed as an I/O port, if the corresponding bit of IPS is 0; as an interrupt source, if the corresponding bit is 1. The assignment of the H-Counter output to G0/IR0, . . . ,G3/IR6 by means of reg. OCASN overrides the assignment to these pins as I/O ports or interrupt inputs.

At Reset, all the IPS bits are set to 1.

5.15 PDIR — Port Direction Register (R21)

Used only in the 8-bit Bus Mode. This register determines the direction of any of the pins G0/IR0, . . . ,G7/IR14 programmed as I/O ports by the IPS register. A logic 1 indicates an input, while a logic 0 indicates an output.

At Reset, all the PDIR bits are set to 1.

5.16 CCTL — Counter Control Register (R22)

The CCTL register controls the operating modes of the counters. A bit map of CCTL is shown below.

7	6	5	4	3	2	1	0
CCON	CFNPS	COUT1	COUT0	CRUNH	CRUNL	CDCRH	CDCRL

CCON Determines whether the counters are independent or concatenated to form a single 32-bit counter (H+L-Counter). If a 32-bit counter is selected, the bits corresponding to the H-Counter will control the H+L-Counter, while the bits corresponding to the L-Counter are not used.

CCON = 0 => Two 16-bit Counters

CCON = 1 => One 32-bit Counter

CFNPS Determines whether the external clock is prescaled or not.

CFNPS = 0 => Clock Prescaled (divided by 4)

CFNPS = 1 => Clock Not Prescaled.

COUT1 &
COUT0 These bits are effective only when the COUT/SCIN pin is programmed as an OUTPUT (COUTD bit in reg. MCTL is 0). Their logic levels are decoded to provide different outputs for COUT/SCIN, as detailed in the table below:

COUT1	COUT0	COUT/SCIN Output Signal
0	0	Internal Sampling Oscillator
0	1	Zero Detect Of L-Counter
1	0	Zero Detect Of H-Counter
1	1	Zero Detect Of H+L-Counter*

*If the H- and L-Counters are not concatenated and COUT1/COUT0 are both 1, the COUT/SCIN pin is active when either counter reaches zero.

CRUNH Determines the state of either the H-Counter or the H+L-Counter, depending upon the status of CCON.

CRUNH = 0 => H-Counter or H+L-Counter Halted

CRUNH = 1 => H-Counter or H+L-Counter Running

CRUNL Effective only when CCON = 0. This bit determines whether the L-Counter is running or halted.

CRUNL = 0 => L-Counter Halted

CRUNL = 1 => L-counter Running

CDCRH Effective only when CRUNH =0 (Counter Halted). This bit is the single cycle decrement signal for either the H-Counter or the H+L-Counter.

CDCRH = 0 => No Effect

CDCRH = 1 => Decrement H-Counter or H+L-Counter

CDCRL Effective only when CRUNL = 0 and CCON = 0. This bit is the single cycle decrement signal for the L-Counter.

CDCRL = 0 => No Effect

CDCRL = 1 => Decrement L-Counter

NOTE: The bits CDCRL and CDCRH are set when a logic 1 is written into them, but, they are automatically cleared after the end of the write operation. This is needed to accomplish the decrement operation. Therefore, these bits always contain 0 when read.

Reset does not affect the CCTL bits.

5.17 CICTL — Counter Interrupt Control Register (R23)

The CICTL register controls the counter interrupts and records counter interrupt status. Interrupts can be generated from either of the 16-bit counters. When the counters are concatenated, the interrupt control is through the H-Counter control bits. In this case the CIEL bit should be set to zero to avoid spurious interrupts from the L-Counter. A bit map of the CICTL register is shown below.

7	6	5	4	3	2	1	0
CERH	CIRH	CIEH	WENH	CERL	CIRL	CIEL	WENL

CERH H-Counter Error Flag. This bit is set (1) when a second interrupt request from the H-Counter (or H+L-Counter) occurs before the first request is acknowledged.

CIRH H-Counter Interrupt Request. It is set (1) when an interrupt is pending from the H-Counter (or H+L-Counter). It is automatically reset when the interrupt is acknowledged.

CIEH H-Counter Interrupt Enable. When it is set, the H-Counter (or H+L-Counter) interrupt is enabled.

WENH H-Counter Control Write Enable. When WEHN is set (1), bits CERH, CIRH, and CIEH can be written.

CERL L-Counter Error Flag. This bit is set (1) when a second interrupt request from the L-Counter occurs before the first request is acknowledged.

CIRL L-Counter Interrupt Request. It is set (1) when an interrupt is pending from the L-Counter. It is automatically reset when the interrupt is acknowledged.

CIEL L-Counter Interrupt Enable. When it is set (1), the L-Counter interrupt is enabled.

WENL L-Counter Control Write Enable. When WENL is set (1), bits CERL, CIRL, and CIEL can be written.

NOTE: Setting the write enable bits (WENH or WENL) and writing any of the other CICTL bits are concurrent operations. That is, the ICU will ignore any attempt to alter CICTL bits if the proper write enable bit is not set in the data byte.

At reset, all CICTL bits are set to 0.

5.18 LCSV/HCSV — L-Counter Starting Value/H-Counter Starting Value Registers (R24, R25, R26, and R27)

The LCSV and HCSV registers store the start values for the L-Counter and H-Counter, respectively. Each time a counter reaches zero, the start value is automatically reloaded from either LCSV or HCSV, one clock cycle after zero count is reached. Loading LCSV or HCSV from the CPU must be synchronized to avoid writing the registers while the reloading of the counters is occurring. One method is to halt the counters while the registers are loaded.

When the 16-bit counters are concatenated, the LCSV and HCSV registers hold the 32-bit start count, with the least significant byte in R24 and the most significant byte in R27.

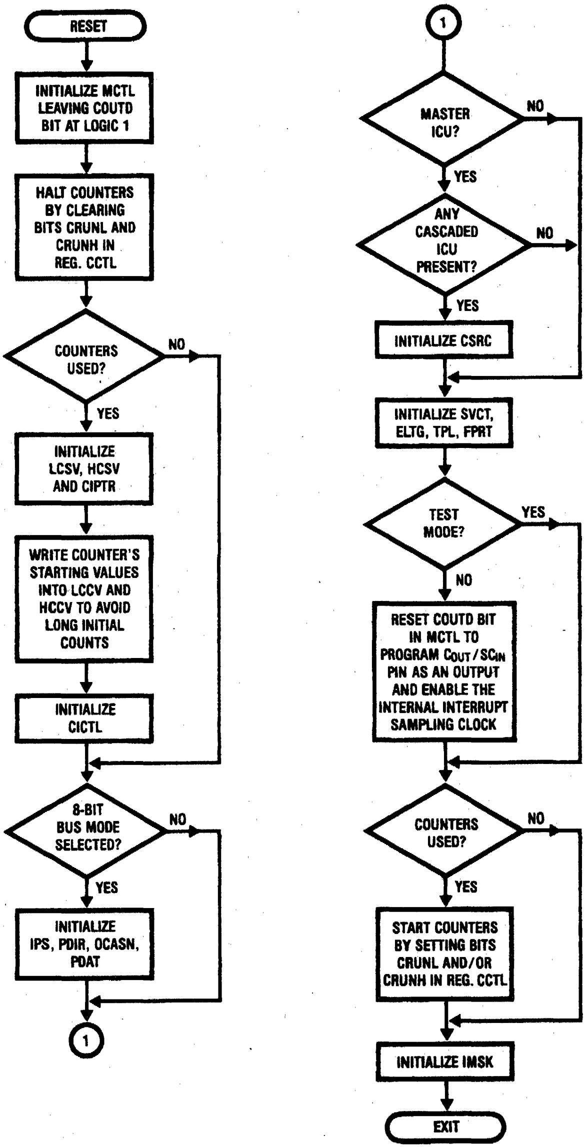
5.19 LCCV/HCCV — L-Counter Current Value/H-Counter Current Value Registers (R28, R29, R30, and R31)

The LCCV and HCCV registers hold the current value of the counters. If the CFRZ bit in the MCTL register is reset (0), these registers are updated on each clock cycle with the current value of the counters. LCCV and HCCV can be read only when the counter readings are frozen (CFRZ bit in the MCTL register is 1). They can be written only when the counters are halted (CRUNL and/or CRUNH bits in the CCTL register are 0). This last feature allows new initial count values to be loaded immediately into the counters, and can be used during initialization to avoid long initial counts.

When the 16-bit counters are concatenated, the LCCV and HCCV registers hold the 32-bit current value, with the least significant byte in R28 and the most significant byte in R31.

5.20 Register Initialization

Figure 5–2 shows a recommended initialization procedure for the ICU that sets up all the ICU registers for proper operation.


RESET
INITIALIZE MCTL LEAVING COUTD BIT AT LOGIC 1
HALT COUNTERS BY CLEARING BITS CRUNL AND CRUNH IN REG. CCTL
COUNTERS USED?
NO
YES
INITIALIZE LCSV, HCSV AND CIPTR
WRITE COUNTER'S STARTING VALUES INTO LCCV AND HCCV TO AVOID LONG INITIAL COUNTS
INITIALIZE CICTL
8-BIT BUS MODE SELECTED?
NO
YES
INITIALIZE IPS, PDIR, OCASN, PDAT
1
1
MASTER ICU?
NO
YES
ANY CASCADED ICU PRESENT?
NO
YES
INITIALIZE CSRC
INITIALIZE SVCT, ELTG, TPL, FPRT
TEST MODE?
YES
NO
RESET COUTD BIT IN MCTL TO PROGRAM COUT/SCIN PIN AS AN OUTPUT AND ENABLE THE INTERNAL INTERRUPT SAMPLING CLOCK
COUNTERS USED?
NO
YES
START COUNTERS BY SETTING BITS CRUNL AND/OR CRUNH IN REG. CCTL
INITIALIZE IMSK
EXIT

TL/C/5117-15

FIGURE 5-2. Recommended ICU's Initialization Sequence.

6 AC Electrical Characteristics

The timing parameters given in this section are defined in Table 6-1. All timing specifications refer to 50% of the rising or falling edge of the appropriate clock waveform and to 0.8V or 2.0V on the appropriate signal as shown in Figure 6-1.

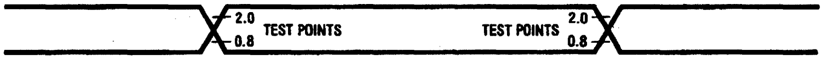
Parameters are valid under the following conditions:

Ambient Temperature	0°C to 70°C
Output Capacitive Load	100 pF max.

Timing specifications for the NS16202 ICU are shown in Figures 6-2 through 6-7.

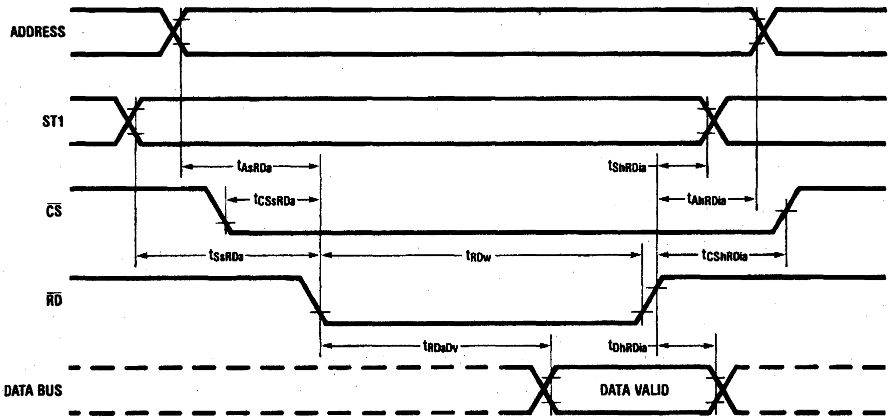
TABLE 6-1. Timing Parameter Definitions

Symbol	Parameter	Min	Max	Units
READ CYCLE				
tAhRDia	Address Hold Time after $\overline{RD}$ Inactive	80		ns
tAsRDa	Address Setup Time before $\overline{RD}$ Active	52		ns
tCShRDia	$\overline{CS}$ Hold Time after $\overline{RD}$ Inactive	80		ns
tCSsRDa	$\overline{CS}$ Setup Time before $\overline{RD}$ Active	40		ns
tDhRDia	Data Hold Time after $\overline{RD}$ Inactive	0	50	ns
tRDaDv	Delay from Read Active to Data Valid		135	ns
tRDw	Read Strobe Width	160		ns
tShRDa	ST1 Setup Time before $\overline{RD}$ Active	50		ns
tSsRDia	ST1 Hold Time after $\overline{RD}$ Inactive	50		ns
WRITE CYCLE				
tAhWRia	Address Hold Time after $\overline{WR}$ Inactive	80		ns
tAsWRa	Address Setup Time before $\overline{WR}$ Active	52		ns
tCShWRia	$\overline{CS}$ Hold Time after $\overline{WR}$ Inactive	80		ns
tCSsWRa	$\overline{CS}$ Setup Time before $\overline{WR}$ Active.	40		ns
tDhWRia	Data Hold Time after $\overline{WR}$ Inactive	150		ns
tDsWRia	Data Setup Time before $\overline{WR}$ Inactive	150		ns
tWRiaPf	Delay from $\overline{WR}$ Inactive (to rg. PDIR) to Port Floating		200	ns
tWRiaPv	Delay from $\overline{WR}$ Inactive to Port Output Valid		200	ns
tWRw	Write Strobe Width	160		ns
OTHER TIMINGS				
tCOUTh	External Sampling Clock High Pulse Width	400		ns
tCOUTl	External Sampling Clock Low Pulse Width	25		ns
tCOUTp	External Sampling Clock Period	425		ns
tCh	External Clock High Time (Without Prescaler)	100		ns
tChp	External Clock High Time (With Prescaler)	50		ns
tC1	External Clock Low Time (without Prescaler)	100		ns
tC1p	External Clock Low Time (with Prescaler)	50		ns
tCtCOUTt	Delay from F.E. of CLK to COUT/SCIN transition		200	ns
tCtGt	Delay from F.E of CLK to G0/IR0, . . . ,G3/IR6 Transitions		200	ns
tCy	Extenal Clock Period (Without Prescaler)	400		ns
tCyp	External Clock Period (With Prescaler)	100		ns
tIRd	Interrupt Signal Duration in Level Trisser	400		ns
tIRi	Interval Between Requests in Edge Trisser	400		ns
tIRld	Interrupt Request to $\overline{INT}$ out Delay		450	ns
tIRw	Interrupt Request Pulse Width in Edge Trisser	25		ns
tPtDt	Delay from Port Input Transition to Data Transition		200	ns
tRSTw	Reset Pulse Width	400		



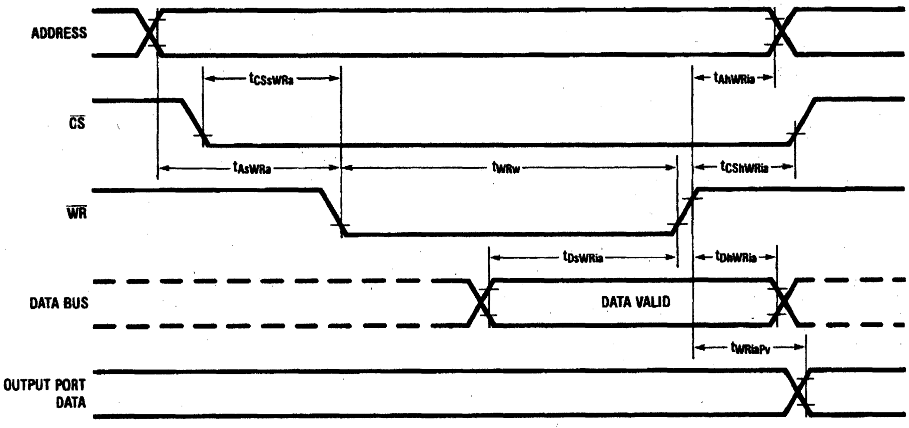
TL/C/5117-16

FIGURE 6-1. Timing Specification Standard.



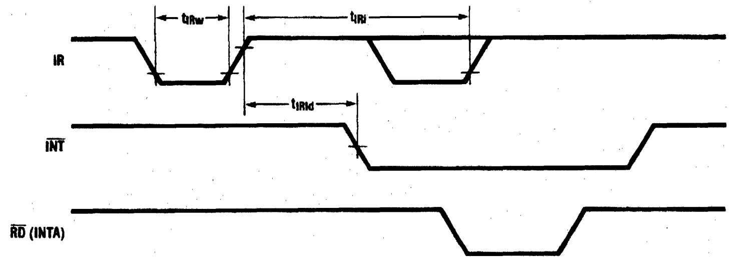
TL/C/5117-17

FIGURE 6-2. READ/INTA Cycle.



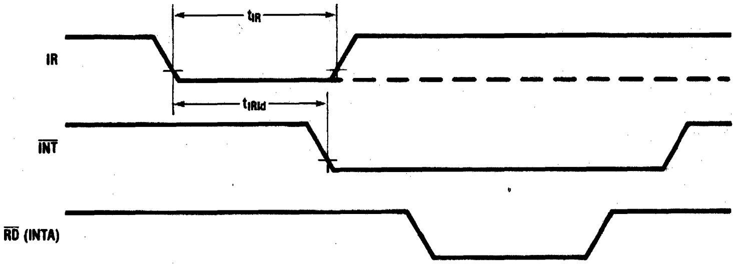
TL/C/5117-18

FIGURE 6-3. Write Cycle.



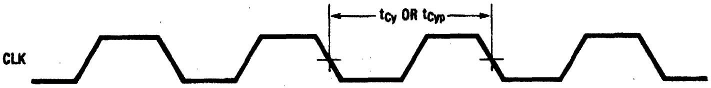
TL/C/5117-19

FIGURE 6-4. Interrupt Timing in Edge Triggering Mode.



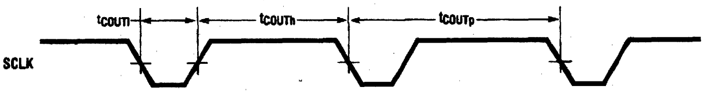
TL/C/5117-20

FIGURE 6-5. Interrupt Timing in Level Triggering Mode.



TL/C/5117-21

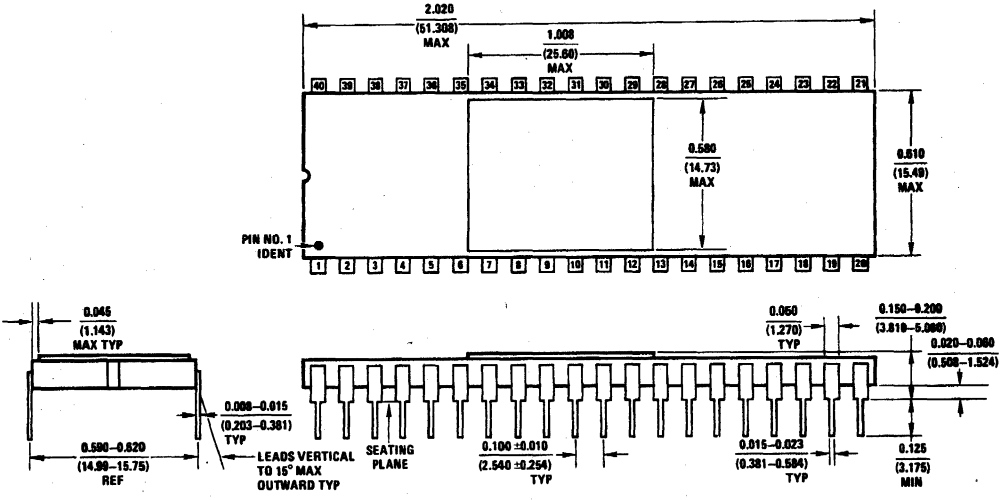
FIGURE 6-6. External Clock Waveform to be Provided at Pin CLK.



TL/C/5117-22

FIGURE 6-7. External Interrupt-Sampling-Clock Waveform to be Provided at Pin COUT/SCIN when in Test Mode.

Physical Dimensions

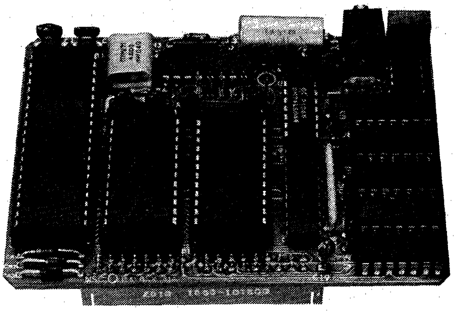


D40C (REV F)

development tools

National Semiconductor

BLX-281A, BLX-281B Speech Synthesis Expansion Modules



- **BLX bus-compatible I/O expansion**
- **Speech synthesis based on DIGITALKER™**
- **Large vocabulary adequate for most applications**
- **On-board filter and half-watt amplifier**
- **Two versions**
  - BLX-281A, factory shipped with installed Maxi-ROMs, contains speech data
  - BLX-281B, factory shipped without ROM/PROMs installed to allow user to implement a custom vocabulary
- **Simple operation for user**
  - I/O write with word/sound address
  - Interrupt asserted when complete
- **BLX bus on-board expansion elimates Multibus™ system bus latency and increases system throughput**
- **Upward compatible with BLX-281 Speech Synthesis Module**

Product Overview

The BLX-281A and BLX-281B Speech Synthesis Expansion Modules are members of the growing line of BLX bus-compatible expansion module products from National Semiconductor Corporation. The BLX-281A and BLX-281B are two different configurations of the same module. The BLX-281A is factory shipped with installed Maxi-ROMs (MM52164-SSR1 and SSR2). The BLX-281B is factory shipped with no ROMs or PROMS installed, allowing the user to install any one of a variety of compatible ROM/PROM devices programmed by the user for a custom vocabulary. Typically, the BLX-281A will be used to experiment with speech synthesis while the BLX-281B is purchased for production by the OEM.

The BLX-281A(B) plugs directly into any BLX bus-compatible host board offering low cost incremental on-board expansion. As a result, any BLX bus-compatible host board may be given the ability to "speak." By simply adding a speaker to a system containing the BLX-281A (programmed PROM too on the BLX-281B), many users can do away with CRTs, printers, rows of LEDs, or other communications devices. This will lower the cost of most systems, and has the added benefit of removing messages which are potentially ambiguous and hard-to-understand for untrained users. The BLX-281A/B has a total capacity of 288 words, sounds, tones, and durations of silence (depending on ROM/PROM devices used), each of which has a unique address. A table of addresses (desired words/sounds) is built, and passed to the BLX-281A/B. An on-board filter and amplifier provide the actual speech signal to a standard miniature phone jack. The BLX module is closely coupled to the host board through the BLX bus, and as such, offers maximum on-board performance, and frees Multibus system traffic for other system resources. Incremental power dissipation is minimal, requiring only 3.7 watts on the BLX-281A.

Functional Description

The BLX-281A/B Speech Synthesis Expansion Module uses the MM54104 Speech Processor Chip from National Semiconductor Corporation. The digitized and compressed speech data for the BLX-281A are contained in an MM52164-SSR1 and SSR2 Maxi-ROM and provide a 144-word vocabulary. The ROM/PROM devices compatible with the BLX-281B are:

ROM	PROM
Super Maxi-ROM (MM42128)	MM2716 (2k UV EPROM)
	MM2732 (4k UV EPROM)
Maxi-ROM (MM52164-SSR1, SSR2)	MM2764 (8k UV EPROM)

The system software communicates with the BLX-281 across the BLX bus using I/O read/write commands.

Vocabulary

The standard vocabulary set offered on the BLX-281 is shown in Table I, along with the assigned addresses for each item. By combining the appropriate words, sounds, tones, and silence durations, speech can be generated to satisfy many applications.

Words required, but not found in the table, can frequently be built. Examples of this are: combine "RE" with "SET" for "RESET", or combine "DEGREE" with "SS" for "DEGREES".

In normal human speech, the brain puts durations of silence between the words to make the sentence flow smoothly. This is provided for in the BLX-281A (see Table I). A suggestion for improved phrase quality is to insert 80ms of silence prior to words to words beginning with the letters K, T, P, B, D, and G, and to add 40ms of silence after words ending in those same letters.

The "voice" output of the BLX-281A/B is a highly intelligible, male voice. If another voice is required, or the application is non-English, or involves unusual terminology, any voice can be processed for use on the BLX-281A/B by the factory.

Host Interface

The BLX bus-compatible host board merely passes the address of the desired word/sound to the BLX-281A/B Speech Synthesis Expansion Module via an I/O write. When the operation is complete, an interrupt is generated. This informs the host of the end of the speech sequence, and allows for cascading of addresses for true, human-quality sentences.

Interrupt Requests

There is one interrupt line from the Speech Processor Chip that generates an interrupt request to the host CPU. It is active on completion of each speech sequence. It is cleared by an I/O read to the BLX-281A/B.

Installation

The BLX-281A/B module plugs directly into either of the female BLX connectors on the host board. The module is then secured at one additional point with nylon hardware to insure the mechanical security of the assembly (see Figures 1 and 2).

