A Texas Instruments Application Report

TMS 9900 Family System Development Manual

PRINTED IN U.S.A.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Concluded)

Section **Page**

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF ILLUSTRATIONS (Concluded)

Figure No. Page

LIST OF TABLES

SECTION I

INTRODUCTION

This manual describes the general operation of the TMS 9900 family of devices, interfacing details, and examples of several hardware configurations. The TMS 9900 family are microprocessors and peripheral support devices, produced using N-channel silicon-gate MOS technology and large-scale integration to provide a high degree of functional density with minimum component count and board area requirements.

The TMS 9900 and the TMS 9980A/81 microprocessors are single-chip 16-bit central-processing units (CPU's) with the word size, instruction set, and addressing capabilities normally associated with full minicomputers. Both single-chip CPU's are easily and economically implemented in small or large systems because of their simple, yet flexible, interfacing and architecture. The TMS 9900 utilizes a 15-bit address bus and a 16-bit data bus to address 32K words of memory space. Individual byte addressing is accomplished by means of a sixteenth address bit maintained internally within the TMS 9900 CPU. The TMS 9980A/81 possesses the same versatility as the TMS 9900, but is implemented with a 14-bit address bus and an 8-bit data bus to directly address 16K bytes of memory space. This combination of CPU's gives the systems designer wide flexibility in the development of microprocessor systems.

Two members of the growing family of support devices for the TMS 9900 and TMS 9980A/81 are the TMS 9901 programmable systems interface and the TMS 9902 asynchronous communications controller. The TMS 9901 provides six dedicated interrupts, seven dedicated I/O ports, nine programmable I/O or interrupt ports, and an interval timer, and is easily stacked to provide additional interrupt and I/O capabilities.

The TMS 9902 provides complete control of an asynchronous-communications channel to include baud rate generation, timing, and data serialization and deserialization. The TMS 9902 also provides an interval timer and all necessary modem control signals.

Texas Instruments continues to introduce additional support circuits and development tools for the TMS 9900 family. One new support circuit, soon to be introduced, is the TMS 9903 synchronous communications controller, which will provide all hardware control of a synchronous communications channel, as well as assuming much of the control overhead from the CPU.

In the following discussions the reader is assumed to be familiar with the TMS 9900 family architecture and instruction set. Specific points not addressed here can be resolved by reference to the following publications, which contain detailed descriptions and specifications of device timing, signal sequences, interface requirements, and instruction sets of the TMS 9900 family.

- TMS 9900 Microprocessor Data Manual

- TMS 9980A/81 Microprocessor Data Manual
- TMS 9901 Programmable Systems Interface Data Manual
- TMS 9902 Asynchronous Communication Controller Data Manual
- 990 Computer Family System Handbook
- TMS 9900 Microprocessor Assembly Language Programmers' Guide

The following sections discuss the TMS 9900 family of microprocessors from the point of view of the TMS 9900. With certain exceptions this discussion also describes the operation of the TMS 9980A/81. Those features specific to the operation of the TMS 9980A/81 are discussed in detail in Section 8.

The TMS 9900 microprocessor is illustrated in Figure 1-1.

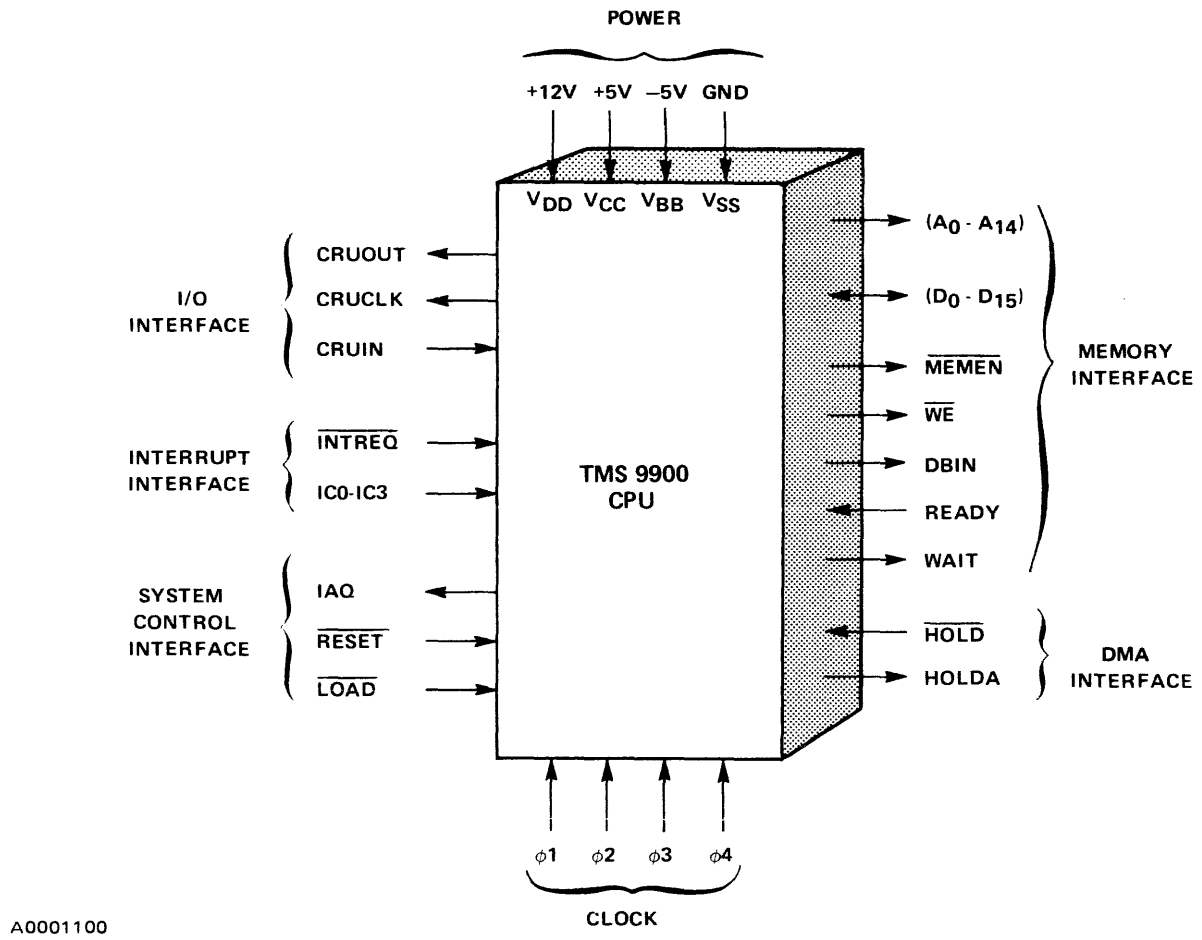


Figure 1-1. TMS 9900 Microprocessor

SECTION II

TMS 9900 ARCHITECTURE

The TMS 9900 has an advanced memory-to-memory architecture. The advantages of this architecture are best illustrated by comparison to other microprocessors currently available.

2.1 Registers

As shown in Figure 2-1, most microprocessors contain a set of registers internal to the device. This places restrictions on the number and size of internal registers due to limitations of LSI device size and density. These internal registers are used to contain high usage data because they may be accessed more efficiently and used more flexibly than memory. However, when more register storage is required than is available, register contents must be temporarily saved in memory. When a processor changes from one operation to another it is often desirable to save all internal register contents in memory so that the program controlling the new function may use all of the internal registers. Then, upon return to the original function, the registers are reloaded from memory.

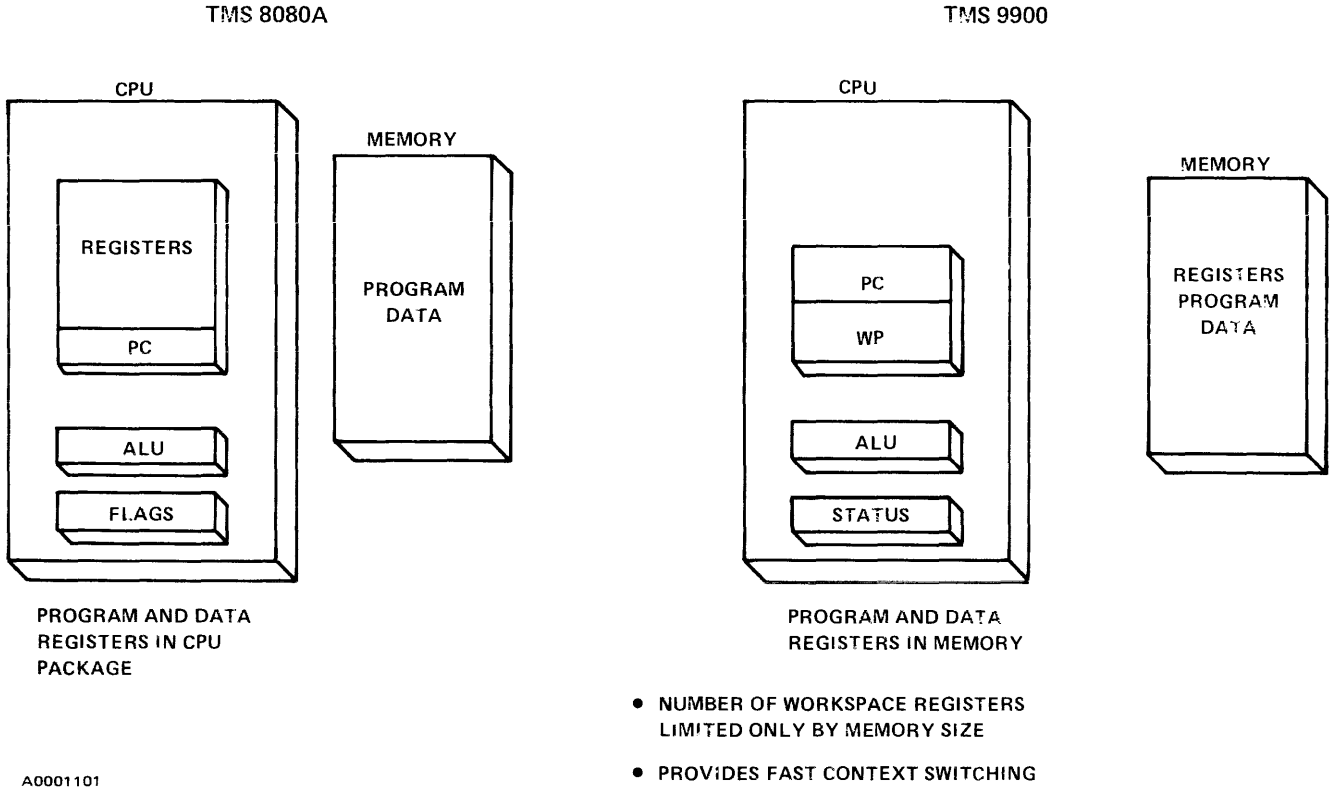


Figure 2-1. TMS 9900 Architecture

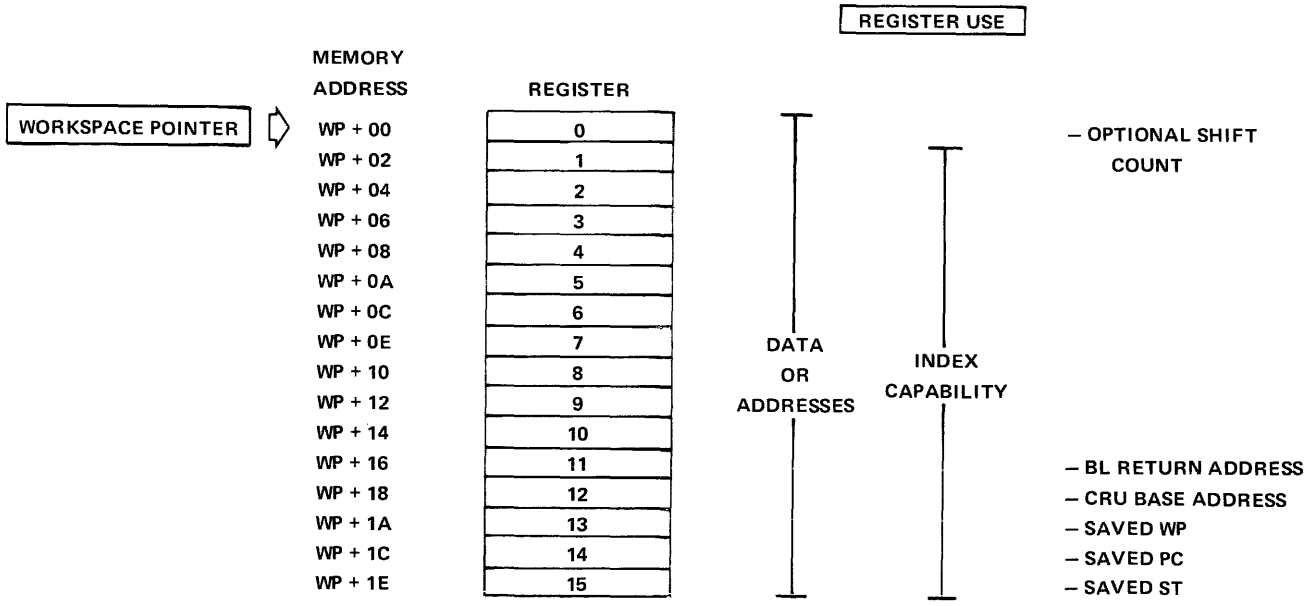
2.1.1 Workspace Pointer

The high usage data registers for the TMS 9900 are defined as blocks of memory called workspaces. The location of a workspace in memory is defined by a single internal register called the workspace pointer. The workspace pointer contains the memory word address of the first of sixteen consecutive memory words in the workspace, thus the processor has access to sixteen 16-bit registers. When a different set of registers is required, the program simply reloads the workspace pointer with the address of the new workspace, resulting in a significant reduction in processor overhead when a new set of registers is required. Also, the number of workspace registers is limited only by the amount of memory in the system.

2.1.2 Workspace Registers

The uses of the workspace registers are shown in Table 2-1. All 16 (WR0–WR15) may be used for storage of addresses, temporary data, and accumulated results. WR1–WR15 may be used as index registers to specify a bias from a fixed-memory location to select an instruction operand. Register 0 may contain the number of bit positions an operand is shifted by the shift instructions (SLA, SRA, SRC, and SRL). WR11 will contain the return address when the branch and link (BL) instruction is executed. Bits 3–14 of WR12 contain the CRU base address for CRU instructions. WR13–WR15 will contain the internal register values which are reloaded when the return to workspace (RTWP) instruction is executed.

TABLE 2-1. TMS 9900 WORKSPACE REGISTERS



2.1.3 Program Counter

The function of the program counter of the TMS 9900 is identical to that of the program counter in other microprocessors in that the PC contains the memory address of the next instruction to be executed. As each instruction is executed, the PC is automatically updated. Except when instructions or operations are performed which directly affect the PC, it is simply incremented to the next consecutive memory-word address as each instruction is executed.

2.1.4 Status Register

The status register of the TMS 9900 is similar to that of other microprocessors in that it contains flag bits which indicate results of the most recent arithmetic or logical operation performed. Additionally, the TMS 9900 contains a 4-bit interrupt mask in ST12–ST15 which defines the lowest-priority level interrupt which will be recognized by the microprocessor.

2.2 Memory-to-Memory Operations

The TMS 9900 instructions are not limited to workspace registers for efficient data storage. The two-address instructions permit operations to be performed on any word in memory by any other word in memory. For example, a single add (A) instruction can add any memory word to any other memory word in the total 32K word memory space. Many microprocessor architectures require a series of instructions in which memory contents are moved to registers, added, and returned to memory. The TMS 9900 architecture minimizes program storage requirements by reducing the number of instructions and also reduces program complexity and documentation requirements, resulting in both lower memory cost and lower development cost for a TMS 9900 system.

2.3 Bus Structures

The TMS 9900 has separate memory, I/O, and interrupt bus structures as shown in Figure 2-2. Each bus structure is optimized for its individual function. The memory bus efficiently handles the uniform width memory words which are transferred in parallel between the CPU and memory. The memory reference instructions operate on parallel data words or bytes, with additional masking instructions to isolate individual bits. The TMS 9900 has separate address, control, and data outputs and uses standard memories without external address latches. The TMS 9900 64-pin package is economical since the elimination of control or address multiplexing eliminates several external circuits and simplifies circuit layout. A complete discussion of memory interface to the TMS 9900 is contained in Section III.

TMS 9900 I/O uses the communication register unit (CRU) concept. Individual or multiple bits are transferred serially between the CPU and peripheral devices, thus the I/O instructions operate on individual bits and on variable length fields. While the TMS 9900 can also use memory mapped I/O, the CRU bus is normally more efficient and economical for I/O operations. Operation, interface, and applications examples for the CRU are contained in Section IV.

The TMS 9900 also has a separate interrupt bus structure that provides multiple vectored and prioritized interrupts without complex external vector generation or slow software polling procedures. Interrupt processing by the TMS 9900 is described in Section IV.

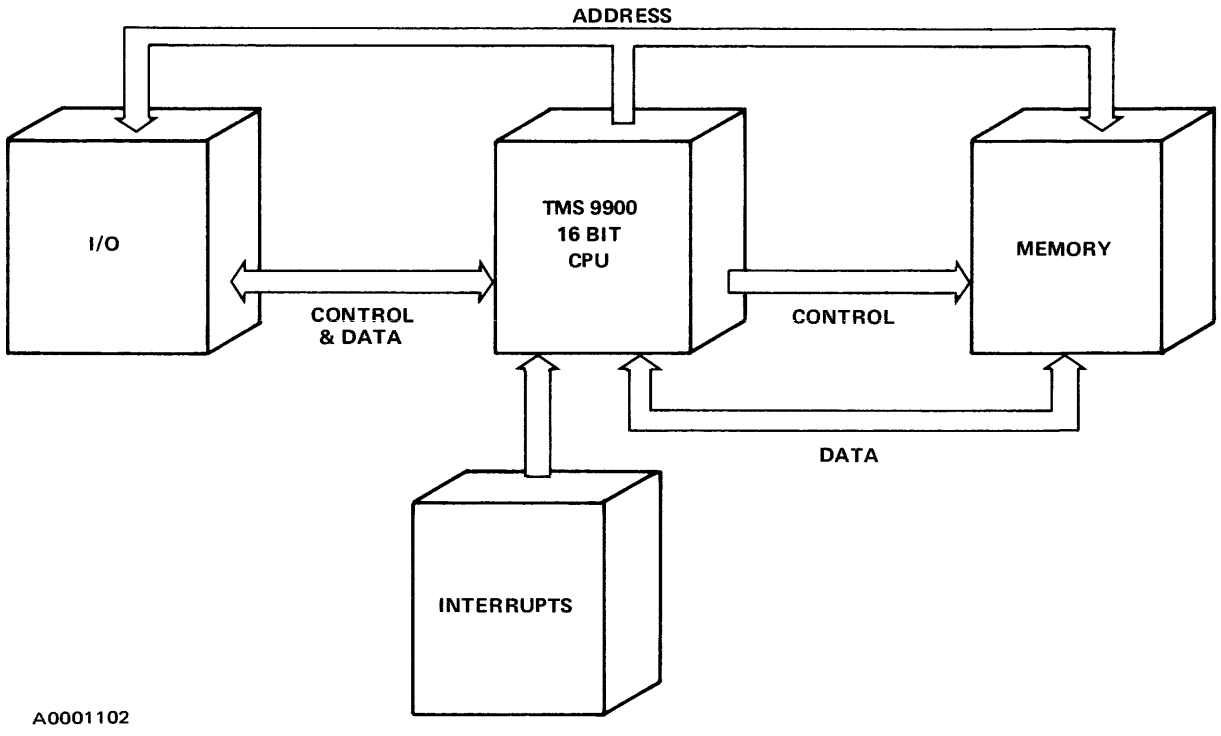


Figure 2-2. TMS 9900 System Bus Structure

2.4 Context Switching

When the TMS 9900 processes an interrupt, $\overline{\text{LOAD}}$, or $\overline{\text{RESET}}$, or executes an extended operation (XOP) or branch and load workspace pointer (BLWP) instruction, an operation called a context switch is automatically performed. The purpose of the context switch is twofold. First, the present internal registers (WP, PC, and ST) are stored in memory; second, new values for the WP and PC are loaded, thus setting up a different workspace and starting point for program execution.

The new values for the WP and PC are contained in dedicated memory locations which are described in Section 3.1. The BLWP instruction operand identifies the memory addresses which contain the new WP and PC values. The actual sequence in which the context switch is performed is as follows:

- The new WP is loaded into the CPU;
- The old ST is stored in WR15 of the new workspace;
- The old PC is stored in WR14 of the new workspace;
- The old WP is stored in WR13 of the new workspace;
- The new PC is loaded into the CPU and instruction execution begins at that point.

When the new program has completed execution, return to the original context is performed by execution of the return workspace pointer (RTWP) instruction, which causes the values contained in WR13–WR14 to be reloaded into the internal registers.

2.5 Machine Cycles

Each operation performed by the TMS 9900 consists of a sequence of machine cycles. In each machine cycle the processor performs a data transfer with memory or the CRU and/or an arithmetic or logical operation internally with the ALU. A detailed discussion of the machine cycles for all instructions is contained in Appendix A.

2.5.1 ALU Machine Cycles

Each ALU machine cycle is two clock cycles long. In an ALU cycle no external data transfer occurs, but the ALU performs an arithmetic or logical operation on two operands contained internally.

2.5.2 Memory Read Machine Cycles

The function of the memory-read cycle is to transfer a word of data contained in memory to the processor. An ALU operation may be performed during a memory-read cycle. Memory-read cycles are a minimum of two clock cycles long. If wait states are inserted to allow access to slow memories, the length of the memory-read cycle is extended.

2.5.3 Memory Write Machine Cycles

The memory-write cycle is identical to the memory-read cycle, except that data is written to rather than read from memory.

2.5.4 CRU Output Machine Cycles

Each CRU output machine cycle is two clock cyles long. In addition to outputting a bit of CRU data, an ALU operation may also be performed internally.

2.5.5 CRU Input Machine Cycles

The CRU input cycle is identical to the CRU output cycle, except that one bit of data is input rather than output.

2.5.6 Instruction Execution Examples

Examples of how sequences of machine cycles are used to execute instructions are illustrated in Figures 2-3 to 2-6. Note that the first machine cycle of each instruction is always a memory-read cycle in which the instruction is fetched, and the second is always an ALU cycle.

2.6 Machine Cycle Limits

Table 2-2 lists information which will be useful for system design. The maximum number of consecutive memory-read cycles is used to calculate the maximum latency for the TMS 9900 to enter the hold state, since the hold state is only entered from ALU, CRU input, or CRU output machine cycles. The minimum frequency of consecutive memory/non-memory cycle sequences occurs when the DIV instruction is executed. This number is used to ensure that the refresh rate meets specifications when the transparent-refresh mode described in paragraph 3.4.2.3 is used, since memory is refreshed in this mode each time an ALU or CRU cycle follows a memory cycle. Figure 2-7 shows the logic to generate a pulse for each memory access cycle, including consecutive cycles shown in Figure 2-8.

SBO OUTBIT

CYCLE	TYPE	FUNCTION
1	Memory Read	Instruction Fetch
2	ALU	Decode Op Code
3	ALU	Calculate Address of WR12
4	Memory Read	Fetch (WR12)
5	ALU	Calculate CRU Address
6	CRU Output	Output Bit, Increment PC

A0001103

Figure 2-3. SBO Instruction Machine Cycles

STCR RO,5

CYCLE	TYPE	FUNCTION
1	Memory Read	Instruction Fetch
2	ALU	Decode Op Code
3	Memory Read	Fetch (WRO)
4	ALU	Calculate Address of WR12
5	Memory Read	Fetch (WR12)
6	ALU	Set Up
7	ALU	Set Up
8-12	CRU Input	Transfer 5 Bits
13	ALU	Set Up
14	ALU	Set Up
15-17	ALU	Zero Filling
18	ALU	Parity Generation
19	ALU	Load WRO Address in MA
20	ALU	Byte Positioning
21	Memory Write	Store Data in WRO, Increment PC

A0001104

Figure 2-4. STCR Instruction Machine Cycles

RTWP

CYCLE	TYPE	FUNCTION
1	Memory Read	Instruction Fetch
2	ALU	Decode Opcode
3	ALU	Calculate Address of WR15
4	Memory Read	Restore Status from WR15
5	Memory Read	Restore PC from WR14
6	Memory Read	Restore WP from WR13
7	ALU	Load PC into MA

A0001105

Figure 2-5. RTWP Instruction Machine Cycles

A *R1,R2

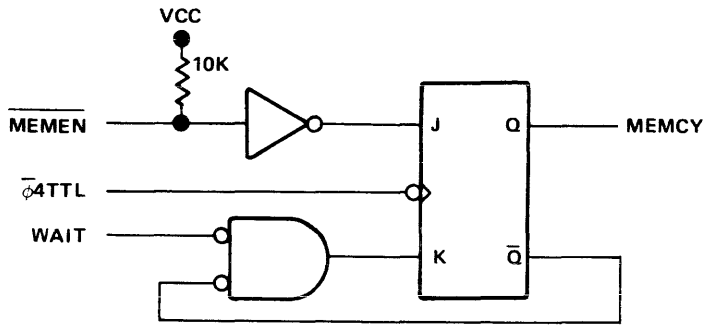
CYCLE	TYPE	FUNCTION
1	Memory Read	Instruction Fetch
2	ALU	Decode Opcode
3	Memory Read	Fetch (WR1)
4	ALU	Set Up
5	Memory Read	Fetch ((WR1))
6	ALU	Set Up
7	Memory Read	Fetch (WR2)
8	ALU	Addition
9	Memory Write	Store Result in WR2, Increment PC

A0001106

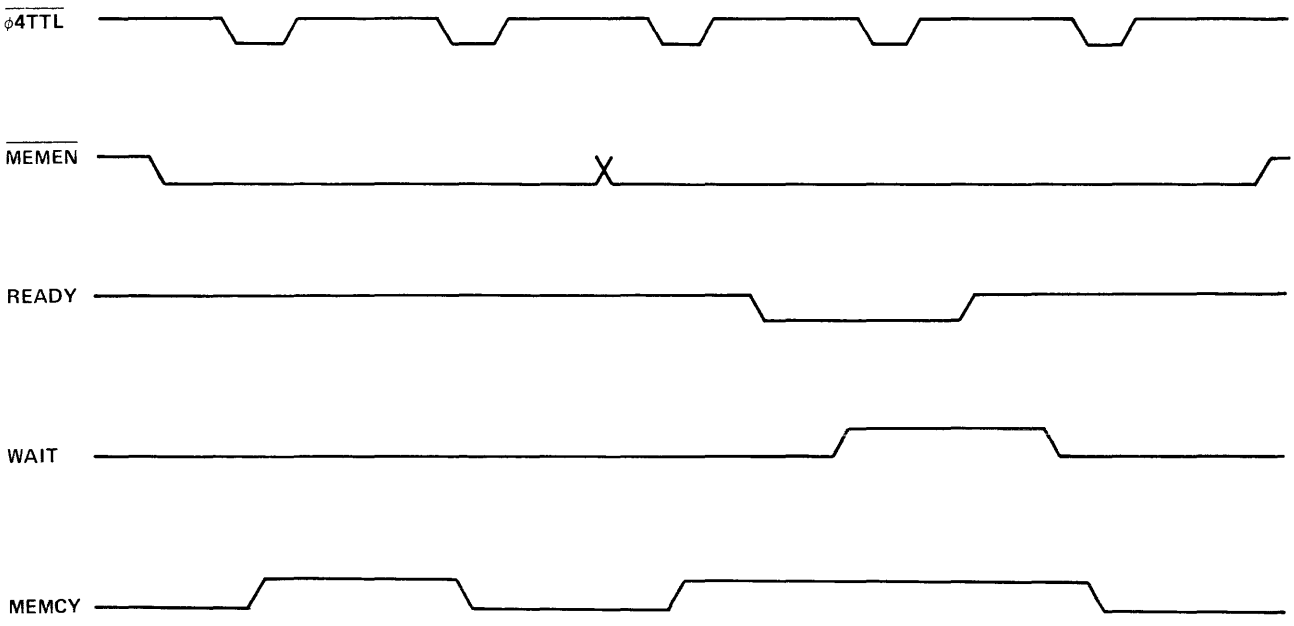
Figure 2-6. A Instruction Machine Cycles

TABLE 2-2. MACHINE CYCLE LIMITS

	MINIMUM	MAXIMUM
Consecutive Memory Read Cycles	1	3
Consecutive Memory Write Cycles	1	1
Consecutive ALU Cycles	1	51
Consecutive CRU Cycles	1	16
Frequency of Consecutive memory/non-memory cycle pairs (used for transparent refresh)	5 pairs (64 machine cycles during DIV.)	



A0001107 Figure 2-7. Memory Cycle Pulse Generation



A0001187

Figure 2-8. Memory Cycle Pulse Timing

SECTION III

MEMORY

The TMS 9900 workspace organization uses the system memory for register files in addition to program and data storage. The TMS 9900 is easily interfaced to any of the standard types of semiconductor memory devices. Texas Instruments provides masked ROMs, field-programmable ROMs (PROMs), and erasable PROMs (EPROMs) for non-volatile program and data storage. RAMs are available in sizes from a 64 x 8 static RAM to the 16K dynamic RAMs for use as temporary program and data storage. TMS 9900 compatible memory devices are listed in Table 3-1.

3.1 Memory Organization

The TMS 9900 instructions build a 16-bit address word which describes a 64K x 8 bit address space. The least-significant address bit is used internally by the CPU to select the even or odd byte, and the other 15 address bits are passed to external memory to describe a 32K x 16 bit address space. Although 32K words are the most locations that the CPU can directly access, additional memory can be accessed using address bank switches or address mapping.

The advanced architecture of the TMS 9900 uses part of the external memory space for workspace register files and for transfer vector storage. For maximum design flexibility, only the locations of the transfer vectors are restricted. A memory map for the TMS 9900 is shown in Figure 3-1. The transfer vectors are the new WP and PC for the context switches described in Section 2.4.

3.1.1 $\overline{RESET}$ Vector

The first two memory words are reserved for storage of the $\overline{\text{RESET}}$ vector. The $\overline{\text{RESET}}$ vector is used to load the new WP and PC whenever the CPU $\overline{\text{RESET}}$ signal occurs. The first word contains the new WP, which is the starting address of the $\overline{\text{RESET}}$ workspace. The second word contains the new PC, which is the starting address of the $\overline{\text{RESET}}$ service routine.

3.1.2 Interrupt Vectors

The next thirty memory words, 0004_{16} through $003E_{16}$ are reserved for storage of the interrupt transfer vectors for levels 1 through 15. Each interrupt level uses a word for the workspace pointer (WP) and a word for the starting address of the service routine (PC). If an interrupt level is not used within a system, then the corresponding two memory words can be used for program or data storage.

TABLE 3-1. TMS 9900
COMPATIBLE MEMORIES

Device	Organization	Package
Dynamic RAMs		
TMS 4030	4096X1	22
TMS 4050	4096X1	18
TMS 4051	4096X1	18
TMS 4060	4096X1	22
TMS 4070	16384X1	16
Static RAMs (NMOS)		
TMS 4033	1024X1	16
TMS 4036-2	64X8	20
TMS 4039-2	256X4	22
TMS 4042-2	256X4	18
TMS 4043-2	256X4	16
ROMs (NMOS)		
TMS 4700	1024X8	24
TMS 4800	2048X8	24
PROMs (TTL)		
SN 74S287	256X4	16
SN 74S471	256X8	20
SN 74S472	512X8	20
SN 74S474	512X8	24
EPROMs (NMOS)		
TMS 2708	1024X8	24
TMS 2716	2048X8	24

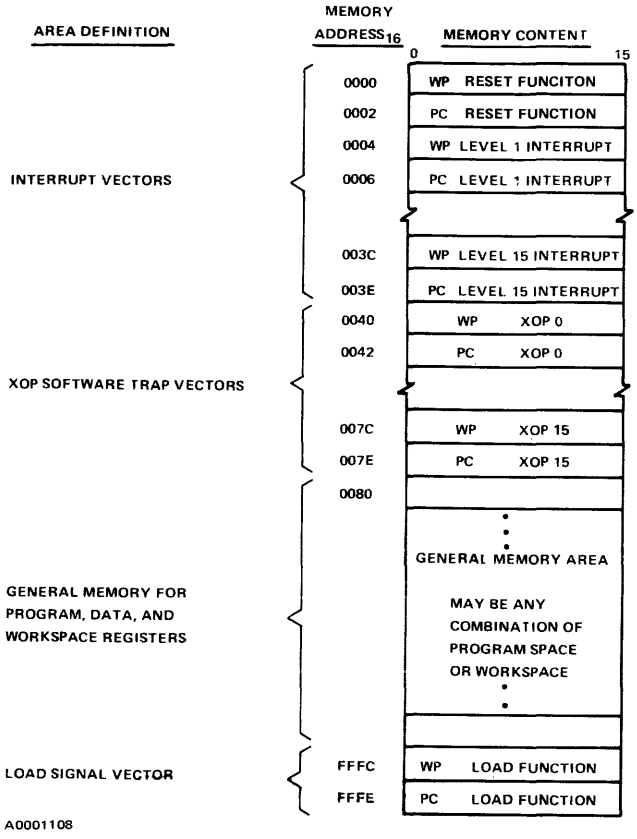


Figure 3-1. TMS 9900
Dedicated Memory Addresses

3.1.3 Software Trap Vectors

The next thirty-two memory words, 0004_{16} through $007E_{16}$, are used for extended-operation software trap vectors. When the CPU executes one of the 16 extended operations (XOPs), the program traps through the corresponding vector. Two words are reserved for each trap vector, with one word for the WP and one word for the PC. If an XOP instruction is not used, the corresponding vector words can be used for program or data storage.

3.1.4 LOAD Vector

The last two memory words $FFFC_{16}$ and $FFFE_{16}$ are reserved for the LOAD vector, with one word for the WP and one word for the PC. the LOAD vector is used whenever the CPU LOAD signal is active (low).

3.1.5 Transfer Vectors Storage

The transfer vectors can be stored either in ROM or RAM, but the reset vector should be in non-volatile memory to ensure proper system start-up. The restart routine should initialize any vector which is in RAM. The program can then manipulate the RAM-based vectors to alter workspace assignments or service routine entry points, while ROM-based vectors are fixed and cannot be altered.

3.2 Memory Control Signals

The TMS 9900 uses three signals to control the use of the data bus and address bus during memory read or write cycles. Memory enable ($\overline{\text{MEMEN}}$) is active (low) during all memory cycles.

Data bus in (DBIN) is active (high) during memory read cycles and indicates that the CPU has disabled the output data buffers.

Write enable ($\overline{\text{WE}}$) is active (low) during memory write cycles and has timing compatible with the read/write (R/W) control signal for many standard RAMs.

3.2.1 Memory Read Cycle

Figure 3-2 illustrates the timing for a memory read machine cycle with no wait states. At the beginning of the machine cycle, MEMEN and DBIN become active and the valid address is output on A0–A14. D0–D15 output drivers are disabled to avoid conflicts with input data. WE remains high for the entire machine cycle. The READY input is sampled on $\phi 1$ of clock cycle 1, and must be high if no wait states are desired. Data is sampled on $\phi 1$ of clock cycle 2, and set-up and hold timing requirements must be observed. A memory-read cycle is never followed by a memory-write cycle,and D0-D15 output drivers remain disabled for at least one additional clock cycle.

3.2.2 Memory Write Cycle

Figure 3-3 illustrates the timing for a memory write machine cycle with no wait states. $\overline{\text{MEMEN}}$ becomes active, and valid address and data are output at the beginning of the machine cycle. DBIN remains inactive for the complete cycle. $\overline{\text{WE}}$ goes low on $\phi 1$ of clock cycle 1 and goes high on $\phi 1$ of clock cycle 2, meeting the address and data set-up and hold timing requirements for the static RAMs listed in Table 3-1. For no wait states, READY must be high during $\phi 1$ of clock cycle 1.

3.2.3 Read/Write Control with DBIN

In some memory systems, particularly with dynamic RAMs, it may be desirable to have READ and WRITE control signals active during the full memory cycle. Figure 3-4 shows how the WRITE signal can be generated, since any memory cycle ($\overline{\text{MEMEN}} = 0$) in which DBIN = 0 is a memory-write cycle.

3.2.4 Slow Memory Control

Although most memories operate with the TMS 9900 at the full system speed, some memories cannot properly respond within the minimum access time determined by the system clock. The system clock could

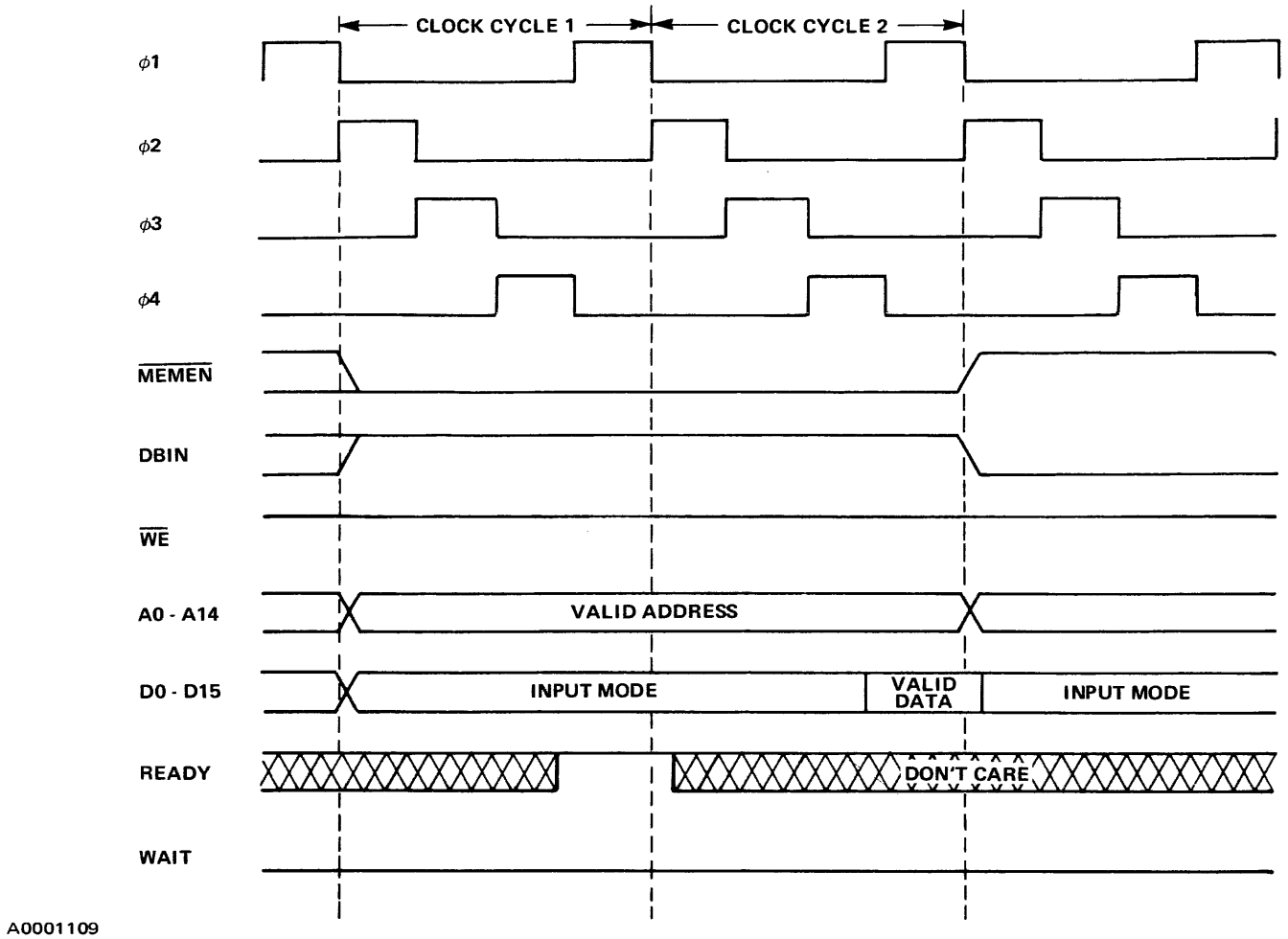


Figure 3-2. Memory-Read Cycle Timing

be slowed down in order to lengthen the access time but the system through-put would be adversely affected since non-memory and other memory reference cycles would be unnecessarily longer. The READY and WAIT signals are used instead to synchronize the CPU with slow memories. The timing for memory-read and memory-write cycles with wait states is shown in Figures 3-5 and 3-6.

The READY input is tested on $\phi 1$ of clock cycle 1 of memory-read and memory-write cycles. If READY = 1, no wait states are used and the data transfer is completed on the next clock cycle. If READY = 0, the processor enters the wait state on the next clock cycle and all memory control, address, and data signals maintain their current levels. The WAIT output goes high on $\phi 3$ to indicate that a wait state has been entered. While in the wait state, the processor continues to sample READY on $\phi 1$, and remains in the wait state until READY = 1. When READY = 1 the processor progresses to clock cycle 2 and the data transfer is completed. WAIT goes low on $\phi 3$. It is important to note that READY is only tested during $\phi 1$ of clock cycle 1 of memory-read and memory-write cycles and wait states, and the specified set-up and hold timing requirements must be met; at any other time the READY input may assume any value. The effect of inserting wait states into memory access cycles is to extend the minimum allowable access time by one clock period for each wait state.

