TEXAS INSTRUMENTS

TM 990

TM 990/189 Microcomputer User's Guide

MICROPROCESSOR SERIES™

November 1979

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Concluded)

APPENDICES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Concluded)

LIST OF TABLES

SECTION 1

INTRODUCTION

1.1 GENERAL

The TM 990/189 is a self-contained, single-board microcomputer system. It is intended for use as a learning aid in the instruction of microcomputer fundamentals, machine and assembly language programming, and microcomputer interfacing. It also demonstrates TMS 9900 family applications and advantages. Figure 1-1 shows the principal TM 990/189 components. The system's features include:

- TMS 9980A (MP9529) microprocessor
- 1024 bytes of random access memory (RAM) expandable on board to 2048 bytes (each byte contains 8 bits of data)
- 4096 bytes of read only memory (ROM) expandable on board to 6144 bytes
- 2 MHz crystal controlled clock
- Audio cassette interface
- 16 bit programmable I/O port and interrupt monitor (TMS 9901)
- 45-key alphanumeric keyboard
- Ten-digit, seven-segment L.E.D. type alphanumeric display
- Visual and acoustic indicators
- Resident system monitor and assembler
- Single step instruction execution

In addition to onboard memory expansion, two other system expansion options are available:

- A TMS 9902 asynchronous communications controller, and accompanying interface circuits for either RS-232-C or 20 mA current loop terminals can be added.
- The bus can be expanded by use of the Bus Interface.

Figure 1-2 shows the system architecture along with the user options.

1.2 MANUAL ORGANIZATION

This manual is organized as follows:

- Section 1 covers board specifications and characteristics. A glossary in paragraph 1.4 explains terms used throughout the manual.
- Section 2 shows how to install, power up, and operate the TM 990/189 microcomputer.

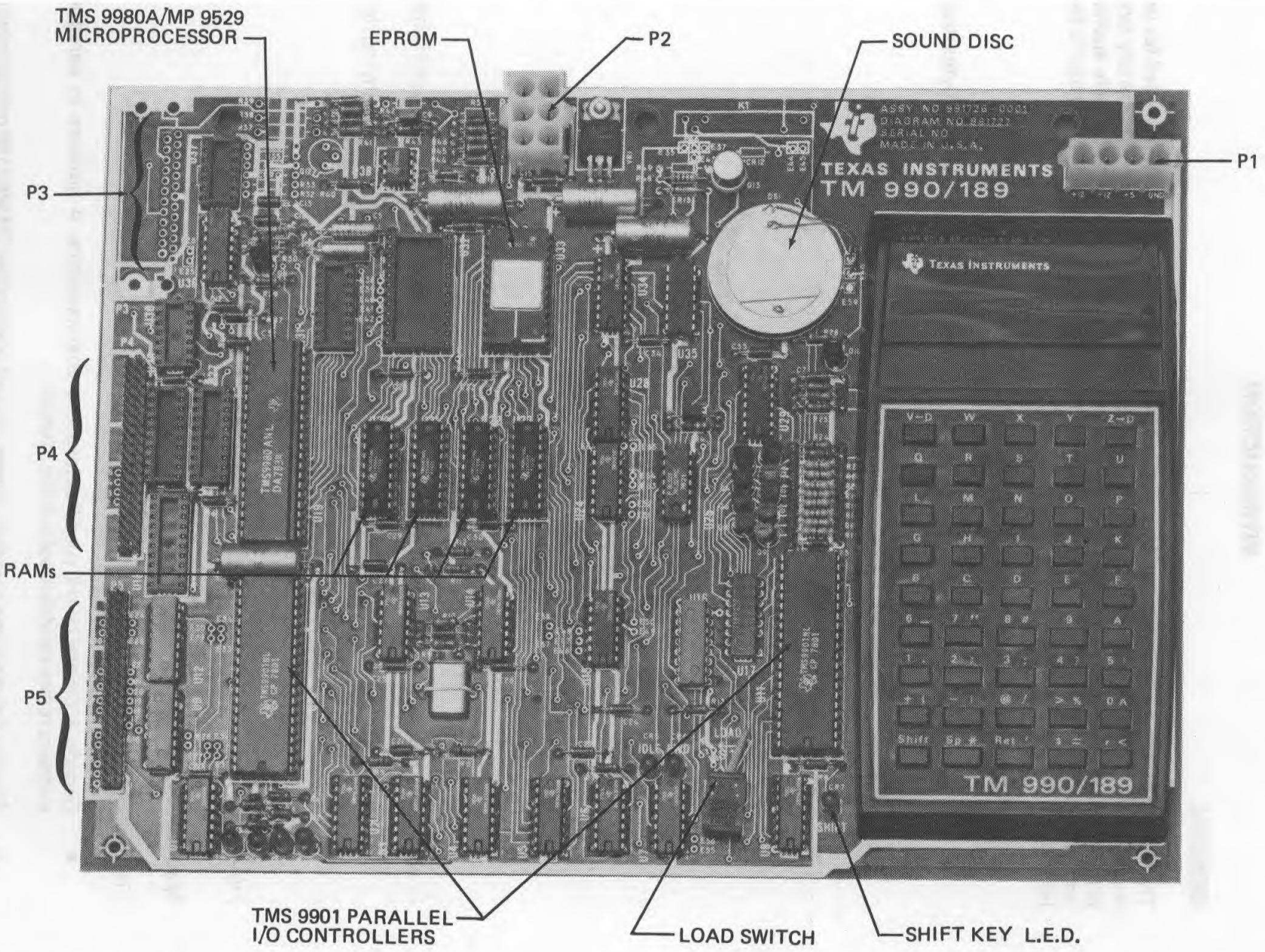


Figure 1-1. Principal TM 990/189 Components

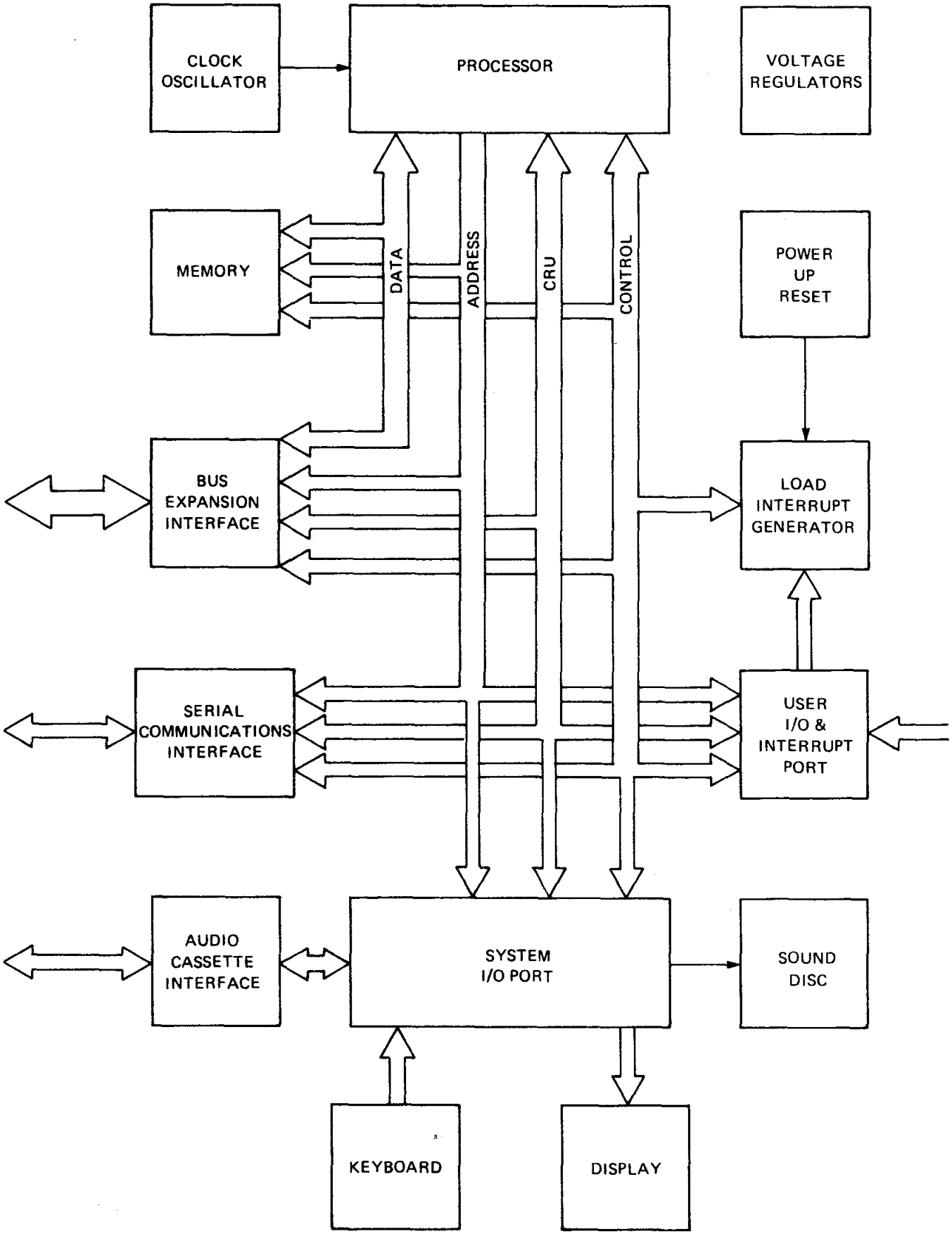


Figure 1-2. System Block Diagram

- Section 3 explains the UNIBUG monitor. UNIBUG commands, XOPs and error indicators are topics included.
- Section 4 is an introduction to the Symbolic Assembler used in the TM 990/189.
- Section 5 covers the instruction set used with the TM 990/189. Instruction formats and addressing modes are explained in detail.
- Section 6 covers assembler directives.
- Section 7 presents the fundamental concepts involved in I/O programming. The Communications Register Unit (CRU) and two interface adaptors (TMS 9901 and TMS 9902) are described along with their programming techniques.
- Section 8 covers theory of operation including memory and I/O mapping, block diagram, and circuit descriptions.
- Section 9 provides the necessary information to utilize the available options and modify the system.
- Section 10 provides a very useful troubleshooting checklist and debugging hints. Both static and dynamic checks are given.

1.3 GENERAL SPECIFICATIONS

Power consumption (Typical):

	+5 V	+12 V	−12 V
1024 bytes RAM, 4096 bytes ROM	595 mA	76 mA	5.6 mA
2048 bytes RAM, 5120 bytes ROM	701 mA	121 mA	36 mA
2048 bytes RAM, 6144 bytes ROM	696 mA	102 mA	16 mA

Clock rate: 2 MHz

Baud rates (set by UNIBUG monitor): 110 baud and 300 baud

Memory size:

RAM (TMS 4014), 1024 bytes expandable on-board to 2048 bytes. (Equivalent RAM is TMS 4045.)

ROM (TMS 4732), 4096 bytes expandable on-board to 6144 bytes.

Board dimensions: 8.15 by 11 inches (20.7 by 27.9 cm)

1.4 GLOSSARY

The following are definitions of terms used with the TM 990/189.

Absolute address: The actual memory address in quantity of bytes. Memory addressing is usually represented in hexadecimal from 0000_{16} to $3FFF_{16}$ for the TM 990/189.

Alphanumeric character: Letters, numbers, and associated symbols.

ASCII Code: A seven-bit code used to represent alphanumeric characters and control.

Assembler: Program that interprets assembly language source statement into object code.

Assembly Language: Mnemonics which can be interpreted by an assembler and translated into an object program.

Bit: The smallest part of a word; it has a value of either a 1 or 0.

Breakpoint: Memory address where a program is intentionally halted. This is a program debugging tool.

Byte: Eight bits or half a word.

Carry: A carry occurs when the most-significant bit is carried out in an arithmetic operation (i.e., resultant cannot be contained in only 16 bits).

Central Processing Unit (CPU): The "heart" of the computer: responsibilities include instruction access and interpretation, arithmetic functions, I/O memory access. The TMS 9980A (MP9529) is the CPU of the TM 990/189.

Command Scanner: A given set of instructions in the UNIBUG monitor which takes the user's input from the terminal and searches a table for the proper code to execute.

Context Switch: Change in program execution environment, includes new program counter (PC) value and new workspace area.

CRU (Communications Register Unit): The TMS 9980A's general purpose, command-driven input/output interface. The CRU provides up to 2048 directly addressable input and output bits.

Effective Address: Memory address resulting from interpretation of an instruction, required for execution of that instruction.

EPROM: See Read Only Memory.

Hexadecimal: Numerical notation in the base 16.

Immediate Addressing: An immediate or absolute value (16-bits) is part of the instruction (second word of instruction).

Indexed Addressing: The effective address is the sum of the contents of an index register and an absolute (or symbolic) address.

Indirect Addressing: The effective address is the contents of a register.

Interrupt: Context switch in which new workspace pointer (WP) and program counter (PC) values are obtained from one of 4 interrupt traps in memory addresses 0000_{16} to 0012_{16}.

I/O: The input/output lines are the signals which connect an external device to the data lines of the TMS 9980A.

Least Significant Bit (LSB): Bit having the smallest value in a byte or word (smallest power of base 2); represented by the right-most bit.

Link: The process by which two or more object code modules are combined into one, with cross-referenced label address locations being resolved.

Loader: Program that places one or more absolute or relocatable object programs into memory.

Machine Language: Binary code that can be interpreted by the CPU.

Monitor: A program that assists in the real-time aspects of program execution such as operator command interpretation and supervisor call execution. Sometimes called supervisor.

Most Significant Bit (MSB): Bit having the most value in a byte or word; the left-most bit representing the highest power of base 2. This bit is sometimes used to show sign with a 1 indicating negative and a 0 indicating positive.

Object Program: The hexadecimal interpretations of source code output by an assembler program. This is the code executed when loaded into memory.

One's Complement: Binary representation of a number in which the negative of the number is the complement or inverse of the positive number (all ones become zeroes, vice versa). The MSB is one for negative numbers and zero for positive. Two representations exist for zero: all ones or all zeroes.

Op Code: Binary operation code interpreted by the CPU to execute an instruction.

Overflow: An overflow occurs when the result of an arithmetic operation cannot be represented in two's complement (i.e., in 15 bits plus sign bit).

Parity: Means for checking validity of a series of bits, usually a byte. Odd parity means an odd number of one bits; even parity means an even number of one bits. A parity bit is set to make all bytes conform to the selected parity. If the parity is not as anticipated, an error flag can be set by software. The parity jump instruction can be used to determine parity.

Program Counter (PC): Hardware register that points to the next instruction to be executed or next word to be interpreted.

PROM: See Read Only Memory.

Random Access Memory (RAM): Memory that can be written to as well as read from (vs. ROM).

Read Only Memory (ROM): Memory that can only be read from (can't change contents). Some can be programmed (PROM) using a PROM Programmer. Some PROM's can be erased (EPROM's) by exposure to ultraviolet light.

Source Program: Programs written in mnemonics that can be translated into machine language (by an assembler).

Status Register (ST): Hardware register that reflects the outcome of a previous instruction and the current interrupt mask.

Supervisor: See Monitor.

Utilities: A unique set of instructions used by different parts of the program to perform the same function. In the case of UNIBUG, the utilities are the I/O XOP's.

Word: Sixteen bits or two bytes.

Workspace Register Area: Sixteen words, designated registers 0 to 15, located in RAM for use by the executing program.

Workspace Pointer (WP): Hardware register that contains the memory address of the beginning (register 0) of the workspace area.

1.5 APPLICABLE DOCUMENTS

The following is a list of documents that provide supplementary information for the TM 990/189 user.

- TMS 9901 Programmable Systems Interface Data Manual
- TMS 9902 Asynchronous Communication Controller (Data Manual)
- TMS 9980A/TMS 9981 Microprocessor Data Manual
- TMS 9900 Family System Development Manual

SECTION 2

INSTALLATION AND OPERATION

2.1 GENERAL

This section covers power supply requirements, power up procedure, operation (keyboard and display), and use of the audio cassette interface.

2.2 REQUIRED EQUIPMENT

- TM 990/189 University Board
- Suitable power supply such as TM 990/519

An equivalent power supply capable of meeting the requirements given in Table 2-1 may be used.

TABLE 2-1. POWER SUPPLY REQUIREMENTS

Voltage	Regulation	CURRENT
+5 V	+/−5%	1.787 A
+12 V	+/−5%	0.214 A
−12 V	+/−5%	0.155 A

2.3 POWER UP PROCEDURE

Figure 2-1 shows how to connect voltage to the TM 990/189 board. A cable is supplied which directly facilitates connection of the TM 990/189 to a TM 990/519 power supply. The connections on each end are positively keyed and prohibit misconnection of the power supply. Also, since the connectors are identical and this cable is wired "one for one," either end may be connected to the TM 990/189 or the TM 990/519.

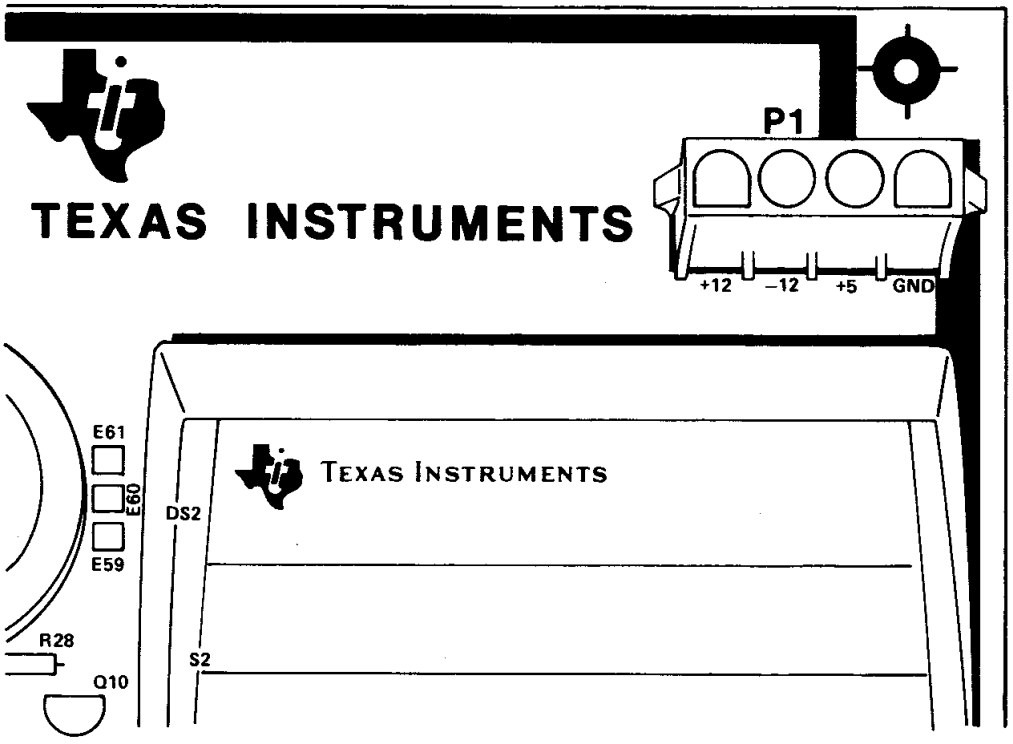


Figure 2-1. Power Supply Hook Up

When using a power supply other than the TM 990/519, the user should remove one connector from the cable and attach the proper connector or plugs for the power supply to be used. The power cable conductors are color coded as follows:

+5 V - Red
+12 V - White
−12 V - Green
Ground - Black

NOTE

It is advisable to check the connections from the TM 990/189 to the power supply output terminals with an ohmmeter before applying power.

CAUTIONS

1. Avoid applying incorrect voltage levels to the TM 990/189. Texas Instruments assumes no responsibility for damage caused by improper wiring or voltage application by the user.

2. Do not operate the TM 990/189 board on metal or other conductive surfaces without use of a protective insulator between the two.

2.4 OPERATION

1. Verify that all wiring has been correctly connected.

2. Apply power to board.

3. The TM 990/189 has a power up load feature. The system initialization routine performs a self-check operation on the major system components and indicates a successful self-check completion by displaying "CPU READY" in the display.

2.5 KEYBOARD

2.5.1 KEYBOARD DESCRIPTION

The keyboard consists of 45 keys. Figure 2-2 and Figure 2-3 show the unshifted and shifted key code designations respectively. Keys are shifted when the SHIFT key is depressed; while in this mode, the shift L.E.D. will illuminate. The keyboard becomes unshifted as soon as any key is pressed with the exception of DISPLAY RIGHT (→D) or DISPLAY LEFT (←D).

2.5.2 KEYBOARD USE

The keyboard is used to enter commands and data to the microprocessor. The TM 990/189 will energize in a power up LOAD state and the display will show CPU READY. At this point the microprocessor will wait for one character to be input. If the character is P, the TM 990/189 will be configured to receive data from a terminal (assuming that the User I/O and Interrupt Port modifications have been made as described in Section 4). If the RETURN (Ret) key is pressed, the microprocessor is configured to accept data from the keyboard. The display will show a question mark indicating that the command scanner is available to interpret keyboard inputs. A flowchart covering power up initialization is given in Figure 2-4.

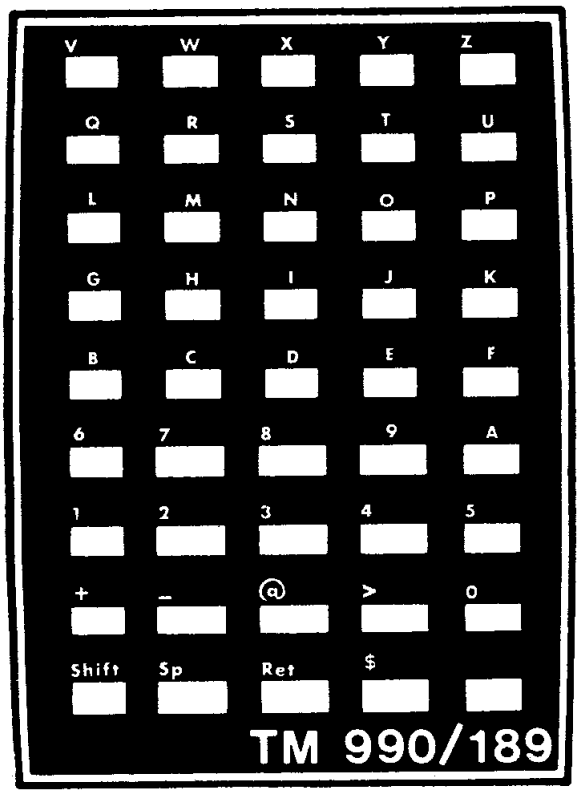


Figure 2-2. Unshifted Key Code Designation

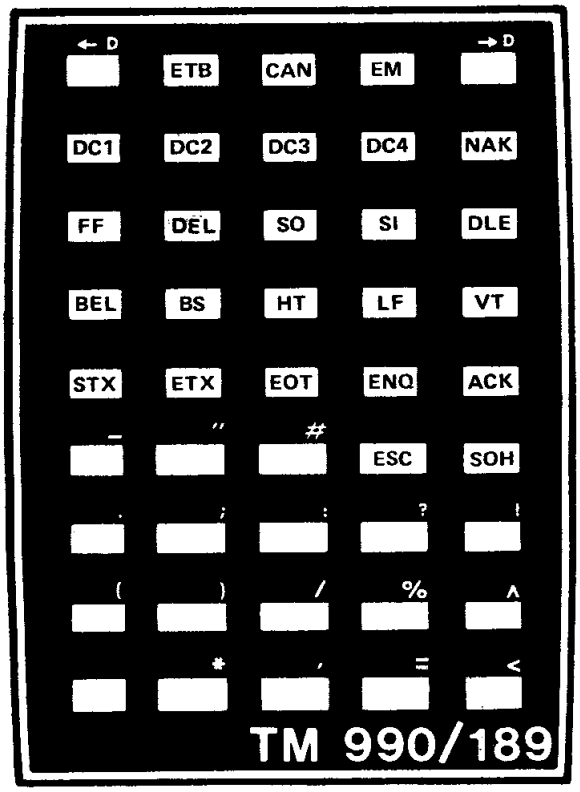


Figure 2-3. Shifted Key Code Designation

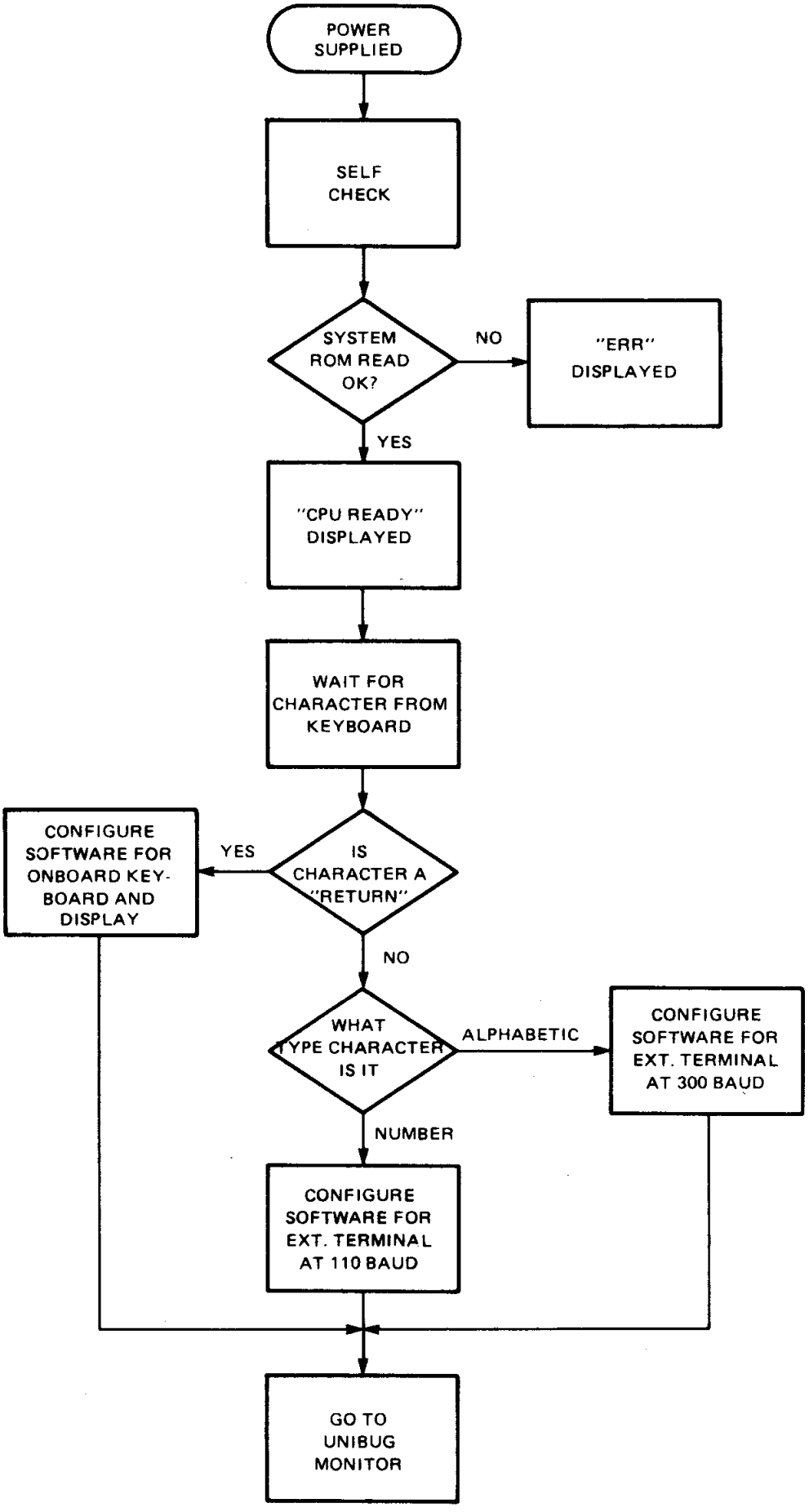


Fig. 2-4. Power Up Initialization Flow Chart

2.5.3 SPECIAL KEYS DESCRIPTION

2.5.3.1 Shift

The SHIFT key carries out the same function as that carried out by the shift key on a typewriter: it invokes a secondary definition for each key on the keyboard. The shift indicator L.E.D. (shown in Figure 1-1) will be activated to indicate that the shift mode has been entered. The shift mode may be exited by pressing any key except DISPLAY RIGHT or DISPLAY LEFT.

2.5.3.2 Display Left (←D)

The Display LEFT key shifts the field of view of the display to the left six character positions in the display buffer. This key does not change the contents of the display buffer. Figure 2-5 illustrates the operation of this key.

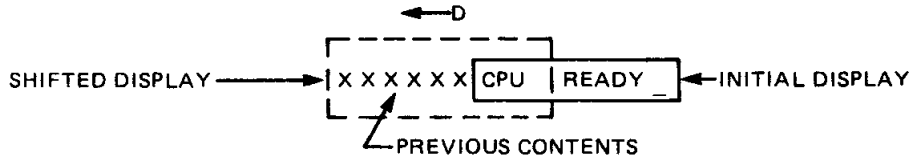


Figure 2-5. Display Left Operation

2.5.3.3 Display Right (→D)

The DISPLAY RIGHT key shifts the field of view of the display to the right six character positions in the display buffer. This key does not change the contents of the display buffer. Figure 2-6 illustrates the operation of this key.

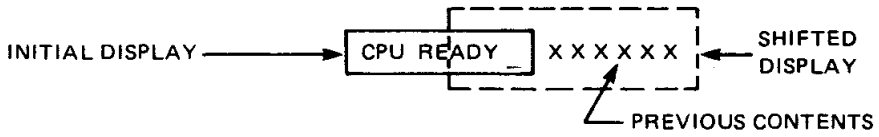


Figure 2-6. Display Right Operation

2.5.3.4 External Terminal Use.

The receipt of an alphabetic or numeric character when "CPU READY" is displayed will cause the TM 990/189 to be configured for communication with the optional TMS 9902 Serial Communications Controller and transfers control to the remainder of the system monitor (see Figure 2-4.)

EXAMPLE: Using the TM 990/189 with an External Terminal*

*See Section 8 for an explanation detailing the installation of the serial communications port and interface circuitry.

Procedure:

1. Plug in the cable from the terminal to connector P3 on the TM 990/189 (See Figure 2-7).

2. Connect the power supply cable from the TM 990/189 power pins to the TM 990/519 power supply.

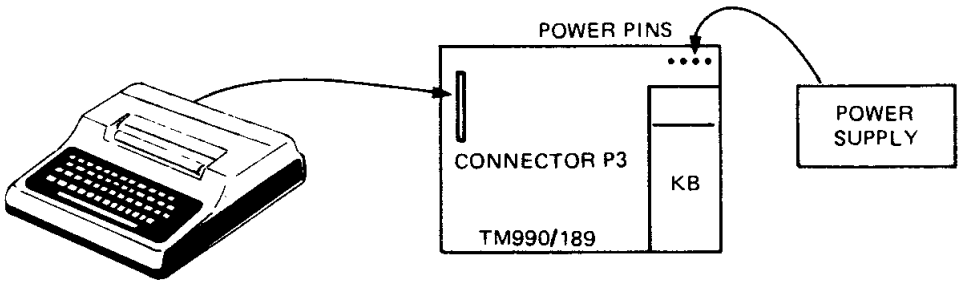


Figure 2-7. External Terminal Hook Up

3. Apply power to the system.

   - Turn terminal power on.
   - Place terminal ON LINE.
   - Turn power supply on.

4. The TM 990/189 will energize in a power up LOAD state and the display will show CPU READY.

5. For communications at 110 baud (10 characters per second) press a numeric key (0 through 9) on the TM 990/189. for communications at 300 baud (30 char/sec) press an alphabetic key (A through Z).

6. UNIBUG prints a "?" prompt. The terminal is now ready for use.

2.5.3.5 LOAD Switch.

The LOAD switch (S1) activates circuitry on the board which generates a non-maskable load interrupt to the processor. Activating the load switch causes the processor to discontinue execution of the current program and pass control to the UNIBUG monitor. Load switch activation does not alter the contents of user memory.

Since the processor cannot ignore the load stimulus, the load switch provides the user with ultimate control over processor actions. Should program control ever be lost (e.g., if the processor gets caught in an infinite loop) pressing LOAD forces control back to UNIBUG where the user can enter commands and direct system operation.

TABLE 2-2. DISPLAY CHARACTER FONT

ASCII CHARACTER	DISPLAY CHARACTER	SEGMENTS ILLUMINATED	ASCII CHARACTER	DISPLAY CHARACTER	SEGMENTS ILLUMINATED
A		a,b,c,e,f	6		a,c,d,e,f,g
B		c,d,e,f,g	7		a,b,c
C		a,d,e,f	8		a,b,c,d,e,f,g
D		b,c,d,e,g	9		a,b,c,d,f,g
E		a,d,e,f,g	0		a,b,c,d,e,f
F		a,e,f,g	SPACE		(none)
G		a,c,d,e,f	@		a,b,d,e,f,g
H		b,c,e,f,g	$		a,c,d,f,p
I		c	*		b,c,g,p
J		b,c,d,e	'		b
K		b,e,f,g	>		a,b
L		d,e,f	+		b,c,g
M		a,c,e,g	-		g
N		c,e,g	(		d,e,g
O		c,d,e,g	)		c,d,g
P		a,b,e,f,g	%		b,e,g,p
Q		a,b,c,f,g	/		b,e,g
R		e,g	=		d,g
S		a,c,d,f	^		a,b,f
T		d,e,f,g	<		a,f,g
U		a,c,d,e,f	,		d,p
V		b.d.f.g	.		p
W		a,c,d,e	;		c,d,p
X		a,d,g	:		c,p
Y		b,c,d,f,g	?		a,b,e,g,p
Z		a,b,d,e	!		b,c,p
1		b,c	_		d
2		a,b,d,e,g	"		b,f
3		a,b,c,d,g	#		b,c,d,g
4		b,c,f,g			
5		a,c,d,f,g			

2.6 L.E.D. DISPLAY

The display is a ten-digit, seven-segment L.E.D. type. It is used to display data, instructions, and error messages. The display segment designation is given in Figure 2-8. Table 2-2 shows display character font. The font gives the ASCII character, the display character, and the segments illuminated.

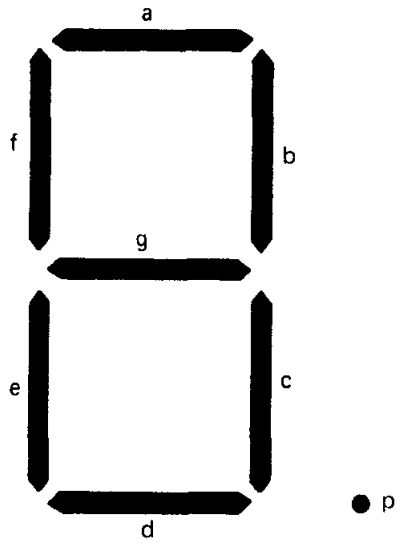


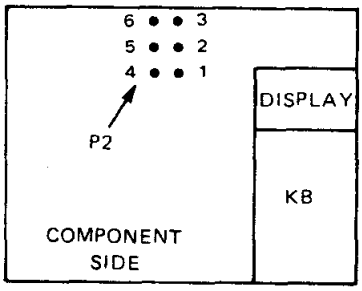
Figure 2-8. Display Segment Designation

2.7 USING AUDIO CASSETTE INTERFACE (ACI)

An audio cassette recorder can be used as a storage medium for programs used with the TM 990/189. The TM 990/189 provides audio cassette interface circuitry and optional user installed recorder control circuitry. The ACI theory of operation is covered in Section 8.