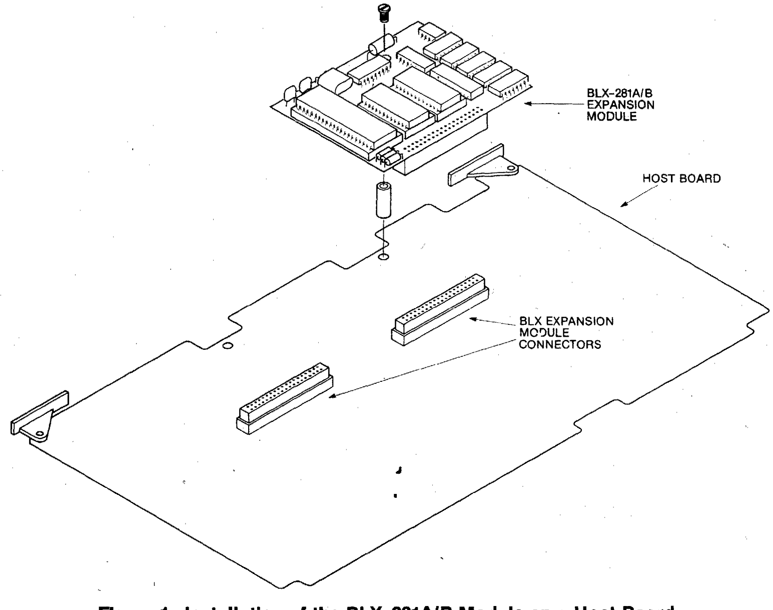


Figure 1. Installation of the BLX-281A/B Module on a Host Board

Table I. Master Word List (BLX-281A)

Word	8-Bit Binary Address	Word	8-Bit Binary Address	Word	8-Bit Binary Address
	Bit 7 Bit 0		Bit 7 Bit 0		Bit 7 Bit 0
THIS IS DIGITALKER	00000000	Q	00110000	IS	01100000
ONE	00000001	R	00110001	IT	01100001
TWO	00000010	S	00110010	KILO	01100010
THREE	00000011	T	00110011	LEFT	01100011
FOUR	00000100	U	00110100	LESS	01100100
FIVE	00000101	V	00110101	LESSER	01100101
SIX	00000110	W	00110110	LIMIT	01100110
SEVEN	00000111	X	00110111	LOW	01100111
EIGHT	00001000	Y	00111000	LOWER	01101000
NINE	00001001	Z	00111001	MARK	01101001
TEN	00001010	AGAIN	00111010	METER	01101010
ELEVEN	00001011	AMPERE	00111011	MILE	01101011
TWELVE	00001100	AND	00111100	MILLI	01101100
THIRTEEN	00001101	AT	00111101	MINUS	01101101
FOURTEEN	00001110	CANCEL	00111110	MINUTE	01101110
FIFTEEN	00001111	CASE	00111111	NEAR	01101111
SIXTEEN	00010000	CENT	01000000	NUMBER	01110000
SEVENTEEN	00010001	400Hz TONE	01000001	OF	01110001
EIGHTEEN	00010010	80Hz TONE	01000010	OFF	01110010
NINETEEN	00010011	20ms SILENCE	01000011	ON	01110011
TWENTY	00010100	40ms SILENCE	01000100	OUT	01110100
THIRTY	00010101	80ms SILENCE	01000101	OVER	01110101
FORTY	00010110	160ms SILENCE	01000110	PARENTHESIS	01110110
FIFTY	00010111	320ms SILENCE	01000111	PERCENT	01110111
SIXTY	00011000	CENTI	01001000	PLEASE	01111000
SEVENTY	00011001	CHECK	01001001	PLUS	01111001
EIGHTY	00011010	COMMA	01001010	POINT	01111010
NINETY	00011011	CONTROL	01001011	POUND	01111011
HUNDRED	00011100	DANGER	01001100	PULSES	01111100
THOUSAND	00011101	DEGREE	01001101	RATE	01111101
MILLION	00011110	DOLLAR	01001110	RE	01111110
ZERO	00011111	DOWN	01001111	READY	01111111
A	00100000	EQUAL	01010000	RIGHT	10000000
B	00100001	ERROR	01010001	SS	10000001
C	00100010	FEET	01010010	SECOND	10000010
D	00100011	FLOW	01010011	SET	10000011
E	00100100	FUEL	01010100	SPACE	10000100
F	00100101	GALLON	01010101	SPEED	10000101
G	00100110	GO	01010110	STAR	10000110
H	00100111	GRAM	01010111	START	10000111
I	00101000	GREAT	01011000	STOP	10001000
J	00101001	GREATER	01011001	THAN	10001001
K	00101010	HAVE	01011010	THE	10001010
L	00101011	HIGH	01011011	TIME	10001011
M	00101100	HIGHER	01011100	TRY	10001100
N	00101101	HOUR	01011101	UP	10001101
O	00101110	IN	01011110	VOLT	10001110
P	00101111	INCHES	01011111	WEIGHT	10001111

Note: Address $8F_H$ is the last legal address in this word list. Exceeding address $8F_H$ will produce pieces of unintelligible, invalid speech data.

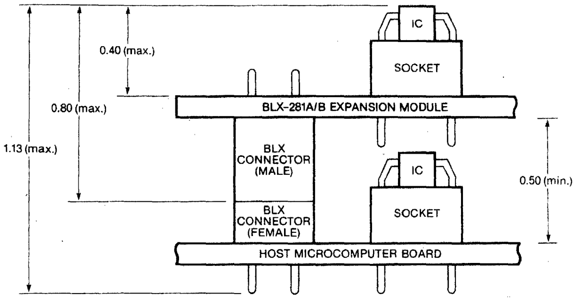


Figure 2. BLX-281A/B Expansion Module Mounting Clearances (inches)

Specifications

Word Size

Data — 8 bits

I/O Addressing

Function	Type of Operation	BLX Connector Port Address
Data Transfer	Write	X0-XF
Interrupt Clear	Read	X0-XF

Note: The port addresses are determined on the host BLC microcomputer. Refer to the Hardware Reference Manual for your host BLC microcomputer to determine the first digit (X) of the connector port address.

Vocabulary — See Table I

Interrupts — One interrupt request at end of speech sequence

Interfaces BLX Bus — All signals TTL compatible
Speaker Port — ½W audio signal into 4-8Ω

Speaker Port Connector Standard miniature phone-jack

Physical Height: 2.85 in. (7.24 cm)
Width: 3.70 in. (9.40 cm)
Depth:
BLX-281A/B Module
0.80 in. (2.04 cm)
BLX-281A/B Module + Host Board
1.13 in. (2.86 cm)
Weight: 1.7 oz. (48 gm)

Electrical $+5V_{DC} \pm 5\%$ @ 385 mA
$+12V_{DC} \pm 5\%$ @ 150 mA

Environmental Operating Temperature: 0°C to 55°C
Relative Humidity: 0% to 90%, non-condensing

Order Information

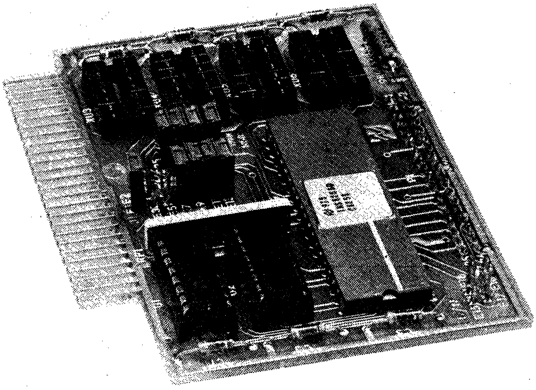
BLX-281A	Speech Synthesis Expansion Module with Maxi-ROMs
BLX-281B	Speech Synthesis Expansion Module without ROM/PROM

Documentation

BLX-281AM	BLX-281A Speech Synthesis Expansion Module User's Manual
BLX-281BM	BLX-281B Speech Synthesis Expansion Module User's Manual

National Semiconductor

BLX-350
Parallel I/O Expansion Module



- BLX bus compatible I/O expansion
- 24 programmable I/O lines with sockets for interchangeable line drivers and terminators
- Three jumper selectable interrupt request sources
- Accessed as I/O port locations
- Single +5V lower power requirement
- BLX bus on-board expansion eliminates Multibus™ system bus latency and increases system throughput

Product Overview

The BLX-350 Parallel I/O Expansion Module is a member of the new line of BLX bus compatible expansion modules from National Semiconductor Corporation. The BLX-350 plugs directly into any BLX bus compatible host board offering incremental on-board expansion. The BLX-350 module provides 24 programmable I/O lines with sockets for interchangeable line drivers and terminators. The BLX board is closely coupled to the host board through the BLX bus, and as such, offers maximum on-board performance and frees Multibus system traffic for other system resources. In addition, incremental power dissipation is minimal, requiring only 1.6 watts (not including optional driver/terminators).

Functional Description

Programmable Interface

The BLX-350 module uses an 8255A-5 Programmable Peripheral Interface (PPI) providing 24 parallel I/O lines. The base-board system software is used to configure the I/O lines in any combination of unidirectional input/output and bidirectional ports indicated in Table I. Therefore, the I/O interface may be customized to meet specific peripheral requirements. In order to take full advantage of the large number of possible I/O configurations, sockets are provided for interchangeable I/O line drivers and terminators. Hence, the flexibility of the I/O interface is further enhanced by the capability of selecting the appropriate combination of optional line drivers and terminators to provide the required sink current, polarity, and driver/termination characteristics for each application. In addition, inverting bidirectional bus drivers (8226) are provided on sockets to allow convenient optional replacement to non-inverting drivers (8216). The 24 programmable I/O lines, signal ground, and +5 volt power (jumper configurable) are brought to a 50-pin edge connector that mates with flat, woven, or round cable.

Interrupt Request Generation

Interrupt requests may originate from three jumper selectable sources. Two interrupt requests can be automatically generated by the PPI when a byte of information is ready to be transferred to the base board CPU (i.e., input buffer is full) or a byte of information has been transferred to a peripheral device (i.e., output buffer is empty). A third interrupt source may originate directly from the user I/O interface (J1 connector).

Installation

The BLX-350 module plugs directly into either of the female BLX connectors on the host board. The module is then secured at one additional point with nylon hardware to insure the mechanical security of the assembly (see Figure 1 and Figure 2).

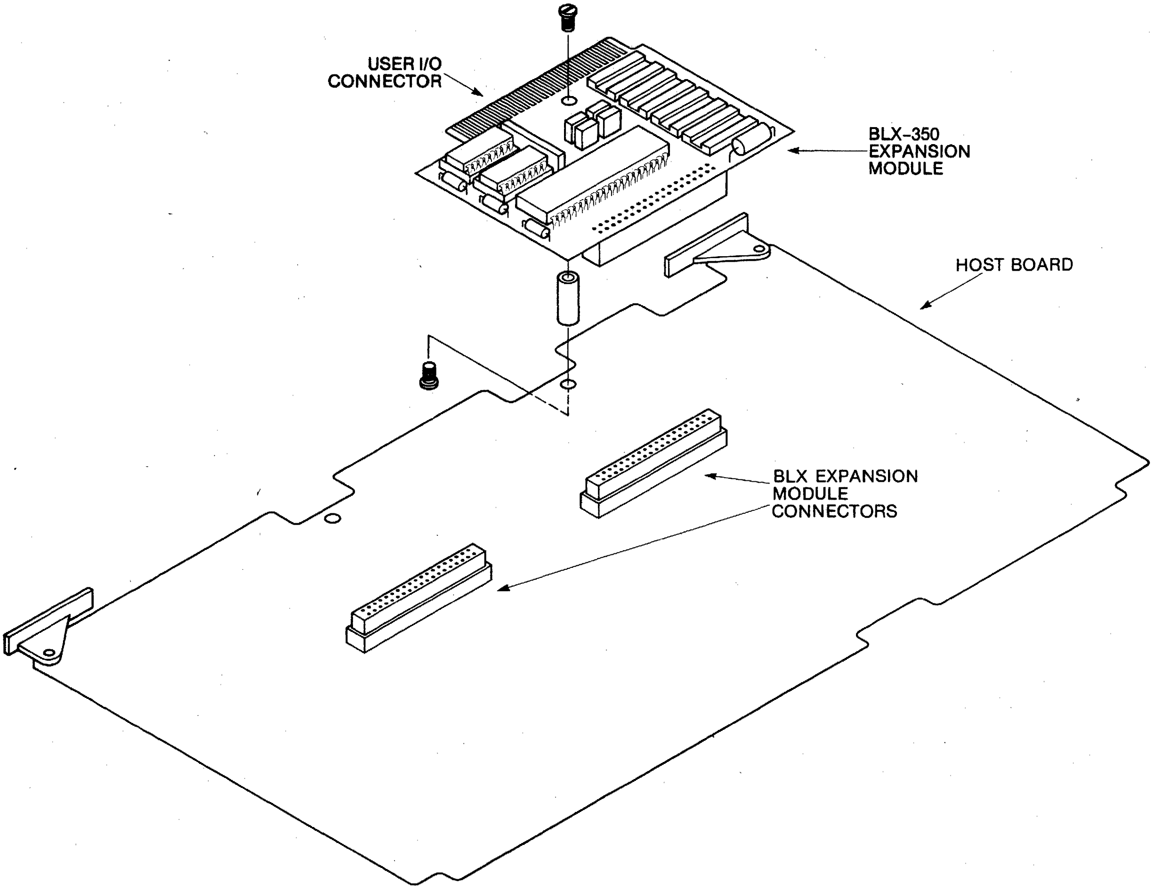


Figure 1. Installation of BLX-350 Module on a Host Board

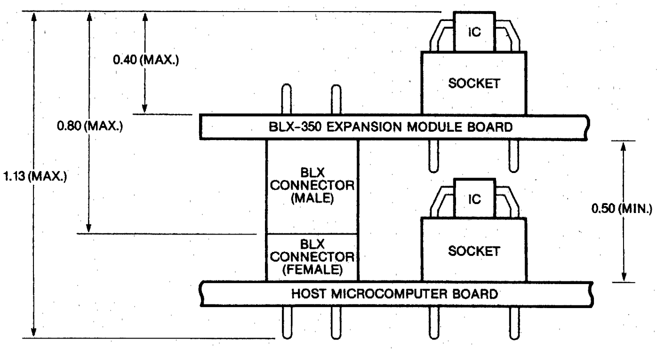


Figure 2. BLX-350 Expansion Module Mounting Clearances (inches)

Table I. Input/Output Port Modes of Operation

Port	Lines (Qty.)	Mode of Operation					Control
		Unidirectional				Bidirectional	
		Input		Output			
		Unlatched	Latched & Strobed	Latched	Latched & Strobed		
A	8	X	X	X	X	X	
B	8	X	X	X	X		
C	4	X		X			X[1]
	4	X		X			X[1]

Note 1. Part of port C must be used as a control port when either port A or port B are used as a latched and strobed input or a latched and strobed output port, or port A is used as a bidirectional port.

Specifications

Word Size

Data — 8 Bits

I/O Addressing

8255A-5 Ports	BLX-350 Address
Port A	X0 or X4
Port B	X1 or X5
Port C	X2 or X6
Control	X3 or X7
Reserved	X8 to XF

Note: The first digit of each port I/O address is listed as "X" since it will change dependent upon the type of host BLC microcomputer used. Refer to the Hardware Reference Manual for your host BLC microcomputer to determine the first digit of the port address.

I/O Capacity 24 programmable lines (see Table I)

Access Time Read: 250 ns max.

Write: 300 ns max.

Note: Actual transfer speed is dependent upon the cycle time of the host microcomputer.

Interrupts Interrupt requests may originate from the programmable peripheral interface (2) or the user specified I/O (1).

Interfaces

BLX Bus — All signals TTL compatible

Parallel I/O — All signals TTL compatible

Parallel I/O Connectors All 25 pairs/50 pins on 0.1″ centers

Connector Type	Vendor	Vendor Part No.
Female	3M	3415-0000 with ears
Flat Crimp	3M AMP Ansley SAE	3415-0000 with ears 88083-1 609-5015 SD6750 Series
Female, Soldered	AMP Viking TI	2-583485-6 3VH25/1JV5 H312125
Female, Wirewrap	TI Viking ITT Cannon	H311125 3VH15/1JND5 EC4A050A1A

Line Drivers and Terminators

I/O Drivers — The following line drivers and terminators are all compatible with the I/O driver sockets on the BLX-350.

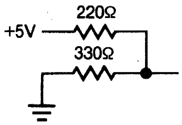
Driver	Characteristic	Sink Current (mA)
7438	I, OC	48
7437	I	48
7432	NI	16
7426	I,OC	16
7409	NI, OC	16
7408	NI	16
7403	I, OC	16
7400	I	16

Note: I = Inverting, NI = Non-Inverting, OC = Open Collector

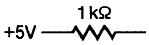
Port 1 has 25 mA totem pole drivers and 1 kΩ terminators.

I/O Terminators — 220Ω/330Ω divider or 1 kΩ pull-up.

220 Ω/330 Ω (BLC-901)



1 KΩ (BLC-902)



Physical

Width: 2.85 in. (7.24 cm)
Length: 3.70 in. (9.40 cm)
Height:*
BLX-350 Board
0.80 in. (2.04 cm)
BLX-350 Board plus Host
1.13 in. (2.86 cm)
Weight: 1.79 oz. (51 gm)

*See Figure 2

Electrical Characteristics

DC Power Requirements

+5V @ 320 mA
Sockets XU3, XU4, XU5, and XU6 empty (as shipped).

+5V @ 500 mA
Sockets XU3, XU4, XU5, and XU6 contain 7438 buffers.

+5V @ 620 mA
Sockets XU3, XU4, XU5, and XU6 contain BLC-901 termination devices

Environmental Operating Temperature:
0°C to 55°C
Relative Humidity: 0% to 90% non-condensing

Ordering Information

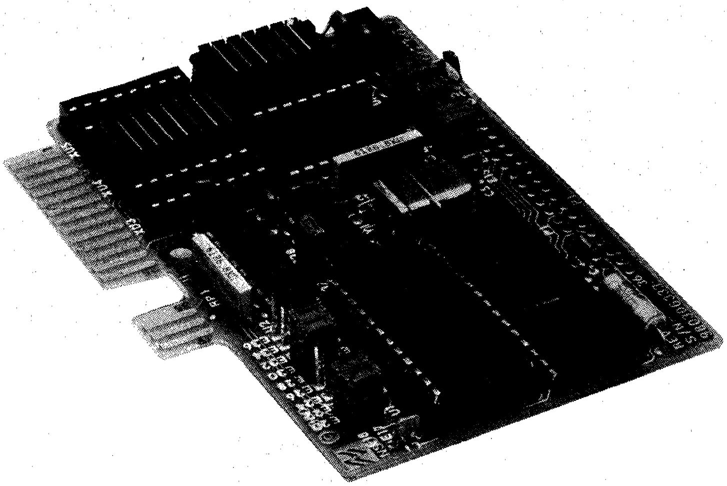
BLX-350	Parallel I/O Expansion Module

Documentation

420306332-001	BLX-350 Parallel I/O Expansion Module User's Manual

National Semiconductor

BLX-351
Serial I/O Expansion Module



- **BLX bus compatible I/O expansion**
- **Programmable synchronous/asynchronous communications channel with RS232C or RS449/442 interface**
- **Software programmable baud rate generator**
- **Two programmable 16-bit BCD or binary timers/event counters**
- **Four jumper selectable interrupt request sources**
- **Accessed as I/O port locations**
- **Low power requirements**
- **Single +5V when configured for RS449/442 interface**
- **BLX bus on-board expansion eliminates Multibus™ system bus latency and increases system throughput**

Product Overview

The BLX-351 Serial I/O Expansion Module is a member of the new line of BLX bus compatible expansion module products from National Semiconductor Corporation. The BLX-351 plugs directly into any BLX bus compatible host board offering incremental on-board I/O expansion. The BLX-351 module provides one RS232C or RS449/442 programmable synchronous/asynchronous communications channel with software selectable baud rates. Two general purpose programmable 16-bit BCD or binary timers/event counters are available to the host board to generate accurate time intervals under software control. The BLX board is closely coupled to the host board through the BLX bus, and as such, offers maximum on-board performance and frees Multibus system traffic for other system resources. In addition, incremental power dissipation is minimal, requiring only 3.3 watts (assumes RS232C interface).

Functional Description

Communications Interface

The BLX-351 module uses the 8251A Universal Synchronous / Asynchronous Receiver / Transmitter (USART) providing one programmable communications channel. The USART can be programmed by the system software to individually select the desired asynchronous or synchronous serial data transmission technique (including IBM Bi-Sync). The mode of operation (i.e. synchronous or asynchronous), data format, control character format, parity, and baud rate are all under program control. The 8251A provides full duplex, double buffered transmit and receive capability. Parity, overrun, and framing error detection are all incorporated in the USART. The command lines, serial data lines, and signal ground lines are brought out to a double edge connector configurable for either an RS232C or RS449/442 interface (see Figure 3). In addition, the BLX-351 module is jumper configurable for either point-to-point or multidrop network connection.

16-Bit Interval Timers

The BLX-351 module uses an 8253 Programmable Interval Timer (PIT) providing 3 fully programmable and independent BCD and binary 16-bit interval timers. One timer is available to the system designer to generate baud rates for the USART under software control. Routing for the outputs from the other two counters is jumper selectable to the host board. In utilizing the BLX-351 module, the systems designer simply configures, via software, each timer independently to meet system requirements. Whenever a given baud rate or time delay is needed, software commmands the programmable timers to select the desired function. The functions of the timers are shown in Table 1. The contents of each counter may be read at any time during system operation.

Interrupt Request Lines

Interrupt requests may originate from four sources. Two interrupt requests can be automatically generated by the USART when a character is ready to be transferred to the host board (i.e. receive buffer is full) or a character has been transmitted (i.e. transmit buffer is empty). In addition, two jumper selectable requests can be generated by the programmable timers.

Installation

The BLX-351 module plugs directly into either of the female BLX connectors on the host board. The module is then secured at one additional point with nylon hardware to insure the mechanical security of the assembly (see Figures 1 and 2).

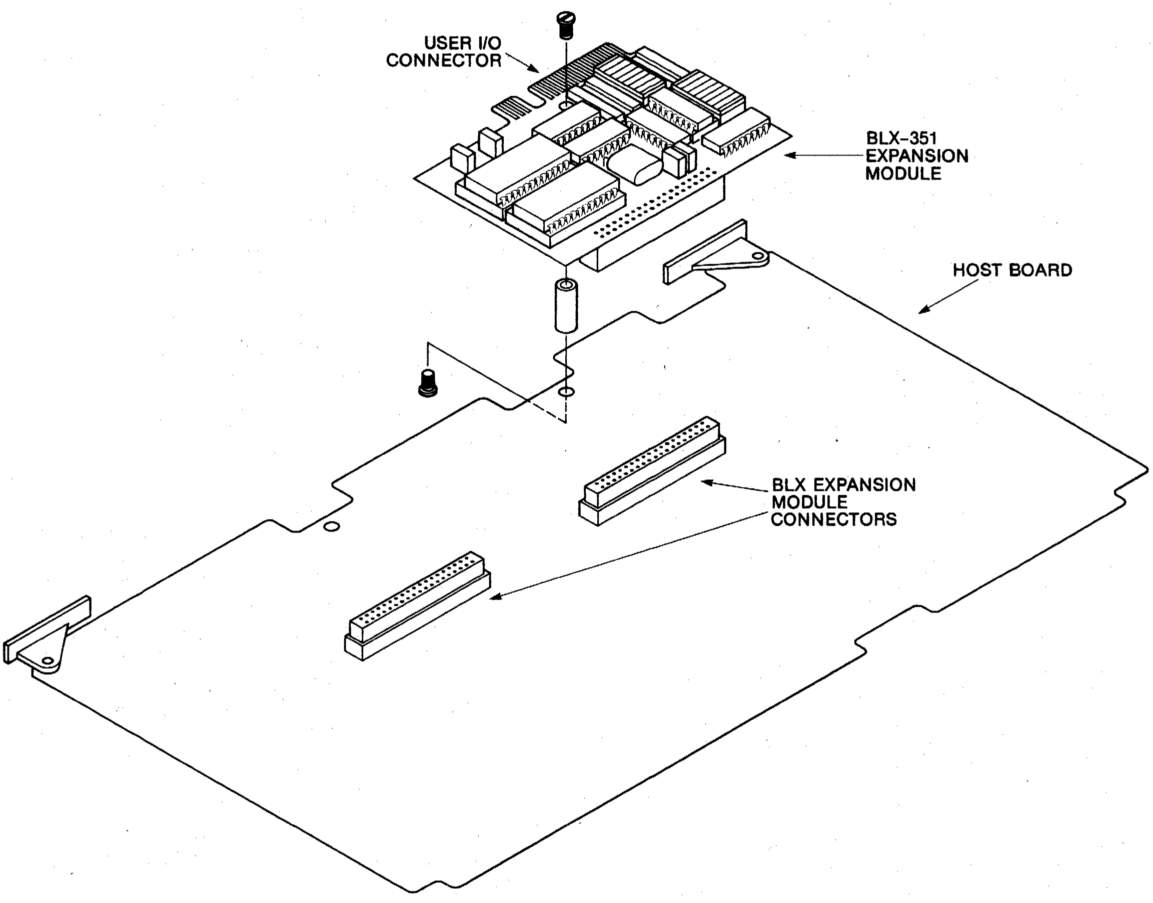


Figure 1. Installation of BLX-351 Module on a Host Board

Table 1. Programmable Timer Functions

Function	Operation
Interrupt on terminal count	When terminal count is reached, an interrupt request is generated. This function is useful for generation of real-time clocks.
Programmable one-shot	Output goes low upon receipt of an external trigger edge and returns high when terminal count is reached. This function is retriggerable.
Rate generator	Divide by N counter. The output will go low for one input clock cycle, and the period from one low going pulse to the next is N times the input clock period.
Square-wave rate generator	Output will remain high until one-half the count has been completed, and go low for the other half of the count.
Software triggered strobe	Output remains high until software loads count (N). N counts after count is loaded, output goes low for one input clock period.
Hardware triggered strobe	Output goes low for one clock period N counts after rising edge counter trigger input. The counter is retriggerable.
Event counter	On a jumper selectable basis, the clock input becomes an input from the external system. CPU may read the number of events occurring after the counting "window" has been enabled or an interrupt may be generated after N events occur in the system.