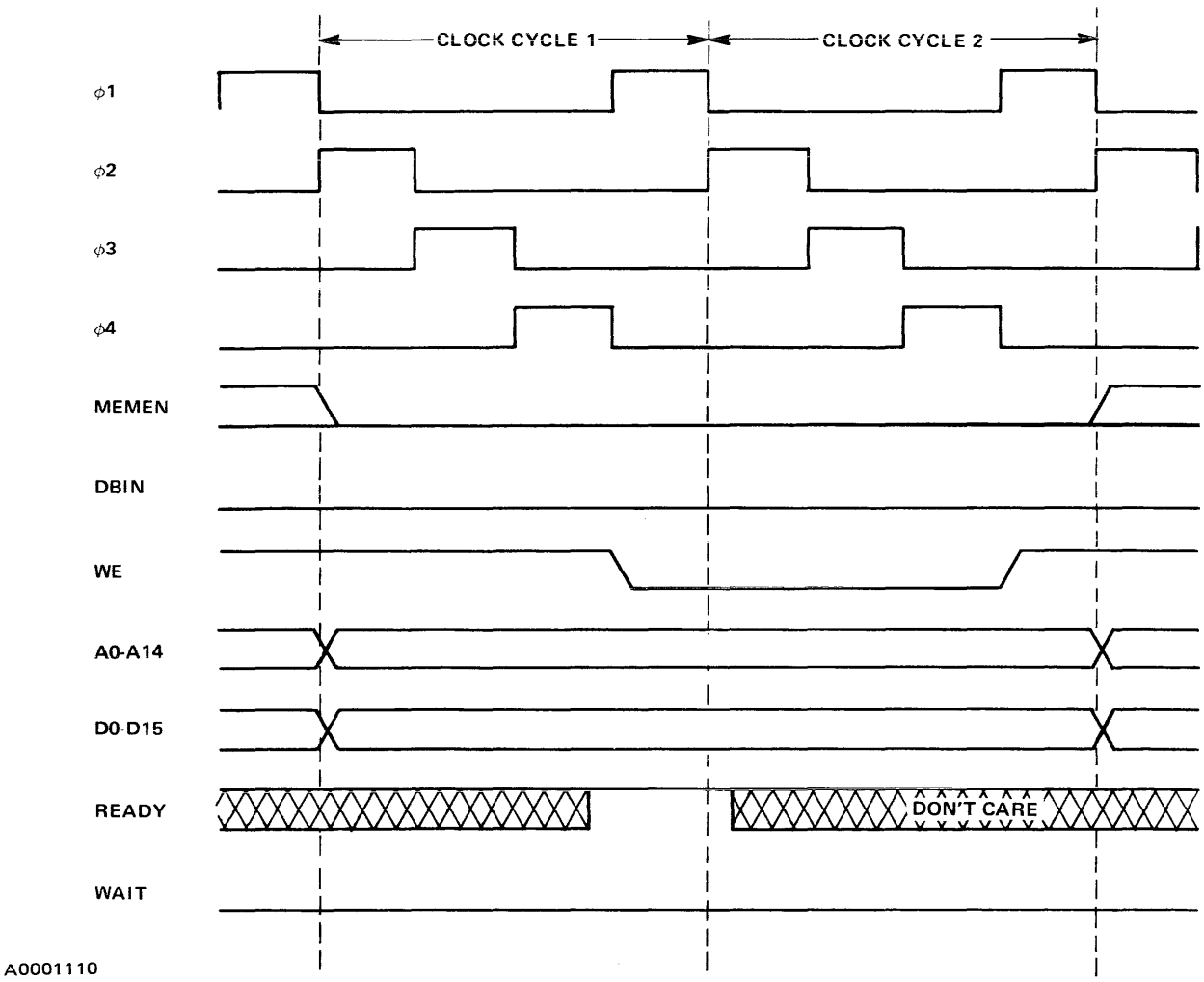


Figure 3-3. Memory-Write Cycle Timing

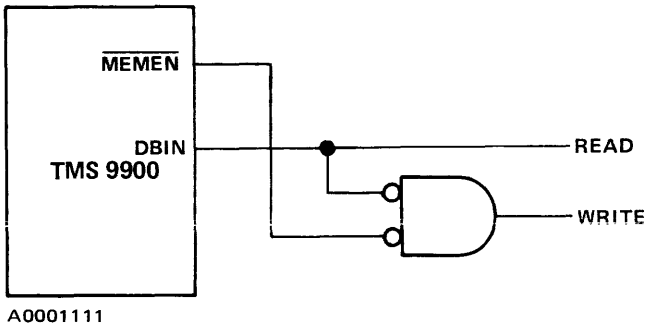


Figure 3-4. Read/Write Control Using $\overline{\text{MEMEN}}$ and DBIN

3.2.5 Wait State Control

Figure 3-7 illustrates the connection of the WAIT output to the READY input to generate one wait state for a selected memory segment. The address decode circuity generates an active low signal ($\overline{\text{SLOMEM}}$ = 0) whenever the slow memory is addressed. For example, if memory addresses 8000_{16}–$FFFE_{16}$ select slow memory, $\overline{\text{SLOMEM}} = \overline{\text{A0}}$. If one wait state is required for all memory, WAIT may be connected directly to READY, causing one wait state to be generated on each memory-read or memory-write machine cycle. Referring again to Figures 3-5 and 3-6, note that the WAIT output satisfies all of the timing

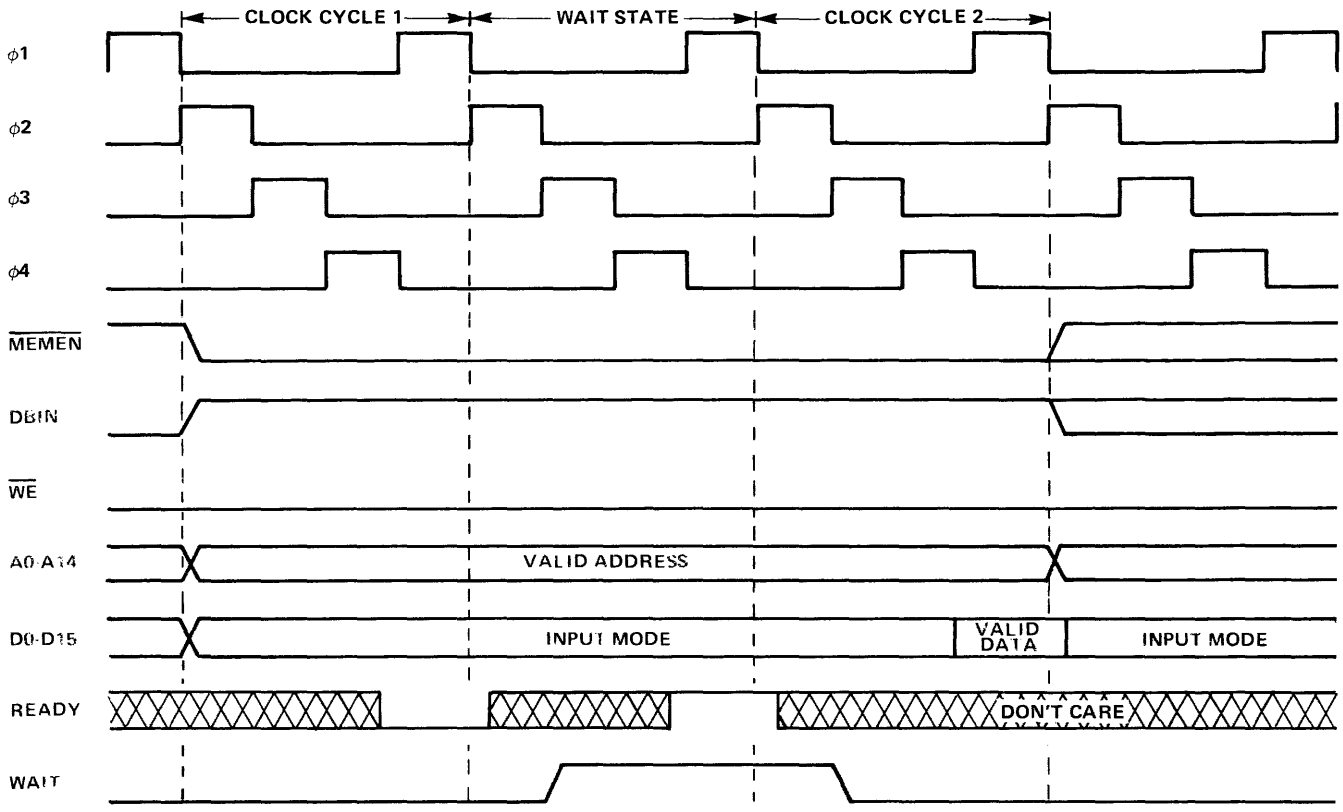


Figure 3-5. Memory-Read Cycle With One Wait State

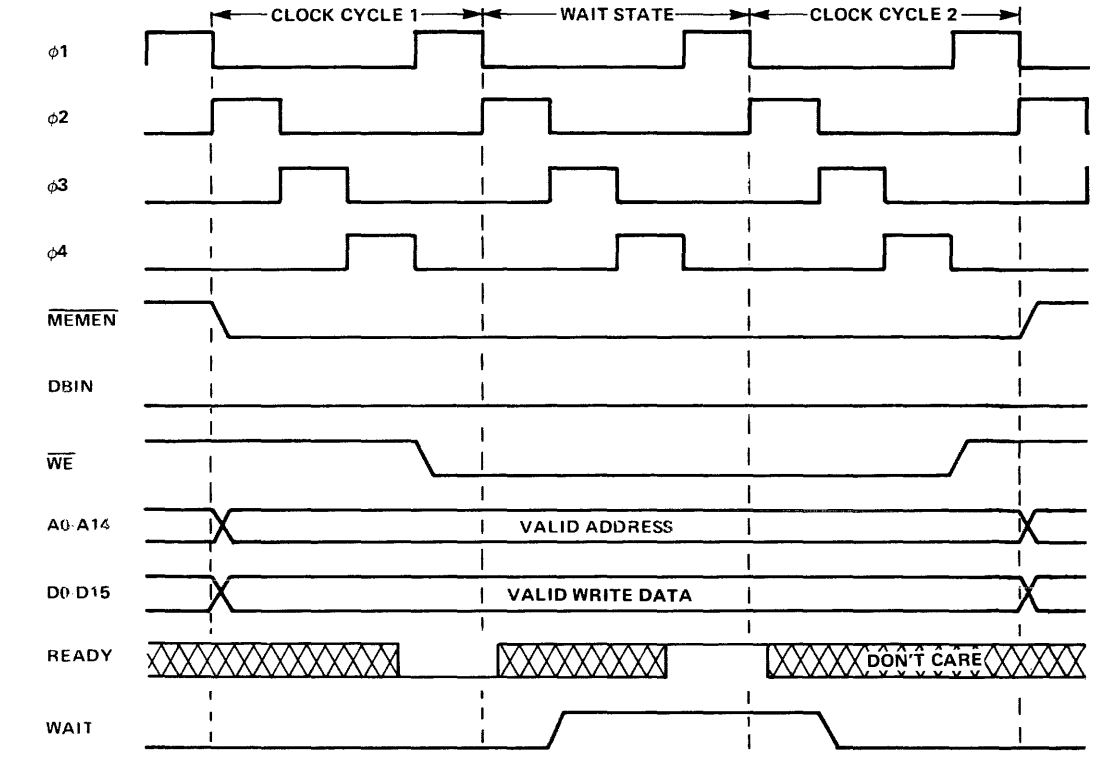


Figure 3-6. Memory-Write Cycle With One Wait State

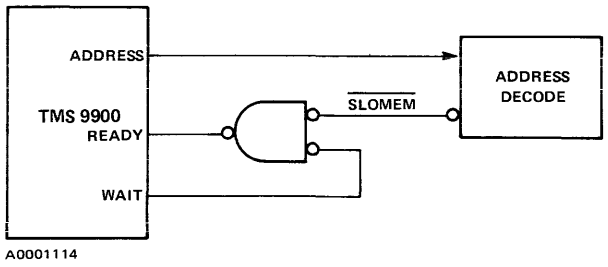


Figure 3-7. Single Wait State for Slow Memory

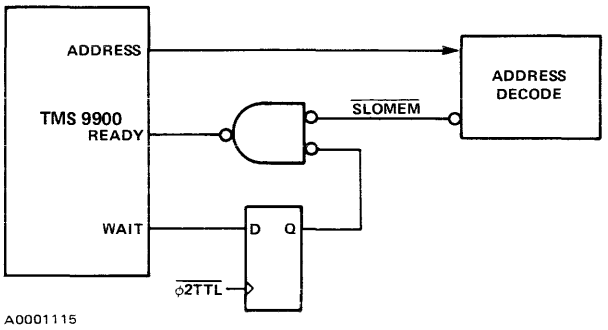


Figure 3-8. Double Wait States for Slow Memory

requirements for the READY input for a single wait state. The address decode signal is active only when a particular set of memory locations has been addressed. Figure 3-8 illustrates the generation of two wait states for selected memory by simply delaying propagation of the WAIT output to the READY input one clock cycle with a D-type flip-flop. The rising edge of ϕ2TTL is assumed to be coincident with the falling edge of the ϕ2 clock input to the TMS 9900.

3.2.6 Memory Access Time Calculation

Maximum allowable memory access time for the TMS 9900 can be determined with the aid of Figure 3-9. Memory control and address signals are output on ϕ2 of clock cycle 1, and are stable 20 ns (t_{PLH}, t_{PHL}) afterwards. Data from memory must be valid 40 ns (t_{su}) before the leading edge of ϕ1 during clock cycle 2. Therefore, memory access time may be expressed by the equation:

$$t_{acc} \leq (1.75 + n)\, t_{cy} - t_{PLH} - t_r - t_{su}$$

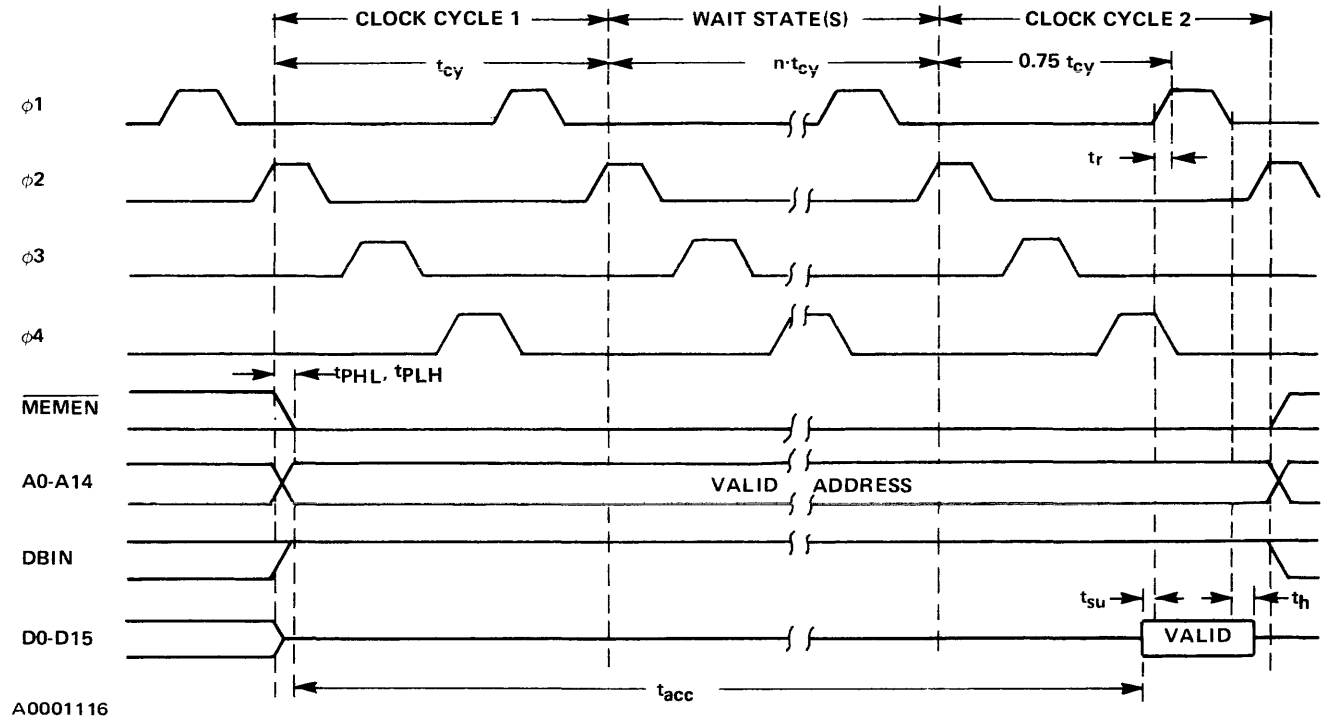


Figure 3-9. Memory Access Timing Calculation

where n equals the number of wait states in the memory-read cycle. Assigning worst-case specified values for t_{PLH} (20 ns), t_r (12 ns), and t_{su} (40 ns), and assuming 3 MHz operation:

$$t_{acc} \leqslant \frac{(1.75 + n)}{0.003} - 72 \text{ ns}$$

Access time is further reduced by address decoding, control signal gating, and address and data bus buffering, when used. The graph in Figure 3-10 illustrates the above equation for varying n and t_{cy} in an unbuffered memory system. Thus, for a known access time for a given device, the number of required wait states can be determined.

For example, a TMS 4042-2 RAM has a 450 nanosecond access time and does not require any wait states. A TMS 4042 has a 1000 nanosecond access time and requires two wait states. Propagation delays caused by address or data buffers should be added to the nominal device access time in order to determine the effective access time.

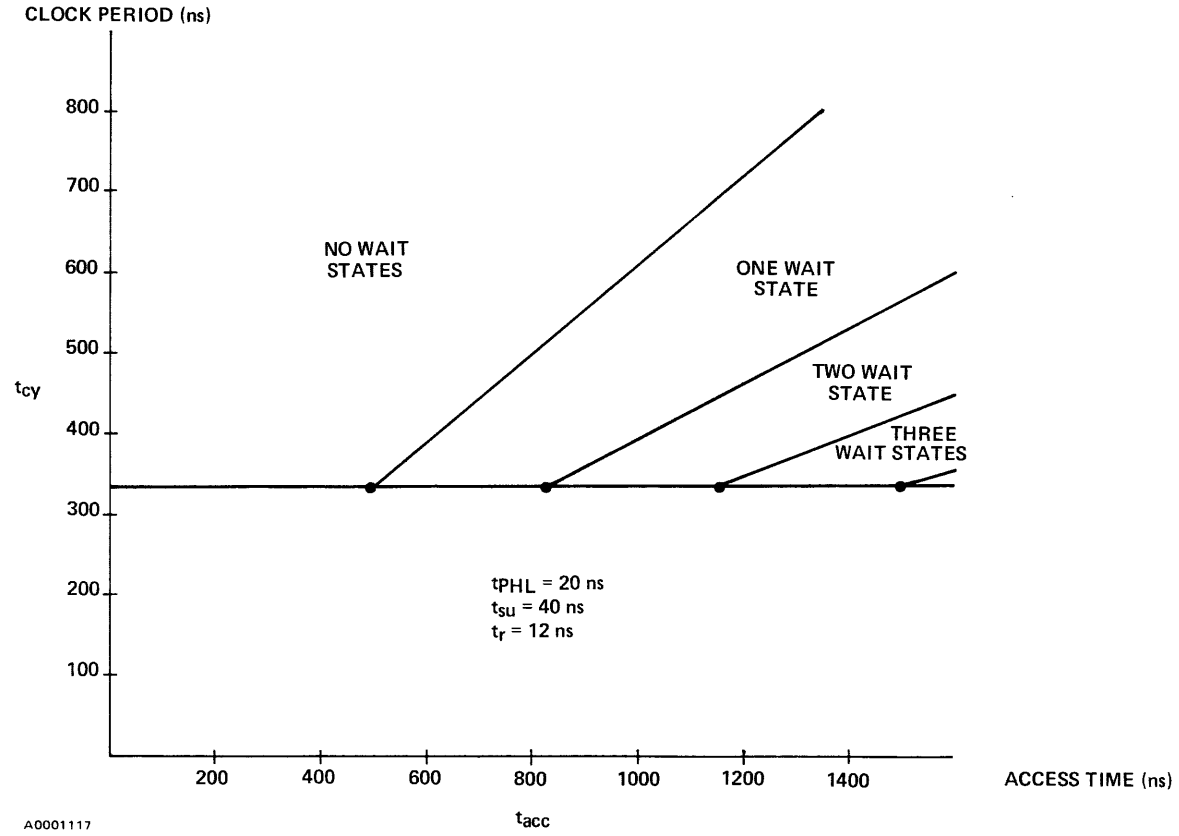


Figure 3-10. Memory Wait Time for Slow Memory

3.3 Static Memory

Static RAMs and PROMs are easily interfaced to the TMS 9900. A TMS 9900 memory system using the TMS 4042-2 256 X 4 static RAM and the TMS 2708 1K X 8 EPROM is shown in Figure 3-11.

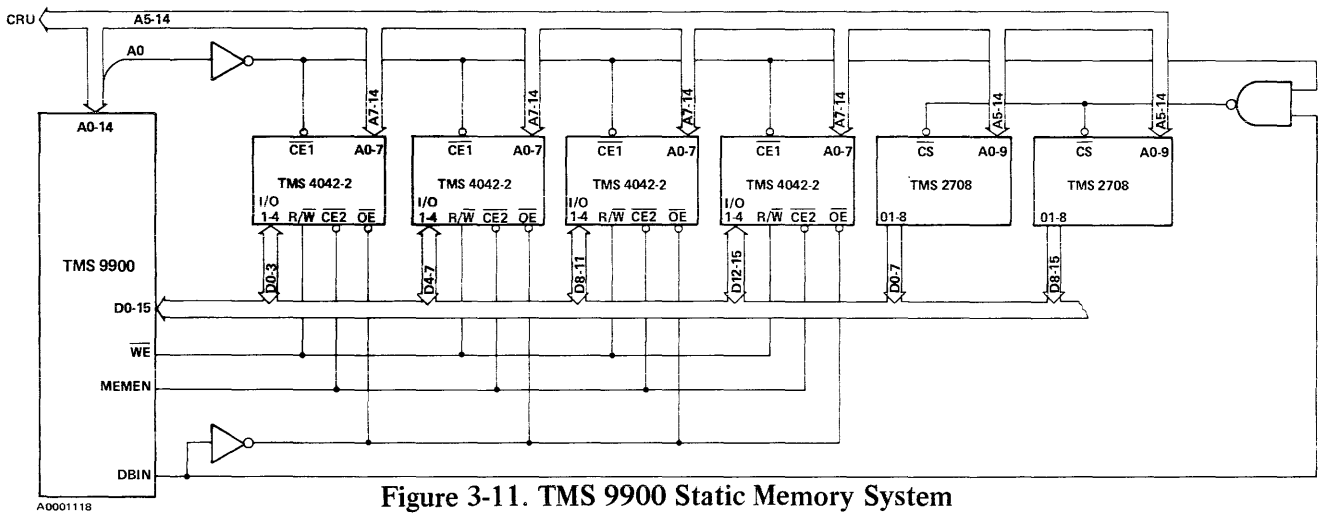


Figure 3-11. TMS 9900 Static Memory System

3.3.1 Address

The most-significant address bit, A0, is used to select either the EPROMs or the RAMs during memory cycles. When A0 is low, the EPROMs are selected, and when A0 is high, the RAMs are selected. Address lines A1 through A4 are not used since the full address space of the TMS 9900 is not required in the example. The lower address bits select internal RAM or EPROM cells. Other memory systems can fully decode the address word for maximum memory expansion.

3.3.2 Control Signals

Since DBIN is also used to select the EPROMs during memory-write cycles, the EPROMs cannot inadvertently be selected and placed into output mode while the CPU is also in the output mode on the data bus. $\overline{\text{MEMEN}}$ is used to select the RAMs during either read or write cycles, and $\overline{\text{WE}}$ is used to select the read/write mode. DBIN is also used to control the RAM output bus drivers.

The TMS 9900 outputs $\overline{\text{WE}}$ three clock phases after the address, data, and $\overline{\text{MEMEN}}$ are output. As a result, the address, data, and enable-hold times are easily met. $\overline{\text{WE}}$ is enabled for one clock cycle and satisfies the minimum write pulse width requirement of 300 nanoseconds. Finally, $\overline{\text{WE}}$ is disabled one clock phase before the address, data, and other control signals and meets the TMS 4042-2, 50-nanosecond minimum data and address hold time.

3.3.3 Loading

The loads on the CPU and memory outputs are well below the maximum rated loads. As a result no buffering is required for the memory system in Figure 3-11. The TMS 4042-2 and the TMS 2708 access times are low enough to eliminate the need for wait states, and the CPU READY input is connected to V_{CC}.

The minimum high-level input voltage of the TMS 2708 is 3 volts while the maximum high-output voltage for the TMS 9900 is 2.4 volts at the maximum specified loading. As described in Section 7.2.1, the TMS 9900 output voltage exceeds 3 volts and pull-up resistors are not needed for the system in Figure 3-11. The loads on the CPU and memory outputs are well below the maximum rated load and thus do not require buffering of the address or data lines.

There are many other Texas Instruments static memories compatible with the TMS 9900 as shown in Table 3-1. The memory devices in the table do not require wait states when used with the TMS 9900 at 3 MHz.

3.4 Dynamic Memory

Memory applications requiring large bit storage can use 4K or 16K dynamic memories for low cost, low power consumption, and high bit density. TMS 9900 systems requiring 4K words or more of RAM, can economically use the 4096-bit TMS 4051, the 16,384-bit TMS 4070, or any of the other dynamic RAMs shown in Table 3-1.

3.4.1 Refresh

The dynamic RAMs must be refreshed periodically to avoid the loss of stored data. The RAM data cells are organized into a matrix of rows and columns with on-chip gating to select the addressed bit. Refresh of the 4K RAM cell matrix is accomplished by performing a memory cycle of each of the 64 row addresses every 2 milliseconds or less. The 16K RAM has 128 row addresses. Performing a memory cycle at any cell on a row refreshes all cells in the row, thus allowing the use of arbitrary column address during refresh.

3.4.2 Refresh Modes

There are several dynamic memory refresh techniques which can be used for a TMS 9900 system. If the system periodically accesses at least one cell of each row every 2 milliseconds, then no additional refresh circuitry is required. A CRT controller which refreshes the display periodically is an example of such a system.

Refresh control logic must be included, however, in many systems since the system cannot otherwise ensure that all rows are refreshed every 2 milliseconds. The dynamic memory in such TMS 9900 systems can be refreshed in the block, cycle stealing, or transparent mode.

3.4.2.1 Block Refresh. The block mode of refresh halts the CPU every 2 milliseconds and sequentially refreshes each of the rows. The block technique halts execution for a 128 (4K) or 256 (16K) clock cycle period every 2 milliseconds. Some TMS 9900 systems cannot use this technique because of the possible slow response to priority interrupts or because of the effect of the delay during critical timing or I/O routines.

3.4.2.2 Cycle Stealing. The cycle stealing mode of refresh "steals" a cycle from the system periodically to refresh one row. The refresh interval is determined by the maximum refresh time and the number of rows to be refreshed. The 4K dynamic RAMs have 64 rows to be refreshed every 2 milliseconds and thus require a maximum cycle stealing interval of 31.2 microseconds.

A cycle stealing refresh controller for the TMS 4051 4K dynamic RAM is shown in Figure 3-12. The refresh timer generates the refresh signal (RFPLS) every 30 microseconds. The refresh request signal (RFREQ) is true until the refresh cycle is completed. The refresh grant signal (RFGNT) goes high during the next CPU clock cycle in which the CPU is not accessing the dynamic memory. The refresh memory cycle takes two clock cycles to complete after RFGNT is true. During the second clock cycle, however, the CPU can

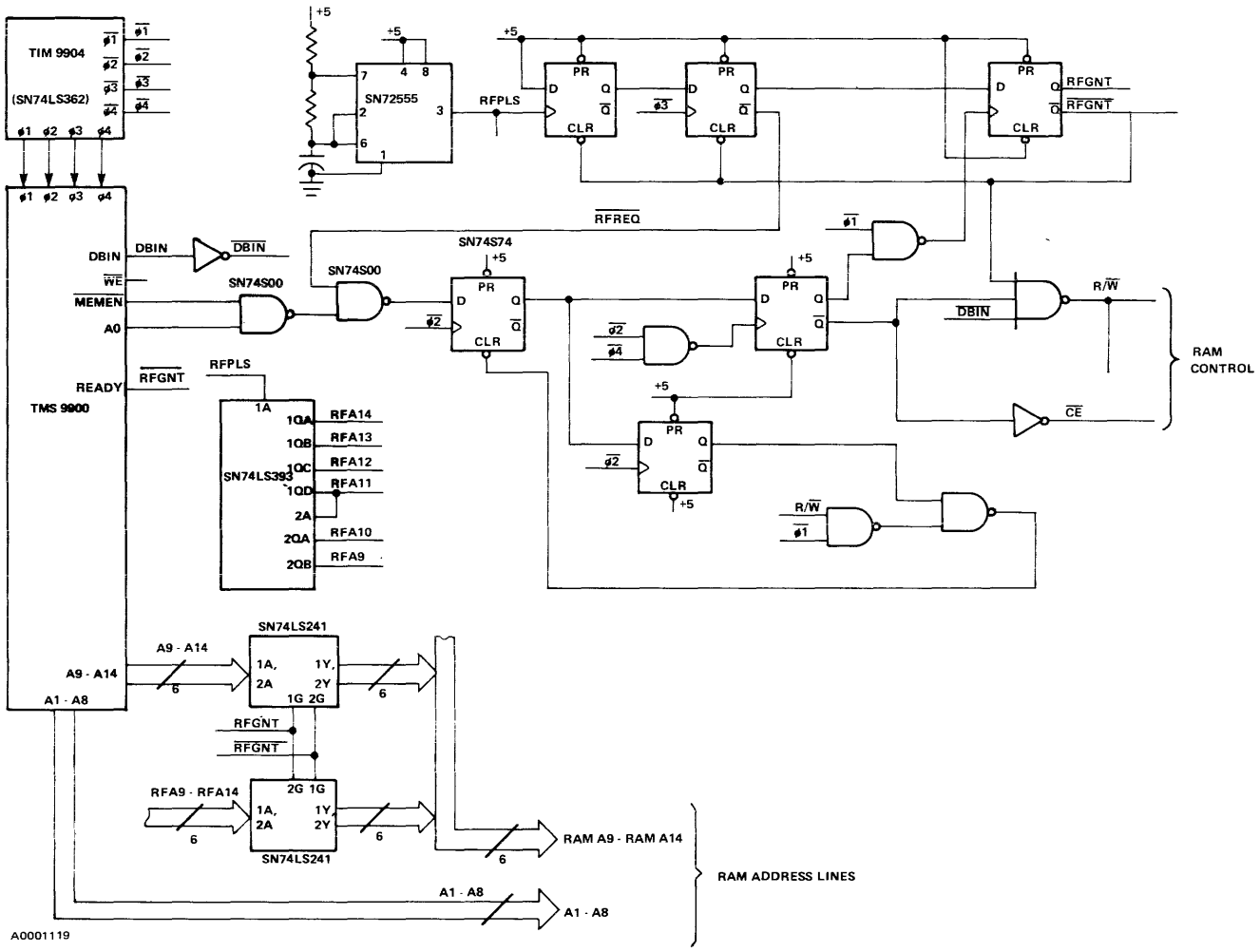


Figure 3-12. Cycle-Stealing Dynamic RAM Refresh for TMS 4051

attempt to access the dynamic memory since the CPU is not synchronized to the refresh controller. If the CPU does access memory during the last clock cycle of the refresh memory cycle, the refresh controller makes the memory not-ready for the remainder of the refresh memory cycle, and the CPU enters a wait state during this interval. The dynamic memory row address during the refresh memory cycle is the output of a modulo-64 counter. The counter is incremented each refresh cycle in order to refresh the rows sequentially.

The dynamic memory timing controller generates the proper chip enable timing for both CPU and refresh initiated memory cycles. The timing controller can be easily modified to operate with other dynamic RAMs.

Since the TMS 9900 performs no more than three consecutive memory cycles, the refresh request will be granted in a maximum of three memory cycles. Some systems may have to block DMA, which uses $\overline{\text{HOLD}}$ as described in Section 3.4. RFREQ can be used in such systems to disable HOLDA temporarily in order to perform a refresh memory cycle if the DMA block transfer is relatively long (greater than 30 microseconds).

The cycle stealing mode "steals" clock cycles only when the CPU attempts to access the dynamic memory during the last half of the refresh cycle. Even if this interference occurs during each refresh cycle, a maximum of 64 clock cycles are "stolen" for refresh every 2 milliseconds.

3.4.2.3 Transparent Refresh. The transparent refresh mode eliminates this interference by synchronizing the refresh cycle to the CPU memory cycle. The rising edge of $\overline{\text{MEMEN}}$ marks the end of a memory cycle immediately preceding a non-memory cycle. The $\overline{\text{MEMEN}}$ rising edge can initiate a refresh cycle with no interference with memory cycles. The refresh requirement does not interfere with the system throughput since only non-memory cycles are used for the refresh cycles. The worst-case TMS 9900 instruction exeuction sequence (all divides) will guarantee the complete refresh of a 4K or 16K dynamic RAM within 2 milliseconds.

While the transparent refresh mode eliminates refresh-related system performance degradation, the system power consumption can be higher since the RAMs are refreshed more often than required. As many as one-half of the CPU machine cycles can be refresh cycles, resulting in multiple refresh cycles for each row during the refresh interval. This situation can be corrected by adding a timer to determine the start of the refresh interval and an overflow detector for the refresh row counter. When every row has been refreshed during an interval, the refresh circuit is disabled until the beginning of the next interval. Since each row is refreshed only once, the system power consumption is reduced to a minimum.

Direct memory access using $\overline{\text{HOLD}}$ should guarantee that sufficient non-memory cycles are available for refresh during large block transfers. An additional refresh timer can be used to block HOLDA in order to provide periodic refresh cycles.

3.5 Buffered Memory

The TMS 9900 outputs can drive approximately two standard TTL inputs and 200 picofarads. Higher capacitive loads may be driven, but with increased rise and fall times. Many small memory systems can thus be directly connected to the CPU without buffer circuits. Larger memory systems, however, may require external bipolar buffers to drive the address or data buses because of increased loading. Texas Instruments manufactures a number of buffer circuits compatible with the TMS 9900. The SN74LS241 noninverting-octal buffer with three-state outputs is an example of a buffer circuit.

A TMS 9900 memory system with address and data bus buffering is shown in Figure 3-13. The system consists of sets of four 256 X 4 memory devices in parallel to provide the 16-bit data word. The four sets of four devices provide a total of 1024 words of memory. The memory devices can be the TMS 4043-2 NMOS static RAM or the SN74S287 bipolar PROM. The devices within each set must be all RAMs or all PROMs, and the sets can all be RAM, all PROM, or any mixture of sets.

The SN74S412 octal buffer/latch is designed to provide a minimum high-level output voltage of 3.65 V. Buffered TMS 9900 memory systems containing the TMS 4700 ROM or the TMS 4908 EPROM, for example, require input voltages in excess of the output voltages of many buffer circuits. The SN74S412 can be used to buffer the memories without the pull-up resistors needed for buffers.

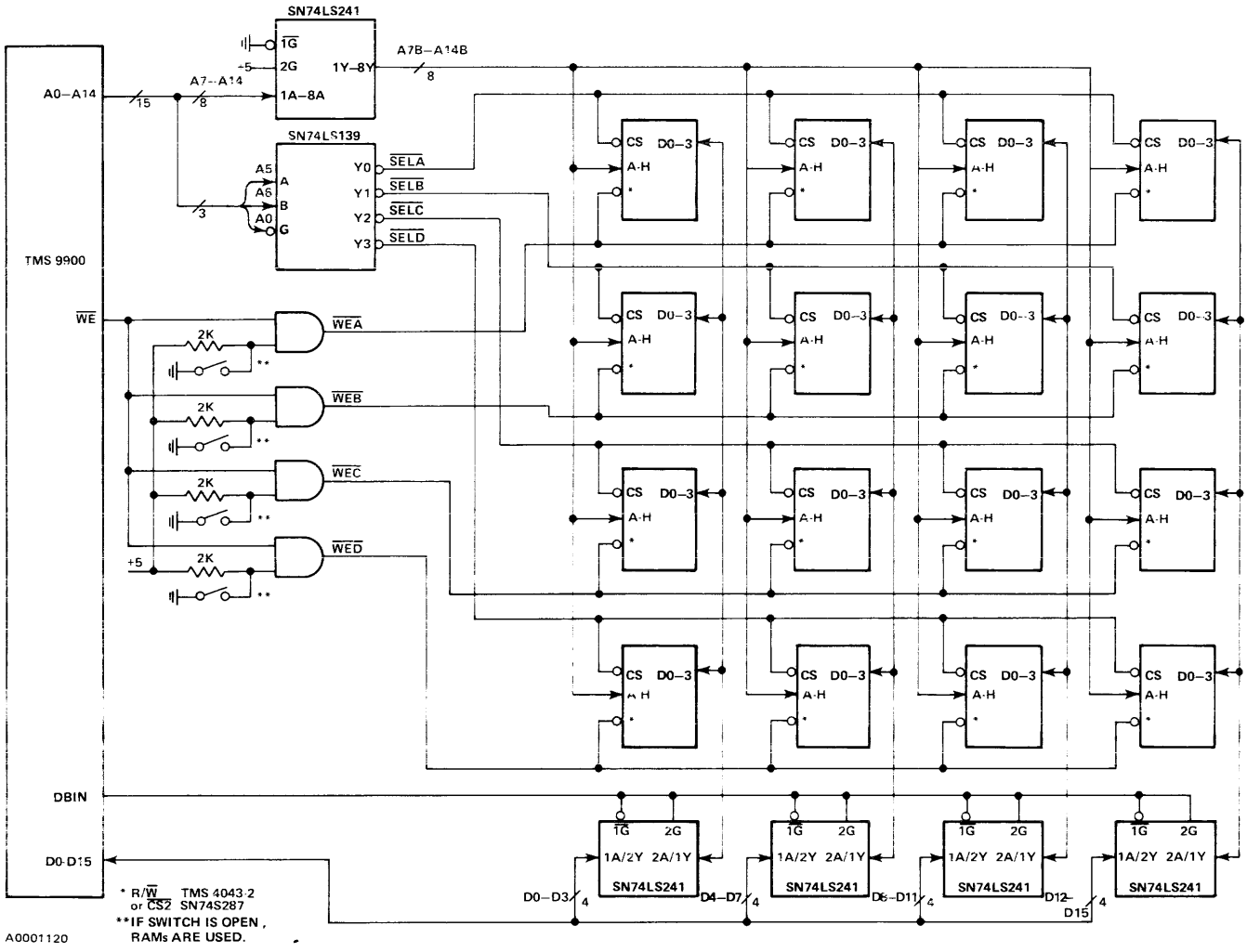


Figure 3-13. Buffered Memory with Mixed PROM/ROM

3.6 Memory Parity

Parity or other error detection/correction schemes are often used to minimize the effects of memory errors. Error detection schemes such as parity are used to indicate the presence of bad data, while error correction schemes correct single or multiple errors.

The SN742LS280 parity generator/checker can be used to implement memory parity in a TMS 9900 system. The system in Figure 3-14 uses two SN74LS280 circuits to generate and to check the odd-memory parity. During memory write cycles, the generated parity bit is output to bit D16 of the memory. During memory read cycles, the parity is checked and an interrupt, PARERR, is generated if the parity is even.

It should be noted that the faulty memory word will have already been used by the CPU as an op code, address, or data before the interrupt is generated. This can cause trouble in determining the exact location of the error. For example, an error in bit 8 of the CLR op code will cause the CPU to branch

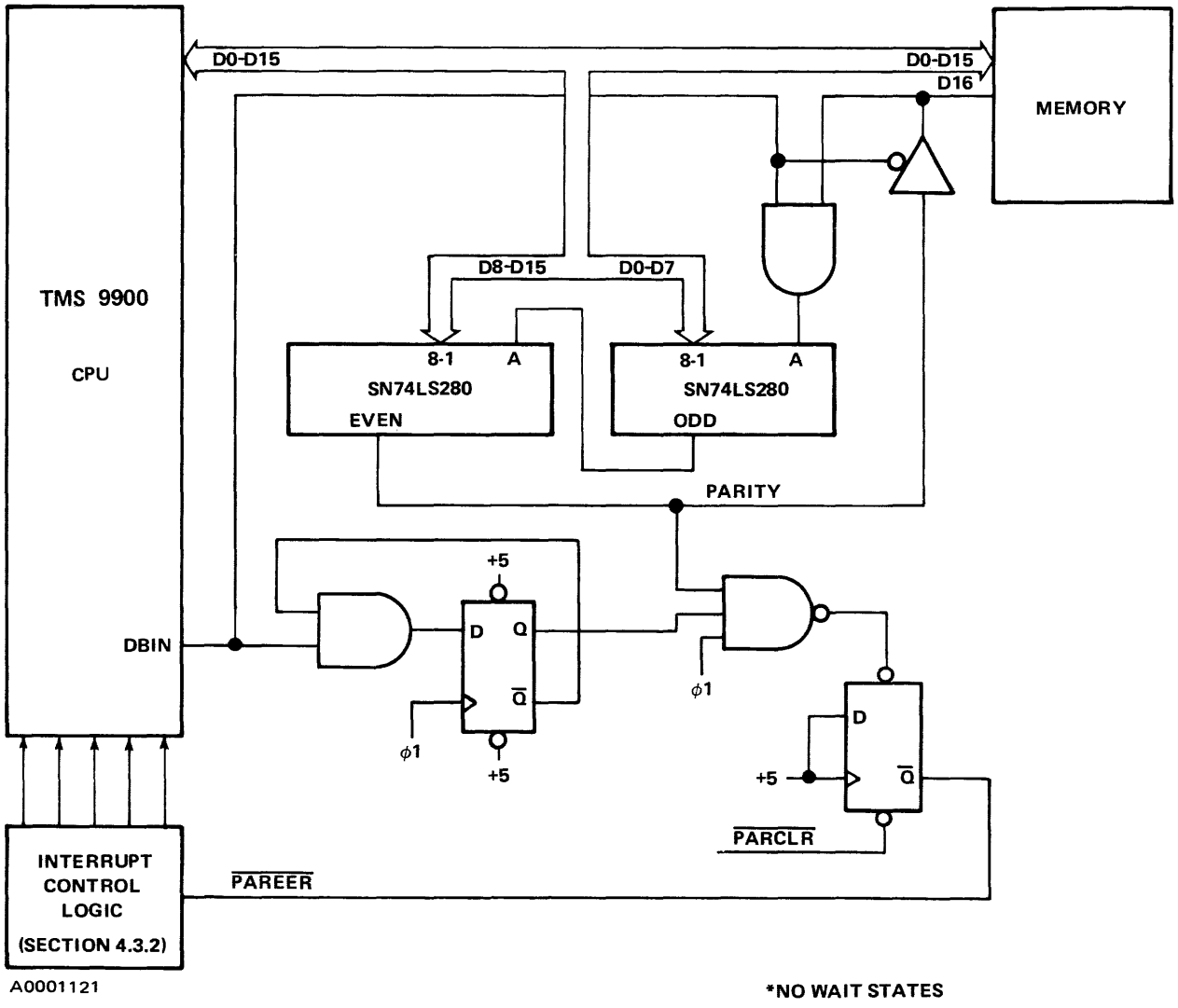


Figure 3-14. Memory Parity Generator Checker

unconditionally. When the interrupt is serviced, there would then be no linkage to the part of the program at which the error occurred. A diagnostic routine can often isolate such errors by scanning the memory and checking parity under program control. Such a parity error in the diagnostic itself can be extremely difficult to isolate.

An external address latch clocked by IAQ can be used to retain program linkage under the above circumstances. When the parity error is detected, the address latch is frozen, thus pointing to the address of the instruction during which the parity error occurred.

3.7 Direct Memory Access

The TMS 9900 controls CRU-based I/O transfers between the memory and peripheral devices. Data must pass through the CPU during these program-driven I/O transfers, and the CPU may need to be synchronized with the I/O device by interrupts or status-bit polling.

Some I/O devices, such as disk units, transfer large amounts of data to or from memory. Program driven I/O can require relatively large response times, high program overhead, or complex programming techniques. Consequently, direct memory access (DMA) is used to permit the I/O device to transfer data to or from memory without CPU intervention. DMA can result in a high I/O response time and system throughput, especially for block data transfers. The DMA control circuitry is somewhat more expensive and complex than the economical CRU I/O circuitry and should therefore be used only when required.

TMS 9900-based DMA can occur in the same modes as dynamic memory refresh: block, cycle stealing, or transparent. The transparent DMA mode is implemented similarly to the refresh mode and must be synchronized with memory refresh cycles if dynamic memory is used. The block and cycle stealing modes, however, use the CPU $\overline{\text{HOLD}}$ capability and are more commonly used. The I/O device holds $\overline{\text{HOLD}}$ active (low) when a DMA transfer needs to occur. At the beginning of the next available non-memory cycle, the CPU enters the hold state and raises HOLDA to acknowledge the $\overline{\text{HOLD}}$ request. The maximum latency time between the hold request and the hold acknowledge is equal to three clock cycles plus three memory cycles. The minimum latency time is equal to one clock cycle. A 3-megahertz system with no wait cycles has a maximum hold latency of nine clock cycles or 3 microseconds and a minimum hold latency of one clock cycle or 0.3 microseconds.

When HOLDA goes high, the CPU address bus, data bus, DBIN, MEMEN, and WE are in the high-impedance state to allow the I/O device to use the memory bus. The I/O device must then generate the proper address, data, and control signals and timing to transfer data to or from the memory as shown in Figure 3-15. Thus the DMA device has control of the memory bus when the TMS 9900 enters the hold state (HOLDA = 1), and may perform memory accesses without intervention by the microprocessor. Since DMA operations, in effect remove the TMS 9900 from control while memory accesses are being performed, no further discussion is provided in this manual. Because the lines shown in Figure 3-15 go into high impedance when HOLDA = 1, the DMA controller must force these signals to the proper levels. The I/O device can use the memory bus for one transfer (cycle-stealing mode) or for multiple transfers (block mode). At the end of the DMA transfer, the I/O device releases $\overline{\text{HOLD}}$ and normal CPU operation proceeds. TMS 9900 $\overline{\text{HOLD}}$ and HOLDA timing are shown in Figure 3-16.

3.8 Memory Layout

It is generally advantageous to lay out memory devices as arrays in the system. The advantages are twofold. First, positioning the devices in an orderly fashion simplifies identification of a particular memory element when troubleshooting. Second, and most important, layout of memory arrays simplifies layout, shortens interconnections, and generally allows a more compact and efficient utilization of board space. Crosstalk between adjacent lines in memory arrays is minimized by running address and data lines parallel to each other, and by running chip enable signals perpendicular to the address lines.