The TM 990/189 monitor provides two commands for use with a recorder. These commands are the L (load memory from cassette) and D (dump memory to cassette). Following entry of the L command, software will activate the motor control bit and begin looking for data from the audio cassette interface. Software will synchronize with the data, convert the data into proper digital format, and deposit the data in memory. After the data is loaded, the motor control bit is deactivated and the module identifier displayed. After entry of the D command and the parameters, the monitor will respond with READY and wait for a "Y" (yes) input from the keyboard. Upon receipt of the "Y" ready acknowledgement, the motor control bit is activated and an appropriate time allowed for the cassette deck to reach operating speed. Memory will be dumped from the start address to the stop address in 990 object record format. Following completion of the memory jump, the motor control bit will be deactivated. The syntax and description for these commands is given in Section 3.

Figure 2-9 shows the location of the pins for audio in, audio out, and motor control.



P2-1 MOTOR B (⏚)
P2-2 AUDIO OUTPUT GROUND (⏚)
P2-3 AUDIO INPUT GROUND (⏚)
P2-4 MOTOR A
P2-5 AUDIO OUTPUT
P2-6 AUDIO INPUT

Figure 2-9. ACI Connector Pins

2.7.1 RECORDER CONSIDERATIONS

In order to use the audio cassette efficiently several other aspects relevant to the recorder need further comment. These topics include:

- Cable hook up
- Volume and tone control positions
- Tape position
- Digital tape counter.

CAUTION

The TM 990/189 audio cassette interface is not compatible with certain models of audio cassette decks. Reliable operation is only warranted with the following audio cassette models:

General Electric 3-5121B
Panasonic RQ-413 AS
Realistic CTR-40
Realistic CTR-41
Sears Model No. 799-21683700
Sharp RD610.

A Panasonic Model RQ-413AS tape recorder will be used for illustrative purposes. Table 2-3 provides a listing of the connections that are needed between the TM 990/189 and the recorder.

TABLE 2-3. RECORDER AND ACI CONNECTIONS

Recorder	P2 Connections
Auxiliary Input	P2-5 (Audio Output) and P2-2 (Ground)
Earphone Output	P2-6 (Audio Input) and P2-3 (Ground)

The volume control should be set to approximately 80 percent of maximum and the tone control to mid-point.

The tape should be turned to the desired place on the tape where the recording is to be made and the initial value of the digital tape counter noted. After a "dump operation" is completed, the final reading of the digital tape counter should also be recorded. In this manner a number of programs may be stored and located on a single tape.

2.7.2 EXAMPLE OPERATION

This example will provide a step-by-step approach to using the audio cassette interface. It is assumed that the optional cassette control relay has not been installed.

NOTE

If the relay circuitry is installed, a cable will be required between the remote input on the recorder to P2-4 (Motor A) and P2-1 (Motor B). This will allow the processor to start and stop the recorder. It should be noted at this time that this cable would have to be unplugged from the remote input for rewind operations.

Procedure:

1. Initial Set Up

   a. Make the necessary connections between the recorder and P2 as given in Table 2-3. Do not make any connections to the remote input on the recorder.

   b. Apply power to the TM 990/189 and the recorder.

   c. Enter the example program given in Section 3.3 into memory. Examine memory to ensure that the program has been accurately entered.

   d. After inserting the cassette, position the tape to the desired starting place for the recording. If the desired starting place is the beginning of the tape, reset the digital tape counter to 000 prior to recording.

   e. Set the volume and tone controls to 80 percent and 50 percent respectively.

2. Dump Routine

   The D command will be used to dump memory from the "start address" to the "stop address." The "entry address" is the address in memory where it is desired to begin program execution. In this case the proper syntax (See Section 3.3.6) for the D command is as follows:

   D200 < T2 > 20 E < T2 > 200 < T2 > IDT = EXAMPLE < T1 > READY? < Y >

   After entering the D command, the monitor will respond with READY and wait for a Y (yes) keyboard entry indicating that the receiving device is ready. This allows the user to check recorder controls, tape position and the like before proceeding with the dump. If the recorder is set up correctly, press the RECORD button and then enter a Y on the keyboard to execute the D command. An appropriate time is allowed for the recorder to reach operating speed as indicated by a count down on the sound disc and LEDs CR0-CR3. In a similar fashion, the sound disc and the LEDs will signal when the dump has been completed.

   If the user wants to expedite the dump sequence, two asterisks can be used following the IDT tag. Example: IDT = **EXAMPL could be inserted in the syntax above to reduce dump time.

3. Load Routine

   a. In order to verify that the program can be properly loaded back into memory, change the contents of M.A. 0200 to 020E to zeros.

   b. Rewind the tape to the dump starting position as indicated by the digital tape counter — in this case reading 000.

   c. Enter the L command on the TM 990/189 keyboard. CR6 (FWD LED) will light indicating that the processor is now ready for the recorder to be turned on.

   d. Depress the PLAY button on the recorder. The SHIFT LED will go on and off while data transfer is occurring. LEDs CR0-CR3 and the sound disc will signal the end of the load routine.

   e. When the FWD indicator extinguishes, stop the recorder.

4. Check Out Routine

   a. The display on the TM 990/189 should display the name identifying the program. In this case, EXAMPLE should be displayed.

   b. Use the M command to inspect memory locations M.A. 0200 to M.A. 020E and verify that the contents are correct.

   c. The program counter has been loaded with the starting address of the program: the program is now ready to be executed.

SECTION 3

UNIBUG MONITOR

3.1 GENERAL

A monitor is a program that implements the UNIBUG commands and subroutines. A command specifes the operation that is to be performed (example: Memory Inspect/Change). A subroutine is a program that carries out a specific task (example: typewriter program configures the TM 990/189 so that the keyboard acts as a typewriter keyboard). The monitor used in the TM 990/189 provides sixteen commands and seven subroutines.

The commands and subroutines available in the TM 990/189 reside in a TMS 4732 ROM (board socket U 33). This section provides a description of the commands and subroutines available in the monitor.

3.2 UNIBUG COMMANDS

The 16 UNIBUG commands are given in Table 3-1. The following paragraphs will give the syntax and description for each command along with examples involving most of the commands. Conventions used to define command syntax are listed in Table 3-2.

TABLE 3-1. UNIBUG COMMANDS

Input	Results	Paragraph
A	Assembler Execute	3.3.3
B	Assembler Execute With Current Symbol Table	3.3.4
C	CRU Inspect/Change	3.3.5
D	Dump Memory to Cassette	3.3.6
E	Execute to Breakpoint	3.3.7
F	Status Register Inspect/Change	3.3.8
J	Jump to EPROM	3.3.9
L	Load Memory from Cassette	3.3.10
M	Memory Inspect/Change	3.3.2
P	Program Counter Inspect/Change	3.3.11
R	Workspace Register Inspect/Change	3.3.12
S	Single Step	3.3.13
T	"Typewriter" Program	3.3.1
W	Workspace Pointer Inspect/Change	3.3.14
Ret	New Line Request	3.3.15
$	Terminal Load/Dump Option	3.3.16

TABLE 3-2. COMMAND SYNTAX CONVENTIONS

Convention Symbol	Explanation
< >	Required items to be supplied by the user
[ ]	Optional items to be supplied by the user
(Ret)	Return
T1	Space
T2	Space or comma
T3	Space, comma, or return
LF	Line Feed
R or Rn	Register (n = 0 to 15)
WP	Current User Workspace Pointer contents
PC	Current User Program Counter contents
ST	Current User Status Register contents

NOTE

Except where indicated otherwise, no space is necessary between the parts of these commands. All numeric input is assumed to be hexadecimal; the last four digits input will be the value used. Thus a mistaken numerical input can be corrected by merely making the last four digits the correct value. If fewer than four digits are input, they are right justified with leading zeros.

Prior to discussing the aforementioned commands, an example program will be given. The example program serves as an aid to developing an understanding of command operations.

3.3 PROGRAM EXAMPLE

a. Problem:
Write a program that will add 33_{10} and 15_{10} and display the answer.

b. Program Solution:

```
LWPI    >0300    Load immediate to workspace pointer.

LI      0,33     Load R0 with first number (33₁₀)
LI      1,15     Load R1 with second number (15₁₀)

A       1,0      Add, answer in R0 (memory address 300₁₆)
XOP     0,10     Display contents of R0
XOP     1,13     Turn display on
```

c.

Program		Address	Hex Contents
LWPI	>300	0200	02E0
		0202	0300
LI	0,33	0204	0200
		0206	0021
LI	1,15	0208	0201
		020A	000F
A	1,0	020C	A001
XOP	0,10	020E	2E80
XOP	1,13	0210	2F41

d. To enter the previous program:

1. Apply power to the TM 990/189

2. The TM 990/189 will energize in a power up LOAD state and the display will show CPU READY.

DISPLAY	ENTER	COMMENTS
CPU READY_		
	(Ret)	
?_		UNIBUG commands can be entered now
	M	Memory Inspect/Change
?M_		
	200	M.A. 0200
?M 200_		
	(Ret)	
0200 = XXXX_		Current Contents M.A. 0200
	02E0	Enter New Contents
XXXX 02E0_		
	(Sp)	Advance to Next M.A.
0202 = XXXX_		Current Contents M.A. 0202
	0300	Enter New Contents
0202 0300_		
	(Sp)	
0204 = XXXX_		
	0200	
XXXX 0200		
	(Sp)	
0206 = XXXX_		
	0021	
XXXX 0021_		
	(Sp)	
0208 = XXXX_		
	0201	
XXXX 0201_		
	(Sp)	
020A = XXXX_		
	000F	
XXXX 000F_		
	(Sp)	
020C = XXXX_		
	A001	
XXXX A001_		
	(Sp)	
020E = XXXX		
	2E80	
XXXX 2E80		
	(Sp)	
0210 = XXXX		
	2F41	The entire program has been entered
XXXX 2F41		at this point

e. Now that the example program has been entered, it will be examined for errors prior to executing it. To examine the previous program:

DISPLAY	ENTER	COMMENTS
	(Ret)	
?_		
	M	Memory Inspect/Change
?M_		
	200	Program Starting M.A.
?M 200_		
	(Ret)	
0200=02E0_		Data is Correct (DIC)
	(Sp)	
0202=0300		DIC
	(Sp)	
0204=0200_		DIC
	(Sp)	
0206=0021_		DIC
	(Sp)	
0208=0201_		DIC
	(Sp)	
020A=000F_		DIC
	(Sp)	
020C=A001		DIC
	(Sp)	
020E=2E80		DIC
	(Sp)	
0210=2F41		DIC
		Note: If an error is found at any M.A., simply enter the correct data and proceed to the next M.A.

Now that the sample program has been presented and the proper method for entering and examining a program given,the method for executing the previous program will be given.

f. To execute the previous program:

1. Set the program counter using the P command to the starting address of the program (0200 in this case).

2. Use the E command without breakpoint to execute the program.

DISPLAY	ENTER	COMMENTS
	(Ret)	
?_		
	P	
P=XXXX		Current PC value
	0200	Set PC to 0200
XXXX 0200		
	(Ret)	
?_		
	E	
?E_		
	(Ret)	Execute program
0030_		The answer is displayed

3.3.1 TYPEWRITER PROGRAM (T)

Syntax

T

Description

Each time the T command is entered, a subroutine called the "Typewriter" program is called up. This program allows the user to use the keyboard as a typewriter and insert a string of characters in the display buffer. This program is quite useful in familiarizing the user with the characters produced at the display (see Table 2-1).

EXAMPLE:

DISPLAY	ENTER	COMMENTS
	(LOAD)	Initialize System
CPU READY_		
	(Ret)	Provides entry into command scanner
?_		
	T	Call "Typewriter" program
?T_		
	A	
?T A_		Character A displayed
	B	
?T AB_		Character B is added to display

3.3.2 MEMORY INSPECT/CHANGE (M)

Syntax

M [address] < T3 >

Description

Memory inspect/change "opens" the memory location specified, displays it, and gives the option of changing the data in the location. If a memory location is not specified, the default of 0000_{16} is used. The termination character causes the following:

- If a return, control is returned to the command scanner.
- If a space, the next memory location is opened and displayed.
- If a minus sign, the previous memory location is opened and displayed.

If a hexadecimal value is entered before the termination character, the displayed memory location is updated to the value entered. The last four digits input will be the value used; thus a mistaken numerical input can be corrected by merely making the last four digits the correct value. If fewer than four digits are input, they are right justified. The following example checks memory for the program loaded in paragraph 3.3(d).

Example:

DISPLAY	ENTER	COMMENTS
	(LOAD)	Initialize System
CPU READY_		
	(Ret)	Provides entry into Command Monitor
?_		
	M	Memory Inspect/Change Command
?M_		
	200	Memory address (M.A.) 0200 entered
?M 200_		
	(Ret)	
0200=02E0_		Contents of M.A. 0200
	(Sp)	
0202=0300_		
	(Sp)	
0204=0200_		Contents of M.A. 0204
	-	Minus entered
0202=0300_		Contents of M.A. 0202
	(Sp)	
0204=0200_		Contents of M.A. 0204
	1234	Enter new contents (M.A. 0204)
0200 1234_		
	(Sp)	
0206=0021_		Contents of M.A. 0206
	-	Minus entered
0204=1234		New contents in M.A. 0204

3.3.3 ASSEMBLER EXECUTE (A)

Syntax

A [address] < T3 >

Description

After entry of the "A" command, the monitor passes program control to the resident Symbolic Assembler. The assembler's symbol table is cleared. The Symbolic Assembler is covered in Section 4.

3.3.4 ASSEMBLER EXECUTE WITH CURRENT SYMBOL TABLE (B)

Syntax

B [address] < T3 >

Description

After entry of the "B" command, the monitor passes program control to the Symbolic Assembler. The previous symbol table is kept, allowing the user to resume assembly of a program after exiting the assembler. The Symbolic Assembler is covered in Section 4.

3.3.5 CRU INSPECT/CHANGE (C)

Syntax

C [CRU R12 address] < T2 > [No. of CRU bits] < T3 >

Definition

The CRU inspect/change monitor command is used to inspect/change the contents of the Communication Register Unit (CRU). The CRU inspect/change monitor command displays from 1 to 16 CRU bits, right justified. The command syntax includes the CRU base address and the number of CRU bits to be displayed. The CRU base address is the 16-bit contents of R12. The user must use 2 × CRU bit address desired to obtain data about CRU bits. As an example, assume that seven CRU bits beginning with CRU bit 80_{16} are requested. If either the CRU R12 address or the number of CRU bits is not specified, the default values of 0000_{16} and 16 bits are used respectively.

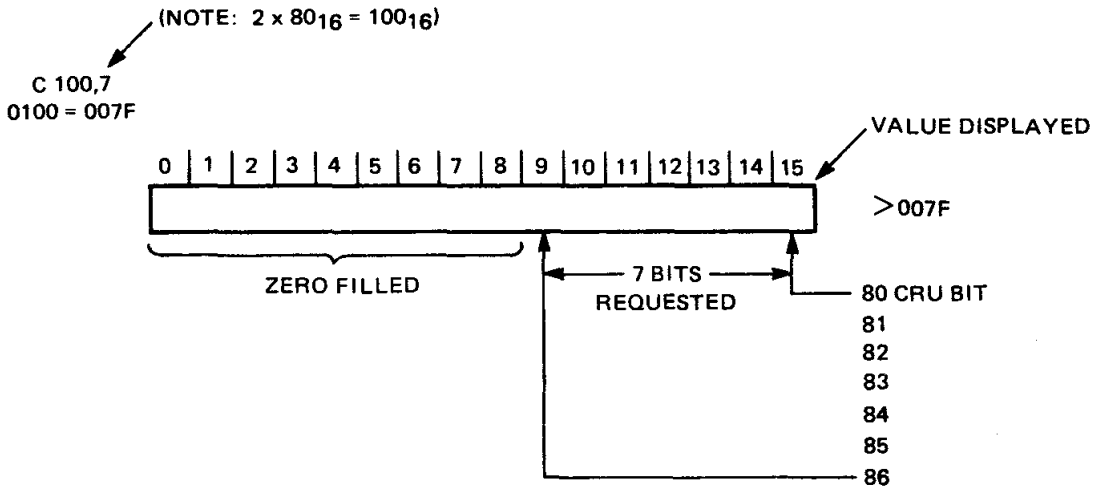


Figure 3-1. CRU Bits Inspected By C Command

Examples:

1. Examine eight CRU input bits. CRU base address is 20_{16}.

DISPLAY	ENTER	COMMENTS
	(LOAD)	
CPU READY_		
	(Ret)	
?_		
	C	
?C_		
	20, 8	
?C 20,8_		
	(Ret)	
0020=00F0		Eight CRU bits = F0

2. Set value of eight CRU output bits at CRU base address 20_{16}; new value is 02_{16}.

DISPLAY	ENTER	COMMENTS
?_		
	C	
?C_		
	20,8	
?C 20,8		
	(Ret)	
0020=00F0_		Previous value = F0
	2	Enter new value
20=00F0 2_		
	(Ret)	
?_		
	C	Check to see if new value entered
?C_		
	20,8	
?C 20,8_		
	(Ret)	
0020=0002_		New value = 02

3.3.6 DUMP MEMORY TO CASSETTE (D)

Syntax

MONITOR PROMPTS

D <start address> <T2> <stop address> <T2><entry addres > <T2> IDT = <name> <T1> READY <Y>

Description

The D command allows a program to be stored on Audio Cassette Tape. Memory is dumped from "start address" to "stop address." "Entry address" is the address in memory where it is desired to begin program execution. After entering a space or comma following the entry address, the monitor responds with an "IDT=" prompt asking for an input of up to eight characters that will identify the program. After entering the D command, the monitor will respond with "READY" and wait for a Y keyboard entry indicating the receiving device is ready. This allows the user to verify switch settings, etc., before proceeding with the dump. Upon receipt of the ready acknowledge (Y input), the motor control bit is activated, the FWD light is illuminated, and an appropriate time is allowed for the cassette deck to reach operating speed. Memory is dumped from the start address to the stop address in absolute 990 object record format. Following completion of the memory dump, the motor control bit is deactivated and the FWD light extinguished.

As an example, assume that the starting address of the program to be dumped is M.A. 0200 and the stop address is M.A. 0210: the entry address in this example will be the same as the starting address. The correct syntax would be as follows:

D 200 < T2 > 210 < T2> 200 < T2 > IDT = EXAMPLE < T1 > READY < Y >

The use of a double asterisk following the IDT tag will allow for a faster dump sequence. The correct syntax would be as follows:

D 200 < T2 > 210 < T2 > 200 < T2 > IDT = ** EXAMPL < T1 > READY < Y >

A detailed step-by-step example illustrating the use of this command is given in Section 2.7.

3.3.7 EXECUTE TO BREAKPOINT (E)

Syntax

E [breakpoint address] < T3 >

Description

The E command can be used for two purposes: it can be used to execute an entire program, and it can be used to execute a program up to a specified address (breakpoint address) where it is desired to halt the program. This feature allows longer programs to be broken into smaller sections for debugging.

Program execution begins at the values of PC, WP, and ST previously established. An optional breakpoint address can be specified; if given, execution continues to breakpoint address at which time control returns to the monitor and the contents of the PC will be displayed. If no breakpoint address is specified, then execution begins with no halt point specified.

To illustrate one use of this command the sample program given in paragraph 3.3 was previously entered into memory. To execute the sample program, the E command is used (without breakpoint).

Example:

DISPLAY	ENTER	COMMENTS
?_		
	P	
P=XXXX	0200	Starting address of program to be executed
XXXX 0200		
	(Ret)	PC set to 0200
?_	E	
?E_		
	(Ret)	Execute program
0030_		Solution

3.3.8 STATUS REGISTER INSPECT/CHANGE (F)

Syntax

F

Description

This command is used to inspect or change the contents of the Status Register.

Example:

DISPLAY	ENTER	COMMENTS
?_		
	F	
?F = XXXX_		Contents of Status Register
	44	Enter new contents
=XXXX 44_		
	(Ret)	
?_		
	F	Check status register contents
?F = 0044_		New contents

3.3.9 JUMP TO START OF EXPANSION EPROM (J)

Syntax

J [VALUE 1] < T2 > [VALUE 2] < T2 > [VALUE 3] < T3 >

Description

The J command causes a branch to memory address 0800_{16}, the beginning address of the onboard expansion EPROM at socket U32. The user program at that address will define its own workspace and return-to-program procedure (return vectors are *not* transferred as in a BLWP instruction). The command format allows the insertion of three parameters that will be stored in three contiguous memory locations beginning at address $>80_{16}$. The first parameter following the insertion of the J key will be stored in the first address, the second parameter in the next address and the third parameter in the last address. The parameters will be delimited by commas; if a parameter is not entered in a field, its corresponding memory address will be all zeroes.

Example:

DISPLAY	ENTER	COMMENTS
?_		
	J	
?J		
	4142,4344	Parameters to be stored in memory addresses
?J		0080_{16} and 0082_{16}
4142,4344		
	(Ret)	Program at 0800_{16} executes

3.3.10 LOAD MEMORY FROM CASSETTE (L)

Syntax

L

Description

The L command is used to load a program into memory from tape on an audio cassette recorder/player. After loading the cassette into the cassette recorder, position the tape at the start of the program that is to be read into memory. Following entry of the L command, the UNIBUG monitor activates the motor control bit, illuminates the FWD indicator, and begins looking for data from the audio cassette interface. The data will be synchronized, interpreted and deposited in memory. After the data is loaded, the motor control bit is deactivated, the FWD light extinguished, and the name of the program as recorded on the object program is displayed. A detailed step-by-step example illustrating the use of this command is given in Section 2.7.

3.3.11 PROGRAM COUNTER INSPECT/CHANGE (P)

Syntax

P [new PC] < (Ret) >

Description

This command is used to inspect or change the contents of the program counter. As an example, assume that a program starting at M.A. 0200 is to be executed. The program counter would first have to be set to the starting address of the program prior to executing it.

Example:

DISPLAY	ENTER	COMMENTS
?_		
	P	
?P=XXXX		PC is at this address
	200	Enter new address
=XXXX 200_		
	(Ret)	PC contains address 0200
?_		
	P	Check PC address
?P=0200		PC address = 0200
	(Ret)	Return to UNIBUG

3.3.12 WORKSPACE REGISTER INSPECT/CHANGE (R)

Syntax

R [Workspace Register number] < T3 >

Description

The R command is used to display the contents of any workspace register and allows the user to change the register contents. The workspace begins at the address given by the Workspace Pointer.

The R command followed by an optional register number in hexadecimal and a return causes the display of the specified register's contents. The user may then enter a new value into the register by entering a hexadecimal value. If a Workspace Register number is not specified, register zero is the default value. The following are termination characters whether or not a new value is entered:

- A space causes display of the next register (next memory word).
- A minus sign causes display of the previous register (previous memory word).
- A return returns control to the command scanner.

The workspace pointer (W) must be set to the user RAM before attempting to change register contents. M.A. 0200 will be used in the following example.

Example:

DISPLAY	ENTER	COMMENTS
?_	W	Inspect workspace pointer address
?W=XXXX		Workspace pointer address
	200	Input new contents
=XXXX 200		
	(Ret)	New WP address = 0200
?_		
	R	
?R_		
	0	
?R 0_		
	(Ret)	
R0=XXXX		Contents of R0
	222	Enter new contents
=XXXX 222_		
	(Ret)	
?_		Check to see new R0 contents
	R	
?R_		
	0	
?R 0_		
	(Ret)	
R0=0222_		R0 has new contents

3.3.13 SINGLE STEP(S)

Syntax

S < (Ret) >

Description

The S command is used to execute one instruction at a time in a program. Each time the S command is entered, a single instruction is executed starting at the address in the Program Counter. After the command is executed, the contents of the workspace pointer, status register on any other workspace register can be examined. The program given in paragraph 3.3 was entered prior to running this example.

Example:

DISPLAY	ENTER	COMMENTS
?_		
	P	Examine PC
?P = XXXX_		
	200	Enter starting address of program
=XXXX 200		
	(Ret)	
?_		
	S	Execute 1st step of program (further S key entries execute successive steps)
?S 0204_		
	(Ret)	
?_		
	W	Examine workspace pointer
?W = 0300_		Workspace pointer contents

3.3.14 WORKSPACE POINTER INSPECT/CHANGE (W)

Syntax

W < T3 >

Description

This command allows the user to inspect or change the workspace pointer which is the memory address of register zero.

Example:

DISPLAY	ENTER	COMMENTS
?_		
	W	
?W = XXXX_		Workspace pointer address
	44	Input new contents
=XXXX 44_		
	(Ret)	
?_		
	W	Check new address
?W = 0044_		New workspace pointer address

3.3.15 NEW LINE REQUEST (RET)

Syntax

(Ret)

Description

Receipt of a New Line Request command causes the monitor to respond with a carriage return, line feed, and new line request if a terminal is used. If the keyboard is used, the previous display will be blanked and a prompt (?_) displayed.

Example:

DISPLAY	ENTER
	(LOAD)
CPU READY_	
	(Ret)
?_	

3.3.16 TERMINAL LOAD/DUMP OPTION ($)

Syntax

$ (Ret)

Description

When using an external data terminal, entry of the $ command causes all subsequent load and dump memory operations to input from and output to the terminal instead of the audio cassette. Therefore programs could be stored on digital cassette or paper tape by use of the appropriate data terminal. When loading from the terminal, the monitor automatically sends the appropriate control characters (DC1 for playback on and DC3 for playback off) to the terminal to control playback/reader operation. Likewise when dumping to the terminal, the monitor sends control characters (DC2 for record on and DC4 for record off) to initiate record/punch operations. Users should examine the capabilities of their terminals to determine if they are equipped to respond to the playback and record control characters automatically. In cases where terminals are not so equipped, users will be required to initiate the appropriate playback and record operations manually based on the status of the FWD indicator as in the case of the audio cassette.

3.4 USER ACCESSIBLE UTILITIES

UNIBUG contains seven utility subroutines that perform I/O functions as listed in Table 3-3. These subroutines are called through the XOP (extended operation) assembly language instruction.

TABLE 3-3. USER ACCESSIBLE UTILITIES

XOP	Function	Paragraph
8	Write 1 Hexadecimal Character to Terminal	3.4.1
9	Read Hexadecimal Word from Terminal	3.4.2
10	Write 4 Hexadecimal Characters to Terminal	3.4.3
11	Echo Character	3.4.4
12	Write 1 Character to Terminal	3.4.5
13	Read 1 Character from Terminal	3.4.6
14	Write Message to Terminal	3.4.7

NOTES

1. All characters are in ASCII code.

2. Most of the XOP format examples herein use a register for the source address, however, all XOP's can also use a symbolic memory address or any of the addressing forms available for the XOP instruction.

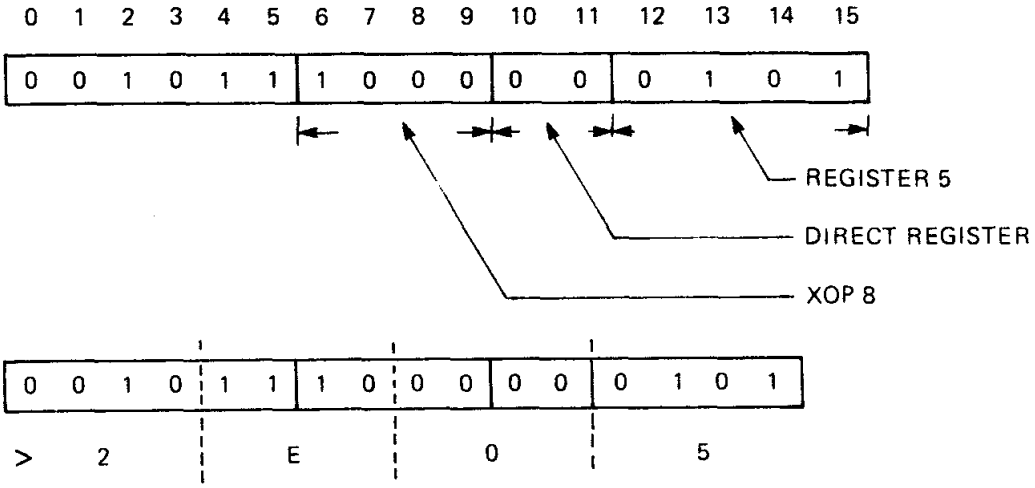
The XOP instruction uses Format 9. The machine language format is given below.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
General Format:	0	0	1	0	1	1	D (XOP NUMBER)				T_S		SR			

Bits 6-9 contain the desired XOP number. Bits 10, 11 indicate the source register address mode (Ts) and bits 12-15 indicate the source register number (SR). An example will best illustrate the use of this format.

Example:

Assume that XOP 8 is to be used to write one hexadecimal character from register five to the display.



XOPs 8, 10, and 12 can be used with the display provided that XOP 13 (needed to turn the display on) is added to the program following the required XOP. To illustrate the use of XOP 8 in conjunction with the display, the following example program will be used.

PROGRAM	ADDRESS	HEX CONTENTS
LWPI > 300	0200	02E0
	0202	0300
LI 0,3	0204	0200
	0206	0003
LI 1,2	0208	0201
	020A	0002
A 1,0	020C	A001
XOP 0,8	020E	2E00
XOP 1,13	0210	2F41

This program adds two numbers (3 and 2) and places their sum in register 0. XOP 8 is used to display the answer and XOP 13 is used to turn the display on. This XOP could be used to display answers in the range of 0 to F (hexadecimal).

3.4.1 WRITE ONE HEXADECIMAL CHARACTER TO TERMINAL (XOP 8)

Format: XOP Rn,8

The least significant four bits of user register Rn are converted to their ASCII coded hexadecimal equivalent (0 to F) and output on the terminal or display. Control returns to the instruction following the extended operation.

Example:

Assume user register 5 contains 203C16. The assembly language (A.L.) and machine language (M.L.) values are shown below.

A.L. XOP R5,8 SEND 4 LSB'S OF R5 TO TERMINAL

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
M.L.	0	0	1	0	1	1	1	0	0	0	0	0	0	1	0	1	>2E05

Terminal Output: C

3.4.2 READ HEXADECIMAL WORD FROM TERMINAL (XOP 9)

Format:	XOP	Rn,9	
	DATA	NULL	ADDRESS OF CONTINUED EXECUTION IF NULL IS ENTERED
	DATA	ERROR	ADDRESS OF CONTINUED EXECUTION IF NON-HEX NO. ENTERED
	(NEXT INSTRUCTION)		EXECUTION CONTINUED HERE IF VALID HEX NUMBER AND TERMINATOR ENTERED

Binary representation of the last four hexadecimal digits input from the terminal is accumulated in user register Rn. The termination character is returned in register Rn+1. Valid termination characters are space, minus, comma, and a carriage return. Return to the calling task is as follows:

- If a valid termination character is the only input, return is to the memory address contained in the next word following the XOP instruction (NULL above).
- If a non-hexadecimal character or an invalid termination character is input, control returns to the memory address contained in the second word following the XOP instruction (ERROR above).
- If a hexadecimal string followed by a valid termination character is input, control returns to the word following the DATA ERROR statement above.

Example:

```
A.L.    XOP     R6,9    READ HEXADECIMAL WORD INTO R6
        DATA    >3F80   RETURN ADDRESS, IF NO NUMBER
        DATA    >3F86   RETURN ADDRESS, IF ERROR
```

M.L. M.A.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
3F00	0	0	1	0	1	1	1	0	0	1	0	0	0	1	1	0	>2E46
3F02	0	0	1	1	1	1	1	1	1	0	0	0	0	0	0	0	>3F80
3F04	0	0	1	1	1	1	1	1	1	0	0	0	0	1	1	0	>3F86

If the valid hexadecimal character string 12C is input from the terminal followed by a carriage return, control returns to memory address (M.A.) $FFB6_{16}$ with register 6 containing $012C_{16}$ and register 7 containing $000D_{16}$.

If the hexadecimal character string 12C is input from the terminal followed by an ASCII plus (+) sign, control returns to location $FFC6_{16}$. Registers 6 and 7 are returned to the calling program without being altered. "+" is an invalid termination character.

If the only input from the terminal is a carriage return, register 6 is returned unaltered while register 7 contains $000D_{16}$. Control is returned to address $FFC0_{16}$.

3.4.3 WRITE FOUR HEXADECIMAL CHARACTERS TO TERMINAL (XOP 10)

Format: XOP Rn,10

The four-digit hexadecimal representation of the contents of user register Rn is output to the terminal or display. Control returns to the instruction following the XOP call.

Example:

Assume register 1 contains $2C46_{16}$.

```
A.L.    XOP     R1,10   WRITE HEX NUMBER
```

M.L.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
	0	0	1	0	1	1	1	0	1	0	0	0	0	0	0	1	>2E81

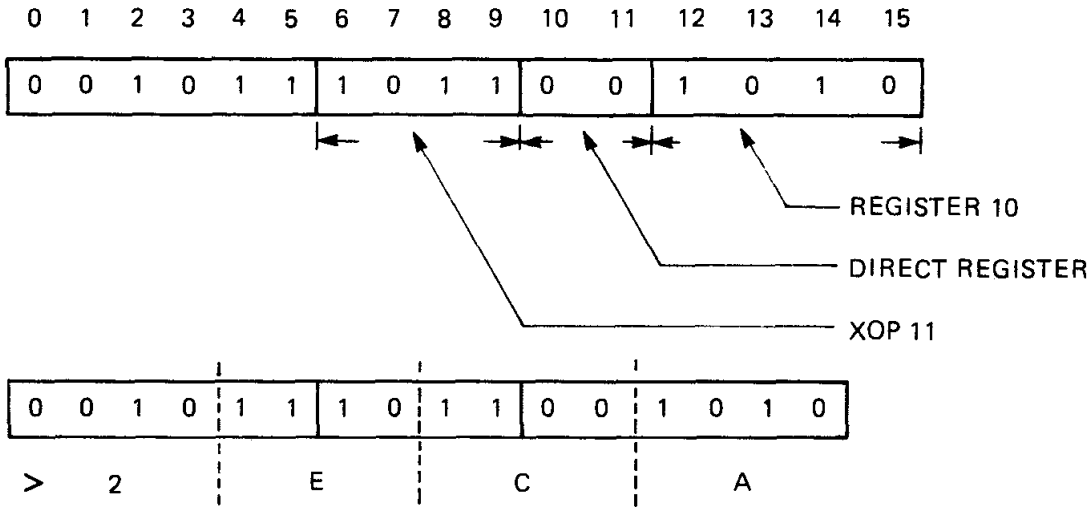
Terminal Output: 2C46

3.4.4 ECHO CHARACTER (XOP 11)

This is a combination of XOP's 13 (read character) and 12 (write character). A character in ASCII code is read from the keyboard placed in the left byte of Rn, then written (echoed back) to the display. Control returns to the instruction following the XOP after a character is read and written. By using a code to determine a character string termination, a series of characters can be echoed and stored at a particular address. The typewriter program uses this XOP.

Example:

The keyboard will generate a character that will be stored in R10, then written (echoed back) to the display.



The following program (Typewriter program) uses XOP 11. The JMP $-2 instruction allows the program to be continually repeated.

PROGRAM	ADDRESS	HEX CONTENTS
LWPI > 220	0200	02E0
	0202	0220
XOP R10,11	0204	2ECA
JMP $-2	0206	10FE

3.4.5 WRITE ONE CHARACTER TO TERMINAL (XOP 12)

Format: XOP Rn,12

The ASCII character in the left byte of user register Rn is output to the terminal. The right byte of Rn is ignored. Control is returned to the instruction following the call.

3.4.6 READ ONE CHARACTER FROM TERMINAL (XOP 13)

Format: XOP Rn,13

The ASCII representation of the character input from the terminal is placed in the left byte of user register Rn. The right byte of register Rn is zeroed. When this utility is called, control is returned to the instruction following the call only after a character is input.