Specifications

Word Size

Data — 8 bits

I/O Addressing

I/O Address	Chip Select	Function
X0, X2, X4, or X6	8251A USART	Write: Data Read: Data
X1, X3, X5, or X7		Write: Mode or Command Read: Status
X8 or XC	8253 PIT	Write: Counter 0 Load Count ÷ N) Read: Counter 0
X9 or XD		Write: Counter 1 (Load Count ÷ N) Read: Counter 1
XA or XE		Write: Counter 2 (Load Count ÷ N) Read: Counter 2
XB or XF		Write: Control Read: None

NOTE: The first digit of each port I/O address is listed as "X" since it will change depending on the type of host BLC microcomputer used. Refer to the Hardware Reference Manual for your host microcomputer to determine the first digit of the I/O address.

Access Time

Read — 250 ns max.

Write — 300 ns max.

Note: Actual transfer speed is dependent upon the cycle time of the host microcomputer.

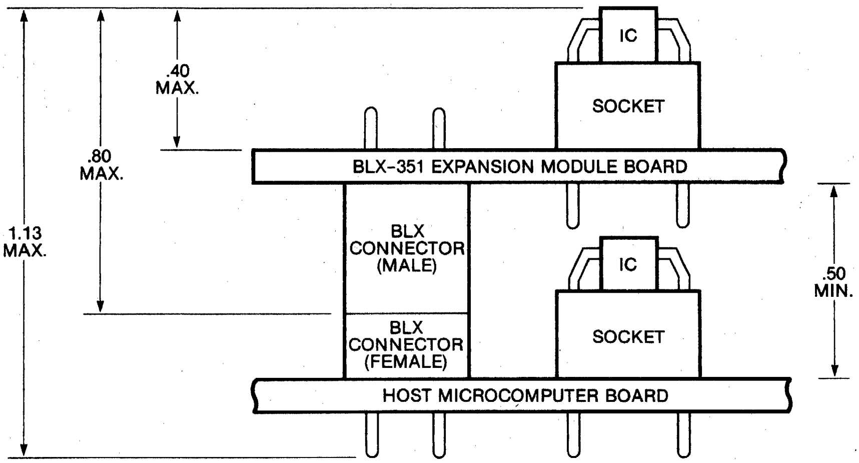


Figure 2. BLX-351 Expansion Module Mounting Clearance (Inches)

Serial Communications

Synchronous — 5-8-bit characters; internal character synchronization; automatic sync insertion; even, odd or no parity generation/detection.

Asynchronous — 5-8-bit characters; break character generation and detection; 1, 1½, or 2 stop bits; false start bit detection; even, odd or no parity generation/detection.

Sample Baud Rate:

8253 PIT Frequency[1] (KHz, Software Selectable)	8251A USART Baud Rate (Hz)[2]		
	Synchronous	Asynchronous	
		÷16	÷64
307.2	—	19200	4800
153.6	—	9600	2400
76.8	—	4800	1200
38.4	38400	2400	600
19.2	19200	1200	300
9.6	9600	600	150
4.8	4800	300	75
2.4	2400	150	—
1.76	1760	110	—

Note 1: Frequency selected by I/O writes of appropriate 16-bit frequency factor to Baud Rate Register.

Note 2: Baud rates shown here are only a sample subset of possible software-programmable rates available. Any frequency from 18.75 Hz to 614.4 KHz may be generated utilizing on-board crystal oscillator and 16-bit Programmable Interval Timer (used here as frequency divider).

Interval Timer and Baud Rate Generator

Input Frequency (selectable):

1.23 MHz ± 0.1% (.813 μs period nominal)
153.6 KHz ± 0.1% (6.5 μs period nominal)

Output Frequency:

	Rate Generator (Frequency)		Real-Time Interrupt (Interval)	
	Min.	Max.	Min.	Max.
Single Timer[1]	18.75 Hz	614.4 Hz	1.63 μs	53.3 ms
Single Timer[2]	2.34 Hz	76.8 KHz	13.0 μs	426.7 ms
Dual Timer[3] (Counters 0 and 1 in series)	0.000286 Hz	307.2 KHz	3.26 μs	58.25 min.
Dual Timer[4] (Counters 0 and 1 in series)	0.0000358 Hz	38.4 KHz	26.0 μs	7.77 hrs.

Note 1: Assuming 1.23 MHz clock input.

Note 2: Assuming 153.6 KHz clock input.

Note 3: Assuming Counter 0 has 1.23 MHz clock input.

Note 4: Assuming Counter 0 has 153.6 KHz clock input.

Interrupts

Interrupt requests may originate from the USART (2) or the programmable timer (2).

Interfaces

BLX bus — all signals TTL compatible

Serial — configurable for EIA Standards RS232C or RS449/422

EIA Standard RS232C signals provided and supported:

Clear to Send (CTS)
Data Set Ready (DSR)
Data Terminal Ready (DTR)
Request to Send (RTS)
Receive Clock (RXC)
Receive Data (RXD)
Transmit Clock (DTE TXC)
Transmit Data (TXD)

EIA Standard RS449/422 signals provided and supported:

Clear to Send (CS)
Data Mode (DM)
Terminal Ready (TR)
Request to Send (RS)
Receive Timing (RT)
Receive Data (RD)
Terminal Timing (TT)
Send Data (SD)

Physical

Width: 2.85 in. (7.24 cm.)
Length: 3.70 in. (9.40 cm)
Height*: BLX-351 Board
0.80 in. (2.04 cm.)
BLX-351 Board and Host Board
1.13 in. (2.86 cm)
Weight: 1.79 oz. (51 gm.)

*(See Figure 2)

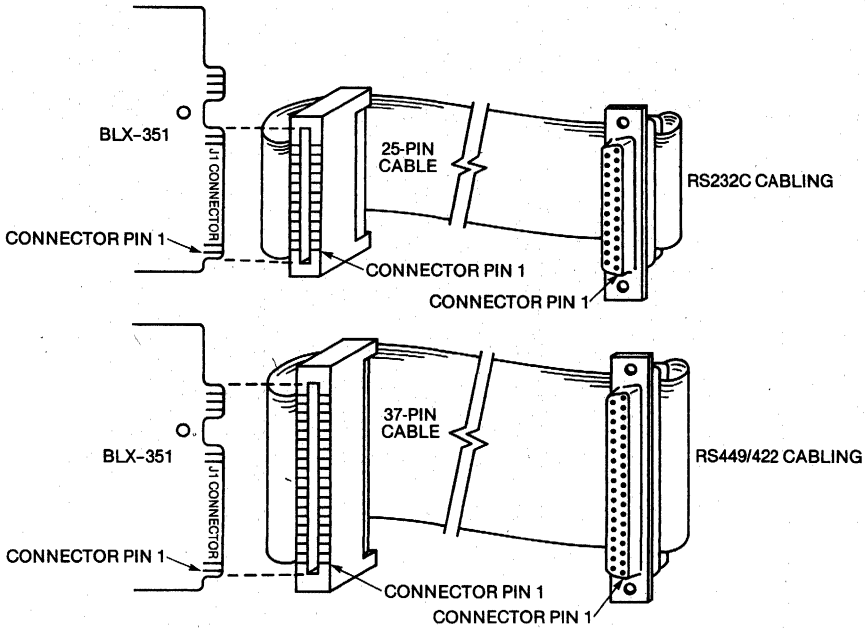


Figure 3. Cable Construction and Installation for RS232C and RS449/422 Interface

Serial Interface Connectors

Configuration	Mode[2]	Expansion Module Edge Connector	Cable	Connector
RS232C	DTE	26-pin[5], 3M-3642-0001	3M[3]-3349/25	25-pin[7], 3M-3482-1000
RS232C	DCE	26-pin[5], 3M-3642-0001	3M[3]-3349/25	25-pin[7], 3M-3483-1000
RS449	DTE	40-pin[6], 3M-3464-0001	3M[4]-3349/37	37-pin[1], 3M-3502-1000
RS449	DCE	40-pin[6], 3M-3464-0001	3M[4]-3349/37	37-pin[1], 3M-3503-1000

Note 1: Cable housing 3M-3845-4000 may be used with the connector.

Note 2: DTE — Data Terminal mode (male connector), DCE — Data Set mode (female connector).

Note 3: Cable is tapered at one end to fit the 3M-3462 connector.

Note 4: Cable is tapered to fit 3M-3464 connector.

Note 5: Pin 26 of the edge connector is not connected to the flat cable.

Note 6: Pins 37, 39, and 40 of the edge connector are not connected to the flat cable.

Note 7: May be used with cable housing 3M-3485-1000.

Electrical Characteristics

DC Power Requirements

Mode	Voltage	Amps (Max.)
RS232C	+5V ± 0.25V	460 mA
	+12V ± 0.6V	30 mA
	−12V ± 0.6V	30 mA
RS449/422	+5V ± 0.25V	530 mA

Environmental

Temperature —	0 to 55°C
Humidity —	0% to 90%, noncondensing

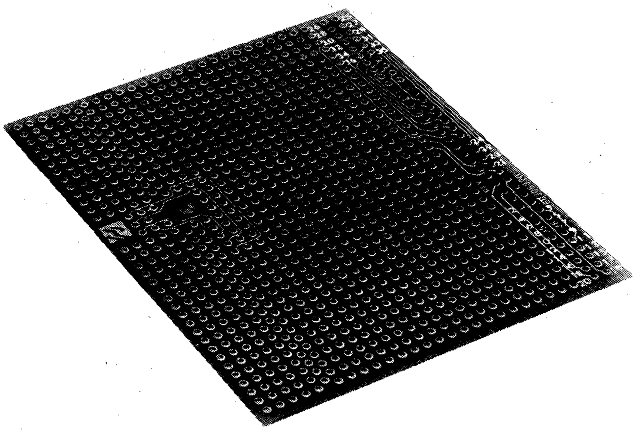
Ordering Information

BLX-351 —	Serial I/O Expansion Module

Documentation

420306333-001 —	BLX-351 Serial I/O Expansion Module User's Manual

BLX-391
Prototyping Expansion Module



- **BLX bus-compatible expansion**
- **Permits easier user construction of custom circuitry for BLX/BLC systems**
- **Capacity for up to fourteen 16-pin equivalent integrated circuits**
- **Components can be socketed or flow soldered**
- **Provisions for right-angle card edge connectors can accommodate up to 120 pins for user-defined I/O**
- **BLX bus on-board expansion eliminates Multibus™ system bus latency and increases system throughput**

Product Overview

The BLX-391 Prototyping Expansion Module is a member of the new line of BLX bus-compatible expansion module products from National Semiconductor Corporation. The BLX-391 plugs directly into any BLX bus-compatible host board offering low cost incremental on-board expansion capabilities. As a result, any BLX bus-compatible host board may be expanded to perform any function that a user can incorporate onto a single-size (2.85″ x 3.70″) board. The BLX-391 can accomodate up to fourteen 16-pin equivalent integrated circuits, which can either be soldered or socketed. Wire-wrapping is accomplished on the component side of the board with individual wire-wrap pins or with wire-wrap strips. Card edge connectors are application-dependent, so provision is made to accomodate right angle connectors up to 120 pins. The BLX expansion module is closely coupled to the host board through the BLX bus, and as such, offers maximum on-board performance and frees Multibus system traffic for other system resources. The BLX-391 comes in kit form, providing the bare printed circuit board, a male BLX connector, and nylon mounting hardware.

Functional Description

The BLX-391 Prototyping Expansion Module is a general purpose prototyping printed circuit board, of traditional design, which meets the BLX single-size (2.85″ x 3.70″) form factor. A male BLX connector is included to allow connection to a BLX bus-compatible, Series/80 host board consistent with the design criteria followed on the entire family of BLX expansion module products.

BLX Bus

The BLX bus interface provided on BLX bus-compatible Series/80 host boards conforms to those signals listed in Table 1. With the exception of several signals (such as MPST/—Module Present), all signals are merely extensions of the microprocessor bus, and, as such, meet the specifications for that microprocessor. The custom circuitry must be designed to be compatible with the actual BLX bus interface on the BLX bus-compatible Series/80 host board being used. (Any questions concerning the BLX bus should be addressed to your local Field Applications Engineer.)

Wire-Wrap

All wire-wrapping is accomplished on the component-side of the BLX-391 in order to remain within the envelope defined for the BLX expansion module family (see Figure 3). This necessitates the use of wire-wrap pins or strips. Recommended sizes and suitable vendors are provided in the specifications.

I/O Connector

Since the card edge I/O connector will differ with each application, only provision for a user-specified connector is offered. To remain within the envelope defined for the BLX expansion module family, the appropriate right angle connectors are suggested for all designs. Recommended sizes and suitable vendors are provided in the specifications.

Installation

The BLX-391 module plugs directly into either of the female BLX connectors on the host board. The module is then secured at one additional point with nylon hardware to insure the mechanical security of the assembly (see Figure 1 and Figure 2).

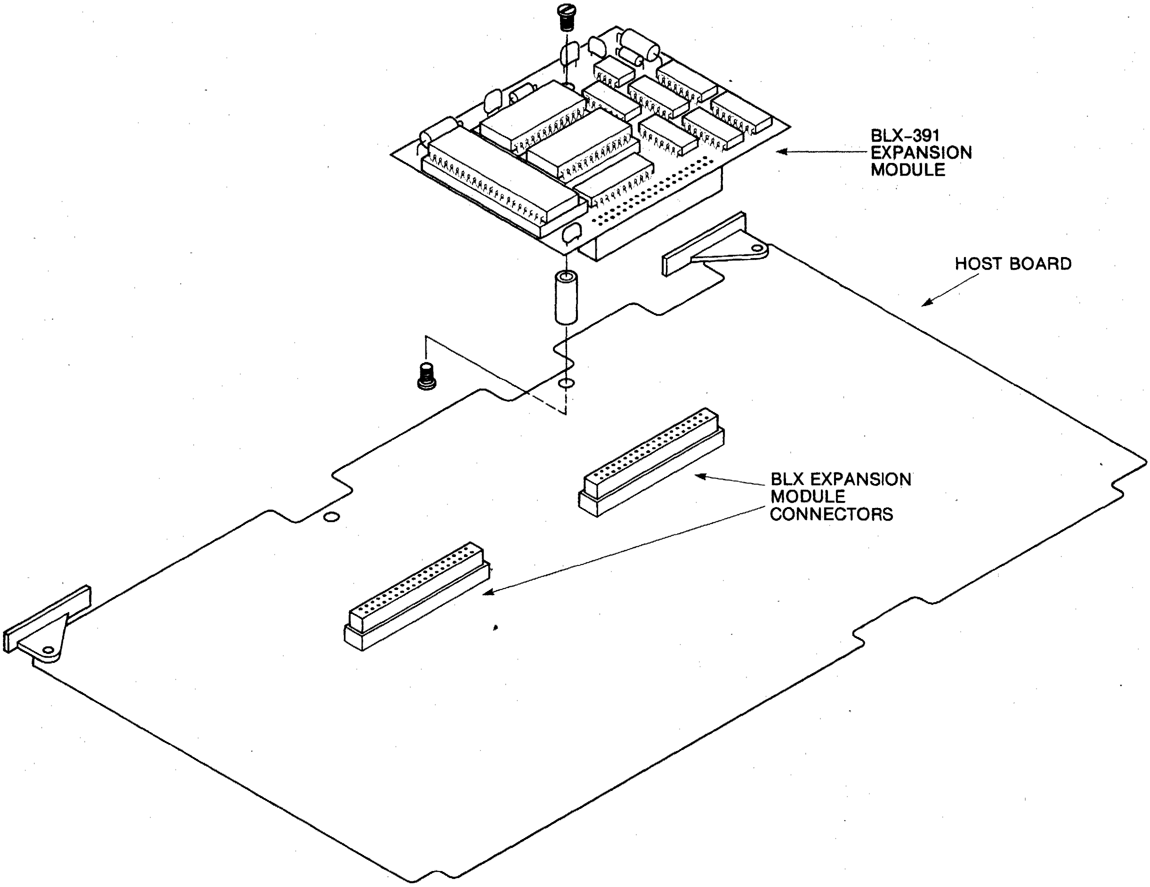


Figure 1. Installation of BLX-391 Module on a Host Board

Table I. BLX Bus Signal Pin Assignments

Pin	Mnemonic	Description	Pin	Mnemonic	Description
35	GND	Signal Ground	36	+5V	+5 Volts
33	MD0	MDATA Bit 0	34	MDRQT	M DMA Request
31	MD1	MDATA Bit 1	32	MDACK/	M DMA Acknowledge
29	MD2	MDATA Bit 2	30	OPT0	Option 0
27	MD3	MDATA Bit 3	28	OPT1	Option 1
25	MD4	MDATA Bit 4	26	TDMA	Terminate DMA
23	MD5	MDATA Bit 5	24		Reserved
21	MD6	MDATA Bit 6	22	MCS0/	M Chip Select 0
19	MD7	MDATA Bit 7	20	MCS1/	M Chip Select 1
17	GND	Signal Ground	18	+5V	+5 Volts
15	IORD/	I/O Read Command	16	MWAIT/	M Wait
13	IOWRT/	I/O Write Command	14	MINTR0	M Interrupt 0
11	MA0	M Address 0	12	MINTR1	M Interrupt 1
9	MA1	M Address 1	10		Reserved
7	MA2	M Address 2	8	MPST/	Module Present
5	RESET	Reset	6	MCLK	M Clock
3	GND	Signal Ground	4	+5V	+5 Volts
1	+12V	+12 Volts	2	−12V	−12 Volts

All undefined pins are reserved for future use.

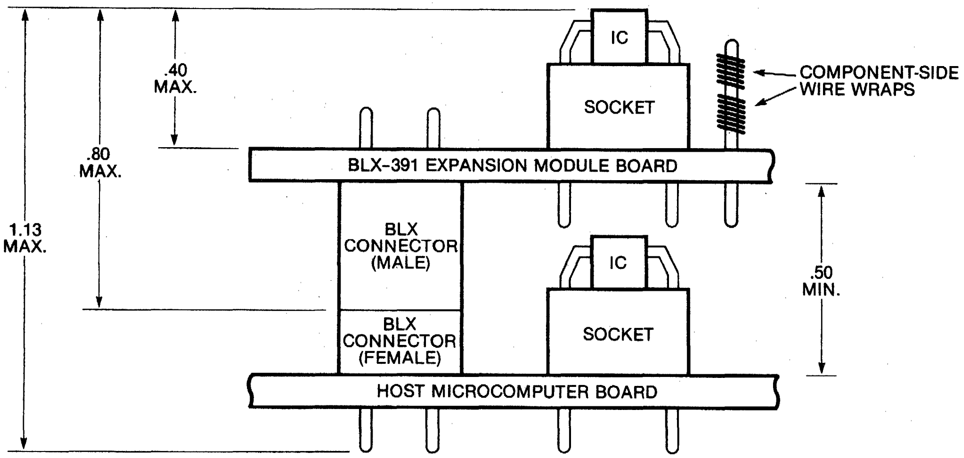


Figure 2. BLX-391 Expansion Module Mounting Clearances (inches)

Specifications

Wire-Wrap Hardware —	Sockets: Dual-in-line, low profile, on 0.10″ centers. Use the following (or equivalent):
	Wire-Wrap Strips: Strips with pins on 0.10″ centers, with height above the board not to exceed 0.40″. Use the following (or equivalent): Circuit Assembly Corp. CA-536SP100-230-430
	Wire-Wrap Pins: 0.40″ length max. Use the following (or equivalent): Berg Electronics 65474-004
Connectors —	Right-angle connector on 0.10″ centers (board side). Use the following (or equivalent): Circuit Assembly Corp.
	Plugs 26-pin: CA-D26RSP100-230-090 50-pin: CA-D50RSP100-230-090
	Sockets 26-pin: CA-26-IDS 50-pin: CA-50-IDS
Environmental	Operating Temperature: 0°C to 55°C Relative Humidity: 0% to -90%, noncondensing
Physical	Height: 2.85 in. (7.24 cm) Width: 3.70 in. (9.40 cm) Depth: BLX-391 with circuitry installed 0.80 in. (2.04 cm) BLX-391 with circuitry + host board 1.13 in. (2.86 cm) Weight: 1 oz. (28.35 gm)

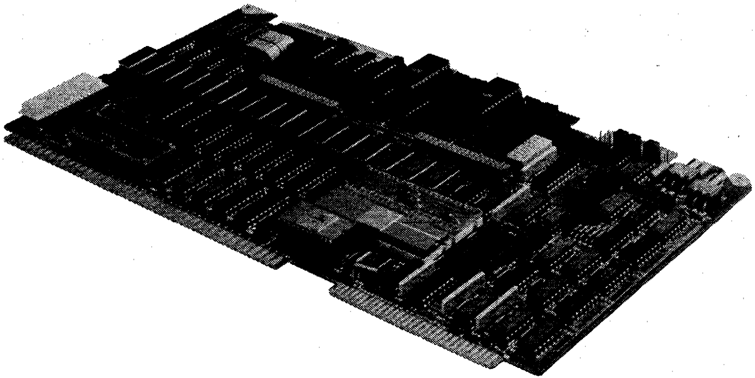
Order Information

BLX-391	Prototyping Expansion Module with BLX connector, nylon mounting hardware, and design aids.

National Semiconductor

PRELIMINARY

DB16000 Development Board



- NS16032 microprocessor
- 128 Kbytes RAM
- 8 Kbytes PROM (expandable to 16 K bytes)
- Sockets for NS16082 MMU, NS16081 FPU, NS16202 ICU
- 24 programmable parallel I/O lines
- Stand-alone development capability
- Cross software development using NSX16™
- Bus interface
- RS232 port for communication to host system and user applications
- 2 BLX™ expansion module connectors for additional I/O capability
- TDS firmware provides edit, assembly and debug capabilities in a stand-alone configuration

Product Overview

The DB16000 Development Board is a complete microcomputer system using the National Semiconductor NS16000 family of advanced microprocessors. The board includes a CPU, support chips, on-board memory, and a wide range of both standard and optional I/O interface devices. TDS development firmware, residing in on-board PROM, completes the system.

The DB16000 is specifically designed for the development and evaluation of both hardware and software for the NS16032 CPU, related slave processors and support chips. With the DB16000, users can examine the architecture, instruction set, and bus interface of the NS16032.

As a development tool, the DB16000 is a powerful hardware and software debugging aid for the NS16032 CPU. TDS firmware provides edit, assembly and debug capabilities in a stand-alone configuration. Cross-development software packages allow the DB16000 to be used with the STARPLEX II™ development system or the VAX™ series running under the VMS™ operating system. The NSX16 cross-support package allows programmers to compile or assemble NS16032 programs, then download them to the DB16000 (via a serial data link) to be executed. High-level symbolic debugging is then possible through the DBG16 source-level symbolic debugger which is part of the NSX16.

Shipped fully assembled, the DB16000 requires only suitable $+5V_{DC}$ and $\pm 12V_{DC}$ power sources and a data terminal for operation. The DB16000 provides a complete pin-out of the CPU address, data and control functions. This permits custom interfacing to a wide range of computer systems.

The standard I/O interface includes an RS232 compatible serial port and 24 parallel data lines. Also privided on the DB16000 are two connectors that allow BLX expansion modules to be used with the DB16000.

Sockets are provided for mounting optional additional members of the NS16000 family. These include the NS16081 Floating-Point Unit, the NS16082 Memory Management Unit, and the NS16202 Interrupt Control Unit. All sockets are fully integrated into the DB16000's printed wiring.