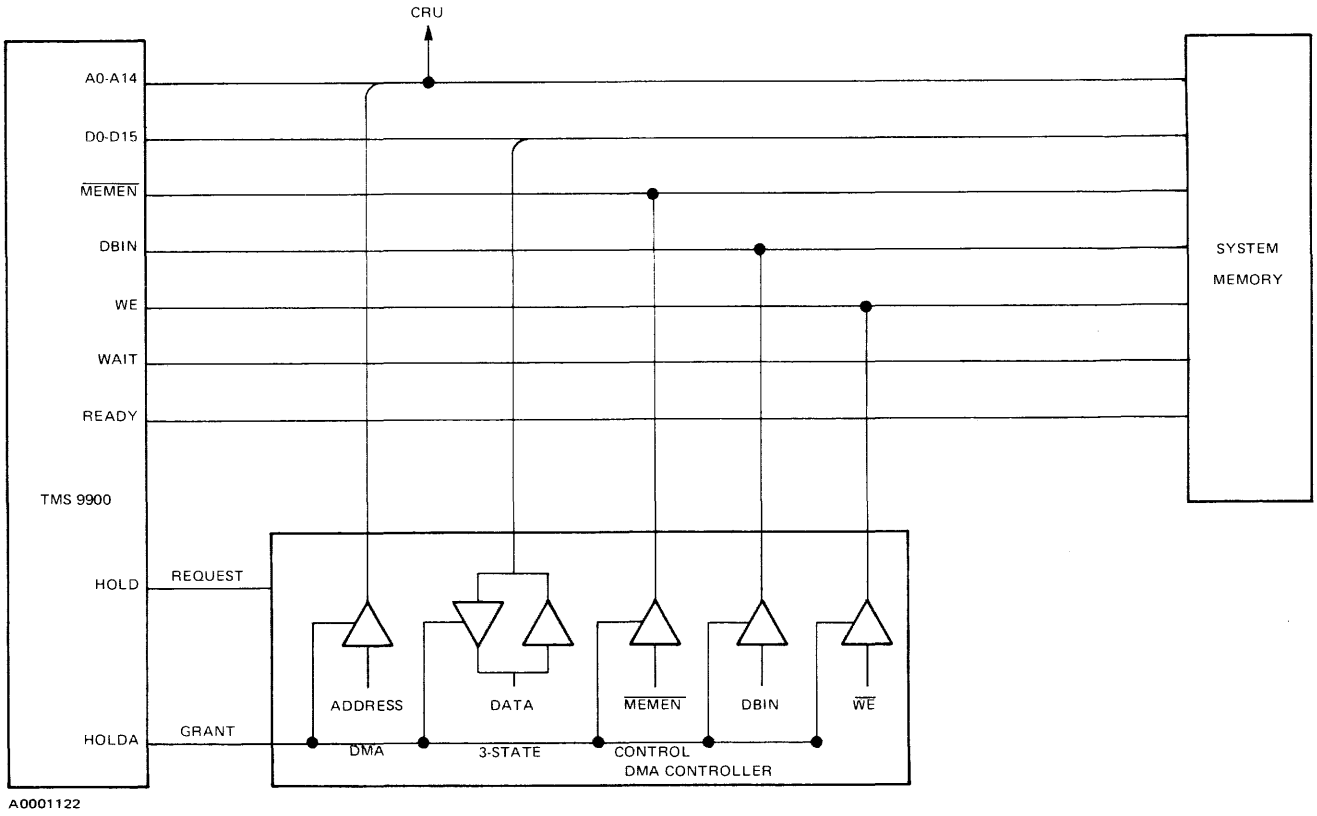


Figure 3-15. DMA Bus Control

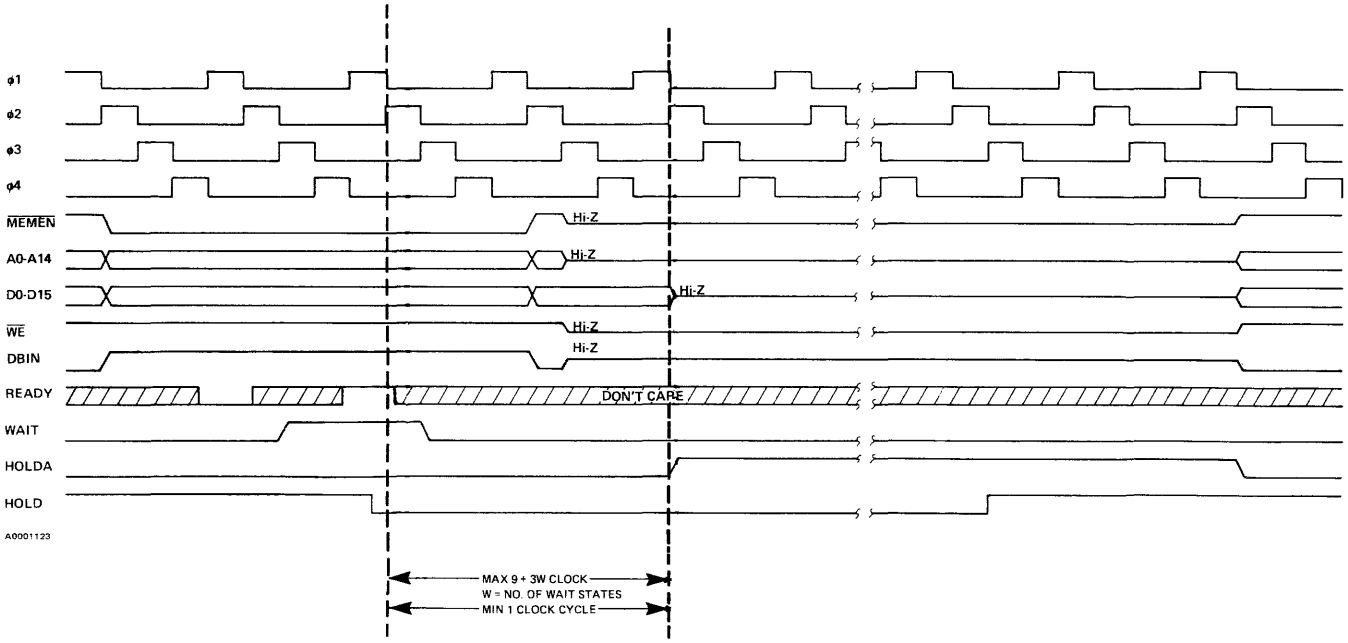


Figure 3-16. $\overline{\text{HOLD}}$ and HOLDA Timing

Memory devices, particularly dynamic RAMs generally require substantially greater supply currents when addressed than otherwise. It is therefore important that all power and ground paths be as wide as possible to memory arrays. Furthermore, in order to avoid spikes in supply voltages, it is advisable to decouple supply voltages with capacitors as close as possible to the pins of the memory devices. As an example, a system containing a 4K x 16-bit array of TMS 4051s should contain one 15 μF and one 0.05 μF capacitor for each set of four memory devices; with the large capacitors decoupling V_{DD}, and the small capacitors decoupling V_{BB}.

SECTION IV

INTERRUPTS

The TMS 9900 provides fifteen maskable interrupt levels in addition to the $\overline{\text{RESET}}$ and $\overline{\text{LOAD}}$ functions. The CPU has a priority ranking system to resolve conflicts between simultaneous interrupts and a level mask to disable lower priority interrupts. Once an interrupt is recognized, the CPU performs a vectored context switch to the interrupt service routine. The $\overline{\text{RESET}}$ and $\overline{\text{LOAD}}$ functions are initiated by external input signals and should not be confused with the RSET and LREX instructions which are described in Section V.

4.1 RESET

The $\overline{\text{RESET}}$ signal is normally used to initialize the CPU following a power-up. When active (low), the $\overline{\text{RESET}}$ signal inhibits $\overline{\text{WE}}$ and CRUCLK, places the CPU memory bus and control signals in a high-impedance state, and resets the CPU. When the $\overline{\text{RESET}}$ signal is released, the CPU fetches the restart vector from locations 0000 and 0002, stores the old WP, PC, and ST into the new workspace, resets all status bits to zero and starts execution at the new PC. The $\overline{\text{RESET}}$ signal must be held active for a minimum of three clock cycles. The $\overline{\text{RESET}}$ machine cycle sequence is shown in Figure 4-1.

A convenient method of generating the $\overline{\text{RESET}}$ signal is to use the Schmitt-triggered D-input of the TIM 9904 clock generator. An RC network connected to the D-input maintains an active $\overline{\text{RESET}}$ signal for a short time immediately following the power-on, as shown in Figure 4-2.

CYCLE	TYPE	FUNCTION
*	*	Loop While Reset is Active
1	ALU	Set Up
2	ALU	Set Up
3	Memory	Fetch New WP, Move Status To T Reg, Clear Status
4	ALU	Set Up
5	Memory	Store Status
6	ALU	Set Up
7	Memory	Store PC
8	ALU	Set Up
9	Memory	Store WP
10	ALU	Set Up
11	Memory	Fetch New PC
12	ALU	Set Up MAR for Next Instruction

A0001124

Figure 4-1. $\overline{\text{RESET}}$ Machine Cycles

4.2 LOAD

The $\overline{\text{LOAD}}$ signal is normally used to implement a restart ROM loader or front panel functions. When active (low), the $\overline{\text{LOAD}}$ signal causes the CPU to perform a non-maskable interrupt. The $\overline{\text{LOAD}}$ signal can be used to terminate a CPU idle state.

The $\overline{\text{LOAD}}$ signal should be active for one instruction period. Since there is no standard TMS 9900 instruction period, IAQ should be used to determine instruction boundaries. If the $\overline{\text{LOAD}}$ signal is active during the time that the $\overline{\text{RESET}}$ signal is released, the CPU will perform the $\overline{\text{LOAD}}$ function immediately after the $\overline{\text{RESET}}$ function is completed. The CPU performs the $\overline{\text{LOAD}}$ function

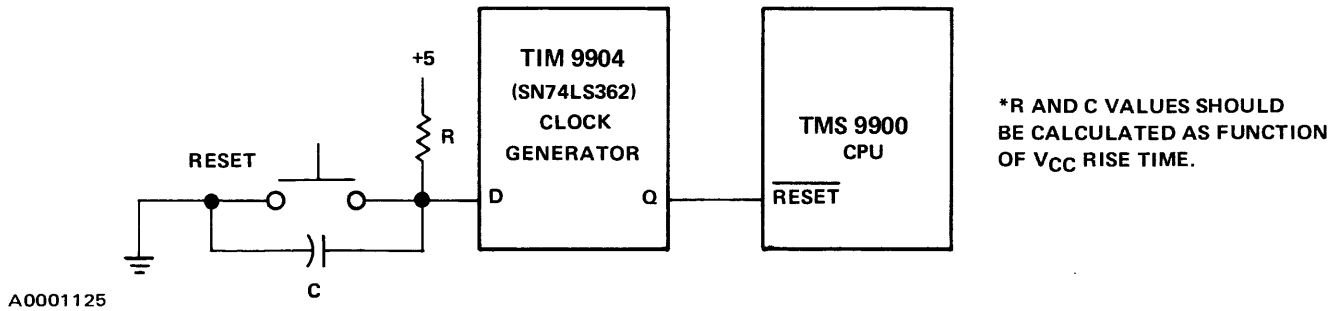


Figure 4-2. $\overline{\text{RESET}}$ Generation

by fetching the $\overline{\text{LOAD}}$ vector from addresses $FFFC_{16}$ and $FFFE_{16}$, storing the old WP, PC, and ST in the new workspace, and starting the $\overline{\text{LOAD}}$ service routine at the new PC, as shown in Figure 4-3.

An example of the use of the $\overline{\text{LOAD}}$ signal is a bootstrap ROM loader. When the $\overline{\text{LOAD}}$ signal is enabled, the CPU enters the service routine, transfers a program from peripheral storage to RAM, and then transfers control to the loaded program.

Figure 4-4 illustrates the generation of the $\overline{\text{LOAD}}$ signal for one instruction period.

CYCLE	TYPE	FUNCTION
1	ALU	Set Up
2	Memory Read	Fetch New WP
3	ALU	Set Up
4	Memory Write	Store Status
5	ALU	Set Up
6	Memory Write	Store PC
7	ALU	Set Up
8	Memory Write	Store WP
9	ALU	Set Up
10	Memory Read	Fetch New PC
11	ALU	Set UP MAR for Next Instruction

A0001126

Figure 4-3. $\overline{\text{LOAD}}$ Machine Cycle Sequence

4.3 Maskable Interrupts

The TMS 9900 has 16 interrupt levels with the lower 15 priority levels used for maskable interrupts. The maskable interrupts are prioritized and have transfer vectors similar to the $\overline{\text{RESET}}$ and $\overline{\text{LOAD}}$ vectors.

4.3.1 Interrupt Service

A pending interrupt of unmasked priority level is serviced at the end of the current instruction cycle with two exceptions. The first instruction of a $\overline{\text{RESET}}$, $\overline{\text{LOAD}}$, or interrupt service routine is executed before the CPU tests the $\overline{\text{INTREQ}}$ signal. The interrupt is also inhibited for one instruction if the current instruction is a branch and load workspace pointer instruction (BLWP) or an extended operation (XOP). The one instruction delay permits one instruction to be completed before an interrupt context switch can occur. A LIMI instruction can be used as the first instruction in a routine to lock out higher priority maskable interrupts.

The pending interrupt request should remain active until recognized by the CPU during the service routine. The interrupt request should then be cleared under program control. The CRU bit manipulation instructions can be used to recognize and clear the interrupt request.

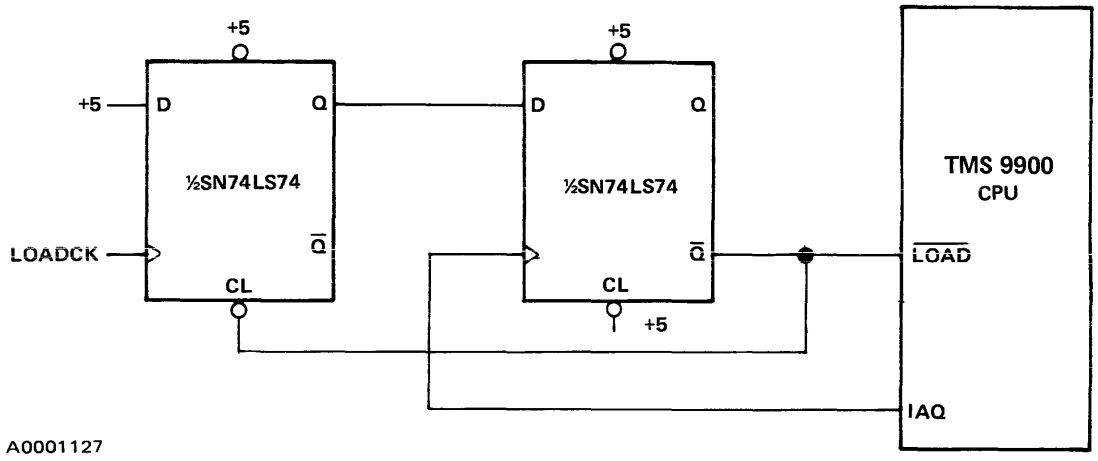


Figure 4-4. $\overline{\text{LOAD}}$ Generation

The interrupt context switch causes the interrupt vector to be fetched, the old WP, PC, and ST to be saved in the new workspace, and the new WP and PC to be loaded. Bits 12–15 of ST are loaded with a value of one less than the level of the interrupt being serviced. The old WP, PC, and ST are stored in the new workspace registers 13, 14, and 15. When the return instruction is executed, the old WP, PC, and ST are restored to the CPU. Since the ST contains the interrupt mask, the old interrupt level is also restored. Consequently, all interrupt service routines should terminate with the return instruction in order to restore the CPU to its state before the interrupt.

The linkage between two interrupt service routines is shown in Figure 4-5, and the interrupt machine cycle sequence is shown in Figure 4-6.

4.3.2 Interrupt Signals

The TMS 9900 has five inputs used for maskable interrupts. The $\overline{\text{INTREQ}}$ signal is active (low) when a maskable interrupt is pending. If $\overline{\text{INTREQ}}$ is active at the end of the instruction cycle, the CPU compares the priority code on IC0 through IC3 to the interrupt mask (ST12–ST15). If the interrupt code of the pending interrupt is equal to or less than the current interrupt mask, the CPU executes a vectored interrupt; otherwise, the interrupt request is ignored. The interrupt priority codes are shown in Table 4-1. Note that the level-0 interrupt code should not be used for external interrupts since level 0 is reserved for RESET.

Figure 4-7 illustrates the use of the TMS 9901 programmable system interface for generation of the interrupt code from individual interrupt input lines. The TMS 9901 provides six dedicated and nine programmable latched, synchronized, and prioritized interrupts, complete with individual enabling/disabling masks. Figure 4-8 illustrates the use of an SN74148 priority encoder and an SN74373 octal latch to provide a TTL eight-input interrupt system. Synchronization prevents transitions of IC0–IC3 while the code is being read. A single-interrupt system with an arbitrarily chosen level-7 code is shown in Figure 4-9. The single-interrupt input does not need to be synchronized since the hardwired interrupt code is always stable.

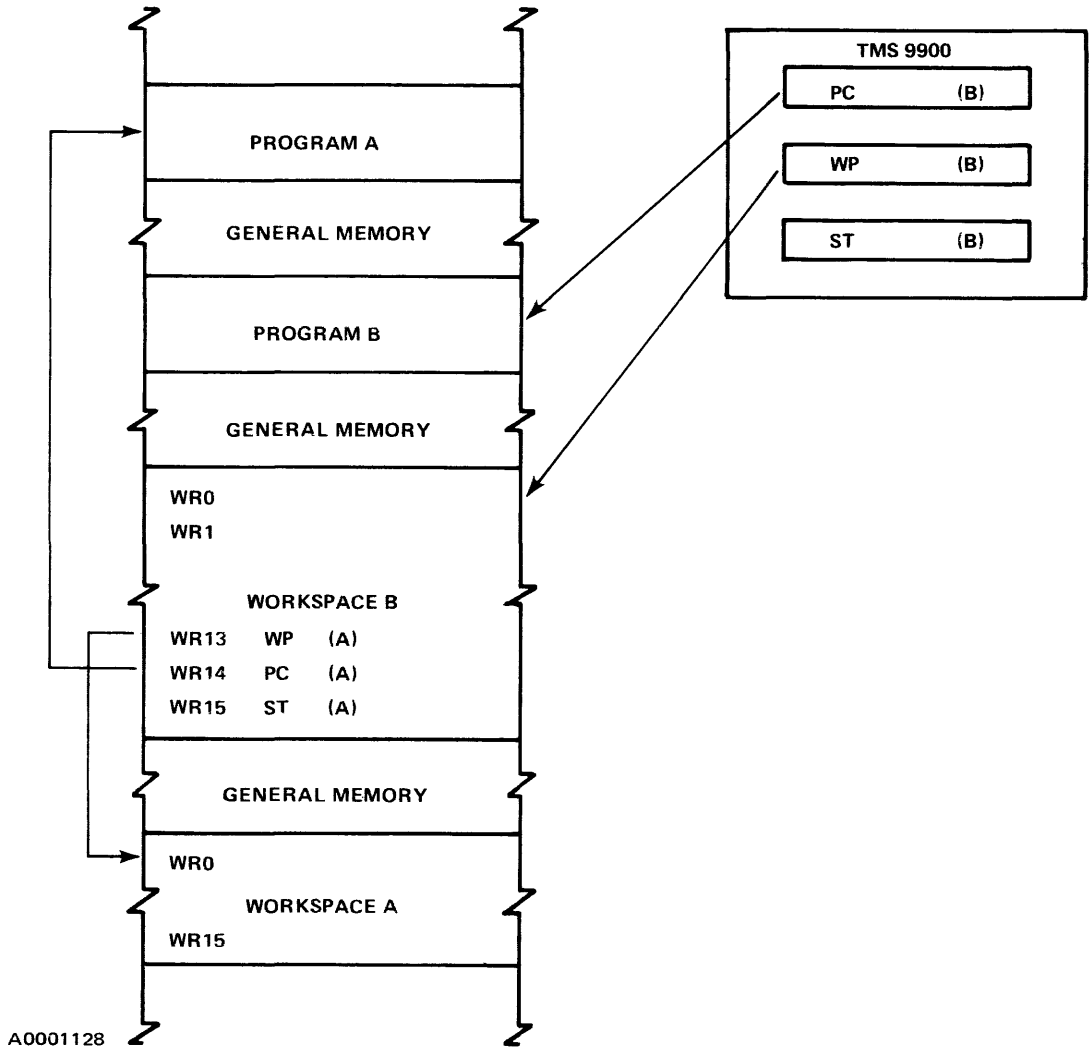


Figure 4-5. Interrupt Linkage

4.3.3 Interrupt Masking

The TMS 9900 uses a four-bit field in the status register, ST12 through ST15, to determine the current interrupt priority level. The interrupt mask is automatically loaded with a value of one less than the level of the maskable interrupt being serviced. The interrupt mask is also affected by the load interrupt mask instruction (LIMI).

Since the interrupt mask is compared to the external interrupt code before an interrupt is recognized, an interrupt service routine will not be halted due to another interrupt of lower or equal priority unless a LIMI instruction is used to alter the interrupt mask. The LIMI instruction can be used to alter the interrupt-mask level in order to disable intervening interrupt levels. At the end of the service routine, a return (RTWP) restores the interrupt mask to its value before the current interrupt occurred.

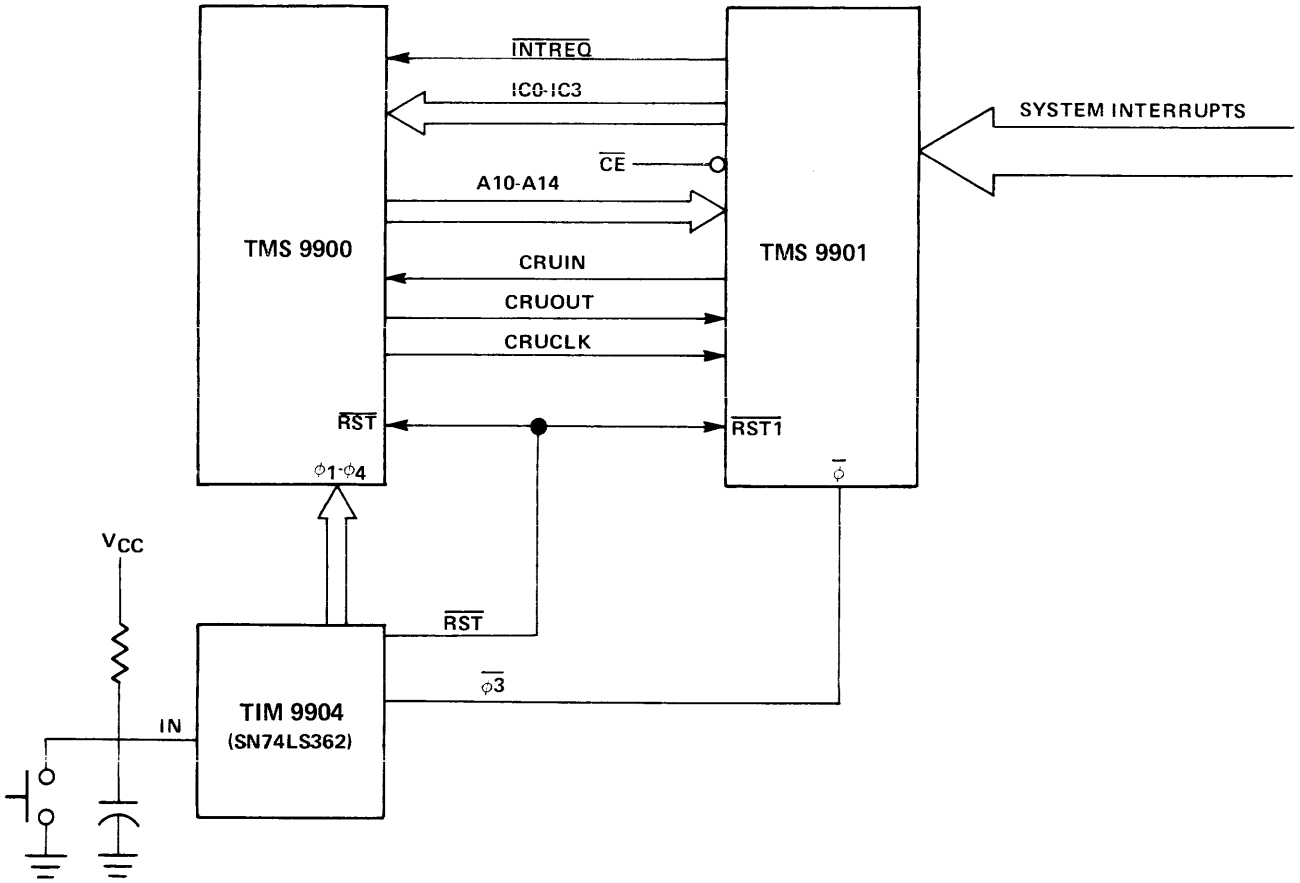
CYCLE	TYPE	FUNCTION
1	ALU	Set Up
2	Memory Read	Fetch New WP
3	ALU	Set Up
4	Memory Write	Store Status
5	ALU	Set Up
6	Memory Write	Store PC
7	ALU	Set Up
8	Memory Write	Store WP
9	ALU	Set Up
10	Memory Read	Fetch New PC
11	ALU	Set Up MAR for Next Instruction

A0001129

Figure 4-6. Interrupt Processing Machine Cycle Sequence

Note that the TMS 9900 actually generates the interrupt vector address using IC0–IC3 five clock cycles after it has sampled INTREQ and four clock cycles after it has compared the interrupt code to the interrupt mask in the status register. Thus, interrupt sources which have individual masking capability can cause erroneous operation if a command to the device to mask the interrupt occurs at a time when the interrupt is active and just after the TMS 9900 has sampled INTREQ but before the vector address has been generated using IC0–IC3.

The individual interrupt masking operation can be easily allowed if the masking instruction is placed in a short subroutine which masks all interrupts



A0001130

Figure 4-7. System With 15 External Interrupts

TABLE 4-1. INTERRUPT PRIORITY CODES

Interrupt Level		Vector Location (Memory Address In Hex)	Device Assignment	Interrupt Mask Values To Enable Respective Interrupts (ST12 thru ST15)	Interrupt Codes IC0 thru IC3
(Highest priority)	0	00	Reset	0 through F*	0000
	1	04	External device	1 through F	0001
	2	08	↓	2 through F	0010
	3	0C	↓	3 through F	0011
	4	10	↓	4 through F	0100
	5	14	↓	5 through F	0101
	6	18	↓	6 through F	0110
	7	1C	↓	7 through F	0111
	8	20	↓	8 through F	1000
	9	24	↓	9 through F	1001
	10	28	↓	A through F	1010
	11	2C	↓	B through F	1011
	12	30	↓	C through F	1100
	13	34	↓	D through F	1101
	14	38	↓	E and F	1110
(Lowest priority)	15	3C	External device	F only	1111

*Level 0 can not be disabled.

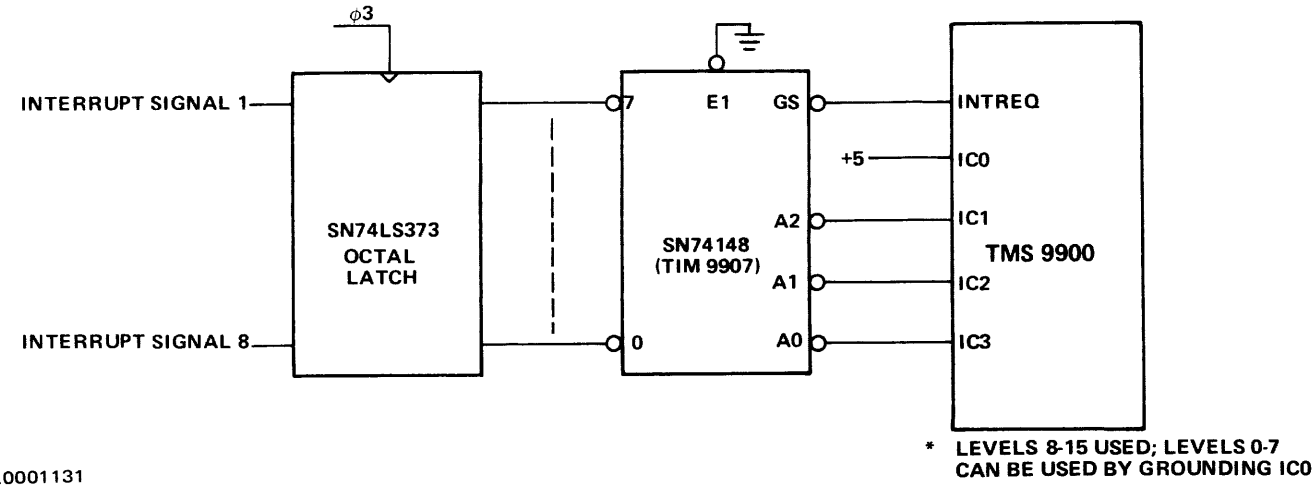


Figure 4-8. Eight-Input Interrupt System With Synchronization

with a LIMI 0 instruction before individually masking the interrupt at the device, as shown in Figure 4-10.

4.3.4 Interrupt Processing Example

The routine in Figure 4-11 illustrates the use of the LIMI instruction as a privileged or non-interruptable instruction. The level-5 routine sets a CRU bit and then loops until a corresponding CRU bit is true. The

first instruction in the routine is completed before a higher priority interrupt can be recognized. The LIMI instruction, however, raises the CPU priority level to level 0 in order to disable all other maskable interrupts. Consequently, the level-5 routine will run to completion unless a $\overline{\text{RESET}}$ signal or a $\overline{\text{LOAD}}$ signal is generated. At the end of the routine, the RTWP instruction restores the CPU to its state before the level-5 interrupt occurred.

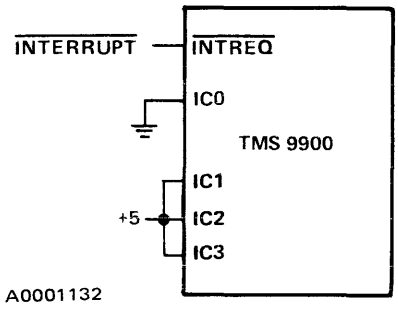


Figure 4-9. Single-Interrupt System

INCORRECT

```
        XXX
        SBO     0           SET MASK (INTERRUPT CAN OCCUR
                            DURING SBO CAUSING ERRONEOUS
        YYY                 OPERATION)
```

CORRECT

```
        XXX
        BLWP    9           (WR9)  =  ADDRESS OF SBW
                            (WR10) =  ADDRESS OF SB1
        XXXX

SB1     LIMI    0           CLEAR STATUS MASK TO INHIBIT INTERRUPTS
        MOV     @ 24(13), 12  MOVE CRU BASE ADDRESS TO WR12
        SBO     0           SET MASK
        RTWP                RETURN

SBW     BSS     32          SUBROUTINE WORKSPACE
```

A0001133

Figure 4-10. External Interrupt Clearing Routine

```
Level 5   LIMI   0      Disable Maskable INTREQs
          SBO    ACK    Set CRU Output Bit
Loop      TB     RDY    Test CRU Input Bit
          JNE    LOOP   Loop Until Input True
          RTWP          Return
A0001134
```

Figure 4-11. LIMI Instruction Routine

SECTION V

INPUT/OUTPUT

The TMS 9900 has three I/O modes: direct memory access (DMA), memory mapped, and communications register unit (CRU). This multi-mode capability enables the designer to optimize a TMS 9900 I/O system to match a specific application. One or all modes can be used, as shown in Figure 5-1.

5.1 Direct Memory Access

DMA is used for high-speed block data transfer when CPU interaction is undesirable or not required. The DMA control circuitry can be relatively complex and expensive when compared to other I/O methods.

5.2 Memory Mapped I/O

Memory mapped I/O permits I/O data to be addressed as memory with parallel data transfer through the system data bus. Memory mapped I/O requires a memory bus compatible interface; that is, the device is addressed in the same manner as a memory, thus the interface is identical to that of memory. Figure 5-2 shows a memory mapped I/O interface with eight latched outputs and eight buffered inputs. In using memory mapped I/O for output only, care must be taken in developing the output device strobe to ensure it

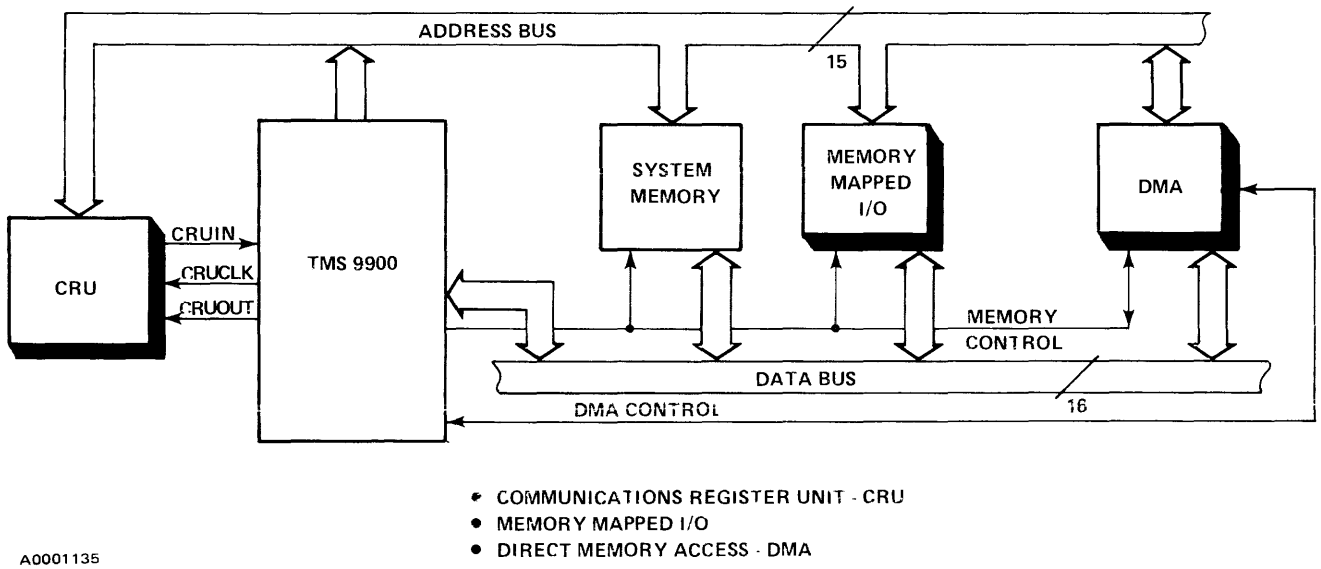


Figure 5-1. TMS 9900 I/O Capability

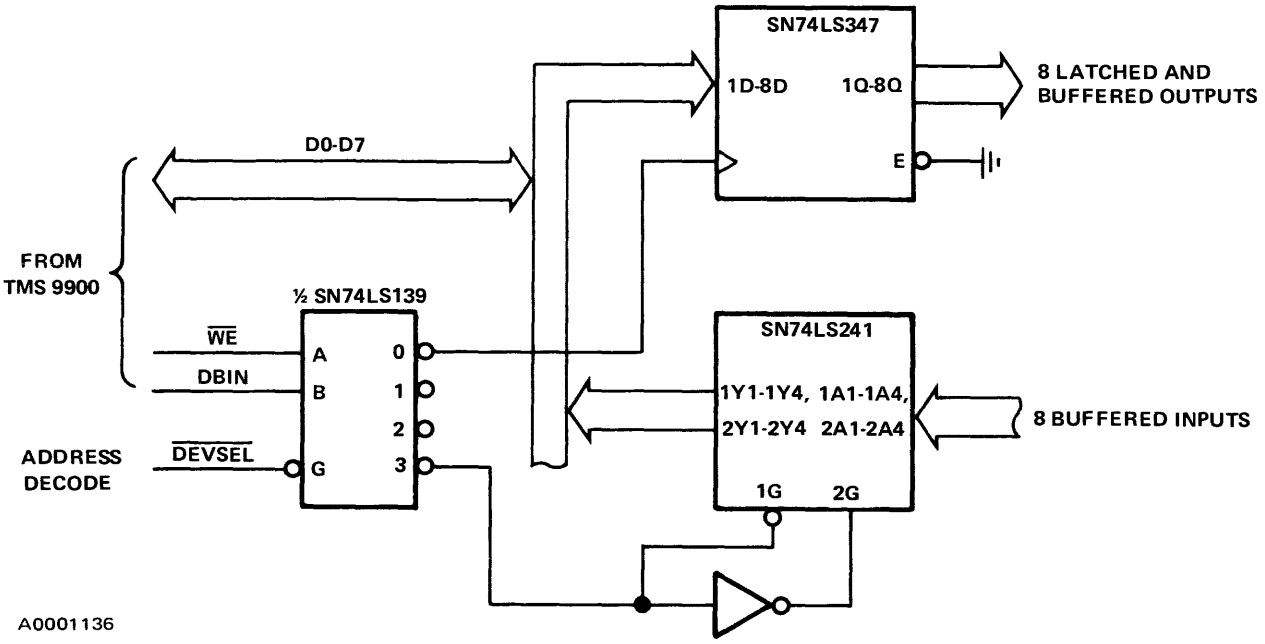


Figure 5-2. 8-Bit Memory Mapped I/O Interface

is not enabled during the initial read of the memory address, since the TMS 9900 family of processors first reads, then writes data to a memory location in write operations. This can be effectively accomplished by using the processor write control signal $\overline{WE}$ in decoding the output address.

5.3 Communication Register Unit (CRU)

CRU I/O uses a dedicated serial interface for I/O. The CRU instructions permit transfer of one to sixteen bits. The CRU interface requires fewer interface signals than the memory interface and can be expanded without affecting the memory system. In the majority of applications, CRU I/O is superior to memory mapped I/O as a result of the powerful bit manipulation capability, flexible field lengths, and simple bus structure.

The CRU bit manipulation instructions eliminate the masking instructions required to isolate a bit in memory mapped I/O. The CRU multiple-bit instructions allow the use of I/O fields not identical to the memory word size, thus permitting optimal use of the I/O interface. Therefore, the CRU minimizes the size and complexity of the I/O control programs, while increasing system throughput.

The CRU does not utilize the data bus which is used only for system memory. This can reduce the complexity of printed circuit board layouts for most systems. The standard 16-pin CRU I/O devices are less expensive and easier to insert than larger, specially designed, memory mapped I/O devices. The smaller I/O devices are possible as a result of the serial CRU bus which eliminates the need for several pins dedicated to a parallel-data bus with multiple control lines. System costs are lower because of simplified circuit layouts, increased density, and lower component costs.

5.3.1 CRU Interface

The interface between the TMS 9900 and CRU devices consists of A0–A14, CRUIN, CRUOUT, and CRUCLK as shown in Figure 5-1. A0–A2 indicate whether data is to be transferred and A3–A14 contain the address of the selected bit for data transfers; therefore, up to 2^{12} or 4,096 bits of input and output may be individually addressed. CRU operations and memory-data transfers both use A0–A14; however, these operations are performed independently, thus no conflict arises.

5.3.2 CRU Machine Cycles

Each CRU operation consists of one or more CRU output or CRU input machine cycles, each of which is two clock cyles long. As shown in Table 5-1, five instructions (LDCR, STCR, SBO, SBZ, TB) transfer data to or from the TMS 9900 with CRU machine cycles, and five external control instructions (IDLE, RSET, CKOF, CKON, LREX) generate control signals with CRU output machine cycles.

TABLE 5-1. INSTRUCTIONS GENERATING CPU CYCLES

Instruction	Number of CRU Cycles	Type of CRU Cycles	A0-A2	Data Transfer
LDCR	1-16	Output	0 0 0	Yes
STCR	1-16	Input	0 0 0	Yes
SBO	1	Output	0 0 0	Yes
SBZ	1	Output	0 0 0	Yes
TB	1	Input	0 0 0	Yes
IDLE	1	Output	0 1 0	No
RSET	1	Output	0 1 1	No
CKOF	1	Output	1 0 1	No
CKON	1	Output	1 1 0	No
LREX	1	Output	1 1 1	No

5.3.2.1 CRU Output Machine Cycles. Figure 5-3 shows the timing for CRU output machine cycles. Address (A0–A14) and data (CRUOUT) are output on $\phi 2$ of clock cycle 1. One clock cycle later, the TMS 9900 outputs a pulse on CRUCLK for ½ clock cycle. Thus, CRUCLK can be used as a strobe, since address and data are stable during the pulse. Referring again to Table 5-1, it is important to note that output data is transferred only when A0–A2 = 000. Otherwise, no data transfer should occur, and A0–A2 should be decoded to determine which external control instruction is being executed. These external control instructions may be used to perform simple control operations. The generation of control strobes for external instructions and a data transfer strobe (OUTCLK) is illustrated in Figure 5-4. If none of the external control instructions is used, A0–A2 need not be decoded for data transfer since they will always equal 000.

5.3.2.2 CRU Input Machine Cycles. The timing for CRU input machine cycles is shown in Figure 5-5. The address is output at the beginning of the first clock cycle. The CRUIN data input is sampled on 01 of clock cycle 2. Thus, CRU input is accomplished by simply multiplexing the addressed bit onto the CRUIN input. A0–A2 will always be 000, and may be ignored. CRU input machine cycles cannot be differentiated from ALU cycles by external logic, thus no operations (such as clearing interrupts) other than CRU input should be performed during CRU input machine cycles.

5.3.3 CRU Data Transfer

The CRU base address is defined by bits 3–14 of the current WR12 when a CRU data transfer is performed. Bits 0–2 and bit 15 of WR12 are ignored for CRU address determination.

5.3.3.1 Single Bit Instructions. For single-bit CRU instructions (SBO, SBZ, TB), the address of the CRU bit to or from which data is transferred is determined as shown in Figure 5-6. Bits 8–15 of the machine code instruction contain a signed displacement. This signed displacement is added to the CRU base address (bits 3–14 of WR12). The result of this addition is output on A3–A14 during the CRU output or the CRU input machine cycle.

For example, assume the instruction "SBO 9" is executed when WR12 contains a value of 1040_{16}. The machine code for "SBO 9" is $1D09_{16}$ and the signed displacement is 0009_{16}. The CRU base address is 0820_{16} (bits 0–2 and bit 15 are ignored). Thus, the effective CRU address is $0820_{16} + 0009_{16} = 0829_{16}$, and this value is output on A0–A14 during the CRU output machine cycle.

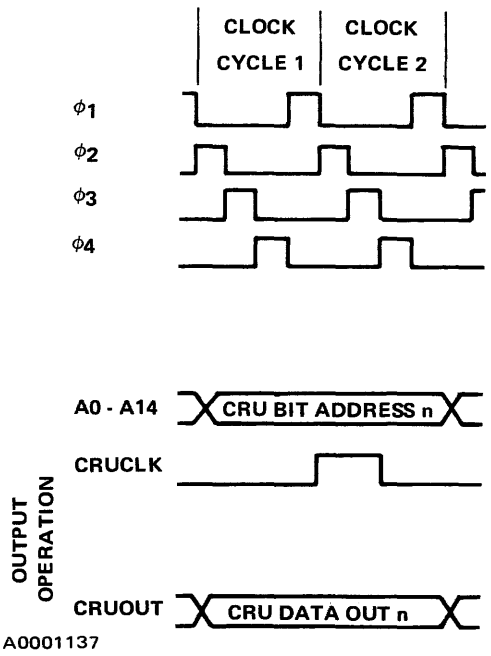


Figure 5-3. CRU Output Machine Cycle Timing

As a second example, assume that the instruction "TB–32" is executed when WR12 = 100_{16}. The effective CRU address is 80_{16} (CRU base) + $FFE0_{16}$ (signed displacement) = 60_{16}. Thus, the "TB–32" instruction in this example causes the value of the CRU input bit at address 60_{16} to be transferred to bit 2 of the status register. This value is operated on by the JEQ or JNE instructions, which cause the PC to be changed depending on the value of ST2.

5.3.3.2 LDCR Instruction. The LDCR may transfer from 1 to 16 bits of output data with each instruction. The output of each bit is performed by a CRU output machine cycle; thus, the number of CRU output machine cycles performed by a LDCR instruction is equal to the number of bits to be transferred.