3.4.7 WRITE MESSAGE TO TERMINAL (XOP 14)

Format: XOP @MESSAGE,14

3.5 UNIBUG ERROR MESSAGE

If an error is detected due to incorrect operation, the message "ERR" will be displayed. The following errors can be detected:

1. Invalid command entered.

2. Invalid termination character detected.

Error 1 is the result of entering an invalid command. As an example, assume that the N key was pressed instead of the M key. An error would be indicated by the display and further entry of the data inhibited. To clear the error, press Ret.

Error 2 is the result of entering an invalid termination character. In the event of error 2, the command is terminated. Clear the error using the Ret key then reissue the command and parameters with a valid termination character.

SECTION 4

SYMBOLIC ASSEMBLER

4.1 GENERAL

An assembler is a program that interprets assembly language source statements into object code. The TM 990/189 Symbolic Assembler assembles the 69 instructions of the TMS 9980 as well as:

- The pseudo instruction NOP which assembles as the instruction JMP $+2 (i.e., acts as a "no operation" or "go to next instruction")
- The following assembler directives (Directives used with the assembler are explained in Section 6):
  - — AORG: Absolute origin of statement (absolute start location)
  - — BSS: Block of memory reserved with starting symbol
  - — DATA: Sixteen-bits of immediate value
  - — END: End of program, exit to monitor, load program counter
  - — EQU: Symbol equated to value in operand
  - — TEXT: String of ASCII coded characters

Comments and labels (two characters maximum) can be used with this assembler. The assembler program is contained in a TMS 4732 ROM located at board socket U33.

Two UNIBUG commands can be used to call the assembler (A and B). If the A command is used, the previous symbol table will be cleared. If the B command is used, the previous symbol table is kept.

4.2 LABELS AND COMMENTS

4.2.1 LABELS

Labels may consist of one or two characters. The first character must be alphabetic and a second character must be alphanumeric. Labels may be used either as resolved (defined in label field of previous instruction) or unresolved (to be defined in label field of upcoming instructions) references. Labels may be used in the label field only once in a program.

4.2.2 COMMENTS

Comments can be a part of the source statement. The comment field may include any printable character and is concluded by a return.

4.2.3 USE DOLLAR SIGN TO INDICATE "AT THIS LOCATION"

Use the dollar ($) sign to indicate a current value of the location counter (the location counter contains the next address at which object will be loaded). If the location counter contains a value of 0200_{16}, then the following comments apply as shown in these statements:

```
D1    EQU   $          D1 VALUE = LOCATION COUNTER VALUE = HEX 200
E1    EQU   $+4        E1 VALUE = LOC COUNTER + 4 = 204
F1    EQU   D1         F1 AND D1 HAVE SAME VALUE = HEX 200
      LI    R7,$       HEX 200 TO R7
      LI    R8,D1      HEX 200 TO R8
      LI    R9,$+2     HEX 20A TO R9 (LOC COUNTER NOW AT HEX 208)
      LI    R10,E1     HEX 204 TO R10
```

NOTE

In EQU (equate) directives, labels must be equated to either absolute values or defined labels.

4.2.4 EXPRESSIONS

Expressions can be used containing addition or subtraction functions. For example, if the location counter (contains the next memory address at which object will be loaded) contains 0200_{16}, then the following comments apply as shown in these statements:

```
A1    EQU   $          A1 VALUE = LOCATION COUNTER VALUE = HEX 200
B1    EQU   $+8        B1 = LOCATION COUNTER + 8 = HEX 208
C1    EQU   A1         C1 = A1 VALUE = HEX 200
      LI    R0,A1      HEX 200 TO R0
      LI    R1,A1+4    HEX 204 TO R1
      LI    R2,A1+C1   HEX 400 TO R2
      LI    R3,A1+B1+C1 HEX 608 TO R3
      LI    R4,A1-B1   HEX FFF8 TO R4
```

4.2.5 CANCEL SOURCE STATEMENT BEING INPUT

If it is desired to cancel a source statement in the middle of entering it from the keyboard, press the SHIFT key followed by the letter X key. The current location counter contents will be displayed, waiting for new input. This must be executed prior to entering a return after the source statement. The SHIFT X key is the ASCII cancel function. When using an external terminal, the CANCEL function is generated by holding the CONTROL key while pressing X. ASCII coding is explained in Appendix C.

4.2.6 TRANSLATE CHARACTERS INTO ASCII CODE USING SINGLE QUOTES

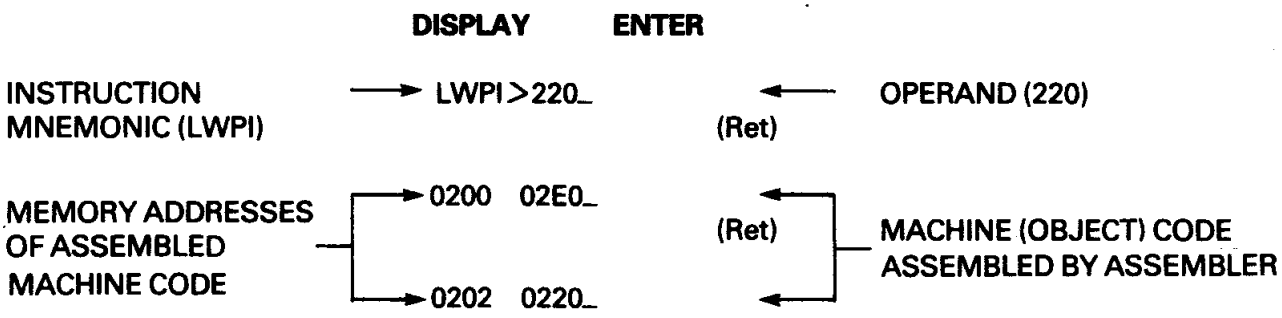
If it is desired to translate alphabetical or numerical keyboard values in ASCII code, enclose the characters in single quotes. This is the normal procedure for the TEXT assembler directive (Section 6); however, it can also apply in other situations. For example:

```
A     EQU   'AB'       ASCII FOR AB = HEX 4142
      LI    R1,A       LOAD HEX 4142 IN R1
      LI    R1,'AB'    LOAD HEX 4142 IN R1
A1    DATA  'AB'       ASSEMBLE HEX 4142 HERE
```

4.3 ASSEMBLER ACTION

The Symbolic Assembler accepts assembly language inputs from the keyboard. As each instruction is input, the assembler interprets it, places the resulting machine code in an absolute address, and prints the machine code (in hexadecimal) next to its absolute address.

Example:



4.4 OPERATION

4.4.1 CALLING THE ASSEMBLER

- Call up the monitor by pressing the LOAD switch and the Ret key.
- Press either the A or B key. (If the A command is used, the previous symbol table will be cleared).
- Enter the hexadecimal address at which the program is to be assembled.
- Press Ret and entry to the assembler will occur.

Example:

DISPLAY	ENTER	COMMENTS
	(LOAD)	
CPU READY_		
	(Ret)	Monitor Entry Gained
?_		
	A	Assembler Call
?A_		
	0200	Starting Assembly Address
?A 0200_		
	(Ret)	
0200_		Assembler Entry Gained

4.4.2 EXITING TO THE MONITOR

Enter the END directive and two returns to cause an exit from the assembler and return control to the monitor.

4.5 ENTERING INSTRUCTIONS

Any of the 69 instructions applicable to the TM 990/189 microcomputer can be interpreted by the Symbolic Assembler. An instruction generally consists of four fields: Label, Opcode, Operand(s), and Comment. The Label field is optional and its omission is indicated by a space. The Label field may be followed by one or more spaces. There should be a single space between the mnemonic and the operand(s). In the case of multiple operands, a single comma should be used between the two. The operand(s) can be followed by a Comment field. The operand should be followed by a space if

the Comment field is desired or a return, if not. The Comment field may include any printable character and is concluded by a return. Up to the return ending either the Operand field or the Comment field, the instruction may be cancelled by use of the Cancel command explained in paragraph 4.2.5.

Examples:

1. LWPI >220
   Single space between mnemonic and operand

2. LI 0,33
   Single comma between multiple operands

3. N1 DATA 10
   Space after label and opcode fields

4.

DISPLAY	ENTER	COMMENTS
0200		
	(Sp)	Omit label field
00		
	LWPI >220	Enter instruction
LWPI >220		

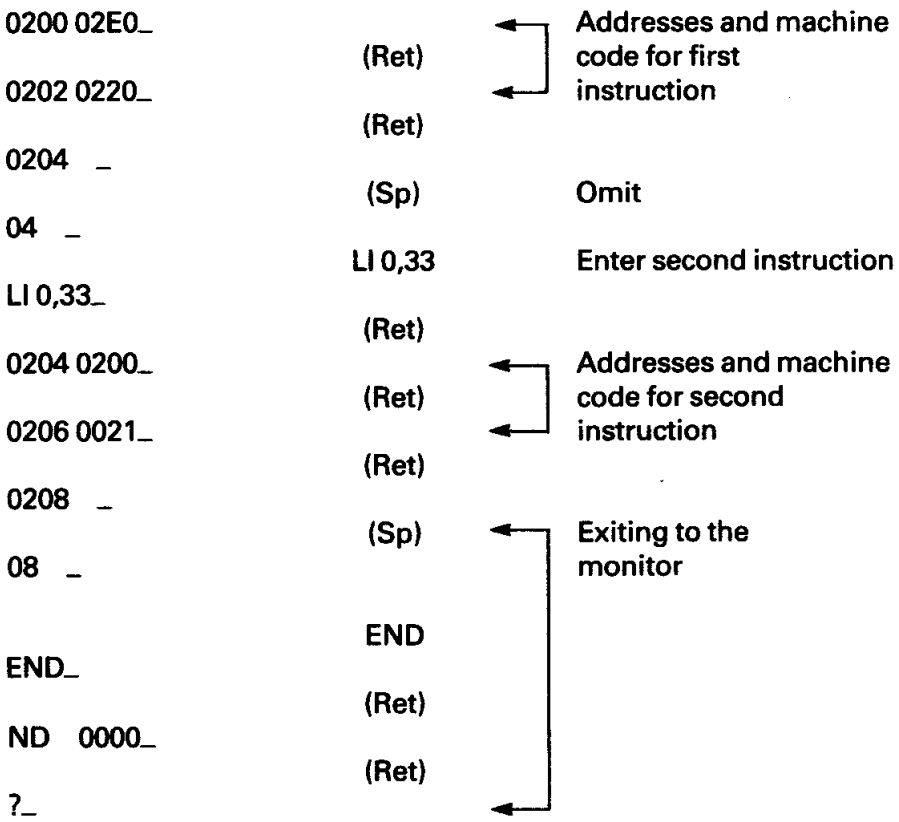
5.

INSTRUCTION	TERMINATOR
LWPI >200	A. Return (ret) - comment field omitted B. Space - comment field to be used

6. The following example illustrates these functions:

   A. Calling the assembler (paragraph 4.4.1)
   B. Enter instruction one (paragraph 4.5)
   C. Enter instruction two (paragraph 4.5)
   D. Exiting to the monitor (paragraph 4.4.2)

DISPLAY	ENTER	COMMENTS
	(LOAD)	
CPU READY_		
	(Ret)	Monitor entry gained
?_		
	A	Assembler call
?A_		
	0200	Starting assembly address
?A 0200_		
	(Ret)	
0200 _		Assembler entry gained
	(Sp)	Omit label field
0 0 _		
	LWPI >220	Enter first instruction
LWPI >220_		
	(Ret)	



The following additional concepts apply to instruction entry:

1. Register numbers are in decimal or hexadecimal. Only decimal register numbers can be predefined (preceded by an R).

   ```
   LI R13, 33
   LI >D, 33
   ```

2. Jump instruction operand can be $, $+n, $−n, or M where n is a decimal or hexadecimal value of bytes (+256 ≥ n ≥ −254) and M is a memory address in decimal or hexadecimal.

   ```
   JMP  $+0
   JMP  $−2
   JMP  $+2
   JMP  >210
   ```

3. Absolute numerical values can be decimal or hexadecimal. Decimal values have no prefix in an operand. Hexadecimal values are preceded by the greater-than sign (>).

   ```
   LI   R13, > 33
   LI   R13, 51
   ```

4. Where an address can be either a register or symbolic memory location, the symbolic address is preceded by an at sign (@) to differentiate a numerical memory address from a register number. These alphanumeric labels must be preceded by an @ sign; numerical values preceded by an @ sign will be assembled as an absolute address.

```
MOV     @ST, R1          Move ST Contents to R1
A       @SM, @ > FE00    Move SM Contents to M.A. > FE00
```

NOTE

Jump and immediate operand instructions do not use the at sign before a symbol.

4.6 ERRORS

Syntax errors are indicated by an 'ERR' message. A displacement range error (such as with jump instructions and single-bit CRU instructions) will be flagged with an RERR message.

1. Syntax error. The instruction syntax was incorrect:

DISPLAY	ENTER	COMMENTS
0200		
	(Sp)	
00		
	LDA	
LDAERR_		Error message (ERR)
	(Ret)	Use (Ret) and enter proper mnemonic

2. Range error. The operand is out of range for its field.

DISPLAY	ENTER	COMMENTS
0200		
	(Sp)	
00		
	LI R44	
LI R44_		
	(Ret)	
LI R44ERR_		Error message (ERR)
	(Ret)	Cancel
0200		
	(Sp)	
00		
	LI R4, 200	Enter proper data
LI R4,200_		
	(Ret)	
0200 0204_		
	(Ret)	Properly assembled code
0202 00C8		

3. Displacement Error. The jump instruction destination is more than +256 or −254 bytes away.

DISPLAY	ENTER	COMMENTS
0200		
	(Sp)	
00		
	JNC $+300	
JNC $+300_		
	(Ret)	
$+300RERR		Error message (RERR)
	(Ret)	Cancel and enter proper jump displacement

4.7 PSEUDO-INSTRUCTION

The assembler also interprets one pseudo-instruction. This pseudo-instruction is not an additional instruction but actually is an additional mnemonic that conveniently represents a member of the instruction set. The NOP mnemonic can be used in place of a JMP $+2 instruction which is essentially a no-op (no operation). This can be used to replace an existing instruction in memory, or it can be included in code to force additional execution time in a routine. Both NOP and JMP $+2 assemble to the machine code 1000_{16}.

DISPLAY	ENTER	COMMENTS
0200		
	(Sp)	
00		
	JMP $+2	
JMP $+2_		
	(Ret)	
0200 1000_		
0200		Both JMP $+2 and NOP assemble to the machine code 1000_{16}
	(Sp)	
00		
	NOP	
NOP_		
	(Ret)	
0200 1000		

4.8 TM 990/189 SYMBOLIC ASSEMBLER LISTING

4.8.1 LISTING FORMAT

Figure 4-1 identifies the different fields of the listing.

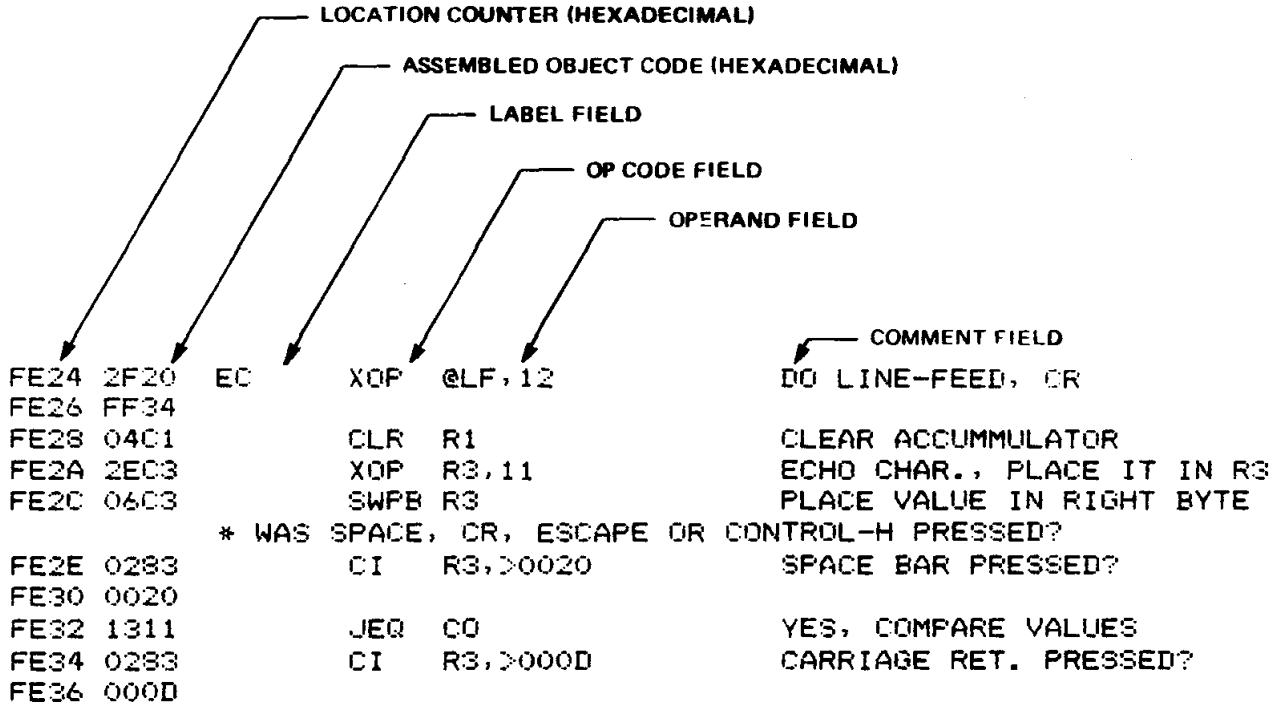


Figure 4-1. Listing and Source Statement Fields

4.8.1.1 Location Counter

This is the hexadecimal number showing the location of assembled object code. This location is relative to the beginning of the program; thus it should begin with location 0000_{16}. One exception is where an absolute origin assembler directive (AORG) is used.

Essentially, the location counter number is the location in memory of the corresponding object code after a program has been loaded into memory. For example, the object code in Figure 4-1 at M.A. $FE24_{16}$ is $2F20_{16}$, at M.A. $FE26_{16}$ it is $FF34_{16}$, etc.

4.8.1.2 Assembled Object Code

This column contains the resulting object code in hexadecimal after the source statement has been assembled.

4.8.1.3 Label Field

This two-character field contains an alphanumeric label that identifies the location of the source statement.

4.8.1.4 Op Code Field

This four-character field contains assembly language operation code mnemonics. It is separated from the label field and operand field by at least one space.

4.8.1.5 Operand Field

This field contains the operands of the instruction. This field is separated from the op code and comment fields by at least one space.

4.8.1.6 Comment Field

The comments in this field are abbreviated auxiliary data to help further understand the instruction or the data flow.

SECTION 5

INSTRUCTION SET FOR THE TM 990/189

5.1 GENERAL

This section covers the instruction set used with the TM 990/189 including assembly language and machine language. This instruction set is compatible with other members of the 990 family.

Other topics include:

- User Memory (paragraph 5.2)
- Hardware and software registers (paragraphs 5.3 and 5.4)
- Instruction forms and addressing modes (paragraph 5.5)

The TM 990/189 microcomputer is designed for use by a variety of users with varying technical backgrounds and available support equipment. Because a TM 990/189 user has the capability of writing his programs in machine language and entering them into memory using the UNIBUG monitor, emphasis is on binary/hexadecimal representations of assembly language statements. The assembly langauge described herein can be assembled on a 990 family assembler such as the TM 990/189 Symbolic Assembler explained in Section 4. If an assembler is used, this section assumes that the user will be aware of all prerequisites for using the particular assembler including assembler directives.

It is also presumed that all users learning this instruction set have a working knowledge in:

- ASCII coded character set (described in Appendix C).
- Decimal/hexadecimal, binary number system (described in Appendix D).

5.2 USER MEMORY

The memory provided in the TM 990/189 microcomputer consists of RAM (read/write memory) and ROM (read only memory). The RAM is used for user programs while the ROM contains the monitor and assembler programs. The monitor program provides keyboard commands, I/O programs, and other user utilities. The memory address (M.A.) value is the number of bytes beginning at 0000_{16}; thus all word addresses are even values from 0000_{16} to $3FFE_{16}$.

Figure 5-1 shows the memory map for the TM 990/189. Interrupt and XOP vectors extend from M.A. 0000_{16} to M.A. $007F_{16}$. UNIBUG Monitor workspaces extend from M.A. 0080_{16} to M.A. 0147_{16}. If the assembler is used, the symbol table begins at M.A. 0146_{16}. Four bytes are used for each label; the number of labels that are used will determine the beginning address for user RAM (Y). As an example, if 50 labels are used, 200 bytes will be needed for the label table. The end of the label table will be $o146_{16} + C8_{16} = 20E_{16}$ (note that $200_{10} = C8_{16}$). Therefore the start of the permissable user RAM would be M.A. 210_{16} in this case.

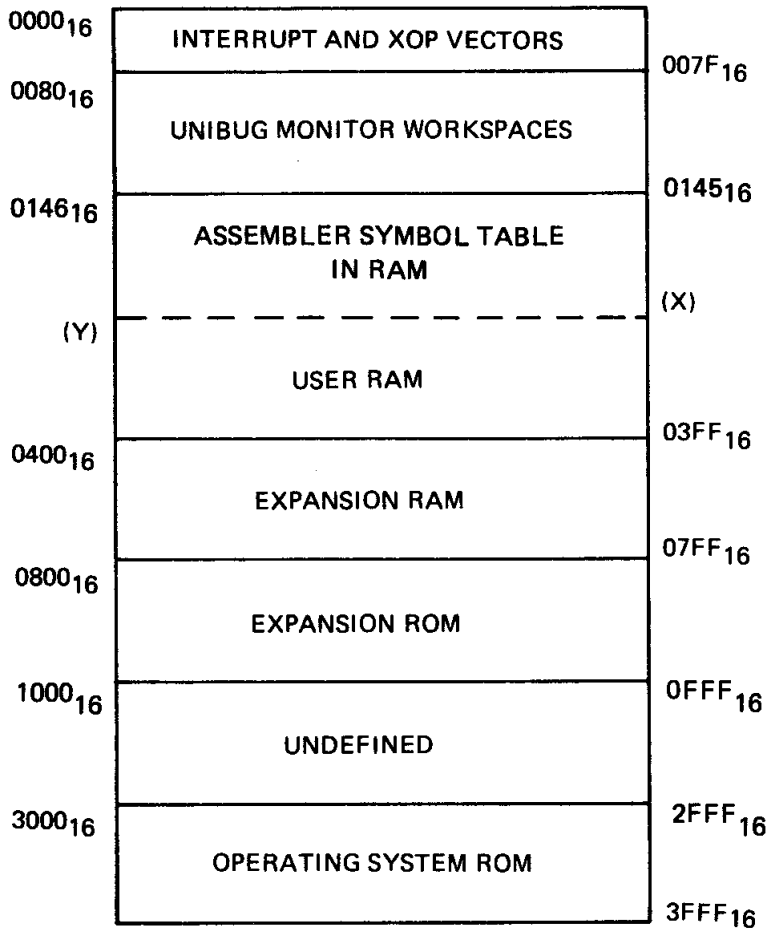


Figure 5-1. Memory Map

5.3 HARDWARE REGISTERS

Figure 5-2 shows the architecture of the TMS 9980A (MP 9529) microprocessor with affiliated RAM and ROM memory. The TM 990/189 uses three major hardware registers in executing the instruction set: Program Counter (PC), Workspace Pointer (WP), and Status Register (ST). These registers reside in the microprocessor and are controlled by the programmer.

5.3.1 PROGRAM COUNTER (PC)

This register contains the memory address of the next instruction to be executed. After an instruction image is read in for interpretation by the processor, the PC is incremented by two so that it "points" to the next sequential 16-bit memory word.

5.3.2 WORKSPACE POINTER (WP)

This register contains the memory address of the beginning of the register file currently being used by the program under execution. This workspace consists of 16 contiguous memory words designated registers 0 to 15. The WP points to register 0. Paragraph 5.4 explains a workspace in detail.

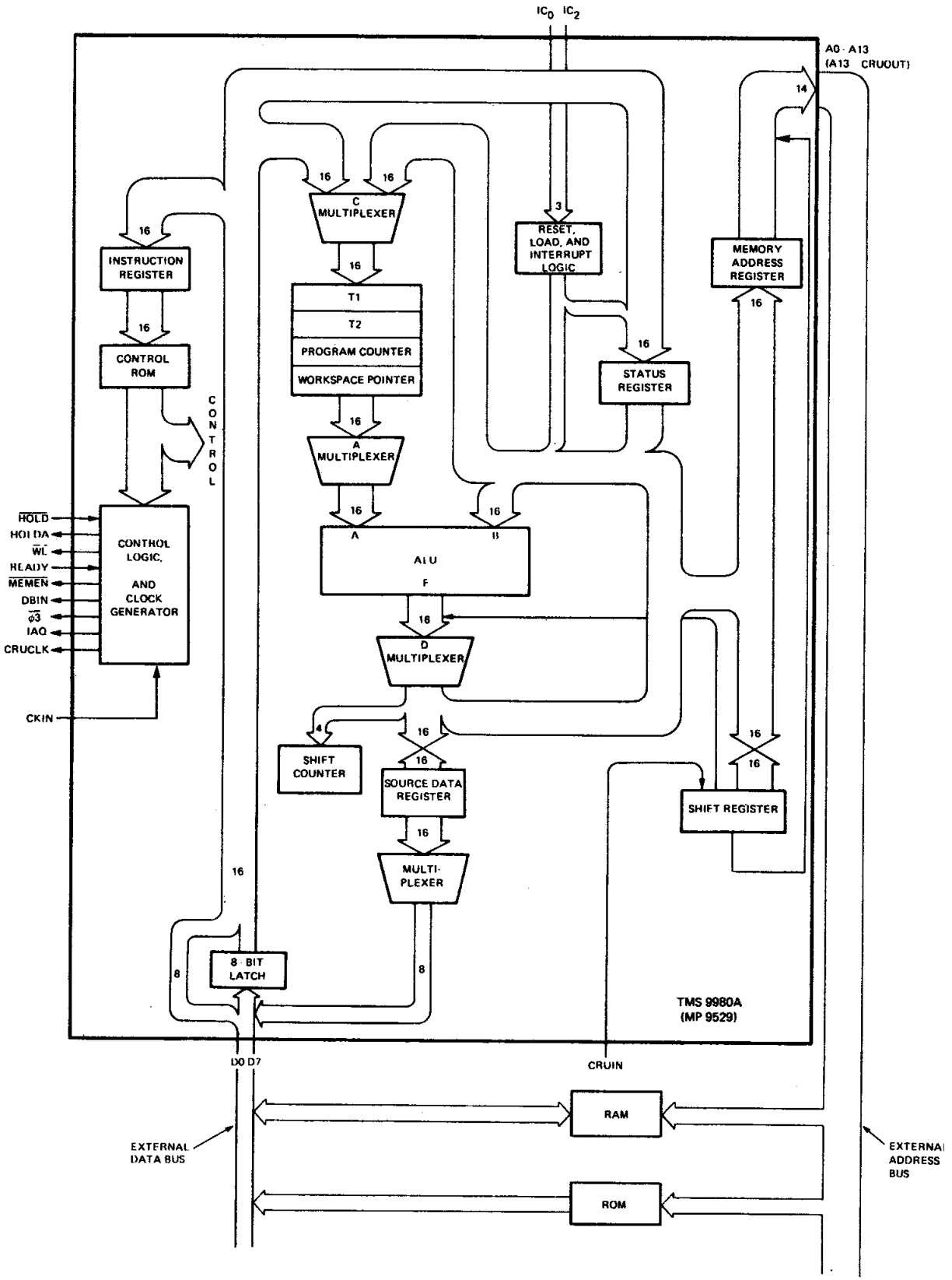


Figure 5-2. TMS 9980A With RAM/ROM Memory

5.3.3 STATUS REGISTER (ST)

The Status Register contains relevant information on preceding instructions and current interrupt level. Included are:

- Results of logical and two's complement comparisons (many instructions automatically compare the results to zero).
- Carry and overflow.
- Odd parity found (byte instructions only).
- XOP being executed.
- Lowest priority interrupt level that will be currently recognized by the processor.

The status register is shown in Figure 5-3.

5.3.3.1 Logical Greater Than

This bit contains the result of a comparison of words or bytes as unsigned binary numbers. In this case, the most significant bit (MSB) does not indicate a positive or negative sign. The MSB of words being logically compared represents 2^{15} (32,768), and the MSB of bytes being logically compared represents 2^{7} (128).

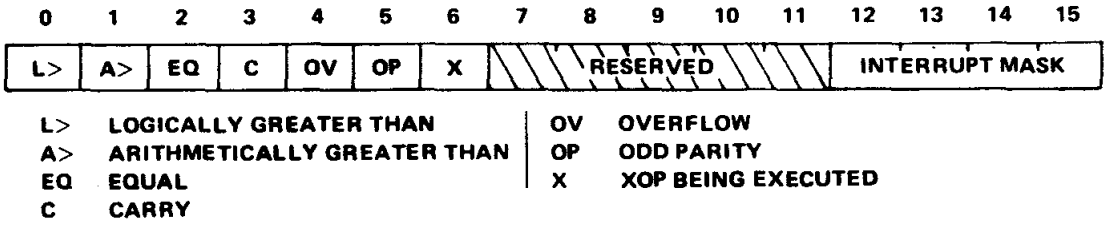


Figure 5-3. Status Register

5.3.3.2 Arithmetic Greater Than

The arithmetic greater than bit contains the result of a comparison of words or bytes as two's complement numbers. In this comparison, the MSB of words or bytes being compared represents the sign of the number, zero for positive, or one for negative.

5.3.3.3 Equal

The equal bit is set when the words or bytes being compared are equal.

5.3.3.4 Carry

The carry bit is set by a carry out of the MSB of a word or byte (sign bit) during arithmetic operations. The carry bit is used by the shift operation to store the value of the last bit shifted out of the workspace register being shifted.

5.3.3.5 Overflow

The overflow bit is set when the result of an arithmetic operation is too large or too small to be correctly represented in two's complement (arithmetic) representation. In addition operations, overflow is set when the MSB's of the operands are equal and the MSB of the result is not equal to the MSB of the destination operand. In subtraction operations, the overflow bit is set when the MSB's of the operands are not equal, and the MSB of the result is not equal to the MSB of the destination operand. For a divide operation, the overflow bit is set when the most significant sixteen bits of the dividend (a 32-bit value) are greater than or equal to the divisor. For an arithmetic left shift, the overflow bit is set if the MSB of the workspace register being shifted changes value. For the absolute value and negate instructions, the overflow bit is set when the source operand is the maximum negative value, 8000_{16}.

5.3.3.6 Odd Parity

The odd parity bit is set in byte operations when the parity of the result is odd, and is reset when the parity is even. The parity of a byte is odd when the number of bits having a value of one is odd; when the number of bits having a value of one is even, the parity of the byte is even.

5.3.3.7 Extended Operation

The extended operation bit of the Status Register is set to one when a software implemented extended operation (XOP) is initiated.

5.3.3.8 Status Bit Summary

Table 5-1 lists the instruction set and the status bits affected by each instruction.

5.4 SOFTWARE REGISTERS

Registers used by programs are contained in memory. This speeds up context-switch time because the content of only one register (WP hardware register) needs to be saved instead of the entire register file. The WP, PC, and ST register contents are saved in a context switch.

A workspace is a contiguous 16 word area; its memory location can be designated by placing a value in the WP register through software or a keyboard monitor command. A program can use one or several workspace areas, depending upon register requirements.

More than three-fourths of the instructions can address the workspace register file; all shift instructions and most immediate operand instructions use workspace registers exclusively.

Figure 5-4 is an example of a workspace file in high-order memory (RAM). A workspace in ROM would be ineffective since it could not be written into. Note that several registers are used by particular instructions.

TABLE 5-1. STATUS BITS AFFECTED BY INSTRUCTIONS

MNEMONIC	L >	A >	EQ	C	OV	OP	X	MNEMONIC	L >	A >	EQ	C	OV	OP	X
A	X	X	X	X	X	–	–	LDCR	X	X	X	–	–	1	–
AB	X	X	X	X	X	X	–	LI	X	X	X	–	–	–	–
ABS	X	X	X	X	X	–	–	LIMI	–	–	–	–	–	–	–
AI	X	X	X	X	X	–	–	LREX	–	–	–	–	–	–	–
ANDI	X	X	X	–	–	–	–	LWPI	–	–	–	–	–	–	–
B	–	–	–	–	–	–	–	MOV	X	X	X	–	–	–	–
BL	–	–	–	–	–	–	–	MOVB	X	X	X	–	–	X	–
BLWP	–	–	–	–	–	–	–	MPY	–	–	–	–	–	–	–
C	X	X	X	–	–	–	–	NEG	X	X	X	X	X	–	–
CB	X	X	X	–	–	X	–	ORI	X	X	X	–	–	–	–
CI	X	X	X	–	–	–	–	RSET	–	–	–	–	–	–	–
CLR	–	–	–	–	–	–	–	RTWP	X	X	X	X	X	X	X
COC	–	–	X	–	–	–	–	S	X	X	X	X	X	–	–
CZC	–	–	X	–	–	–	–	SB	X	X	X	X	X	X	–
DEC	X	X	X	X	X	–	–	SBO	–	–	–	–	–	–	–
DECT	X	X	X	X	X	–	–	SBZ	–	–	–	–	–	–	–
DIV	–	–	–	–	X	–	–	SETO	–	–	–	–	–	–	–
IDLE	–	–	–	–	–	–	–	SLA	X	X	X	X	X	–	–
INC	X	X	X	X	X	–	–	SOC	X	X	X	–	–	–	–
INCT	X	X	X	X	X	–	–	SOCB	X	X	X	–	–	X	–
INV	X	X	X	–	–	–	–	SRA	X	X	X	X	–	–	–
JEQ	–	–	–	–	–	–	–	SRC	X	X	X	X	–	–	–
JGT	–	–	–	–	–	–	–	SRL	X	X	X	X	–	–	–
JH	–	–	–	–	–	–	–	STCR	X	X	X	–	–	1	–
JHE	–	–	–	–	–	–	–	STST	–	–	–	–	–	–	–
JL	–	–	–	–	–	–	–	STWP	–	–	–	–	–	–	–
JLE	–	–	–	–	–	–	–	SWPB	–	–	–	–	–	–	–
JLT	–	–	–	–	–	–	–	SZC	X	X	X	–	–	–	–
JMP	–	–	–	–	–	–	–	SZCB	X	X	X	–	–	X	–
JNC	–	–	–	–	–	–	–	TB	–	–	X	–	–	–	–
JNE	–	–	–	–	–	–	–	X	2	2	2	2	2	2	2
JNO	–	–	–	–	–	–	–	XOP	2	2	2	2	2	2	2
JOC	–	–	–	–	–	–	–	XOR	X	X	X	–	–	–	–
JOP	–	–	–	–	–	–	–								

NOTES

1. When an LDCR or STCR instruction transfers eight bits or less, the OP bit is set or reset as in byte instructions. Otherwise these instructions do not affect the OP bit.
2. The X instruction does not affect any status bit; the instruction executed by the X instruction sets status bits normally for that instruction. When an XOP instruction is implemented by software, the XOP bit is set, and the subroutine sets status bits normally.