Functional Description

Central Processor

The NS16032 is the central processor for the DB16000. It features a 32-bit internal/16-bit external structure with a very powerful instruction set and programming model designed for high-level language support. Support for full virtual memory and high-speed IEEE proposed standard floating-point operations is available.

Memory

128 Kbytes of dynamic RAM are provided.

Four EPROM sockets are provided, which will accept 2716 or 2732 EPROMs. In the configuration provided, TDS-16 firmware occupies these sockets.

Input/Output

Parallel I/O

24 parallel I/O lines are provided via an 8255A Programmable Peripheral Interface. These may be divided into two 8-bit ports and two 4-bit ports. Additional I/O may be obtained by using the BLX expansion connectors.

Serial I/O

The RS232 serial link is provided via an 8251A Universal Synchronous/Asynchronous Receiver/Transmitter (USART). This link provides for communication with terminals or other computers. The data transmission clocks are generated by the ICU counter/timer if present, making the baud rate programmable. With only the ICU emulator installed, it is set at 9600 baud. Additional I/O may be obtained by using the BLX expansion connectors.

Timers

With the NS16202 Interrupt Control Unit installed there is one 16-bit timer available to the user and one dedicated to the USART. As shipped, a hardware prescaler for the USART occupies the ICU socket.

Connectors

There are six connectors available:
1 bus interface (P1)
1 local bus expansion (for custom boards) (P2)
1 RS232 port (J2)
24 Parallel I/O lines (J1)
2 BLX sockets for I/O expansion (J3, J4)

Tiny Development System (TDS)

The TDS firmware allows the user to create programs by entering source via the editor. This source is then assembled to produce executable code suitable for debugging. These functions have the following features:

Assembler:

—Subset of existing NS16000 assembler
—Supports FPU (user installed on the board) by providing long and short format real number data initialization
—Generates listings to either a printer at the parallel port, or any RS232 device connected to a port via a BLX-351 expansion module
—Symbolic definition of static base or PC segment

Debugger:

—Numerical arguments to command can be in four bases: decimal, hex, long and short real
—Program flow visually traced by displaying source line at all breakpoints or step stops
—Memory/register print or change commands
—Step-thru program commands: step "n" instructions, step while variable in range, step until variable reached

Editor:

—Command to insert, replace, delete, type lines
—Automatic line number maintenance
—Save and retrieve source from audio cassette recorder
—Debug data displayed by type command after assembly

User Program Run Time Support:

—Accessed via a supervisor call instruction
—Routines to do terminal I/O
—Printer driver access to parallel port
—Routine to convert binary value to ASCII string
—Routine to convert ASCII string to binary value
—Conversion in four bases: decimal, hex, long and short real

Modes of Operation

There are three main modes of operation; stand-alone, stand-aside and transparent. In the first mode, editor, assembler and debugging facilities are provided by resident TDS firmware, while in the other two modes an assembler, debugger, linker, librarian and Pascal compiler resident on the host system give the user a more efficient means of developing programs.

In addition, the bus interface will support single bus master applications.

Equipment Required for Use of Board in Various Modes

A cable 26-pin female edge connector to 25-pin RS232 male is required in all modes (National part numbers DB16CABLE-1 or DB16CABLE-2). This is for the serial I/O.

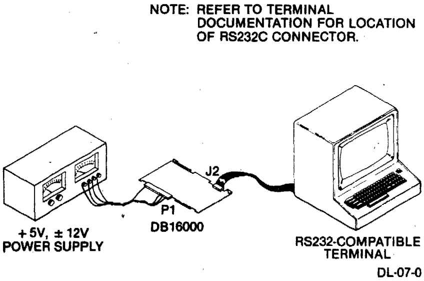
In addition, the following equipment is recommended for each mode:

1. Stand-alone
   — RS232 terminal
2. Stand-aside
   — Host computer (STARPLEX II or VAX)
   — NSX16 Cross Software Package (STARPLEX II or VAX (VMS) version)
   — RS232 terminal (only needed if not resident in host system)
3. Transparent
   — Host computer (VAX)
   — NSX16 Cross Software Package (VAX (VMS) version)
   — RS232 terminal
   — RS232-compatible serial data port expansion module (suggested: National Semiconductor model BLX-351 Serial I/O Expansion Module)
   — 2nd RS232 cable

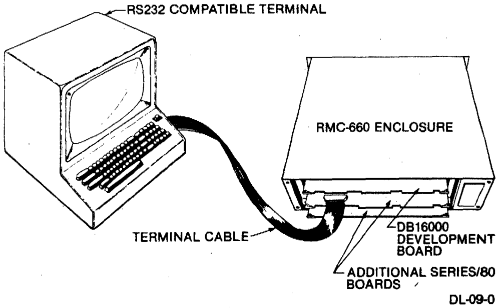
Operating Environment

Two environments are possible:

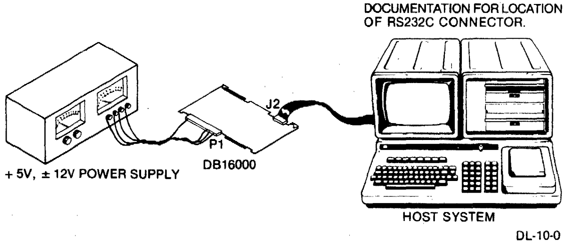
1. Bench power supply, BLC-604 chassis (only in mode 2), and BLC-957 power supply cable.
2. RMC-660 (8 slot) or RMC655 (4 slot) chassis with integral power supply.


NOTE: REFER TO TERMINAL DOCUMENTATION FOR LOCATION OF RS232C CONNECTOR.
J2
P1
DB16000
+5V, ± 12V POWER SUPPLY
RS232-COMPATIBLE TERMINAL
DL-07-0

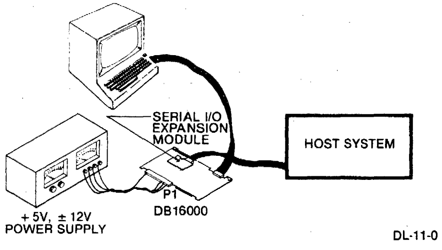
Connections for DB16000 Stand-Alone Operation


RS232 COMPATIBLE TERMINAL
RMC-660 ENCLOSURE
TERMINAL CABLE
DB16000 DEVELOPMENT BOARD
ADDITIONAL SERIES/80 BOARDS
DL-09-0

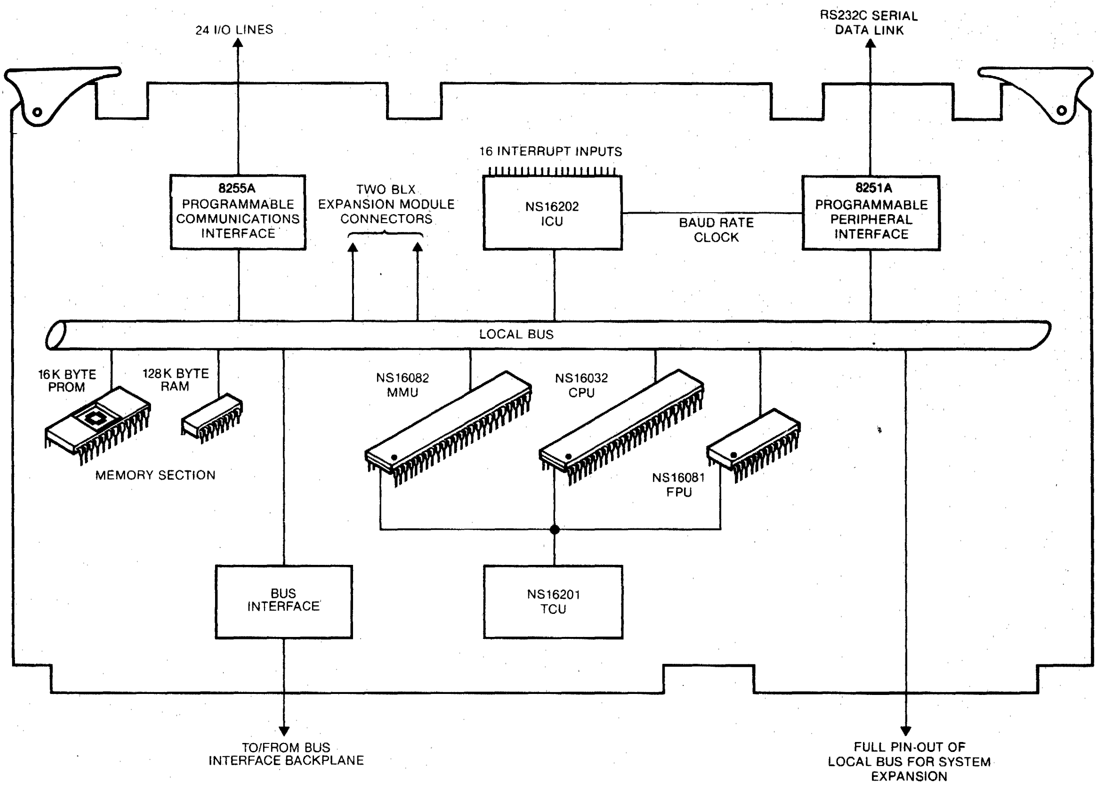
DB16000 Shown in RMC-660 Enclosure

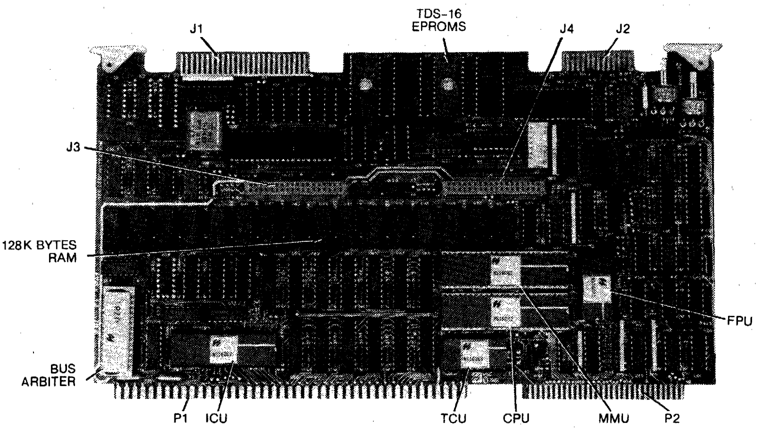

NOTE: REFER TO HOST SYSTEM DOCUMENTATION FOR LOCATION OF RS232C CONNECTOR.
J2
P1
DB16000
+5V, ± 12V POWER SUPPLY
HOST SYSTEM
DL-10-0

Connections for DB16000 Stand-Aside Mode of Operation (Host Computer Mode)



Connections for DB16000 Transparent Mode of Operation


24 I/O LINES
RS232C SERIAL DATA LINK
16 INTERRUPT INPUTS
8255A PROGRAMMABLE COMMUNICATIONS INTERFACE
TWO BLX EXPANSION MODULE CONNECTORS
NS16202 ICU
BAUD RATE CLOCK
8251A PROGRAMMABLE PERIPHERAL INTERFACE
LOCAL BUS
16K BYTE PROM
128K BYTE RAM
NS16082 MMU
NS16032 CPU
NS16081 FPU
MEMORY SECTION
BUS INTERFACE
NS16201 TCU
TO/FROM BUS INTERFACE BACKPLANE
FULL PIN-OUT OF LOCAL BUS FOR SYSTEM EXPANSION



P1 Bus Interface
P2 Local Bus Expansion
J1 Parallel I/O
J2 RS232 Port
J3 BLX Expansion
J4 BLX Expansion

Note: 4 EPROMs should be installed.

Specifications

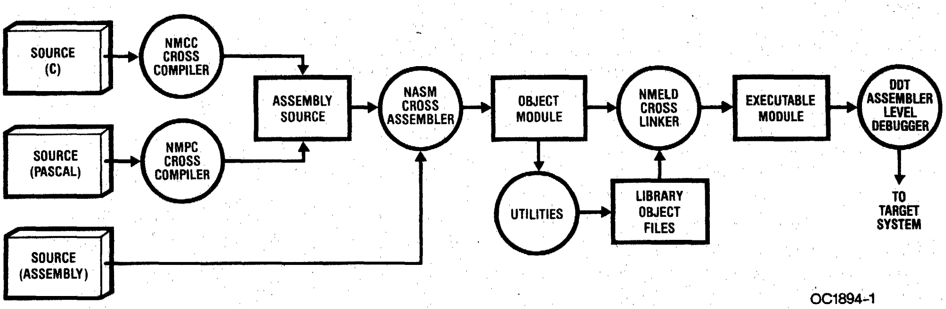
Connectors

Local bus expansion (P2) —	CDC VPB01B30A00A2 AMP PES-14559 TI H311130
Parallel I/O (J1) —	3M 3415-001 AMP 2-86792-3
Serial I/O (J2) —	3M 3462-0001 flat AMP 1-583715-1 round
Bus Interface (P1) —	SAE FUPH7212-86MTNE
Power	+5V@6A +12V@20mA −12V@20mA
Physical	Height — 6.75 in. (17.15 cm) Width — 12 in. (30.48 cm) Depth — 0.50 in. (1.27 cm) Weight — 14 oz. (396.9 g)

Ordering Information

DB16000	Development board
Documentation	
420306573-002	DB16000 Development Board Users Manual (Included with board)
420306440-001	TDS reference manual (Included with board)
420306565-001	Programmers' Reference Manual (Order separately)
420306619-002	Assembler reference manual (Order separately)

GENIX™ Cross Support Software



- **NS16000™ advanced software development package for VAX™/UNIX™ environment**
- **Derived from GENIX, an implementation of Berkeley 4.1 bsd UNIX for the NS16032**
- **C compiler**
- **Optional Pascal complier for system programming**
- **Powerful assembler supporting the full NS16000 architecture**
- **Interactive debugger with efficient command interface**
- **Runs on DEC® VAX11 with Berkeley 4.1 bsd UNIX System**

Product Overview

The GENIX cross software package supports an advanced software development environment for the NS16000 family. It is designed to run on DEC's VAX11 series with the Berkeley 4.1 bsd UNIX operating system. The language tools comprise the same package as is available on the SYS16™ development system.

Included are a C compiler, an optional Pascal compiler, NS16000 assembler, linker, libraries, utilities, and an interactive debugger. The GENIX cross support software package provides a full complement of tools to make the generation of NS16000 code an easy task. Programs thus developed can be downloaded via serial port to the DB16000 development board or the ISE/16™ In-System Emulator for execution and debugging. (ISE™ software will be available separately as part of an ISE/16 package.)

Components

nmcc—C Compiler

Compatible with the portable C compiler (pcc) of the Berkeley 4.1 bsd UNIX system. The C compiler accepts C source and generates NS16000 assembly language code.

nmpc—Pascal Compiler (OPTIONAL)

ANSI standard Pascal with modular software extensions. Accepts Pascal source and generates NS16000 assembly language code. Extensions include features such as import/export in support of full modularity. Designed to fully utilize the NS16000 architecture.

nasm—NS16000 Assembler

The assembler produces NS16000 object code in extended UNIX a.out format. It accepts complex expressions, external symbolic references, and external address arithmetic.

nmeld—Linker

Modules generated by the assembler can be linked by nmeld with the supplied or user-generated libraries to produce executable files.

Include, libc.a, libpc.a—Libraries

The libraries contain standard UNIX include files, the C library, and the Pascal library. Libraries are for code generation for licensed UNIX target systems.

nar, nnm, nranlib, nsize, nstrip—Utilities

Utilities provide the necessary tools to construct user defined libraries and to facilitate performance improvement.

ddt—Interactive Debugger

The interactive debugger allows remote debugging at the assembly language source level. It communicates with the DB16000 monitor via a serial link allowing execution and debugging on the board. Instructions may be displayed symbolically and breakpoints set by instruction, if the instruction is at an entry point or next in line from single-stepping. Breakpoints can also be set for register value match. Single-stepping is possible at the machine instruction level, or the procedure level. ddt supports debugging in physical address space, supervisor virtual address space, and user virtual address space.

cu16—Remote Communciation Utility

cu16 provides communication between the host system and the DB16000 board.

monitor—DB16000 Monitor

The monitor is compatible with ddt, and is provided in PROM firmware for the DB16000 evaluation board. This monitor is also provided in source form so that NS16000 customers can modify the monitor to suit their target system.

nburn—EPROM Programmer

nburn programs EPROMs. It is designed for use with a Datamedia I/O System 19 EPROM burner.

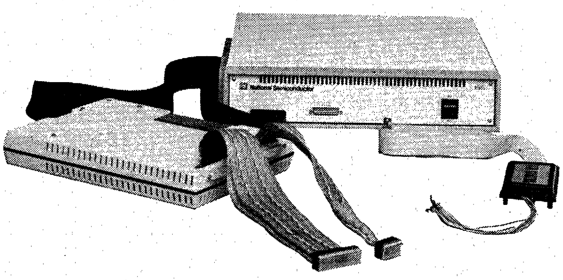
Order Information

NS-XC-16*	GENIX cross support software package, on 1600 bpi magnetic tape, tar format
NS-XC-PAS*	nmpc Pascal compiler option

*Software license agreement must be signed prior to order entry.

PRELIMINARY

NS08032 In-System Emulator (ISE/08™)



- Operation up to 6MHz
- Emulation of NS08032 Central Processing Unit, NS16201 Timing Control Unit
- Host resident high-level language and assembly language symbolic debugger
- Generalized event driven system
- Memory mapping, up to 30K bytes
- Write protection/detection of 2K byte memory blocks
- Program flow tracing, up to 255 non-sequential fetches
- Complete bus activity trace
- Qualified tracing
- Pre-, post-, or center-triggering on trace
- Count-down event counter
- Count-up execution timer/counter
- Supports Memory Management Unit functions
- Runs on VAX/11 (VMS) and STARPLEX II™ hosts
- Hierarchical help facility (on-line manual)
- Self-diagnostic

Description

The NS08032 In-System Emulator (ISE/08) is a powerful tool for both hardware and software development of NS08032microprocessor-based products.

When used with a host system such as VAX (VMS) or STARPLEX II Development Systems, ISE/08emulates a complete NS16000™ chip set. This chip set includes the 08032 Central Processing Unit (CPU), and the 16201 Timing Control Unit (TCU). ISE/08 allows users to test and debug both hardware and software in their own hardware environment. ISE/08 operates in either of two modes: emulation mode, when ISE/08 is actually running the user's program, or monitor mode, when ISE/08 is communicating with the user via the host system.

ISE/08 is a complete unit, including an internal clock oscillator and 30K bytes of dedicated user's ISE™ memory. With ISE/08, users can easily stop emulation and examine the contents of CPU registers, slave processor registers, and memory.

ISE/08 consists of the ISE hardware, the ISE monitor, a host-dependent debugger (IDBG08), an RS232 serial port cable and manual.

ISE/08 hardware is the circuitry required for emulation of a user's target system. It interfaces to the host system with an RS232-compatible serial link and provides a second RS232 port for an optional terminal connection. The ISE/08 hardware also has two target cables for connections to the target system. The target cables plug into the target system CPU, and TCU sockets.

The ISE monitor is the ISE hardware control program that monitors the host system serial data link. The ISE monitor receives monitor commands from the host system, acknowledges these commands, and generates the appropriate responses. The ISE monitor also controls the target system emulation program.

IDBG08 is the interactive debugger program for ISE/08. It runs on the host system and makes the host system facilities available to the ISE/08 user. IDBG08 automatically translates commands entered at a host system terminal to the equivalent ISE monitor commands, and communicates with the ISE monitor via the serial data link.

Hardware Description

The ISE/08 hardware is housed in three enclosures: the ISE Support Box, the Emulator Pod, and the TTL Status Pod. Figure 1 is a block diagram of ISE/08 hardware. The ISE/08 enclosures are described in the following paragraphs.

The ISE Support Box is the largest enclosure. It contains the emulation support circuits for trace, breakpoints, and mapped memory. It also contains power supplies and the hardware for the RS232 serial ports.

The Emulator Pod contains the 08032 CPU, and 16201 TCU required for target system emulation. It also contains the ISE Monitor firmware and houses the ISE/08 controls and indicators. Figure 2 shows the location of the ISE/08 controls and indicators. Table 1 lists the function of each switch and LED.

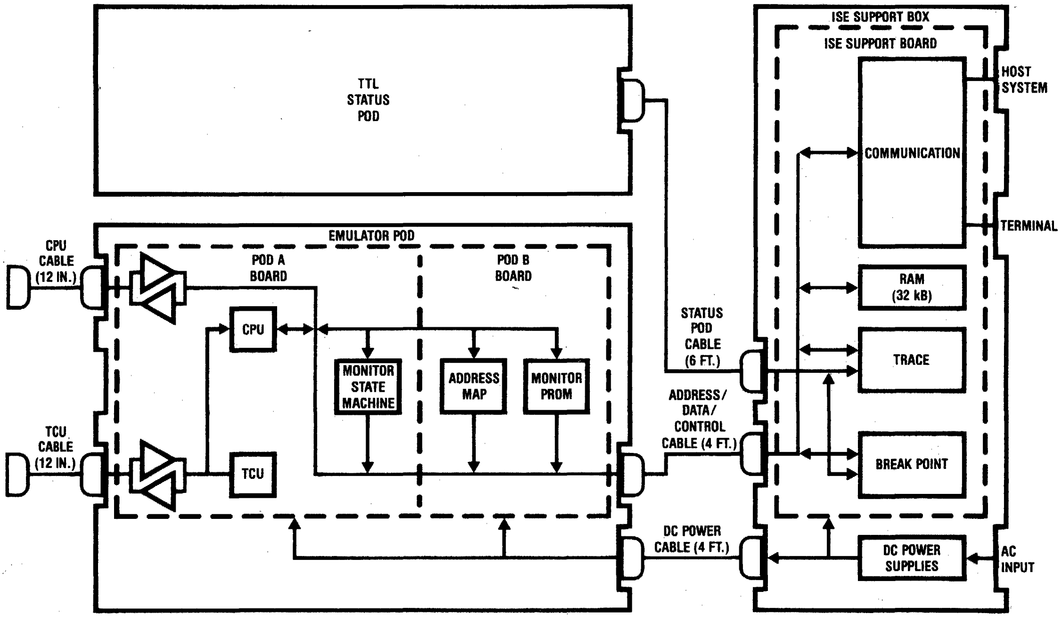
The Emulator Pod connects to the ISE Support Box through a 4-foot twisted-pair cable assembly. Connections to the target system are made with 12-inch target cables. One target cable is provided for each member of the 16000 chip set. (CP and TCU).

The Status Pod is the smallest enclosure. It provides TTL-compatible input and output signals for use during ISE operation. The Status Pod has eleven leads and seven binder posts that can be connected to either the target system or test equipment such as logic analyzers or oscilloscopes. Table 2 lists the function of each lead and post on the Status Pod. The Status Pod connects to the ISE Box front panel status connector via a 6-foot cable.

ISE/08 Software Overview

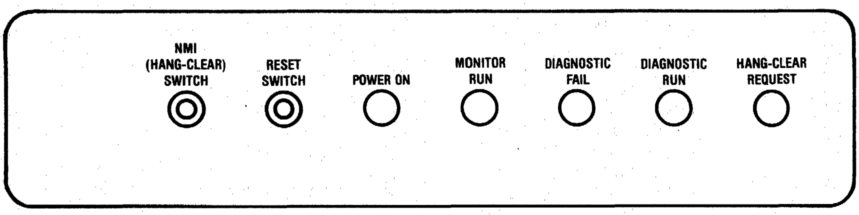
The ISE/08 software consists of two modules; the ISE monitor, residing in firmware on the Emulator Pod, and the ISE Debugger (IDBG08), residing in the host system. The monitor controls the ISE hardware. IDBG08, a high-level language debugger program, drives the ISE/08 unit. IDBG08 runs on the host computer and it communicates with the ISE/08 unit. Optionally, IDBG08 can also communicate with a terminal connected to ISE/08. The ISE/08 unit communicates with the IDBG08 program (running on the host) only while the ISE/08 unit is running the monitor program (in monitor mode), not while it is running the user's program (in emulation mode).

IDBG08 software is available for VAX/11 (VMS) and STARPLEX II hosts.



TL/R5290-1

FIGURE 1. ISE/08 Block Diagram



TL/R5290-2

FIGURE 2. ISE/08 Controls and Indicators.

Table 1. ISE/08 Control and Indicator Functions

Control/Indicator	Function
NMI Switch	When pressed, <HANG-CLEAR> occurs. <HANG-CLEAR> restores control to ISE monitor.
RESET Switch	When pressed, resets the ISE hardware.
POWER ON	Indicates power to ISE.
MONITOR RUN	Indicates ISE monitor is running.
DIAGNOSTIC RUN	Indicates ISE diagnostics are running.
DIAGNOSTIC FAIL	Indicates failure during diagnostic tests.
HANG-CLEAR REQUEST	Indicates CPU has stopped executing instructions.