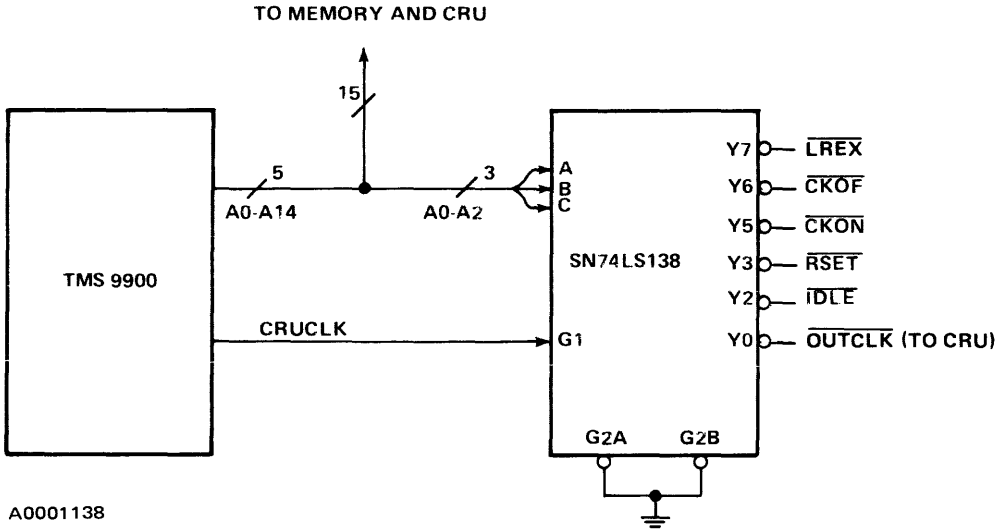


Figure 5-4. CRU Control Strobe Generation

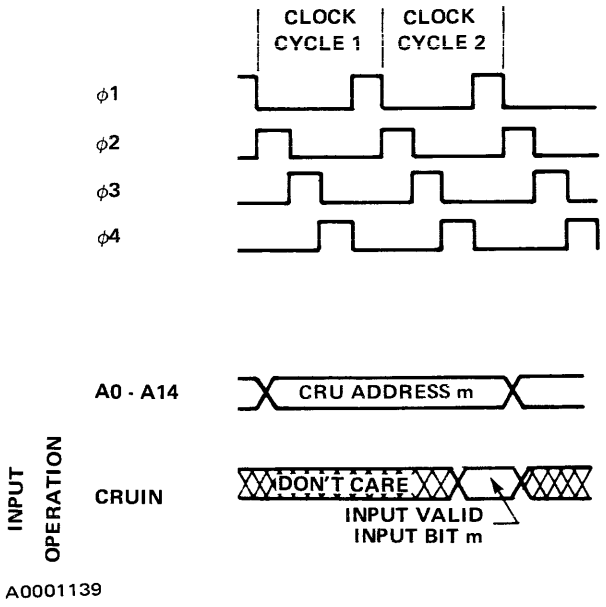


Figure 5-5. CRU Input Machine Cycle Timing

As an example, assume that the instruction "LDCR @ 600,10" is executed, and that WR12 = 800_{16} and the memory word at address 600 contains the bit pattern shown in Figure 5-7. In the first CRU output machine successive machine cycle the address is incremented by one and the next least-significant bit of the operand is output on CRUOUT, until 10 bits have been output. It is important to note that the CRU base address is unaltered by the LDCR instruction, even though the address is incremented as each successive bit is output.

5.3.3.3 STCR Instruction. The STCR instruction causes from 1 to 16 bits of CRU data to be transferred into memory. Each bit is input by a CRU input machine cycle.

Consider the circuit shown in Figure 5-8. The CRU interface logic multiplexes input signals m-t onto the CRUIN line for addresses 200_{16}–207_{16}. If WR12 @ 400_{16} when the instruction "STCR @ 602,6" is executed, the operation is performed as shown in Figure 5-9. At the end of the instruction, the six LSBs of memory byte 602 are loaded with m-r. The upper bits of the operand are forced to zero. The generation of SEL200 may be simplified if the total CRU address space is not used.

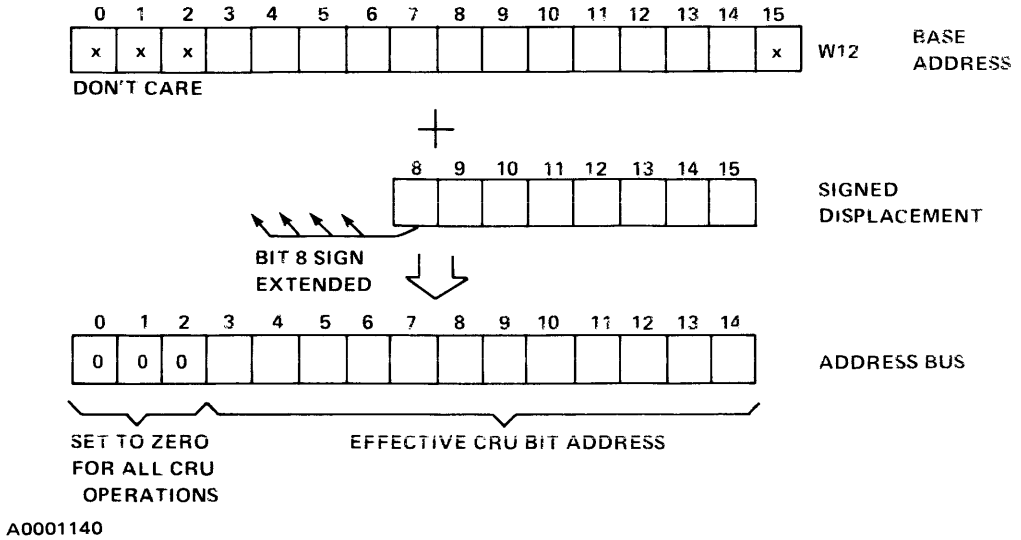


Figure 5-6. TMS 9900 Single-Bit CRU Address Development

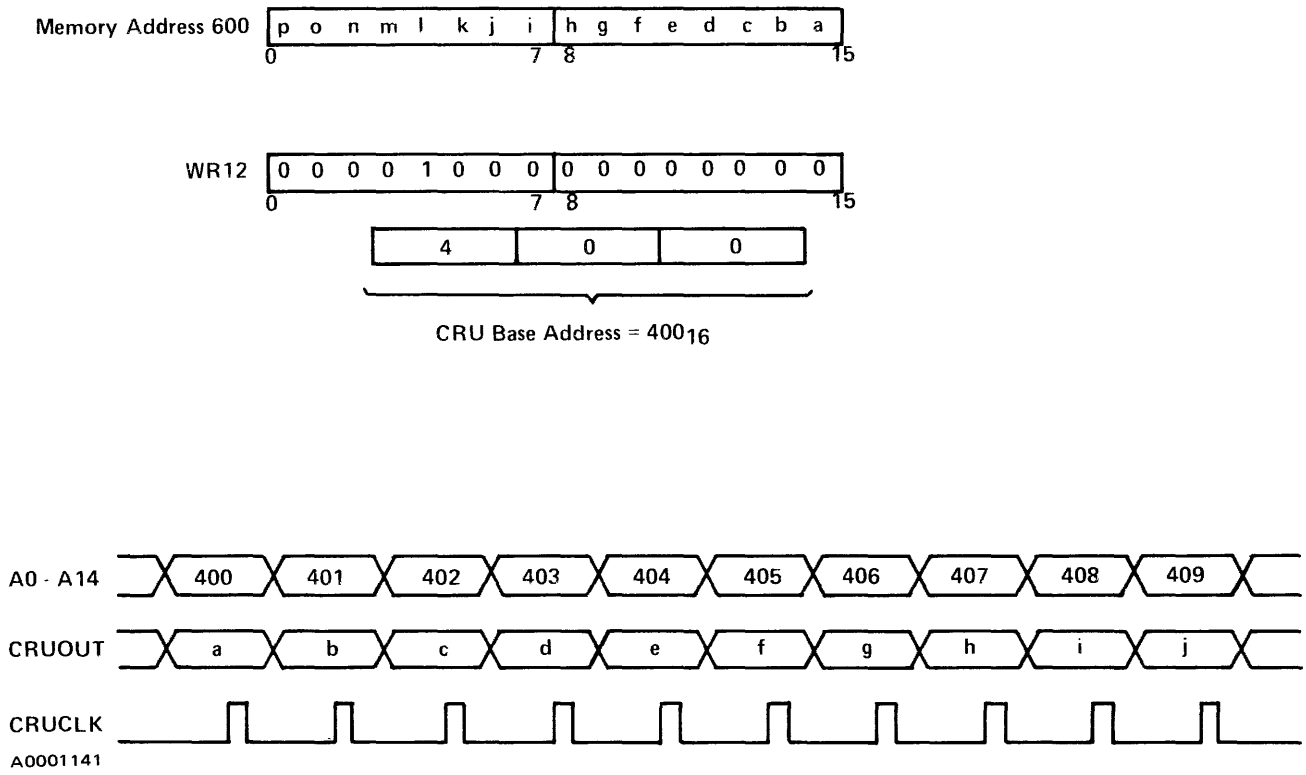


Figure 5-7. Multiple-Bit CRU Output

5.3.4 CRU Interface Logic

CRU based I/O interfaces are easily implemented using either TMS CRU peripheral devices such as the TMS 9901 or the TMS 9902, or TTL multiplexers and addressable latches, such as the TIM 9905 (SN74LS251) and the TIM 9906 (SN74LS259). These I/O circuits can be easily cascaded with the addition of simple address decoding logic.

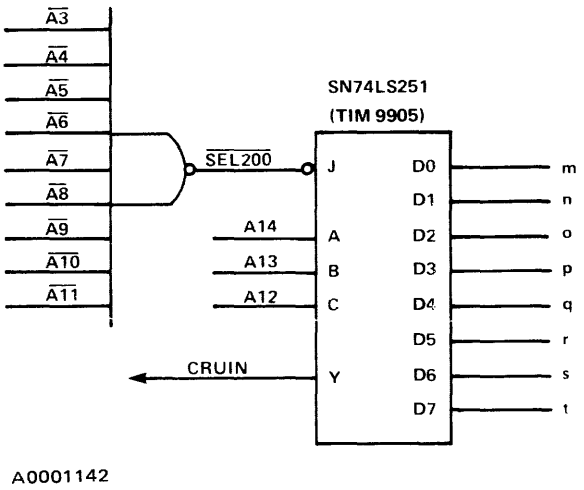


Figure 5-8. Example CRU Input Circuit

5.3.4.1 TTL Outputs. The TIM 9906 (SN74LS259) octal-addressable latch can be used for CRU outputs. The latch outputs are stable and are altered only when the CRUCLK is pulsed during a CRU output transfer. Each addressable latch is enabled only when addressed as determined by the upper address bits. The least-significant address bits (A12–A14) determine which of the eight outputs of the selected latch is to be set equal to CRUOUT during CRUCLK, as shown in Figure 5-10.

5.3.4.2 TTL Inputs. The SN74LS151 and TIM 9905 (SN74LS251) octal multiplexers are used for CRU inputs, as shown in Figure 5-11. The

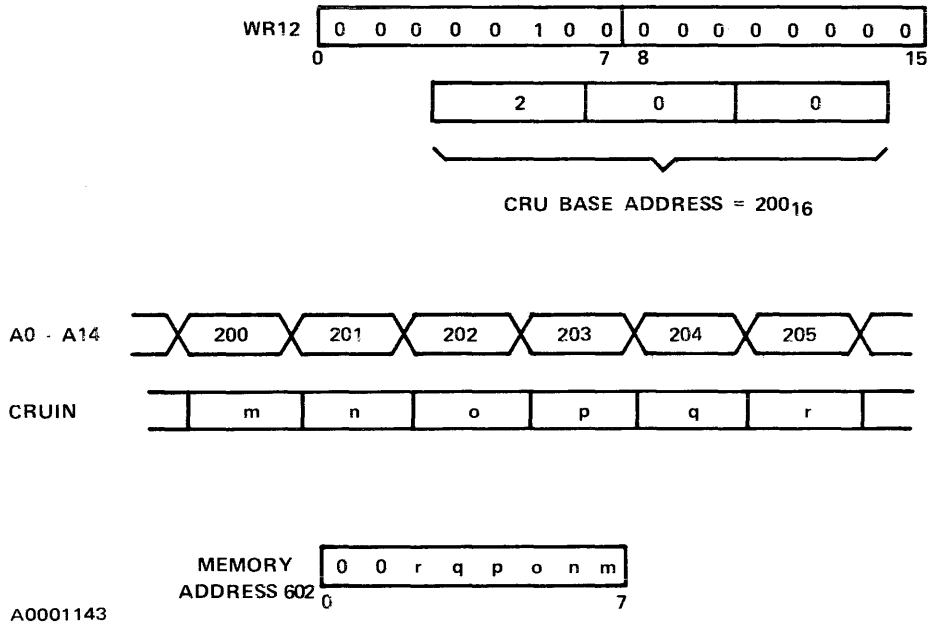


Figure 5-9. Multiple-Bit CRU Input

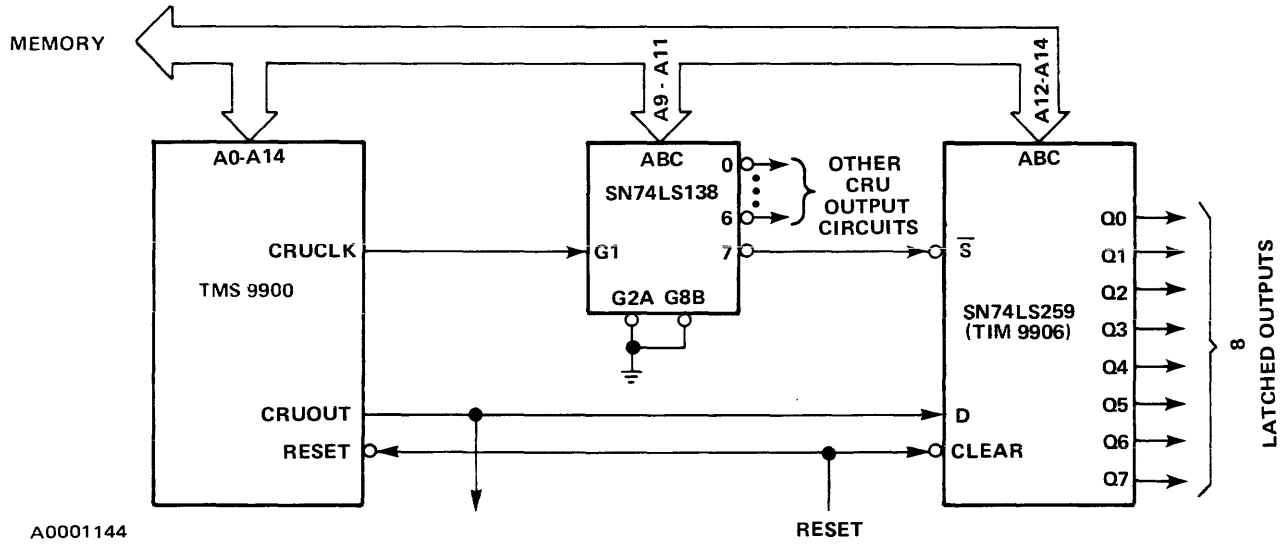


Figure 5-10. Latched CRU Interface

multiplexers are continuously enabled with CRUIN equal to the addressed input. The TIM 9905 should be used for larger systems since its three-state outputs permit simple "wire-ORing" of parallel-input multiplexers.

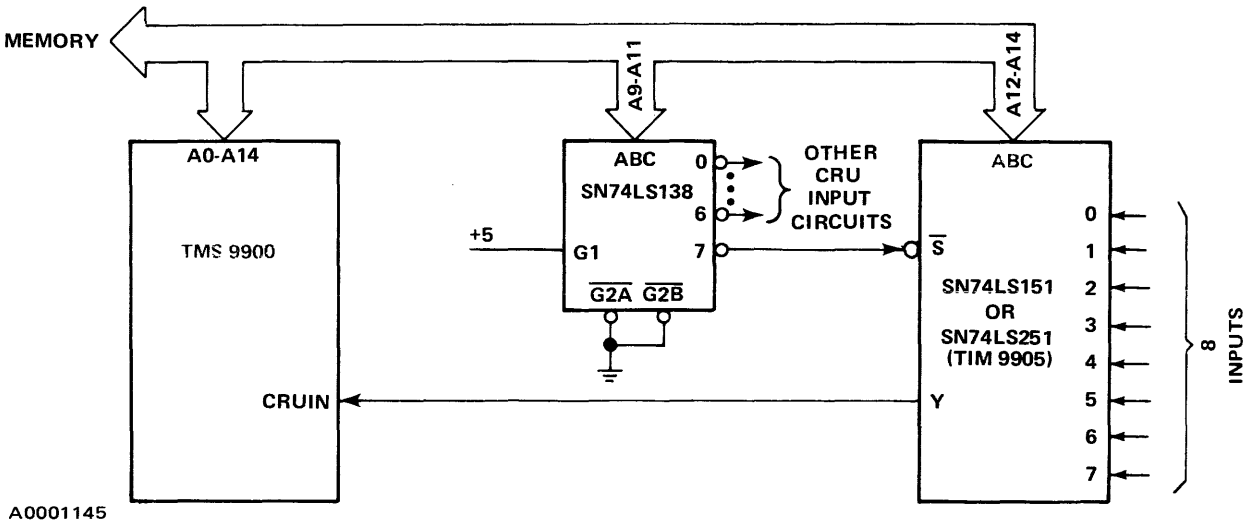


Figure 5-11. Multiplexer CRU Interface

5.3.4.3 Expanding CRU I/O. A CRU interface with eight inputs and eight outputs is shown in Figure 5-12, using the TMS 9901. An expanded interface with 16 inputs and 16 outputs is shown in Figure 5-13, using TTL devices. The CRU inputs and outputs can be expanded up to 4096 inputs and 4096 outputs by decoding the complete CRU address. Larger I/O requirements can be satisfied by using memory mapped I/O or by using a CRU bank switch, which is set and reset under program control. When reset, the lower CRU I/O bank is selected, and when set, the upper CRU I/O bank is selected. In actual system applications, however, only the exact number of interface bits required need to be implemented. It is not necessary to have a 16-bit CRU output register to interface a 10-bit device.

5.4 CRU Paper Tape Reader Interface

CRU interface circuits are used to interface data and control lines from external devices to the TMS 9900. This section describes an example interface for a paper tape reader.

The paper tape reader is assumed to have the following characteristics:

1. It generates a TTL-level active-high signal (SPROCKET HOLE) on detection of a sprocket hole on the paper tape.

2. It generates an 8-bit TTL active-low data which stays valid during SPROCKET HOLE = 1.

3. It responds to a TTL-level active-high command (Paper Tape RUN) signal by turning on when PTRUN = 1 and turning off when PTRUN = 0.

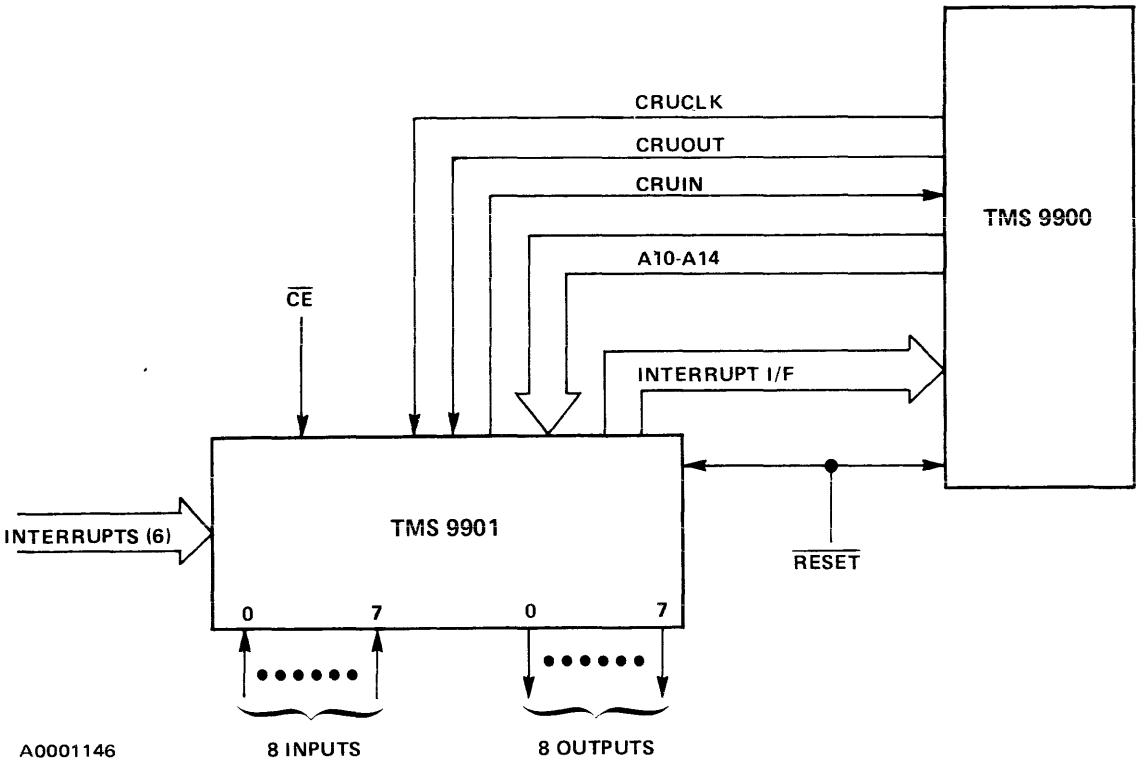


Figure 5-12. 8-Bit CRU Interface

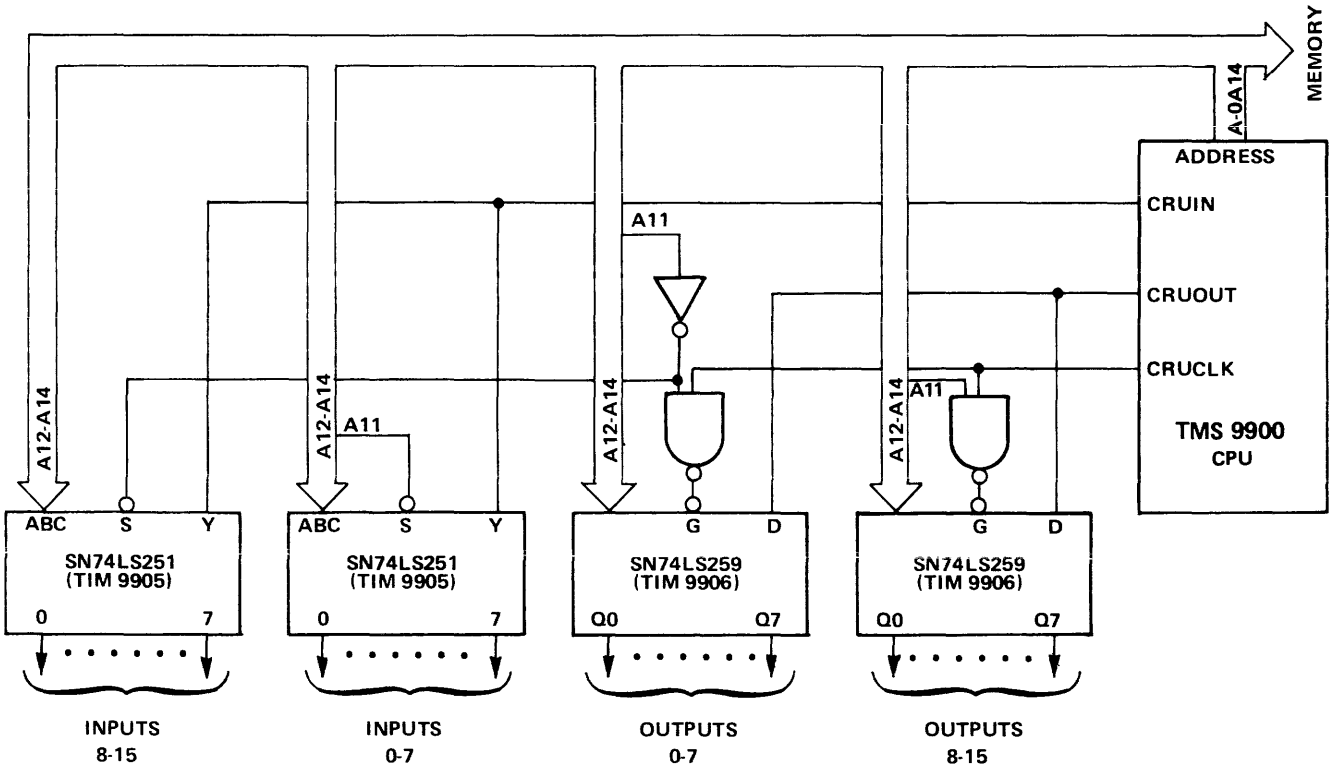


Figure 5-13. 16-Bit CRU Interface

5.4.1 Operation

Figure 5-14 illustrates the circuitry to interface the reader to the CRU. The interface is selected when $\overline{\text{PTRSEL}}$ = 0; $\overline{\text{PTRSEL}}$ is decoded from the A0–A11 address outputs from the TMS 9900. Thus, the output of the SN74LS251 is active only when $\overline{\text{PTRSEL}}$ = 0; otherwise, the output is in high impedance and other devices may drive CRUIN. The data inputs are selected by A12–A14 and inverted, resulting in active high data input on CRUIN. The positive transition of SPROCKET HOLE causes $\overline{\text{PTRINT}}$ to go low. $\overline{\text{PTRINT}}$ is the active low interrupt from the interface. $\overline{\text{PTRINT}}$ is set high, clearing the interrupt, whenever a CRU output machine cycle is executed and the address causes $\overline{\text{PTRSEL}}$ to be active. When a one is output, PTRUN is set, enabling the reader, and the reader is disabled when a zero is output to the device. Thus, any time PTRUN is set or reset, the interrupt is automatically cleared.

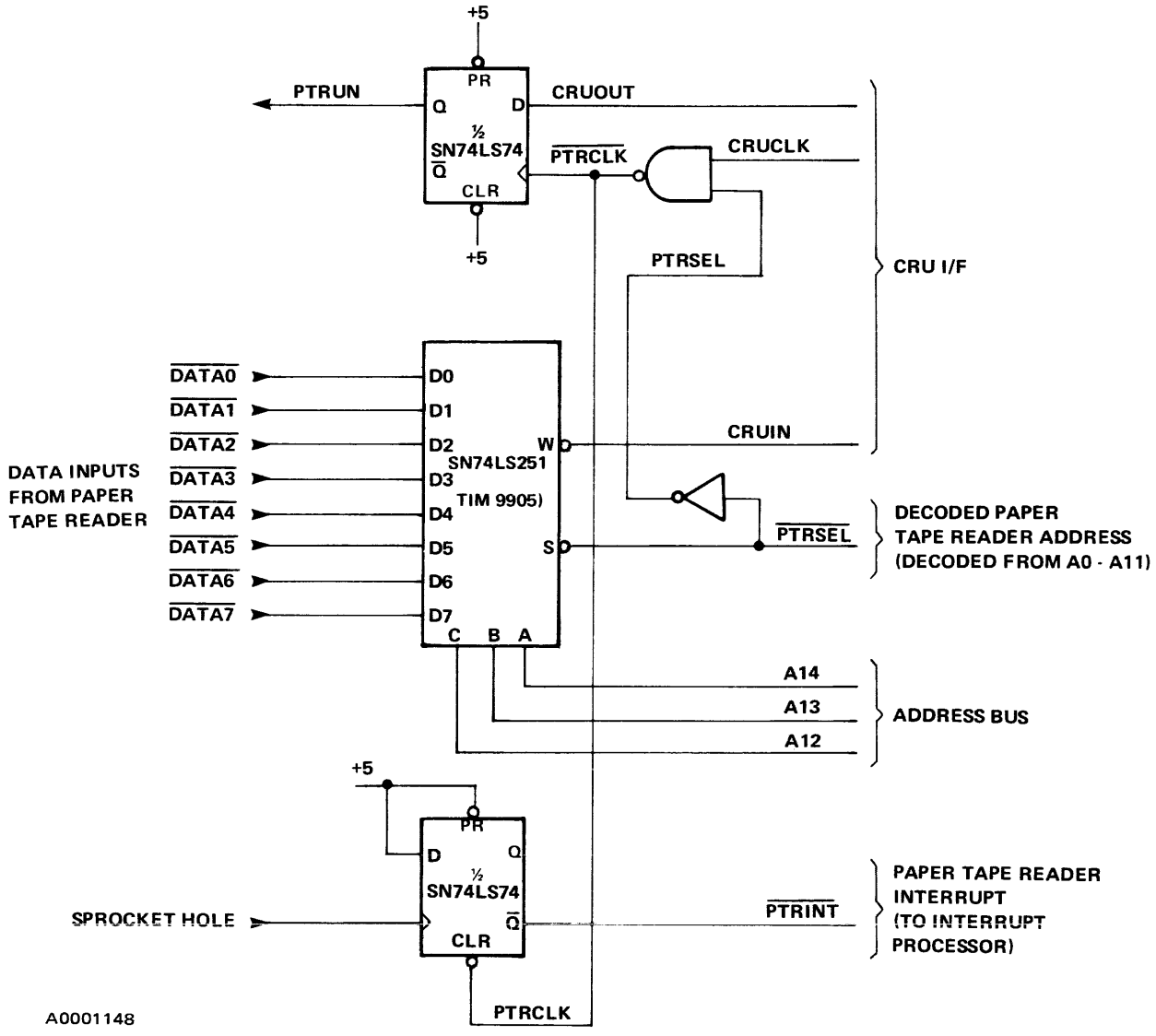


Figure 5-14. Paper Tape Reader Interface

5.4.2 Software Control

The software routine in Figure 5-15 controls the paper-tape reader interface described above. It is a re-entrant procedure that can be shared by several readers. The assumptions are that:

1. Each reader has its own workspace which is set up on the trap location for that reader's interrupt.

2. The workspace registers are allocated as shown in Figure 5-15.

3. The CRU input bits 0–7 (relative to CRU base) are reader data. CRU output bit 0 controls PTRUN and clears the interrupt.

The procedure has two entry points. It is entered by a calling routine at PTRBEG to start the reader and it returns control to that routine. It is entered at PTRINT via interrupt to read a character. The return in this case is to the interrupted program.

```
PTRINT    STCR    *R11,8
          CB      *R11+,R9
          JEQ     PTREND
          DEC     R10
          JEQ     PTREND
PTRBEG    SBO     PTRUN
          RTWP
PTREND    SBZ     PTRUN
          CLR     R8
          RTWP
A0001149
```

Figure 5-15. Paper Tape Reader Control Program

The control program may be used by any number of paper-tape reader interfaces, as long as each interface has a separate interrupt level and workspace. As each reader issues an interrupt, the TMS 9900 will process the interrupt beginning at location PTRINT. However, the workspace unique to the interrupting device is used. The organization of memory to control two paper tape readers is shown in Figure 5-16. The interrupt-transfer vector causes the appropriate WP value to be loaded. In both cases PTRINT, the entry point for the control program, is loaded into the PC.

5.5 TMS 9902 Interface

This section describes an example CRU interface using the TMS 9902 asynchronous communications controller. The TMS 9902 transmits and receives data over a serial communications link. It performs its own baud rate generation, parity and format generation, and error checking, and additionally provides an internal interval timer with interrupt capability. The TMS 9902 interfaces directly with the TMS 9900 family CRU with a minimum of additional logic required only for chip-select decode.

5.5.1 Operation

Figure 5-17 illustrates the circuit for the TMS 9902 interface. The CPU uses LDCR and STCR instructions to transfer character data, and TB, SB0, and SB2 instructions to test and to alter the status and control bits.

The TMS 9902 sets RBRL to a logic one and generates an interrupt request via the TMS 9901 programmable systems interface whenever a received character is ready to be read. It correspondingly sets

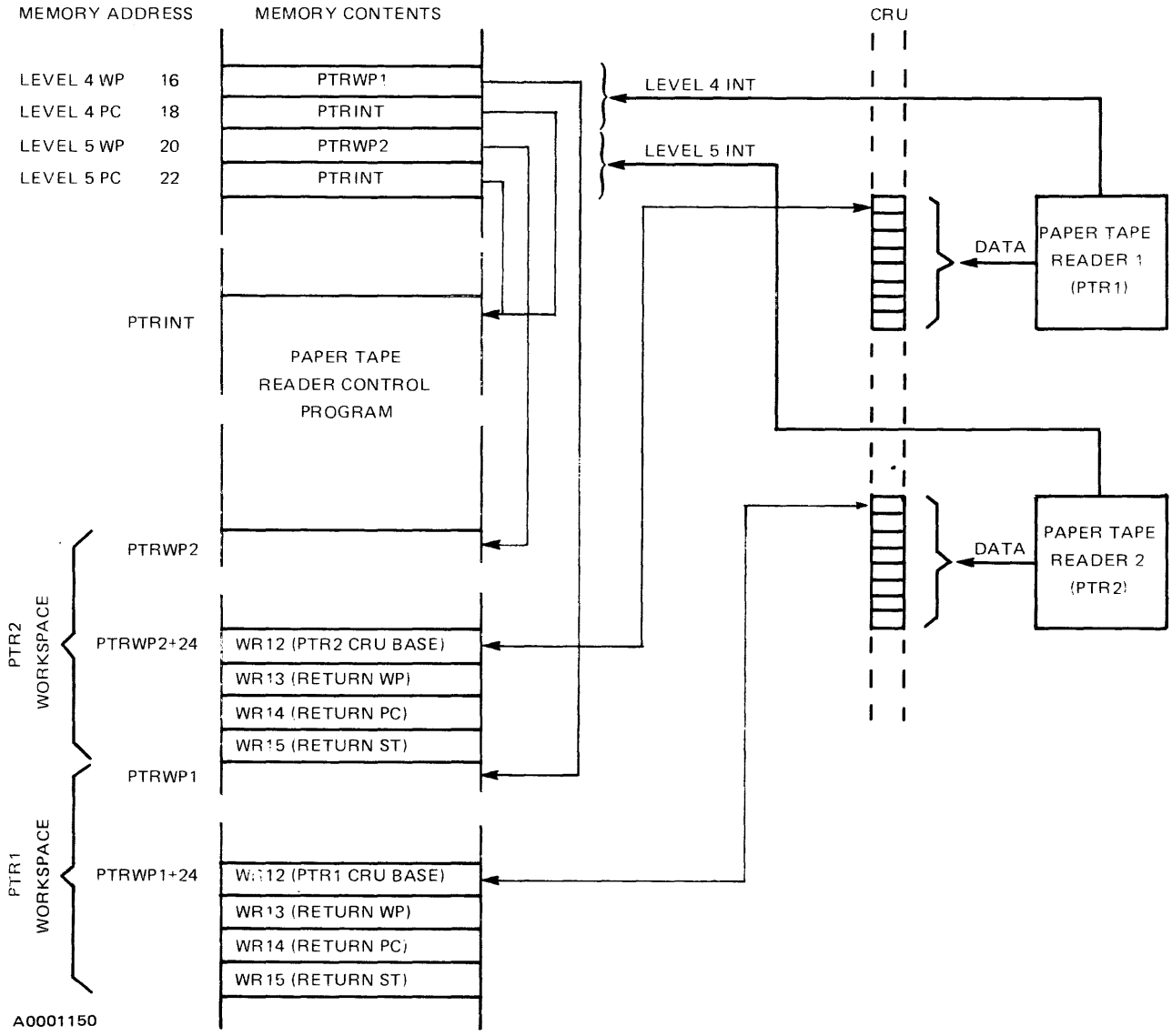


Figure 5-16. Software Configuration for Two Paper Tape Readers with Common Control Program

TBRE to a logic one and generates an interrupt in a transmit operation when a character can be loaded into the transmit buffer. The CPU recognizes $\overline{\text{INTREQ}}$ and context switches to the service routine when the interrupt code is less than or equal to the CPU interrupt mask. Data is transferred and the interrupts are cleared in the service routines, and processing of the interrupted routine is resumed at the point of interruption. While a number of techniques for hardware-interrupt processing and prioritizing exist, the TMS 9901 programmable system interface is used in this example due to its ease of design and its simplicity of interconnection.

The XOUT and RIN signals are buffered through SN75188 and SN75189 interface circuits to provide an RS-232C compatible interface. The 75188 and SN75189 interface devices are also used to condition the modem control signals $\overline{\text{RTS}}$, $\overline{\text{CTS}}$, and $\overline{\text{DSR}}$.

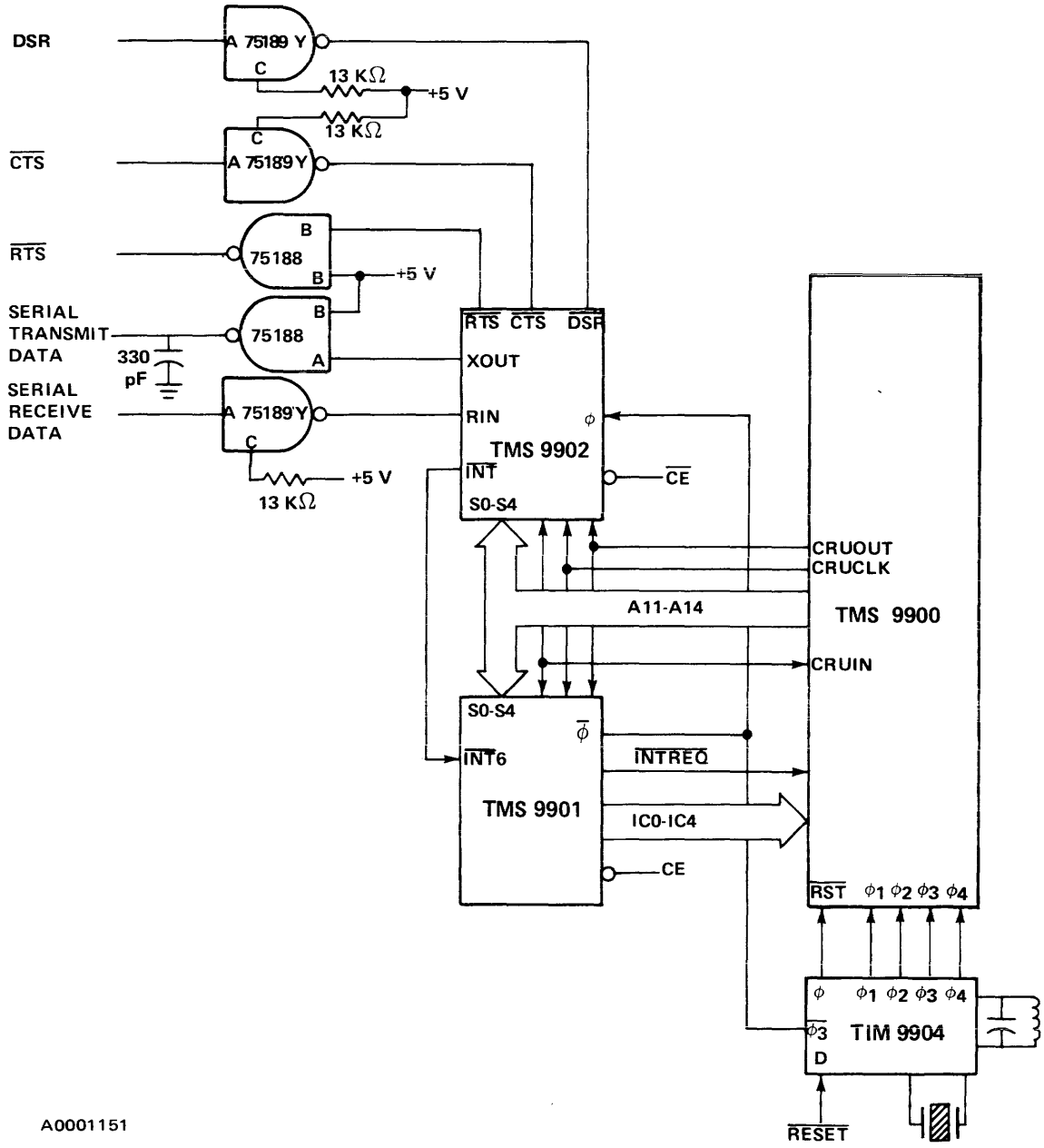


Figure 5-17. TMS 9902 Interface

5.5.2 Software Routines

The software routines in Figure 5-18 control the TMS 9902. The initialization routine can be a part of the normal system power-up reset. Upon being reset the device is cleared and all load flags are set. Loading the control register with $A2_{16}$ sets the device for seven bits per character, even parity, one stop bit, and sets the internal clock rate at one-third the system clock rate. Loading the control register also resets the load control register flag, setting up the next register, the timer interval register, to be loaded. In this example,

```
DEVICE
INITIALIZATION

            LI      R12, >40        INITIALIZE CRU BASE
            SBO     31              RESET DEVICE AND SET LOAD FLAGS
            LDCR    @ CTL, 8        LOAD CONTROL REGISTER AND RESET CTL LOAD FLAG
            SBZ     13              RESET INTERVAL LOAD FLAG
                                    LOAD READ DATA RATE REG.,
            LDCR    @ RDR, 12       LOAD XMT DATA RATE REG. & RESET LRDR, LXDR
              :
CTL         BYTE    >A2             7 BITS/CHAR, EVEN PARITY, 1 STOP BIT, 0 DIVIDE BY 3
RXDR        BYTE    <4D0            300 BAUD OPERATION

TRANSMIT OPERATION INITIALIZATION

XMTOP       LI      R0, DATAD       INITIALIZE DATA ADDRESS
            LI      R1, DATCNT      INITIALIZE DATA COUNT
            LI      R12, CRUBAS     INITIALIZE CRU BASE
            SBO     16              TURN TRANSMITTER ON
            SBO     19              ENABLE TRANSMIT INTERRUPTS
            B       *R11            RETURN

RECEIVE OPERATION INITIALIZATION

RCVOP       LI      R1, RCVBUF      INITIALIZE BUFFER ADDRESS
            LI      R2, BLVCNT      INITIALIZE RCV COUNT
            LI      R3, EOB         INITIALIZE EOB CHARACTER
            SBO     18              ENABLE RCV INTERRUPTS
            B       *R11            RETURN
EOB         BYTE    >OD             END OF BLOCK CHARACTER

INTERRUPT SERVICE ROUTINE

INT6        TB      21              RCVR INTERRUPT?
            JEQ     RCV             YES
            TB      22              XMTR INTERRUPT?
            JEQ     XMT             YES
            RTWP                    NOT EITHER, GO HOME
XMT         LDCR    *R0+, 8         TRANSMIT CHARACTER
            DEC     R1              DECREMENT COUNTES
            JEQ     XMTEND          END IT IF FINISHED
            RTWP                    RETURN
XMTEND      SBZ     19              MASK XBRE INTERRUPT
            RTWP                    RETURN
RCV         TB      9               RECEIVE ERROR?
            JEQ     RCVERR          YES, GO PROCESS IT
            STCR    *R1,8           STORE CHARACTER
            SBZ     18              RESET RBRL
            DEC     R2              BLOCK FULL
            JEQ     RCVEND          YES, EXIT
            CB      *R1+, R3        NO; EOB CHARACTER?
            JEQ     RCVEND          YES; EXIT
            RTWP                    GO HOME
RCVEND      SBZ     18              INHIBIT RCV INTERRUPTS
            RTWP
A0001152
```

Figure 5-18. TMS 9902 Control Programs

the timer is not used so the load timer register flag is reset, enabling the loading of the receive data rate register and the transmit data rate register. Here both transmit and receive data rates are set for 300-baud operation.

To initiate an input or output operation the receive or transmit initialization routines are executed to set up the desired operation. From that point the operation is interrupt driven by the $\overline{INT}$ signal from the TMS 9902.

5.6 Software – UART

Some systems may not require the full capabilities offered by an external UART. For example, a small system can be interfaced to a 733 ASR for initial program loading only. The transmission is used in half-duplex and the program has no significant tasks to perform between input characters. A software-UART function can be used in which the program serially inputs and outputs bits with software format conversion. The system in Figure 5-19 uses a software-UART program to minimize the system component count at the expense of added software.

The software-UART consists of a receive routine and a transmit routine which are called by XOP or BLWP instructions. The routines share a common workspace and a common delay subroutine. The seven-bit plus parity transmit and receive characters are passed through register 0 of the calling program. Since the software-UART routines must run to completion, the interrupt mask is cleared to disable all maskable interrupts. Registers 4, 5, 8, and 10 are used for temporary results. Registers 6, 7, 9, and 12 are initialized by the system restart routine. The delay routines introduce software determined delays equal to half-bit and full-bit times for a 300 bps data terminal. The receive routine rejects input characters with bad parity, false start bits, or missing stop bits. The software-UART routines are shown in Figure 5-20.