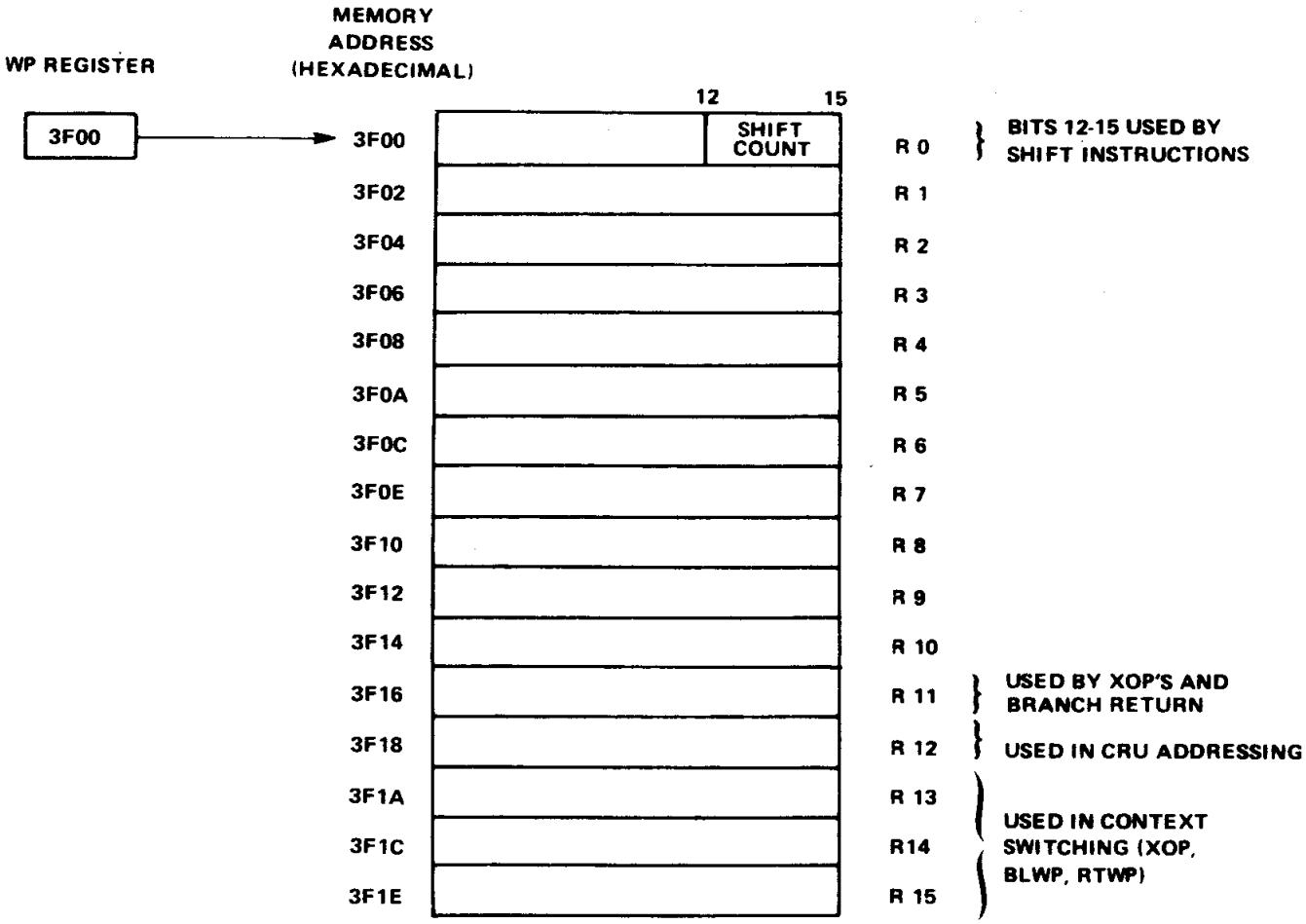


Figure 5-4. Workspace Example

5.5 INSTRUCTION FORMATS AND ADDRESSING MODES

Instructions designate the operations for a computer to perform. The TM 990/189 microcomputer can execute 69 instructions. To implement this instruction set, nine instruction formats are used. Figure 5-5 shows the TM 990/189 instruction formats.

In order to construct instructions in machine language, the programmer must have a knowledge of the fields and formats of the instructions. This knowledge is often very important in debugging operations because it allows the programmer to change bits within an instruction in order to solve an execution problem.

Each 16 bit word is broken down into fields. As an example, Format 1 consists of six different fields. These fields are Op Code, B, T_D, DR, T_S and SR. Examination of other formats in Figure 5-5 will yield three more fields: these fields are Signed Displacement, C, and R. These nine fields can be divided into five groups. A brief description of each field group follows:

1. Op Code — purpose of instruction (op codes are listed alphabetically in Table 5-2 and by format number in Table 5-3).

2. T_D or T_S — Type of Addressing Mode used for destination or source registers or if symbolic addressing is used.

FORMAT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	GENERAL USE
1	OP CODE				B		T_D		DR			T_S		SR			ARITHMETIC
2	OP CODE									SIGNED DISPLACEMENT							JUMP
3	OP CODE							WR				T_S		SR			LOGICAL
4	OP CODE							C				T_S		SR			CRU
5	OP CODE									C				R			SHIFT
6	OP CODE											T_S		SR			PROGRAM
7	OP CODE												NOT USED				CONTROL
8	OP CODE											N		R			IMMEDIATE
9	OP CODE								DR			T_S		SR			MPY, DIV, XOP

KEY

B	BYTE INDICATOR (1=BYTE)
T_D	DESTINATION ADDRESS TYPE*
DR	DESTINATION REGISTER
T_S	SOURCE ADDRESS TYPE*
SR	SOURCE REGISTER
C	CRU TRANSFER COUNT OR SHIFT COUNT
R	REGISTER
N	NOT USED

*T_D OR T_S	ADDRESS MODE TYPE
00	DIRECT REGISTER
01	INDIRECT REGISTER
10	SYMBOLIC MEMORY ADDRESSING, NOT INDEXED (SR OR DR = 0) SYMBOLIC MEMORY ADDRESSING, INDEXED (SR OR DR > 0)
11	INDIRECT REGISTER, AUTOINCREMENT REGISTER

Figure 5-5. TM 990/189M Instruction Formats

3. R, DR, SR — register fields, R (one register involved), DR (destination register), and SR (source register).

4. Signed Displacement — a signed word (not byte) count to be added to the program counter in jump instructions or a signed value to be added to the CRU base address in single-bit CRU instructions.

5. C — number of bits to be transferred or shifted. For C = 1 to C = 15, 1 to 15 bits will be transferred or shifted. If C = 0, 16 bits will be transferred or the shift count will be in the LSB's of register 0.

TABLE 5-2. OP CODES (ALPHABETICAL)

Mnemonic	Op Code (Binary Value) 012345678910	Op Code (Hex Value)*	Format*
A	1010	A	1
AB	1011	B	1
ABS	0000011101	074	6
AI	00000010001	022	8
ANDI	00000010010	024	8
B	0000010001	044	6
BL	0000011010	068	6
BLWP	0000010000	040	6
C	1000	8	1
CB	1001	9	1
CI	00000010100	028	8
CKOF	00000011110	03C	7
CKON	00000011101	03A	7
CLR	0000010011	04C	6
COC	001000	20	3
CZC	001001	24	3
DEC	0000011000	060	6
DECT	0000011001	064	6
DIV	001111	3C	3
IDLE	00000011010	034	7
INC	0000010110	058	6
INCT	0000010111	05C	6
INV	0000010101	054	6
JEQ	00010011	13	2
JGT	00010101	15	2
JH	00011011	1B	2
JHE	00010100	14	2
JL	00011010	1A	2
JLE	00010010	12	2
JLT	00010001	11	2
JMP	00010000	10	2
JNC	00010111	17	2
JNE	00010110	16	2
JNO	00011001	19	2
JOC	00011000	18	2
JOP	00011100	1C	2
LDCR	001100	30	4
LI	00000010000	020	8
LIMI	00000011000	030	8
LREX	00000011111	03E	7
LWPI	00000010111	02E	8
MOV	1100	C	1
MOVB	1101	D	1
MPY	001110	38	3
NEG	0000010100	050	6
ORI	00000010011	026	8
RSET	00000011011	036	7
RTWP	00000011100	038	7
S	0110	6	1
SB	0111	7	1
SBO	00011101	1D	2
SBZ	00011110	1E	2
SETO	0000011100	070	6
SLA	00001010	0A	5
SOC	1110	E	1
SOCB	1111	F	1
SRA	00001000	08	5

TABLE 5-2. OP CODES (ALPHABETICAL) (Concluded)

Mnemonic	Op Code (Binary Value) 012345678910	Op Code (Hex Value)*	Format*
SRC	00001011	0B	5
SRL	00001001	09	5
STCR	001101	34	4
STST	00000010110	02C	8
STWP	00000010101	02A	8
SWPB	0000011011	06C	6
SZC	0100	4	1
SZCB	0101	5	1
TB	00011111	1F	2
X	0000010010	048	6
XOP	001011	2C	9
XOR	001010	28	3

*NOTES

1. The op code value for Format 1 instructions was obtained by combining the 3-bit op code and the 1-bit byte field.

2. To obtain the op code (hex value) for a particular instruction, divide the op code bits into groups of four starting at the left and convert each group into its hex equivalent (supply zeroes to complete any incomplete block of four).

Example:

Instruction	Op Code	Op Code Grouped	Op Code (Hex Value)
DIV	001111	0011 1100	3C

Supplied Zeros

TABLE 5-3. OP CODES BY FORMAT

Format No.	Mnemonic	Op Code 012345678910	B	Op Code (Hex Value)*
1	A	101	0	A
1	AB	101	1	B
1	C	100	0	8
1	CB	100	1	9
1	MOV	110	0	C
1	MOVB	110	1	D
1	S	011	0	6
1	SB	011	1	7
1	SOC	111	0	E
1	SOCB	111	1	F
1	SZC	010	0	4
1	SZCB	010	1	5
2	JEQ	00010011		13
2	JGT	00010101		15
2	JH	00011011		1B
2	JHE	00010100		14
2	JL	00011010		1A
2	JLE	00010010		12
2	JLT	00010001		11
2	JMP	00010000		10
2	JNC	00010111		17
2	JNE	00010110		16
2	JNO	00011001		19
2	JOC	00011000		18
2	JOP	00011100		1C
2	SBO	00011101		1D
2	SBZ	00011110		1E

TABLE 5-3. OP CODES BY FORMAT (Concluded)

Format No.	Mnemonic	Op Code 012345678910	B	Op Code (Hex Value)*
2	TB	00011111		1F
3	COC	001000		20
3	CZC	001001		24
3	XOR	001010		28
3	MPY	001110		38
3	DIV	001111		36
4	LDCR	001100		30
4	STCR	001101		34
5	SLA	00001010		0A
5	SRA	00001000		08
5	SRC	00001011		0B
5	SRL	00001001		09
6	B	0000010001		044
6	BL	0000011010		068
6	BLWP	0000010000		040
6	CLR	0000010011		04C
6	SETO	0000011100		070
6	INV	0000010101		054
6	NEG	0000010100		050
6	ABS	0000011101		074
6	SWPB	0000011011		06C
6	INC	0000010110		058
6	INCT	0000010111		05C
6	DEC	0000011000		060
6	DECT	0000011001		064
6	X	0000010010		048
7	IDLE	00000011010		034
7	RSET	00000011011		036
7	CKOF	00000011110		03C
7	CKON	00000011101		03A
7	LREX	00000011111		03E
7	RTWP	00000011100		038
8	AI	00000010001		022
8	ANDI	00000010010		024
8	CI	00000010100		028
8	LI	00000010000		020
8	ORI	00000010011		026
8	LWPI	00000010111		02E
8	LIMI	00000011000		030
8	STST	00000010110		02C
8	STWP	00000010101		02A
9	XOP	001011		2C

*NOTES

1. The op code value for Format 1 instructions was obtained by combining the 3-bit op code and the 1-bit byte field.

2. To obtain the op code (hex value) for a particular instruction, divide the op code bits into groups of four starting at the left and convert each group into its hex equivalent (supply zeroes to complete any incomplete block of four).

5.5.1 ADDRESSING MODES

The TM 990/189 microcomputer provides seven addressing modes. The addressing modes are as follows:

1. Direct Register Addressing
2. Indirect Register Addressing
3. Indirect Register Autoincrement Addressing
4. Symbolic Memory Addressing, Not Indexed
5. Symbolic Memory Addressing, Indexed
6. Immediate Addressing
7. Program Counter Relative Addressing

These addressing modes are described in the following paragraphs.

5.5.1.1 Direct Register Addressing ($T=00_2$)

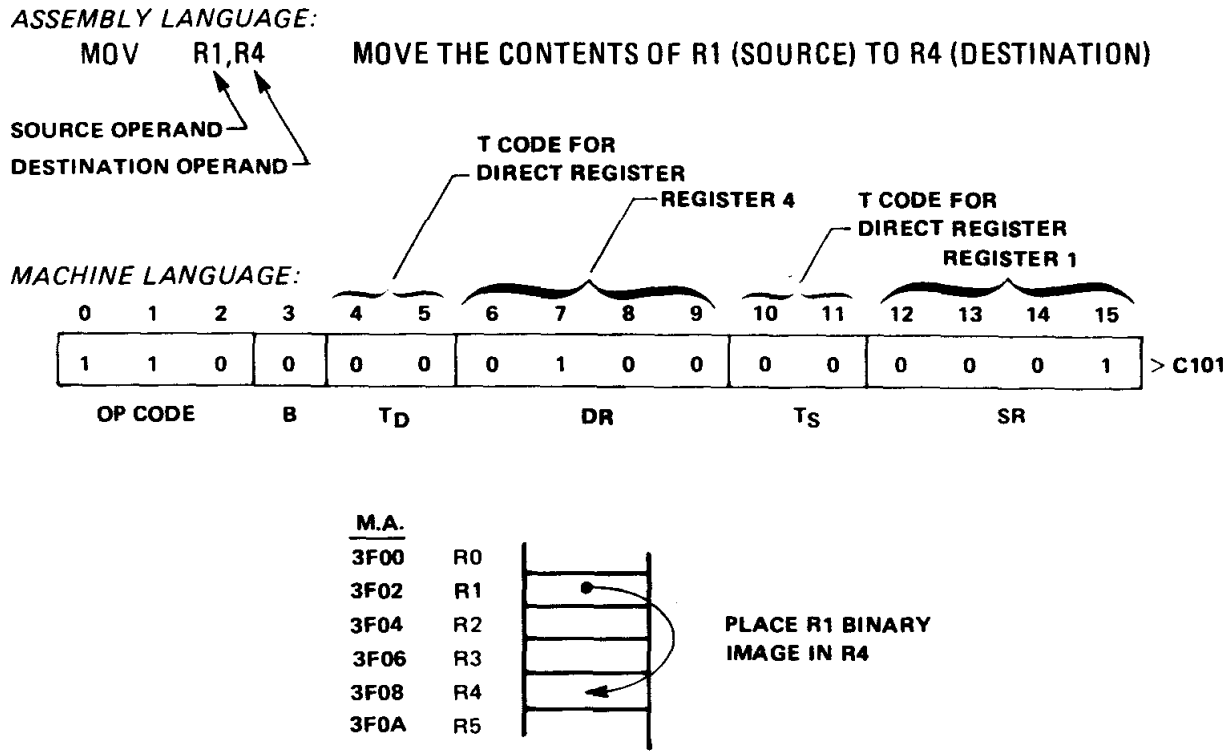
In direct register addressing, execution involves data contained within one of the 16 workspace registers. In the first example in Figure 5-6, both the source and destination operands are registers as noted in the assembly language example at the top of the figure. Both T fields contain 00_2 to denote direct register addressing and their associated register fields contain the binary value of the number of the register affected. The 110_2 in the op code field identifies this instruction as a move instruction. Since the B field contains a zero, the data moved will be the full 16 bits of the register (1 byte instruction addressing a register would address the left byte of the register). The instruction specifies moving the contents of register 1 to register 4, thus changing the contents of register 4 to the same value as in register 1. Note that the assembly language statement is constructed so that the source register is the first item in the operand while the destination register is the second item in the operand. This order is reversed in the machine language construction with the destination register and its T field first and the source register and its T field second.

5.5.1.2 INDIRECT REGISTER ADDRESSING ($T=01_2$)

In indirect register addressing, the register does not contain the data to be affected by the instruction; instead, the register contains the address within memory of where that data is stored. For example, the instruction in Figure 5-7 specifies to move the contents of register 1 to the address which is contained in register 4 (indirect register 4). (Note that an indirect register address is written as a term preceded by an asterisk (*).)

Instead of moving the value in register 1 to register 4 as was the case in Figure 5-6, the CPU must first read in the 16-bit value in register 4 and use that value as a memory address at which location the contents of register 1 will be stored. In the example, register 4 contains the value $3D00_{16}$. This instruction stores the value in register 1 into memory address (M.A.) $3D00_{16}$.

EXAMPLE 1



EXAMPLE 2

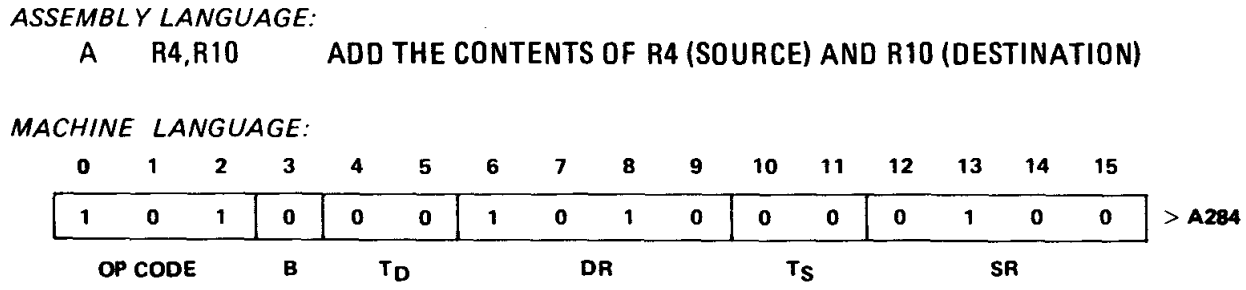
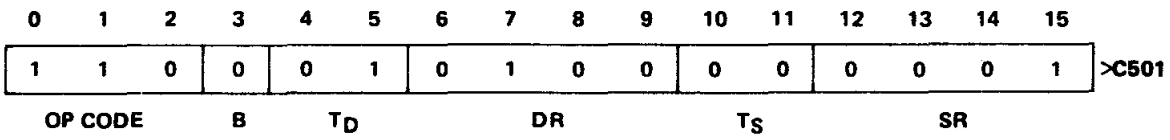


Figure 5-6. Direct Register Addressing Examples

ASSEMBLY LANGUAGE:

```
MOV    R1,*R4        MOVE THE CONTENTS OF RI (SOURCE) TO ADDRESS IN R4 (DESTINATION)
```

MACHINE LANGUAGE:



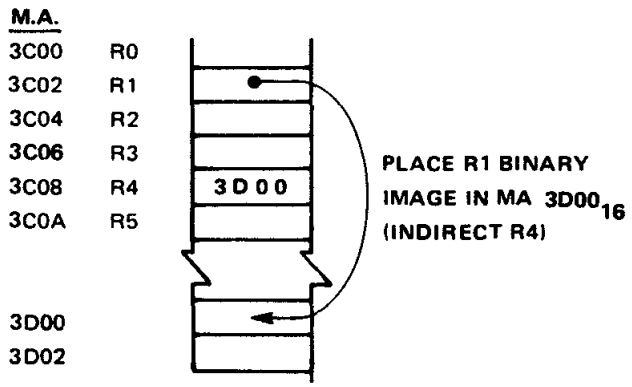


Figure 5-7. Indirect Register Addressing Example

In direct register addressing, the contents of a register are addressed. In indirect register addressing, the CPU goes to the register to find out what memory location to address. This form of addressing is especially suited for repeating an instruction while accessing successive memory addresses. For example, if you wished to add a series of numbers in 100 consecutive memory locations, you could place the address of the first number in a register, and execute and add indirect through that register, causing the contents of the first memory address (source operand) to be added to another register or memory address (destination operand). Then you could increment the contents of the register containing the address of the number, loop back to the add instruction, and repeat the add, only this time you will be adding the contents of the next memory address to the accumulator (destination operand). This way a whole string of data can be summed using a minimum of instructions. Of course, you would have to include control instructions that would signal when the entire list of 100 addresses have been added, but there are obvious advantages in speed of operation, better utilization of memory space, and ease in programming.

5.5.1.3 Indirect Register Autoincrement Addressing ($T=11_2$)

Indirect register autoincrement addressing is the same as indirect register addressing (paragraph 5.5.1.2) except for an additional feature — automatic incrementation of the register. This saves the requirement of adding an increment (by one or two) instruction to increment the register being used in the indirect mode. The increment will be a value of one for byte instructions (e.g., add byte or AB) or a value of two for full word instructions (e.g., add word or A).

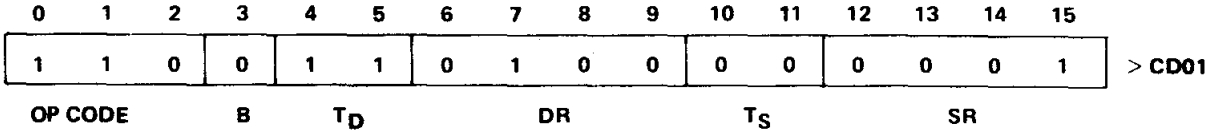
In assembly language the register number is preceded by an asterisk (*) and followed by a plus sign (+) as shown in Figure 5-8. Note in the figure that the contents of register 4 was incremented by two since the instruction was a move word (vs. byte) instruction. If the example used a move byte instruction, the contents of the register would be incremented by one so that successive bytes would be addressed (the 16-bit word addresses in memory are always even numbers or multiples of two since each contains two bytes). Bytes are also addressed by various instructions of the 990 instruction set.

Note that only a register can contain the indirect address.

ASSEMBLY LANGUAGE:

```
MOV    R1,*R4+      MOVE THE CONTENTS OF RI TO ADDRESS CONTAINED IN R4,
                    INCREMENT ADDRESS BY 2
```

MACHINE LANGUAGE:



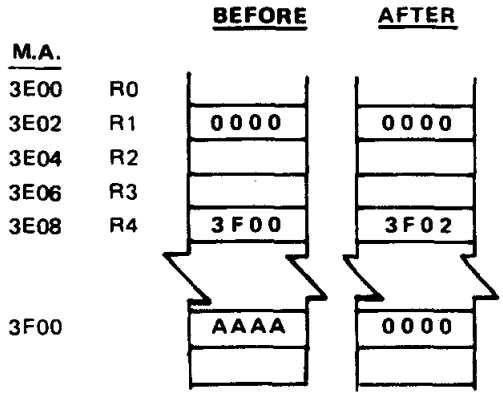


Figure 5-8. Indirect Register Autoincrement Addressing Example

5.5.1.4 Symbolic Memory Addressing, Not Indexed (T=10_2)

This mode does not use a register as an address or as a container of an address. Instead, the address is a 16-bit value stored in the second or third word of the instruction. The SR or DR fields will be all zeroes as shown for the destination register field in the first example of Figure 5-9. When the T field contains 10_2, the CPU retrieves the contents of the memory location and uses these contents as the effective address. In assembly language, a symbolic address is preceded by an at sign (@) to differentiate a numerical memory address from a register number. All alphanumeric labels must be preceded by an @ sign; numerical values preceded by an @ sign will be assembled as an absolute address.

In the second example in Figure 5-9, both the source and destination operands are symbolic memory addresses. In this case, the source address is the first word following the instruction and the destination is the second word following the instruction in machine language.

EXAMPLE 1

ASSEMBLY LANGUAGE:

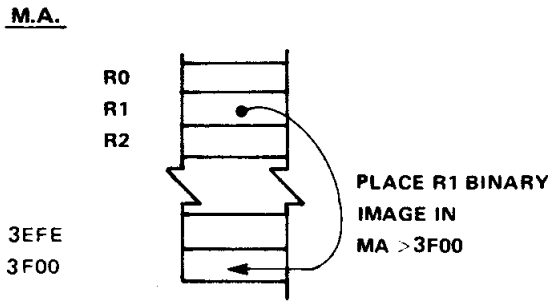
MOV R1,@>3F00 MOVE THE CONTENTS OF R1 TO ADDRESS >3F00

NOTE

The > sign indicates hexidecimal representation.

MACHINE LANGUAGE:

	OP CODE			B	T_D		DR				T_S		SR				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1st WORD	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	1	>C801
2nd WORD	0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	>3F00



EXAMPLE 2

ASSEMBLY LANGUAGE:

MOV @>3F0A,@>3F08 MOVE THE CONTENTS OF >3F0A TO >3F08

MACHINE LANGUAGE:

	OP CODE			B	T_D		DR				T_S		SR				
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1st WORD	1	1	0	0	1	0	0	0	0	0	1	0	0	0	0	0	>C820
2nd WORD	0	0	1	1	1	1	1	1	0	0	0	0	1	0	1	0	>3F0A(SOURCE)
3rd WORD	0	0	1	1	1	1	1	1	0	0	0	0	1	0	0	0	>3F08(DESTINATION

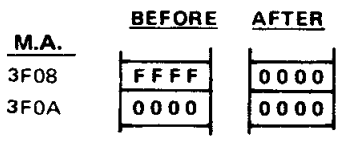


Figure 5-9. Symbolic Memory Addressing Examples

5.5.1.5 Symbolic Memory Addressing, Indexed (T=10_2)

Note that the T field for indexed as well as non-indexed symbolic addressing is the same (10_2). In order to differentiate between the two different modes, the associated SR or DR field is interrogated; if this field is all zeroes (0000_2), non-indexed addressing is specified; if the SR or DR field is greater than zero, indexing is specified and the non-zero value is the index register number. As a result, register 0 cannot be used as an index register.

In assembly language, the symbolic address is followed by the number of the index register in parentheses. In the example in Figure 5-10, the source operand is non-indexed symbolic memory addressing while the destination operand is indexed symbolic memory addressing. In this case, the destination effective address is the sum of the $3F02_{16}$ value in the source memory address word plus the value in the index register (0004_{16}). The effective address in this case is $3F06_{16}$ as shown by the addition in the left part of the figure.

Note that only symbolic addressing can be indexed.

ASSEMBLY LANGUAGE:

```
MOV    @>3F00,@>3F02(R1)      MOVE THE CONTENTS OF >3F00 TO >3F02 + RI CONTENTS
```

MACHINE LANGUAGE:

OP CODE			B	T_D		DR				T_S		SR				
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	1	0	0	1	0	0	0	0	1	1	0	0	0	0	0	>C860
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	>3F00 (SOURCE)
0	0	1	1	1	1	1	1	0	0	0	0	0	0	1	0	>3F02 (DESTINATION

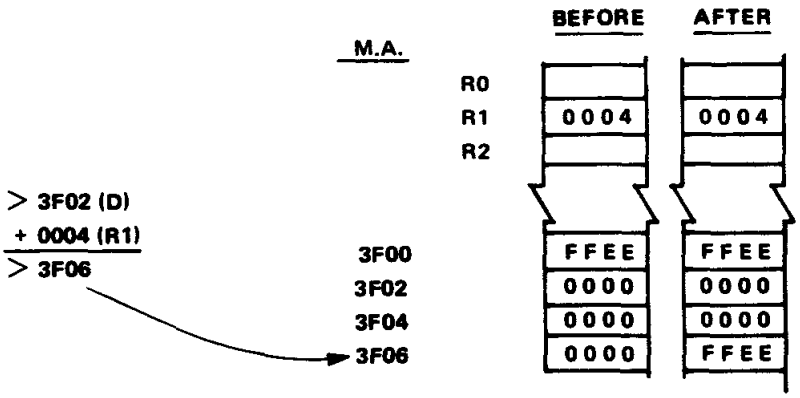


Figure 5-10. Symbolic Memory Addressing, Indexed Example

5.5.1.6 Immediate Addressing

This mode allows an absolute value to be specified as an operand; this value is used in connection with a register contents or is loaded into the WP or the Status Register interrupt mask. Examples are shown below:

```
LI      R2, 100      LOAD 100 INTO REGISTER 2
CI      R8,> 100     COMPARE R8 CONTENTS TO > 100, RESULTS IN ST
LWPI    > 3C00       SET WP TO MA > 3C00
```

5.5.1.7 Program Counter Relative Addressing

This mode allows a change in Program Counter contents, either an unconditional change or a change conditional on Status Register contents. Examples are shown below:

```
JMP     $+6          JUMP TO LOCATION, 6 BYTES FORWARD
JMP     THERE        JUMP TO LOCATION LABELLED THERE
JEQ     $+4          IF ST EQ BIT = 1, JUMP 4 BYTES (MA + 4)
JMP     > 3E26       JUMP TO M.A. > 3E26 (LINE-BY-LINE ASSEMBLER ONLY)
```

The dollar symbol ($) means "from this address"; thus, $+6 means "this address plus 6 bytes."

5.6 INSTRUCTIONS

Table 5-4 lists terms used in describing the instructions of the TM 990/189. Table 5-5 is an alphabetical list of instructions. Table 5-6 is a numerical list of instructions by op code. Examples are shown in both assembly language (A.L.) and machine language (M.L.). The greater-than sign (>) indicates hexadecimal.

TABLE 5-4. INSTRUCTION DESCRIPTION TERMS

TERM	DEFINITION	TERM	DEFINITION
B	Byte indicator (1 = byte, 0 = word)	T_D	Destination address modifier
C	Bit count	T_S	Source address modifier
DR	Destination address register	WR or R	Workspace register
DA	Destination address	WRn or Rn	Workspace register n
IOP	Immediate operand	(n)	Contents of n
LSB(n)	Least significant (right most) bit of (n)	a → b	a is transferred to b
M.A.	Memory Address	(a) → b	Contents of a are transferred to b
MSB(n)	Most significant (left most) bit of (n)	[n]	Absolute value of n
N	Don't care	+	Arithmetic addition
PC	Program counter	−	Arithmetic subtraction
Result	Result of operation performed by instruction	AND	Logical AND
SR	Source address register	OR	Logical OR
SA	Source address	⊕	Logical exclusive OR
ST	Status register	n	Logical complement of n
STn	Bit n of status register	>	Hexadecimal value

TABLE 5-5. INSTRUCTION SET, ALPHABETICAL INDEX

ASSEMBLY LANGUAGE MNEMONIC	MACHINE LANGUAGE OP CODE	FORMAT	STATUS REG. BITS AFFECTED	RESULT COMPARED TO ZERO	INSTRUCTION
A	A000	1	0-4	X	Add (word)
AB	B000	1	0-5	X	Add (byte)
ABS	0740	6	0-2	X	Absolute Value
AI	0220	8	0-4	X	Add Immediate
ANDI	0240	8	0-2	X	AND Immediate
B	0440	6			Branch
BL	0680	6	–		Branch and Link (R11)
BLWP	0400	6	–		Branch; New Workspace Pointer
C	8000	1	0-2		Compare (word)
CB	9000	1	0-2,5		Compare (byte)
CI	0280	8	0-2		Compare Immediate
CKOF	03C0	7	–		User Defined
CKON	03A0	7	–		User Defined
CLR	04C0	6	–		Clear Operand
COC	2000	3	2		Compare Ones Corresponding
CZC	2400	3	2		Compare Zeroes Corresponding
DEC	0600	6	0-4	X	Decrement (by one)
DECT	0640	6	0-4	X	Decrement (by two)
DIV	3C00	9	4		Divide
IDLE	0340	7	–		Computer Idle
INC	0580	6	0-4	X	Increment (by one)
INCT	05C0	6	0-4	X	Increment (by two)
INV	0540	6	0-2	X	Invert (One's Complement)
JEQ	1300	2	–		Jump Equal (ST2=1)
JGT	1500	2	–		Jump Greater Than (ST1=1), Arithmetic
JH	1B00	2	–		Jump High (ST0=1 and ST2=0), Logical
JHE	1400	2	–		Jump High or Equal (ST0 or ST2=1), Logical
JL	1A00	2	–		Jump Low (ST0 and ST2=0), Logical
JLE	1200	2	–		Jump Low or Equal (ST0=0 or ST2=1), Logical
JLT	1100	2	–		Jump Less Than (ST1 and ST2=0), Arithmetic
JMP	1000	2	–		Jump Unconditional
JNC	1700	2	–		Jump No Carry (ST3=0)
JNE	1600	2	–		Jump Not Equal (ST2=0)
JNO	1900	2	–		Jump No Overflow (ST4=0)
JOC	1800	2	–		Jump On Carry (ST3=1)

TABLE 5-5. INSTRUCTION SET, ALPHABETICAL INDEX (Concluded)

ASSEMBLY LANGUAGE MNEMONIC	MACHINE LANGUAGE OP CODE	FORMAT	STATUS REG. BITS AFFECTED	RESULT COMPARED TO ZERO	INSTRUCTION
JOP	1C00	2	—		Jump Odd Parity (ST5=1)
LDCR	3000	4	0-2,5	X	Load CRU
LI	0200	8	—	X	Load Immediate
LIMI	0300	8	12-15		Load Interrupt Mask Immediate
LREX	03E0	7	12-15		Load and Execute
LWPI	02E0	8	—		Load Immediate to Workspace Pointer
MOV	C000	1	0-2	X	Move (word)
MOVB	D000	1	0-2,5	X	Move (byte)
MPY	3800	9	—		Multiply
NEG	0500	6	0-2	X	Negate (Two's Complement)
ORI	0260	8	0-2	X	OR Immediate
RSET	0360	7	12-15		Reset AU
RTWP	0380	7	0-15		Return from Context Switch
S	6000	1	0-4	X	Subtract (word)
SB	7000	1	0-5	X	Subtract (byte)
SBO	1D00	2	—		Set CRU Bit to One
SBZ	1E00	2	—		Set CRU Bit to Zero
SETO	0700	6	—		Set Ones
SLA	0A00	5	0-4	X	Shift Left Arithmetic
SOC	E000	1	0-2	X	Set Ones Corresponding (word)
SOCB	F000	1	0-2,5	X	Set Ones Corresponding (byte)
SRA	0800	5	0-3	X	Shift Right (sign extended)
SRC	0B00	5	0-3	X	Shift Right Circular
SRL	0900	5	0-3	X	Shift Right Logical
STCR	3400	4	0-2,5	X	Store From CRU
STST	02C0	8	—		Store Status Register
STWP	02A0	8	—		Store Workspace Pointer
SWPB	06C0	6	—		Swap Bytes
SZC	4000	1	0-2	X	Set Zeroes Corresponding (word)
SZCB	5000	1	0-2,5	X	Set Zeroes Corresponding (byte)
TB	1F00	2	2		Test CRU Bit
X	0480	6	—		Execute
XOP	2C00	9	6		Extended Operation
XOR	2800	3	0-2	X	Exclusive OR

TABLE 5-6. INSTRUCTION SET, NUMERICAL INDEX

MACHINE LANGUAGE OP CODE (HEXADECIMAL)	ASSEMBLY LANGUAGE MNEMONIC	INSTRUCTION	FORMAT	STATUS BITS AFFECTED
0200	LI	Load Immediate	8	0-2
0220	AI	Add Immediate	8	0-4
0240	ANDI	And Immediate	8	0-2
0260	ORI	Or Immediate	8	0-2
0280	CI	Compare Immediate	8	0-2
02A0	STWP	Store WP	8	–
02C0	STST	Store ST	8	–
02E0	LWPI	Load WP Immediate	8	–
0300	LIMI	Load Int. Mask	8	12-15
0340	IDLE	Idle	7	–
0360	RSET	Reset AU	7	12-15
0380	RTWP	Return from Context Sw.	7	0-15
03A0	CKON	User Defined	7	–
03C0	CKOF	User Defined	7	–
03E0	LREX	Load & Execute	7	–
0400	BLWP	Branch; New WP	6	–
0440	B	Branch	6	–
0480	X	Execute	6	–
04C0	CLR	Clear to Zeroes	6	–
0500	NEG	Negate to Ones	6	0-2
0540	INV	Invert	6	0-2
0580	INC	Increment by 1	6	0-4
05C0	INCT	Increment by 2	6	0-4
0600	DEC	Decrement by 1	6	0-4
0640	DECT	Decrement by 2	6	0-4
0680	BL	Branch and Link	6	–
06C0	SWPB	Swap Bytes	6	–
0700	SETO	Set to Ones	6	–
0740	ABS	Absolute Value	6	0-2
0800	SRA	Shift Right Arithmetic	5	0-3
0900	SRL	Shift Right Logical	5	0-3
0A00	SLA	Shift Left Arithmetic	5	0-4
0B00	SRC	Shift Right Circular	5	0-3
1000	JMP	Unconditional Jump	2	–
1100	JLT	Jump on Less Than	2	–
1200	JLE	Jump on Less Than or Equal	2	–
1300	JEQ	Jump on Equal	2	–
1400	JHE	Jump on High or Equal	2	–
1500	JGT	Jump on Greater Than	2	–
1600	JNE	Jump on Not Equal	2	–
1700	JNC	Jump on No Carry	2	–
1800	JOC	Jump on Carry	2	–
1900	JNO	Jump on No Overflow	2	–
1A00	JL	Jump on Low	2	–
1B00	JH	Jump on High	2	–
1C00	JOP	Jump on Odd Parity	2	–
1D00	SBO	Set CRU Bits to Ones	2	–
1E00	SBZ	Set CRU Bits to Zeroes	2	–
1F00	TB	Test CRU Bit	2	2
2000	COC	Compare Ones Corresponding	3	2

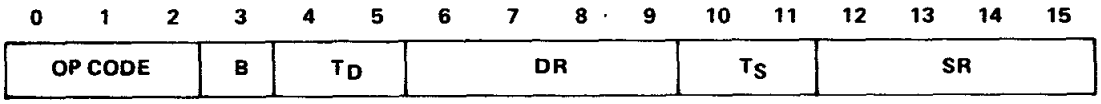
TABLE 5-6. INSTRUCTION SET, NUMBERICAL INDEX (Concluded)

MACHINE LANGUAGE OP CODE (HEXADECIMAL	ASSEMBLY LANGUAGE MNEMONIC	INSTRUCTION	FORMAT	STATUS BITS AFFECTED
2400	CZC	Compare Zeroes Corresponding	3	2
2800	XOR	Exclusive Or	3	0-2
2C00	XOP	Extended Operation	9	
3000	LDCR	Load CRU	4	0-2,5
3400	STCR	Store CRU	4	0-2,5
3800	MPY	Multiply	9	--
3C00	DIV	Divide	9	4
4000	SZC	Set Zeroes Corresponding (Word)	1	0-2
5000	SZCB	Set Zeroes Corresponding (Byte)	1	0-2,5
6000	S	Subtract Word	1	0-4
7000	SB	Subtract Byte	1	0-5
8000	C	Compare Word	1	0-2
9000	CB	Compare Byte	1	0-2,5
A000	A	Add Word	1	0-4
B000	AB	Add Byte	1	0-5
C000	MOV	Move Word	1	0-2
D000	MOVB	Move Byte	1	0-2,5
E000	SOC	Set Ones Corresponding (Word)	1	0-2
F000	SOCB	Set Ones Corresponding (Byte)	1	0-2,5

5.6.1 FORMAT 1 INSTRUCTIONS

These are dual operand instructions with multiple addressing modes for source and destination operands.