Table 2. Status Pod Signal Description

Status Pod Label	ISE Function
1-WHT-USRCLK-U	IS0 (input sync 0)
2-BLK-GND	Common Ground
3-BRN-EXTO-U	EXT0 (external input 0)
4-RED-EXT1	EXT1 (external input 1)
5-ORN-EXT2	EXT2 (external input 2)
6-YEL-EXT3	EXT3 (external input 3)
7-GRN-EXT4	EXT4 (external input 4)
8-BLU-EXT5	EXT5 (external input 5)
9-VIO-EXT6	EXT6 (external input 6)
10-GRY-EXT7	EXT7 (external input 7)
11-WHT-USEBRK/U	IS1 (input sync 1)
TBRUN	Multi-Processor Sync
BK SYNCH/-U	DO (output sync)
TR SYNCH/-U	TO (trace sync)
GND	Common Ground
TSYNC31/	Not Used
TSYNC21	Not Used
GND	Common Ground

IDBG08, The ISE/08 Debugger

IDBG08 is user compatible with the standard non-ISE NS16000 Cross-Software Debugger, IDBG08. Compatibility minimizes learning time for users of the various development tools. IDBG08 fully supports all the power debugging and emulation facilities provided by the ISE/08 hardware, and supplements these features with a very powerful software-based program debugging environment.

The basic debugging features of IDBG08 are:

(1) Both high-level and assembly languages are supported.

(2) Breakpoints can be set at the source code level, even when using high-level languages.

(3) Variables can be accessed by their source code names, i.e., IDBG08 is symbolic in nature.

(4) Procedure parameters and local variables are easily displayed.

(5) Structured data types and pointers are easily displayed.

(6) Both command and history files are supported.

(7) Memory can be displayed in many different ways, including a disassembly mode displaying memory as 08032 instructions.

(8) All the emulation and debug facilities provided by the ISE/08 hardware are supported.

The ISE Monitor

When the ISE/08 unit is not running an emulation program, it is running a program called the ISE monitor. The monitor communicates with IDBG08 and it provides a command protocol that allows the host complete control of the ISE/08 hardware.

The monitor is invisible to the user, who normally communicates with the system via the friendly IDBG08 program.

Optional Terminal Feature

As an option, the ISE monitor communciates with a terminal connected to the ISE/08 unit. This terminal also serves as a terminal for the host. Thus the ISE/08 unit and the user's terminal share one RS232 port from the host.

Operation with the optional terminal is called Transparent Mode; operation without the optional terminal is called Stand-Aside Mode.

Conversion Kit for NS08032 In-System Emulation (Available December 1983)

An optional conversion kit is available for those who wish to do NS08032 development work. Contained in this kit are the following: ISE/08 Emulator Pod, ISE/08 Symbolic Debugger (IDBG08), ISE/08 Monitor Firmware, and ISE/08 Manual. Thus, because the ISE Support Box can be used for either ISE/16 or ISE/08 development work, a user wishing to do NS08032 development work but who already has an ISE/08 unit can purchase this conversion kit (in comparison to the purchase of an entire ISE/08 unit).

ISE/08 Operation

Human Interface

ISE/08 is easy to learn and easy to use. The software includes a complete on-line manual. Invoking the "HELP" command gives a summary of all ISE/08 commands, an individual command, or an individual command's parameters. This feature helps the user get his work done quickly with less frustration.

Real-Time Emulation

The ISE/08 unit has its own CPU, and TCU components. These components are connected to the target system via cables, and they perform the same functions, with close to the same timing characteristics as they would if mounted in the target system. ISE/08 does not add wait states in its operation.

Emulation memory, resident in ISE/08, can be used in lieu of target system memory. This feature is implemented by the mapping capabilities. ISE/08 can run and debug programs, without a working target system.

User target memory address space (whether it exists or not) can be mapped onto the ISE/08 emulation memory. A memory read or write operation to an address mapped onto emulation memory is performed on emulation memory only and not on the target system's memory.

Memory from the entire 24-bit physical address space of the CPU or can be mapped onto emulation memory if the following restrictions are observed:

(1) Up to four, non-contiguous segments can be defined.

(2) The address range mapped by a segment must lie within an integral 128K byte division of the address space, e.g. 00000 to h'1FFFF, or h'20000 to h'3FFFF.

(3) The address range mapped by a segment must start at the beginning of an integral 2K byte division of the address space, and end at the end of one such division e.g. h'00 to h'FFF, or h'2800 to h'37FF.

(4) The total memory space mapped by all segments must not exceed 30 K bytes.

Associated with the emulation memory mapping scheme is a capability for write protection/detection. Any 2K byte block within any of the four 128K byte segments selected can be protected. A write operation to a protected memory segment causes an IM (Illegal Map) event to occur. Write operations to protected memory are inhibited only if they occur on emulation memory. They are not inhibited if they occur in target system memory.

Related commands:

MC — Map Create
MP — Map Print

Note: For the syntax of these, and other commands listed in this section, refer to the IDBG08 Command Summary.

Generalized Events

To provide a versatile way of observing and controlling the significant state changes on the microprocessor, ISE/08 allows the use and definition of "events". In general, a simple event is a breakpoint, a bus change, or a significant observation. An event can also be a logical combination of simple events (an Event-Expression).

Simple Event Definition

The simple events are:

- Breakpoints
- Latched Breakpoints
- Counter Done
- Status Pod Inputs
- Illegal Map
- Trace Done

Breakpoint Events

ISE/08 provides three common breakpoint events, named A, B, and C. The breakpoint event can be used in two ways:

(1) Execution Breakpoint — occurs just prior to execution of an instruction fetched from a specified address.

(2) Memory Reference Breakpoint — occurs on a match when sampling:

- Address Bits
- Data Bits
- External Status Bits
- User/Supervisor Pin
- Data Direction Pin
- And where any of the above options or bits can be masked.

Either virtual or physical addresses can be sampled.

ISE/08 also provides a range breakpoint event, R. The range breakpoint occurs on any read or write operation to an address in a specified address range.

All breakpoints can cause emulation to stop immediately. Also, if used with the No Stop (/NS) option, breakpoints can be combined with other events to cause a variety of action.

Latched Breakpoint Events, Counted Events

Latched breakpoint events, named LA, LB, LC, occur at some time after a cycle where the corresponding breakpoint event (A, B, or C) has taken place. The occurrence of a latched breakpoint event remains asserted until cleared.

Events, instructions, memory cycles, and clock cycles can be counted with the breakpoint counter (up to 12 bits). Upon reaching a certain count provided by the Define Counter (DC) command, the Counter Done (CD) event takes place.

Other Simple Events

The other simple events available are:

(1) IS0, IS1 — Status Pod Input Sync 0 and Input Sync 1.

(2) IM — Write operation to write-protected address.

(3) TD — End of trace.

Related commands:

BC — Breakpoint Create
BD — Breakpoint Delete
DP— Breakpoint Print

Event-Expressions

An event-expression is a Boolean expression made up of simple events, i.e., a logical combination of simple events. This allows the user to generate many different event combinations, tailored to system activity of particular interest to the user. These generalized events are used by many ISE/08 commands such as stop, trace, event counting, etc. Event-expressions provide creative and flexible debugging procedures.

Event-expressions can be evaluated as either logically true or logically false. Valid logic operations for event-expressions are: Negation (NOT), AND, and OR.

Stopping Execution on Events

A common debugging activity is to stop emulation on the occurrence of an event of interest. Stopping emulation puts ISE/08 in the monitor mode so the user can examine and alter the state of the CPU, memory, and ISE/08 functions. Emulation can be stopped on either simple events or event-expressions.

Related commands:

DS — Define Stop
BS — Breakpoint Create

Flexible Tracing

ISE/08 maintains a 255-entry trace memory. Trace memory captures bus activity in one of two trace modes. The trace modes are:

- Program Flow Trace
- Memory Bus Trace

Program Flow Trace

The Program Flow Trace mode captures the CPU Program Counter address of 255 non-sequential instructions. This mode also maintains a count of sequential instructions executed between each non-sequential instruction stored in the trace memory.

Memory Bus Trace

The Memory Bus Trace mode captures a summary of system parameters during 255 memory bus access cycles. The following parameters are captured:

- Address bus contents
- Data bus contents
- CPU Status (data transfer, non-sequential fetch, interrupt acknowledge, etc.)
- Status Pod External Inputs
- States of the Following CPU Pins:
  PFSC — Program Flow Status (start of instruction)
  UNS — User/Not Supervisor
  NDDIN — Not Data In

A tracing event can qualify the memory bus tracing mode. This event allows the user to reduce the number of events captured.

When enabled, tracing in either mode continues until a specified terminating event occurs. The actual end of tracing can be delayed after the terminating event by a count of 1 to 255. This allows trace data to be captured before, after, or around the terminating event.

Execution Timer

The execution timer is a 24-bit counter with an overflow flag that may be used to count events, instruction cycles, memory cycles, or clock cycles. The timer may be programmed to start and stop counting on specific events. This permits using the execution timer to determine relative timing differences between various events. One use of this feature is to measure software or hardware performance.

Related Commands:

DE — Define Execution Timer
LD — List Definitions

Event Trigger for External Test Equipment

ISE/08 events can trigger external test equipment, such as oscilloscopes and logic analyzers. This test equipment can be used in conjunction with ISE/08 debugging features to solve system timing problems. Two external trigger sources are provided:

- General Event (or Event-Expression)
- Trace Trigger Event (i.e., an event that causes an entry into trace memory.)

The external trigger signals are available at two status pod outputs:

- BKSYNCH/-U (General Event)
- TRSYNCH/-U (Trace Trigger Event)

Related Commands:

DO — Define Output Sync Command

ISE/08 Timing Options

ISE/08 includes the following timing options:

- Sampling time can be set to sample either virtual or physical addresses
- Status Pod external lines can be sampled at either data valid or address valid times
- The emulation clock frequency can be set to one of the following frequencies:
  1.5 MHz
  3.0 MHz
  6.0 MHz
  Target Board Frequency

Note: Selection of target board frequency may require synchronous and/or asynchronous delay compensation. Refer to ISE/08 User's Manual, Chapter 6, for details.

Related Commands:

SO — Select Options

Self-Test Diagnostics

At power-up or reset, ISE/08 runs a diagnostic program to verify ISE software integrity and proper hardware function.

Required User-Supplied Equipment

For use under VAX/11 systems:

- Valid DEC VAX/11 configuration, with available RS232 port.
- VMS Operating System, Version 3.0 or later.
- NSX-08 Cross Software Package, or NS-ASM-08 NS16000 Cross Assembler Package.

For use with STARPLEX II systems:

- STARPLEX II Development System.
- STARPLEX II Operating System, Version G or later.
- SFW-90-A010 NS16000 Cross-Assembler Package.

For use with a system that has a Berkeley 4.1 based UNIX™ Operating System:
[Contact Marketing for Availability Information.]

- Valid computer system with an available RS232 port.
- Appropriate cross software package.
  [Contact Marketing for further information.]

Specifications

Environmental	Operating Temperature +10°C to +40°C Storage Temperature −20°C to +65°C
Power	3A @ 115 V_{AC}, 50/60 Hz, single phase 1.5A @ 220 V_{AC}, 50/60 Hz, single phase Approximately 1170 BTU.

Physical

ISE Support Box —	Height: 4.125 in. (10.5 cm) Width: 19.0 in. (48.3 cm) Depth: 17.5 in. (44.5 cm)
Emulation Pod —	Height: 2.25 in. (6.4 cm) Width: 9.25 in. (23.5 cm) Depth: 14.0 in. (35.6 cm)
TTL Status Pod —	Height: 1.0 in. (2.5 cm) Width: 3.125 in. (7.9 cm) Depth: 6.125 in. (15.6 cm)
Cable Lengths —	ISE Support Box to Emulation Pod: 4.0 ft. (1.22M) ISE Support Box to TTL Status Pod: 6.0 ft. (1.83M) Emulation Pod to Target Board: 1.0 ft. (0.30M)

Electrical

Operating Frequency —	User selectable to one of the following: 1.5MHz 3.0MHz 6.0MHz Target Board Frequency **Note:** Selection of target board frequency may require synchronous and/or asynchronous delay compensation. Refer to ISE/08 User's Manual, Chapter 6, for details.
Target Interface Electrical Characteristics —	See Tables 3 through 5.

Order Information

Complete ISE/08 Units

NS-ISE-08	ISE/08 (NS08032), 115V$_{AC}$ for VAX/11 (VMS) Computer System.
SPM-90-A1608	ISE/16 (NS08032), 115V$_{AC}$ for STARPLEX II Development Systems.
NS-SYS-2008	ISE/16 (NS08032), 115V$_{AC}$ for UNIX OS based operating systems. [Contact Marketing for Availability Information.]

Conversion Kits to Allow for ISE/16 Emulation

[Contain ISE/16 Emulator Pod, ISE Debugger (IDBG16), appropriate ISE/16 monitor firmware, and ISE/16 manual.]

AEE-90-A1632	ISE/08 to ISE/16 kit for STARPLEX II use.
AEE-ISE-16	ISE/08 to ISE/16 kit for VAX/11 (VMS) use.
AEE-ISENIX-16	ISE/08 to ISE/16 kit for UNIX based OS systems use. [Contact Marketing for Availability Information.]

Documentation

TBD	ISE/08 User's Manual (Included with NS-ISE-08, and SPM-90-A1608.)

Documentation Conventions

The following documentation conventions are used in describing the IDBG08 commands and parameters. Upper-case and lower-case letters are used in these conventions; any combination of upper-case and lower-case letters may actually be used when entering commands.

UPPER-CASE letters show the command letters, parameters and options. The names must be entered exactly as shown.

Spaces and blanks have been added for readability. When actually entering commands, spaces and blanks may only appear between the command and its parameters and between the parameters and the local radix.

< > — angle brackets enclose descriptive names (in lower-case) for user-supplied parameters/options.

{ } — braces enclose more than one item out of which one, and only one, must be used. The items are separated from each other by a logical OR sign "|".

[ ] — brackets enclose optional item(s).

| — logical OR sign separates items out of which one, and only one, may be used.

... — three consecutive periods indicate optional repetition of the preceding item.

Table 3. Electrical Characteristics for TCU Interface

Signal Name	Interface Device	Input And/Or Output Current		Propagation Delay Time T_{pd}*
		I_{OH}	I_{OL}	
OUTGOING SIGNALS:				
NTSO	74S244	15mA	64mA	14.6ns
CTTL	74S244	15mA	64mA	14.6ns
FCLK	74S244	15mA	64mA	14.6ns
NDBE	74S244	15mA	64mA	14.6ns
NRD	74S244	15mA	64mA	14.6ns
NWR	74S244	15mA	64mA	14.6ns
NRST	74S244	15mA	64mA	14.6ns
RDY	74S244	15mA	64mA	14.6ns
		I_{IH}	I_{IL}	
INCOMING SIGNALS:				
NPER	74S244	50μA	400μA	14.6ns
NCWAIT	74S244	50μA	400μA	14.6ns
NWAIT1	74S244	50μA	400μA	14.6ns
NWAIT2	74S244	50μA	400μA	14.6ns
NWAIT3	74S244	50μA	400μA	14.6ns
NWAIT4	74S244	50μA	400μA	14.6ns
XCTL1	74S244	50μA	400μA	14.6ns
NCEN	74S244	50μA	400μA	14.6ns
NRST1	74S244	50μA	400μA	14.6ns

*Interface device, plus cable.

Table 4. Electrical Characteristics for CPU Interface

Signal Name	Interface Device	Input And/Or Output Current				Propagation Delay TimeT_{pd}*
		I_{OH}	I_{OL}	I_{IH}	I_{IL}	
BIDIRECTIONAL SIGNALS:						
NSPC	none	—	—	—	—	1.4ns
A15	74S244	15mA	64mA	—	—	14.6ns
A14	74S244	15mA	64mA	—	—	14.6ns
A13	74S244	15mA	64mA	—	—	14.6ns
A12	74S244	15mA	64mA	—	—	14.6ns
A11	74S244	15mA	64mA	—	—	14.6ns
A10	74S244	15mA	64mA	—	—	14.6ns
A09	74S244	15mA	64mA	—	—	14.6ns
A08	74S244	15mA	64mA	—	—	14.6ns
AD07	74S244	15mA	64mA	—	—	14.6ns
AD06	74S244	15mA	64mA	—	—	14.6ns
AD05	74S244	15mA	64mA	—	—	14.6ns
AD04	74S244	15mA	64mA	—	—	14.6ns
AD03	74S244	15mA	64mA	—	—	14.6ns
AD02	74S244	15mA	64mA	—	—	14.6ns
AD01	74S244	15mA	64mA	—	—	14.6ns
AD00	74S244	15mA	64mA	—	—	14.6ns
OUTGOING SIGNALS:						
A23	74S244	15mA	64mA	—	—	14.6ns
NILO	74S244	15mA	64mA	—	—	14.6ns
ST0	74S244	15mA	64mA	—	—	14.6ns
ST1	74S244	15mA	64mA	—	—	14.6ns
ST2	74S244	15mA	64mA	—	—	14.6ns
ST3	74S244	15mA	64mA	—	—	14.6ns
NPFS	74S244	15mA	64mA	—	—	14.6ns
NDDIN	74S244	15mA	64mA	—	—	14.6ns
NADS	74S244	15mA	64mA	—	—	14.6ns
UNS	74S244	15mA	64mA	—	—	14.6ns
HHLDA	74S244	15mA	64mA	—	—	14.6ns
A22	74S244	15mA	64mA	—	—	14.6ns
A21	74S244	15mA	64mA	—	—	14.6ns
A20	74S244	15mA	64mA	—	—	14.6ns
A19	74S244	15mA	64mA	—	—	14.6ns
A18	74S244	15mA	64mA	—	—	14.6ns
A17	74S244	15mA	64mA	—	—	14.6ns
A16	74S244	15mA	64mA	—	—	14.6ns
INCOMING SIGNALS:						
TSYSPWR	1N4002	—	—	—	—	—
NINT	74S244	—	—	50μA	400μA	14.6ns
NNMI	74S244	—	—	50μA	400μA	14.6ns
NHOLD	74S244	—	—	50μA	400μA	14.6ns

*Interface device, plus cable.

IDBG08 Command Summary

The following is a comprehensive list of the IDBG08 commands. Commands are in alphabetical order. See the ISE/08 User's Manual for a detailed description of each command.

<table>
<tr><th>Command</th><th>Syntax</th><th>Function</th></tr>
<tr><td>Begin</td><td>B [<file>] [/NL] [/NI] [/R] [/Z]</td><td>(if no switches) Loads the program <file> into target board memory, and initializes registers.
/NL — No Load
/NI — No Initialize
/R — Reset
/Z — Zero-Fill data areas</td></tr>
<tr><td>Breakpoint Create</td><td>BC [A,|B,|C,] <address> [/NS]</td><td>Creates execution breakpoint A,B, or C at specified <address>.
/NS — No Stop</td></tr>
<tr><td></td><td>BC [A,|B,|C,] {<address> | <mask>} <breakpoint-options> [/NS]</td><td>Creates memory reference breakpoint A, B, or C at address specified by <address> or <mask> and with specified <breakpoint-options>.
/NS — No Stop</td></tr>
<tr><td></td><td>BC R, <address-range> [/NS]</td><td>Creates range breakpoint R at specified <address-range>.
/NS — No Stop</td></tr>
<tr><td>Breakpoint Delete</td><td>BD [A|B|C|R]</td><td>Deletes specified breakpoints.</td></tr>
<tr><td>Breakpoint Revive</td><td>BR [A|B|C|R]</td><td>Revives specified breakpoint.</td></tr>
<tr><td>Breakpoint Print</td><td>BP [A|B|C|R]</td><td>Prints address and conditions of specified breakpoints.</td></tr>
<tr><td>Command File</td><td>@ {<file> | <n>}</td><td>Executes command <file> or debugger string sequence beginning at <n>.</td></tr>
<tr><td>Debugger String</td><td>$ <n> = [<string>]</td><td>Sets debugger string <n> to <string>.</td></tr>
<tr><td>Define Counter</td><td>DC <n> [/B =<event-expression>]
[/C ={<event-expression> |I|M|C}]
DC/R</td><td>Defines set up for ISE counter.
<n> — Number of counts
/B — Begin event
/C — Counter type
/R — Reset</td></tr>
<tr><td>Define Execution-Timer</td><td>DE [/B =<event-expression>] [/E =< event-expression>]
[/C ={<event-expression> |I|M|C}]
DE/R</td><td>Defines set up for ISE execution timer.
/B — Begin event
/E — End event
/C — Count type
/R — Reset</td></tr>
<tr><td>Define Output Sync</td><td>DO <event-expression></td><td>Defines output sync event.</td></tr>
<tr><td>Define Stop</td><td>DS <event-expression></td><td>Defines stop event.</td></tr>
</table>

IDBG08 Command Summary (Continued)

Command	Syntax	Function
Define Trace	DT [/E=<event-expression> [/D=<n>]] [/P | /M [= {<address> | <mask>}] <qualify-options>]]]	Defines the end, delay, and trace mode parameters for trace. /E — End event /D — Delay count /P — Program Flow mode /M — Memory Bus mode /R — Reset
Disassemble	D <address-range> [/I=<n>] [/NA]	Disassemble instructions in <address-range>. /NA — No Address /I — Number of Instructions to be disassembled.
Go	G [/F=<address>] [[/T=] <breakpoint>]	Starts execution of the program at the current PC address of from the <address>. Execution continues until <breakpoint>.
Help	H [<string>]	Displays general help or command syntax or parameter syntax.
In	I {<address> | <register>} [<radix>] <value1> [<value2>]	Checks that contents of <address> or <register> are in the range specified by <value1> and <value2 >, inclusively.
List Calls	LC [<n>]	Lists first <n> entries in call chain.
List Definitions	LD [/T|/E|/C|/O|/S]	Lists current definitions. /T — Trace definition /E — Execution timer definition /C — Counter definition /O — Output sync definition /S — Stop definition
List Files	LF [<line> [/<file>]]	Lists lines in <file>.
List Information	LI	Lists current IDBG16 status.
List Modules	LM	Lists modules in current program.
List Procedures	LP	Lists procedures in current program.
List Strings	LS	Lists current debugger string values.
List Trace	LT [<n> |*|/A|/J]	Lists nine trace entries centered around entry <n> or around the trigger point (*). /A — All entries /J — Jumps only
Map Create	MC [<address-range>] [/S={A|B|C|D}] [/M=<n> |/NM] [/P|/NP]	Creates segment assignment and mapping for special <address-range>. /S — Segment assignment /M — Mapping to ISE block <n> /NM — No Mapping /P — Write Protection /NP — No Protection

IDBG08 Command Summary (Continued)

Command	Syntax	Function
Map Print	MP [/S = {A|B|C|D}]	Prints current mapping for specified segment.
Memory Fill	MF <address-range> [<radix>] <value>	Fills memory at <address-range> with <value>.
Memory Move	MM <address-range>,<address> [<radix>]	Moves memory from <address-range> to <address>.
Memory Search	MS <address-range> [<radix>] <value>	Searches for <value> in <address-range>.
On	O {FAIL|RESET|NMI|EXIT [({A|B|C|R|S|HC})] {@ <n> |/O}	Sets IDBG16 response on condition: @ <n> — Executes debugger string sequence on condition beginning with $ <n>, /O — Response Off, restores normal response.
Print	P [<address-range> |<register-range>] [<radix>]	Prints contents of <address-range> or <registers>.
Print Address	PA <address>	Prints absolute address and module area associated with <address>.
Protection Create	PC <address-range> [/UW|/UR|/SW|SR] [/V|/NV] [/R|/NR] [/M|/NM] [/T = <gn>] [/P]	Creates protection/translation for pages specified by <address-range>.
Potection Print	PP [<address-range>]	Prints protection level status for pages specified by <address-range>.
Quit	Q [/S]	Terminates session. /S — Save IDBG16 status in IDBG16.IND
Repeat	<cr>	Repeats previous command.
Replace	R {<address> | <register>} [/NV] [<radix>] [<value>]	Replaces contents of <address> or <register> with <value>. /NV — No Verify
Select Echo	SE [/O]	Selects echo mode. /O — Echo Off.
Select Full	SF [/O]	Selects full symbolic PC. /O — Full Off.
Select History	SH {<file> [/F] |/O}	Selects history file <file>: /F — Full history (with responses) /O — History Off
Select Link	SL <file> [,<file>] [/L]	Selects communications channel(s): /L — List communications
Select Module	SM <module>	Selects module.