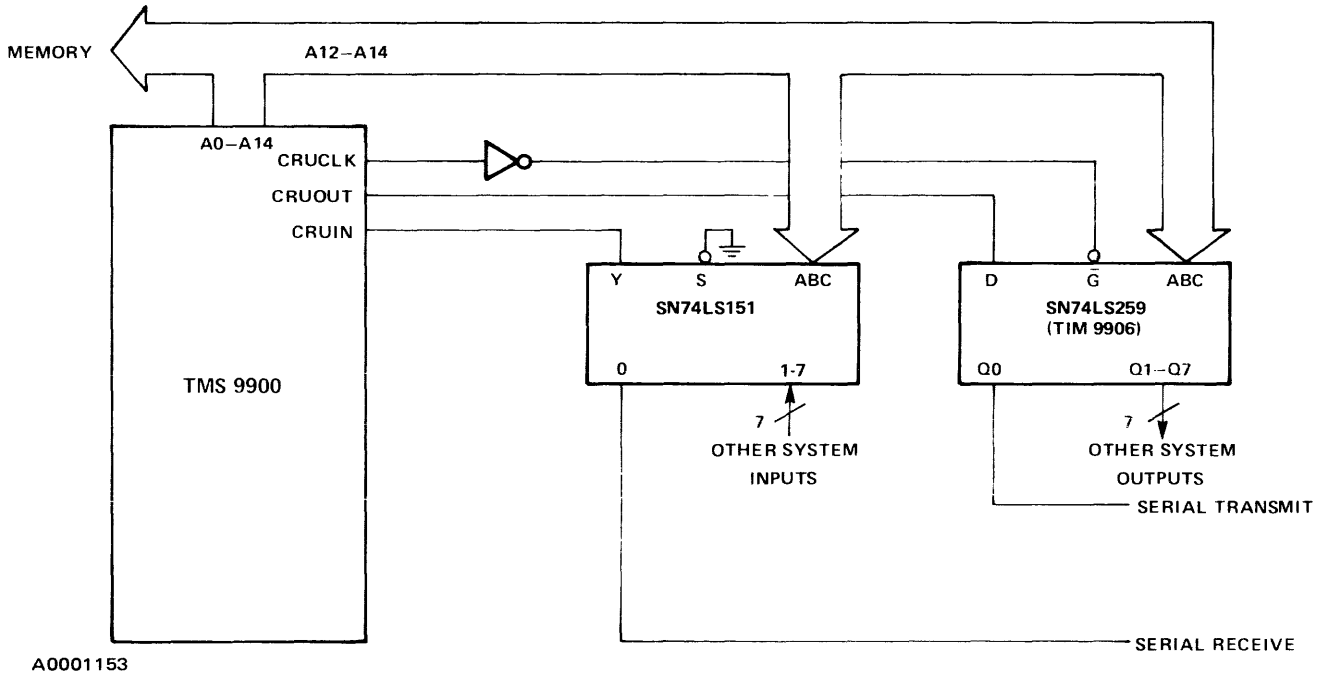


Figure 5-19. Software UART CRU Interface

```
                                                        PAGE 0001

0001                    IDT  'SWUART'          SOFTWARE UART CONTROL PROGRAM
0002                  *
0003                  * REGISTER EQUATES
0004                  *
0005          0004    R4      EQU  4            LOOP COUNTER
0006          0005    R5      EQU  5            BIT COUNT
0007          0006    R6      EQU  6            HALF-BIT DELAY LINKAGE
0008          0007    R7      EQU  7            FULL-BIT DELAY LINKAGE
0009          0008    R8      EQU  8            CHARACTER ACCUMULATOR
0010          0009    R9      EQU  9            BIT MASK ( >8000 )
0011          000A    R10     EQU  10           SCRATCH
0012          000B    R11     EQU  11           LINKAGE
0013          000C    R12     EQU  12           CRU BASE
0014          000D    R13     EQU  13           RETURN WP
0015                  *
0016                  * CRU EQUATES
0017                  *
0018          0000    CRUBAS  EQU  0            CRU BASE ADDRESS
0019          0000    INBIT   EQU  0            RECEIVE INPUT
0020          0000    OUTBIT  EQU  0            TRANSMIT OUTPUT
0021                  *
0022                  * HALF-BIT AND FULL-BIT TIME DELAY SUBROUTINES
0023                  *
0024                  * ENTRY:  BL   *R6  FOR HALF-BIT DELAY
0025                  *         BL   *R7  FOR FULL-BIT DELAY
0026                  *
0027                  * R6 = HBIT, R7 = FBIT
0028                  *
0029          00F8    HBDLY   EQU  248          HALF-BIT DELAY COUNT
0030          0000'   FBIT    EQU  $            FULL-BIT DELAY ENTRY
0031    0000  C28B            MOV  R11,R10      SAVE LINKAGE
0032    0002  0696            BL   *R6          CALL HALF-BIT DELAY
0033    0004  C2CA            MOV  R10,R11      RESTORE LINKAGE
0034          0006'   HBIT    EQU  $            HALF-BIT DELAY ENTRY
0035    0006  0204            LI   R4,HBDLY     INITIALIZE LOOP COUNTER
        0008  00F8
0036    000A  0604    DLOOP   DEC  R4           DECREMENT COUNT
0037    000C  16FE            JNE  DLOOP        LOOP UNTIL END OF DELAY
0038    000E  045B            RT                RETURN
0040                  *
0041                  * RECEIVER CONTROL PROGRAM
0042                  *
0043                  * ENTRY:  BLWP @RCVVCT
0044                  *
0045                  * UPON RETURN TO CALLING PROGRAM, THE LEFT BYTE OF
0046                  * WR0 OF THE CALLING PROGRAM CONTAINS THE RECEIVED
0047                  * CHARACTER.  CHARACTERS WITH PARITY OR FRAMING
0048                  * ERRORS ARE IGNORED.
0049                  *
0050          0010    RCV     EQU  $            RECEIVE ENTRY POINT
0051    0010  0300            LIMI 0            MASK ALL INTERRUPTS
        0012  0000
0052    0014  0205            LI   R5,8         INITIALIZE BIT COUNT
        0016  0008
0053    0018  1F00    RLOOP1  TB   INBIT        TEST FOR START BIT
0054    001A  13FE            JEQ  RLOOP1       NO,CONTINUE TESTING
0055    001C  0696            BL   *R6          DELAY HALF-BIT TIME
0056    001E  1F00            TB   INBIT
0057    0020  13FB            JEQ  RLOOP1       IF FALSE START BIT, START OVER
0058    0022  0697    RLOOP2  BL   *R7          DELAY FULL-BIT TIME
0059    0024  0919            SRL  R9,1         SHIFT FOR NEXT BIT
0060    0026  1F00            TB   INBIT        READ INPUT
0061    0028  16--            JNE  RSKIP        IF INPUT IS 0,SKIP
0062    002A  E209            SOC  R9,R8        ELSE SET MSB
0063    002C  0605    RSKIP   DEC  R5           LAST DATA BIT?
        002E**1601
0064    002E  16F9            JNE  RLOOP2       NO,LOOP
0065    0030  0697            BL   *R7          YES, DELAY FULL-BIT TIME
0066    0032  1F00            TB   INBIT        TEST FOR STOP BIT
0067    0034  16ED            JNE  RCV          IF ERROR, IGNORE CHARACTER
0068    0036  D748            MOVB R8,*R13      IF NO ERROR, STORE CHARACTER
0069    0038  1C--            JOP  REXIT        IF PARITY ERROR,
0070    003A  10EA            JMP  RCV          IGNORE CHARACTER
0071    003C  4749    REXIT   SOC  R9,*R13      MASK PARITY BIT
        003E**1C01
0072    0040  045B            RTWP              RETURN
```

A0001154

Figure 5-20. Software UART Control Program (Page 1 of 2)

```
0074                    *
0075                    * TRANSMITTER CONTROL PROGRAM
0076                    *
0077                    * ENTRY:  BLWP @XMTVCT
0078                    *
0079                    * THE LEFT BYTE OF WR0 OF THE CALLING PROGRAM
0080                    * IS SERIALLY TRANSMITTED, PRECEDED BY A START
0081                    * BIT, CORRECTED FOR PARITY, AND FOLLOWED
0082                    * BY A STOP BIT.
0083                    *
0084  0040  0300  XMT    LIMI 0                  MASK ALL INTERRUPTS
      0042  0000
0085  0044  04C8        CLR  R8                 CLEAR CHARACTER ACCUMULATOR
0086  0046  0205        LI   R5,10              LOAD BIT COUNT
      0048  000A
0087  004A  D21D        MOVB *R13,R8            FETCH CHARACTER
0088  004C  1C--        JOP  XSKIP              CORRECT PARITY
0089  004E  2A09        XOR  R9,R8
0090  0050  0918  XSKIP  SRL  R8,1
      004C**1C01
0091  0052  E209        SOC  R9,R8              APPEND STOP BIT
0092  0054  0968        SRL  R8,6               RIGHT JUSTIFY CHARACTER
0093  0056  0918  XLOOP  SRL  R8,1               SHIFT OUT LSB
0094  0058  18--        JOC  XONE               IF ONE, SET OUTBIT TO ONE
0095  005A  1E00        SBZ  OUTBIT             ELSE, SET TO ZERO
0096  005C  10--        JMP  XNEXT
0097  005E  1D00  XONE   SBO  OUTBIT
      0058**1802
0098  0060  0697  XNEXT  BL   *R7                DELAY FULL-BIT TIME
      005C**1001
0099  0062  0605        DEC R5                  LAST BIT?
0100  0064  16F8        JNE  XLOOP              NO, CONTINUE
0101  0066  0380        RTWP                    YES, RETURN
0103                    *
0104                    * INITIALIZATION PROGRAM
0105                    *
0106                    * ENTRY:  BLWP @ENTVCT
0107                    *
0108                    * REGISTERS ARE INITIALIZED
0109                    *
0110        0068' ENTER  EQU  $                  ENTRY POINT
0111  0068  0206        LI   6,HBIT             HALF-BIT SUBROUTINE
      006A  0006'
0112  006C  0207        LI   7,FBIT             FULL-BIT SUBROUTINE
      006E  0000'
0113  0070  0209        LI   9,>8000            BIT MASK
      0072  8000
0114  0074  020C        LI   12,CRUBAS          CRU BASE ADDRESS
      0076  0000
0115  0078  045B        RTWP                    RETURN
0116                    *
0117                    * PROGRAM WORKSPACE
0118                    *
0119  007A        UARTWP BSS  32
0120                    *
0121                    * ENTRY TRANSFER VECTORS
0122                    *
0123  009A  007A' ENTVCT DATA UARTWP,ENTER       INITIALIZATION ENTRY VECTOR
      009C  0068'
0124  009E  007A' RCVVCT DATA UARTWP,RCV         RECEIVER ENTRY VECTOR
      00A0  0010'
0125  00A2  007A' XMTVCT DATA UARTWP,XMT         TRANSMIT ENTRY VECTOR
      00A4  0040'
0126                    END                     END OF PROGRAM
```

A0001154

Figure 5-20. Software UART Control Program (Page 2 of 2)

5.7 Burroughs SELF-SCAN Display Interface

This section describes a TMS 9900 CRU interface to a Burrough's SELF-SCAN® panel display model SS30132-0070. The display panel has a 32-position, single-row character array with a repertoire of 128 characters.

The panel display operates in a serial-shift mode in which characters are shifted into the panel one at a time. Characters are shifted in right-to-left and can be shifted or backspaced left-to-right. A clear pulse erases the display.

The CRU display interface is shown in Figure 5-21 and a display control subroutine is shown in Figure 5-22. The subroutine is called by one of two XOP instructions, XOP0 and XOP1. The calling routine passes

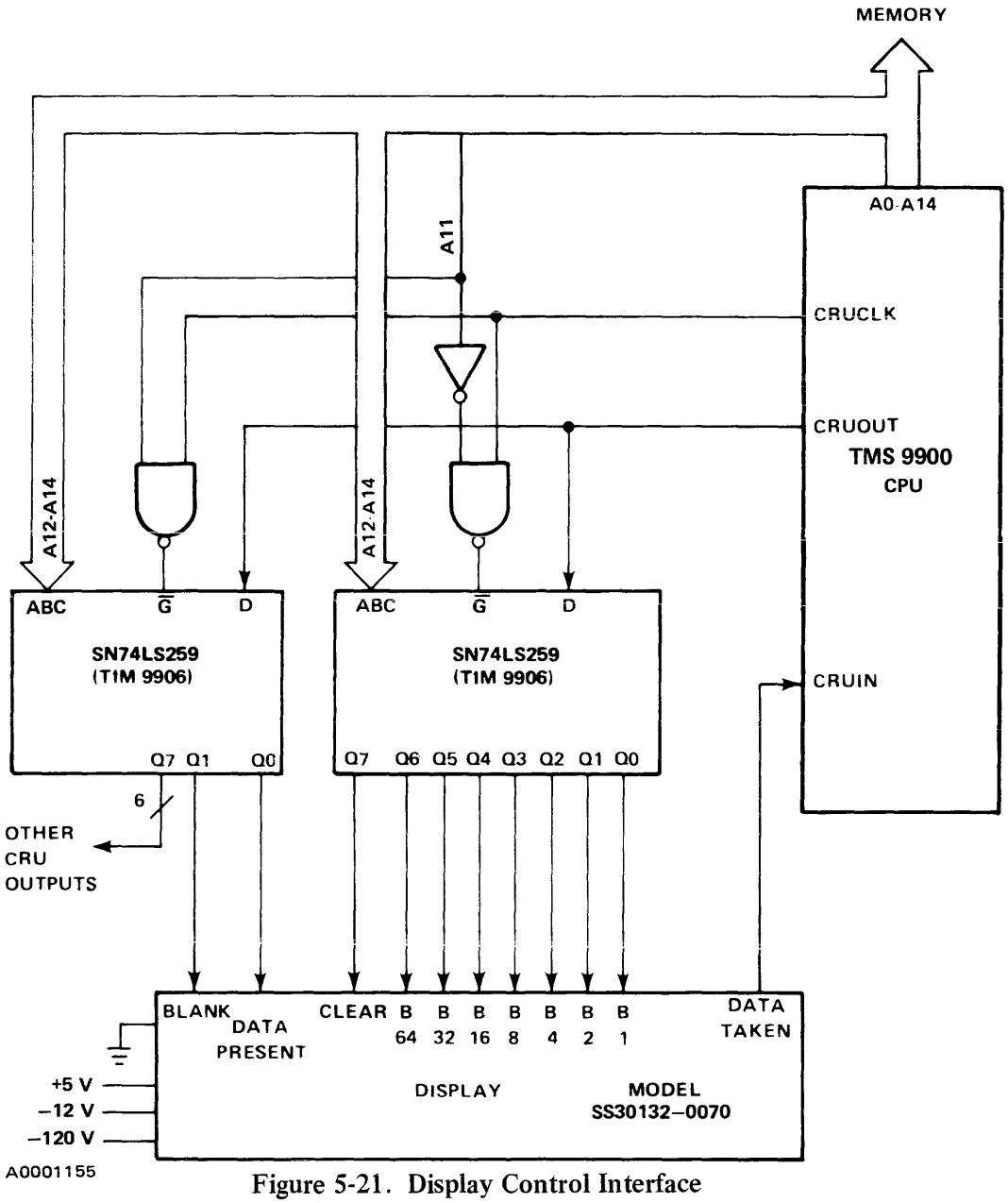


Figure 5-21. Display Control Interface

the address and length of the output string in registers 8 and 9 of its workspace. The two XOP subroutines share the same workspace and perform the same function except that XOP1 clears the panel display first. The backspace feature is not used. The panel display is blanked during character entry.

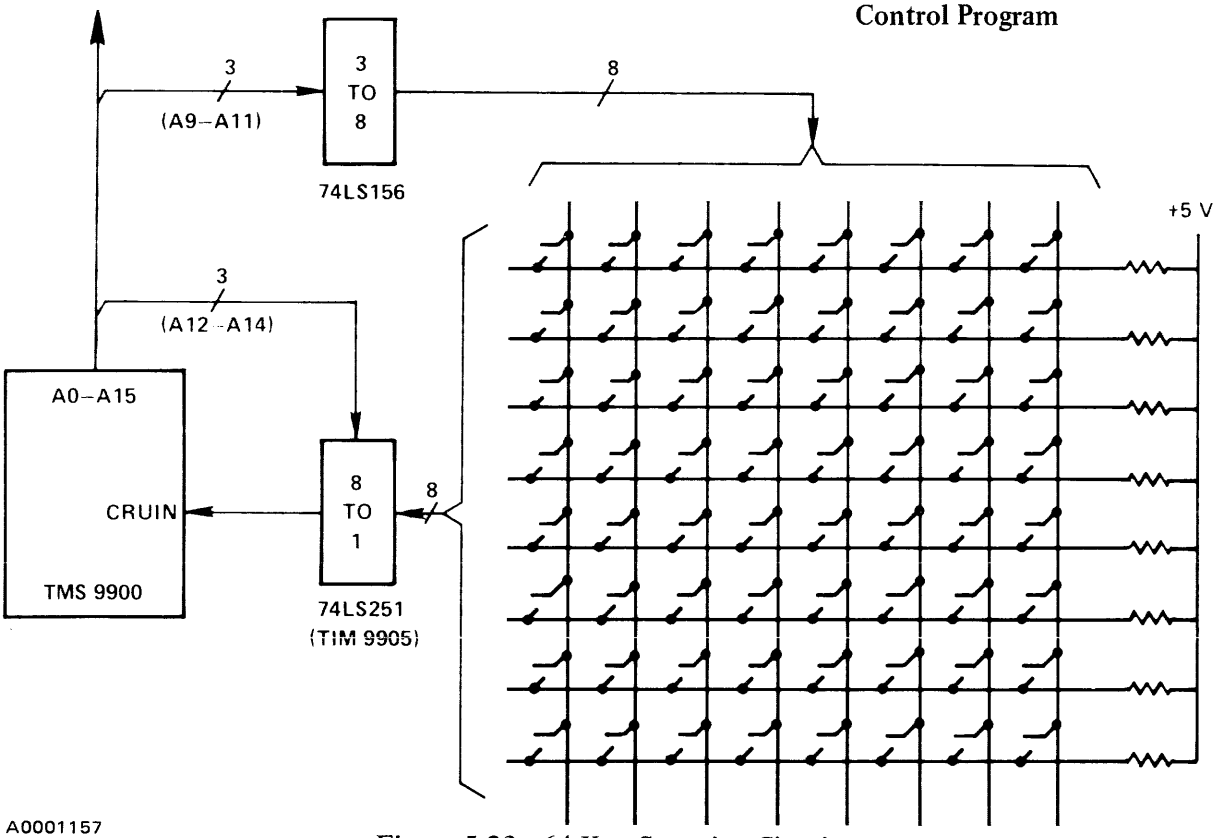
5.8 Matrix Keyboard Interface

Figure 5-23 shows how two 16-pin devices and eight resistors can be used to interface a 64-key keyboard connected as a row-column matrix. This circuit requires a 64-bit CRU input-address space. Columns are selected by A9–A11, and specific keys in each column are addressed by A12–A14. Therefore, individual keys may be read by the TB (test bit) instruction, or the entire array may be read by four 16-bit STCR instructions, incrementing the CRU base address by 16 between each STCR instruction. Functions such as switch debounce and key rollover may be performed in software.

```
EIGHT   EQU   8
NINE    EQU   9

RXOP1   SBZ   7              Clear Panel
        LI    R1,11

LOOP1   DEC   R1             Delay >67μsec
        JNE   LOOP1
        SBO   7
RXOP2   SBO   9              Blank Panel
        MOV   @EIGHT (13),1  Load Address
        MOV   @NINE (13),2   Load Length

LOOP2   LDCR  *1+,7          Output Char
        SBO   8              Data Present

WAIT    TB    0              Wait for Data Taken
        JEQ   Wait
        SBZ   8
        DEC   2              Decrement Count
        JNE   LOOP2          Loop Until Through
        SBZ   9              Unblank Panel
        RTWP                 Return
```

A0001156

Figure 5-22. Burrough's SELF-SCAN Display Control Program



A0001157

Figure 5-23. 64-Key Scanning Circuit

SECTION VI

AUXILIARY SYSTEM FUNCTIONS

This section describes the circuitry and application of several functions which may be useful in a TMS 9900 system.

6.1 Unused Op Codes

The unused op codes for the TMS 9900 are shown in Table 6-1. An instruction which consists of any of these codes will cause the TMS 9900 to perform no operation other than the updating of the PC to point to the next sequential even word address and the processing of RESET, LOAD, and interrupts in the normal way. It is not recommended that any of these codes be used to perform the function of a no-op, however, since future upward-compatible members of the TMS 9900 microprocessor family will use these codes for presently unimplemented instructions. The TMS 9900 assembler has defined the pseudo-instruction "NOP" to generate a code of 1000_{16}, which is equivalent to the "JMP $+2" instruction.

6.1.1 Unused Op Code Detection

The occurrence of unused op codes may be detected by the circuitry shown in Figure 6-1. The signal ILOPCD is set high at the end of memory cycle which fetches the unused op code. On the first clock cycle of each instruction fetch, the IAQT1 signal is set high. Then, when memory data is read (on the next clock cycle in which the processor is not in a WAIT state) the IAQT1 signal is reset and, if $\overline{\text{ILODEC}} = 0$, the illegal op code flag ILOPCD is set. It may be convenient to use ROM, a PLA, or random logic to generate the signal $\overline{\text{ILODEC}}$, which is low when any of the unused codes is decoded. If WAIT states do not occur, one SN74LS32 gate may be deleted.

6.1.2 Unused Op Code Processing

The occurrence of an unused op code is an error condition which may be caused by device failure, signal noise, or errors in software. Therefore, the error flag generated should be a high-priority (low-level) interrupt which causes the processor to suspend operation before any further instructions are executed. The flag should be cleared by the $\overline{\text{CLILOP}}$ signal only after the cause of the error is determined.

6.2 Software Front Panel

A useful feature during hardware and software development is a means for controlling the operation of the processor at a more basic level than is required during normal system operation. For instance, it is desirable to start and stop execution of a program segment, to execute single instructions, to inspect and to modify memory and internal registers, and to load programs into RAM. These features are associated with the front

TABLE 6-1. UNUSED OP CODES

HEXADECIMAL	BINARY
	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 0 0 0 – 0 1 F F	0 0 0 0 0 0 0 x x x x x x x x x
0 3 2 0 – 0 3 3 F	0 0 0 0 0 0 1 1 0 0 1 x x x x x
0 7 8 0 – 0 7 F F	0 0 0 0 0 1 1 1 1 x x x x x x x
0 C 0 0 – 0 F F F	0 0 0 0 1 1 x x x x x x x x x x

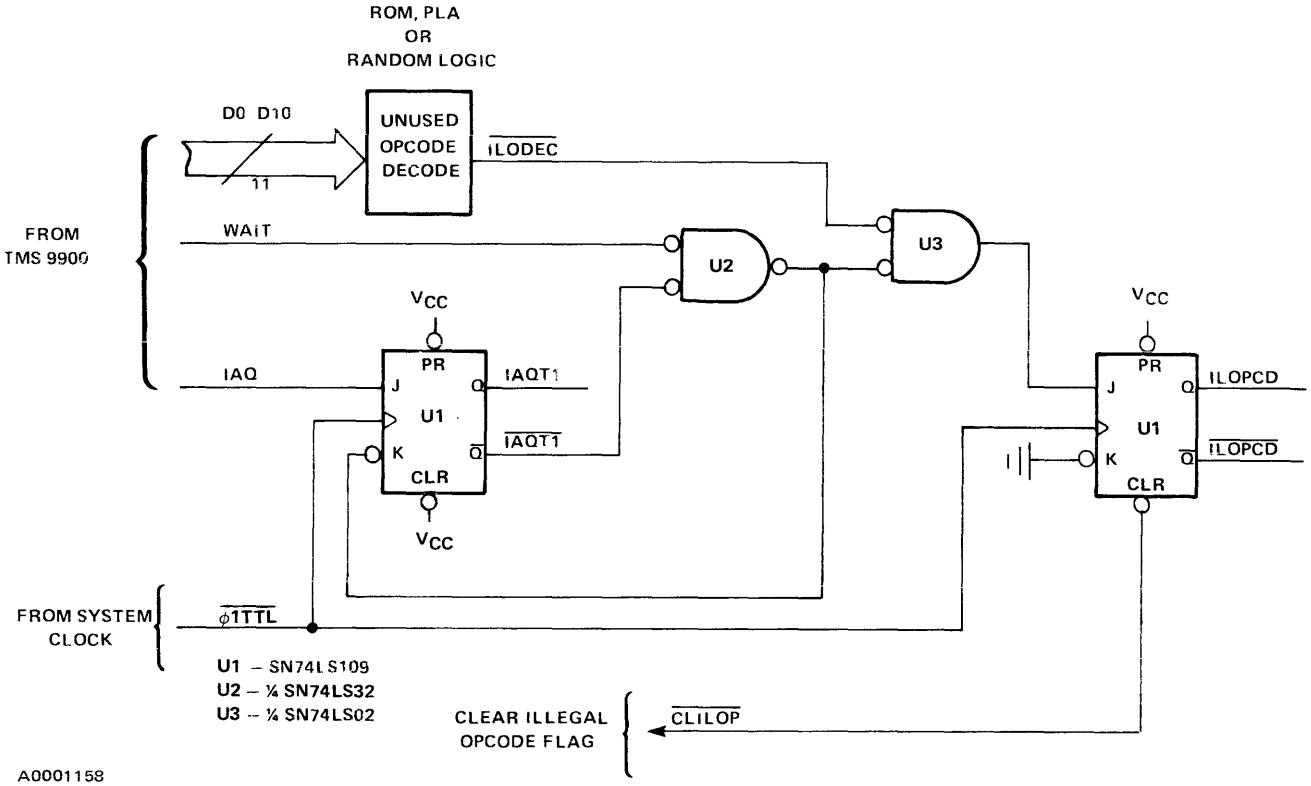


Figure 6-1. Illegal Op Code Detection Circuitry

panel of a conventional minicomputer and are normally implemented in hardware. Similar capabilities may be provided for any TMS 9900 system very economically using a combination of hardware and software.

6.2.1 System Configuration for Software Front Panel

Figure 6-2 illustrates the hardware required to implement a software-controlled front panel for a TMS 9900 system. In this configuration the LOAD function, including the LOAD vector at addresses $FFFC_{16}$-$FFFF_{16}$ is dedicated to the front panel. Some portion of the memory space must be allocated for

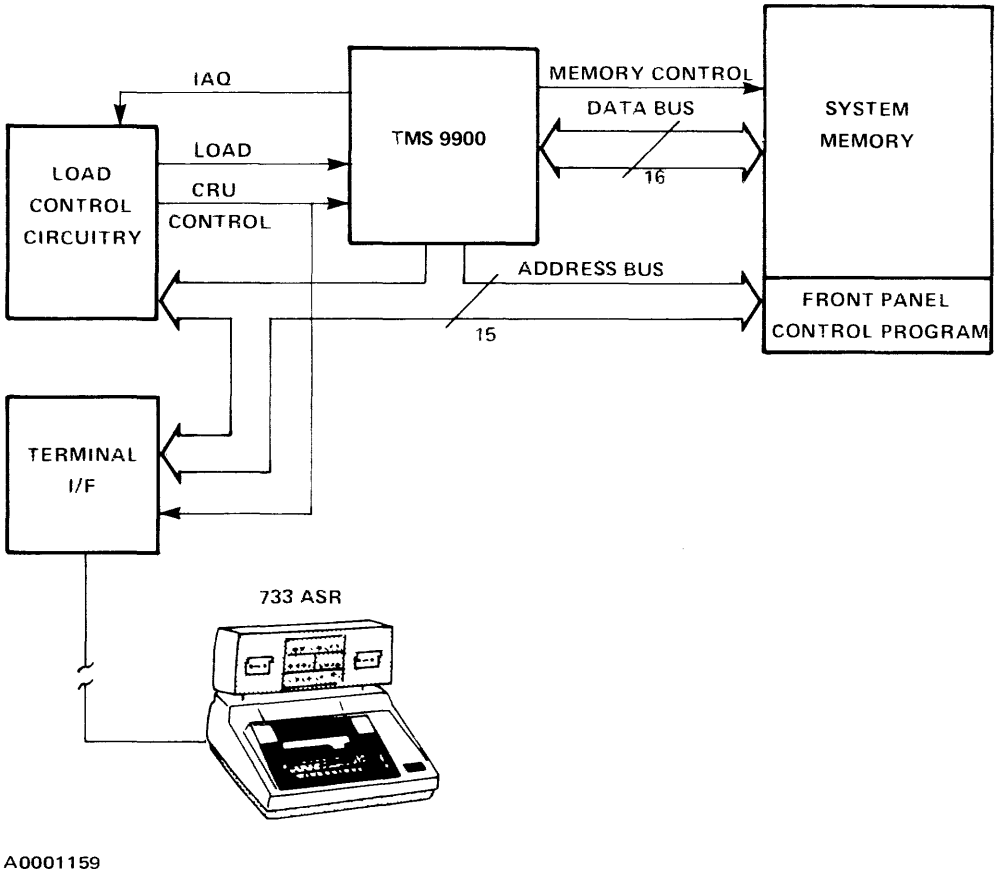


Figure 6-2. System Configuration for Software Front Panel

the control program and temporary storage. Circuitry to control generation of the $\overline{\text{LOAD}}$ input to the processor requires approximately five networks. Finally, some means of communication between the user and the system must be provided. In this example, a teleprinter such as the 733 ASR was selected because of the word degree of flexibility it provides, with keyboard entry, a printer, and dual cassette transport for program loading. Other devices, such as a switch panel with an LED display, would serve the same purpose more economically with a sacrifice in flexibility and ease of use.

6.2.2 Memory Requirements

As shown in Figure 6-3, the $\overline{\text{LOAD}}$ vector at memory addresses $FFFC_{16}$-$FFFF_{16}$ is required. It is normally convenient to use the contiguous memory area for the control program, since both will be contained in ROM or PROM. The size of the control program is a direct function of the number and sophistication of

utilities provided in the control program. Control program RAM requirements are minimal and may be satisfied with system RAM or implemented separately as desired.

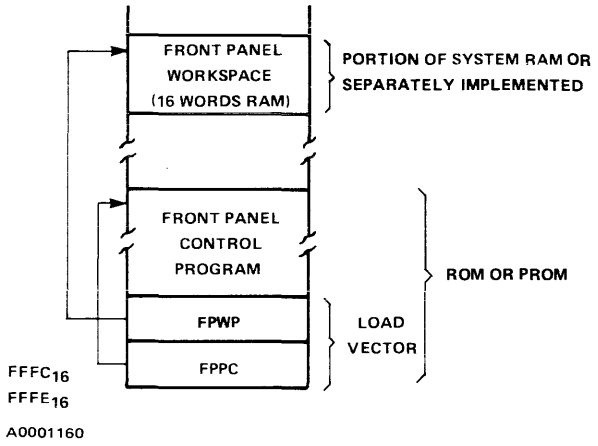


Figure 6-3. Software Front Panel Memory Requirements

6.2.3 Description of Operation

The system is operating in one of two modes whenever power is supplied to the system. In the front-panel mode (FPMODE=1) the processor is executing under the supervision of the front-panel control program. In this mode the user may communicate with the system via the terminal. Commands are entered through the keyboard and transmitted to the system, where the control program interprets the command and performs the desired operation. Commands may cause memory contents to be changed or displayed, particular programs to be executed, or other control operations such as setting up breakpoints or single instruction execution. The front-panel mode is entered through the LOAD function, thus the previous context is saved and may be restored when returning to the run mode (FPMODE=0). In the run mode the processor is executing programs at any location in memory, uncontrolled by the user until the LOAD function again becomes active.

6.2.3.1 Entry Into Front-Panel Mode. Figure 6-4 illustrates the front-panel control circuitry used to generate the $\overline{\text{LOAD}}$ signal to the TMS 9900. Initially, the power-on $\overline{\text{RESET}}$ signal to the system causes FDMODE to be reset. When the user desires to gain control, he actuates a pushbutton, causing $\overline{\text{HALT}}$ to be active. During the next instruction fetch by the processor (AQ=1), FPMODE is set and the system enters the front-panel mode. If the processor is currently executing an IDLE instruction, waiting for an unmasked interrupt to occur, FPMODE is set by the $\overline{\text{IDLE}}$ output of the SN74LS138, which is pulsed each machine cycle. The $\overline{\text{LOAD}}$ signal remains active until the next instruction is fetched. The $\overline{\text{LOAD}}$ signal is therefore active for one instruction. This ensures that the $\overline{\text{LOAD}}$ function is executed, since this input is tested by the TMS 9900 at the end of each instruction cycle. It is necessary that the LOAD signal be active for no more than one instruction cycle; otherwise, the $\overline{\text{LOAD}}$ operation is repeated. Each time the $\overline{\text{LOAD}}$ function is performed, the internal registers of the TMS 9900 are saved in registers WR13–15 of the front-panel workspace. Thus, if the $\overline{\text{LOAD}}$ operation is repeated, the saved internal register contents would be destroyed when the internal registers are again stored. After the $\overline{\text{LOAD}}$ context switch is performed, the processor begins execution of the front-panel control program.

6.2.3.2 Single Instruction Execution. Part of the circuitry in Figure 6-4 allows the front-panel control program to execute single instructions in general memory and then to return to front-panel control. The CKOF instruction is used to initiate this operation, causing the CKOF signal to be pulsed. CKOF is gated with FPMODE and LOAD to ensure that no instructions outside the control program may initiate this operation. The gated CKOF signal sets SIEQ. The timing for the LOAD signal generation is shown in Figure 6-5. The instruction sequence to perform single instruction execution (SIE) is

•
•
•

CKOF SET SIEQ

RTWP RELOAD INTERNAL REGISTERS

•
•
•

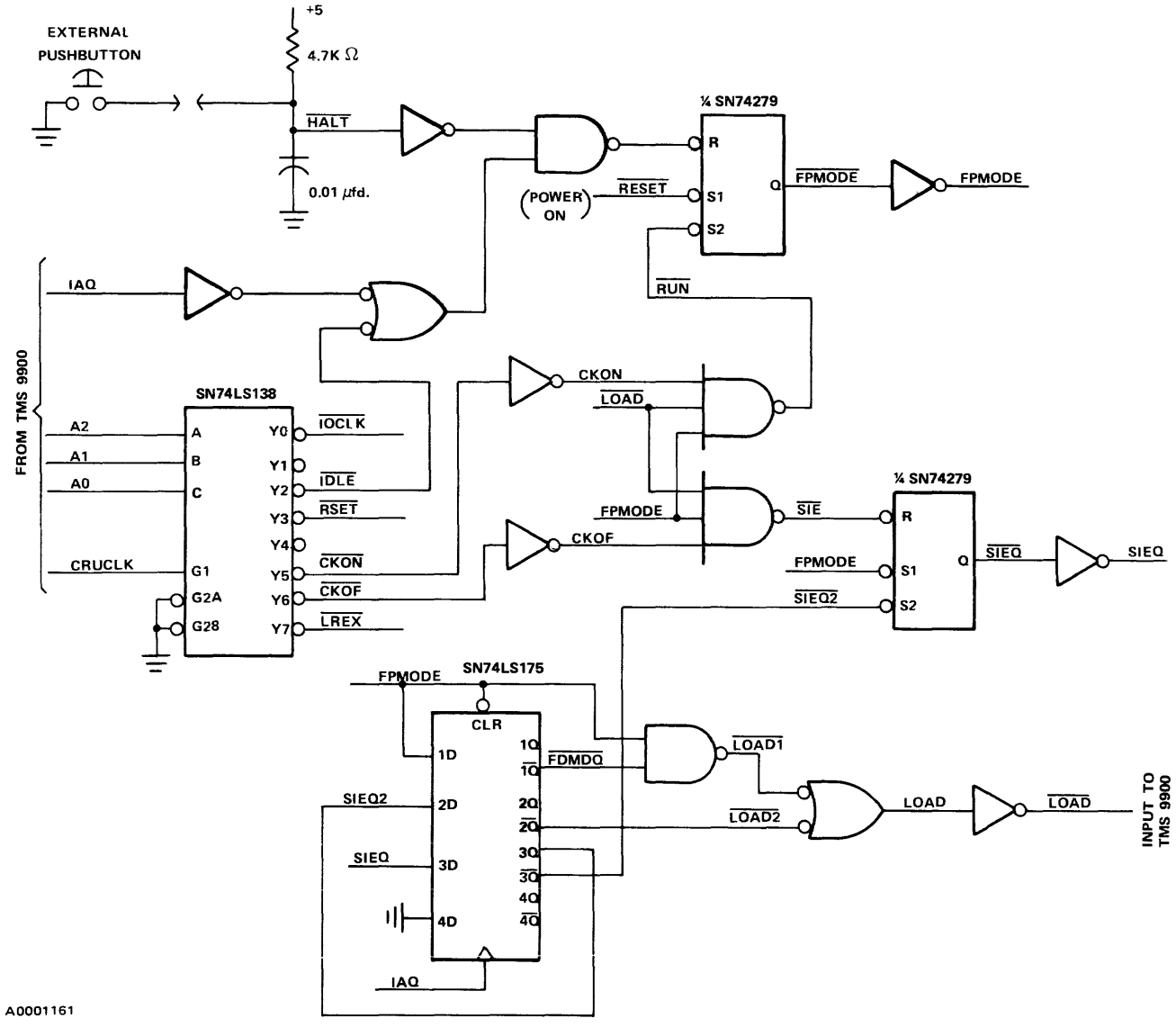


Figure 6-4. Front Panel Control Circuitry

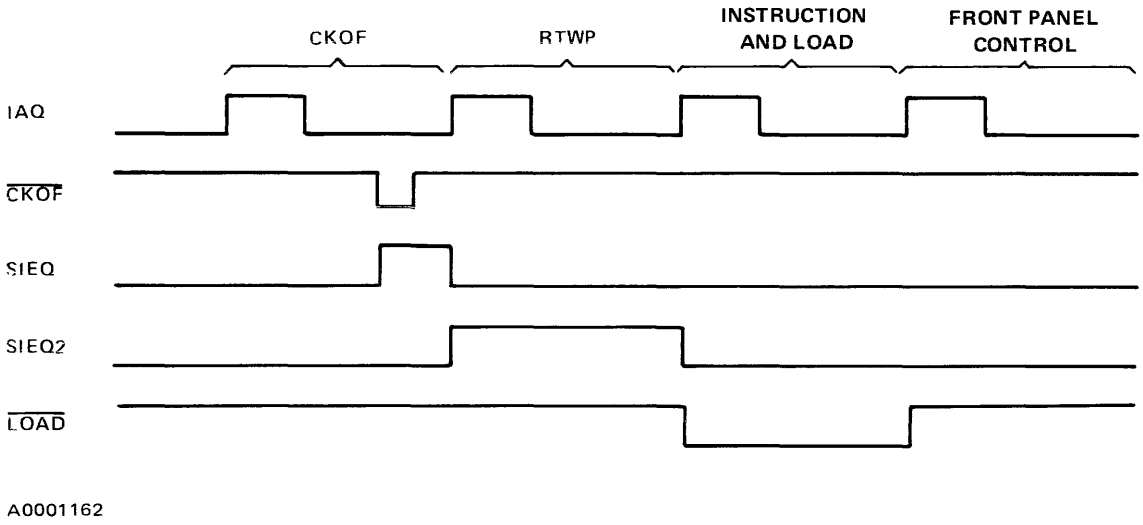


Figure 6-5. Single Instruction Execution Timing

The circuitry in Figure 6-4 causes the $\overline{\text{LOAD}}$ signal to be generated after delaying one instruction; therefore, the $\overline{\text{LOAD}}$ signal is active during the first instruction after RTWP, and control returns to the front-panel control program. This SIE function can also be used to implement breakpoints and constant updates of program operation. For example, a breakpoint may be accomplished by repetitively performing SIE and comparing the saved PC with the desired value.

6.2.3.3 Return to Run Mode. With the circuitry in Figure 6-4, the processor returns to the run mode by executing the following instruction sequence:

```
      .
      .
      .
CKON    RESET FPMODE
RTWP    RELOAD INTERNAL REGISTER
```

The CKON pulse, gated with FPMODE and $\overline{\text{LOAD}}$, resets FPMODE and, after the RTWP instruction is executed, the processor resumes run mode operation and continues until the HALT pushbutton is again pressed.

SECTION VII

ELECTRICAL REQUIREMENTS

This section reviews the TMS 9900 electrical requirements, including the system clock generation and interface signal characteristics. The "TMS 9900 DATA MANUAL" should be used for minimum and maximum values.

7.1 TMS 9900 Clock Generation

The TMS 9900 requires a non-overlapping four-phase clock system with high-level MOS drivers. Additional TTL outputs are typically required for external signal synchronization or for dynamic memory controllers. A single-chip clock driver, the TIM 9904, can be used to produce these clock signals. An alternative clock generator uses standard TTL logic circuits and discrete components.

The TMS 9900 requires four non-overlapping 12 V clocks. The clock frequency can vary from 2 kilohertz to 3 to 250 picofarads. The clock rise and fall times must not exceed 100 nanoseconds and must be 10 to 15 nanoseconds for higher frequencies in order to satisfy clock pulse width requirements. While the clocks must not overlap, the delay time between clocks must not exceed 50 microseconds at lower frequencies. The typical clock timing for 3 MHz is illustrated in Figure 7-1.

7.1.1 TIM 9904 Clock Generator

The TIM 9904 (SN74LS362) is a single-chip clock generator and driver for use with the TMS 9900. The TIM 9904 contains a crystal-controlled oscillator, waveshaping circuitry, a synchronizing flip-flop, and quad MOS/TTL drivers as shown in Figure 7-2.

The clock frequency is selected by either an external crystal or by an external TTL-level oscillator input. Crystal operation requires a 16X input crystal frequency since the TIM 9904 divides the input frequency for waveshaping. For 3-megahertz operation, a 48-megahertz crystal is required. The LC tank inputs permit the use of overtone crystals. The LC network values are determined by the network resonant frequency:

$$f = \frac{1}{2\pi \sqrt{LC}}$$

For less precise frequency control, a capacitor can be used instead of the crystal.

The external-oscillator input can be used instead of the crystal input. The oscillator input frequency is 4X the output frequency. A 12-megahertz input oscillator frequency is required for a 3-megahertz output

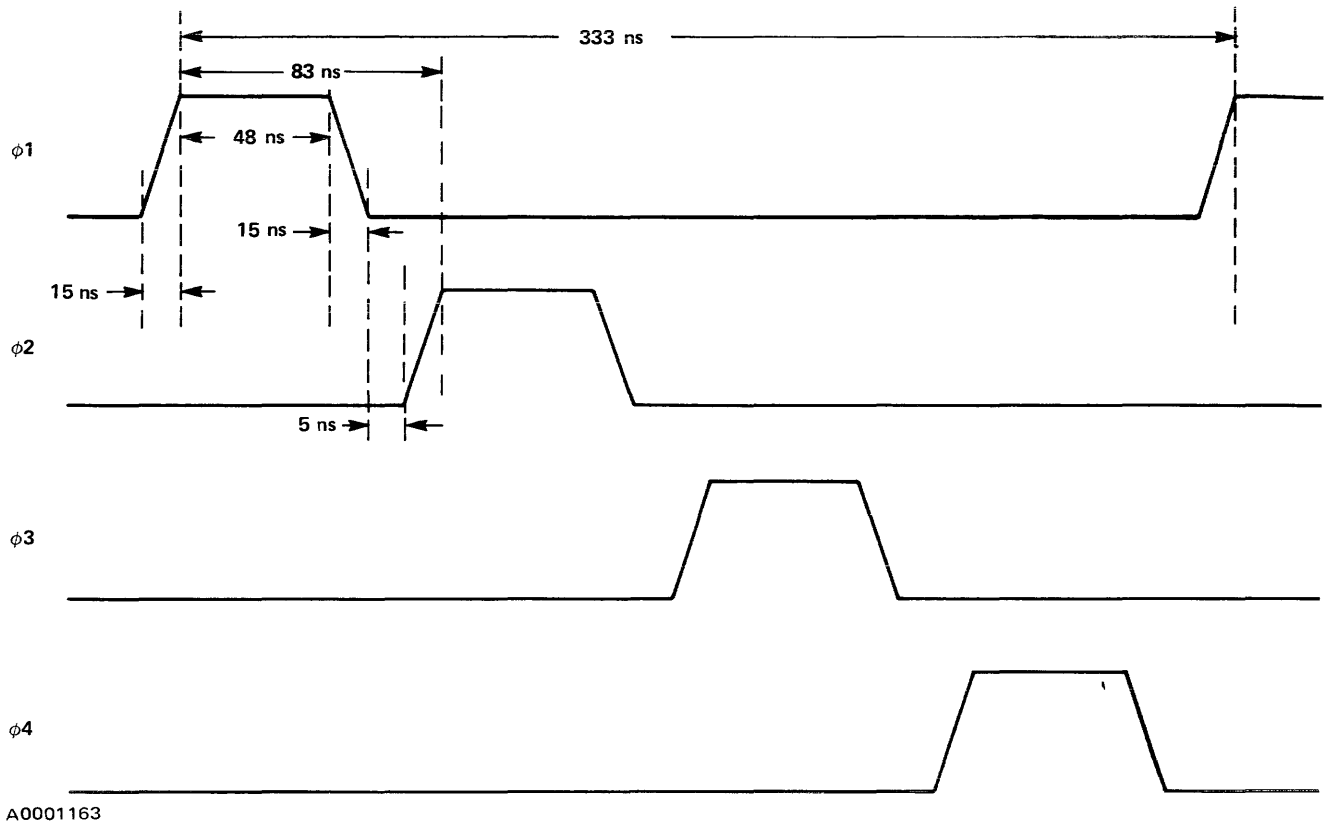


Figure 7-1. TMS 9900 Typical Clock Timing

frequency. A 4X TTL-compatible oscillator output (OSCOUT) is provided in order to permit the derivation of other system timing signals from the crystal or oscillator frequency source.