GENERAL FORMAT:



If B = 1, the operands are bytes and the operand addresses are byte addresses.
If B = 0, the operands are words and the operand addresses are word addresses.

MNEMONIC	OP CODE 0 1 2	B 3	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
A	1 0 1	0	Add	Yes	0-4	(SA)+(DA) → (DA)
AB	1 0 1	1	Add bytes	Yes	0-5	(SA)+(DA) → (DA)
C	1 0 0	0	Compare	No	0-2	Compare (SA) to (DA) and set appropriate status bits
CB	1 0 0	1	Compare bytes	No	0-2,5	Compare (SA) to (DA) and set appropriate status bits
MOV	1 1 0	0	Move	Yes	0-2	(SA) → (DA)
MOVB	1 1 0	1	Move bytes	Yes	0-2,5	(SA) → (DA)
S	0 1 1	0	Subtract	Yes	0-4	(DA) – (SA) → (DA)
SB	0 1 1	1	Subtract bytes	Yes	0-5	(DA) – (SA) → (DA)
SOC	1 1 1	0	Set ones corresponding	Yes	0-2	(DA) OR (SA) → (DA)
SOCB	1 1 1	1	Set ones corresponding bytes	Yes	0-2,5	(DA) OR (SA) → (DA)
SZC	0 1 0	0	Set zeroes corresponding	Yes	0-2	(DA) AND ($\overline{SA}$) → (DA)
SZCB	0 1 0	1	Set zeroes corresponding bytes	Yes	0-2,5	(DA) AND ($\overline{SA}$) → (DA)

EXAMPLES

(1) ASSEMBLY LANGUAGE:

```
A    @>100,R2        ADD CONTENTS OF MA >100 & R2, SUM IN R2
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	>A0A0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	>0100

(2) ASSEMBLY LANGUAGE:

```
CB   R1,R2       COMPARE BYTE R1 TO R2, SET ST
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	>9081

NOTE

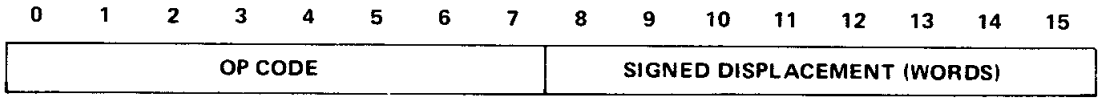
In byte instruction designating a register, the left byte is used. In the above example, the left byte (8 MSB's) of R1 is compared to the left byte of R2, and the ST set to the results.

5.6.2 FORMAT 2 INSTRUCTIONS

5.6.2.1 Jump Instructions

Jump instructions cause the PC to be loaded with the value PC+2 (signed displacement) if bits of the Status Register are at specified values. Otherwise, no operation occurs and the next instruction is executed since the PC was incremented by two and now points to the next instruction. The signed displacement field is a word (not byte) count to be added to PC. Thus, the jump instruction has a range of −128 to 127 words (−256 to 254 bytes) from the memory address following the jump instruction. No ST bits are affected by a jump instruction.

GENERAL FORMAT:



MNEMONIC	OP CODE 0 1 2 3 4 5 6 7	MEANING	ST CONDITION TO CHANGE PC
JEQ	0 0 0 1 0 0 1 1	Jump equal	ST2 = 1
JGT	0 0 0 1 0 1 0 1	Jump greater than	ST1 = 1
JH	0 0 0 1 1 0 1 1	Jump high	ST0 = 1 and ST2 = 0
JHE	0 0 0 1 0 1 0 0	Jump high or equal	ST0 = 1 or ST2 = 1
JL	0 0 0 1 1 0 1 0	Jump low	ST0 = 0 and ST2 = 0
JLE	0 0 0 1 0 0 1 0	Jump low or equal	ST0 = 0 or ST2 = 1
JLT	0 0 0 1 0 0 0 1	Jump less than	ST1 = 0 and ST2 = 0
JMP	0 0 0 1 0 0 0 0	Jump unconditional	unconditional
JNC	0 0 0 1 0 1 1 1	Jump no carry	ST3 = 0
JNE	0 0 0 1 0 1 1 0	Jump not equal	ST2 = 0
JNO	0 0 0 1 1 0 0 1	Jump no overflow	ST4 = 0
JOC	0 0 0 1 1 0 0 0	Jump on carry	ST3 = 1
JOP	0 0 0 1 1 1 0 0	Jump odd parity	ST5 = 1

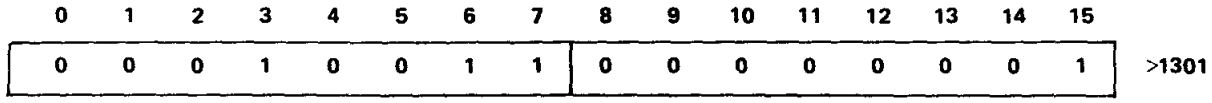
In assembly language, $ in the operand indicates "at this instruction". Essentially JMP $ causes an unconditional loop to the same instruction location, and JMP $+2 is essentially a no-op ($+2 means "here plus two bytes"). Note that the number following the $ is a *byte* count while displacement in machine language is in *words*.

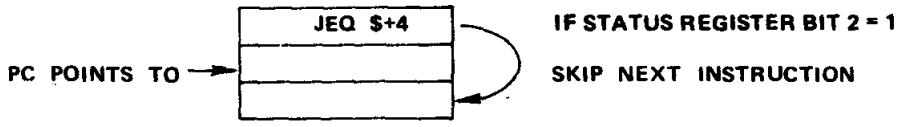
EXAMPLES

(1) ASSEMBLY LANGUAGE:

```
JEQ   $+4      IF EQ BIT SET, SKIP 1 INSTRUCTION
```

MACHINE LANGUAGE:



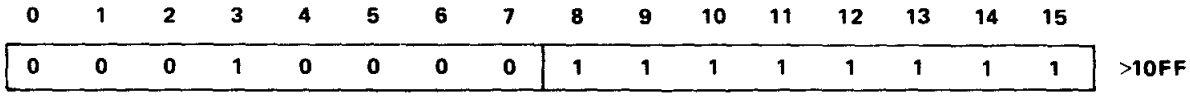


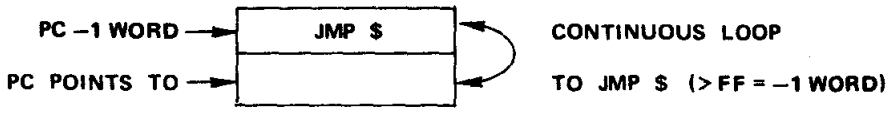
The above instruction continues execution 4 bytes (2 words) from the instruction location or, in other words, two bytes (one word) from the Program Counter value (incremented by 2 and now pointing to next instruction while JEQ executes). Thus, the signed displacement of 1 word (2 bytes) is the value to be added to the PC.

(2) ASSEMBLY LANGUAGE:

```
JMP   $      REMAIN AT THIS LOCATION
```

MACHINE LANGUAGE:





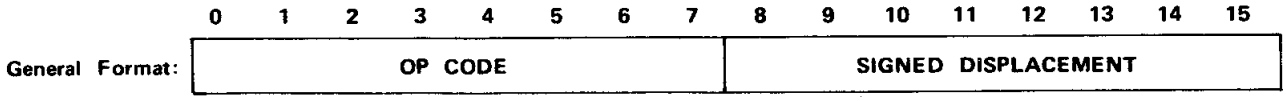
This causes an unconditional loop back to one word less than the Program Counter value (PC + FF = PC−1 word). The Status Register is not checked. A JMP $+2 means "go to the next instruction" and has a displacement of zero (a no-op). No-ops can substitute for deleted codes or can be used for timing purposes.

5.6.2.2 CRU Single-Bit Instructions

These instructions test or set values at the Communications Register Unit (CRU). The CRU bit is selected by the CRU address in bits 4 to 14 of register 12 plus the signed displacement value. The selected bit is set to a one or zero, or it is tested and the bit value placed in equal bit (2) of the Status Register. The signed displacement has a value of −128 to 127.

NOTE

CRU addressing is discussed in detail in Section 7.



MNEMONIC	OP CODE 0 1 2 3 4 5 6 7	MEANING	STATUS BITS AFFECTED	DESCRIPTION
SBO	0 0 0 1 1 1 0 1	Set bit to one	--	Set the selected CRU output bit to 1.
SBZ	0 0 0 1 1 1 1 0	Set bit to zero	--	Set the selected CRU output bit to 0.
TB	0 0 0 1 1 1 1 1	Test bit	2	If the selected CRU input bit = 1, set ST2.

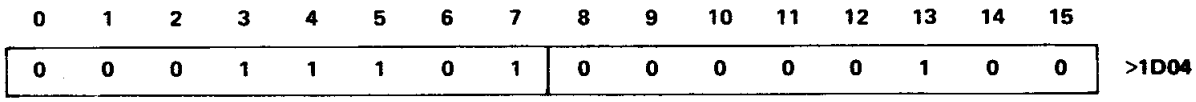
EXAMPLE

R12, BITS 4 TO 14 = >100

ASSEMBLY LANGUAGE:

```
SBO   4      SET CRU ADDRESS >104 TO ONE
```

MACHINE LANGUAGE:



5.6.3 FORMAT 3/9 INSTRUCTIONS

These are dual operand instructions with multiple addressing modes for the source operand, and workspace register addressing for the destination. The MPY and DIV instructions are termed format 9 but both use the same format as format 3. The XOP instruction is covered in paragraph 5.6.9.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
General Format:	OP CODE						DR (REGISTER ONLY)				T_S		SR			

MNEMONIC	OP CODE 0 1 2 3 4 5	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
COC	0 0 1 0 0 0	Compare ones corresponding	No	2	Test (DR) to determine if 0's are in each bit position where 1's are in (SA). If so, set ST2.
CZC	0 0 1 0 0 1	Compare zeros corresponding	No	2	Test (DR) to determine if 0's are in each bit position where 1's are in (SA). If so, set ST2.
XOR	0 0 1 0 1 0	Exclusive OR	Yes	0-2	(DR) ⊕ (SA) → (DR)
MPY	0 0 1 1 1 0	Multiply	No		Multiply unsigned (DR) by unsigned (SA) and place unsigned 32-bit product in DR (most significant) and DR + 1 (least significant). If WR15 is DR, the next word in memory after WR15 will be used for the least significant half of the product.
DIV	0 0 1 1 1 1	Divide	No	4	If unsigned (SA) is less than or equal to unsigned (DR), perform no operation and set ST4. Otherwise divide unsigned (DR) and (DR) by unsigned (SA). Quotient → (DR), remainder → (DR+1). If DR=15, the next word in memory after WR15 will be used for the remainder.

Exclusive OR Logic $1 \oplus 0 = 1$
$0 \oplus 0 = 0$
$1 \oplus 1 = 0$

EXAMPLES

(1) ASSEMBLY LANGUAGE:

```
MPY    R2,R3        MULTIPLY CONTENTS OF R2 AND R3, RESULT IN R3 AND R4
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	1	1	1	0	0	0	1	1	0	0	0	0	1	0	>38C2

	BEFORE	AFTER	
R2	0002	0002	
R3	0003	0000	32-BIT
R4	N	0006	RESULT

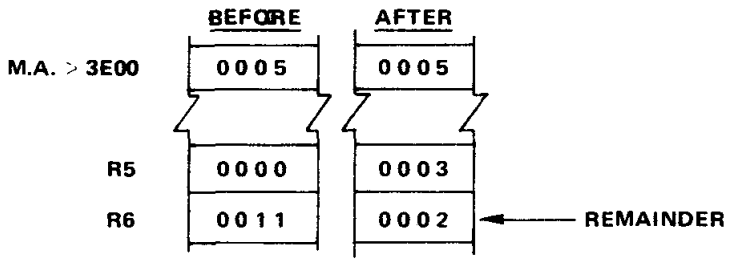
The destination operand is always a register, and the values multiplied are 16-bits, unsigned. The 32-bit result is placed in the destination register and destination register +1, zero filled on the left.

(2) ASSEMBLY LANGUAGE:

```
DIV   @>3E00, R5      DIVIDE CONTENTS OF R5 AND R6 BY VALUE AT M.A. >3E00
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	1	1	1	1	0	1	0	1	1	0	0	0	0	0	>3D60
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	>3E00



The unsigned 32-bit value in the destination register and destination register +1 is divided by the source operand value. The result is placed in the destination register. The remainder is placed in the destination register +1.

(3) ASSEMBLY LANGUAGE:

```
COC   R10,R11       ONES IN R10 ALSO IN R11?
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	1	0	0	0	1	0	1	1	0	0	1	0	1	0	>22CA

Locate all binary ones in the source operand. If the destination operand also has ones in these positions, set the equal flag in the Status Register; otherwise, reset this flag. The following sets the equal flag:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
R10	1	0	1	0	1	0	1	0	0	0	0	0	1	1	0	0	>AA0C
R11	1	1	1	0	1	1	1	1	1	1	0	0	1	1	0	1	>EFCD

Set EQ bit in Status Register to 1.

5.6.4 FORMAT 4 (CRU MULTIBIT) INSTRUCTIONS

General Format:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE						C				T_S		SR			

The C field specifies the number of bits to be transferred. If C = 0, 16 bits will be transferred. The CRU base register (WR 12, bits 4 through 14) defines the starting CRU bit address. The bits are transferred serially and the CRU address is incremented with each bit transfer, although the contents of WR12 are not affected. T_S and SA provide multiple mode addressing capability for the source operand. If 8 or fewer bits are transferred (C = 1 through 8), the source address is a byte address. If 9 or more bits are transferred (C = 0, 9 through 15), the source address is a word (even number) address. If the source is addressed in the workspace register indirect autoincrement mode, the workspace register is incremented by 1 if C = 1 through 8, and is incremented by 2 otherwise.

MNEMONIC	OP CODE 0 1 2 3 4 5	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
LDCR	0 0 1 1 0 0	Load communcation register	Yes	0-2,5†	Beginning with LSB of (SA), transfer the specified number of bits from (SA) to the CRU.
STCR	0 0 1 1 0 1	Store communcation register	Yes	0-2,5†	Beginning with LSB of (SA), transfer the specified number of bits from the CRU to (SA). Load unfilled bit positions with 0.

†ST5 is affected only if $1 \leqslant C \leqslant 8$.

EXAMPLE

ASSEMBLY LANGUAGE:

```
LDCR    @>3E00,8        LOAD 8 BITS ON CRU FROM M.A. >3E00
```

MACHINE LANGUAGE:

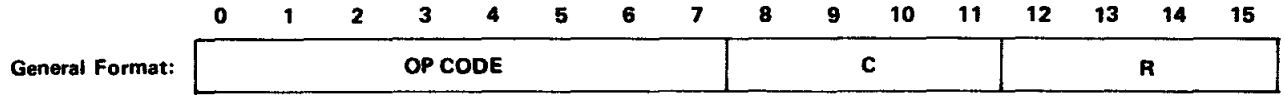
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	>3220
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	>3E00

NOTE

CRU addressing is discussed in detail in Section 7.

5.6.5 FORMAT 5 (SHIFT) INSTRUCTIONS

These instructions shift (left, right, or circular) the bit patterns in a workspace register. The last bit value shifted out is placed in the carry bit (3) of the Status Register. If the SLA instruction causes a one to be shifted into the sign bit, the ST overflow bit (4) is set. The C field contains the number of bits to shift.



If C = 0, bits 12 through 15 of R0 contain the shift count. If C = 0 and bits 12 through 15 of WR0 = 0, the shift count is 16.

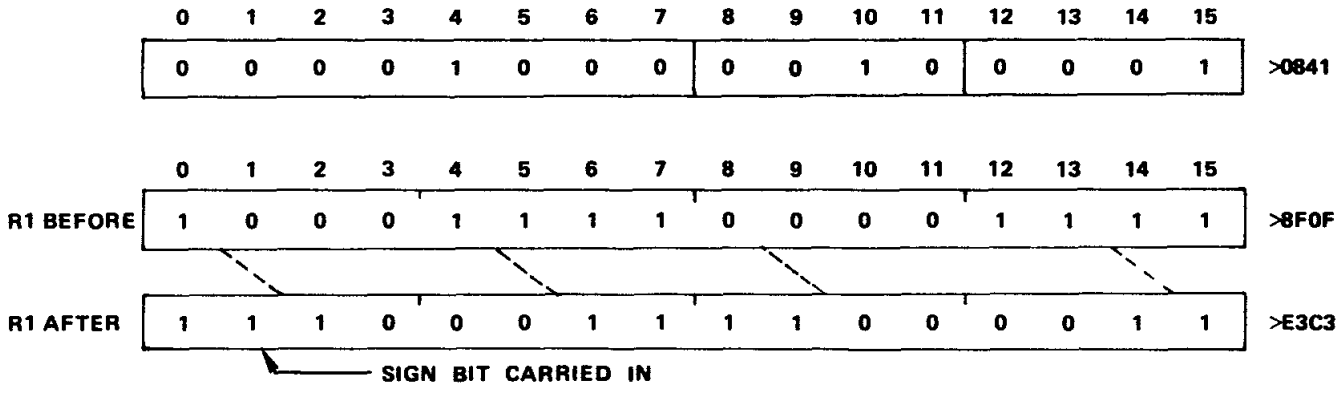
MNEMONIC	OP CODE 0 1 2 3 4 5 6 7	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
SLA	0 0 0 0 1 0 1 0	Shift left arithmetic	Yes	0-4	Shift (R) left. Fill vacated bit positions with 0.
SRA	0 0 0 0 1 0 0 0	Shift right arithmetic	Yes	0-3	Shift (R) right. Fill vacated bit positions with original MSB of (R).
SRC	0 0 0 0 1 0 1 1	Shift right circular	Yes	0-3	Shift (R) right. Shift previous LSB into MSB.
SRL	0 0 0 0 1 0 0 1	Shift right logical	Yes	0-3	Shift (R) right. Fill vacated bit positions with 0's.

EXAMPLES

(1) ASSEMBLY LANGUAGE:

```
SRA    R1,2      SHIFT R1 RIGHT 2 POSITIONS, CARRY SIGN
```

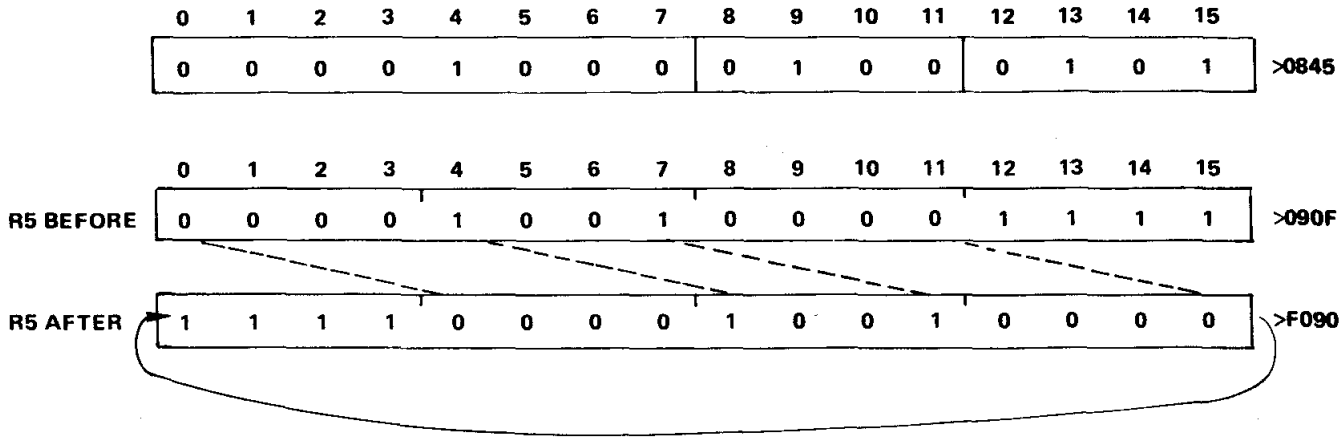
MACHINE LANGUAGE:



(2) ASSEMBLY LANGUAGE:

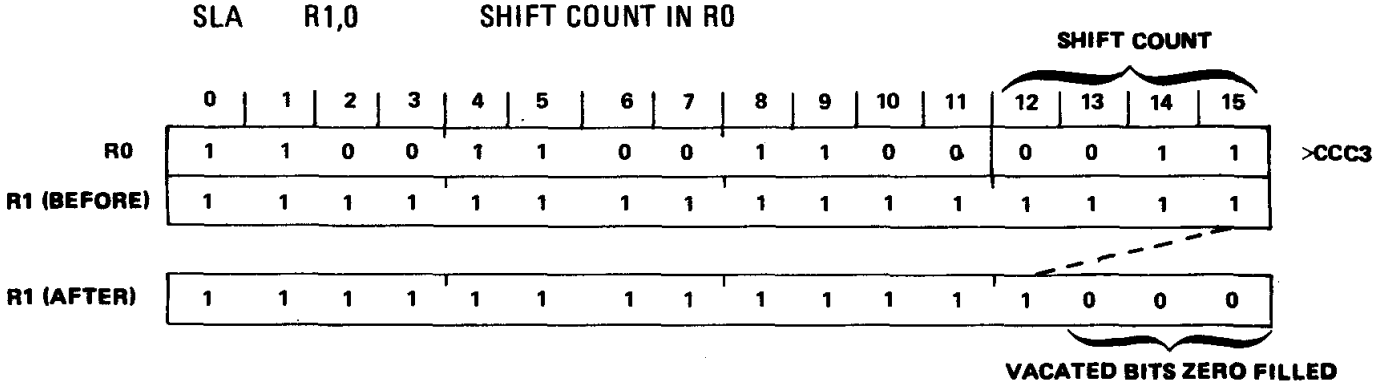
SRC R5,4 CIRCULAR SHIFT R5 4 POSITIONS

MACHINE LANGUAGE:



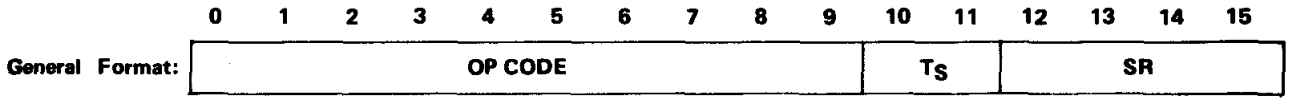
(3) ASSEMBLY LANGUAGE:

SLA R1,0 SHIFT COUNT IN R0



5.6.6 FORMAT 6 INSTRUCTIONS

These are single operand instructions.



The T_S and S fields provide multiple mode addressing capability for the source operand.

MNEMONIC	OP CODE 0 1 2 3 4 5 6 7 8 9	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
B	0 0 0 0 0 1 0 0 0 1	Branch	No	–	SA → (PC)
BL	0 0 0 0 0 1 1 0 1 0	Branch and link	No	–	**(PC) → (R11); SA → (PC)**
BLWP	0 0 0 0 0 1 0 0 0 0	Branch and load workspace pointer	No	–	**(SA) → (WP); (SA+2) → (PC);** (old WP) → (new WR13); (old PC) → (new WR14); (old ST) → (new WR15); the interrupt input ($\overline{\text{INTREQ}}$) is not tested upon completion of the BLWP instruction.
CLR	0 0 0 0 0 1 0 0 1 1	Clear operand	No	–	**0000 → (SA)**
SETO	0 0 0 0 0 1 1 1 0 0	Set to ones	No	–	$FFFF_{16}$ → (SA)
INV	0 0 0 0 0 1 0 1 0 1	Invert	Yes	0-2	$\overline{(SA)}$ → (SA) **(ONE'S complement)**
NEG	0 0 0 0 0 1 0 1 0 0	Negate	Yes	0-4	–(SA) → (SA) **(TWO'S complement)**
ABS	0 0 0 0 0 1 1 1 0 1	**Absolute value***	No	0-4	**[(SA)] → (SA)**
SWPB	0 0 0 0 0 1 1 0 1 1	Swap bytes	No	–	(SA), bits 0 thru 7 → (SA), bits 8 thru 15; (SA), bits 8 thru 15 → (SA), bits 0 thru 7.
INC	0 0 0 0 0 1 0 1 1 0	Increment	Yes	0-4	(SA) + 1 → (SA)
INCT	0 0 0 0 0 1 0 1 1 1	Increment by two	Yes	0-4	(SA) + 2 → (SA)
DEC	0 0 0 0 0 1 1 0 0 0	Decrement	Yes	0-4	(SA) – 1 → (SA)
DECT	0 0 0 0 0 1 1 0 0 1	Decrement by two	Yes	0-4	(SA) – 2 → (SA)
X†	0 0 0 0 0 1 0 0 1 0	Execute	No	–	Execute the instruction at SA.

***Operand is compared to zero for setting the status bit (i.e., before execution).**

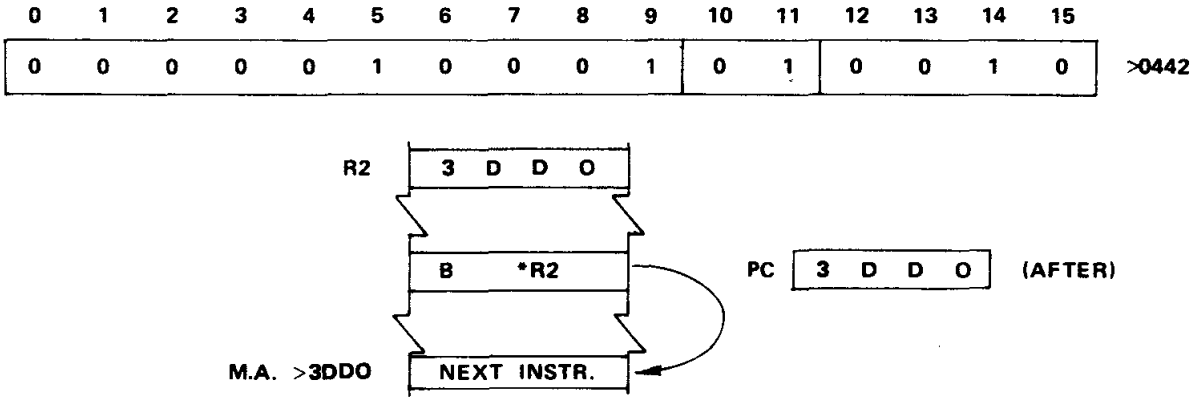
†If additional memory words for the execute instruction are required to define the operands of the instruction located at SA, these words will be accessed from PC and the PC will be updated accordingly. The instruction acquisition signal (IAQ) will not be true when the TMS 9900 accesses the instruction at SA. Status bits are affected in the normal manner for the instruction executed.

EXAMPLES

(1) ASSEMBLY LANGUAGE:

```
B    *R2      BRANCH TO M.A. IN R2
```

MACHINE LANGUAGE:

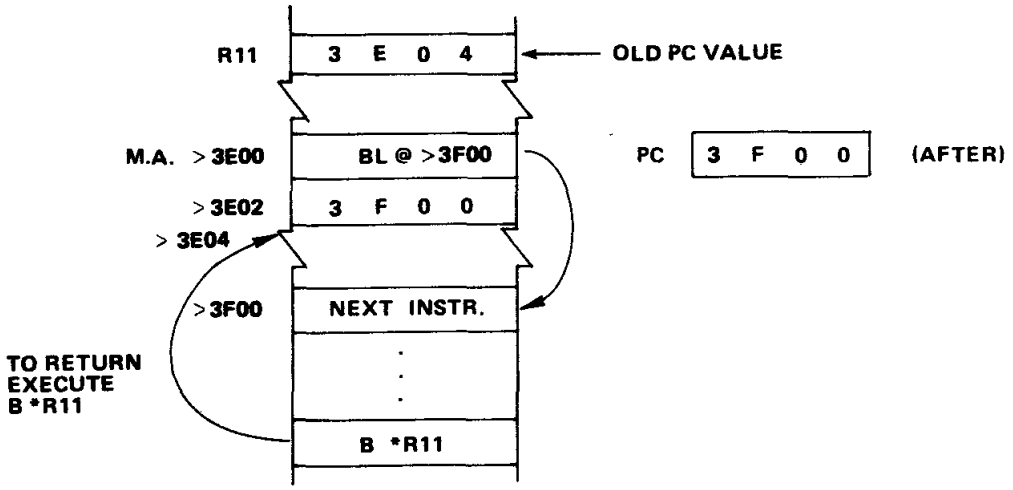


(2) *ASSEMBLY LANGUAGE:*

BL @ >3F00 BRANCH TO M.A. >3F00, SAVE OLD PC VALUE (AFTER EXECUTION) IN R11

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	0	>04A0
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	>3F00



(3) *ASSEMBLY LANGUAGE:*

BLWP @ >3F00 BRANCH, GET NEW WORKSPACE AREA

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	>0420
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	>3F00

This context switch provides a new workspace register file and stores return values in the new workspace. See Figure 5-11. The operand (> 3F000 above) is the M.A. of a two-word transfer vector, the first word the new WP value, the second word the new PC value.

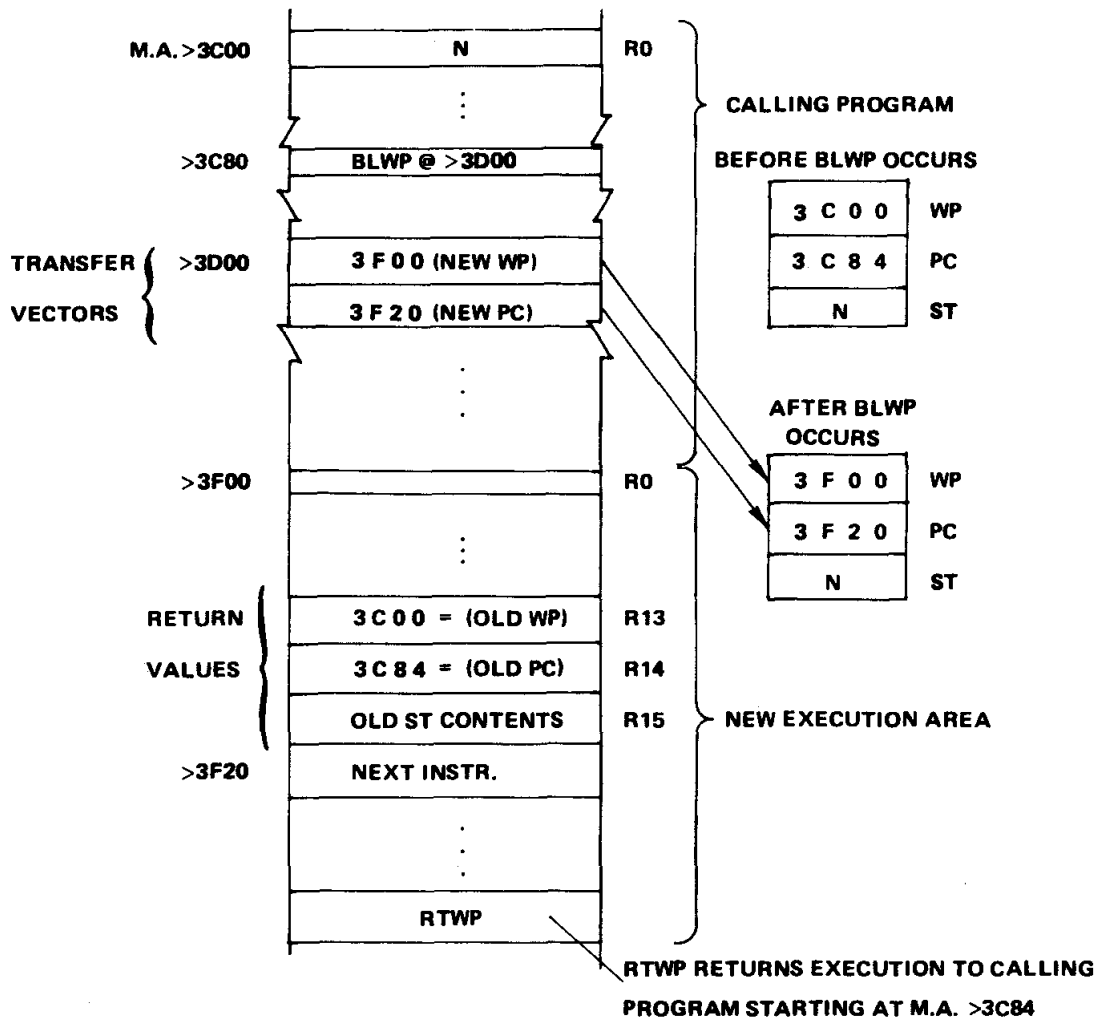
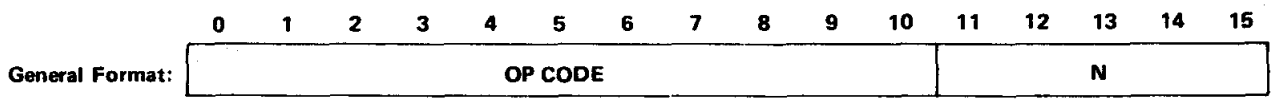


Figure 5-11. BLWP Example

5.6.7 FORMAT 7 (RTWP, CONTROL) INSTRUCTIONS



External instructions cause the three address lines (A13, A0, A1) to be set to the levels described in the table below and cause the CRUCLK line to be pulsed, allowing external control functions to be interpreted during CRUCLK at A13, A0, and A1. The RSET instruction resets the I/O lines on the TMS 9901 to input lines; the TMS 9902 is not affected. RSET also clears the interrupt mask in the Status Register. The LREX instruction causes a delayed load interrupt, delayed by two IAQ cycles after LREX execution. The load operation gives control to the monitor.

MNEMONIC	OP CODE 0 1 2 3 4 5 6 7 8 9 10	MEANING	STATUS BITS AFFECTED	DESCRIPTION	ADDRESS BUS* A13 A0 A1
IDLE	0 0 0 0 0 0 1 1 0 1 0	Idle	–	Suspend TMS 9980 instruction execution until an interrupt, $\overline{\text{LOAD}}$, or $\overline{\text{RESET}}$ occurs	L H L
RSET	0 0 0 0 0 0 1 1 0 1 1	Reset I/O & SR*	12–15	0 → ST12 thru ST15, RESET	L H H
CKOF	0 0 0 0 0 0 1 1 1 1 0	User defined		---	H H L
CKON	0 0 0 0 0 0 1 1 1 0 1	User defined		---	H L H
LREX	0 0 0 0 0 0 1 1 1 1 1	Load interrupt		Control to *UNIBUG*	H H H
RTWP	0 0 0 0 0 0 1 1 1 0 0	Return from Subroutine	0–15	(R13) → (WP) (R14) → (PC) (R15) → (ST)	

* This instruction causes the interrupt mask in the TMS 9980 to be zeroed, generates a signal to reset the I/O devices (except 9902's), and also traps to location 0000_{16} (causes a level 0 or RESET interrupt).

Essentially, the RTWP instruction is a return to the next instruction that follows the BLWP instruction (i.e., RTWP is a return from a BLWP context switch, similar to the B *R11 return from a BL instruction). BLWP provides the necessary values in registers 13, 14, and 15 (see Figure 5-11.)

EXAMPLE

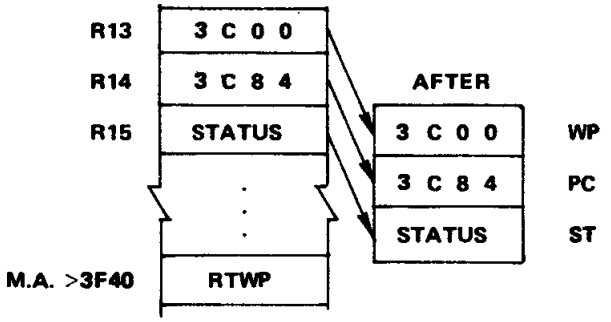
ASSEMBLY LANGUAGE:

```
RTWP      RETURN FROM CONTEXT SWITCH
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	>0380

RTWP RETURN TO PREVIOUS PC(R14), WP(R13), AND ST(R15) VALUES

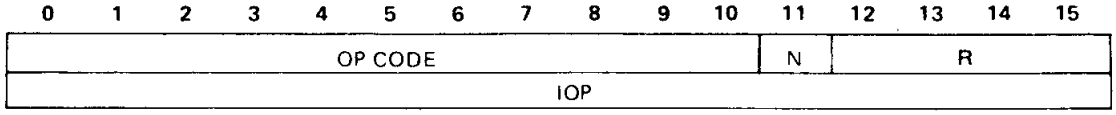


EXECUTION BEGINS AT M.A. >3C84
WITH R0 AT M.A. >3C00.