IDBG08 Command Summary (Continued)

Command	Syntax	Function
Select Options	SO [/AS={NADS|NPAV} [/XS={D|A}] [/EC={10|5|2.5|U}] [/MC={10|EC}] [/L={C|NC}] [/T={0|1|N|A}]	Selects current ISE operation options. /AS — Address Sample time /XS — External Sample time /EC — Emulator Clock frequency /MC — Monitor Clock frequency /L — Latched Clear or No-Clear /T — Translation
Select Procedure	SP [<n> | <procedure>:]	Selects procedure.
Select Radix	SR <radix>	Select global radix.
Step	S [<gn>]	Executes <gn> machine instructions (Assembly programs) or one Pascal statement (Pascal programs). <gn> illegal in Pascal.
Step Call	SC	Executes until a call or return.
Step Down	SD	Executes one instruction inside a procedure; skips over call instructions.
Step Instruction	SI [<gn>]	Executes <gn> machine instructions.
Step Until	SU {<address> | <register>} [<radix>] <value1> [<value2>]	Executes instructions until contents of <address> or <register> within the range specified by <value1> and <value2>.
Step While	SW {<address> | <register>} [<radix>] [<value1>] [<value2>]	Executes instructions while contents of <address> or <register> are within the range specified by <value1> and <value2>.

Parameter Summary

The following is a comprehensive list of command parameters. See the ISE/16 User's Manual for a detailed description of each.

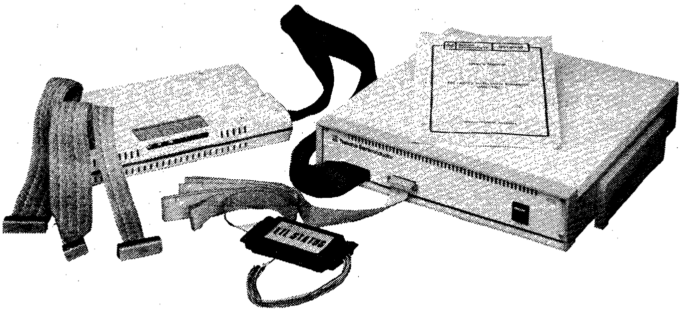
Command	Syntax	Function
Number (<n>)	<digits>	An unsigned, decimal number in the range 0 to 32767.
General Number (gn)	[H'|Q'|O'|D'] [–] <digits>	A signed, octal, decimal, or hexadecimal number in the range of -2^{31} to $2^{31}-1$.
Mask	M' {0|1|X|_}...	An unsigned number which consists of binary digits, "don't care" bits, and optional underscores. A mask represents the value of address, data, status, or external bits.
Name	<letters, numbers, underscores, tildes>	A combination of letters, digits, underscores, and tildes which does not start with a digit.
Module	<name>	The name of a module in the program.
Procedure	<name> [# <n>]	The name of a procedure in the selected module. # <n> specifies the <n>th procedure having <name> in the selected module.
Symbol	<name>	The name of a variable in the selected module or procedure.
Register	<register-name> [% <n>]	One of the registers shown in Table 3. % <n> specifies the field starting at the <n>th bit in <register>.
Register-Range	{•CPU__ | •MMU__ | •FPU__ | •PSR__ | •MSR__ | •FSR__ | <register>}	Specifies all CPU, MMU, or FPU registers or all PSR, MSR, and FSR fields.

Parameter Summary (Continued)

Command	Syntax	Function
Address	<basic-address> [+<abs> | –<abs> | % <abs> | ∧ | • <field> | <indexing>]...	A byte or bit address. The address consists of a basic address and any combination of optional operators. +<abs> — Adds <abs> to address. –<abs> — Subtracts <abs> from address. %<abs> — Takes <abs>th bit at address. ∧ — Takes contents as address. • <field> — Address is address of a field in a Pascal record. <indexing> — Address is address of an array element.
Address-Range	{<address1>..<address2> | < address> ! <n> | <address>}	The range of addresses from <address1> to <address2>, from <address> to <address> + (<n> – 1)*(<current radix>), or from <address> to itself.
Radix	[%] <n> <base>	Specifies the length and type of input/output in IDBG16 commands. [%] specifies length. If % is specified, length is in bits. <n> must be within the range 1 to 256. <base> specifies type and may be binary (B), decimal (D), octal (O), hexadecimal (H), hexadecimal dump (H), floating-point (F), logical (L), ASCII (A), Pascal set (S), or Pascal string (G).
Value		A value to be entered or displayed after issuing a Print, Replace, Step, Memory, or In command. Syntax is defined by current radix.
File		The name of any file in the host system. File syntax is host dependent.
Event	{IS0|IS1|IM|TD|CD|A|B|C|LA|LB|LC|R}	The name of an ISE response to a specific set of run-time conditions.
Event-Expression	(['] <event> [*['] <event>...[+ ['] <event> [*<event>]...]...]) (1) (0)	A Boolean expression consisting of one or more <events>s and the logical NOT ('), AND (*), and OR (+) operators. (1) — always true. (2) — always false.

National Semiconductor

NS16032 In-System Emulator (ISE/16™)



- **Operation up to 6MHz**
- **Emulation of NS16032 Central Processing Unit, NS16082 Memory Management Unit, NS16201 Timing Control Unit**
- **Host resident high-level language and assembly language symbolic debugger**
- **Generalized event driven system**
- **Memory mapping, up to 30K bytes**
- **Write protection/detection of 2K byte memory blocks**
- **Program flow tracing, up to 255 non-sequential fetches**
- **Complete bus activity trace**
- **Qualified tracing**
- **Pre-, post-, or center-triggering on trace**
- **Count-down event counter**
- **Count-up execution timer/counter**
- **Supports Memory Management Unit functions**
- **Runs on VAX/11 (VMS) and STARPLEX II™ hosts**
- **Hierarchical help facility (on-line manual)**
- **Self-diagnostic**

Description

The NS16032 In-System Emulator (ISE/16) is a powerful tool for both hardware and software development of NS16032 microprocessor-based products.

When used with a host system such as VAX (VMS) or STARPLEX II Development Systems, ISE/16 emulates a complete NS16000™ chip set. This chip set includes the 16032 Central Processing Unit (CPU), the 16082 Memory Management Unit (MMU), and the 16201 Timing Control Unit (TCU). ISE/16 allows users to test and debug both hardware and software in their own hardware environment. ISE/16 operates in either of two modes: emulation mode, when ISE/16 is actually running the user's program, or monitor mode, when ISE/16 is communicating with the user via the host system.

ISE/16 is a complete unit, including an internal clock oscillator and 30K bytes of dedicated user's ISE™ memory. With ISE/16, users can easily stop emulation and examine the contents of CPU registers, slave processor registers, and memory.

ISE/16 consists of the ISE hardware, the ISE monitor, a host-dependent debugger (IDBG16), an RS232 serial port cable and manual.

ISE/16 hardware is the circuitry required for emulation of a user's target system. It interfaces to the host system with an RS232-compatible serial link and provides a second RS232 port for an optional terminal connection. The ISE/16 hardware also has three target cables for connections to the target system. The target cables plug into the target system CPU, MMU, and TCU sockets.

The ISE monitor is the ISE hardware control program that monitors the host system serial data link. The ISE monitor receives monitor commands from the host system, acknowledges these commands, and generates the appropriate responses. The ISE monitor also controls the target system emulation program.

IDBG16 is the interactive debugger program for ISE/16. It runs on the host system and makes the host system facilities available to the ISE/16 user. IDBG16 automatically translates commands entered at a host system terminal to the equivalent ISE monitor commands, and communicates with the ISE monitor via the serial data link.

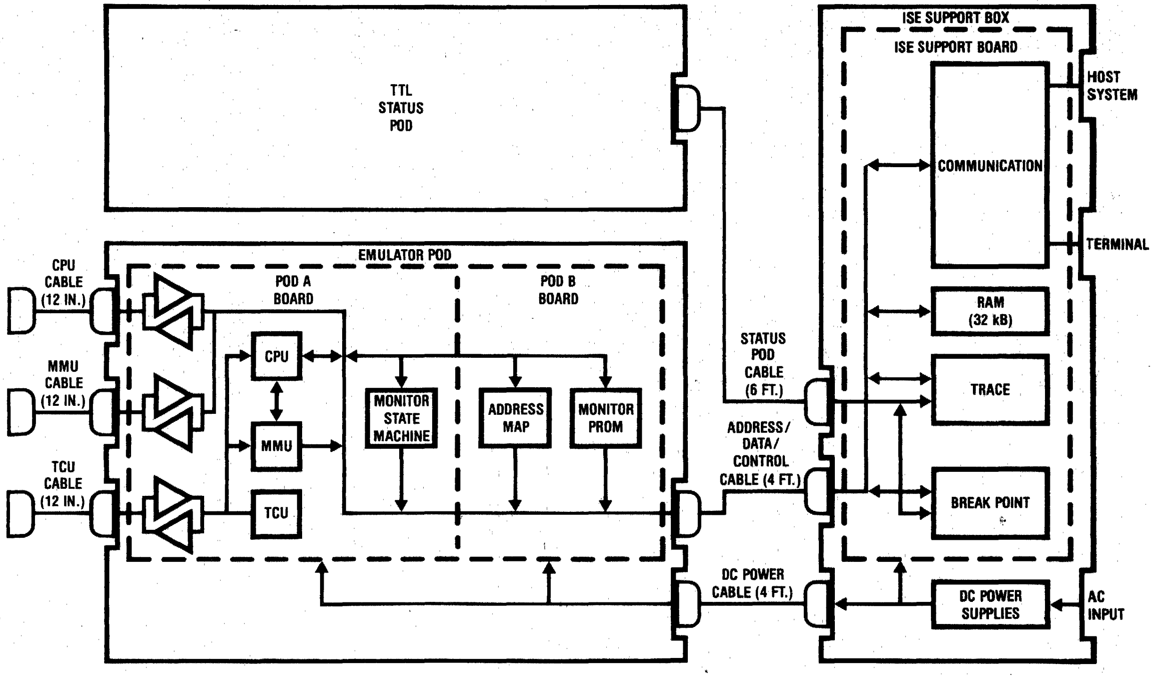
Hardware Description

The ISE/16 hardware is housed in three enclosures: the ISE Support Box, the Emulator Pod, and the TTL Status Pod. Figure 1 is a block diagram of ISE/16 hardware. The ISE/16 enclosures are described in the following paragraphs.

The ISE Support Box is the largest enclosure. It contains the emulation support circuits for trace, breakpoints, and mapped memory. It also contains power supplies and the hardware for the RS232 serial ports.

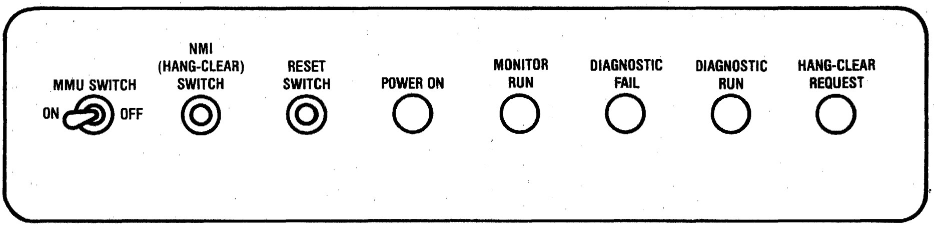
The Emulator Pod contains the 16032 CPU, 16082 MMU, and 16201 TCU required for target system emulation. It also contains the ISE Monitor firmware and houses the ISE/16 controls and indicators. Figure 2 shows the location of the ISE/16 controls and indicators. Table 1 lists the function of each switch and LED.

The Emulator Pod connects to the ISE Support Box through a 4-foot twisted-pair cable assembly. Connec-



TL/R5127-1

FIGURE 1. 'ISE/16 Block Diagram



TL/R5127-2

FIGURE 2. ISE/16 Controls and Indicators

tions to the target system are made with 12-inch target cables. One target cable is provided for each member of the 16000 chip set. (CPU, MMU, and TCU).

The Status Pod is the smallest enclosure. It provides TTL-compatible input and output signals for use during ISE operation. The Status Pod has eleven leads and seven binder posts that can be connected to either the target system or test equipment such as logic analyzers or oscilloscopes. Table 2 lists the function of each lead and post on the Status Pod. The Status Pod connects to the ISE Box front panel status connector via a 6-foot cable.

ISE/16 Software Overview

The ISE/16 software consists of two modules; the ISE monitor, residing in firmware on the Emulator Pod, and the ISE Debugger (IDBG16), residing in the host system. The monitor controls the ISE hardware. IDBG16, a high-level language debugger program, drives the ISE/16 unit. IDBG16 runs on the host computer and it communicates with the ISE/16 unit. Optionally, IDBG16 can also communicate with a terminal connected to ISE/16. The ISE/16 unit communicates with the IDBG16 program (running on the host) only while the ISE/16 unit is running the monitor program (in monitor mode), not while it is running the user's program (in emulation mode).

IDBG16 software is available for VAX/11 (VMS) and STARPLEX II hosts.

IDBG16, The ISE/16 Debugger

IDBG16 is user compatible with the standard non-ISE NS16000 Cross-Software Debugger, DBG16. Compatibility minimizes learning time for users of the various development tools. IDBG16 fully supports all the power debugging and emulation facilities provided by

Table 1. ISE/16 Control and Indicator Functions

Control/Indicator	Function
MMU Switch	When on, it enables MMU operation (Mbit in CPU Configuration Register set to 1). When off, disables MMU (Mbit set to 0).
NMI Switch	When pressed, <HANG-CLEAR> occurs. <HANG-CLEAR> restores control to ISE monitor.
RESET Switch	When pressed, resets the ISE hardware.
POWER ON	Indicates power to ISE.
MONITOR RUN	Indicates ISE monitor is running.
DIAGNOSTIC RUN	Indicates ISE diagnostics are running.
DIAGNOSTIC FAIL	Indicates failure during diagnostic tests.
HANG-CLEAR REQUEST	Indicates CPU has stopped executing instructions.

Table 2. Status Pod Signal Description

Status Pod Label	ISE Function
1-WHT-USRCLK-U	IS0 (input sync 0)
2-BLK-GND	Common Ground
3-BRN-EXTO-U	EXT0 (external input 0)
4-RED-EXT1	EXT1 (external input 1)
5-ORN-EXT2	EXT2 (external input 2)
6-YEL-EXT3	EXT3 (external input 3)
7-GRN-EXT4	EXT4 (external input 4)
8-BLU-EXT5	EXT5 (external input 5)
9-VIO-EXT6	EXT6 (external input 6)
10-GRY-EXT7	EXT7 (external input 7)
11-WHT-USEBRK/U	IS1 (input sync 1)
TBRUN	Multi-Processor Sync
BK SYNCH/-U	DO (output sync)
TR SYNCH/-U	TO (trace sync)
GND	Common Ground
TSYNC31/	Not Used
TSYNC21	Not Used
GND	Common Ground

the ISE/16 hardware, and supplements these features with a very powerful software-based program debugging environment.

The basic debugging features of IDBG16 are:

(1) Both high-level and assembly languages are supported.

(2) Breakpoints can be set at the source code level, even when using high-level languages.

(3) Variables can be accessed by their source code names, i.e., IDBG16 is symbolic in nature.

(4) Procedure parameters and local variables are easily displayed.

(5) Structured data types and pointers are easily displayed.

(6) Both command and history files are supported.

(7) Memory can be displayed in many different ways, including a disassembly mode displaying memory as 16032 instructions.

(8) All the emulation and debug facilities provided by the ISE/16 hardware are supported.

The ISE Monitor

When the ISE/16 unit is not running an emulation program, it is running a program called the ISE monitor. The monitor communicates with IDBG16 and it provides a command protocol that allows the host complete control of the ISE/16 hardware.

The monitor is invisible to the user, who normally communicates with the system via the friendly IDBG16 program.

Optional Terminal Feature

As an option, the ISE monitor communciates with a terminal connected to the ISE/16 unit. This terminal also serves as a terminal for the host. Thus the ISE/16 unit and the user's terminal share one RS232 port from the host.

Operation with the optional terminal is called Transparent Mode; operation without the optional terminal is called Stand-Aside Mode.

Conversion Kit for NS16008 In-System Emulation (Available December 1983)

An optional conversion kit is available for those who wish to do NS16008 development work. Contained in this kit are the following: ISE/08™ Emulator Pod, ISE/08 Symbolic Debugger (IDBG08), ISE/08 Monitor Firmware, and ISE/08 Manual. Thus, because the ISE Support Box can be used for either ISE/16 or ISE/08 development work, a user wishing to do NS16008 development work but who already has an ISE/16 unit can purchase this conversion kit (in comparison to the purchase of an entire ISE/08 unit).

ISE/16 Operation

Human Interface

ISE/16 is easy to learn and easy to use. The software includes a complete on-line manual. Invoking the "HELP" command gives a summary of all ISE/16 commands, an individual command, or an individual command's parameters. This feature helps the user get his work done quickly with less frustration.

Real-Time Emulation

The ISE/16 unit has its own CPU, MMU, and TCU components. These components are connected to the target system via cables, and they perform the same functions, with close to the same timing characteristics as they would if mounted in the target system. ISE/16 does not add wait states in its operation.

Emulation memory, resident in ISE/16, can be used in lieu of target system memory. This feature is implemented by the mapping capabilities. ISE/16 can run and debug programs, without a working target system.

User target memory address space (whether it exists or not) can be mapped onto the ISE/16 emulation memory. A memory read or write operation to an address mapped onto emulation memory is performed on emulation memory only and not on the target system's memory.

Memory from the entire 24-bit physical address space of the CPU or MMU can be mapped onto emulation memory if the following restrictions are observed:

(1) Up to four, non-contiguous segments can be defined.

(2) The address range mapped by a segment must lie within an integral 128K byte division of the address space, e.g. 00000 to h'1FFFF, or h'20000 to h'3FFFF.

(3) The address range mapped by a segment must start at the beginning of an integral 2K byte division of the address space, and end at the end of one such division e.g. h'00 to h'FFF, or h'2800 to h'37FF.

(4) The total memory space mapped by all segments must not exceed 30K bytes.

Associated with the emulation memory mapping scheme is a capability for write protection/detection. Any 2K byte block within any of the four 128K byte segments selected can be protected. A write operation to a protected memory segment causes an IM (Illegal Map) event to occur. Write operations to protected memory are inhibited only if they occur on emulation memory. They are not inhibited if they occur in target system memory.

Related commands:

MC — Map Create
MP — Map Print

Note: For the syntax of these, and other commands listed in this section, refer to the IDBG16 Command Summary.

Generalized Events

To provide a versatile way of observing and controlling the significant state changes on the microprocessor, ISE/16 allows the use and definition of "events". In general, a simple event is a breakpoint, a bus change, or a significant observation. An event can also be a logical combination of simple events (an Event-Expression).

Simple Event Definition

The simple events are:

- Breakpoints
- Latched Breakpoints
- Counter Done
- Status Pod Inputs
- Illegal Map
- Trace Done

Breakpoint Events

ISE/16 provides three common breakpoint events, named A, B, and C. The breakpoint event can be used in two ways:

(1) Execution Breakpoint — occurs just prior to execution of an instruction fetched from a specified address.

(2) Memory Reference Breakpoint — occurs on a match when sampling:

- Address Bits
- Data Bits
- External Status Bits
- User/Supervisor Pin
- High Byte Enable Pin
- Data Direction Pin
- And where any of the above options or bits can be masked.

Either virtual or physical addresses can be sampled.

ISE/16 also provides a range breakpoint event, R. The range breakpoint occurs on any read or write operation to an address in a specified address range.

All breakpoints can cause emulation to stop immediately. Also, if used with the No Stop (/NS) option, breakpoints can be combined with other events to cause a variety of action.

Latched Breakpoint Events, Counted Events

Latched breakpoint events, named LA, LB, LC, occur at some time after a cycle where the corresponding breakpoint event (A, B, or C) has taken place. The occurrence of a latched breakpoint event remains asserted until cleared.

Events, instructions, memory cycles, and clock cycles can be counted with the breakpoint counter (up to 12 bits). Upon reaching a certain count provided by the Define Counter (DC) command, the Counter Done (CD) event takes place.

Other Simple Events

The other simple events available are:

(1) IS0, IS1 — Status Pod Input Sync 0 and Input Sync 1.

(2) IM — Write operation to write-protected address.

(3) TD — End of trace.

Related commands:

BC — Breakpoint Create

BD — Breakpoint Delete

DP — Breakpoint Print

Event-Expressions

An event-expression is a Boolean expression made up of simple events, i.e., a logical combination of simple events. This allows the user to generate many different event combinations, tailored to system activity of particular interest to the user. These generalized events are used by many ISE/16 commands such as stop, trace, event counting, etc. Event-expressions provide creative and flexible debugging procedures.

Event-expressions can be evaluated as either logically true or logically false. Valid logic operations for event-expressions are: Negation (NOT), AND, and OR.

Stopping Execution on Events

A common debugging activity is to stop emulation on the occurrence of an event of interest. Stopping emulation puts ISE/16 in the monitor mode so the user can examine and alter the state of the CPU, memory, and ISE/16 functions. Emulation can be stopped on either simple events or event-expressions.

Related commands:

DS — Define Stop

BS — Breakpoint Create

Flexible Tracing

ISE/16 maintains a 255-entry trace memory. Trace memory captures bus activity in one of two trace modes. The trace modes are:

- Program Flow Trace
- Memory Bus Trace

Program Flow Trace

The Program Flow Trace mode captures the CPU Program Counter address of 255 non-sequential instructions. This mode also maintains a count of sequential instructions executed between each non-sequential instruction stored in the trace memory.

Memory Bus Trace

The Memory Bus Trace mode captures a summary of system parameters during 255 memory bus access cycles. The following parameters are captured:

- Address bus contents
- Data bus contents
- CPU Status (data transfer, non-sequential fetch, interrupt acknowledge, etc.)
- Status Pod External Inputs
- States of the Following CPU Pins:
  PFSC — Program Flow Status (start of instruction)
  UNS — User/Not Supervisor
  NHBE — Not High Byte Enable
  NDDIN — Not Data In

A tracing event can qualify the memory bus tracing mode. This event allows the user to reduce the number of events captured.

When enabled, tracing in either mode continues until a specified terminating event occurs. The actual end of tracing can be delayed after the terminating event by a count of 1 to 255. This allows trace data to be captured before, after, or around the terminating event.

Execution Timer

The execution timer is a 24-bit counter with an overflow flag that may be used to count events, instruction cycles, memory cycles, or clock cycles. The timer may be programmed to start and stop counting on specific events. This permits using the execution timer to determine relative timing differences between various events. One use of this feature is to measure software or hardware performance.

Related Commands:

DE — Define Execution Timer
LD — List Definitions

Event Trigger for External Test Equipment

ISE/16 events can trigger external test equipment, such as oscilloscopes and logic analyzers. This test equipment can be used in conjunction with ISE/16 debugging features to solve system timing problems. Two external trigger sources are provided:

- General Event (or Event-Expression)
- Trace Trigger Event (i.e., an event that causes an entry into trace memory.)

The external trigger signals are available at two status pod outputs:

- BKSYNCH/-U (General Event)
- TRSYNCH/-U (Trace Trigger Event)

Related Commands:

DO — Define Output Sync Command

ISE/16 Timing Options

ISE/16 includes the following timing options:

- Sampling time can be set to sample either virtual or physical addresses
- Status Pod external lines can be sampled at either data valid or address valid times
- The emulation clock frequency can be set to one of the following frequencies:

  1.5 MHz
  3.0 MHz
  6.0 MHz
  Target Board Frequency

Note: Selection of target board frequency may require synchronous and/or asynchronous delay compensation. Refer to ISE/16 User's Manual, Chapter 6, for details.

Related Commands:

SO — Select Options

Self-Test Diagnostics

At power-up or reset, ISE/16 runs a diagnostic program to verify ISE software integrity and proper hardware function.

Required User-Supplied Equipment

For use under VAX/11 systems:

- Valid DEC VAX/11 configuration, with available RS232 port.
- VMS Operating System, Version 3.0 or later.
- NSX-16 Cross Software Package, or NS-ASM-16 NS16000 Cross-Assembler Package.

For use with STARPLEX II systems:

- STARPLEX II Development System.
- STARPLEX II Operating System, Version G or later.
- SFW-90-A010 NS16000 Cross-Assembler Package.

For use with a system that has a Berkeley 4.1 based UNIX™ Operating System:
[Contact Marketing for Availability Information.]

- Valid computer system with an available RS232 port.
- Appropriate cross software package.
  [Contact Marketing for further information.]

Specifications

Environmental	Operating Temperature +10°C to +40°C Storage Temperature −20°C to +65°C
Power	3A @ 115 V_{AC}, 50/60 Hz, single phase 1.5A @ 220 V_{AC}, 50/60 Hz, single phase Approximately 1170 BTU.
Physical	
ISE Support Box —	Height: 4.125 in. (10.5 cm) Width: 19.0 in. (48.3 cm) Depth: 17.5 in. (44.5 cm)
Emulation Pod —	Height: 2.25 in. (6.4 cm) Width: 9.25 in. (23.5 cm) Depth: 14.0 in. (35.6 cm)
TTL Status Pod —	Height: 1.0 in. (2.5 cm) Width: 3.125 in. (7.9 cm) Depth: 6.125 in. (15.6 cm)
Cable Lengths —	ISE Support Box to Emulation Pod: 4.0 ft. (1.22 M) ISE Support Box to TTL Status Pod: 6.0 ft. (1.83 M) Emulation Pod to Target Board: 1.0 ft. (0.30 M)

Electrical

Operating Frequency —	User selectable to one of the following: 1.5 MHz 3.0 MHz 6.0 MHz Target Board Frequency **Note:** Selection of target board frequency may require synchronous and/or asynchronous delay compensation. Refer to ISE/16 User's Manual, Chapter 6, for details.
Target Interface Electrical Characteristics —	See Tables 3 through 5.