The oscillator frequency is divided by four to provide the proper frequency for each of the 4-clock phases. A high-level MOS output and an inverted TTL-compatible output is provided by each clock phase. The MOS-level clocks are used for the TMS 9900 CPU while the TTL clocks are used for system timing.

The D-type flip-flop is clocked by $\phi 3$ and can be used to synchronize external signals such as $\overline{\text{RESET}}$. The Schmitt-triggered input permits the use of an external RC network for power-on $\overline{\text{RESET}}$ generation. The RC values are dependent on the power supply rise time and should hold $\overline{\text{RESET}}$ low for at least three clock cycles after the supply voltages reach the minimum voltages.

All TIM 9904 TTL-compatible outputs have standard short circuit protection. The high-level MOS clock outputs, however, do not have short circuit protection.

7.1.2 TTL Clock Generator

Figure 7-3 illustrates an alternate TMS 9900 clock generator circuit. This system is driven by a 36 megahertz clock (minimal duty cycle restriction) which can be derived by any of several popular clock

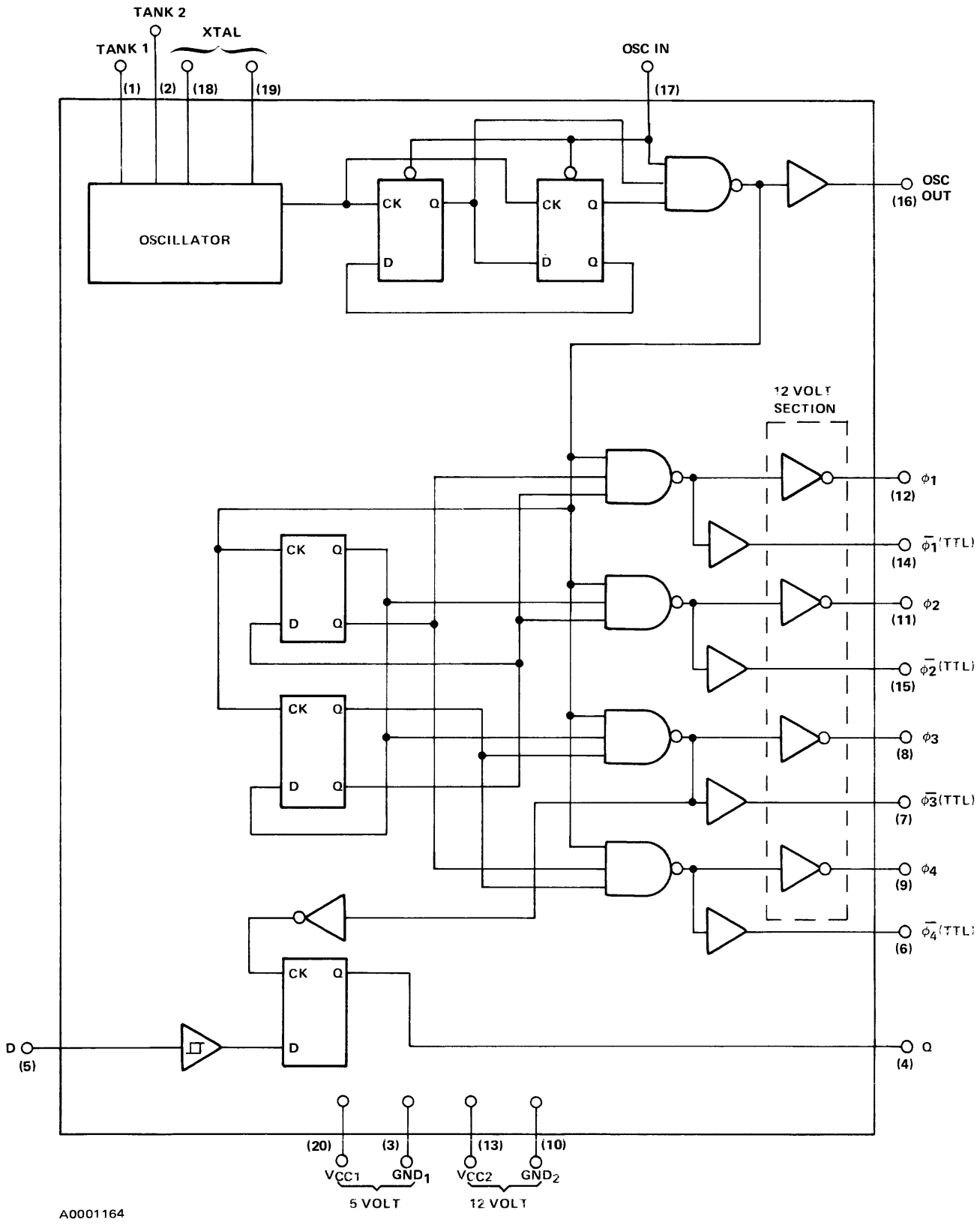


Figure 7-2. TIM 9904 Clock Generator

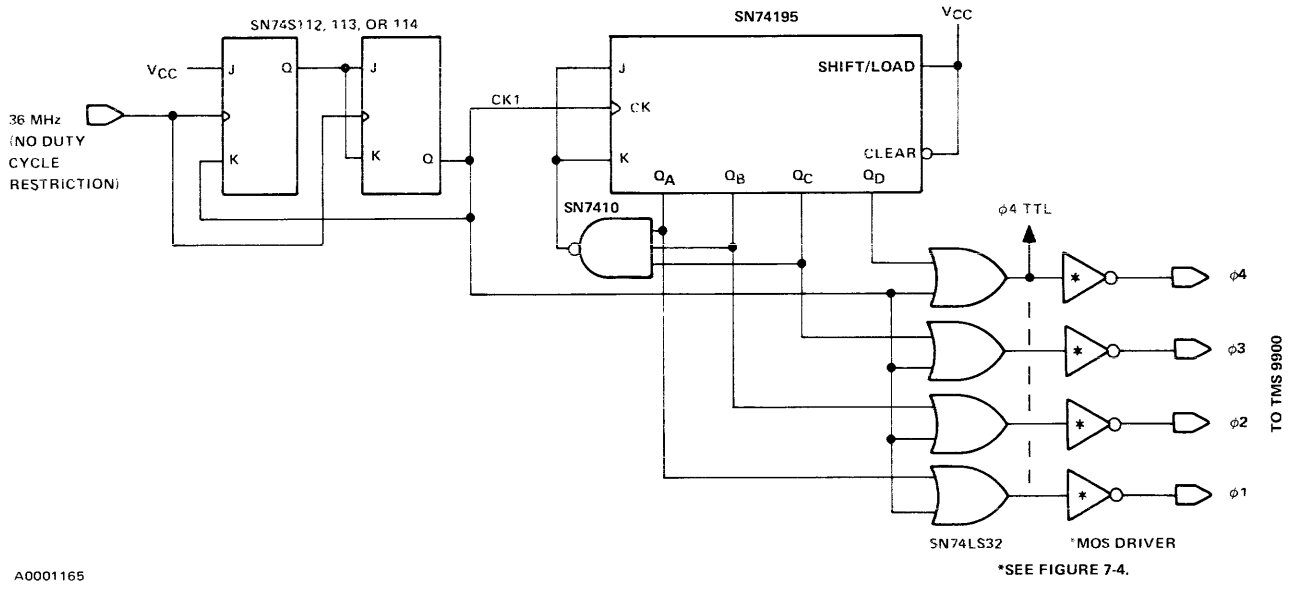


Figure 7-3. TMS 9900 Clock Generator

oscillator designs. The first stage is shown on the timing diagram in Figure 7-4 as CK1 and has a 33% duty cycle. The second stage of the generator is based on the SN74195 shift register. The shift register is connected to generate the consecutive 4-phase clocks shown for one phase as QA in the timing diagram. The shift register outputs are then gated with the input clock to form the non-overlapping TTL clocks (ϕ1TTL).

The non-overlapping TTL clocks can be translated to 12 V MOS levels by several methods. Integrated memory drivers such as the SN75355 or SN75365 can be used but may force operation at reduced frequency due to supply, temperature, or device variation.

The discrete-transistor driver shown in Figure 7-5 has been designed to allow operation at 3 megahertz over standard commercial temperature and voltage ranges.

This driver uses inexpensive 2N3703s and 2N3704s and broad tolerance passive components. Resistor tolerances can be 10% with capacitor variations as much as 20% without affecting its performance noticeably. It shows very little sensitivity to transistor variations and its propagation times are largely unaffected by output capacitive loading. It produces rise times in the 10–12 ns region with fall times from 8–10 ns, driving 200 pF capacitive loads. Propagation times for this driver are such that it produces an output pulse that is wider than its input pulse. This driver can easily be used at 3 megahertz without special selection of components. It does have the advantage of taking nine discrete components per driver, but if assembly costs are prohibitive, these can be reduced by using two Q2T2222 and two Q2T2905 transistor packs. The Q2T2222 is basically four NPN transistors of the 2N2222 type while the Q2T2905 has four PNP, 2N2905 type transistors in single 14-pin dual-in-line packages. Thus, all four drivers can be built using two packages each of these quad packs.

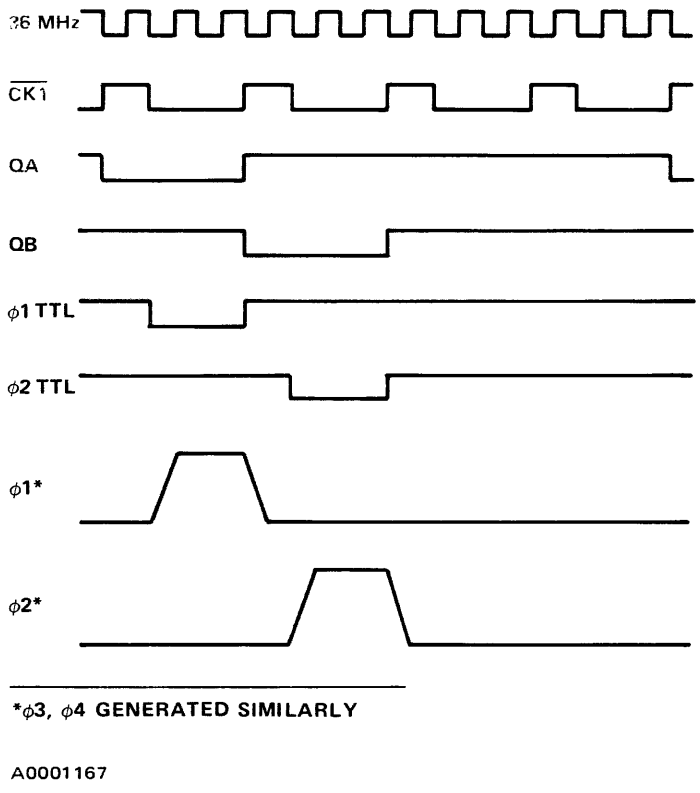


Figure 7-4. Timing Diagram for TMS 9900 Clock Generator

7.2 TMS 9900 Signal Interfacing

The non-clock CPU inputs and outputs are TTL compatible and can be used with bipolar circuits without external pull-up resistors or level shifters. The TMS 9900 inputs are high impedance to minimize loading on peripheral circuits. The TMS 9900 outputs can drive approximately two TTL loads, thus eliminating the need for buffer circuits in many systems.

7.2.1 Switching Levels

The TMS 9900 input switch levels are compatible with most MOS and TTL circuits and do not require pull-up resistors to reach the required high-level input switching voltage. The TMS 9900 output levels can drive most MOS and bipolar inputs. Some typical switching levels are shown in Table 7-1.

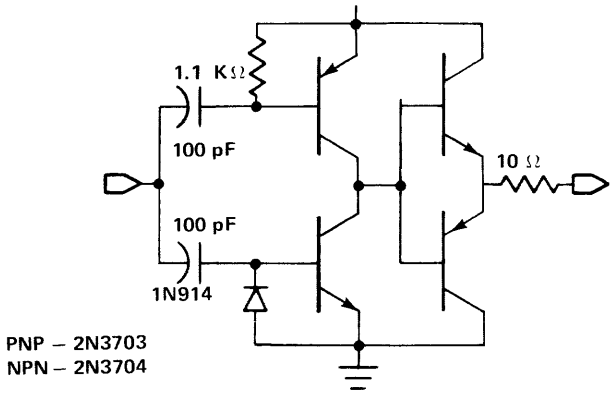


Figure 7-5. MOS Level Clock Drivers

It should be noted that some MOS circuits such as the TMS 4700 ROM and the TMS 2708 EPROM have a minimum high-level input voltage of 3 V to 3.3 V, which exceeds the TMS 9900 minimum high-level output voltage of 2.4 V. The TMS 9900 high-level output voltage exceeds 3.3 V; however, longer transition times as shown in Figure 7-6 are required.

TABLE 7-1. SWITCH LEVELS

SWITCHING LEVEL (V)	TMS 9900	TMS 4908	TMS 4042-2	SN 74XX	SN 74LSXX
V_{IH} min	2.2	3.0	2.2	2.0	2.0
V_{IL} max	0.6	0.65	0.65	0.8	0.7
V_{OH}* min	2.4	3.7	2.2	2.4	2.7
V_{OL} max	0.5	0.45	0.45	0.5	0.5

*V_{OH} exceeds 2.4 V as shown in Figure 7-6.

7.2.2 Loading

The TMS 9900 has high-impedance inputs to minimize loading on the system buses. The CPU data bus presents a maximum current load of ±75 μA when DBIN is high. $\overline{WE}$, $\overline{MEMEN}$, and DBIN cause a maximum current load of ±75 μA during HOLDA. Otherwise, the TMS 9900 inputs present a current load of only ±10 μA. The data bus inputs have a 25-picofarad input capacitance, and all other non-clock inputs have a 15-picofarad input capacitance.

The TMS 9900 outputs can drive approximately two standard TTL loads. Since most memory devices have high-impedance inputs, the CPU can drive small memory systems without address or data buffers. If the bus load exceeds the equivalent of two TTL unit loads, external buffers are required.

The TMS 9900 output switching characteristics are determined for approximately 200 picofarads. Higher capacitive loads can be driven with degraded switching characteristics as shown in Figure 7-7.

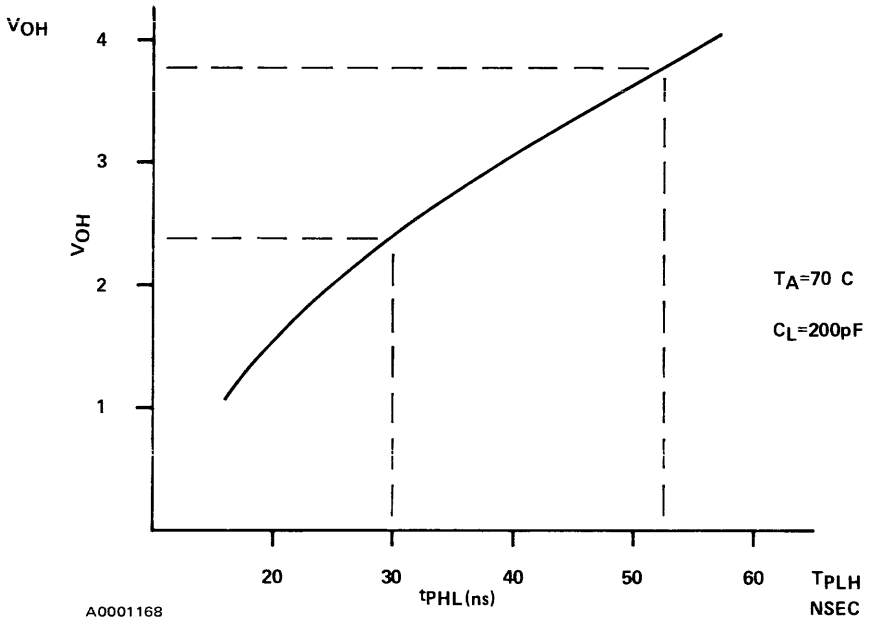


Figure 7-6. t_{PLH} vs V_{OH} Typical Output Levels

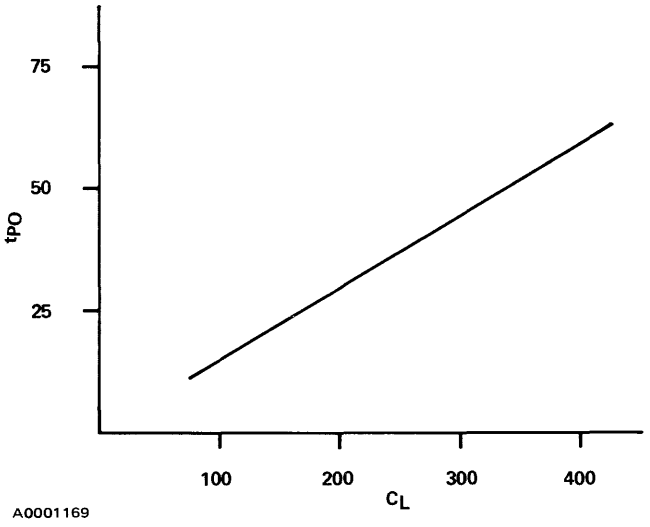


Figure 7-7. t_{PO} vs Load Capacitance (Typical)

7.2.3 Recommended Interface Logic

The TMS 9900 is compatible with the logic from any of the common TTL logic families. The Texas Instruments low-power Schottky logic circuits are, however, recommended for use in microprocessor systems. The SN74LSXX circuits have higher impedance inputs than standard TTL, allowing more circuits to be used without buffering. The SN74LSXX gates also consume less power at similar switching speeds. Texas Instruments has a wide assortment of Bipolar support circuits which can be used with the TMS 9900, as shown in Table 7-2. Note that three circuits which are particularly useful in many applications have been dual symbolized with TIM 99XX numbers for easy reference.

There are a number of buffer circuits available for use in TMS 9900 systems. The SN75S241 and SN74LS241 non-inverting octal buffers with three-state outputs can be used as memory address drivers or as bidirectional data transceivers. The SN74S240 and SN74LS240 are similar, but with inverted outputs. The SN74LS241 can be used as either a memory-address buffer or as a transceiver for bidirectional data transfers. The use of a single circuit type for both functions can result in a lower inventory and parts cost. The buffer switching times can be derated for higher capacitive loading as required.

7.2.4 System Layout

The pin assignments of the TMS 9900 are such that sets of signals (data bus, address bus, interrupt port, etc.) are grouped together. The layout of a printed circuit board can be simplified by taking advantage of these groups by locating associated circuitry (address buffers, interrupt processing hardware, etc.) as close as possible to the TMS 9900 interface. Shortened conductor runs result in minimal noise and compact and efficient utilization of printed circuit board area.

It is particularly important that the drivers for $\phi 1 - \phi 4$ be located as close as possible to the inputs of the TMS 9900, since these signals have fast rise and fall times while driving fairly high capacitance over a wide

voltage range. The 12 volt supply to the clock drivers should be decoupled with both high (15 μF) and low (0.05 μF) value capacitors in order to filter out high and lower frequency variations in supply voltage.

All voltage inputs to the TMS 9900 should be decoupled at the device. Particular attention should be paid to the +5 volt supply. All data and address lines are switched simultaneously. The worst-case condition occurs when all data and address signals switch to a low level simultaneously and they are each sinking 3.2 mA. It is thus possible for the supply current to vary nearly 100 mA over a 20 ns interval. Careful attention must be paid by the designer to avoid supply voltage spiking. The exact values for capacitors should be determined empirically, based on actual system layout and drive requirements.

TABLE 7-2. TMS 9900 BIPOLAR SUPPORT CIRCUITS

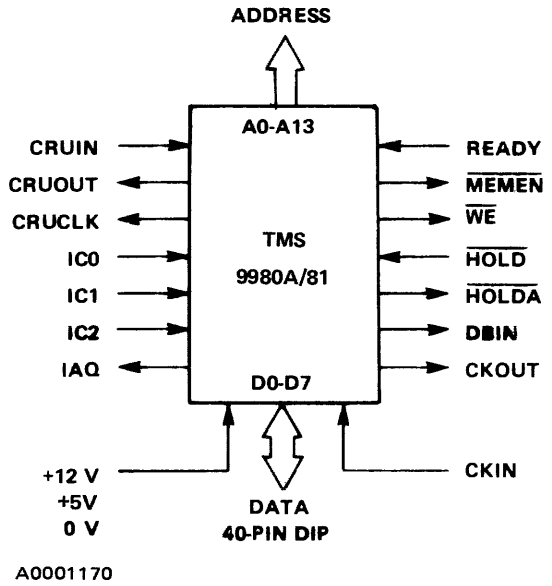
DEVICE	FUNCTION	PACKAGE
	BUFFERS (3-STATE)	
SN74125	QUAD Inverting Buffer	14
SN74126	QUAD Inverting Buffer	14
SN74LS240	OCTAL Inverting Buffer/Transceiver	20
SN74LS241	OCTAL Noninverting Buffer/Transceiver	20
SN74LS242	OCTAL Inverting Transceiver	14
SN74LS243	OCTAL Noninverting Transceiver	14
SN74S240	OCTAL Inverting Buffer/Transceiver	20
SN74S241	OCTAL Noninverting Buffer/Transceiver	20
SN74365	Hex Noninverting Buffer	16
SN74366	Hex Inverting Buffer	16
SN74367	Hex Noninverting Buffer	16
SN74368	Hex Inverting Buffer	16
	LATCHES	
SN74LS259 (TIM9906)	OCTAL Addressable Latch	16
SN74LS373	OCTAL Transparent Latch (3-state)	20
SN74LS412	OCTAL I/O Port (3-state)	24
	DATA MULTIPLEXERS	
SN74LS151	OCTAL Multiplexer	16
SN74LS251 (TIM9905)	OCTAL Multiplexer (3-state)	16
	OTHER SUPPORT CIRCUITS	
SN74148 (TIM 9907)	Priority Encoder	16
SN74LS74	Dual D-type flip-flop	14
SN74LS174	Hex D-type flip-flop	16
SN74LS175	Qual D-type flip-flop	16
SN74LS37	QUAD 2-Input nand Buffers	14
SN74LS362	Clock Generator	20

SECTION VIII

TMS 9980A/81

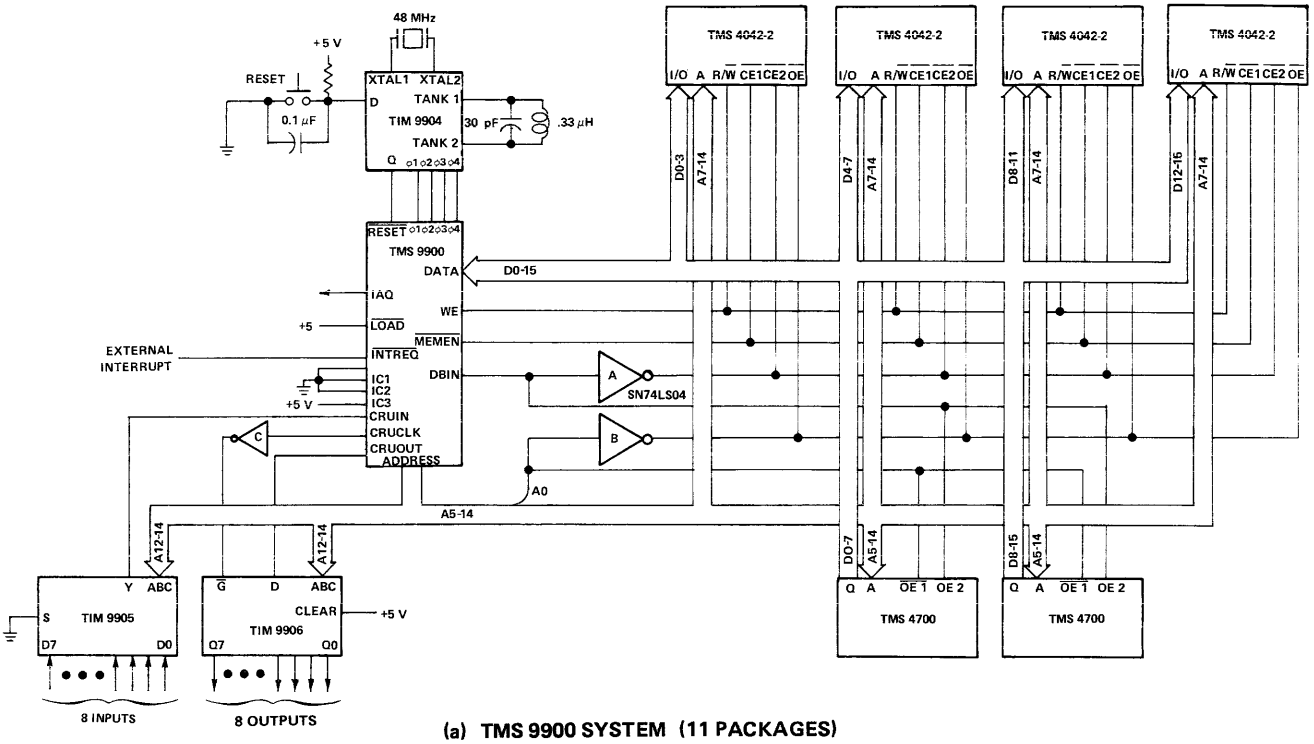
The TMS 9980A/81 microprocessor is an eight-bit central-processing unit produced using N-channel silicon-gate MOS technology (Figure 8-1). The TMS 9980A/81 extends the power and versatility of the TMS 9900 16-bit machine to those applications requiring an 8-bit data bus, providing the instruction set and addressing capabilities of a full minicomputer to such applications as intelligent or stand-alone terminals, communications controllers and other byte-oriented applications.

The TMS 9980A/81 is fully software compatible with the TMS 9900, yet provides opportunities for lower package count and simplified design effort for smaller systems. For example, the TMS 9900 configuration shown in Figure 8-2(a) requires 11 packages versus seven packages for the TMS 9980A/81 system shown in Figure 8-2(b). The 40-pin package of the TMS 9980A/81 also requires less board space, providing a further reduction in space requirements.

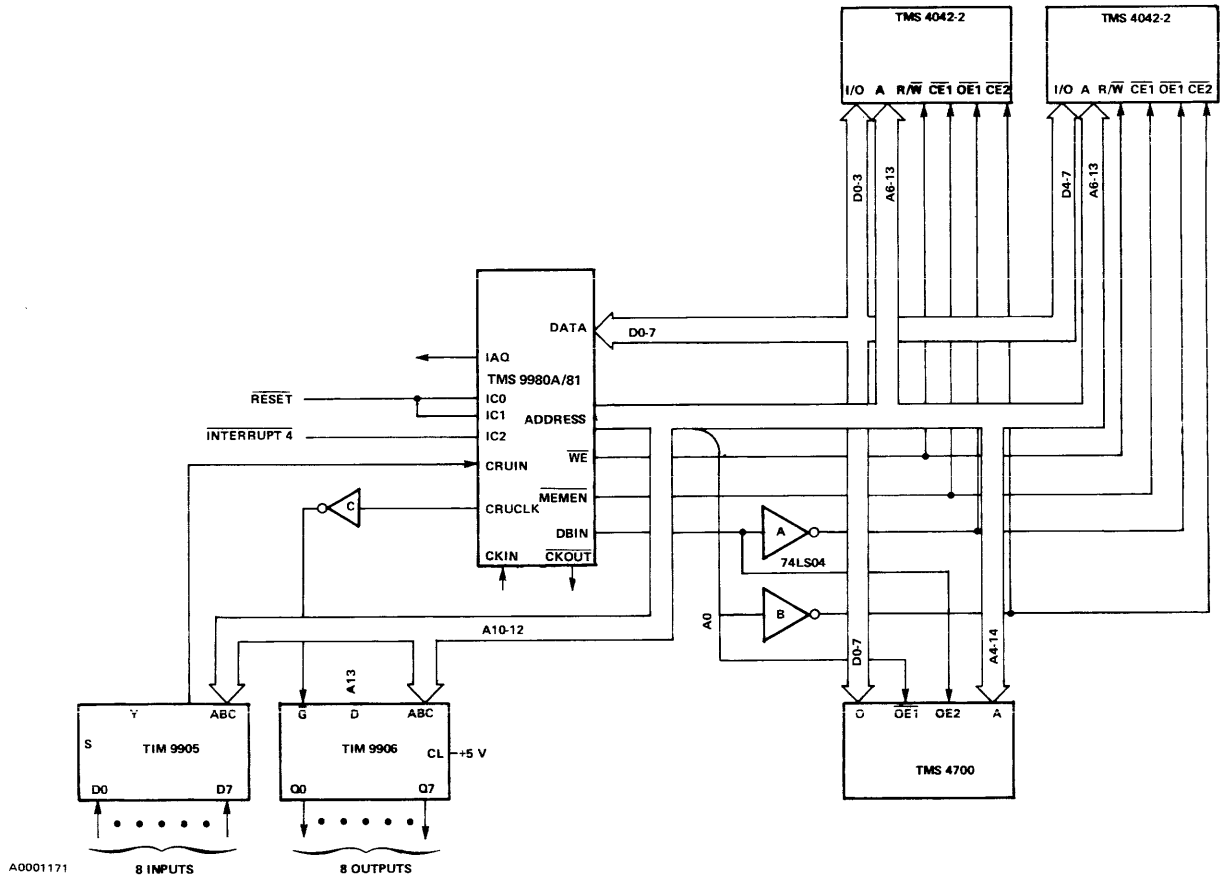


- 16 BIT CPU
- FULL 9900 INSTRUCTION SET
- ON-CHIP CLOCK GENERATOR
- 8-BIT DATA BUS
- 16K BYTE MEMORY ADDRESS
- 2K CRU BIT ADDRESS
- 6 INTERRUPTS
- DMA
- N-CHANNEL SILICON-GATE PROCESS

Figure 8-1. TMS 9980A/81 Microprocessor



(a) TMS 9900 SYSTEM (11 PACKAGES)



(b) TMS 9980A/81 SYSTEM (7 PACKAGES)

Figure 8-2. TMS 9980A/81 and TMS 9900 Configurations

8.1 Architecture

The TMS 9980A/81 employs the same advanced memory-to-memory architecture as the TMS 9900. The memory is addressable in up to 16,384 8-bit bytes with its memory word defined as 16 bits or two consecutive bytes.

Memory words are restricted to be on even address boundaries; i.e., the most-significant half (8 bits) resides at the even address and the least-significant half resides at the subsequent odd address. A byte can reside at either an even or an odd address. The formats are shown in Figure 8-3.

All memory accesses by the TMS 9980A/81 CPU result in a 16-bit transfer. Therefore, a byte instruction referencing an odd byte address will result in a read or a write of the complete word located at the even word address.

8.2 Memory

The TMS 9980A/81 instructions build a 14-bit address word which describes a 16K X 8 bit address space. All address bits are brought out to define completely the 16K X 8 bit address space, as opposed to the TMS 9900 which manages the least-significant address bit internally for byte addressing.

The advanced architecture of the TMS 9980A/81 uses part of the external memory space for workspace register files and for transfer vector storage. For maximum design flexibility only the locations of the transfer vectors are restricted. A memory map of the TMS 9980A/81 is shown in Figure 8-4. Note that since the TMS 9980A/81 has only four maskable external interrupts, only addresses 0004_{16} through 0013_{16} are used as interrupt vectors, as opposed to the TMS 9900 which has 15 external interrupts.

The TMS 9980A/81 utilizes the same three signals to control the use of the data bus and address bus during memory read and write cycles as the TMS 9900, described in Section 3.2. Memory read and write cycles are

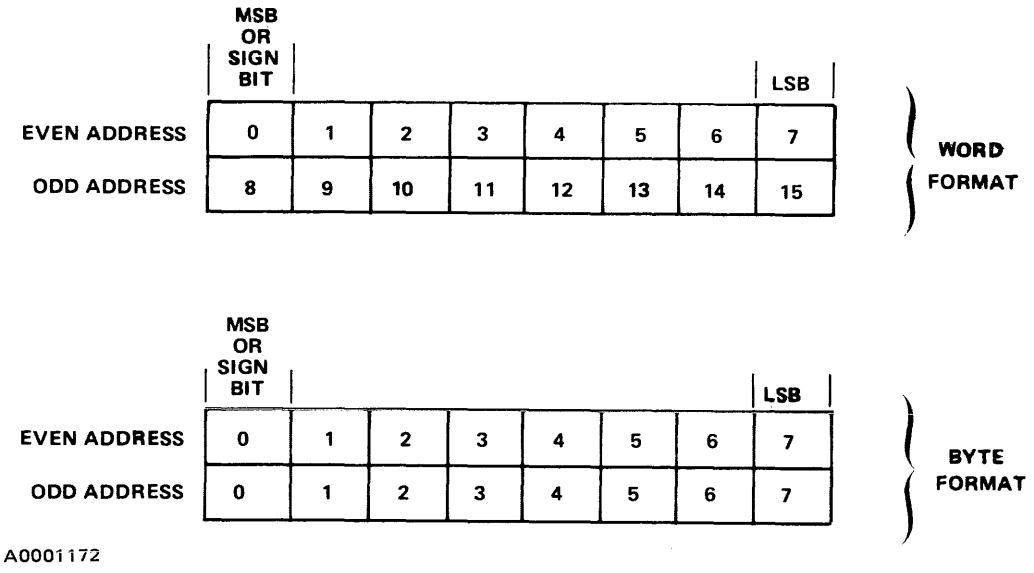


Figure 8-3. TMS 9980A/81 Memory Data Formats

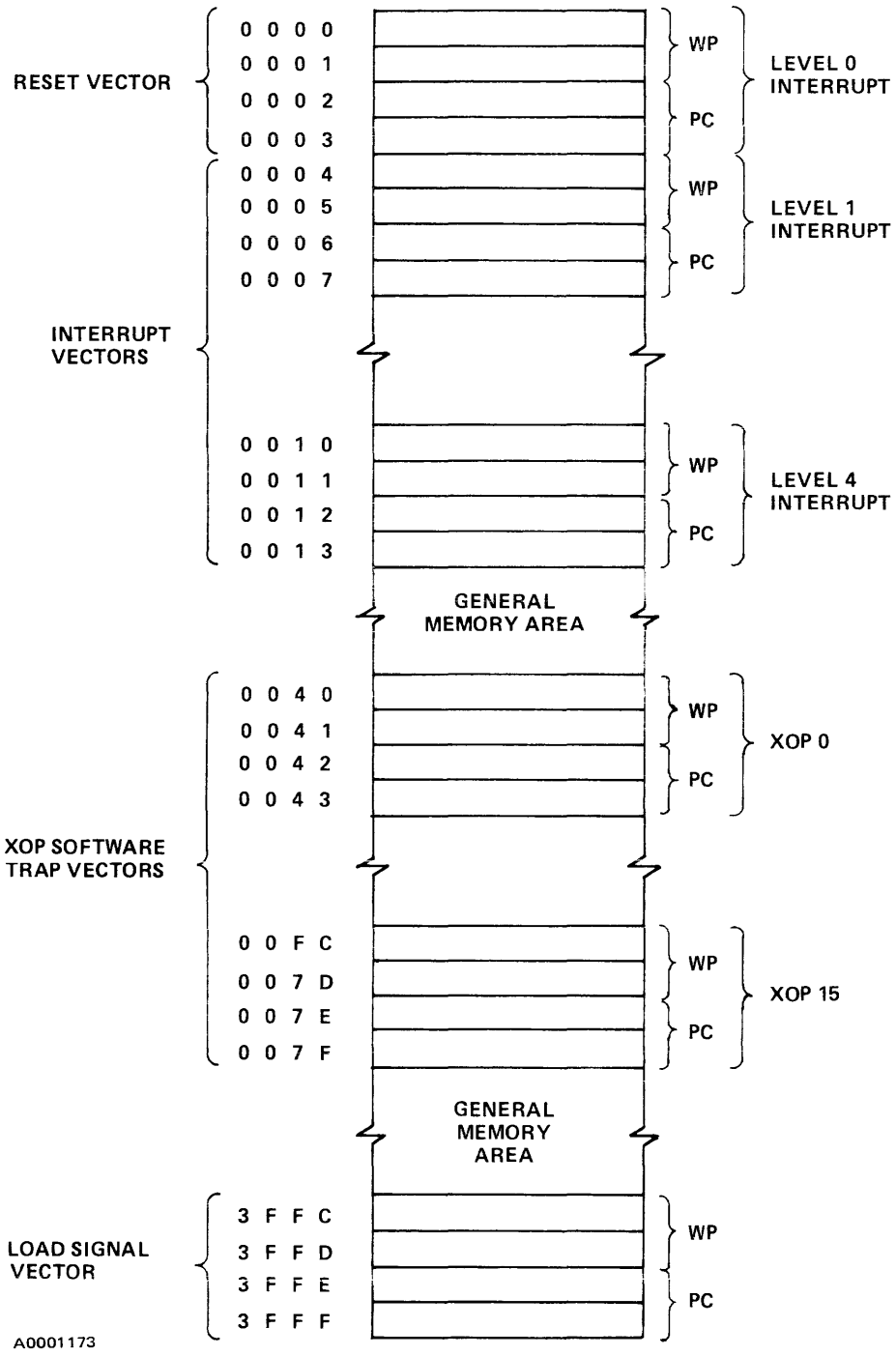


Figure 8-4. TMS 9980A/81 Memory Map

depicted in Figure 8-5(a) for bus timing with no wait states and in Figure 8-5(b) for memory bus timing with one wait state. Note, however, that the TMS 9980A/81 does not provide the WAIT control signal, as in the TMS 9900, when the CPU has entered the WAIT state. Therefore, READY must be generated externally even when only a single WAIT state is desired. This can easily be accomplished, however, by a circuit similar to that shown in Figure 8-6 to produce one wait state, or that shown in Figure 8-7 to generate two wait states. The address-decode circuitry generates an active-high signal SLOMEM whenever the slow memory is addressed. If memory addresses 8000_{16}–$FFFF_{16}$ select slow memory example, SLOMEM = A_0. READY then provides the proper memory timing.

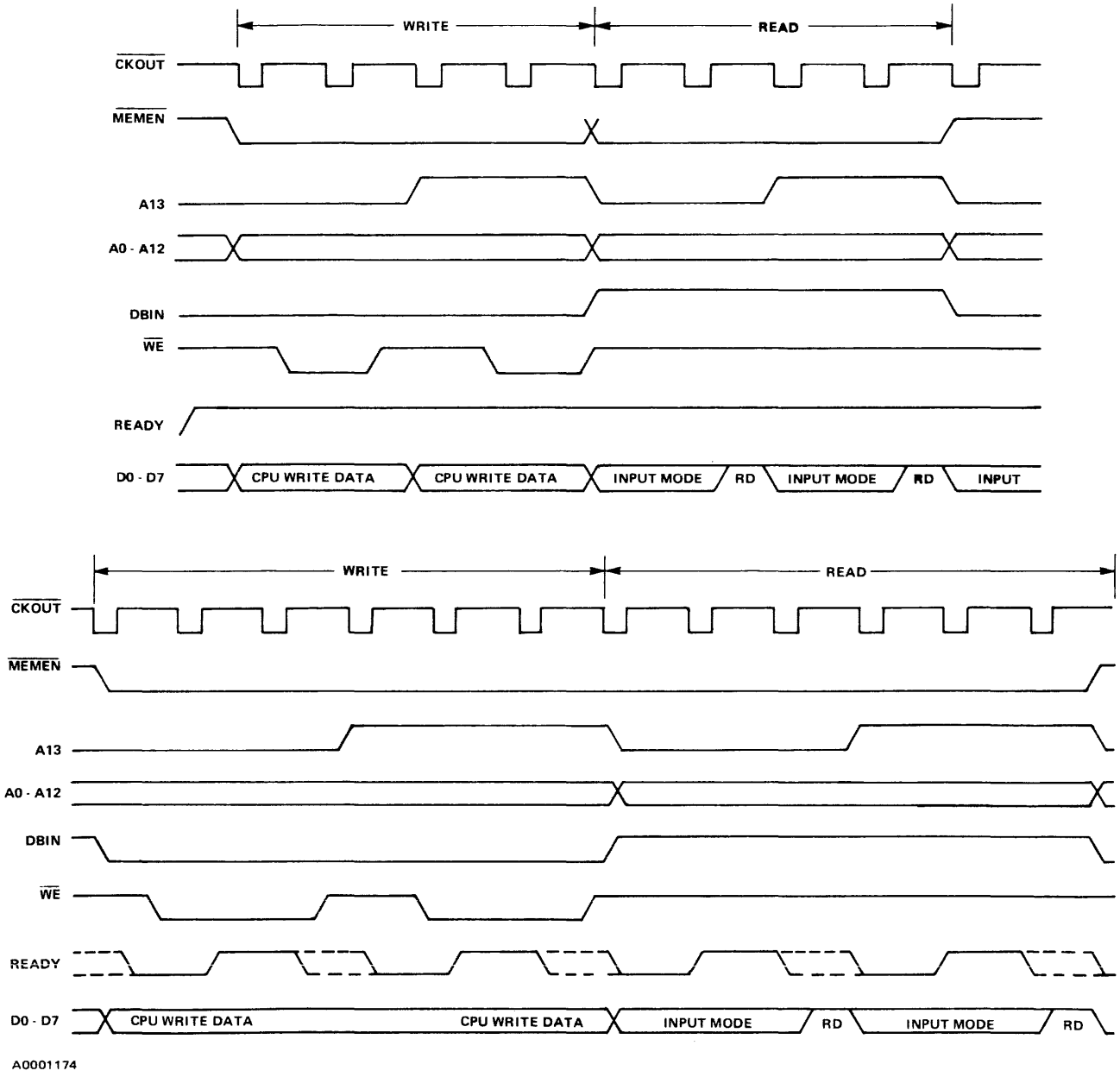


Figure 8-5. TMS 9980A/81 Memory Bus Timing

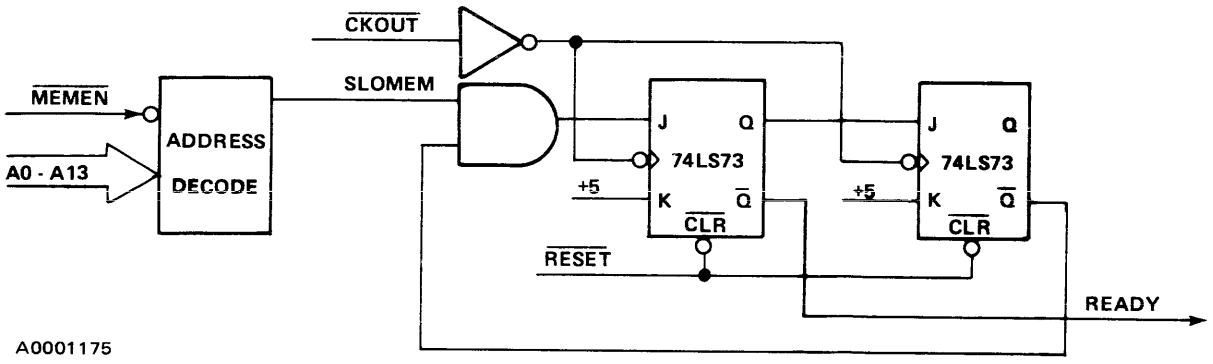


Figure 8-6. Single Wait States For Slow Memory

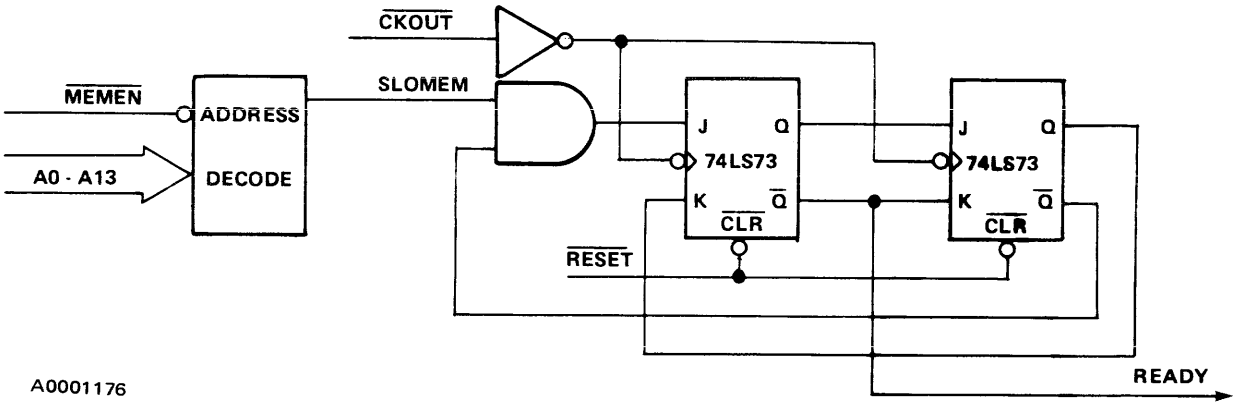


Figure 8-7. Double Wait States For Slow Memory

The TMS 9980A/81 uses direct-memory access (DMA) to permit high volume data transfer between an external I/O device and memory without direct CPU intervention. $\overline{\text{HOLD}}$ and HOLDA timing are shown in Figure 8-8. The maximum latency time between a hold request and a hold acknowledge by the TMS 9980A/81 is three clock cycles plus six memory cycles. Thus, the worst case response will be 7.5 microseconds for a system operating at 2 MHz.