5.6.8 FORMAT 8 (IMMEDIATE, INTERNAL REGISTER LOAD/STORE) INSTRUCTIONS

5.6.8.1 Immediate Register Instructions

General format:



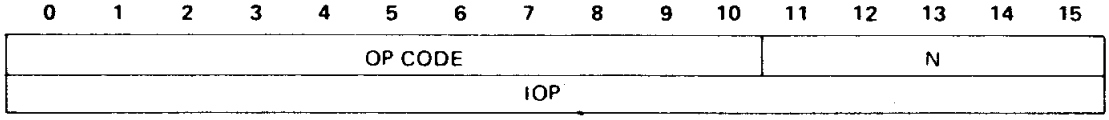
MNEMONIC	OP CODE (0 1 2 3 4 5 6 7 8 9 10)	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
AI	0 0 0 0 0 0 1 0 0 0 1	Add immediate	Yes	0-4	(R) + IOP → (R)
ANDI	0 0 0 0 0 0 1 0 0 1 0	AND immediate	Yes	0-2	(R) AND IOP → (R)
CI	0 0 0 0 0 0 1 0 1 0 0	Compare immediate	Yes	0-2	Compare (R) to IOP and set appropriate status bits
LI	0 0 0 0 0 0 1 0 0 0 0	Load immediate	Yes	0-2	IOP → (R)
ORI	0 0 0 0 0 0 1 0 0 1 1	OR immediate	Yes	0-2	(R) OR IOP → (R)

AND Logic: 0·1, 1·0 = 0; 0·0 = 0; 1·1 = 1

OR Logic: 0 + 1, 1 + 0 = 1; 0 + 0 = 0; 1 + 1 = 1

5.6.8.2 Internal Register Load Immediate Instructions

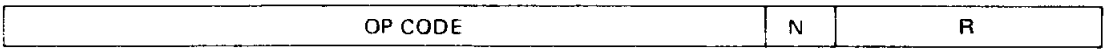
General format:



MNEMONIC	OP CODE (0 1 2 3 4 5 6 7 8 9 10)	MEANING	DESCRIPTION
LWPI	0 0 0 0 0 0 1 0 1 1 1	Load workspace pointer immediate	IOP → (WP), no ST bits affected
LIMI	0 0 0 0 0 0 1 1 0 0 0	Load interrupt mask	IOP, bits 12 thru 15 → ST12 thru ST15

5.6.8.3 Internal Register Store Instructions

General format:



No ST bits are affected.

MNEMONIC	OP CODE (0 1 2 3 4 5 6 7 8 9 10)	MEANING	DESCRIPTION
STST	0 0 0 0 0 0 1 0 1 1 0	Store status register	(ST) → (R)
STWP	0 0 0 0 0 0 1 0 1 0 1	Store workspace pointer	(WP) → (R)

EXAMPLES

(1) ASSEMBLY LANGUAGE:

```
AI    R2,>FF      ADD >FF TO CONTENTS OF R2
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	>0222
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	>00FF

	BEFORE	AFTER
R2	0 0 0 F	0 1 0 E

(2) ASSEMBLY LANGUAGE:

```
CI    R2,>10E     COMPARE R2 TO >10E
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	>0282
0	0	0	0	0	0	0	1	0	0	0	0	1	1	1	0	>010E

R2 contains "after" results (>10E) of instruction in Example (1) above; thus the ST equal bit becomes set.

(3) ASSEMBLY LANGUAGE:

```
LWPI    >3E00       WP SET AT >3E00 (M.A. OF R0)
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	0	0	0	1	0	1	1	1	0	0	0	0	0	>02E0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	>3E00

This is used to define the workspace area in a task, usually placed at the beginning of a task.

(4) ASSEMBLY LANGUAGE:

```
STWP    R2          STORE WP CONTENTS IN R2
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	>02A2

This places the M.A. of R0 in a workspace register.

5.6.9 FORMAT 9 (XOP) INSTRUCTION

Other format 9 instructions (MPY, DIV) are explained in paragraph 5.6.3 (format 3).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
General Format:	0	0	1	0	1	1	DR (XOP NUMBER)				T_S		SR			

The T_S and SR fields provide multiple mode addressing capability for the source operand. When the XOP is executed, ST6 is set and the following transfers occur:

$(40_{16} + 4D) \rightarrow (WP)$ — First vector at 40_{16}
$(42_{16} + 4D) \rightarrow (PC)$ — Each vector uses 4 bytes (2 words)
SA → (new R11)
(old WP) → (new WR13)
(old PC) → (new WR14)
(old ST) → (new WR15)

The TMS 9980 does not test interrupt request (INTREQ) upon completion of the XOP instruction, but does test for RESET and LOAD.

An XOP is a means of calling one of 16 subtasks available for use by any executing task. The memory area between M.A. 40_{16} and $7E_{16}$ is reserved for the transfer vectors of XOP's 0 to 15 (see Figure 5-1). Each XOP vector consists of two words, the first a WP value, the second a PC value, defining the workspace pointer and entry point for a new subtask. These values are placed in their respective hardware registers when the XOP is executed.

The old WP, PC, and ST values (of the XOP calling task) are stored (like the BLWP instruction) in the new workspace, registers 13, 14, and 15. Return to the calling routine is through the RTWP instruction. Also stored, in the new R11, is the M.A. of the source operand. This allows passing a parameter to the new subtask, such as the memory address of a string of values to be processed by the XOP-called routine. Figure 5-12 depicts calling an XOP to process a table of data; the data begins at M.A. $3F00_{16}$.

XOP's 8 to 14 are used by the UNIBUG monitor, calling software routines (supervisor calls) as requested by tasks. This user-accessible software performs tasks such as write to terminal, convert binary to hex ASCII, etc. These monitor XOP's are discussed in Section 3.

ASSEMBLY LANGUAGE:

```
XOP     @ >3F00,4
```

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	1	0	1	1	0	1	0	0	1	0	0	0	0	0	>2D20
0	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0	>3F00

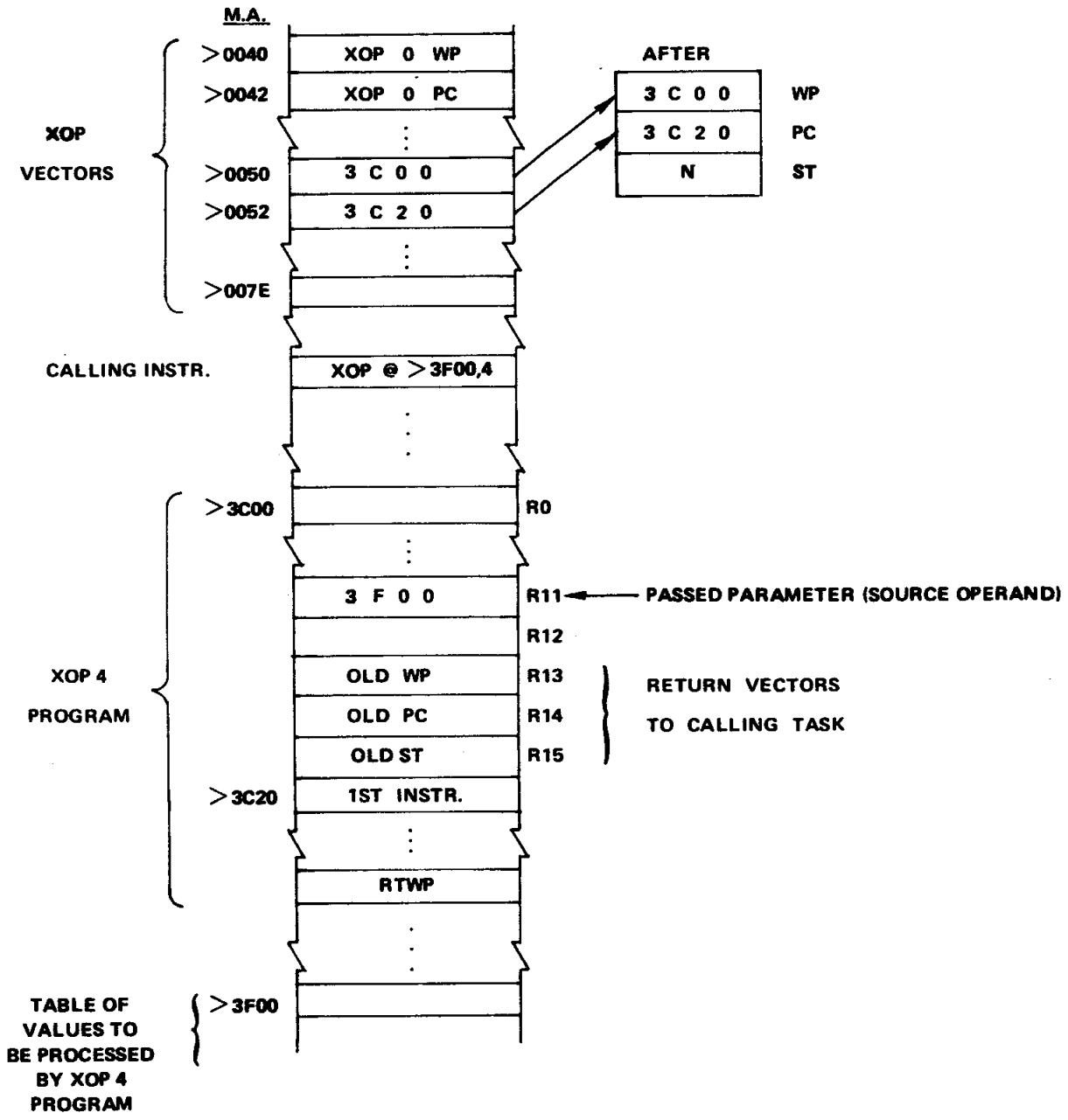


Figure 5-12. XOP Example

5.7 COMPARISON OF JUMPS, BRANCHES AND XOPS

See Table 5-7.

TABLE 5-7. COMPARISON OF JUMPS, BRANCHES, XOP'S

Mnemonic	Paragraph	Definition Summary
JMP	5.6.2	One-word instruction, destination restricted to +127, −128 words from Program. Counter value.
B	5.6.6	Two-word instruction, branch to any memory location.
BL	5.6.6	Same as B with PC return address in R11.
BLWP	5.6.7	Same as B with new workspace; old WP, PC and ST contents (return vectors) are in new R13, R14, R15.
XOP	5.6.9	Same as BLWP with address of parameter (source operand) in new R11. Sixteen XOP vectors outside program in M.A. 40_{16} to $7E_{16}$; can be called by any program.

SECTION 6

ASSEMBLER DIRECTIVES

6.1 GENERAL

This section defines the following six assembler directives recognized by the Symbolic Assembler (described in Section 4). These directives and corresponding paragraph number are:

- AORG Absolute origin of statement (absolute location) 6.2.1
- BSS Block of memory starting with symbol 6.2.2
- DATA Sixteen-bit immediate value 6.2.3
- END End of source code 6.2.4
- EQU Label equated to symbol or value 6.2.5
- TEXT Code character string in ASCII code 6.2.6

6.2 DIRECTIVE FORMATS

Syntax used in this subsection:

< > Required items to be supplied by the user

[ ] Optional items to be supplied by the user

T1 Space
T2 Space or comma
T3 Space, comma, or return

6.2.1 AORG DIRECTIVE

Format:

[label] < T1 > < AORG > <T1 > < location > <T1 > [comment]

The AORG directive places a value in the location counter and begins assembly at the location specified. The location value must be in decimal or hexadecimal. By default, the location counter for the assembler begins at 0000_{16} and is incremented by two (bytes) for each word occupied by the instruction. When a label is used with the AORG directive, it is assigned the value in the location counter. Comment field is optional.

Example:

AORG > 200 Begin assembling source code at location counter value of > 200

AORG 200 Begin assembling source code at location counter value of > C8

6.2.2 BSS DIRECTIVE

Format:

[label] < T1 > <BSS > < T1 > < number of bytes > < T1 > [comment]

The BSS (block with starting symbol) directive advances the location counter (which the assembler uses to count the bytes of machine code) a quantity of bytes as specified in the directive. In essence, it "reserves" a block of bytes starting at the location counter value; this block will be void of object code. An optional label (in the label field) can be specified to identify the first location in the block. The byte count can be in decimal or hexadecimal.

6.2.3 DATA DIRECTIVE

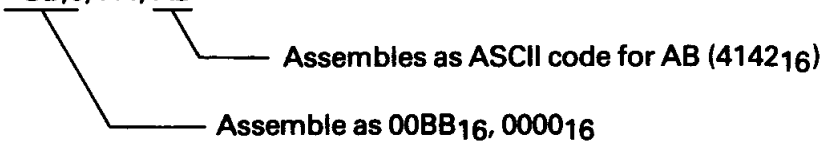
Format:

[label] < T1 > < DATA > < T1 > < 1-16 bit value ,.., 1-16 bit value > < T1 > [comment]

This directive places 16 bit values into (successive) memory locations. Data is placed at even address locations. Operand values can be chained (i.e., successive 1 to 16 bit values separated by commas).

Example:

DATA >FFFF,1764, >BB,0,444,'AB'



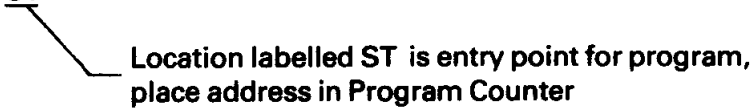
6.2.4 END DIRECTIVE

Format:

[label] < T1 > < END > < T1 > [entry point] < T1 > [comment]

This directive is mandatory for each program. It designates to the assembler that this is the final input from the source program and causes a transfer of control back to the monitor. This is the last input to the assembler, and the only means of direct transfer from the assembler to the monitor. When the optional label is used, it is assigned the current value in the location counter. The optional load-point operand field contains a symbol or absolute memory address specifying the entry point (execution start) of the program. When the entry-point operand is used, the entry point address will be placed in the Program Counter so that the program can be executed by the E command immediately after being loaded. Example:

END ST



6.2.5 EQU DIRECTIVE

Format:

< label > < T1 > < EQU > < T1 > < value or expression > < T1 > [comment]

This directive assigns a value to a label for use during assembly. The expression field can contain an absolute numeric value or expression. Expressions are further defined in paragraph 4.2.4. This directive allows the user to substitute easily remembered mnemonics for absolute values in program source lines. The optional label will be assigned the current value in the location counter.

Examples:

1. SM EQU 1 SUM ACCUMULATOR
   allows using SM for register 1 such as
   MOV @>FC00,SM MOVE QTY TO R1
   instead of
   MOV @>FC00,1 MOVE QTY TO R1

2. IN EQU 9681 DIVISOR CONSTANT
   allows this constant value to be used in subsequent source lines
   LI R1,IN PLACE CONSTANT IN R1
   instead of remembering the constant value.

3. If IN has been previously defined as above, the following
   MOV @IN+4,@OT
   will result in moving the value located four bytes beyond location IN into location OT.

   A label can be equated to a string of labels being added or subtracted (expression). For example:

4. A EQU 5
   B EQU 10
   C EQU A+B EQUALS VALUE 15
   NO DATA C+A EQUALS VALUE 20
   would result in the value 20 in location NO.

6.2.6 TEXT DIRECTIVE

Format:

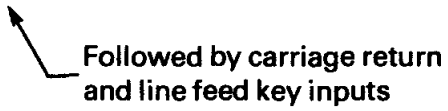
[label] < T1 > < TEXT > < T1 > [−,] < 'character string' > < T1 > [comment]

This directive, like the DATA directive, is used to generate absolute data for program use. The DATA statement operand is interpreted as a numerical value. The TEXT statement operand contains an alphanumeric character string of keyboard inputs which are to be interpreted into ASCII code. Besides keyboard characters, the user can also input control characters (e.g., carriage return, line feed, DC1, DC2) which are output in ASCII code via the keyboard. ASCII code is defined in Appendix C. Character string inputs in the operand field are enclosed in single quotes (SHIFT/Ret). The assembler begins all character strings on an even boundary and places a zero byte after the last character that can be used as a delimiter by the XOP I/O commands.

The optional label field will be assigned the value in the location counter; this value will identify the location of the first character in the string.

Examples:

1. CM TEXT 'LOAD TAPE, HIT CR_ _'



2. CM TEXT 'LOAD TAPE, HIT CR.'

SECTION 7

I/O PROGRAMMING USING THE CRU

7.1 GENERAL

The Communications Register Unit (CRU) is a separate I/O port, used only for data transfers to or from external devices. It is the medium through which the microprocessor communicates with the outside world. The CRU is an internal port of the TMS 9980A microprocessor.

An interface is a common boundary between automatic data-processing systems or parts of a single system. In the TM 990/189, an interface will be needed between the microprocessor and an I/O device (e.g., terminal or keyboard). The TM 990/189 provides circuitry for two types of Texas Instruments interface devices: two onboard TMS 9901 Programmable Systems Interfaces (PSI's) and an optional TMS 9902 Asynchronous Communications Controller (ACC).

The TMS 9902 ACC can be used as an interface between the TMS 9980A and a terminal (such as those in Texas Instruments' Silent 700 series). Figure 7-1 shows a system using the optional TMS 9902. Figure 7-2 shows the TMS 9901 PSI used as an interface between the keyboard and the TMS 9980A. An additional TMS 9901 is provided onboard as a user I/O port.

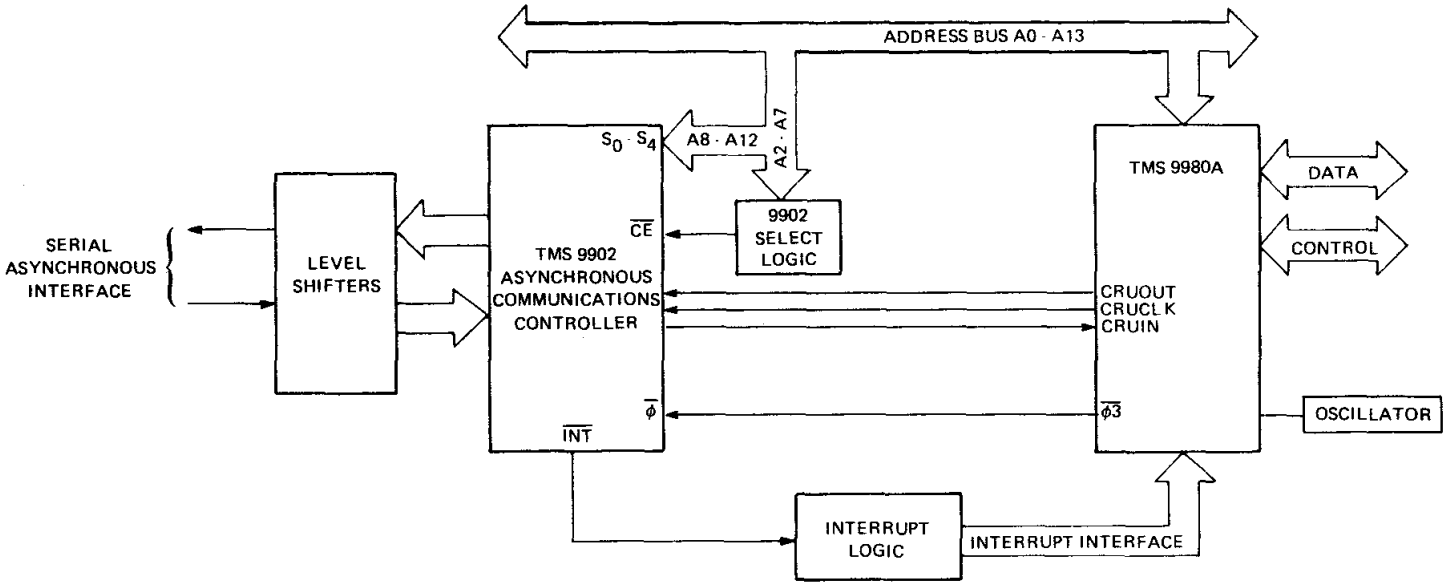


Figure 7-1. Typical Application TMS 9902 Asynchronous Communication Controller (ACC)

Careful examination of Figures 7-1 and 7-2 will reveal three CRU control lines between the interface and the microprocessor. These lines are used for serial data input (CRUIN), serial data output (CRUOUT), and data timing strobes (CRUCLK). In order to transfer data in or out, CRU programming must be understood. When CRU instructions are executed, data is written or read through the CRUOUT or CRUIN pins respectively of the TMS 9980A. These signals come from or are sent to designated devices selected by decode logic attached to the address bus of the microprocessor. CRUOUT is also address line A13, it is not used to designate a CRU address.

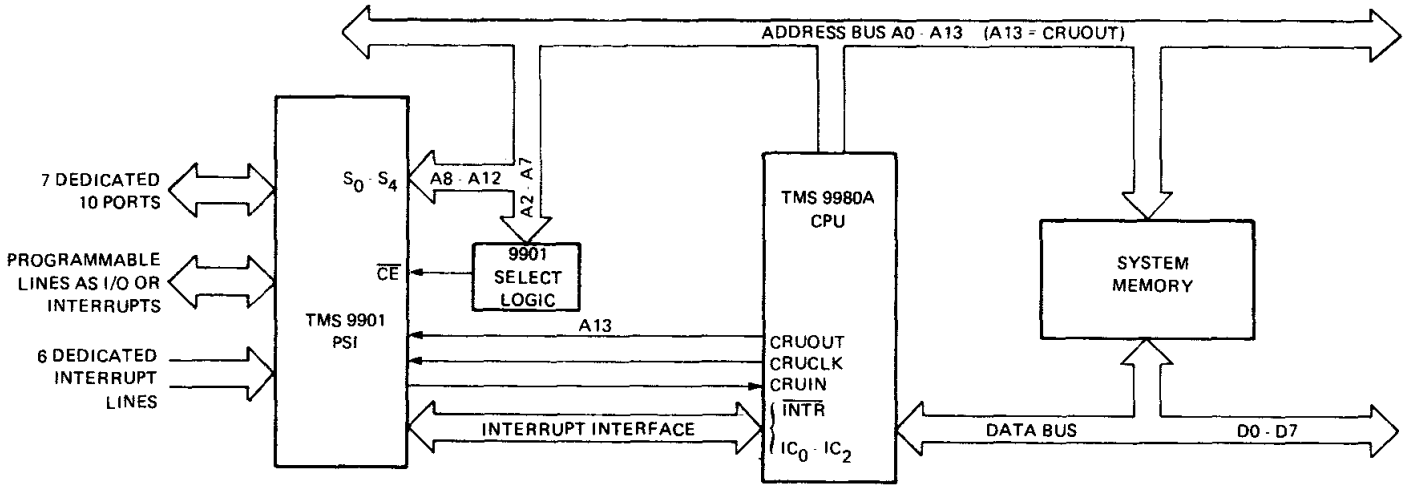


Figure 7-2. Typical TMS 9901 Programmable System Interface (PSI) Application

Essentially, the CRU process follows this sequence:

1. The CRU instruction is executed.
2. Bit address of the desired external device is placed on the address bus.
3. Decode logic on the address bus enables an external device so that it can serially send or receive using the CRU input or output lines and clock.
4. Bits are serially sent or received over the CRU lines.

The CRU address is loaded by software and maintained in register 12 of the workspace register area. Only bits 4 through 14 of the register are interpreted by the CPU for the desired CRU address, and this 11-bit value is called the CRU base address or bit number.

As mentioned before, the TM 990/189 drives (via the CRU port) the onboard TMS 9901 parallel interface and the optional TMS 9902 serial interface. These interfaces are accessed (enabled) through the CRU addresses noted in Table 7-1. This table also lists the functions of the other CRU addresses which can be used for on-card or off-card I/O use.

The TMS 9901 and the TMS 9902 interfaces can be used as interval timers, further detailed information on these two devices can be obtained from their respective data manuals.

TABLE 7-1. CRU ADDRESS MAP

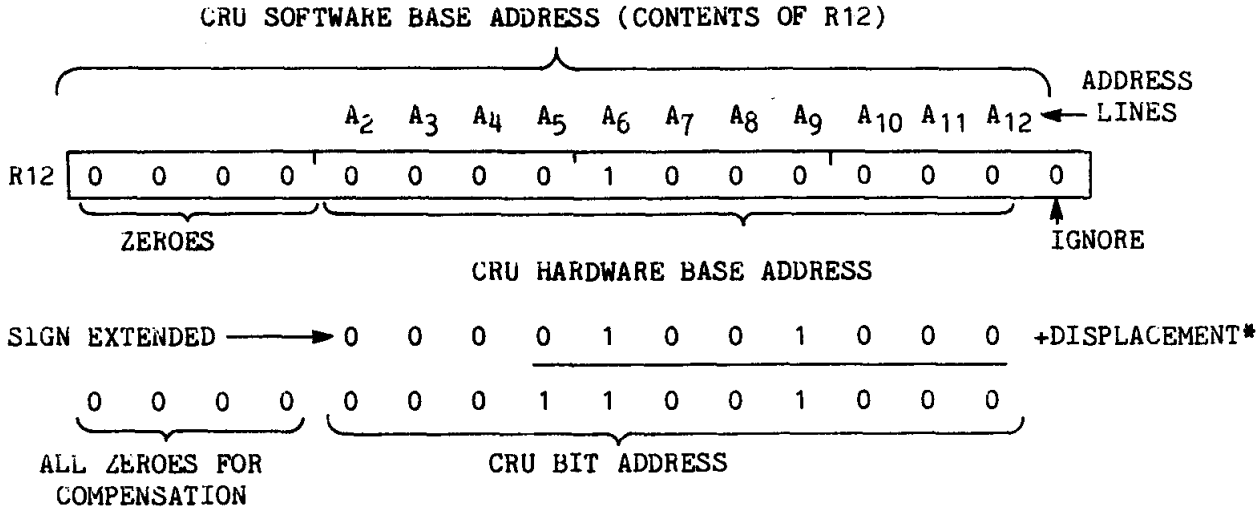
Contents of R12 (Bits 0 to 15)	CRU Base Address (R12, Bits 4 to 14)	Function
0000 to 003E	0000 to 001F	User TMS 9901 programmable interface at U10 (uses port P5)
0400 to 043E	0200 to 021F	System TMS 9901 programmable interface at U11
0800 to 083E	0400 to 041F	System TMS 9902 asynchronous interface (EIA interface using port P3)
0C00 to 0C3E	0600 to 061F	User CRU addresses*

*Expandable to 512 bits, see Section 9.

Paragraph 7-2 explains CRU addressing while timing is covered in paragraph 7.3. Paragraph 7-4 describes the five CRU instructions.

7.2 CRU ADDRESSING

The CRU software base address is contained in the 16 bits of register 12. From the CRU software base address, the processor is able to determine the CRU hardware base address and the resulting CRU bit address. These three CRU addressing forms are shown in Figure 7-3.



*The displacement added to the CRU hardware base address is a signed eight-bit value, with sign extended, used only when executing one of the single-bit instructions (TB, SBO, and SBZ). Shown above is an example displacement of 48_{16} (72_{10}) added to an example hardware base address of 040_{16} (64_{10}).

Figure 7-3. CRU Base and Bit Addresses

7.2.1 CRU BIT ADDRESS

The CRU bit address is the address that will be placed on the address bus at the beginning of a CRU instruction. This is the address bus value that, when decoded by hardware attached to the address bus, will enable the device so that it can be driven by the CRU control and clock lines. The CRU bit address is the sum of the displacement value of the CRU instruction (displacement applies to instructions TB, SBO, and SBZ only) and the CRU hardware base address in bits 4 to 14 of register 12. Note that the sign bit of the eight-bit value is extended to the left and added as part of the displacement. The resulting CRU bit address will be placed on address lines A_2 to A_{12}; address lines A_0 and A_1 always will be zeroes.

7.2.2 CRU HARDWARE BASE ADDRESS

The CRU hardware base address is the value in bits 4 to 14 of register 12. For instructions that do not specify a displacement (the LCDR and STCR do not), the CRU hardware base address is the same as the CRU bit address on address lines A_2 to A_{12} as explained in paragraph 7.2.1. An important aspect of the CRU hardware base address is that it does not use the least significant bit of register 12 (bit 15); this bit is ignored in deriving the CRU bit address.

7.2.3 CRU SOFTWARE BASE ADDRESS

The CRU software base address is the entire 16-bit contents of register 12. In essence, this is the CRU hardware base address divided by two. Bits 0, 1, 2, 3, and 15 of the CRU software base address are ignored in deriving the CRU hardware base address and the CRU bit address.

Because bit 15 of R12 is not used, some confusion can result in programming. Instead of loading the CRU base address in bits 0 to 15 of Register 12 (e.g., LI R12, 400 to address the TMS 9902 at CRU base address 400_{16}), the programmer must shift the base address value one bit to the left so that it is in bits 4 to 14 instead of in bits 5 to 15. Several programming methods can be used to ensure this correct placement. All of the following examples place the TMS 9902 base address of 400_{16} correctly in R12.

```
LI    R12, >800        PLACES 400 IN BITS 3 TO 14
          or
LI    R12, >400*2      MULTIPLY BASE ADDRESS BY 2 (NOT
                       RECOGNIZED BY THE TM 990/189 ASSEMBLER)
          or
LI    R12, >400        BASE ADDRESS IN BITS 4 TO 15
SLA   R12, 1           SHIFT BASE ADDRESS ONE BIT TO THE LEFT
```

From a programming standpoint, it may be best to view addressing of the CRU through the entire 16 bits of R12. In this context, blocks of a maximum of 16 CRU bits can be addressed, and in order to address an adjacent 16-bit block, a value of 0020_{16}, CRU bits 0 to F_{16} can be addressed. By adding 0020_{16} to R12, CRU bits 10_{16} to $1F_{16}$ can be addressed, etc.

7.3 CRU TIMING

Timing during the execution of a CRU output instruction (e.g., LDCR) and a CRU input instruction (e.g., STCR) is shown in Figure 7-4.

In a CRU output operation, the CRU base address in bits 4 to 14 of R12 is valid on the address bus (lines A_2 to A_{12}) when signal $\phi 3$ (inverted phase $\phi 3$) is active. At the next phase $\phi 3$, CRUCLK is active; at this same time, the data at A_{13} (CRUOUT) is true. When $\phi 3$ becomes active again, the next consecutive CRU address (as in an LDCR operation) is active on address lines A_2 to A_{12}, and at the next $\phi 3$ phase, CRUCLK is active again. The cycle repeats itself until the designated number of CRU bits are output (made available) at A_{13}.

Thus the output of data using the CRU involves making the desired CRU address available on address lines A_2 to A_{12}, and sampling the logic level at A_{13} (during CRUCLK).

In a CRU input operation, the desired address is placed on address lines A_2 to A_{12}. The next time $\phi 3$ becomes active, data will be sampled at the CRUIN line. Data should remain valid for two clock phases beginning with $\phi 3$ becoming active.

7.4 CRU INSTRUCTIONS

The five instructions that program the CRU interface are:

- LDCR: Place the CRU base address on address lines A_2 to A_{12}. Load from memory a pattern of 1 to 16 bits and serially transmit this pattern through the CRUOUT pin of the TMS 9980. Increment the address on A_2 to A_{12} after each CRUOUT transmission.

- STCR: Place the CRU base address on address lines A_2 to A_{12}. Store into memory a pattern of 1 to 16 bits obtained serially at the CRUIN pin of the TMS 9980. Increment the address on A_2 to A_{12} after each CRUIN sampling.

- SBO: Place the CRU base address plus the instruction signed displacement on address lins A_2 to A_{12}. Send a logical one through the CRUOUT pin of the TMS 9980.

- SBZ: Place the CRU base address plus the instruction signed displacement on address lines A_2 to A_{12}. Send a logical zero through the CRUOUT pin of the TMS 9980.

- TB: Place the CRU base address plus the instruction signed displacement on address lines A_2 to A_{12}. Test the value at the CRUIN pin of the TMS 9980, and reflect the test results in the equal bit of the Status Register (one or zero).

The LDCR and STCR instructions use a byte or word of memory depending respectively if 1 to 8 bits or more than 8 bits are to be loaded or stored. In STCR instructions, the right bits of the memory area are used for storage, and unused left-side bits are zero filled. Figure 7-4 depicts an LDCR instruction using a byte of memory. Figure 7-6 depicts an STCR instruction using a word of memory.