Order Information

Complete ISE/16 Units

NS-ISE-16	ISE/16 (NS16032), 115 V_{AC} for VAX/11 (VMS) Computer System.
SPM-90-A1632	ISE/16 (NS16032), 115 V_{AC} for STARPLEX II Development Systems.
NS-SYS-2004	ISE/16 (NS16032), 115 V_{AC} for UNIX OS based operating systems. [Contact Marketing for Availability Information.]

Conversion Kits to Allow for ISE/08 Emulation
[Contain ISE/08 Emulator Pod, ISE Debugger (IDBG08), appropriate ISE/08 monitor firmware, and ISE/08 manual.]

AEE-90-A1608	ISE/16 to ISE/08 kit for STARPLEX II use.
AEE-ISE-08	ISE/16 to ISE/08 kit for VAX/11 (VMS) use.
AEE-ISENIX-08	ISE/16 to ISE/08 kit for UNIX based OS systems use. [Contact Marketing for Availability Information.]

Documentation

420306675-002	ISE/16 User's Manual (Included with NS-ISE-16, and SPM-90-A1632.)

Table 3. Electrical Characteristics for TCU Interface

Signal Name	Interface Device	Input And/Or Output Current		Propagation Delay Time T_{pd}*
		I_{OH}	I_{OL}	
OUTGOING SIGNALS:				
NTSO	74S244	15mA	64mA	14.6ns
CTTL	74S244	15mA	64mA	14.6ns
FCLK	74S244	15mA	64mA	14.6ns
NDBE	74S244	15mA	64mA	14.6ns
NRD	74S244	15mA	64mA	14.6ns
NWR	74S244	15mA	64mA	14.6ns
NRST	74S244	15mA	64mA	14.6ns
RDY	74S244	15mA	64mA	14.6ns
		I_{IH}	I_{IL}	
INCOMING SIGNALS:				
NPER	74S244	50μA	400μA	14.6ns
NCWAIT	74S244	50μA	400μA	14.6ns
NWAIT1	74S244	50μA	400μA	14.6ns
NWAIT2	74S244	50μA	400μA	14.6ns
NWAIT3	74S244	50μA	400μA	14.6ns
NWAIT4	74S244	50μA	400μA	14.6ns
XCTL1	74S244	50μA	400μA	14.6ns
NCEN	74S244	50μA	400μA	14.6ns
NRST1	74S244	50μA	400μA	14.6ns

*Interface device, plus cable.

Table 4. Electrical Characteristics for MMU Interface

Signal Name	Interface Device	Input And/Or Output Current		Propagation Delay Time T_{pd}*
		I_{OH}	I_{OL}	
OUTGOING SIGNALS:				
A24	74S244	15mA	64mA	14.6ns
MMUMINT	74S244	15mA	64mA	14.6ns
NPAV	74S244	15mA	64mA	14.6ns
NABT	74S244	15mA	64mA	14.6ns
NFLT	74S244	15mA	64mA	14.6ns
NHLDA0	74S244	15mA	64mA	14.6ns
		I_{IH}	I_{IL}	
INCOMING SIGNALS:				
NHOLD	74S244	50μA	400μA	14.6ns

*Interface device, plus cable.

Table 5. Electrical Characteristics for CPU Interface

Signal Name	Interface Device	Input And/Or Output Current				Propagation Delay TimeT_{pd}*
		I_{OH}	I_{OL}	I_{IH}	I_{IL}	
BIDIRECTIONAL SIGNALS:						
NSPC	none	—	—	—	—	1.4ns
AD15	8T28	10mA	48mA	25μA	200μA	18.4ns
AD14	8T28	10mA	48mA	25μA	200μA	18.4ns
AD13	8T28	10mA	48mA	25μA	200μA	18.4ns
AD12	8T28	10mA	48mA	25μA	200μA	18.4ns
AD11	8T28	10mA	48mA	25μA	200μA	18.4ns
AD10	8T28	10mA	48mA	25μA	200μA	18.4ns
AD09	8T28	10mA	48mA	25μA	200μA	18.4ns
AD08	8T28	10mA	48mA	25μA	200μA	18.4ns
AD07	8T28	10mA	48mA	25μA	200μA	18.4ns
AD06	8T28	10mA	48mA	25μA	200μA	18.4ns
AD05	8T28	10mA	48mA	25μA	200μA	18.4ns
AD04	8T28	10mA	48mA	25μA	200μA	18.4ns
AD03	8T28	10mA	48mA	25μA	200μA	18.4ns
AD02	8T28	10mA	48mA	25μA	200μA	18.4ns
AD01	8T28	10mA	48mA	25μA	200μA	18.4ns
AD00	8T28	10mA	48mA	25μA	200μA	18.4ns
OUTGOING SIGNALS:						
A23	74S244	15mA	64mA	—	—	14.6ns
NILO	74S244	15mA	64mA	—	—	14.6ns
ST0	74S244	15mA	64mA	—	—	14.6ns
ST1	74S244	15mA	64mA	—	—	14.6ns
ST2	74S244	15mA	64mA	—	—	14.6ns
ST3	74S244	15mA	64mA	—	—	14.6ns
NPFS	74S244	15mA	64mA	—	—	14.6ns
NDDIN	74S244	15mA	64mA	—	—	14.6ns
NADS	74S244	15mA	64mA	—	—	14.6ns
UNS	74S244	15mA	64mA	—	—	14.6ns
NHBE	74S244	15mA	64mA	—	—	14.6ns
HHLDA	74S244	15mA	64mA	—	—	14.6ns
A22	74S244	15mA	64mA	—	—	14.6ns
A21	74S244	15mA	64mA	—	—	14.6ns
A20	74S244	15mA	64mA	—	—	14.6ns
A19	74S244	15mA	64mA	—	—	14.6ns
A18	74S244	15mA	64mA	—	—	14.6ns
A17	74S244	15mA	64mA	—	—	14.6ns
A16	74S244	15mA	64mA	—	—	14.6ns
INCOMING SIGNALS:						
TSYSPWR	1N4002	—	—	—	—	—
NINT	74S244	—	—	50μA	400μA	14.6ns
NNMI	74S244	—	—	50μA	400μA	14.6ns
NHOLD	74S244	—	—	50μA	400μA	14.6ns

*Interface device, plus cable.

Documentation Conventions

The following documentation conventions are used in describing the IDBG16 commands and parameters. Upper-case and lower-case letters are used in these conventions; any combination of upper-case and lower-case letters may actually be used when entering commands.

UPPER-CASE letters show the command letters, parameters and options. The names must be entered exactly as shown.

Spaces and blanks have been added for readability. When actually entering commands, spaces and blanks may only appear between the command and its parameters and between the parameters and the local radix.

< >—angle brackets enclose descriptive names (in lower-case) for user-supplied parameters/options.

{ }—braces enclose more than one item out of which one, and only one, must be used. The items are separated from each other by a logical OR sign "|".

[ ]—brackets enclose optional item(s).

|—logical OR sign separates items out of which one, and only one, may be used.

...—three consecutive periods indicate optional repetition of the preceding item.

IDBG16 Command Summary

The following is a comprehensive list of the IDBG16 commands. Commands are in alphabetical order. See the ISE/16 User's Manual for a detailed description of each command.

Command	Syntax	Function
Begin	B [<file>] [/NL] [/NI] [/R] [/Z]	(if no switches) Loads the program <file> into target board memory, and initializes registers. /NL— No Load /NI — No Initialize /R— Reset /Z—Zero-Fill data areas
Breakpoint Create	BC [A,|B,|C,] <address> [/NS]	Creates execution breakpoint A,B, or C at specified <address>. /NS— No Stop
	BC [A,|B,|C,] {<address> | <mask>} <breakpoint-options> [/NS]	Creates memory reference breakpoint A, B, or C at address specified by <address> or <mask> and with specified <breakpoint-options>. /NS— No Stop
	BC R, <address-range> [/NS]	Creates range breakpoint R at specified <address-range>. /NS— No Stop
Breakpoint Delete	BD [A|B|C|R]	Deletes specified breakpoints.
Breakpoint Revive	BR [A|B|C|R]	Revives specified breakpoint.
Breakpoint Print	BP [A|B|C|R]	Prints address and conditions of specified breakpoints.
Command File	@ {<file> | <n>}	Executes command <file> or debugger string sequence beginning at <n>.
Debugger String	$ <n> = [<string>]	Sets debugger string <n> to <string>.
Define Counter	DC <n> [/B =<event-expression>] [/C = {<event-expression> |I|M|C}] DC/R	Defines set up for ISE counter. <n>— Number of counts /B— Begin event /C—Counter type /R— Reset

IDBG16 Command Summary (Continued)

Command	Syntax	Function
Define Execution-Timer	DE [/B =<event-expression>] [/E =< event-expression>] [/C ={<event-expression> |I|M|C}] DE/R	Defines set up for ISE execution timer. /B — Begin event /E — End event /C — Count type /R — Reset
Define Output Sync	DO <event-expression>	Defines output sync event.
Define Stop	DS <event-expression>	Defines stop event.
Define Trace	DT [/E =<event-expression> [/D =<n>]] [/P | /M [= { <address> | <mask>}] <qualify-options>]]]	Defines the end, delay, and trace mode parameters for trace. /E — End event /D — Delay count /P — Program Flow mode /M — Memory Bus mode /R — Reset
Disassemble	D <address-range> [/I =<n>] [/NA]	Disassemble instructions in <address-range>. /NA — No Address /I — Number of Instructions to be disassembled.
Go	G [/F =<address>] [[/T =] <breakpoint>]	Starts execution of the program at the current PC address of from the <address>. Execution continues until <breakpoint>.
Help	H [<string>]	Displays general help or command syntax or parameter syntax.
In	I {<address> | <register>} [<radix>] <value1> [<value2>]	Checks that contents of <address> or <register> are in the range specified by <value1> and <value2 >, inclusively.
List Calls	LC [<n>]	Lists first <n> entries in call chain.
List Definitions	LD [/T|/E|/C|/O|/S]	Lists current definitions. /T — Trace definition /E — Execution timer definition /C — Counter definition /O — Output sync definition /S — Stop definition
List Files	LF [<line> [/<file>]]	Lists lines in <file>.
List Information	LI	Lists current IDBG16 status.
List Modules	LM	Lists modules in current program.
List Procedures	LP	Lists procedures in current program.
List Strings	LS	Lists current debugger string values.
List Trace	LT [<n> |*|/A|/J]	Lists nine trace entries centered around entry <n> or around the trigger point (*). /A — All entries /J — Jumps only

IDBG16 Command Summary (Continued)

Command	Syntax	Function
Map Create	MC [<address-range>] [/S = {A|B|C|D}] [/M = <n> |/NM] [/P|/NP]	Creates segment assignment and mapping for special <address-range>. /S — Segment assignment /M — Mapping to ISE block <n> /NM — No Mapping /P — Write Protection /NP — No Protection
Map Print	MP [/S = {A|B|C|D}]	Prints current mapping for specified segment.
Memory Fill	MF <address-range> [<radix>] <value>	Fills memory at <address-range> with <value>.
Memory Move	MM <address-range>,<address> [<radix>]	Moves memory from <address-range> to <address>.
Memory Search	MS <address-range> [<radix>] <value>	Searches for <value> in <address-range>.
On	O {FAIL|RESET|NMI|EXIT [({A|B|C|R|S|HC})]} {@ <n> |/O}	Sets IDBG16 response on condition: @ <n> — Executes debugger string sequence on condition beginning with $ <n>, /O — Response Off, restores normal response.
Print	P [<address-range> |<register-range>] [<radix>]	Prints contents of <address-range> or <registers>.
Print Address	PA <address>	Prints absolute address and module area associated with <address>.
Protection Create	PC <address-range> [/UW|/UR|/SW|SR] [/V|/NV] [/R|/NR] [/M|/NM] [/T = <gn>] [/P]	Creates protection/translation for pages specified by <address-range>.
Potection Print	PP [<address-range>]	Prints protection level status for pages specified by <address-range>.
Quit	Q [/S]	Terminates session. /S — Save IDBG16 status in IDBG16.IND
Repeat	<cr>	Repeats previous command.
Replace	R {<address> | <register>} [/NV] [<radix>] [<value>]	Replaces contents of <address> or <register> with <value>. /NV — No Verify
Select Echo	SE [/O]	Selects echo mode. /O — Echo Off.
Select Full	SF [/O]	Selects full symbolic PC. /O — Full Off.
Select History	SH {<file> [/F] |/O}	Selects history file <file>: /F — Full history (with responses) /O — History Off
Select Link	SL <file> [,<file>] [/L]	Selects communications channel(s): /L — List communications
Select Module	SM <module>	Selects module.

IDBG16 Command Summary (Continued)

Command	Syntax	Function
Select Options	SO [/AS={NADS|NPAV} [/XS={D|A}] [/EC={10|5|2.5|U}] [/MC={10|EC}] [/L={C|NC}] [/T={0|1|N|A}]	Selects current ISE operation options. /AS — Address Sample time /XS — External Sample time /EC — Emulator Clock frequency /MC — Monitor Clock frequency /L — Latched Clear or No-Clear /T — Translation
Select Procedure	SP [<n> | <procedure>:]	Selects procedure.
Select Radix	SR <radix>	Select global radix.
Step	S [<gn>]	Executes <gn> machine instructions (Assembly programs) or one Pascal statement (Pascal programs). <gn> illegal in Pascal.
Step Call	SC	Executes until a call or return.
Step Down	SD	Executes one instruction inside a procedure; skips over call instructions.
Step Instruction	SI [<gn>]	Executes <gn> machine instructions.
Step Until	SU {<address> | <register>} [<radix>] <value1> [<value2>]	Executes instructions until contents of <address> or <register> within the range specified by <value1> and <value2>.
Step While	SW {<address> | <register>} [<radix>] [<value1>] [<value2>]	Executes instructions while contents of <address> or <register> are within the range specified by <value1> and <value2>.

Parameter Summary

The following is a comprehensive list of command parameters. See the ISE/16 User's Manual for a detailed description of each.

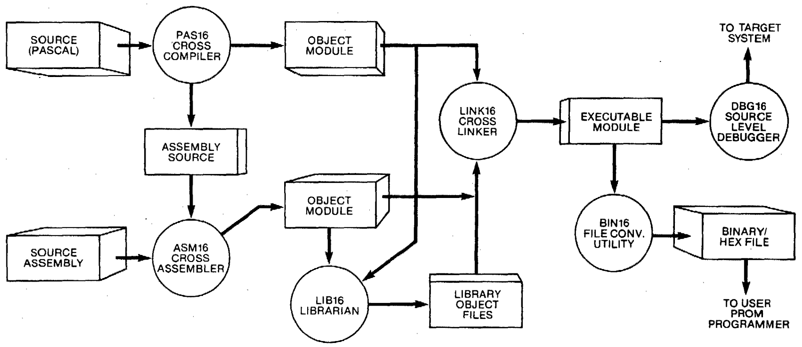
Command	Syntax	Function
Number (<n>)	<digits>	An unsigned, decimal number in the range 0 to 32767.
General Number (gn)	[H'|Q'|O'|D'] [-] <digits>	A signed, octal, decimal, or hexadecimal number in the range of -2^{31} to $2^{31}-1$.
Mask	M'{0|1|X|_}...	An unsigned number which consists of binary digits, "don't care" bits, and optional underscores. A mask represents the value of address, data, status, or external bits.
Name	<letters, numbers, underscores, tildes>	A combination of letters, digits, underscores, and tildes which does not start with a digit.
Module	<name>	The name of a module in the program.
Procedure	<name> [# <n>]	The name of a procedure in the selected module. # <n> specifies the <n>th procedure having <name> in the selected module.
Symbol	<name>	The name of a variable in the selected module or procedure.
Register	<register-name> [% <n>]	One of the registers shown in Table 3. % <n> specifies the field starting at the <n>th bit in <register>.

Parameter Summary (Continued)

Command	Syntax	Function
Register-Range	{•CPU__ | •MMU__ | •FPU__ | •PSR__ | •MSR__ | •FSR__ | <register>}	Specifies all CPU, MMU, or FPU registers or all PSR, MSR, and FSR fields.
Address	<basic-address> [+ <abs> | – <abs> | % <abs> | ∧ | • <field> | <indexing>]...	A byte or bit address. The address consists of a basic address and any combination of optional operators. +<abs> — Adds <abs> to address. –<abs> — Subtracts <abs> from address. %<abs> — Takes <abs>th bit at address. ∧ — Takes contents as address. • <field> — Address is address of a field in a Pascal record. <indexing> — Address is address of an array element.
Address-Range	{<address1>..<address2> | < address> ! <n> | <address>}	The range of addresses from <address1> to <address2>, from <address> to <address> + (<n> – 1)*(<current radix>), or from <address> to itself.
Radix	[%] <n> <base>	Specifies the length and type of input/output in IDBG16 commands. [%] specifies length. If % is specified, length is in bits. <n> must be within the range 1 to 256. <base> specifies type and may be binary (B), decimal (D), octal (O), hexadecimal (H), hexadecimal dump (H), floating-point (F), logical (L), ASCII (A), Pascal set (S), or Pascal string (G).
Value		A value to be entered or displayed after issuing a Print, Replace, Step, Memory, or In command. Syntax is defined by current radix.
File		The name of any file in the host system. File syntax is host dependent.
Event	{IS0|IS1|IM|TD|CD|A|B|C|LA|LB|LC|R}	The name of an ISE response to a specific set of run-time conditions.
Event-Expression	([′] <event> [*[′] <event>...[+ [′] <event> [*<event>]...]...]) (1) (0)	A Boolean expression consisting of one or more <events>s and the logical NOT (′), AND (*), and OR (+) operators. (1) — always true. (2) — always false.

National Semiconductor

NSX16™ Cross Software Package



- **Runs under STARPLEX II™ operating system or DEC® VAX™/VMS™ operating system**
- **Compatible with ANSI standard Pascal**
- **Supports NS16081 floating-point unit**
- **Pascal run-time support environment for DB16000 development board**
- **Pascal compiler directly produces NS16000 code**
- **High-level symbolic debugger allows debug at source level**

Product Overview

NSX16 is a comprehensive software development package that includes all the components necessary to produce NS16000 native code. Intended as a support package to facilitate the development of software for NS16000-based systems, NSX16 has been designed to run on two hardware configurations. These are National Semiconductor's STARPLEX II running the STARPLEX II operating system and Digital Equipment's VAX11 series running the VMS operating system.

Consisting of a Pascal compiler, NS16000 cross-assembler, linker, librarian, and source-level debugger, NSX16 provides the full ensemble of tools to make the generation of NS16000 code an easy task. Code thus developed may then be downloaded via a serial port to the DB16000 development board or the ISE/16™ emulator for execution and debug.

NSX16 consists of the following components:

- — PAS16, the Pascal Cross-Compiler
  Note: Not available for STARPLEX II.
- — RTS16, the Run-Time Support Package
  Note: Not available for STARPLEX II.
- — ASM16, NS16000 Cross-Assembler
- — FPS16, Floating-Point Support
- — LINK16, the Cross Linker
- — LIB16, the Librarian
- — BIN16, the File Conversion Utility
- — DBG16, the Source-Level Symbolic Debugger
- — MON16, DB16000 Monitor (Firmware)

PAS16

Designed to be compatible with the ANSI standard, with listed extensions and restrictions, the Pascal cross-compiler is capable of accepting compatible Pascal source and generating NS16000 code. Extensions include features such as IMPORT/EXPORT in support of full modularity and FAST variables for code optimization. Also included is the run-time support environment for the DB16000 development board.

Note: PAS16 is not available for STARPLEX II Development Systems.

ASM16

The cross-assembler produces relocatable NS16000 object code. It accepts complex expressions, floating-point scientific notation, external symbol references and can handle external address arithmetic.

FPS16

The Floating-Point Support Library provides floating-point mathematical routines, which can be called from the user Pascal or Assembly language program, and emulation of the 16081 Floating-Point Unit (FPU), which is user-transparent. FPS16 also allows the user to control how FPS16 operates (for example, the user can specify whether FPS16 rounds towards zero, positive or negative infinity).

FPS16 conforms to the IEEE Standard formats and provides all required and most of the recommended IEEE Standard routines of IEEE Task P754 Draft 10 of *A Proposed Standard for Binary Floating-Point Arithmetic.*

LINK16

Modules generated by the cross-compiler or assembler are linked by LINK16 to produce executable modules. LINK16 is interactive, allowing the user to include additional files and libraries at link time whenever symbol matching is unsuccessful. LINK16 provides an extensive repertoire of directives to support complex system configurations. Directives can be entered from disk or directly from the console. LINK16 permits user control of RAM/ROM allocation.

LIB16

The librarian maps module characteristics and builds module libraries.

BIN16

The BIN16 program is a utility that converts 16000 executable files into a format acceptable for PROM programmers.

DBG16

DBG16 is an interactive symbolic debugger that allows debugging at the source level. It communicates with the DB16000 monitor via a serial link allowing execution and debug on the board. Source code may be displayed and breakpoints set by line numbers. Single-stepping is possible at the machine instruction level, the Pascal statement level, the procedure level, or until a register/address value match occurs. DBG16 supports debugging of multimodule programs, drawing all information needed to support debug from source files and the output of the linker. It supports command files and output to a file to serve as a history file.

MON16

The monitor program enables the user to load, execute and debug programs. It has the following features:

- Examine/Change Registers/Memory
- Block Move/Load/Print memory locations
- Search for/Fill block of data
- Insert multiple breakpoints
- Start program and break after a specified number of instructions
- Step for specified number of instructions
- Step until (while) a specified condition is reached

The following additional debug features may also be implemented with the Memory Management Unit (MMU) present:

- Break on memory read and/or write (virtual or physical)
- Break on non-sequential instruction fetch
- Show last 2 non-sequential instruction addresses fetched and the number of instructions executed after each nonsequential fetch

Supported Hardware

- STARPLEX II
  STARPLEX™ Operating System
  Revision G or later
- DEC VAX11 Family
  VMS Operating System version 2.X or later

Shipping Package

VAX Version:
1600 bpi mag tape (9-track VMS copy format)
User and reference documentation

STARPLEX II Version:
Compatible 8″ floppy diskette, standard soft sector format
User and reference documentation

Order Information*

NSX16	Cross Software Package, VAX/VMS version (Includes PAS16, RTS16 ASM16, LINK16, MON16, LIB16, FPS16, BIN16 and DBG16)
NS-PAS16	Pascal Cross-Compiler, VAX/VMS version
NS-ASM16	NS16000 Cross-Assembler, VAX/VMS version (Includes ASM16, LINK16, LIB16, BIN16 and DBG16)
SFW-90-A010	NS16000 Cross-Assembler, STARPLEX II version (Includes same as NS-ASM16)

*Software license agreement must be signed prior to order entry.

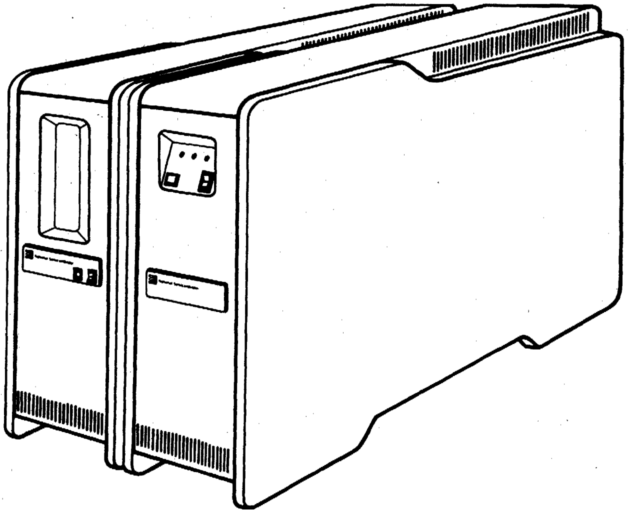
Documentation

The manuals listed below are included with the NSX16, and may also be ordered separately.