8.3 Interrupts

The TMS 9980A/81 provides four maskable interrupt levels in addition to the $\overline{\text{RESET}}$ and $\overline{\text{LOAD}}$ functions. As in the TMS 9900 the CPU has a priority ranking system to resolve conflicts between

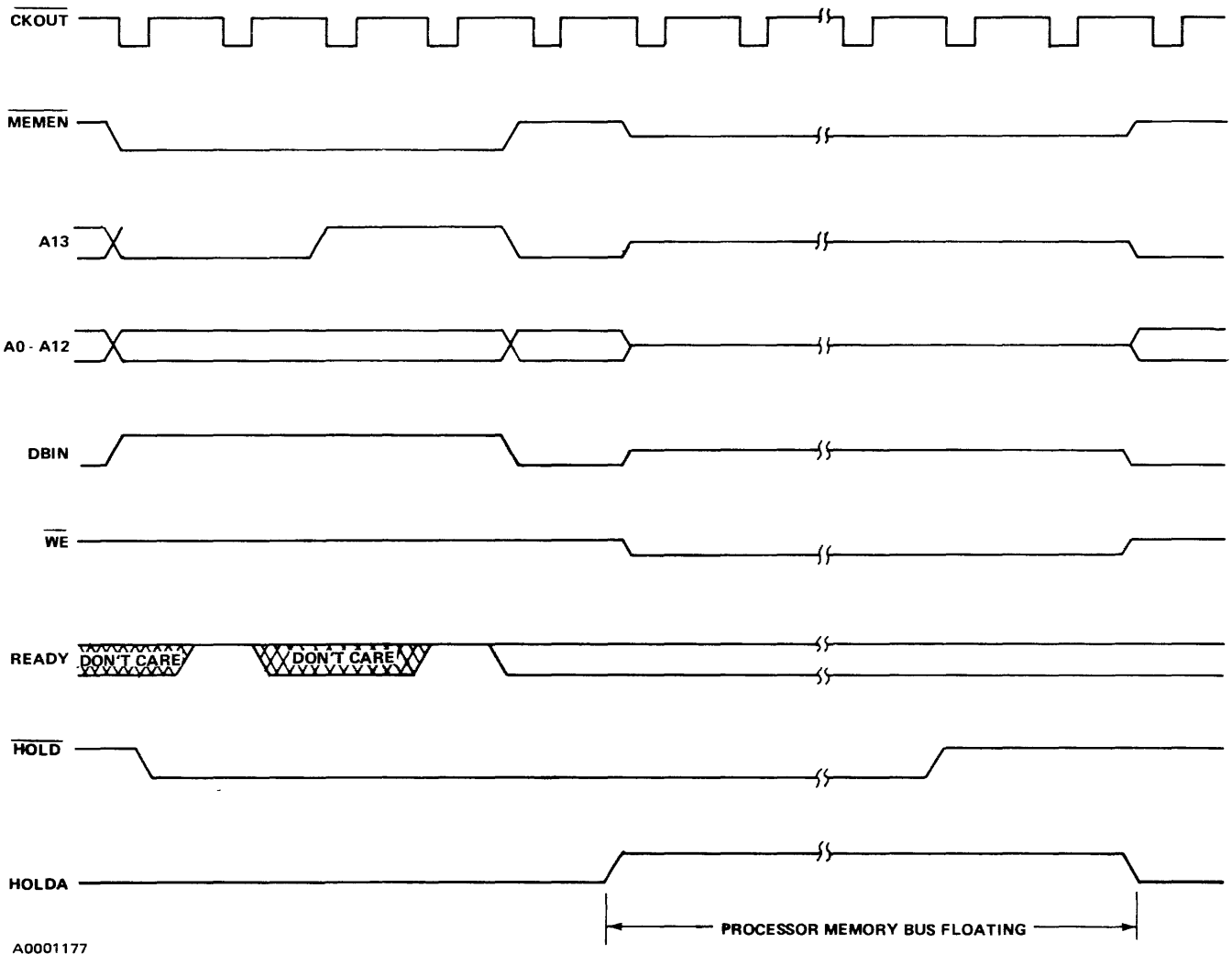


Figure 8-8. TMS 9980A/81 HOLD Timing

simultaneous interrupts and a level mask to disable lower priority interrupts. Once an interrupt is recognized, the CPU performs as a vectored context switch to the interrupt service routine. The $\overline{\text{RESET}}$ and $\overline{\text{LOAD}}$ signals are also encoded on the interrupt-code input lines as opposed to the $\overline{\text{RESET}}$ and $\overline{\text{LOAD}}$ inputs on the TMS 9900. Figure 8-9 illustrates three of the possible interrupt configurations of the TMS 9900. The interrupt vector locations, device assignments, enabling mask values, and the interrupt codes are shown in Table 8-1. Figure 8-9(a) depicts the simplest configuration in which the reset and only one external interrupt are implemented. Figure 8-9(b) also has only one external interrupt, but implements the load signal as well as the reset signal. Figure 8-9(c) uses the SN74LS148 priority encoder to implement and prioritize all four external interrupt signals as well as the reset and load signals. The constraints described in Section 4.3.3 also apply to the TMS 9980A/81 with respect to masking interrupts; that is, an external interrupt mask should be altered only when the interrupt mask is at a level such that the interrupt will not be processed.

8.4 Input/Output

The TMS 9980A/81 uses the same three I/O modes as the TMS 9900: direct memory access (DMA), memory-mapped, and communications register unit (CRU). This multi-mode capability enables the designer to simply and easily optimize the I/O system to the application.

The CRU is a versatile command-driven I/O interface which provides up to 2048 directly addressable input or output bits. Both input and output bits can be addressed either individually or in fields from 1 to 16 bits. The TMS 9980A/81 employs CRUIN, CRUCLK, and A13 (for CRUOUT) and 11 bits (A2–A12) of the address bus to interface with the CRU system. Processor instructions can set, reset, or test any bit in the CRU array or move them between memory and the CRU. Figure 8-10 illustrates the development of a CRU single-bit address. Figure 8-11 depicts an 8-bit I/O interface utilizing the TMS 9901 programmable system interface, and Figure 8-12 depicts a 16-bit I/O interface implemented in TTL using the TIM 9905 (SN74LS251) and TIM 9906 (SN74LS259).

8.5 External Instructions

The TMS 9980A/81 has five external instructions that allow user-defined external functions to be initiated under program control. These instructions are CKON, CKOF, RSET, IDLE, and LREX. These mnemonics, except for IDLE, relate to functions implemented in the 990 minicomputer and do not restrict use of the instructions to initiate various user-defined functions. IDLE also causes the TMS 9980A/81 to enter the idle state and remain until an interrupt, RESET, or LOAD occurs. When any of these five instructions are executed by the TMS 9980A/81, a unique 3-bit code appears on A0, A1, and A13 along with a CRUCLK pulse. When the TMS 9980A/81 is in an idle state, the 3-bit code and CRUCLK pulses occur repeatedly until the idle state is terminated. The codes are shown in Table 8-2, and a circuit is shown in Figure 8-13 which can be used to decode these external instructions.

8.6 TMS 9980A/81 System Clock

The TMS 9980A/81 differs from the TMS 9900 in that it has an internal four-phase clock generator. This internal system clock generator is operated by providing a TTL level periodic wave of frequency equal to or less than 10 MHz to the CKIN terminal using a circuit similar to that shown in Figure 8-14. This external clock signal is divided by four to provide the necessary 2.5 MHz or less system clock frequency. System synchronization is maintained with the $\overline{\text{CKOUT}}$ signal, $\overline{\phi 3}$.

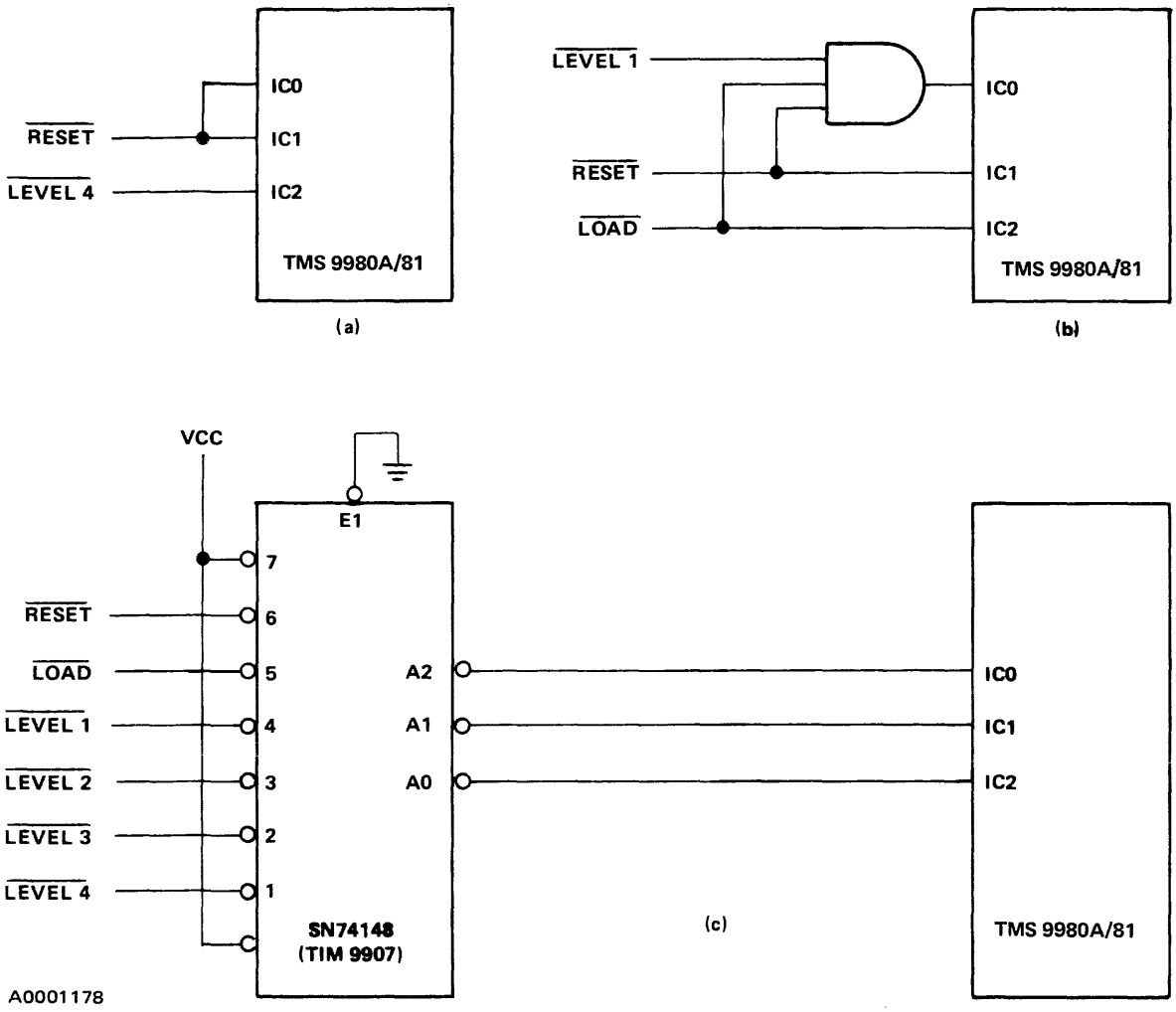


Figure 8-9. TMS 9980A/81 Interrupt Interfaces

TABLE 8-1. INTERRUPT LEVEL DATA

INTERRUPT CODE (IC0–IC2)	FUNCTION	VECTOR LOCATION (MEMORY ADDRESS IN HEX)	DEVICE ASSIGNMENT	INTERRUPT MASK VALUES TO ENABLE (ST12 THROUGH ST15)
1 1 0	Level 4	0 0 1 0	External Device	4 Through F
1 0 1	Level 3	0 0 0 C	External Device	3 Through F
1 0 0	Level 2	0 0 0 8	External Device	2 Through F
0 1 1	Level 1	0 0 0 4	External Device	1 Through F
0 0 1	Reset	0 0 0 0	Reset Stimulus	Don't Care
0 1 0	Load	3 F F C	Load Stimulus	Don't Care
0 0 0	Reset	0 0 0 0	Reset Stimulus	Don't Care
1 1 1	No-Op	-----	-----	-----

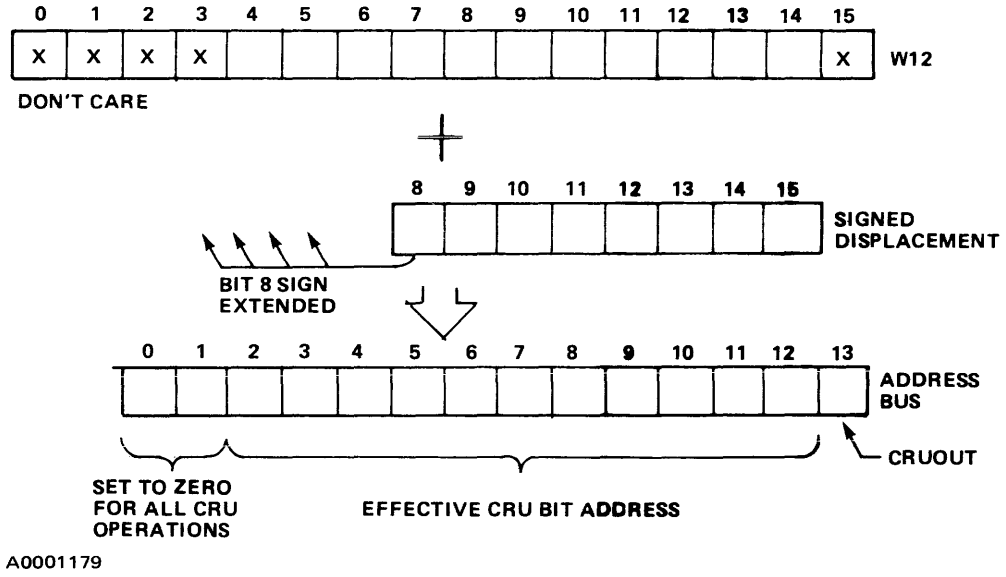


Figure 8-10. TMS 9980A/81 Single-Bit CRU Address Development

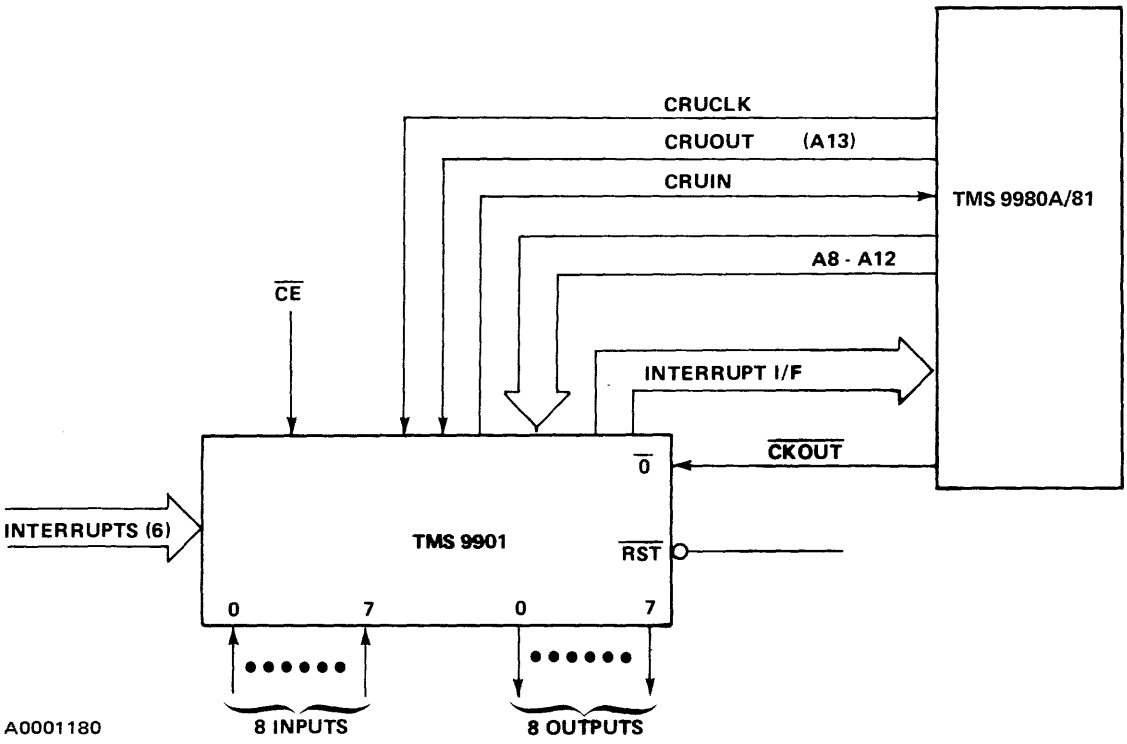


Figure 8-11. 8-Bit CRU Interface

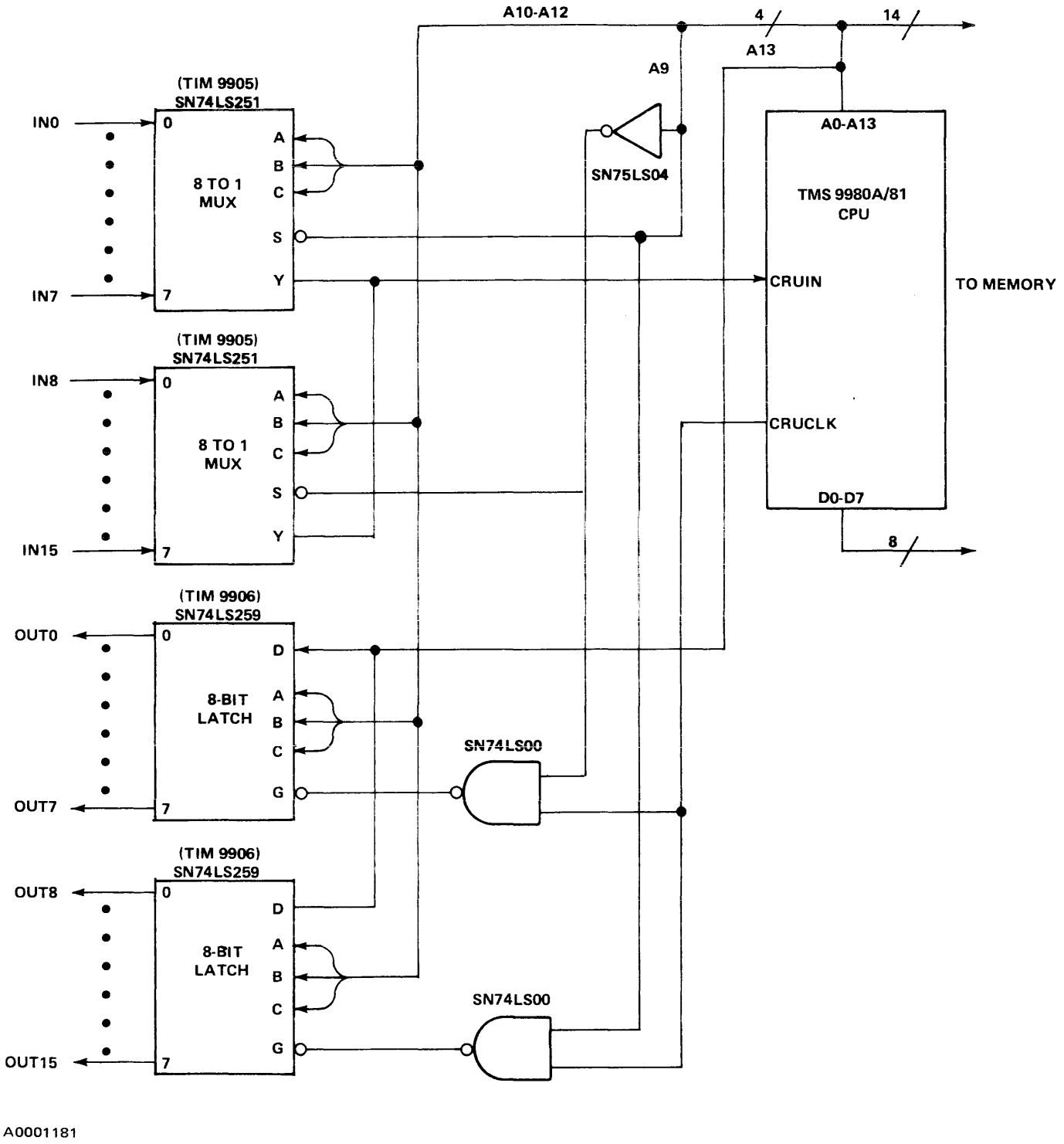


Figure 8-12. TMS 9980A/81 16-Bit Input/Output Interface

TABLE 8-2. EXTERNAL INSTRUCTION CODES

EXTERNAL INSTRUCTION	A13	A0	A1
LREX	H	H	H
CKOF	H	H	L
CKON	H	L	H
RSET	L	H	H
IDLE	L	H	L
CRU INSTRUCTIONS	H/L	L	L

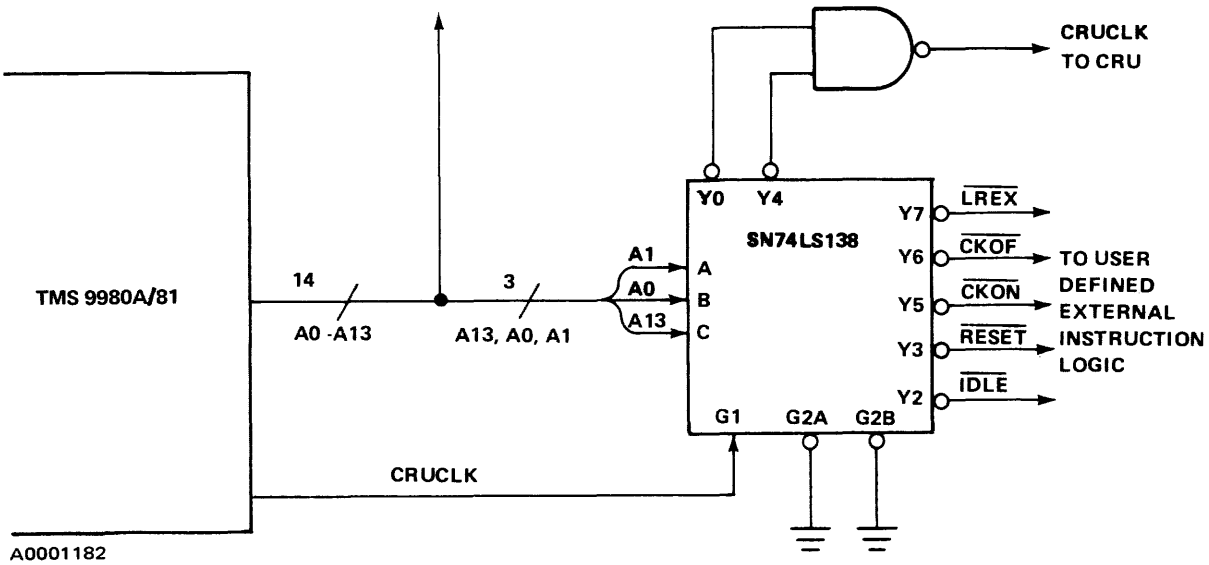


Figure 8-13. External Instruction Decode Logic

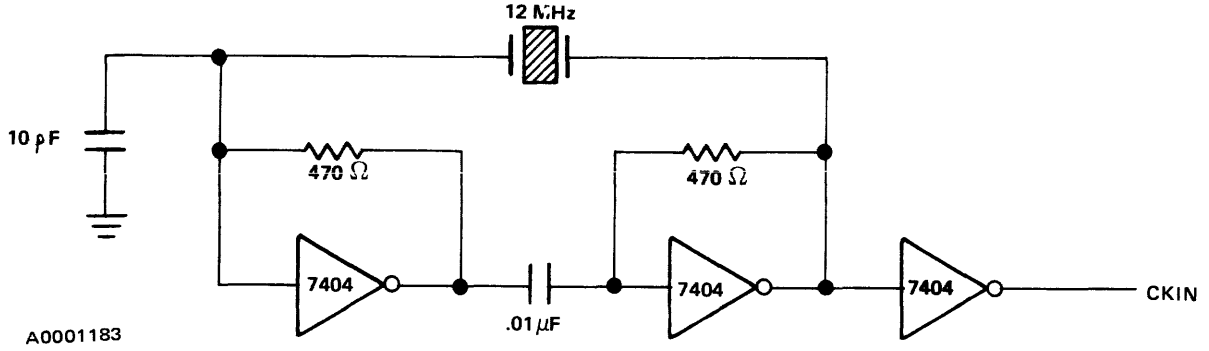


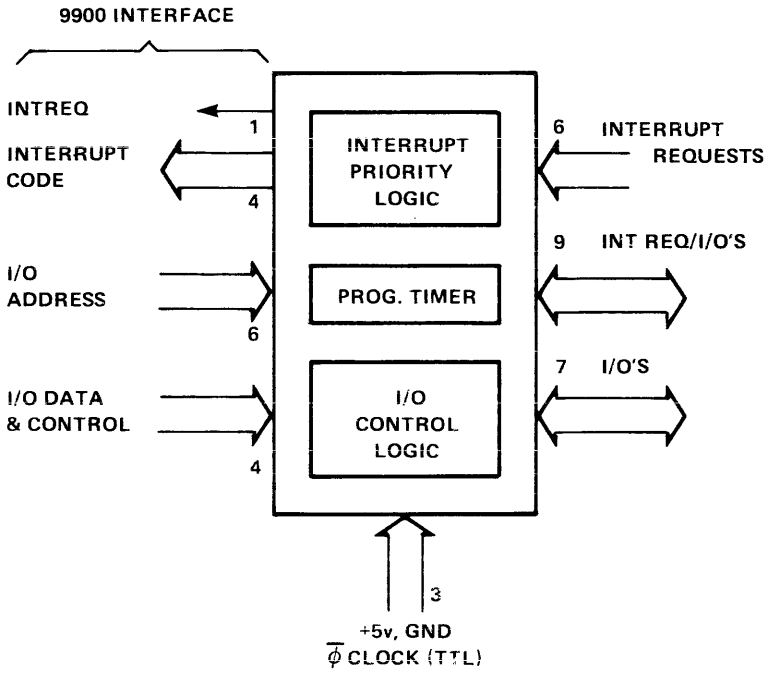
Figure 8-14. Example TMS 9980A/81 Clock Oscillator

SECTION IX

TMS 9900 FAMILY SUPPORT DEVICES

Texas Instruments will continue to introduce additional support circuits and development tools for the TMS 9900 family. The latest support circuits are the TMS 9901 programmable systems interface and the TMS 9902 asynchronous communication controller. Detailed discussion of the use and operation of each are contained in their respective data manual.

The next device to be released is the TMS 9903 synchronous communication controller, which will provide complete synchronous channel control capabilities to the TMS 9900 family. (See Figures 9-1, 9-2, and 9-3.)



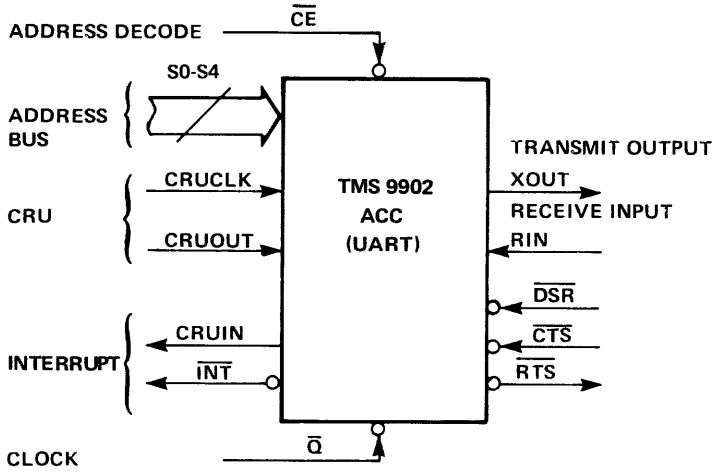
- CRU PERIPHERAL
- 22 SYSTEM INTERFACE PINS
  - 6 DEDICATED INTERRUPT INPUTS
  - INT 3 PROGRAMMABLE AS TIMER INTERRUPT FOR INTERVALS FROM 21 to 699 μSEC
  - 7 DEDICATED I/O PORTS
  - 9 PINS PROGRAMMABLE AS INTERRUPTS OR I/O
- MAY BE STACKED FOR INTERRUPT AND I/O EXPANSION
- SINGLE 5 V SUPPLY
- N-CHANNEL SILICON GATE PROCESS
- 40-PIN DIP

NOTICE: TENTATIVE DATA

THIS INFORMATION PROVIDES TENTATIVE DATA ON A NEW PRODUCT TEXAS INSTRUMENTS RESERVES THE RIGHT TO CHANGE SPECIFICATIONS FOR THIS PRODUCT IN ANY MANNER WITHOUT NOTICE.

A0001184

Figure 9-1. TMS 9901 Programmable System Interface



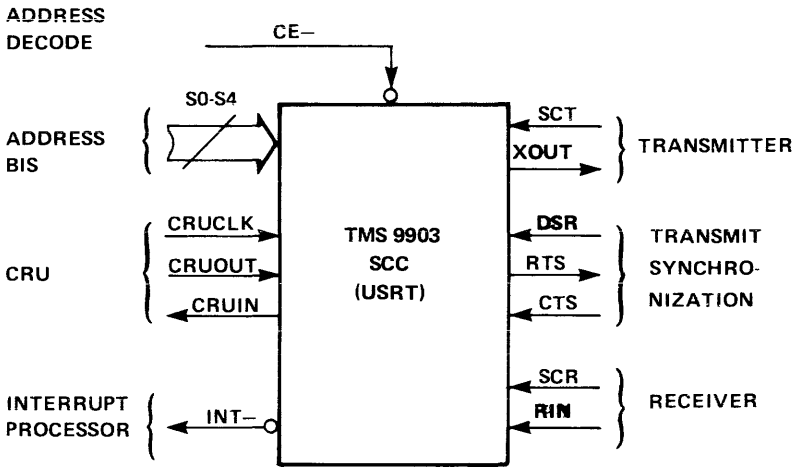
- 9900 CRU PERIPHERAL
- PROGRAMMABLE DATA RATE FROM 110 TO 76,800 BITS/SECOND
- PROGRAMMABLE CHARACTER LENGTH (5-8 BITS), STOP BITS (1, 1½, 2), PARITY (ODD, EVEN, NONE)
- ON-CHIP INTERVAL TIMER (64μSEC TO 16,384 μSEC)
- SINGLE 5v SUPPLY
- N-CHANNEL SILICON GATE PROCESS
- 18 PIN DIP

NOTICE: TENTATIVE DATA

THIS INFORMATION REPRESENTS TENTATIVE DATA ON NEW PRODUCTS. TEXAS INSTRUMENTS RESERVES THE RIGHT TO CHANGE SPECIFICATIONS FOR THESE PRODUCTS IN ANY MANNER WITHOUT NOTICE.

A0001185

Figure 9-2. TMS 9902 Asynchronous Communication Controller



- 9900 CRU PERIPHERAL
- DC TO 250K BITS/SEC DATA RATE
- PROGRAMMABLE SYNC REGISTER AND CHARACTER LENGTH
- BI-SYNC & SDLC COMPATIBLE
- ON-CHIP INTERVAL TIMER (64 μSEC TO 16,384 μSEC)
- SINGLE 5v SUPPLY
- N-CHANNEL SILICON GATE PROCESS
- 20 PIN 300 MIL DIP

NOTICE: TENATIVE DATA

THIS INFORMATION PROVIDES TENATIVE DATA ON A NEW PRODUCT. TEXAS INSTRUMENTS RESERVES THE RIGHT TO CHANGE SPECIFICATIONS FOR THIS PRODUCT IN ANY MANNER WITHOUT NOTICE.

A0001186

Figure 9-3. TMS 9903 Synchronous Communication Controller

APPENDIX A

TMS 9900 FAMILY MACHINE CYCLES

A.1 General Description of Machine Cycles

The TMS 9900 family of microprocessors execute a series of steps to perform an instruction or other operation. This basic step common to all operations is the machine cycle, which requires two clock cycles to execute. (Note: These machine cycles apply equally to the TMS 9980A/81 microprocessor, with the exception of the memory cycle as detailed below.) The TMS 9900 family machine cycles are divided into three categories described in the following paragraphs.

A.1.1 ALU Cycle

The ALU cycle performs an internal operation of the microprocessor. The memory interface control signals and CRU interface control signals are not affected by the execution of an ALU cycle, which takes two clock cycles to execute.

A.1.2 Memory Cycle

The memory cycle primarily performs a data transfer between the microprocessor and the external memory device. Appropriate memory bus control signals are generated by the microprocessor as a result of a memory cycle execution. The memory cycle takes 2+W (where W is the number of wait states) clock cycles to execute.

In the TMS 9980A/81, which has an 8-bit data bus, the memory cycle is composed of two data transfers to move a complete 16-bit word. The TMS 9980A/81 memory cycle takes 4+2W (where W is the number of wait states) clock cycles to execute. For the TMS 9980A/81 the following machine cycle sequences replace the memory sequences used in the instruction discussion.

CYCLE			
1	Memory Read/Write	AB	= Address of most significant byte (A13 = 0)
		DB	= Most significant byte
2	Memory Read/Write	AB	= Address of least significant byte (A13 = 1)
		DB	= Least significant byte

A.1.3 CRU Cycle

The CRU cycle performs a bit transfer between the microprocessor and I/O devices. It takes two clock cycles to execute. The address of the CRU bit is set up during the first clock cycle. For an input operation

the CRUIN line is sampled by the microprocessor during the second clock cycle. For an output operation the data bit is set up on the CRUOUT line at the same time the address is set up. The CRUCLK line is pulsed during the second clock cycle of the CRU output cycle. Please refer to the specific TMS 99XX microprocessor data manual for timing diagrams.

The TMS 9900 executes its operations under the control of a microprogrammed control ROM. Each microinstruction specifies a machine cycle. A microprogram specifies a sequence of machine cycles. The TMS 9900 executes a specific sequence of machine cycles for a specific operation. These sequences are detailed on the following pages. The information can be used by the systems designers to determine the bus contents and other interface behavior at various instants during a certain TMS 9900 operation. This description is maintained at the address bus (AD) and data bus (DB) levels.

A.2 TMS 9900 Machine Cycle Sequences

Most TMS 9900 instructions execution consists of two parts: 1) the data derivation and 2) operation execution. The data derivation sequence depends on the addressing mode for the data. Since the addressing modes are common to all instructions, the data derivation sequence is the same for the same addressing mode, regardless of the instruction. Therefore, the data derivation sequences are described first. These are then referred to in appropriate sequence in the instruction execution description.

A.3 Terms and Definitions

The following terms are used in describing the instructions of the TMS 9900:

TERM	DEFINITION
B	Byte Indicator (1 = byte, 0 = word)
C	Bit count
D	Destination address register
DA	Destination address
IOP	Immediate operand
PC	Program counter
Result	Result of operation performed by instruction
S	Source address register
SA	Source address
ST	Status register
STn	Bit n of status register
SD	Source data register internal to the TMS 9900 microprocessor*
W	Workspace register
SRn	Workspace register n
(n)	Contents of n
Ns	Number of machine cycles to derive source operand
Nd	Number of machine cycles to derive destination operand
AB	Address Bus of the TMS 9900
DB	Data Bus of the TMS 9900
NC	No change from previous cycle

*NOTE: The contents of the SD register remain latched at the last value written by the processor unless changed by the ALU. Therefore, during all memory read or ALU machine cycles the SD register and hence the data bus will contain the operand last written to the data bus by the CPU or the results of the last ALU cycle to have loaded the SD register.

A.4 Data Derivation Sequences

A.4.1 Workspace Register

CYCLE	TYPE	DESCRIPTION
1	Memory Read	AB = Workspace register address
		DB = Operand

A.4.2 Workspace Register Indirect

CYCLE	TYPE	DESCRIPTION
1	Memory Read	AB = Workspace register address
		DB = Workspace register contents
2	ALU	AB = NC
		DB = SD
3	Memory Read	AB = Workspace register content
		DB = Operand

A.4.3 Workspace Register Indirect Auto-Increment (Byte Operand)

CYCLE	TYPE	DESCRIPTION
1	Memory Read	AB = Workspace register address
		DB = Workspace register contents
2	ALU	AB = NC
		DB = SD
3	Memory write	AB = Workspace register address
		DB = (WRn) + 1
4	Memory Read	AB = Workspace register contents
		DB = Operand

A.4.4 Workspace Register Indirect Auto-Increment (Word Operand)

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = Workspace register address
		DB = Workspace register contents
2	ALU	AB = NC
		DB = SD
3	ALU	AB = NC
		DB = SD
4	Memory write	AB = Workspace register address
		DB = (WRn) +2
5	Memory read	AB = Workspace register contents
		DB = Operand

A.4.5 Symbolic

CYCLE	TYPE	DESCRIPTION
1	ALU	AB = NC
		DB = SD
2	ALU	AB = NC
		DB = SD

CYCLE	TYPE	DESCRIPTION
3	Memory read	AB = PC+2 DB = Symbolic address
4	ALU	AB = NC DB = 0000_{16}
5	Memory read	AB = Symbolic address DB = Operand

A.4.6 Indexed

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = Workspace register address DB = Workspace register contents
2	ALU	AB = NC DB = SD
3	Memory read	AB = PC+2 DB = Symbolic address
4	ALU	AB = PC+2 DB = Workspace register contents
5	Memory read	AB = Symbolic address + (WRn) DB = Operand

A.5 Instruction Execution Sequences

A.5.1 A, AB, C, CB, S, SB, SOC, SOCB, SZC, SZCB, MOV, MOVB, COC, CZC, XOR

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
Ns	Insert appropriate sequence for source data addressing mode, from the data derivation sequences	
3+Ns	ALU	AB = NC DB = SD
Nd	Insert appropriate sequence for destination data addressing mode from the data derivation sequences	
3+Ns+Nd	ALU	AB = NC DB = SD
4+Ns+Nd	Memory write	AB = DA (Note 4) DB = Result

NOTES:

1) Since the memory operations of the TMS 9900 microprocessor family fetch or store 16-bit words, the source and the destination data fetched for byte operations are 16-bit words. The ALU operates on

the specified bytes of these words and modifies the appropriate byte in the destination word. The adjacent byte in the destination word remains unaltered. At the completion of the instruction, the destination word, consisting of the modified byte and the adjacent unmodified byte, is stored in a single-memory write operation.

2) For MOVB instruction the destination data word (16 bits) is fetched. The specified byte in the destination word is replaced with the specified byte of the source-data word. The resultant destination word is then stored at the destination address.

3) For MOV instruction the destination data word (16 bits) is fetched although not used.

4) For C, CB, COC, CZC instructions cycle $4+N_s+N_d$ above is an ALU cycle with AB = DA and DB = SD.

A.5.2 MPY (Multiply)

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
Ns	Insert appropriate data derivation sequence according to the source data (multiplier) addressing mode	
3+Ns	ALU	AB = NC DB = SD
4+Ns	Memory read	AB = Workspace register address DB = Workspace register contents
5+Ns	ALU	AB = NC DB = SD
6+Ns	ALU	AB = NC DB = Multiplier
7+Ns	16 ALU	Multiply the two operands AB = NC DB = MSH of partial product
22+Ns	Memory write	AB = Workspace register address DB = MSH of the product
23+Ns	ALU	AB = DA+2 DB = MSH of product
24+Ns	Memory write	AB = DA+2 DB = LSH of the product

A.5.3 DIV (Divide)

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD

CYCLE	TYPE	DESCRIPTION
Ns	Insert appropriate data derivation sequence according to the source data (divisor) addressing mode	
3+Ns	ALU	AB = NC DB = SD
4+Ns	Memory read	AB = Address of workspace register DB = Contents of workspace register
5+Ns	ALU	(Check overflow) AB = NC DB = Divisor
6+Ns	ALU	(Skip if overflow to next instruction fetch) AB = NC DB = SD
7+Ns	Memory read	AB = DA+2 DB = Contents of DA+2
8+Ns	ALU	AB = NC DB = SD
9+Ns	ALU	AB = NC DB = SD
	Divide sequence consisting of Ni cycles where $48 \leqslant Ni \leqslant 32$. Ni is data dependent	AB = NC DB = SD
10+Ns+Ni	ALU	AB = NC DB = SD
11+Ns+Ni	Memory write	AB = Workspace register address DB = Quotient
12+Ns+Ni	ALU	AB = DA+2 DB = Quotient
13+Ns+Ni	Memory write	AB = DA+2 DB = Remainder

A.5.4 XOP

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	Instruction decode AB = NC DB = SD
Ns	Insert appropriate data derivation sequence according to the source data addressing mode	
3+Ns	ALU	AB = NC DB = SD
4+Ns	ALU	AB = NC DB = SA
5+Ns	ALU	AB = NC DB = SD

CYCLE	TYPE	DESCRIPTION
6+Ns	Memory read	AB = 40_{16} + 4 x D
		DB = New workspace pointer
7+Ns	ALU	AB = NC
		DB = SA
8+Ns	Memory write	AB = Address of WR11
		DB = SA
9+Ns	ALU	AB = Address of WR15
		DB = SA
10+Ns	Memory write	AB = Address of workspace register 15
		DB = Status register contents
11+Ns	ALU	AB = NC
		DB = PC+2
12+Ns	Memory write	AB = Address of workspace register 14
		DB = PC+2
13+Ns	ALU	AB = Address of WR13
		DB = SD
14+Ns	Memory write	AB = Address of workspace register 13
		DB = WP
15+Ns	ALU	AB = NC
		DB = SD
16+Ns	Memory read	AB = 42_{16} + 4 x D
		DB = New PC
17+Ns	ALU	AB = NC
		DB = SD

A.5.5 CLR, SETO, INV, NEG, INC, INCT, DEC, DECT

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC
		DB = Instruction
2	ALU	AB = NC
		DB = SD
Ns	Insert appropriate data derivation sequence according to the source data addressing mode	
3+Ns	ALU	AB = NC
		DB = SD
4+Ns	Memory write	AB = Source data address
		DB = Modified source data

NOTE: The operand is fetched for CLR and SETO although not used.

A.5.6 ABS

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC
		DB = Instruction
2	ALU	AB = NC
		DB = SD

CYCLE	TYPE	DESCRIPTION
Ns	Insert appropriate data derivation sequence according to the source data addressing mode	
3+Ns	ALU	Test source data AB = NC DB = SD
4+Ns	ALU	Jump to 5'+Ns if data positive AB = NC DB = SD
5+ns	ALU	Negate source AB = NC DB = SD
6+Ns	Memory write	AB = Source data address DB = Modified source data
5'+Ns	ALU	AB = NC DB = SD

A.5.7 X

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
Ns	Insert the appropriate data derivation sequence according to the source data addressing mode	
3+Ns	ALU	AB = NC DB = SD

NOTE: Add sequence for the instruction specified by the operand.

A.5.8 B

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
Ns	Insert appropriate data derivation sequence according to the source data addressing mode	
3+Ns	ALU	AB = NC DB = SD

NOTE: The source data is fetched, although it is not used.

A.5.9 BL

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
Ns	Insert appropriate data derivation sequence according to the source data addressing mode	
3+Ns	ALU	AB = NC DB = SD
4+Ns	ALU	AB = Address of WR11 DB = SD
5+Ns	Memory write	AB = Address of WR11 DB = PC+2

NOTE: The source data is fetched although it is not used.