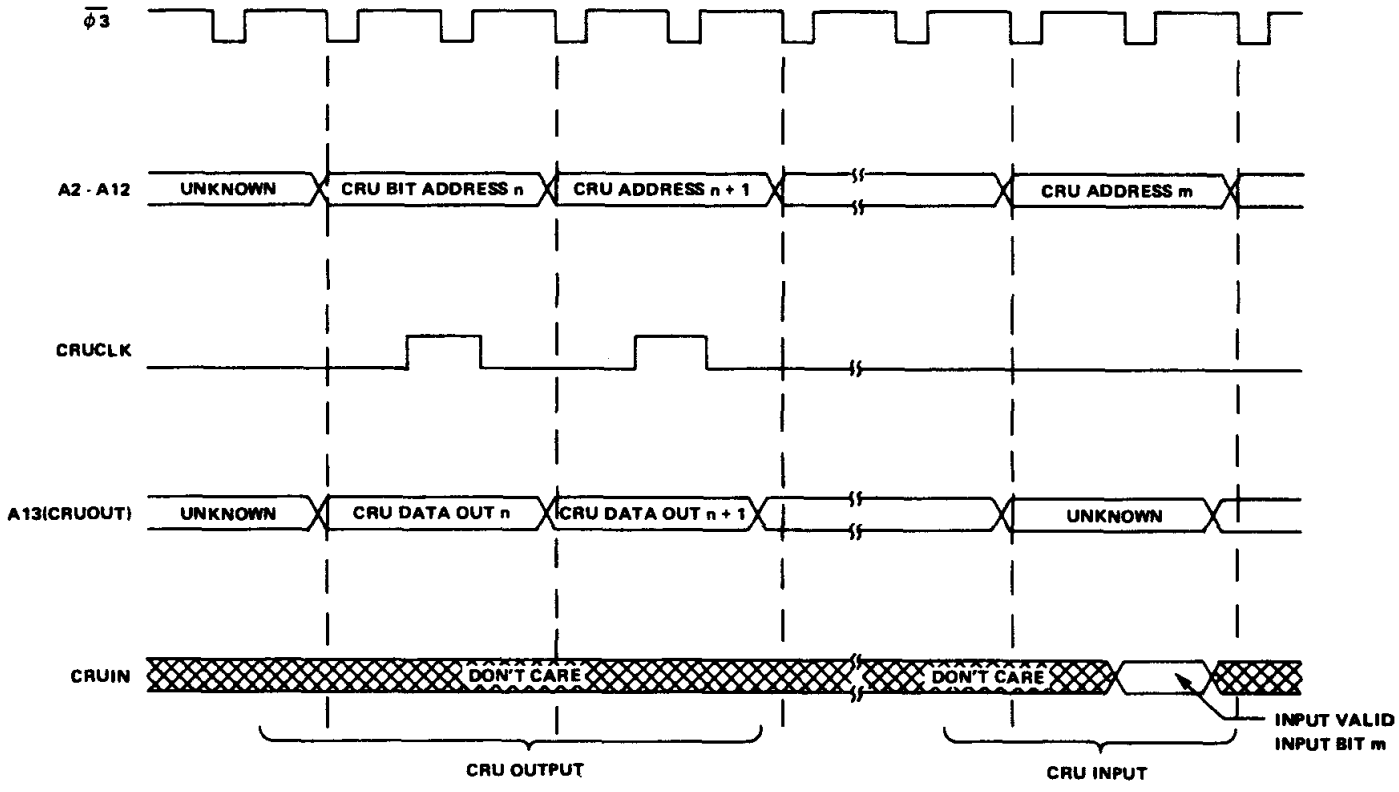


Figure 7-4. TMS 9980 CRU Interface Timing

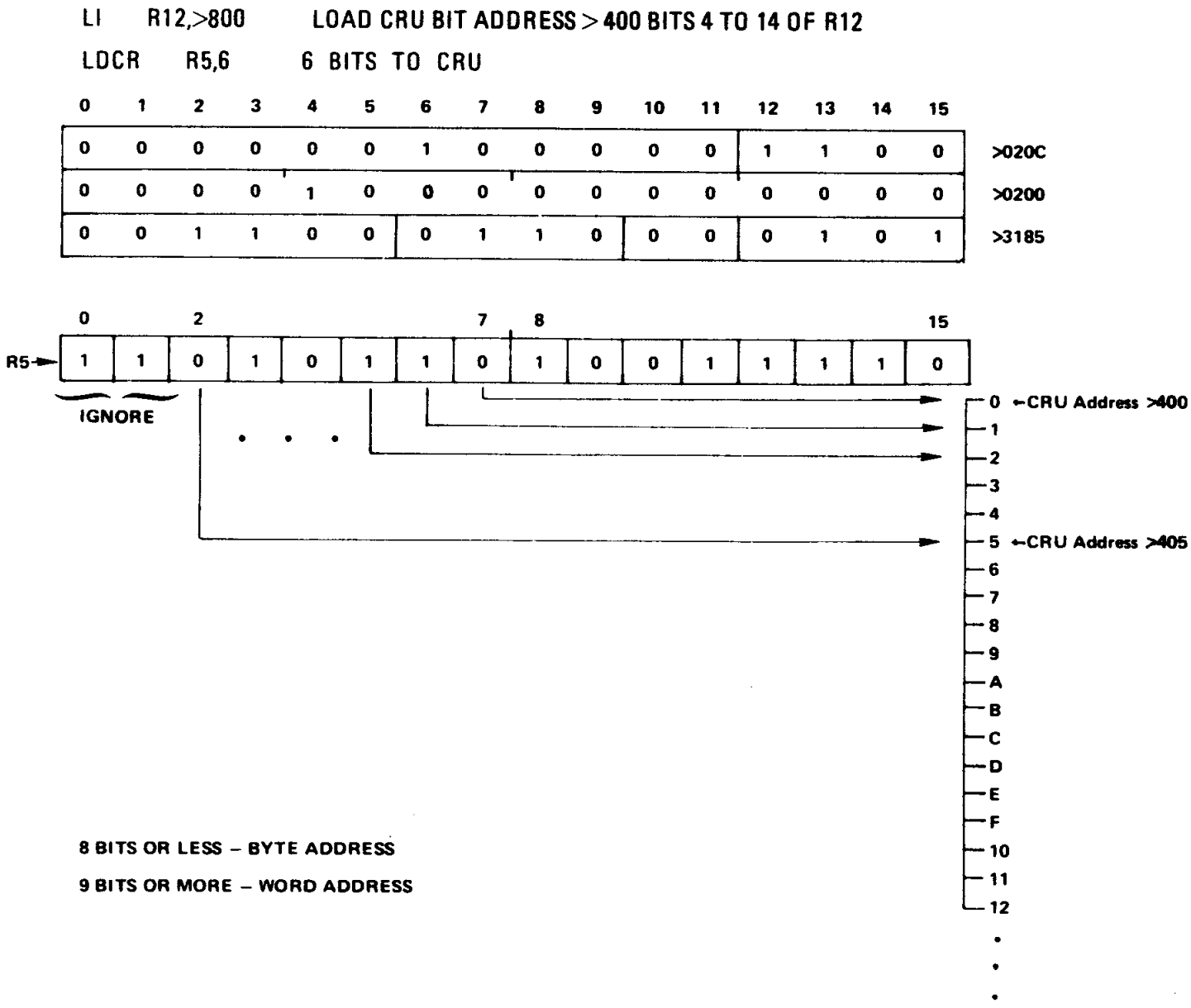


Figure 7-5. LDCR Byte Instruction

The TB, SBO, and SBZ instructions use a maximum displacement of +128 bits and −128 bits from the CRU bit designated in bits 4 to 14 of R12. Thus, if bit 400_{16} is designated in R12, bits 4 to 14, the following assembly language instructions and comments would apply.

```
TB      >10     TEST CRU BIT > 410
SBO     -1      SET CRU BIT > 3FF TO ONE
SBZ     16      SET BIT > 410 TO ZERO
```

The LDCR and STCR instructions address the CRU using the value in R12; these instructions do not have the advantage of specifying a displacement from the R12 value such as used by the CRU bit instructions. If it is necessary to change the CRU address, it is important to understand that only bits 4 to 14 need be modified. For example, if it is desired to load (LDCR) successive groups of 16 CRU ports, a value of 32 (not 16) must be added to the contents of R12 for each group in order to accurately change the contents of R12 bits 4 to 14 (AI R12,32). An alternate method would be to load a new value into R12 (LI R12, > 400; LI R12, > 420, etc.).

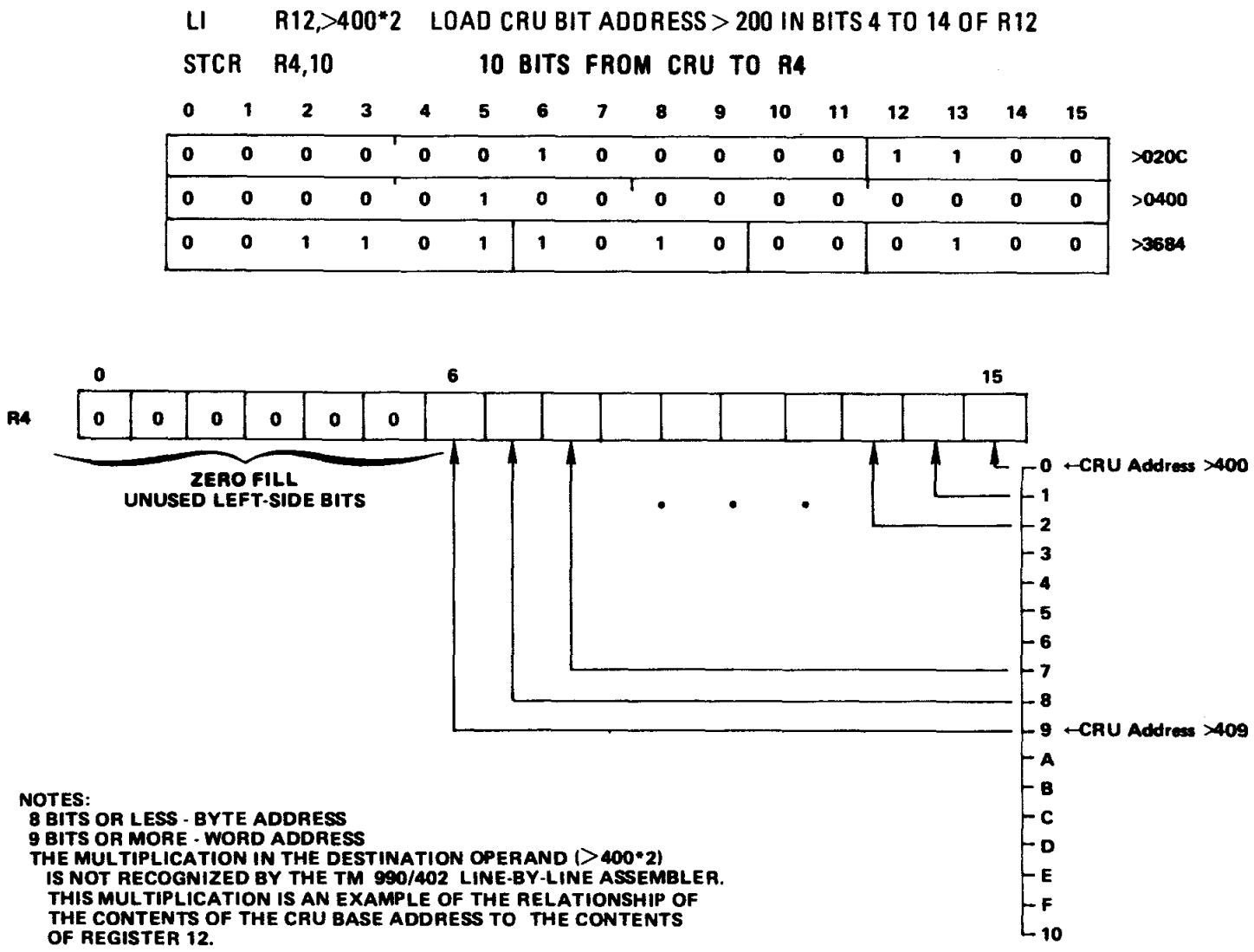


Figure 7-6. STCR Word Instruction

7.5 I/O PROGRAMMING WITH THE TMS 9901

The following figures, 7-7 to 7-12 are examples of addressing the TMS 9901 through the CRU, pointing out in graphic form:

- External I/O in parallel (multibit) and serial (single bit) forms.
- The relationship between the CRU bits addressed and the bits in the source operand of the STCR instructions.
- The relationship between the CRU bit addressed and the displacement in TB, SBO, and SBZ instructions.

The user TMS 9901 at U10 occupies 32 bit positions of CRU space with the low 16 bits beginning at CRU address 0000_{16} and the high 16 bits beginning at CRU address 0020_{16} (R12 contents). To access the low 16 bits of the TMS 9901 through the CRU, load 0000_{16} into register 12.

The high 16 bits starting at CRU address 0020_{16} are the parallel I/O bits, shown in the accompanying figures. These may be set, reset, or read in any order or combination with length from 1 to 16 bits. Since CRU operations are serial, data from the microprocessor (either serial or parallel) is transmitted serially to the TMS 9901, which outputs it in parallel. Likewise, during input, data present at the I/O pins is shifted serially to the microprocessor using the CRU bus for programming. It is necessary only to load register 12 with 0020_{16} and use either the LDCR or STCR instructions. Bear in mind that CRU operations of 1 to 8 bits affect the left byte (more significant half) of a word.

The lower 16 bits of the TMS 9901 starting at CRU address 0000_{16} are used for control of interrupts and the timer function, and to reset the I/O lines to the input mode with output buffers disabled and floating. Interrupt requests are presented to the TMS 9901, each on its own line, and are compared against an internal mask. If the internal interrupt mask allows, the particular interrupt request is encoded onto IC0 through IC3 of the TMS 9901 as explained on page 5 of the TMS 9901 data manual.

1) ASSEMBLY LANGUAGE:

```
LI      R12, >0020
LDCR    R0, 15
```

2) SOURCE ADDRESS IN MEMORY:

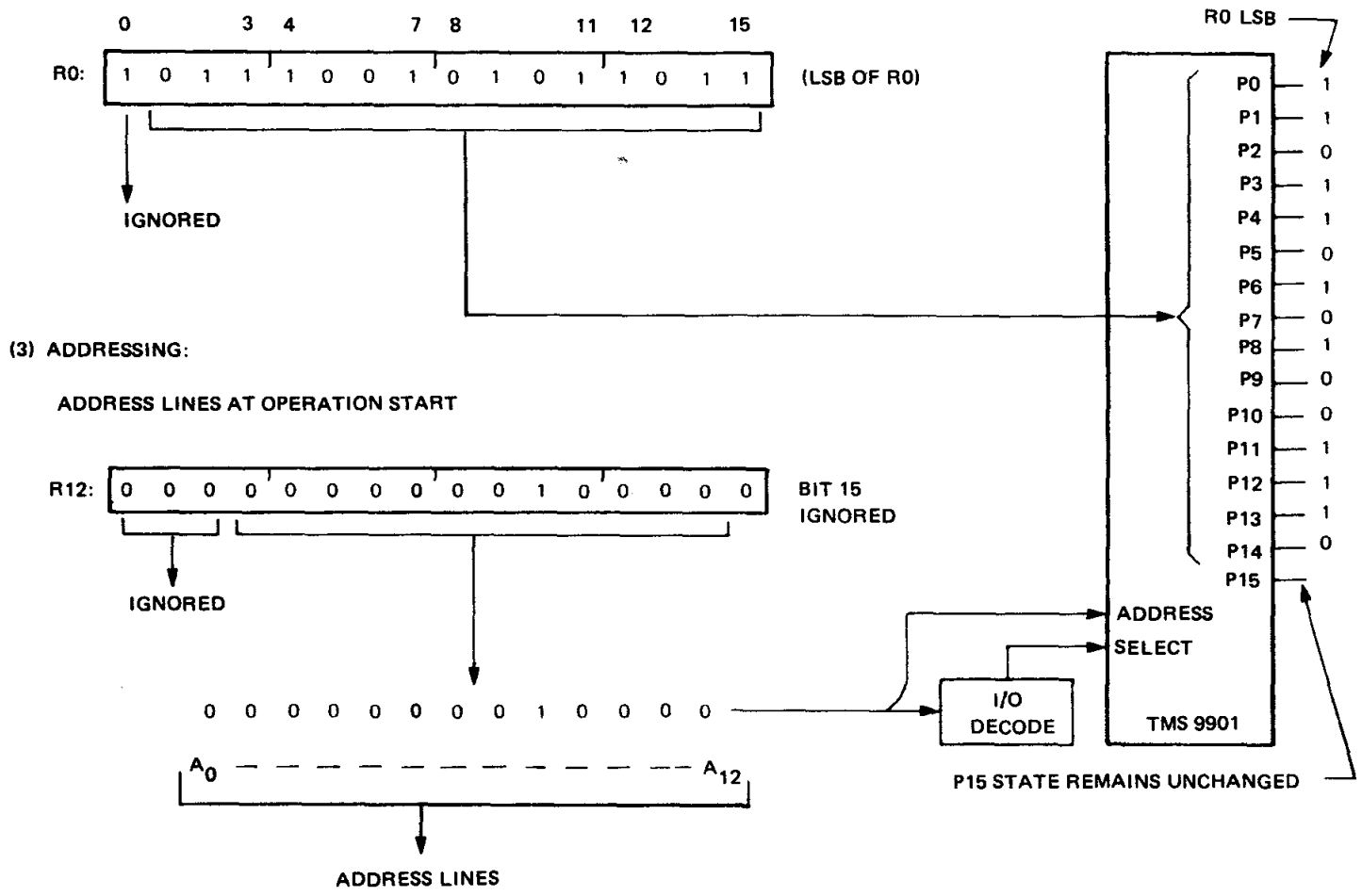


Figure 7-7. LDCR Word Execution to TMS 9901

(1) ASSEMBLY LANGUAGE:

```
LI      R12, >0030
LDCR    R2, 2
```

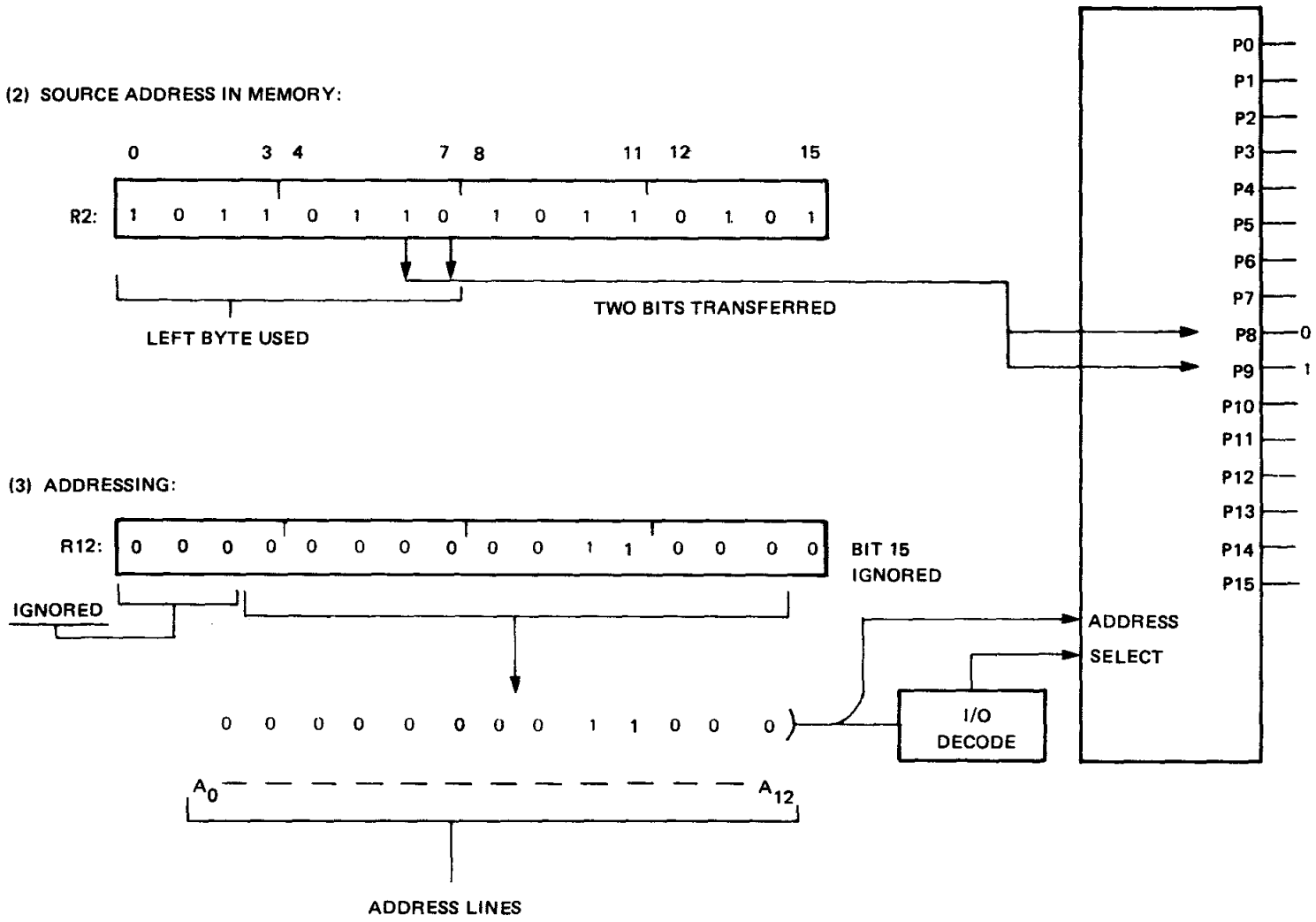
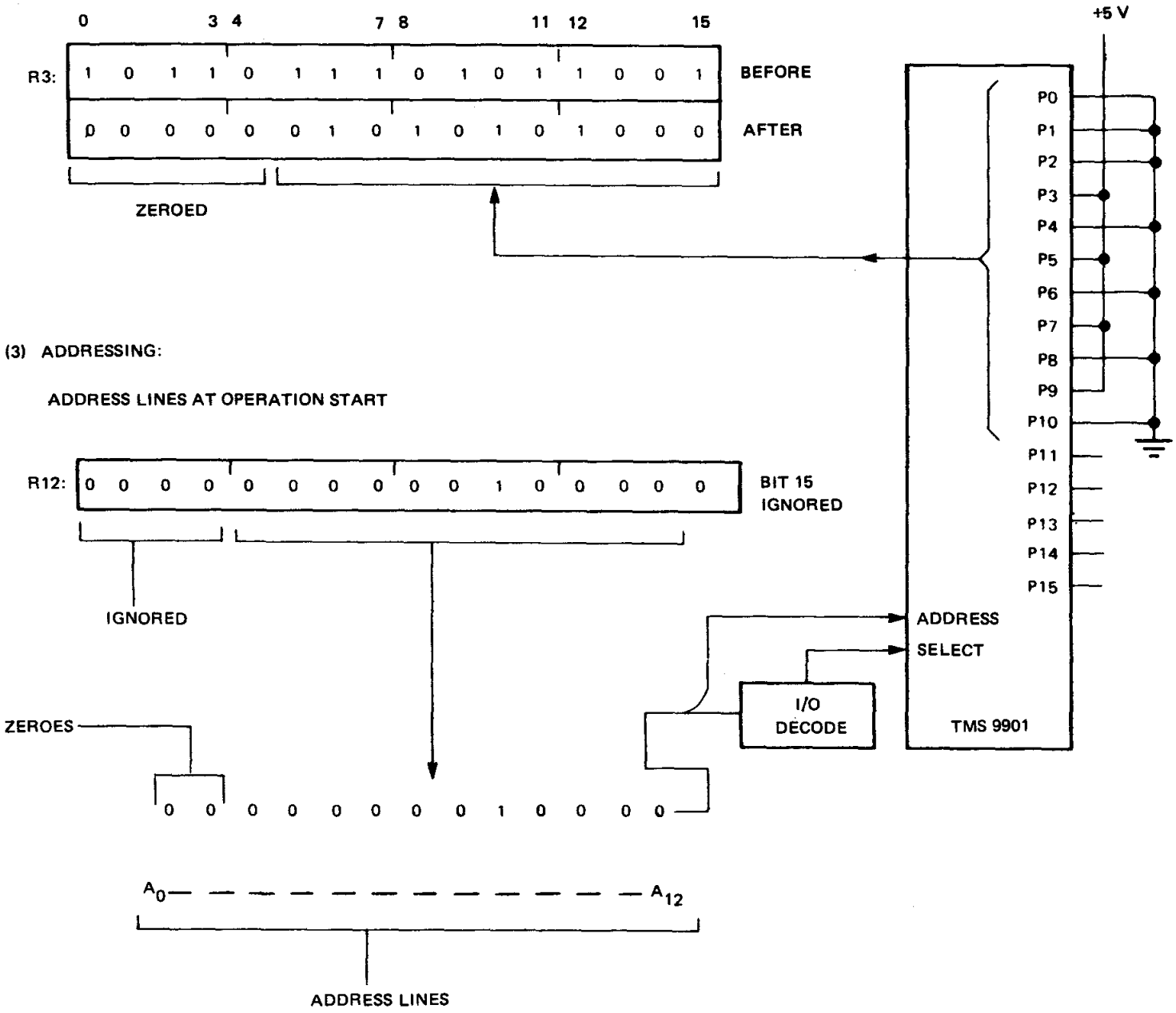


Figure 7-8. LDCR Byte Execution to TMS 9901

(1) ASSEMBLY LANGUAGE:

```
LI      R12, >0020
STCR    R3, 11
```

(2) SOURCE ADDRESS IN MEMORY:



(3) ADDRESSING:

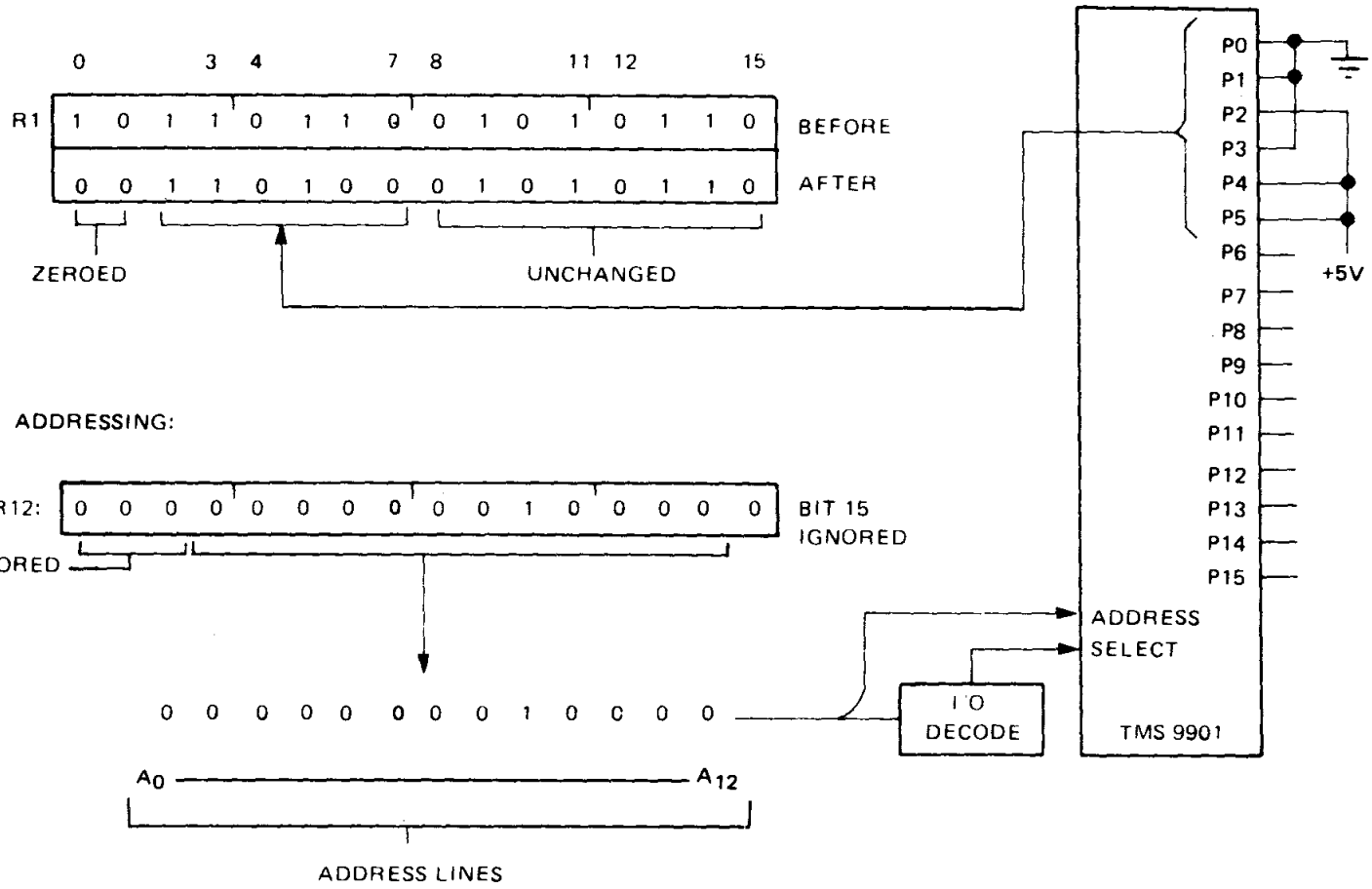
ADDRESS LINES AT OPERATION START

Figure 7-9. STCR Word Execution to TMS 9901

(1) ASSEMBLY LANGUAGE:

```
LI      R12, > 020
STCR    R1, 6
```

2) SOURCE ADDRESS IN MEMORY:

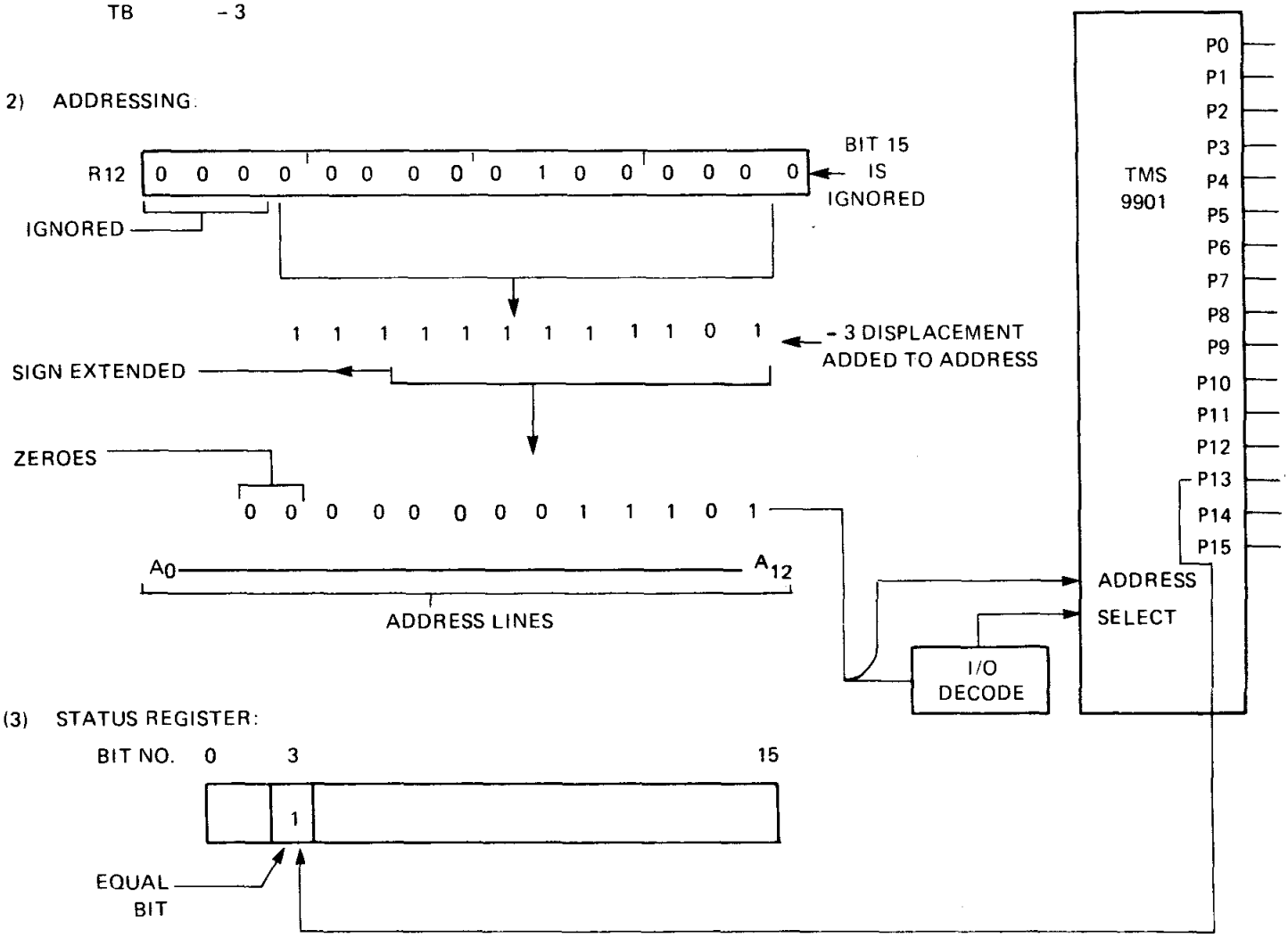


3) ADDRESSING:

Figure 7-10. STCR Byte Execution to TMS 9901

1) ASSEMBLY LANGUAGE

```
LI      R12, > 040
TB      - 3
```



NOTE

IF A JEQ (JUMP ON EQUAL) INSTRUCTION FOLLOWS A TB INSTRUCTION, A 1 FOUND WILL CAUSE A JUMP, AND A 0 FOUND WILL NOT CAUSE A JUMP (1 = EQUAL STATE).