420306612-002	Cross-Assembler Reference Manual
420306618-002	Pascal Reference Manual**
420308038-002	Run-Time Support Library Reference Manual**
420306617-002	Cross-Support Utilities Reference Manual
420306676-002	Symbolic Debugger Reference Manual
420308220-002	Floating-Point Support Library Reference Manual
420308221-002	DB16000 Monitor Reference Manual
424009011-002	NSX16 Operations Manual

**Not available for STARPLEX II.

SYS16™ Multi-User Development System for the NS16000™ Microprocessor Family



TL/F/5266–1

- **GENIX™ enhanced Berkeley 4.1 bsd UNIX operating system**
- **Time-shared support for up to eight users**
- **NS16032 Microprocessor Family based**
- **Demand-Paged Virtual Memory (DPVM) support**
- **Easy to use, proven programming environment**
- **1.25 MB RAM, expandable to 3.25 MB**
- **20 MB Hard Disk, expandable to 140 MB**
- **Streamer Tape backup, with 20 MB cartridges**
- **C and Pascal High Level Language Compilers**
- **NS16000 assembler**
- **Supports emulation of NS16000 Microprocessor Family**

Product Overview

The SYS16 is a multi-user development system which provides powerful software and hardware tools for the development of applications using National Semiconductor's NS16000 Microprocessor Family components.

Based on the NS16032 16-bit Microprocessor, SYS16 gives the designer access to an assembler, high level language compilers and real-time In-System Emulation (ISE™) tools. Total development support is provided for up to eight users, on a time-shared basis.

The SYS16 includes two main modules: the Processor module, which houses most of the electronics and the Disk-Tape module, which houses the hard disk and streamer tape back-up.

Optional disk drive modules may be added to increase system capacity. Disk drive modules contain two drives of 20 MB each.

One terminal is provided with the system. Additional terminals may be added to the system as the demand warrants. Emulation and software development work may be performed concurrently. Shared resources of the hard disk and user-supplied printer lowers the system's cost per user.

National's GENIX Operating System is an enhanced version of Berkeley 4.1 bsd UNIX. These enhancements have been added to fully utilize the advanced architecture of the NS16032 Micrprocessor Family.

Hardware Description

Processor Module

This six-slot module houses most of the electronics for the SYS16. Standard configuration includes boards installed in four of the slots, with the remaining two available for additional Random Access Memory boards. The bus is an extended-CPU National proprietary bus designed for fast interface between the boards. The four boards provided in the standard configuration are the CPU, Serial I/O, Memory, and Disk-Tape Controller.

The CPU board is based on the NS16032 microprocessor family and includes the CPU plus the NS16082 Memory Management Unit, the NS16201 Timing Control Unit, the NS16081 Floating-Point Unit and the NS16202 Interrupt Control Unit. The CPU board also contains diagnostic firmware, one parallel I/O port, one GPIB IEEE-488 port, one RS232 port, and 256 kB of RAM.

The Serial I/O board contains logic supporting eight RS232 ports.

The memory board contains 1 MB of RAM with error checking and correction. Access time is 400 ns. Additional memory boards may be added to the system, up to a total of 3.25 MB.

The Disk-Tape Controller board contains the necessary electronics to control the disk drives and the streamer tape.

Disc-Tape Module

This module houses an 8-inch Winchester hard disk with a capacity of 20 MB (17.8 MB formatted). It also contains a ¼″ streamer tape for backup and ready access for Operating System and other software updates. The tape cartridge has a 20 MB capacity.

Disc only Module

Additional hard disk memory may be added to the system. Disk-only modules are available which house 40 MB each. A total of 3 modules may be added for a total of 140 MB.

Hardware Support:

Parallel Printer Interface: Centronics interface is provided to support both 700 and 300 series printers.

Prom Programming: support is provided for Data I/O System 19.

Software Description

The SYS16 includes the GENIX operating system, an enhanced version of Berkeley 4.1 bsd UNIX. These enhancements allow GENIX to fully support the features of the NS16000 family, providing an advanced, proven programming environment.

The GENIX operating system is a time-shared, demand-paged system with protected address spaces, supporting from one to eight users. It is completely compatible with the NSXC16 cross software package.

Included are a C compiler, based on Berkeley's portable C compiler, NS1600 assembler, linker, libraries, utilities, loader, editor, and debugger. Virtually all of the utilities that make UNIX a powerful operating system are provided.

A Pascal compiler is available as an option.

Physical Specifications

The standard SYS16 consists of the Processor Module, Disk-Tape Module, one terminal, the required interconnect cables, and supporting manuals.

Processor Module: this is a rectangular floor mounted unit with front mounted controls and indicators, and rear mounted I/O connections.

Height:	24 inches
Width:	7.5 inches
Depth:	27 inches
Color:	beige side panels with grey inner frame, and black front and rear
Weight:	38 pounds

Disk-Tape Module: this unit is physically similar to the processor module, with the exception of the Weight, which is 48 pounds.

Terminal: DEC VT100 compatible.

Environmental:

Altitude:	25,000 ft. non-operating 15,000 ft. operating
Temperature:	−20°C to 65°C non-operating 10°C to 40°C operating
Humidity:	5% to 80% max wet bulb 32°C minimum dew point 2°C

Electrical:

Processor Module:

FCC:	Class A
AC Voltage:	90–130 or 180–260 V_{AC}; 47–63 Hz
Fusing:	6A-Domestic 3A-European

Disk-Tape Module: same as processor module

Odering Information

Systems:

NS-SYS-1001:	Full system: Processor Module, Disk-Tape Module, one Terminal, GENIX operating system, cables, and manuals.
NS-SYS-1001E:	same as above configured for European power.

Accessories:

NS-SYS-2001:	Disk Drive Expansion Module with 40 MB
NS-SYS-2001E:	same as above configured for European power
NS-SYS-2002:	1 MB RAM Expansion Board
NS-SYS-2003:	Terminal
NS-SYS-2003E:	Terminal with European power configuration
NS-SYS-2004:	ISE for 16032
NS-SYS-2004E:	ISE for 16032 with European power configuration
NS-SYS-2005:	20 MB Streamer Tape Cartridge
NS-SYS-2006:	Hardware manual
NS-SYS-2007:	Software manual
NS-SYS-2008:	ISE for 08032
NS-SYS-2008E:	ISE for 08032 with European power configuration

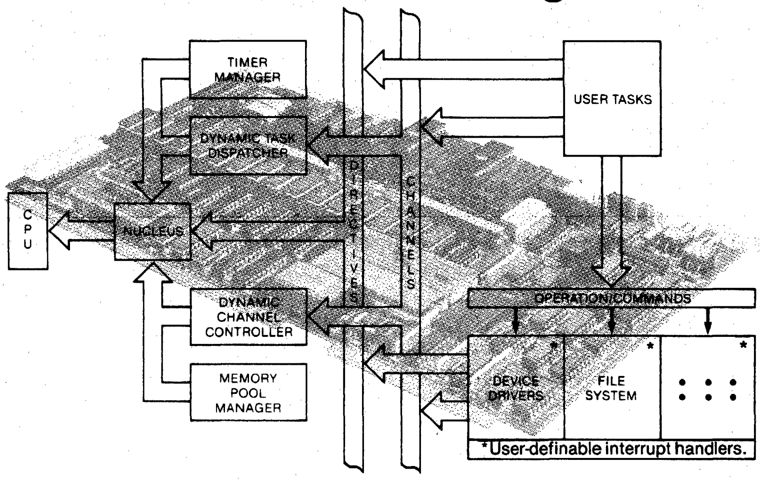
Software:

NS-SYS-3001:	Pascal software

Software

National Semiconductor **PRELIMINARY**

BLMX-16
Board Level, Multitasking Executive



- **Provides a multitasking executive for real-time applications**
- **Supports all NS16000 CPUs**
- **Port of Proven BLMX-80**
- **Easily Configurable Source Package**
  - —Fully user configurable
  - —Hardware independent
- **Extensive user implementation support**
  - —Unique demo, program introduction
  - —Step through configuration
  - —Sample terminal INT handlers
  - —Integrated with 16000 developent boards and monitor
  - —For debug without ISE™
- **ROMable**
- **Reconfigurable**
- **Real-time clock support for time-of-day and event scheduling**
- **Allows up to 256 levels of task priority which can be dynamically assigned**
- **Up to 256 logical channels for task communication**
- **Memory pool management**

Product Overview

The BLMX-16 (Board Level Multitasking Executive) is National Semiconductor's real-time, multitasking executive for NS16000 CPU based applications. Its primary purpose is to simplify the task of designing application software and provides a base upon which users can build a wide range of application systems. BLMX-16 requires only 2K bytes of RAM and only 4K bytes of ROM and is fully compatible with National Semiconductor's NS16000 CPU family and the DB16000 development board.

BLMX-16 allows the user to monitor and control multiple external events that occur asynchronously in real-time, such as intertask communications, system resource access based upon task priority, real-time clock control, and interrupt handling. These functions greatly simplify application development in such areas as instrumentation and control, test and measurement, and data communications. In these applications, BLMX-16 provides an environment in which system programmers can immediately implement software for their particular application without regard to the details of the system interaction.

BLMX-16 executive is fully modular and can be readily configured to suit application needs. It is both hardware and location independent, thus providing a fundamental base on which to build a wide range of applications systems. In addition, it provides a buslike structure that helps to integrate software with the underlying hardware through predefined data structures and interconnect procedures. This architecture ensures maximum standardization for both compatibility and future expansion.

Features

Structured Environment—The BLMX-16 executive and its associated modules support and encourage modular, structured programming, thus providing a consistent structure from application to application, which allows experience gained and software written on one system to be easily transferred to another. Frequently, entire programs may be used in multiple applications, *even if different CPU boards are involved.*

Hardware-Oriented Interface—The BLMX-16 executive provides for an intertask and task/executive communications architecture that is similar to hardware communications. Instead of an array of "mailboxes" (or "message centers"), BLMX-16 incorporates channels. One merely communicates across a channel from his module to the desired destination. This interface is consistent throughout the range of facilities offered, thus reducing the number of concepts to be learned, providing greater control at the task level, and increasing the efficiency of the system and the programming effort.

Wide Choice of CPUs—BLMX-16 is compatible with the full line of 32-bit NS16000 CPUs offered. These include the NS16008, NS32032, and the NS16C032. This means users will be able to move a NS16032 system:

- to an NS16008 for cost-effectiveness,
- to an NS32032 for increased computing power, or
- to an NS16C032 for military or other low-power application.

Time-Of-Day Clock—The BLMX-16 executive has an integral system/time-of-day clock. Included is a real-time clock configurable to a resolution of 10ms. This eliminates the need to allocate the extra memory otherwise required for this feature.

Small Nucleus—The BLMX-16 nucleus was hand-coded in assembly language rather than being compiled from intermediate or high-level languages. The resulting product is therefore smaller and allows the incorporation of more features within the size generally accepted as optimum.

Priority-Oriented Scheduler—The BLMX-16 scheduler ensures that the highest priority task that is ready to execute is given control, which allows the system to be responsive to its external world. Dynamic reprioritization of tasks is supported for the most sophisticated of multitasking systems.

Real-Time Speed—Because BLMX-16 was hand-coded in assembly language, several advantages with regard to speed are gained. Task swapping, channel and message management, and I/O interfacing are executed more quickly than could be expected of a system written in a higher level language.

Direct Interrupt Processing—The BLMX-16 architecture employs interrupt channels which allow device-specific interrupt handling routines to interface directly with the interrupt source. This accomplishes servicing of interrupts without the overhead of task swapping, yet allows the operating system to maintain the integrity of the system. Combining this interrupt service architecture with a device-efficient nucleus results in an operating system that better supports demanding, real-time applications.

User Configurability—BLMX-16 executive-based applications may be configured from a wide range of facilities, selecting only those that meet the specific requirements of the application system. The resultant system contains only the modules necessary for its use, allowing the BLMX-16 executive to fit a wide range of applications from small, special-purpose, dedicated applications to large, general-purpose systems.

Event Driven—In the BLMX-16 executive, each user task exists in its own "closed environment"—a virtual processor. Each virtual processor can synchronize with external/internal occurrences through events. BLMX-16 supports a wide variety of events, including synchronization with task activities, external device operations, and the real-time clock.

Memory Pool Manager—The BLMX-16 executive has an integral memory pool manager. This feature not only reduces the amount of RAM required in an application system (potentially reduces board count), but also allows active modules more buffer area within any given space constraint.

Internal Structure

The kernel may be viewed as composed of a set of functions. These functions are:

1. Nucleus—performs task and channel management and controls executing memory.
2. Timer Manager—performs time-dependent control.
3. Dynamic Task Dispatcher—performs dynamic creation and installation of tasks at run-time.
4. Dynamic Channel Controller—performs dynamic creation and installation of software and interrupt channels at run-time.
5. Memory Pool Manager—performs memory allocation and deallocation for more than 64K of RAM.

The Timer Manager, Dynamic Task Dispatcher, Dynamic Channel Controller, and the Memory Pool Manager all operate under direction of the Executive Nucleus function, which assigns tasks to run on the hardware CPU.

System Functions

The BLMX-16 kernel controls CPU allocation by resolving conflicting needs of individual tasks, and monitors external events. The Event Manager, Task Manager, Channel Manager, Memory Manager, and Timer Manager provide system facilities that are directly accessible from the user task level. These system functions are summarized below:

•Task and Event Management

1. TSKBD —Build a task and schedule it to run.
2. SUSPD —Suspend a task.
3. GTPRI —Get task priority.
4. STPRI —Change run-time task priority.
5. WAITE —Wait for an event or combination of events to occur before resuming task processing.
6. TSTEV —Test the current state of an event.

•Intertask Communication

1. RECV(W) —Receive data from a channel and, optionally, wait for an event to occur.
2. SEND(W) —Send a message to a channel and, optionally, wait for an event to occur.
3. SIGNL —Synchronize with another task through event flags; signal completion.
4. BLDSC —Build software channel.

•Interrupt Handling

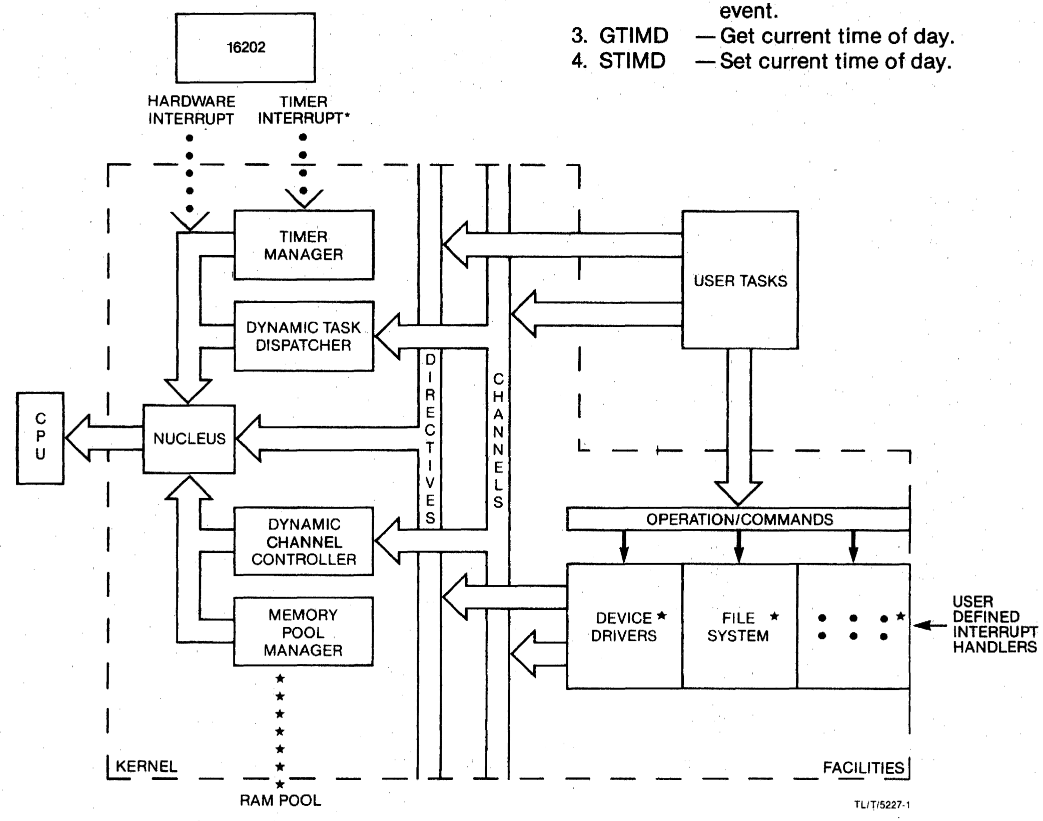
1. LINK(W) —Link a message control block to interrupt a channel and, optionally, wait for an event to occur.
2. INTSV —Interrupt save via executive service.
3. INTEX —Interrupt exit from executive.
4. BLDIC —Build an interrupt channel.
5. CANIO —Cancel I/O in progress.

•Memory Pool Management

1. ALLOC —Allocate a block of pooled memory.
2. DALCT —Deallocate memory back to pool.

•Timer Management

1. MRKT(W)—Mark a time delay and, optionally, wait for an event to occur.
2. CMRKT —Cancel previously posted mark-time event.
3. GTIMD —Get current time of day.
4. STIMD —Set current time of day.



*May or may not be from 16202 ICU.

FIGURE 1. BLMX-16 Structure

BLMX-16 Executive Shipping Configuration

One single-sided, double-density diskette *or* one 1600 bpi magnetic tape containing the following:

- BLMX-16 Kernel
- A Demo Package

Prerequisites

NSX-16 or NS-ASM-16 for VAX11 Package

SFW-90-A010 for STARPLEX II™ Package

Order Information

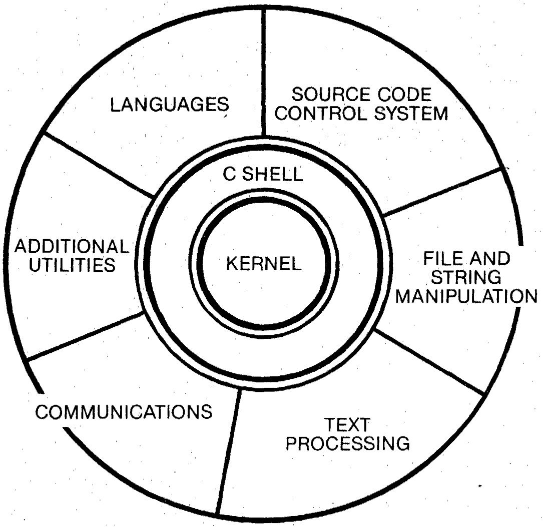
BLMX-16V†	VAX11 Package 1600 bpi magnetic tape (9-track VMS copy format) and reference manual
BLMX-16S†	STARPLEX II Package Single-sided, double-density flexible diskette and reference manual

Documentation

BLMX-16M	BLMX-16 System reference manual

†Software license agreement must be signed prior to order entry.

GENIX™ Operating System



- **Implementation of Berkeley 4.1 bsd UNIX™ for the NS16032**
- **Supports demand-paged virtual memory**
- **Each process runs in a protected linear address space of up to 16 Mbytes**
- **Optimal use of NS16032 architectural features**
- **C compiler**
- **Optional Pascal compiler**
- **Assembler, loader, run-time support library**
- **ddt symbolic assembly level debugger**

General Description

GENIX is a fully Bell licensed implementation of the Berkeley 4.1 bsd UNIX system for the NS16032 microprocessor family. It is available in binary form as the operating system for the SYS16™ development system. GENIX is also available in source form and can be user-modified to operate on NS16032-based target systems that include memory management and disc storage facilities.

GENIX, a multitasking, multiuser operating system, offers an advanced software development environment.

UNIX was originally designed for use on large minicomputer systems. The National Semiconductor GENIX implementation melds the powerful features of UNIX with the mainframe-like architectural features of the NS16032 microprocessor family.

Additional GENIX Features

- Demand-paged virtual memory. Each user has available a full 16 Mbyte virtual address space enabling the execution of programs with large processor main storage requirements.
- Hierarchical file system with 1 Kbyte blocking which results in enhanced throughput for disk I/O intensive applications.
- Standard 4.1 bsd UNIX utilities
  - — vi screen editor
  - — C shell with job control facilities
  - — termcap and cursor control package for support of screen-oriented terminal independent software
  - — nroff documentation preparation facility with popular macro packages
- source code control system (sccs)
- uucp inter-system communcations support program
- High-performance IEEE compatible floating-point instruction set option

Minimum Hardware Requirements

NS16032 32-bit advanced architecture microprocessor

NS16082 demand-paged Memory Management Unit

NS16201 Timing Control Unit

512 Kbyte processor main storage

20 Mbyte disk storage

RS232 full-duplex asynchronous communications port; one minimum, two preferred.

Optional Hardware Supported

NS16081 high-performance IEEE compatible Floating-Point Unit

NS16202 Interrupt Control Unit

Additional RS232 full-duplex asynchronous, communications ports

Streaming tape unit

Centronics-compatible printer

Processor main storage sizes from 512 Kbytes to 16 Mbytes

Sample I/O Drivers

Console driver

Disk driver

Archive streaming tape driver

Multiport asynchronous RS232 communications driver

Drivers will need to be modified for specific end-equipment characteristics and users may add additional drivers to support end-equipment features.

Support Category

Support may be purchased under contract from National Semiconductor Corporation. Support available includes error reporting and logging, maintenance and feature updates, and telephone assistance.

Licensing

This software is provided under license from National Semiconductor Corporation. A valid Western Electric UNIX System III or System V source license is a prerequisite for the National Semiconductor Corporation source license. Licensing provisions include binary distribution rights, source license fees and per-unit royalties.

National Semiconductor Corporation
2900 Semiconductor Drive
Santa Clara, California 95051
Tel: (408) 721-5000
TWX: (910) 339-9240

Electronica NSC de Mexico SA
Hegel No. 153-204
Mexico 5 D.F. Mexico
Tel: (905) 531-1689, 531-0569,
531-8204
Telex: 017-73550

NS Electronics Do Brasil
Avda Brigadeiro Faria Lima 830
8 Andar
01452 Sao Paulo, Brasil
Telex: 1121008 CABINE SAO PAULO
113193 INSBR BR

National Semiconductor GmbH
Furstenriederstraße Nr. 5
D-8000 München 21
West Germany
Tel.: (089) 5 60 12-0
Telex: 522772

National Semiconductor (UK), Ltd.
301 Harpur Centre
Horne Lane
Bedford MK40 1TR
United Kingdom
Tel: 0234-47147
Telex: 826 209

National Semiconductor Benelux
Ave. Charles Quint 545
B-1080 Bruxelles
Belgium
Tel: (02) 4661807
Telex: 61007

National Semiconductor (UK), Ltd.
1, Bianco Lunos Allè
DK-1868 Copenhagen V
Denmark
Tel: (01) 213211
Telex: 15179

National Semiconductor
Expansion 10000
28, Rue de la Redoute
F-92 260 Fontenay-aux-Roses
France
Tel: (01) 660-8140
Telex: 250956

National Semiconductor S.p.A.
Via Solferino 19
20121 Milano
Italy
Tel: (02) 345-2046/7/8/9
Telex: 332835

National Semiconductor AB
Box 2016
Stensätravägen 4/11 TR
S-12702 Skärholmen
Sweden
Tel: (08) 970190
Telex: 10731

National Semiconductor
Calle Nunez Morgado 9
(Esc. Dcha. 1-A)
E-Madrid 16
Spain
Tel: (01) 733-2954/733-2958
Telex: 46133

National Semiconductor Switzerland
Alte Winterthurerstrasse 53
Postfach 567
CH-8304 Wallisellen-Zürich
Tel: (01) 830-2727
Telex: 59000

National Semiconductor
Pasilanraitio 6C
SF-00240 Helsinki 24
Finland
Tel: (90) 14 03 44
Telex: 124854

NS Japan K.K.
POB 4152 Shinjuku Center Building
1-25-1 Nishishinjuku, Shinjuku-ku
Tokyo 160, Japan
Tel: (03) 349-0811
TWX: 232-2015 NSCJ-J

National Semiconductor Hong Kong, Ltd.
1st Floor,
Cheung Kong Electronic Bldg.
4 Hing Yip Street
Kwun Tong
Kowloon, Hong Kong
Tel: 3-899235
Telex: 43866 NSEHK HX
Cable: NATSEMI HX

NS Electronics Pty. Ltd.
Cnr. Stud Rd. & Mtn. Highway
Bayswater, Victoria 3153
Australia
Tel: 03-729-6333
Telex: AA32096

National Semiconductor PTE, Ltd.
10th Floor
Pub Building, Devonshire Wing
Somerset Road
Singapore 0923
Tel: 652 700047
Telex: NATSEMI RS 21402

National Semiconductor Far East, Ltd.
Taiwan Branch
P.O. Box 68-332 Taipei
3rd Floor, Apollo Bldg.
No. 218-7 Chung Hsiao E. Rd.
Sec. 4 Taipei Taiwan R.O.C.
Tel: 7310393-4, 7310465-6
Telex: 22837 NSTW
Cable: NSTW TAIPEI

National Semiconductor (HK) Ltd.
Korea Liaison Office
6th Floor, Kunwon Bldg.
No. 2, 1-GA Mookjung-Dong
Choong-Ku, Seoul, Korea
C.P.O. Box 7941 Seoul
Tel: 267-9473
Telex: K24942

DA-RRD35M93/Printed in U.S.A.