A.5.10 BLWP

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
Ns	Insert appropriate data derivation sequence according to the source data addressing mode	
3+Ns	ALU	AB = NC DB = SD
4+Ns	ALU	AB = Address of WR15 DB = NC
5+Ns	Memory write*	AB = Address of workspace register 15 DB = Status register contents
6+Ns	ALU	AB = NC DB = PC+2
7+Ns	Memory write	AB = Address of workspace register 14 DB = PC+2
8+Ns	ALU	AB = Address or workspace register 13 DB = SD
9+Ns	Memory write	AB = Address of workspace register 13 DB = WP
10+Ns	ALU	AB = NC DB = SD
11+Ns	Memory read	AB = Address of new PC DB = New PC
12+Ns	ALU	AB = NC DB = SD

A.5.11 LDCR

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
Ns	Insert appropriate data derivation sequence	
3+Ns	ALU	AB = NC DB = SD
4+Ns	ALU	AB = NC DB = SD
5+Ns	ALU	AB = Address of WR12 DB = SD
6+Ns	ALU	AB = Address of WR12 DB = SD
7+Ns	Memory read	AB = Address of WR12 DB = Contents of WR12
8+Ns	ALU	AB = NC DB = SD
C	Enable CRUCLK. Shift next bit onto CRUOUT line. Increment CRU bit address on AB. Iterate this sequence C times, where C is number of bits to be transferred.	AB = Address + 2 Increments C Times DB = SD
9+Ns+C	ALU	AB = NC DB = SD

A.5.12 STCR

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
Ns	Insert appropriate data derivation sequence according to the source data addressing mode	
3+Ns	ALU	AB = NC DB = SD
4+Ns	Memory read	AB = Address of WR12 DB = Contents of WR12
5+Ns	ALU	AB = NC DB = SD

CYCLE	TYPE	DESCRIPTION
6+Ns	ALU	AB = NC DB = SD
C	Input selected CRU bit. Increment CRU bit address. Iterate this sequence C times where C is the number of CRU bits to be input.	AB = Address + 2 C times DB = SD
7+Ns+C	ALU	AB = NC DB = SD
8+Ns+C	ALU	AB = NC DB = SD
C′	Right adjust (with zero fill) byte (if C < 8) or word (if 8 < C < 16).	AB = NC DB = SD
C′	= 8-C-1 if C ⩽ 8 = 16-C-1 if < C ⩽ 16	
9+Ns+C+C′	ALU	AB = NC DB = SD
10+Ns+C+C′	ALU	AB = NC DB = SD
11+Ns+C+C′	ALU	AB = Source address DB = SD
12+Ns+C+C′	Memory write	AB = Source address DB = I/O data

NOTE: For STCR instruction the 16-bit word at the source address is fetched. If the number of CRU bits to be transferred is ⩽ 8, the CRU data is right justified (with zero fill) in the specified byte of the source word and source data word thus modified is then stored back in memory. If the bits to be transferred is > 8 then the source data fetched is not used. The CRU data in this case is right justified in 16-bit word which is then stored at the source address.

A.5.13 SBZ, SBO

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
3	ALU	AB = NC DB = SD
4	Memory read	AB = Address of WR12 DB = Contents of WR12
5	ALU	AB = NC DB = SD
6	CRU	Set CRUOUT = 0 for SBZ = 1 for SBO Enable CRUCLK

CYCLE	TYPE	DESCRIPTION
		AB = CRU bit address
		DB = SD

A.5.14 TB

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC
		DB = Instruction
2	ALU	AB = NC
		DB = SD
3	ALU	AB = NC
		DB = SD
4	Memory read	AB = Address of WR12
		DB = Contents of WR12
5	ALU	AB = NC
		DB = SD
6	CRU	Set ST(2) = CRUIN
		AB = Address of CRU bit
		DB = SD

A.5.15 JEQ, JGT, JH, JHE, JL, JLE, JLT, JMP, JNC, JNE, JNO, JOC, JOP

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC
		DB = Instruction
2	ALU	AB = NC
		DB = SD
3	ALU	Skip to cycle #5 if TMS 9900 status satisfies the specified jump condition
		AB = NC
		DB = SD
4	ALU	AB = NC
		DB = Displacement value
5	ALU	AB = NC
		DB = SD

A.5.16 SRA, SLA, SRL, SRC

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC
		DB = Instruction
2	ALU	AB = NC
		DB = SD
3	Memory read	AB = Address of the workspace register
		DB = Contents of the workspace register
4	ALU	Skip to cycle #9 if C ≠ 0
		C = Shift count
		AB = NC
		DB = SD
5	ALU	AB = NC
		DB = SD

CYCLE	TYPE	DESCRIPTION
6	Memory read	AB = Address of WRO DB = Contents of WRO
7	ALU	AB = Source address DB = SD
8	ALU	AB = NC DB = SD
9		AB = NC DB = SD
C	Shift the contents of the specified workspace register in the specified direction by the specified number of bits. Set appropriate status bits.	
9+C	Memory write	AB = Address of the workspace register DB = Result
10+C	ALU	Increment PC AB = NC DB = SD

A.5.17 AI, ANDI, ORI

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
3	ALU	AB = NC DB = SD
4	Memory read	AB = Address of workspace register DB = Contents of workspace register
5	Memory read	AB = PC+2 DB = Immediate operand
6	ALU	AB = NC DB = SD
7	Memory write	AB = Address of workspace register DB = Result of instruction

A.5.18 CI

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = NC
3	Memory read	AB = Address of workspace register DB = Contents of workspace register
4	ALU	AB = NC DB = SD
5	Memory read	AB = PC+2 DB = Immediate operand

CYCLE	TYPE	DESCRIPTION
6	ALU	AB = NC DB = SD
7	ALU	AB = NC DB = SD

A.5.19 LI

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
3	ALU	AB = NC DB = SD
4	Memory read	AB = PC+2 DB = Immediate operand
5	ALU	AB = Address of workspace register DB = SD
6	Memory write	AB = Address of workspace register DB = Immediate operand

A.5.20 LWPI

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
3	ALU	AB = NC DB = SD
4	Memory read	AB = PC+2 DB = Immediate operand
5	ALU	AB = NC DB = SD

A.5.21 LIMI

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC DB = Instruction
2	ALU	AB = NC DB = SD
3	ALU	AB = NC DB = SD
4	Memory read	AB = PC+2 DB = Immediate data
5	ALU	AB = NC DB = SD
6	ALU	AB = NC DB = SD
7	ALU	AB = NC DB = SD

A.5.22 STWP, STST

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC
		DB = Instruction
2	ALU	AB = NC
		DB = SD
3	ALU	AB = Address of workspace register
		DB = SD
4	Memory write	AB = Address of the workspace register
		DB = TMS 9900 internal register contents (WP or ST)

A.5.23 CKON, CKOF, LREX, RSET

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC
		DB = Instruction
2	ALU	AB = NC
		DB = SD
3	ALU	AB = NC
		DB = SD
4	CRU	Enable CRUCLK
		AB = External instruction code
		DB = SD
5	ALU	AB = NC
		DB = SD
6	ALU	AB = NC
		DB = SD

A.5.24 IDLE

CYCLE	TYPE	DESCRIPTION
1	Memory read	AB = PC
		DB = Instruction
2	ALU	AB = NC
		DB = SD
3	ALU	AB = NC
		DB = SD
4	CRU	Enable CRUCLK
		AB = Idle code
		DB = SD
5	ALU	AB = NC
		DB = SD
6	ALU	AB = NC
		DB = NC

A.6 Machine-Cycle Sequences in Response to External Stimuli

A.6.1 RESET

CYCLE	TYPE	DESCRIPTION
1*	ALU	AB = NC
		DB = SD
2	ALU	AB = NC
		DB = SD
3	ALU	AB = 0
		DB = 0
4	Memory read	AB = 0
		DB = Workspace pointer
5	ALU	AB = NC
		DB = Status
6	Memory write	AB = Address of WR15
		DB = Contents of Status register
7	ALU	AB = NC
		DB = PC
8	Memory write	AB = Address of workspace register 14
		DB = PC+2
9	ALU	AB = Address of WR13
		DB = SD
10	Memory write	AB = Address of workspace register 13
		DB = WP
11	ALU	AB = NC
		DB = SD
12	Memory read	AB = 2
		DB = New PC
13	ALU	AB = NC
		DB = SD

A.6.2 LOAD

CYCLE	TYPE	DESCRIPTION
1**	ALU	AB = NC
		DB = SD
2	Memory read	AB = $FFFC_{16}$
		DB = Contents of $FFFC_{16}$
3	ALU	AB = NC
		DB = Status
4	Memory write	AB = Address WR15
		DB = Contents of status register
5	ALU	AB = NC
		DB = PC
6	Memory write	AB = Address of workspace
		DB = PC+2
7	ALU	AB = Address of WR13
		DB = SD
8	Memory write	AB = Address of workspace register 13
		DB = WP

*Occurs immediately after RESET is released.
**Occurs immediately after last clock cycle of preceding instruction.

CYCLE	TYPE	DESCRIPTION
9	ALU	AB = NC DB = SD
10	Memory Read	AB = FFFE DB = New PC
11	ALU	AB = NC DB = SD

A.6.3 Interrupts

CYCLE	TYPE	DESCRIPTION
1*	ALU	AB = NC DB = SD
2	Memory read	AB = Address of interrupt vector DB = WP
3	ALU	AB = NC DB = Status
4	Memory write	AB = Address of WR15 DB = Status
5	ALU	AB = NC DB = PC
6	Memory write	AB = Address of workspace register 14 DB = PC+2
7	ALU	AB = Address of WR13 DB = SD
8	Memory write	AB = Address of workspace register 13 DB = WP
9	ALU	AB = NC DB = SD
10	Memory read	AB = Address of second word of interrupt vector DB = New PC
11	ALU	AB = NC DB = SD

*Occurs immediately after last clock cycle of preceding instruction

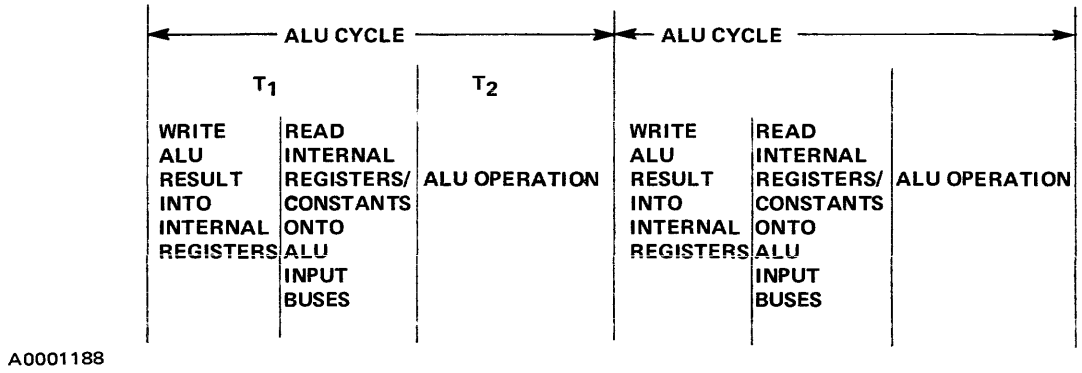


Figure A-1. ALU Cycle

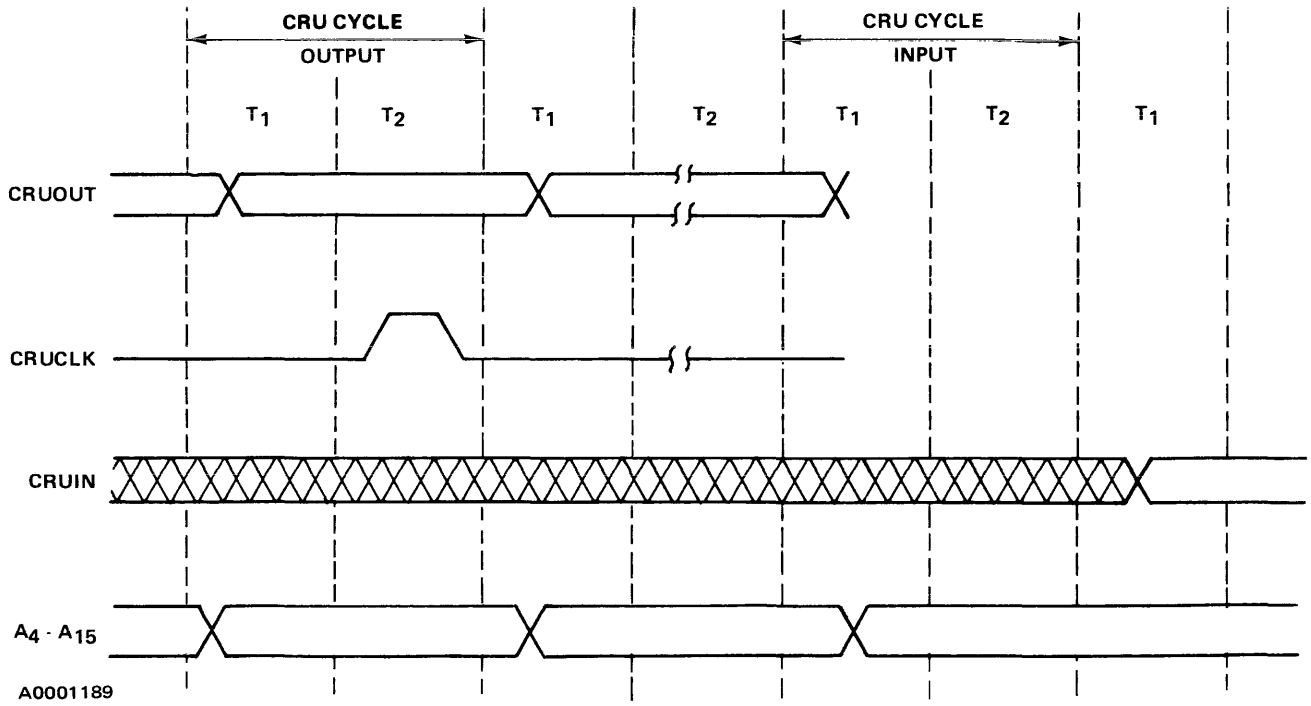


Figure A-2. CRU Cycle

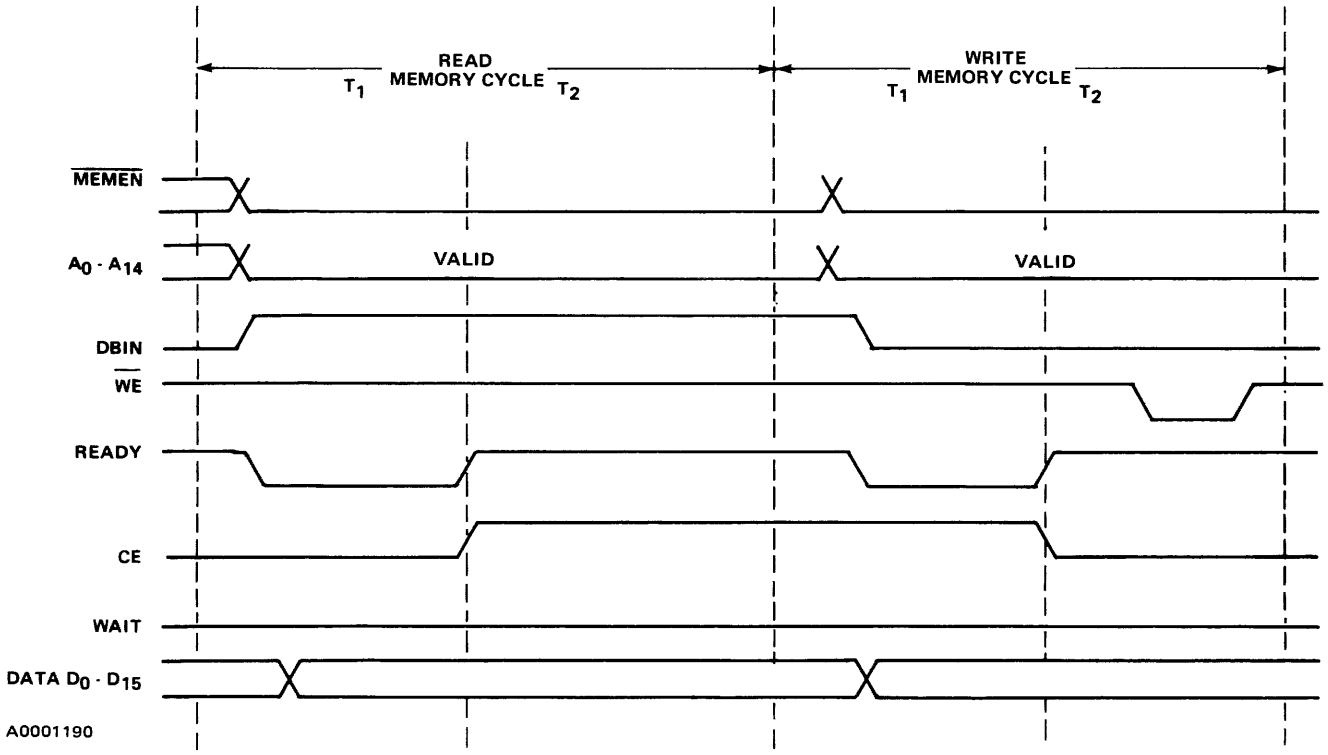


Figure A-3. TMS 9900 Memory Cycle (No Wait States)

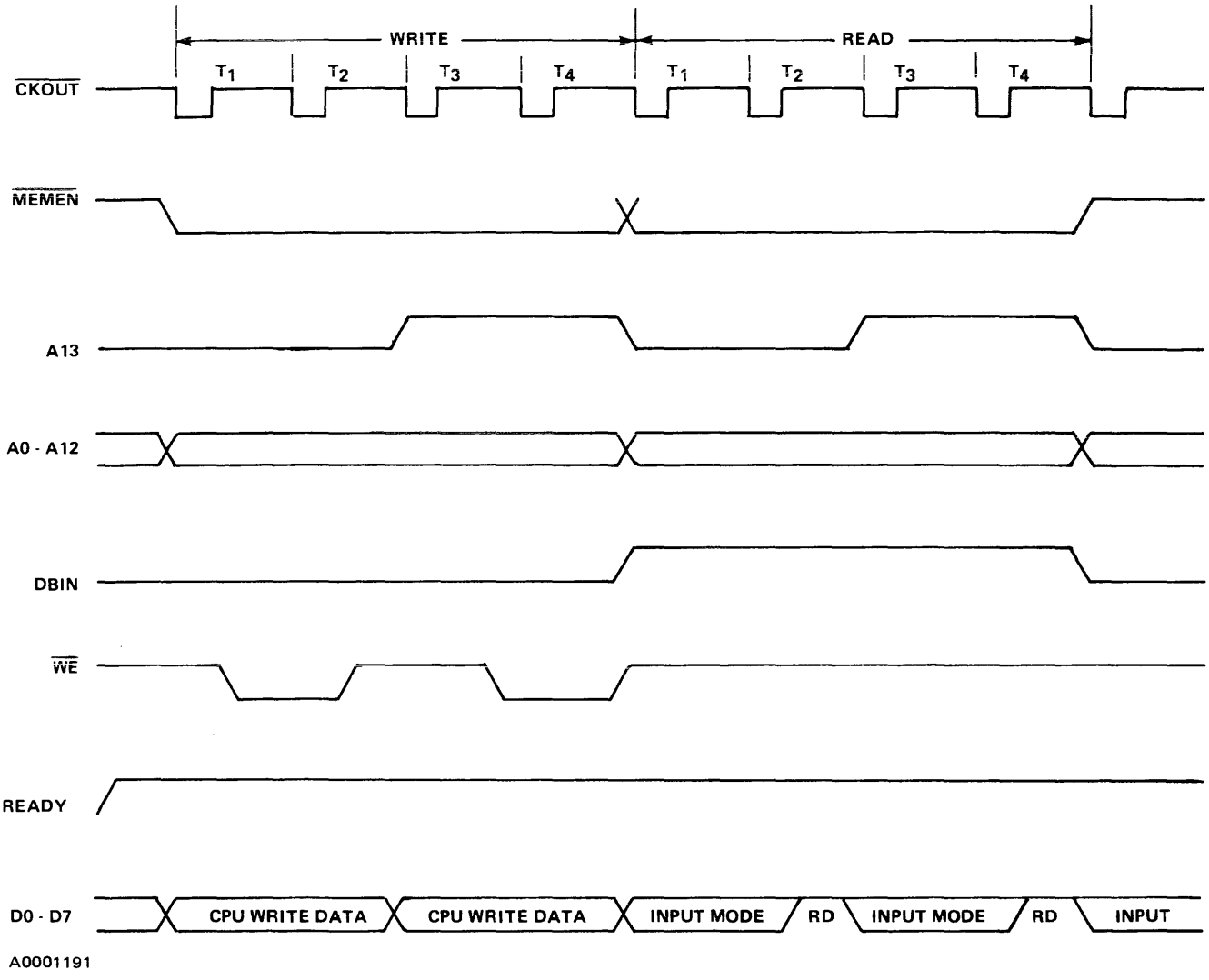


Figure A-4. TMS 9980A/81 Memory Cycle (No Wait States)

TI authorized distributors

ALABAMA

Hallmark Electronics
P.O. Box 1133/(205) 539-0691
Huntsville, Alabama 35801

ARIZONA

Cramer Arizona
2646 E. University/(602) 263-1112
Phoenix, Arizona 85034

R. V. Weatherford Company
3311 W. Earll/(602) 272-7144
Phoenix, Arizona 85017

CALIFORNIA

Cramer/Los Angeles
17201 Daimler St./(714) 979-3000
Irvine, California 92705

Cramer/San Diego
8913 Complex Drive, Suite E
(714) 565-1881
San Diego, California 92123

Cramer/San Francisco
720 Palomar Avenue/(408) 739-3011
Sunnyvale, California 94086

Kierulff Electronics, Inc.
2585 Commerce Way/(213) 685-5511
Los Angeles, California 90040

Kierulff Electronics, Inc.
3969 E. Bayshore/(415) 968-6292
Palo Alto, California 94303

Kierulff Electronics, Inc.
8797 Balboa Avenue/(714) 278-2112
San Diego, California 92123

Radio Products Sales, Inc.
350 S. Kellogg Ave., Suite E
(805) 964-6823
Goleta, California 93017

Radio Products Sales, Inc.
1501 S. Hill St./(213) 748-1271
Los Angeles, California 90015

Radio Products Sales, Inc.
7889 Clairemont Mesa Blvd./(714) 292-5611
San Diego, California 92111

Semiconductor Concepts
21201 Oxnard Street/(213) 884-4560
Woodland Hills, California 91364

TI Supply Company
831 S. Douglas St./(213) 973-2571
El Segundo, California 90245

TI Supply Company
776 Palomar Avenue/(408) 732-5555
Sunnyvale, California 94086

Time Electronics
900 West Olive Street/(213) 649-6900
Inglewood, California 90301

Time Electronics
2629 Terminal Blvd./(415) 965-8000
Mountainview, California 94043

R. V. Weatherford Company
6921 San Fernando Rd./(213) 849-3451
Glendale, California 91201

R. V. Weatherford Company
1550 Babbitt Avenue/(714) 633-9633
Anaheim, California 92805

R. V. Weatherford Company
3240 Hillview Drive/(415) 493-5373
Stanford Industrial Park
Palo Alto, California 94304

R. V. Weatherford Company
1095 East 3rd Street/(714) 623-1261
Pomona, California 91766

R. V. Weatherford Company
7872 Raytheon/(714) 278-7400
San Diego, California 92111

COLORADO

Cramer/Denver
5465 East Evans Place/(303) 758-2100
Denver, Colorado 80222

Kierulff Electronics
10890 East 47th Ave./(303) 371-6500
Denver, Colorado 80239

CONNECTICUT

Arrow Electronics, Inc.
295 Treadwell St./(203) 248-3801
Hamden, Connecticut 06514

Cramer/Connecticut
35 Dodge Avenue/(203) 239-5641
North Haven, Connecticut 06473

Milgray/Connecticut
378 Boston Post Road/(203) 795-0714
Orange, Connecticut 06477

FLORIDA

Arrow Electronics
1001 N.W. 62nd Street, Suite 402
(305) 776-7790
Ft. Lauderdale, Florida 33300

Cramer/EW Orlando
345 N. Graham Ave./(305) 894-1511
Orlando, Florida 32814

Cramer/EW Hollywood
4035 North 29th Ave./(305) 923-8181
Hollywood, Florida 33020

Hallmark Electronics
7233 Lake Ellenor Dr./(305) 855-4020
Orlando, Florida 32809

Hall-Mark Electronics
1302 West McNab Rd./(305) 971-9280
Fort Lauderdale, Florida 33310

GEORGIA

Cramer/EW Atlanta
3923 Oakcliff Industrial Court
(404) 448-9050
Atlanta, Georgia 30340

ILLINOIS

Cramer/Chicago
1911 South Busse Rd./(312) 593-8230
Mt. Prospect, Illinois 60056

Newark Electronics Corp.
500 N. Pulaski Road/(312) 638-4411
Chicago, Illinois 60624

Semiconductor Specialists
P.O. Box 66125
O'Hare International Airport
(312) 279-1000
Chicago, Illinois 60666

TI Supply Company
2420 E. Oakton/(312) 593-7660
Arlington Heights, Illinois 60005

INDIANA

Graham Electronics
133 S. Pennsylvania Street
(317) 634-8202
Indianapolis, Indiana 46204

IOWA

DEECO, Inc.
2500 16th Avenue S.W./(319) 365-7551
Cedar Rapids, Iowa 52406

MARYLAND

Arrow Electronics, Inc.
4801 Benson Ave./(202) 737-1700
Baltimore, Maryland 21227

Kierulff Electronics, Inc.
16021 Industrial Drive/(301) 948-0250
Gaithersburg, Maryland 20760

Milgray/Washington, Inc.
5405 Lafayette Place/(301) 864-1111
Hyattsville, Maryland 20781

Technico Inc.
9130 Red Branch Road/(301) 461-2200
Columbia, Maryland 21209

MASSACHUSETTS

Cramer Electronics, Inc.
85 Wells Avenue/(617) 969-7700
Newton, Massachusetts 02159

Kierulff Electronics, Inc.
13 Fortune Ave./(617) 667-8331
Billerica, Massachusetts 01821

TI Supply Company
504 Totten Pond Road/(617) 890-0510
Waltham, Massachusetts 02154

MICHIGAN

Newark Electronics, Inc.
20700 Hubbell Ave./(313) 967-0600
Detroit, Michigan 48237

Newark Electronics
3645 Linden S.E./(616) 241-6681
Wyoming, Michigan 49508

MINNESOTA

Arrow Electronics
9700 Newton Ave. South/(612) 888-5522
Bloomington, Minnesota 55431

Cramer/Minnesota
7275 Bush Lake Road/(612) 835-7811
Edina, Minnesota 55435

Semiconductor Specialists
8030 Cedar Ave. South/(612) 854-8841
Minneapolis, Minnesota 55420

MISSOURI

LCOMP-St. Louis, Inc.
2605 South Hanley Rd./(314) 647-5505
St. Louis, Missouri 63144

LCOMP-Kansas City, Inc.
2211 Riverfront Dr./(816) 221-2400
Kansas City, Missouri 64120

Semiconductor Specialists
3805 N. Oak Trafficway/(816) 452-3900
Kansas City, Missouri 64116

Semiconductor Specialists
1020 Anglum Road/(314) 731-2400
Hazelwood, Missouri 63042

NEW JERSEY

Arrow Electronics
Pleasant Valley Ave./(609) 235-1900
Moorestown, New Jersey 08057

Arrow Electronics
285 Midland Ave./(201) 797-5800
Saddlebrook, New Jersey 07662

Cramer/Pennsylvania
12 Springdale Road/(215) 923-5950
Cherry Hill, New Jersey 08003

Cramer/New Jersey
1 Barrett Avenue/(201) 935-5600
Moonachie, New Jersey 07074

General Radio Supply Company, Inc.
600 Penn St./(609) 964-8560
Camden, New Jersey 08102

Kierulff Electronics
5 Industrial Drive/(201) 935-2120
Rutherford, New Jersey 07070

Milgray/Delware Valley, Inc
1165 Marlkress Rd
(609) 424-1300/(215) 228-2000
Cherry Hill, New Jersey 08034

TI Supply Company
301 Central Ave./(201) 382-6400
Clark, New Jersey 07066

NEW MEXICO

Cramer/New Mexico
137 Vermont N.E./(505) 265-5767
Albuquerque, New Mexico 87108

R. V. Weatherford Company
2425 Alamo S.E./(505) 842-0863
Albuquerque, New Mexico 87106

NEW YORK

Arrow Electronics, Inc.
900 Broad Hollow Rd./(516) 694-6800
Farmingdale, New York 11735

Arrow Electronics, Inc.
Old Route 9/(914) 896-7530
Fishkill, New York 12524

Cramer/Rochester
3000 S. Winton Road/(716) 275-0300
Rochester, New York 14623

Cramer/Syracuse
6716 Joy Road/(315) 437-6671
East Syracuse, New York 13057

Cramer/Long Island
29 Oser Ave./(516) 231-5600
Hauppage, New York 11787

Milgray Electronics Inc.
191 Hanse Avenue/(516) 546-6000
Freeport, New York 11520

Rochester Radio Supply Co., Inc.
140 W. Main St./(716) 454-7800
Rochester, New York 14614

NORTH CAROLINA

Cramer/EW Winston-Salem
938 Burke St./(919) 725-8711
Winston-Salem, North Carolina 27102

Hallmark Electronics
3000 Industrial Dr./(919) 832-4465
Raleigh, North Caroline 27609

OHIO

Arrow Electronics, Inc.
23500 Mercantile Rd./(216) 464-2000
Cleveland, Ohio 44122

Arrow Electronics
3100 Plainfield Rd./(513) 253-9176
Kettering, Ohio 45432

Cramer/Cleveland
5835 Harper Road/(216) 248-8400
Cleveland, Ohio 44139

ESCO Electronics Inc.
221 Crane St./(513) 226-1133
Dayton, Ohio 45403

OKLAHOMA

TI Supply Company
P.O. Box Drawer "T", Admiral Station
12151 E. Skelly Drive/(918) 437-4555
Tulsa, Oklahoma 74115

OREGON

Almac-Stroum Electronics
P.O. Box 25444
8888 S.W. Canyon Road/(503) 292-3534
Portland, Oregon 97225

PENNSYLVANIA

Arrow Electronics, Inc.
Pleasant Valley Ave./(609) 235-1900
Moorestown, New Jersey 08057

General Radio Supply Co.
600 Penn St./(609) 964-8560
Camden, New Jersey 08102

Milgray/Delaware Valley, Inc.
1165 Marlkress Rd.
(609) 424-1300/(215) 228-2000 Phila.
Cherry Hill, New Jersey 08034

TEXAS

Harrison Equipment Company, Inc.
1616 McGowen/(713) 652-4700
Houston, Texas 77004

R. V. Weatherford Company
3500 West T. C. Jester Blvd.
(713) 688-7406
Houston, Texas 77018

TI Supply Company
6000 Denton Drive/(214) 238-6823
Dallas, Texas 75235

TI Supply Company
8600 Commerce Park/(713) 777-6011
Houston, Texas 77042

UTAH

Standard Supply Company
3424 S. Main/(801) 486-3371
Salt Lake City, Utah 84110

WASHINGTON

Almac-Stroum Electronics
5811 Sixth Ave. So./(206) 763-2300
Seattle, Washington 98108

Kierulff Electronics, Inc.
5940 6th Ave. So./(206) 763-1550
Seattle, Washington 98108

WISCONSIN

Arrow Electronics, Inc.
2925 S. 160th Street/(414) 782-2801
New Berlin, Wisconsin 53151

Semiconductor Specialists
10855 West Potter Rd./(414) 257-1330
Wauwatosa, Wisconsin 53226

AUSTRALIA

Adelaide: A. J. Ferguson Pty. Ltd.
Melbourne: A. J. Ferguson & Co. Pty. Ltd.

AUSTRIA

Vienna: TRANSISTOR Vertriebsges. mbH,
0222/82 94 51

BELGIUM

Brussels: Avi-Elec, 2/733 96 00
Deurne: Greibay Electronics, 31/25 19 25

CANADA

Calgary: Canadian Electronics, Ltd.
(403) 287-1800
Downsview: CESCO Electronics, Ltd.
(416) 661-9222
Edmonton: Canadian Electronics, Ltd.
(403) 452-9393
Montreal: CESCO Electronics, Ltd.
(514) 735-5511
Montreal: Future Electronics
(514) 735-5775
Ottawa: CESCO Electronics, Ltd.
(613) 729-5118
Ottawa: Future Electronics
(613) 232-7757
Quebec City: CESCO Electronics, Ltd.
(418) 524-4641
Rexdale: Future Electronics
(416) 677-7820
Vancouver: Canadian Electronics, Ltd
(604) 324-7911
Vancouver: Future Electronics
(604) 261-1335
Winnipeg: Canadian Electronics, Ltd.
(204) 947-1321

DENMARK

Herlev: TI Supply, 91 74-00

FINLAND

Helsinki: TI Supply, 408 300

FRANCE

Le Plessis Robinson: TISCO, 1-630 23 43
Lagarde: Sofer, 94-27 16 10
Lyon: Radiolex, 78-24 51 72
Lyon: Tekelec, 78-74 37 40
Marseille: Industrielle Electrique,
91-50 52 06
Marseille: Tekelec, 91-47 22 20
Metz: Fachet, 87-68 88 63
Meylan: TISCO, 76-90 45 74
Montrouge: P.E.P., 1-735 33 20
Paris: Pariner, 1-878 65 55
Paris: Radio Voltaire, 1-357 50 11
Rennes: Tekelec, 99-50 62 35
Rungis: Electronique MS, 1-686 74 25
Roubaix: Eltec, 20-70 56 19
Sevres: Tekelec, 1-626 02 35
Strasbourg: Electronique MS, 88-32 88 32
Tarbes: Tarbelec, 62-93 10 82

GERMANY

Aachen: Schiffers Elektronik, 0241/3 05 53
Berlin: TISCO, 030-74 44 041
Essen: TISCO 0201-20 916
Frankfurt a.M.: Spoerle Electronic,
06103/6 20 31-38
Frankfurt-Griesham: TISCO, 0611-39 90 61
Gottingen: Retron GmbH, 0551/6 40 07
Hamburg: TISCO, 040-22 96 478
Hamburg: Walter Kluxen, 040/2 48 91
Hannover: TISCO, 0511-55 60 41
Munchen: Celdis GmbH, 089/45 43 06
Munchen: Neumuller GmbH, 089/59 91-1
Munchen: TISCO, 089-32 50 11
Nurnberg: Celdis GmbH, 0911/20 90 10
Schwieberdingen: Elkose, 07150/31041
Sprendlingen/Ffm: Spoerle Electronic,
06103/604-1
Stuttgard: TISCO, 0711-54-70 01
Toulouse: Electron, 61-62 82 85
Toulouse: Tekelec, 61-40 24 90
Toulouse: TISCO, 61-80 64 70
Wuppertal-Elberfeld: H. M. Muller,
02121/42 60 16

ITALY

Bologna: Lasi, 051/46 78 80
Catania: Thyristor, 095/37 20 45
Firenze: Paoletti Fer., 055/29 45 74
Genova: Pasini Elt., 010/56 10 15
Milano: Lasi Elt., 02/688 63 68
Napoli: Telelux, 081/61 11 33
Padova: IDAC, 049/65 77 21
Roma: Cramer Italia, 06/51 39 387
Roma: SFERA, 06/839 31 72
Torino: Carter, 011/59 25 12
Torino: FIRET, 011/66 65 72
Tortorero Lido/Teramo: De Dominicis,
095/37 20 45

JAPAN

Tokyo: TI Supply, 402-6171

NETHERLANDS

Amsterdam-Schiphol: TI Supply,
020-17 36 36

NEW ZEALAND

Auckland: David J. Reid (N. Z.) Ltd.,
492-189

NORWAY

Oslo: Ingenioforma Morgenstierne & Co.,
37 29 40

PUERTO RICO

Cramer De Puerto Rico
Subdivision General/(809) 892-2600
Bo. Retrio, San German
Puerto Rico 00753

SOUTH AFRICA

Johannesburg: Associated Electronics Pty. Ltd.

SPAIN

Barcelona: Difitronic S.A., 253 24 57

SWEDEN

Stockholm: A. B. Gosta Backstrom,
54 03 90

SWITZERLAND

Zurich: Fabrimex AG, 01/47 06 70

UNITED KINGDOM

Birmingham: TI Supply, (021) 743-5293
Derby: Quarndon Electronics Ltd.,
0332-32651
Edinburgh: TI Supply, (031) 229-1481
Enfield: Blueline Electronic Components,
01-366 6371
Harlow: Blueline Electronic Components,
0279-29588
Harlow: Mogul Electronics Ltd.,
029-39771
London (Slough): Anzac Components Ltd.,
0753-35446
London (Slough): TI Supply, 0753-33411
Portsmouth: SDS Components Ltd.,
0705-65311
Southampton: TI Supply, 0703-27741
Stockport: TI Supply, 061-432 0645

TI worldwide sales offices

ALABAMA

4717 University Drive, Suite 101
Huntsville, Alabama 35805
205-837-7530

ARIZONA

P.O. Box 35160
8102 N 23rd Ave., Suite A
Phoenix, Arizona 85069
602-249-1313

CALIFORNIA

3186J Airway
Costa Mesa, California 92626
714-540-7311

831 S. Douglas St.
El Segundo, California 90245
213-973-2571

7827 Convoy Ct., Suite 412
San Diego, California 92111
714-279-2622

P.O. Box 9064
776 Palomar Avenue
Sunnyvale, California 94086
408-732-1840

COLORADO

9725 E. Hampden St., Suite 301
Denver, Colorado 80231
303-751-1780

CONNECTICUT

2405 Whitney Avenue
Hamden, Connecticut 06518
203-281-0074

FLORIDA

4600 West Commercial Blvd.
Fort Lauderdale, Florida 33319
305-733-3300

1850 Lee Road, Suite 115
Winter Park, Florida 32789
305-644-3535

ILLINOIS

515 W. Algonquin
Arlington Heights, Illinois 60005
312-640-3000

INDIANA

2020 Inwood Drive
Ft. Wayne, Indiana 46805
219-424-5174

2346 S. Lynhurst Dr., Suite 101
Indianapolis, Indiana 46241
317-248-8555

MARYLAND

6024 Jamina Downs
Columbia, Maryland 21045
301-997-4755

MASSACHUSETTS

504 Totten Pond Road
Waltham, Mass. 02154
617-890-7400

MICHIGAN

Central Park Plaza
26211 Central Park Blvd., Suite 215
Southfield, Michigan 48076
313-353-0830

MINNESOTA

A.I.C. Bldg., Suite 202
7615 Metro Blvd.
Edina, Minn. 55435
612-835-2900

MISSOURI

8080 Ward Parkway
Kansas City, Missouri 64114
816-523-2500

NEW JERSEY

1245 Westfield Ave.
Clark, New Jersey 07066
201-574-9800

NEW MEXICO

1101 Cardenas Drive, N.E.,
Room 215
Albuquerque, New Mexico 87110
505-265-8491

NEW YORK

6700 Old Collamer Rd.
East Syracuse, New York 13057
315-463-9291

112 Nanticoke Ave., P.O. Box 618
Endicott, New York 13760
607-754-3900

201 South Avenue
Poughkeepsie, New York 12601
914-473-2900

1210 Jefferson Rd.
Rochester, New York 14623
716-461-1800

1 Huntington Quadrangle, Suite 1C01
Melville, New York 11746
516-293-2560

NORTH CAROLINA

1 Woodlawn Green, Woodlawn Road
Charlotte, North Carolina 28210
704-527-0930

OHIO

Belmont Bldg., Suite 120
28790 Chagrin Blvd.
Cleveland, Ohio 44122
216-464-2990

Hawley Bldg., Suite 101
4140 Linden Avenue
Dayton, Ohio 45432
513-253-3121

OREGON

10700 S.W. Beaverton Hwy.
Suite 11
Beaverton, Oregon 97005
503-643-6759

PENNSYLVANIA

275 Commerce Drive, Suite 300
Fort Washington, Pa. 19034
215-643-6450

TEXAS

6000 Denton Drive
P.O. Box 5012, M/S 366
Dallas, Texas 75222
214-238-6805

8600 Commerce Park Drive
Houston, Texas 77036
713-776-6511

VIRGINIA

Crystal Square 4
1745 Jefferson Davis Hwy., Suite 600
Arlington, Virginia 22202
703-979-9650

3930 Beulah Rd.
Richmond, Virginia 23234
804-275-8148

WASHINGTON

700 112th N.E., Suite 10
Bellevue, Washington 98004
206-455-3480

ARGENTINA

Texas Instruments Argentina S.A.I.C.F.

Km. 25, 5 Ruta Panamericana Don Torcuato
C.C. Box 2296 – Correo Central
Buenos Aires, Argentina
748-1141

ASIA

Texas Instruments Asia Ltd.

Aoyama Tower Bldg.
4, 5, & 6F
24-15 Minami Aoyama
2-Chome, Minato-Ku,
Tokyo, Japan 107
03-402-6171

902, Asian House
1, Hennessy Road
Hong Kong
5/279041

Room 507, Chia Hsin Bldg.
96 Chung Shan North Road, Sec. 2
Taipei, Taiwan

P.O. Box 2093
990 Bendemeer Rd.
Singapore 1, Republic of Singapore

AUSTRALIA

Texas Instruments Australia Ltd.

Unit 1A, 9 Byfield Street, P.O. Box 106
North Ryde, N.S.W. 2113
Sydney, Australia
887 1122

AUSTRIA

Texas Instruments Ges. M.B.H.

Rennweg 17
1030 Wien, Austria
724-186

BELGIUM

Texas Instruments Belgium SA

Avenue Edouard Lacombe 21
B-1040 Brussels, Belgium
733-9623

BRAZIL

Texas Instrumentos Electronicos
do Brasil Ltda.

Rua Padre Pereira Andrade, 591
05469 Sao Paulo, SP, Brasil
260-6347 & 260-5710

CANADA

Texas Instruments Incorporated

945 McCaffery Street
St. Laurent H4T1N3
Quebec, Canada
514-341-3232

280 Centre Str. East
Richmond Hill (Toronto)
Ontario, Canada
416-889-7373

DENMARK

Texas Instruments A/S

Marielundvej 46 D
DK-2730 Herlev, Denmark
917 400

FINLAND

Texas Instruments Finland OY

Freesenkatu 6
P.O. Box 917
00101 Helsinki 10, Finland
40 83 00

FRANCE

Texas Instruments France

La Boursidiere, Bloc A, R.N. 186
92350 Le Plessis Robinson, France
(1) 630 23 43

31, Quai Rambaud
69002 Lyon, France
(78) 37 35 85

9, Place de Bretange
35000 Rennes, France
(99) 79 54 81

L'Autan 100, Alle de Barcelone
31500 Toulouse, France
(61) 21 30 32

1, Av. de la Chartreuse
38240 Meylan, France
(76) 90 45 74

GERMANY

Texas Instruments Deutschland GmbH.

Haggertystrasse 1
8050 Freising, Germany
08161/80-1

Frankfurter Ring 243
8000 Munich 40, Germany
089/32 50 11-15

Lazarettstrasse 19
4300 Essen, Germany
0201/23 35 51

Akazienstrasse 22-26
6230 Frankfurt-Griesheim, Germany
0611/39 90 61

Riethorst 4
3000 Hannover 51, Germany
0511/64 80 21

Krefelderstrasse 11-15
7000 Stuttgart 50, Germany
0711/54 70 01

Kurfuerstendamm 146
1000 Berlin, Germany
030/89 27 063

ITALY

Texas Instruments Italia SpA

Via Della Giustizia 9
20125 Milan, Italy
02-688 31 41

Via L. Mancinella 65
00199 Roma, Italy
06-83 77 45

Via Montebello 27
10124 Torino, Italy
011-83 22 76

MEXICO

Texas Instruments de Mexico S.A.

Poniente 116 #489
Industrial Vallejo
Mexico City, 15, D.F., Mexico
567-92-00

NETHERLANDS

Texas Instruments Holland B.V.

Laan Van de Helende Meesters No. 421
Amstelveen, Holland
47 3391

NORWAY

Texas Instruments Norway A/S

Ryensvingen 15
Oslo 6, Norway
(02) 68 94 85

PORTUGAL

Texas Instruments Equipamento
Electronico LDA

Rua Eng. Frederico
Ulrich 2650
Moreira Da Maia, Portugal
948/1003

SPAIN

Texas Instruments Espana S.A.

Calle Mallorca 272-276
12 Barcelona 12, Spain
2 15 29 50

SWEDEN

Texas Instruments International Trade
Corporation (Sverigefilialen)

Fack
Norra Hannvagen 3
S – 100 54 Stockholm
Sweden
08-23 54 80

UNITED KINGDOM

Texas Instruments Limited

Manton Lane
Bedford, England
0234-67466

Printed in U.S.A.