Figure 7-11. Test CRU Bit at TMS 9901

(1) ASSEMBLY LANGUAGE:

```
LI    R12, >0020
SBZ   7
```

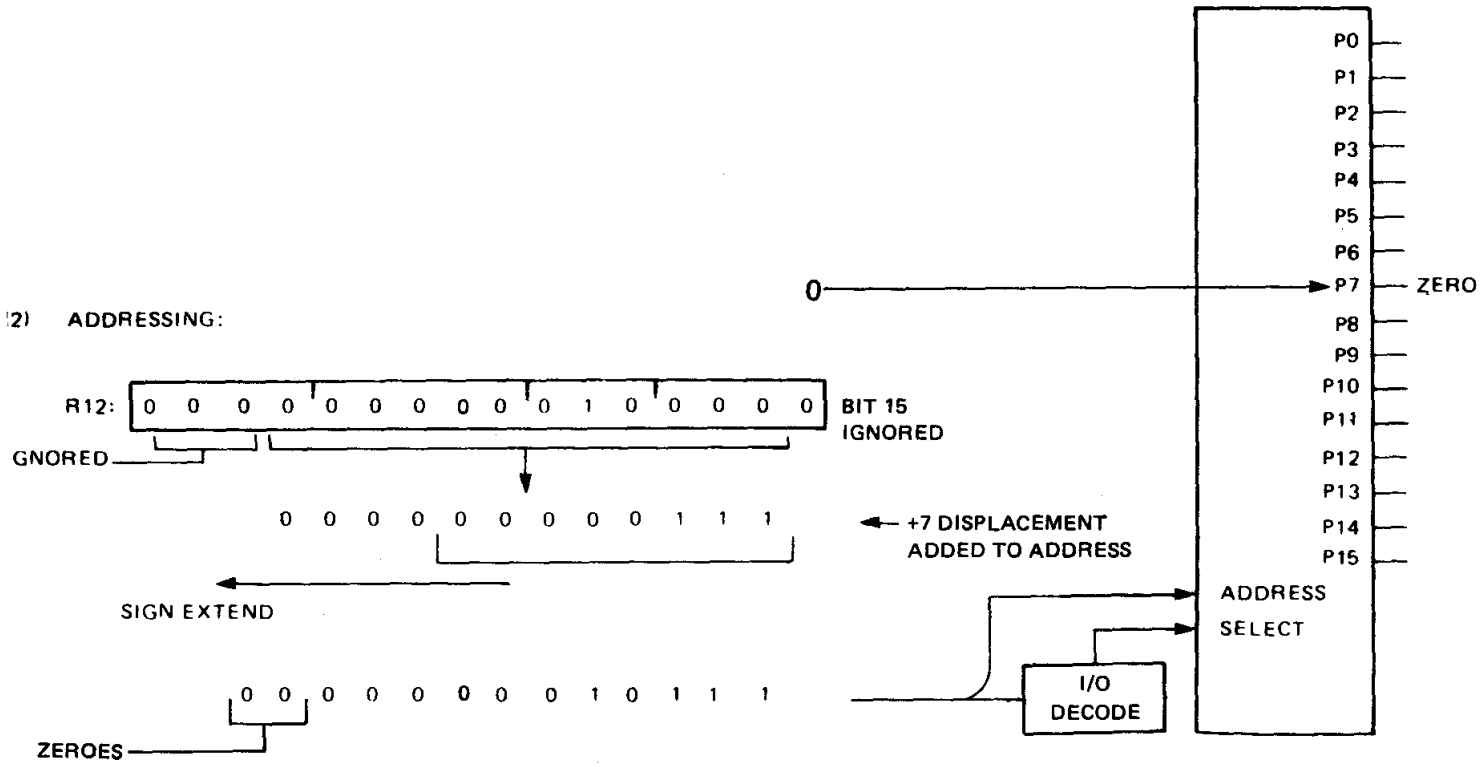


Figure 7-12. Set CRU Bit at TMS 9901

SECTION 8

THEORY OF OPERATION

8.1 INTRODUCTION

This section presents the theory of operation of the TM 990/189 Microcomputer. Information from the following manuals may be used to supplement material in this section:

- TMS 9980 Microprocessor Data Manual
- TMS 9901 Programmable Systems Interface Data Manual
- TMS 9902 Asynchronous Communications Controller Data Manual
- TMS 2532 Programmable Read Only Memory Data Sheet
- TMS 2708/2716 Programmable Read Only Memory Data Sheet
- TMS 4014 Random Access Memory Data Sheet
- TMS 4732 Read Only Memory Data Sheet
- TMS 9900 Family System Design Handbook
- The TTL Data Manual

8.1.1 SYSTEM ARCHITECTURE

Figure 8-1 shows the major function blocks of the TM 990/189 representing the processing, memory, and I/O portions of the microcomputer. Also shown in the figure are the primary signal buses of the system which are: the data bus, address bus, Communications Register Unit (CRU) bus, and the control bus.

Throughout the remainder of this section, the operation of these buses and function blocks will be discussed. The first topic is the buses, since they tie all the blocks together.

8.2 SYSTEM BUSES

8.2.1 DATA BUS

The data bus consists of eight bidirectional lines, D0 through D7, used to transfer information between the processor and memory, either onboard or offboard through the bus expansion interface. D0 is the most significant bit and D7 is the least significant.

The direction of data transfer is controlled by the processor and indicated by the state of control bus signal DBIN which is set to a logic one when the processor has disabled its data bus drivers and entered an input mode.

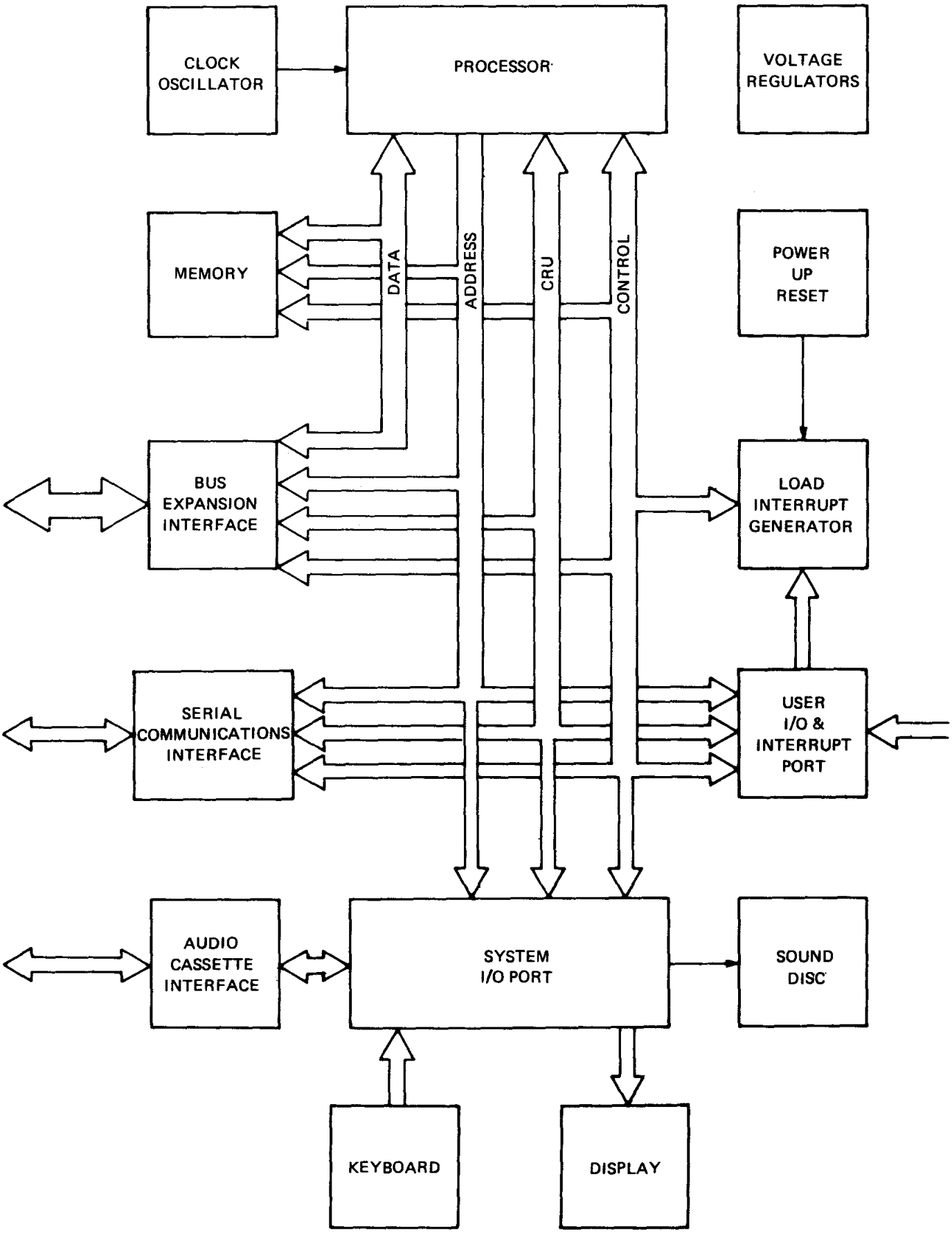


Figure 8-1. System Block Diagram

8.2.2 ADDRESS BUS

The address bus is composed of fourteen lines, A0 through A13 CRUOUT, which are normally driven by the processor and are used to reference individual memory and CRU locations. Memory references are distinguished from CRU operations by observing the state of control bus signal MEMEN— which the processor sets to a logic zero during memory cycles. During memory references A0 through A13 CRUOUT present the address of the byte being accessed, where A0 is the most significant bit and A13 CRUOUT is the least significant bit of the address. During CRU cycles, A0 and A1 are set to zero and A2 through A12 contain the effective CRU bit address being referenced, where A2 is the most significant bit and A12 is the least significant bit of the address. A13 CRUOUT is shared with the CRU bus during CRU operations and is discussed further below.

In addition, A0, A1, and A13 CRUOUT are used during the execution of external instructions (CKON, CKOF, RSET, IDLE, and LREX) to present the 3-bit code identifying each instructions.

8.2.3 CRU BUS

The CRU bus consists of four signals: A13 CRUOUT, CRUCLK, IOCLK, and CRUIN. During CRU output intructions (SBZ, SBO, and LDCR), A13 CRUOUT contains the value of the bit being output. After a delay to allow the CRU bit address (A2 through A12) and the output bit A13 CRUOUT) to stabilize, the processor strobes CRUCLK to latch the output bit in the output device. IOCLK is a buffered derivative of CRUCLK and is the actual strobing signal connected to the output devices. During CRU input instructions (TB and STCR) the processor again sets up the CRU bit address on A2 through A12 and, after a delay for settling, reads the input bit from CRUIN.

8.2.4 CONTROL BUS

The control bus consists of a number of timing, request, and status signals used by the processor and support circuitry. Included in this group are DBIN and MEMEN—, which were mentioned above, along with WE—, READY, DRE—, INT 0 through INT 2, HOLD—, HOLDA, IAQ, and ϕ3—. A brief description of each signal is given in Table 8-1.

TABLE 8-1. CONTROL BUS FUNCTIONS

SIGNAL	ACTIVE STATE	PURPOSE
WE—	LOW	Strobe to memory devices for writing data to memory.
READY	HIGH	Indicates to the processor that the memory is ready to be accessed. Wait states are generated by pulling this line low.
DRE—	LOW	Delays RAM chip selects during write operations to avoid bus conflicts. See paragraph 8.5.1.
INT 0 INT 1 INT 2	HIGH	Encoded interrupt request lines to processor. See paragraph 8.4.
HOLD—	LOW	Requests processor to give up control of address, data buses, MEMEN—, WE—, and DBIN.
HOLDA	HIGH	Acknowledges that processor has given up signals mentioned above, and has suspended activity.
IAQ	HIGH	Indicates that an instruction acquisition is under way.
ϕ3—	LOW	Clock signal generated by the processor for synchronization of external devices.

8.3 PROCESSOR

The processor function block in Figure 8-1 consists of a MP9529 microprocessor which is functionally equivalent to the TMS 9980A. The factors which distinguish the two are: (1) the MP9529 requires a V_{DD} supply of +9.3 volts typical, (2) has a maximum external clock frequency of 8.08 MHz, and (3) has a maximum free air operating temperature of 50°C. They are otherwise identical.

The processor is perhaps the most important part of the system since within it resides the capacity to make decisions and take different courses of action based on those decisions. Processor decisions may result not only through the execution of program instructions involving the processor's STATUS REGISTER (see Figure 8-2) but also as a result of interrupt signal inputs to the processor (see Figure 8-3).

In addition to decision making, the processor is responsible for:

- Instruction acquisition and interpretation
- Timing of most control signals and data transfers
- Data, Address, and CRU bus control.

Figure 8-4 shows the processor signals grouped by function along with the clock oscillator circuit.

8.3.1 CLOCK OSCILLATOR

The clock oscillator, composed of two inverters from U14, R9, R10, C6 and Y1, generates the 8 MHz signal INTCLK which is then buffered and routed to the processor as CKIN. From CKIN, the processor internally generates four mutually exclusive clock signals (ϕ1 through ϕ4) which it uses to synchronize its operations. In addition, the processor inverts and buffers one of its internal clocks to provide ϕ3— which is used externally by processor support and CRU circuitry.

8.3.2 EXTERNAL INSTRUCTION DECODING

Recall that the address bus is used not only to reference memory and CRU locations but also to identify external instructions. These instructions are CKON, CKOF, IDLE, LREX, and RSET. They allow user-defined external functions to be initiated under program control. When any of these five instructions are executed by the TMS 9980A, a unique three-bit code appears on A0, A1, and A13CRUOUT along with a CRUCLK pulse. IDLE also causes the TMS 9980A to enter the idle state and remain until an interrupt, RESET, or LOAD occurs. While in this state, the code and CRUCLK pulses occur repeatedly until the idle state is terminated.

Also recall that the CRU instructions SBZ, SBO, and LDCR also cause CRUCLK to strobe, but that A0 and A1 were always low (logic 0) for CRU operations. To prevent external instruction CRUCLK pulses from affecting CRU devices, the signal IOCLK is decoded by Network U5 from CRUCLK, A0, A1, and A13CRUOUT and is routed to the strobe inputs of CRU devices. U5 also decodes the external instruction codes into mutually exclusive signals for implementation as shown in Figure 8-5. The result of executing external instructions on the TM 990/189 is tabulated in Table 8-2.

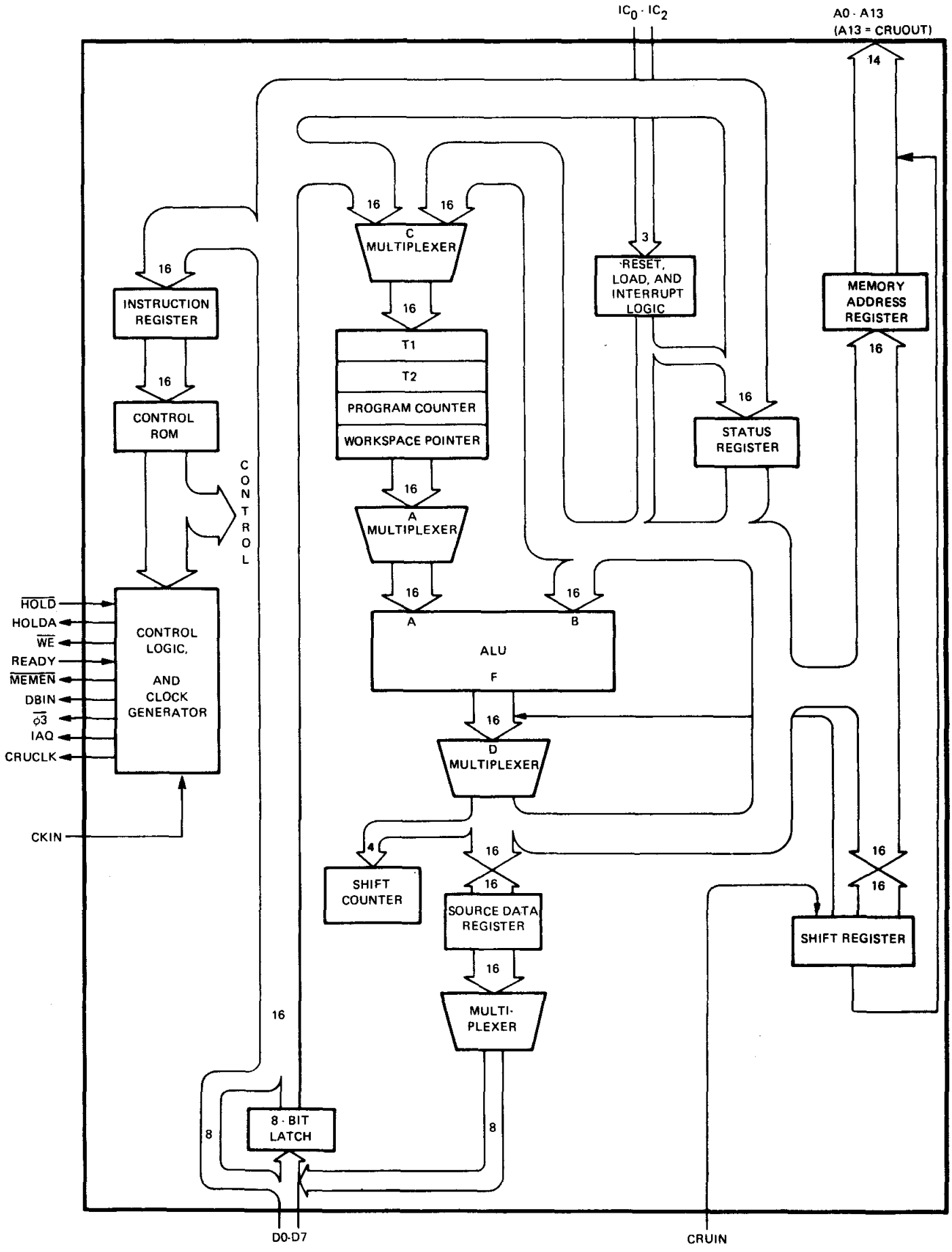


Figure 8-2. TMS 9980A Internal Architecture

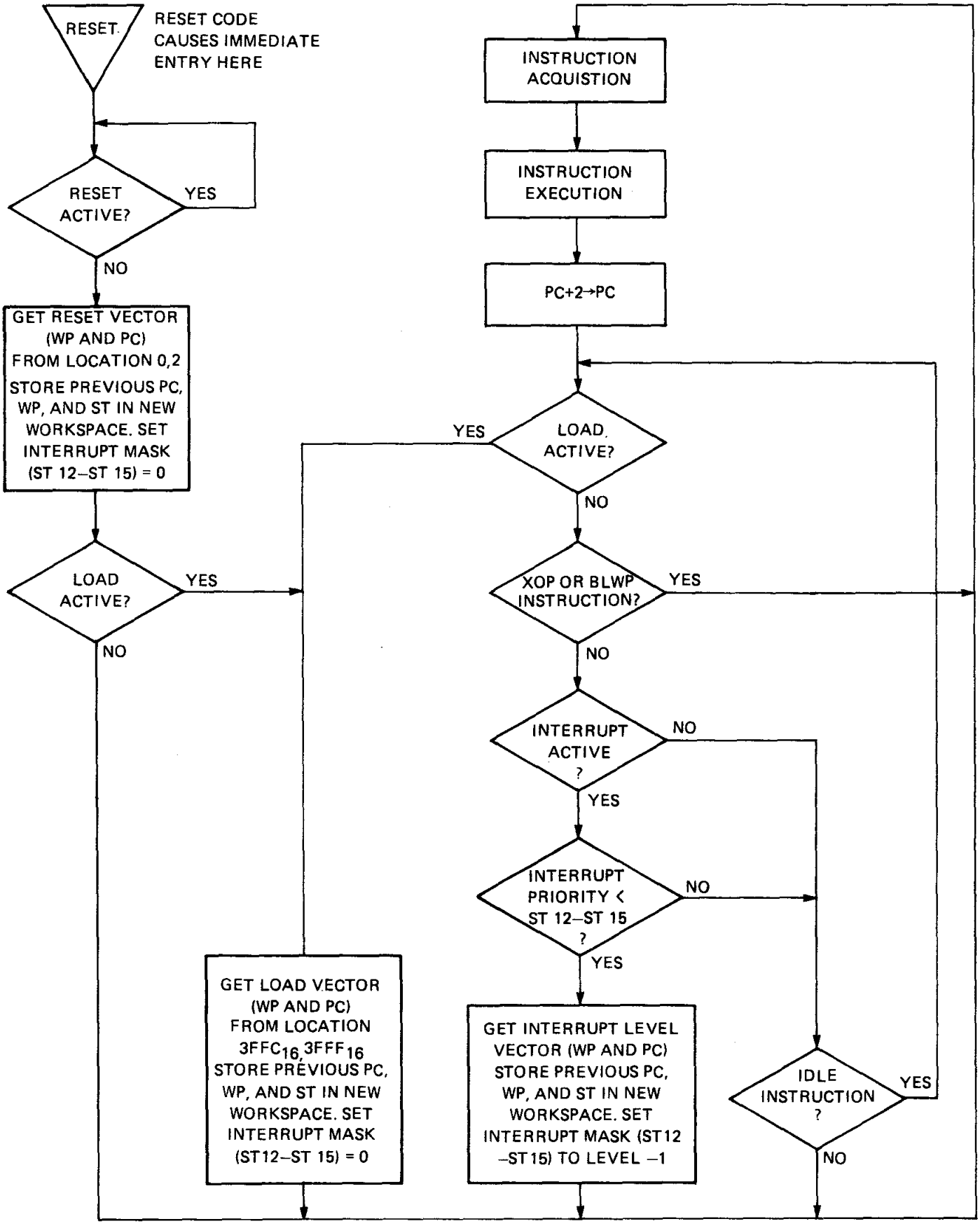


Figure 8-3. TMS 9980A CPU Flow Chart

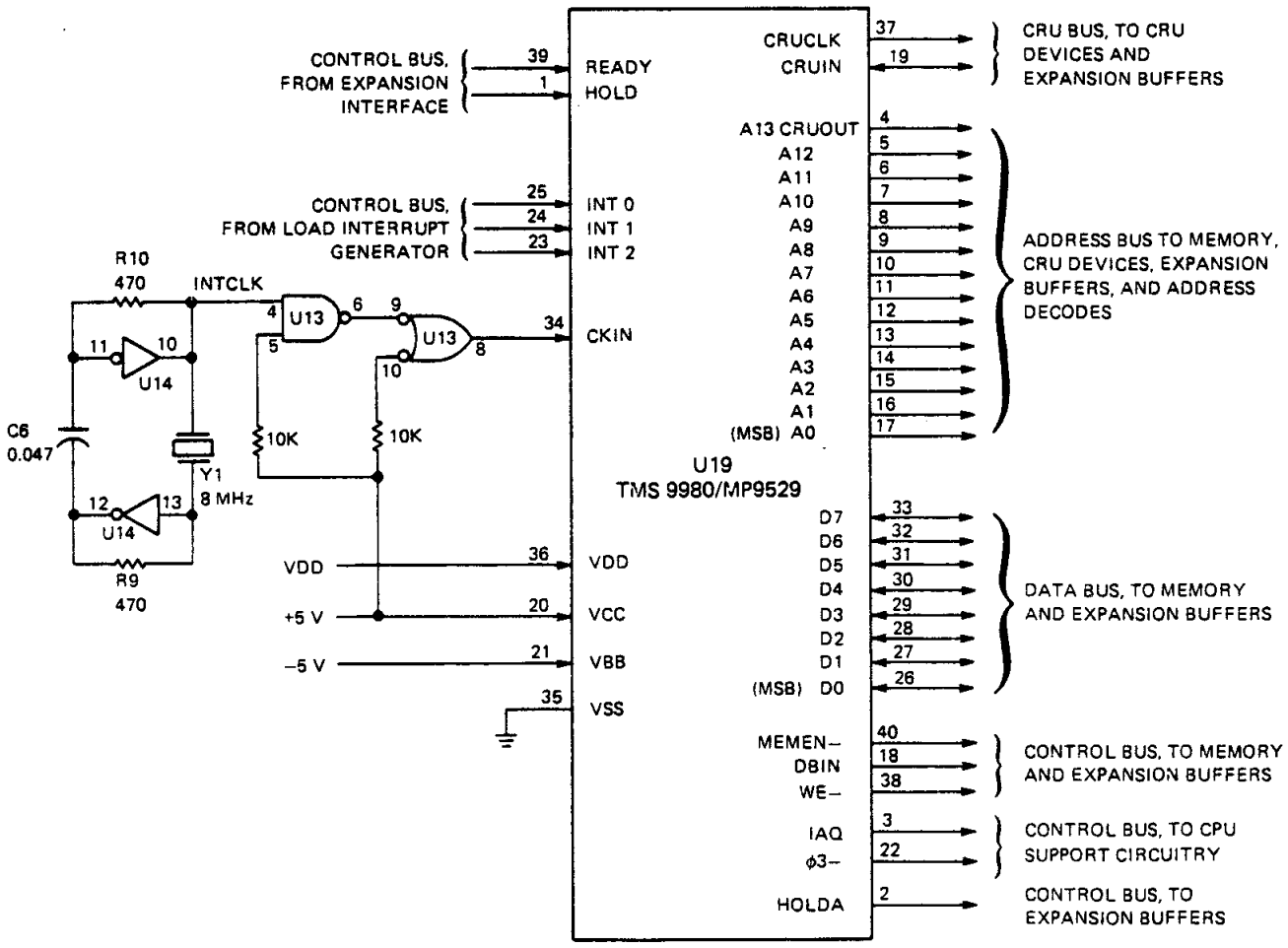


Figure 8-4. TMS 9980A Signals

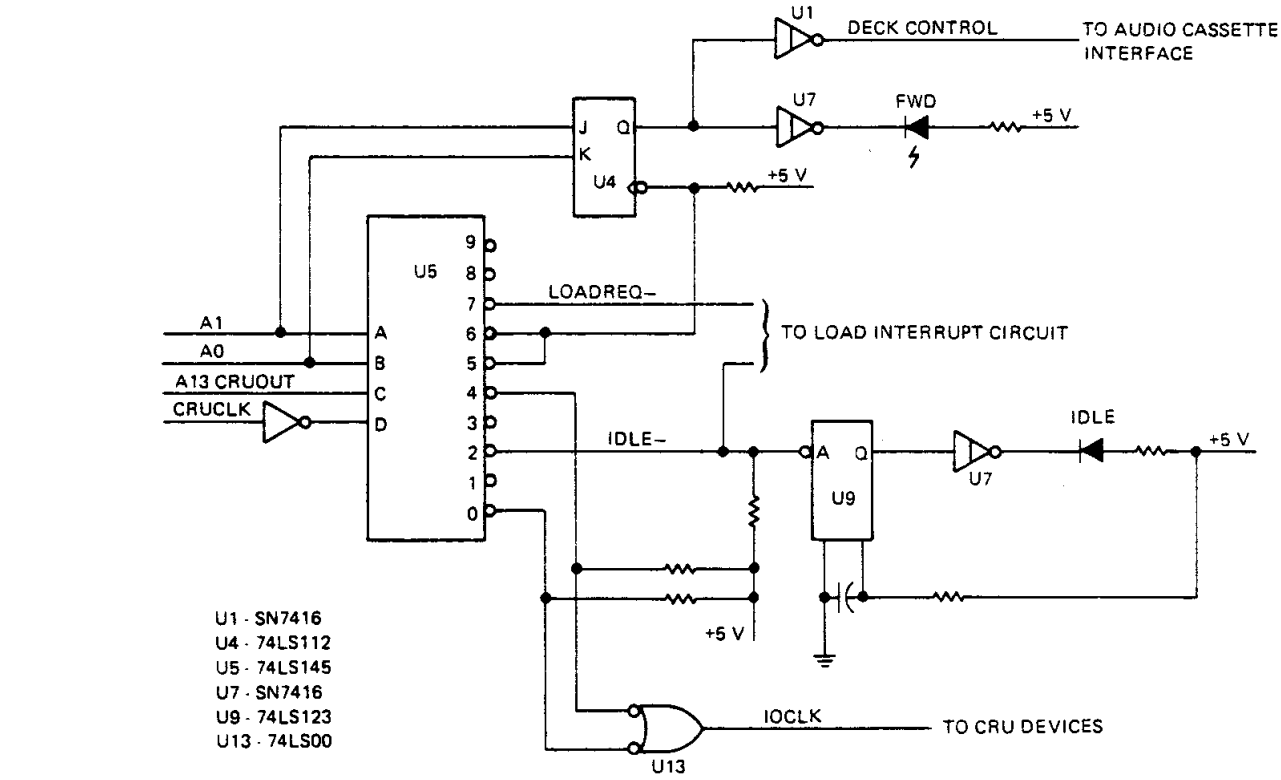


Figure 8-5. External Instruction Decode

TABLE 8-2. EXTERNAL INSTRUCTION RESULTS

INSTRUCTION	EXECUTION RESULTS
CKON	Illuminates FWD indicator, activates DECK CONTROL— signal to audio cassette interface.
CKOF	Extinguishes FWD indicator, deactivates DECK CONTROL— signal.
IDLE	Places TMS 9980A in idle state, illuminates IDLE indicator.
LREX	Causes TMS 9980A to execute a LOAD operation.
RSET	Not implemented. Available for user definition.

Control of the FWD indicator is accomplished through flip-flop U4. The decoded signals corresponding to the CKON and CKOF codes are wire-ORed and used to clock U4, which sets or resets depending on the external instruction code.

Network U9 is used to "stretch" the decoded IDLE— pulses to fully illuminate the IDLE indicator.

8.4 INTERRUPTS

Observe that the TMS 9980A has no separate lines for RESET and LOAD signals. Instead, those functions are decoded from the state of interrupt inputs to the microprocessor INTO, 1, and 2 as shown in Table 8-3. Receipt of an active interrupt code causes the processor to respond as illustrated in the CPU Flow Chart, Figure 8-3. Note that the inputs on INTO to INT2 for interrupt levels 1 to 4 are a binary 3 to 6 (011_2 to 110_2).

TABLE 8-3. TMS 9980A INTERRUPT DECODING

INTERRUPT CODE INT0-INT2	FUNCTION	VECTOR ADDRESS
000	RESET	0000
001	RESET	0000
010	LOAD	3FFC
011	INTERRUPT 1	0004
100	INTERRUPT 2	000B
101	INTERRUPT 3	000C
110	INTERRUPT 4	0010
111	—	

Encoding of the interrupt code signals is performed by the interrupt section of the USER I/O PORT and the LOAD INTERRUPT GENERATOR. Interrupt signal flow is diagrammed in Figure 8-6. The interrupt code is normally generated by the TMS 9901 (U10) in the USER I/O PORT which receives interrupt requests from the SYSTEM I/O PORT and from external sources through connector P5. Interrupt sources are tabulated in Table 8-4.

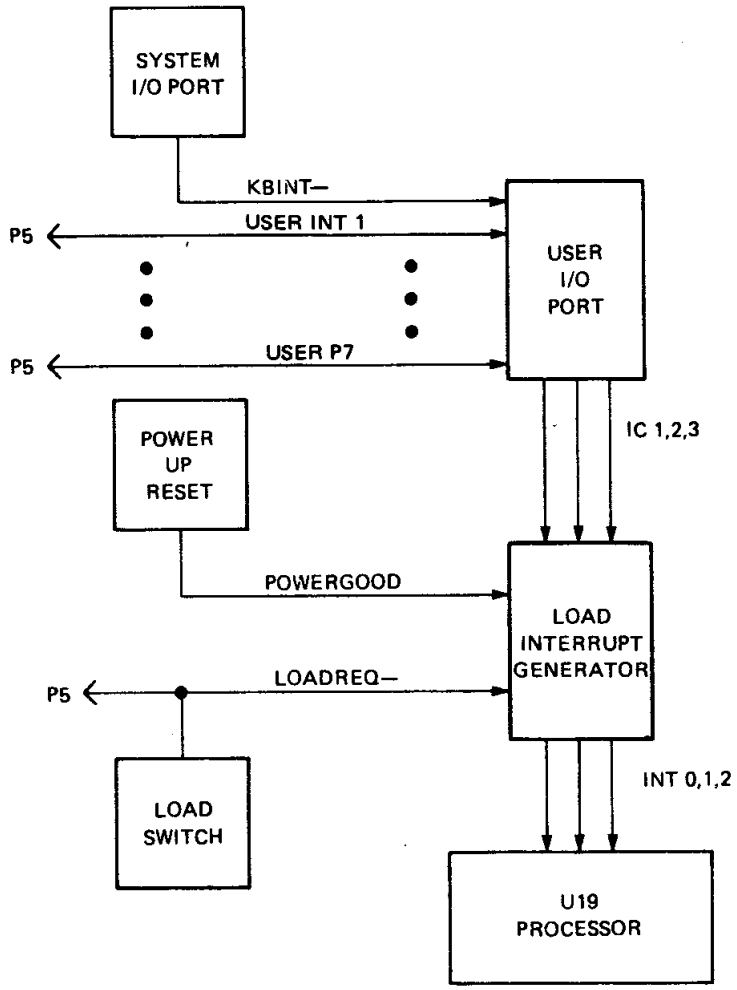


Figure 8-6. Interrupt Signal Flow

TABLE 8-4. INTERRUPT SOURCES

CPU FUNCTION	INTERRUPT SOURCE*	SIGNAL NAME	U10 INTERNAL SIGNAL NAME
RESET	POWER-UP RESET CIRCUIT, P5	RESET—	RST1—
	P5	USER INT 1—	INT 1—
	P5, U10	USER P14	INT 8—/P14
	P5, U10	USER P13	INT 9—/P13
LOAD	P5	USER INT 2—	INT2—
	P5, U10	USER P12	INT 10—/P12
INTERRUPT 1	P5	USER INT 3—	INT 3—
	P5, U10	USER P11	INT 11—/P11
INTERRUPT 2	P5	USER INT 4—	INT 4—
	P5, U10	USER P10	INT 12—/P10
INTERRUPT 3	P5	USER INT 5—	INT 5—
	P5, U10	USER P9	INT 13—/P9
INTERRUPT 4	SYSTEM I/O PORT	KBINT	INT 6—
	P5, U10	USER P8	INT 14—/P8
—	P5, U10	USER P15	INT 7—/P15
	P5, U10	USER P7	INT 15—/P7

*P5 — Connector P5; U10 = User TMS 9901

Two features of the TMS 9901 should be understood before proceeding. First, observe in the TMS 9901 Data Manual that interrupt signals INT7— through INT15— share device pins with input-output signals P15 through P7 respectively. The TMS 9901 automatically disables interrupts from shared pins when that pin is programmed as an output. Without this protection, unwanted interrupts might be generated merely by writing a zero to an I/O bit which shares a device pin with an interrupt input signal.

Second, observe in Table 8-3 that INT7— and INT15— generate no interrupt codes usable to the processor in this system. However, they do affect the TMS 9901. For instance, if enabled, pulling the INT15—/P7 pin (signal USERP7) low will cause the TMS 9901 to set its internal INTREQ— bit low, which could then be polled by the processor. Likewise, pulling the INT7—/P15 pin low will cause the same result with the following added action: since the TMS 9901 prioritizes interrupt requests, INT7— being active effectively disables INT8— through INT14— which do cause processor action in this system. INT 15— would also be disabled.

From the USER PORT, the interrupt code is routed to the LOAD INTERRUPT GENERATOR which may pass the code on to the processor or interpose its own code depending on the state of POWERGOOD and LOADREQ—.

POWERGOOD is generated by the POWER-UP RESET circuit composed of R29, R31, C11, CR8, Q11, U1, U14 and U36 shown in Figure 8-7. At the instant power is applied, C11 begins charging through R31. Q11 and R29 form a voltage follower to buffer the relatively low input impedance of the Schmitt trigger (U36) from the RC network. After approximately one second, the voltage at the emitter of Q11 will have risen to the upper threshold voltage of the Schmitt trigger and its output will go low. That signal is inverted by U14 and becomes POWERGOOD which is fed to a section of U2, a quad two input multiplexer and edge triggered flip-flop. Since POWERGOOD is connected to both D multiplexer inputs, the signal is latched in the flip-flop regardless of the state of the multiplexer select input. The latching operation synchronizes changes in POWERGOOD with $\phi 3$— and produces PG SYNC. Also notice that while POWERGOOD is low, RESET— is also low, thereby driving the I/O ports (U10 and U11) to known states. PG SYNC is then fed back into U2 to generate the interposing reset and load codes, and to U3 in the load delay circuit. Power-up sequence timing is shown in Figure 8-8.

LOADREQ— is generated by the external instruction decoder U5 during an LREX instruction (see Section 8.3.2) or by the load switch S1 and its debounce circuitry U6 and U7 shown in Figure 8-7. The falling edge of LOADREQ— clocks U4, which allows the timing sequence to proceed even if LOADREQ— stays low for an extended period as is possible during actuation of S1. The output of U4 is then clocked by IAQSYNC through two delaying stages in U3 before activating the LOAD— signal. This two-instruction delay is utilized by the UNIBUG monitor when executing a SINGLE STEP command. The timing sequence initiated by a single step execution is shown in Figure 8-9. The timing for a LOAD sequence due to S1 actuation is similar except that LOADREQ— may stay low for an extended time, and the instructions executed before and after LOAD code recognition may differ from those described.

The reader may wonder how a single LOAD vector fixed in EPROM could correctly direct the processor through both power-up and LREX sequences, where obviously different actions are performed. To accomplish this, the UNIBUG monitor keeps a flag in memory to tell it if a single step command is currently being performed. Based on the contents of this flag, the processor branches either to the completion of the single step routine or to the power-up routine as appropriate.

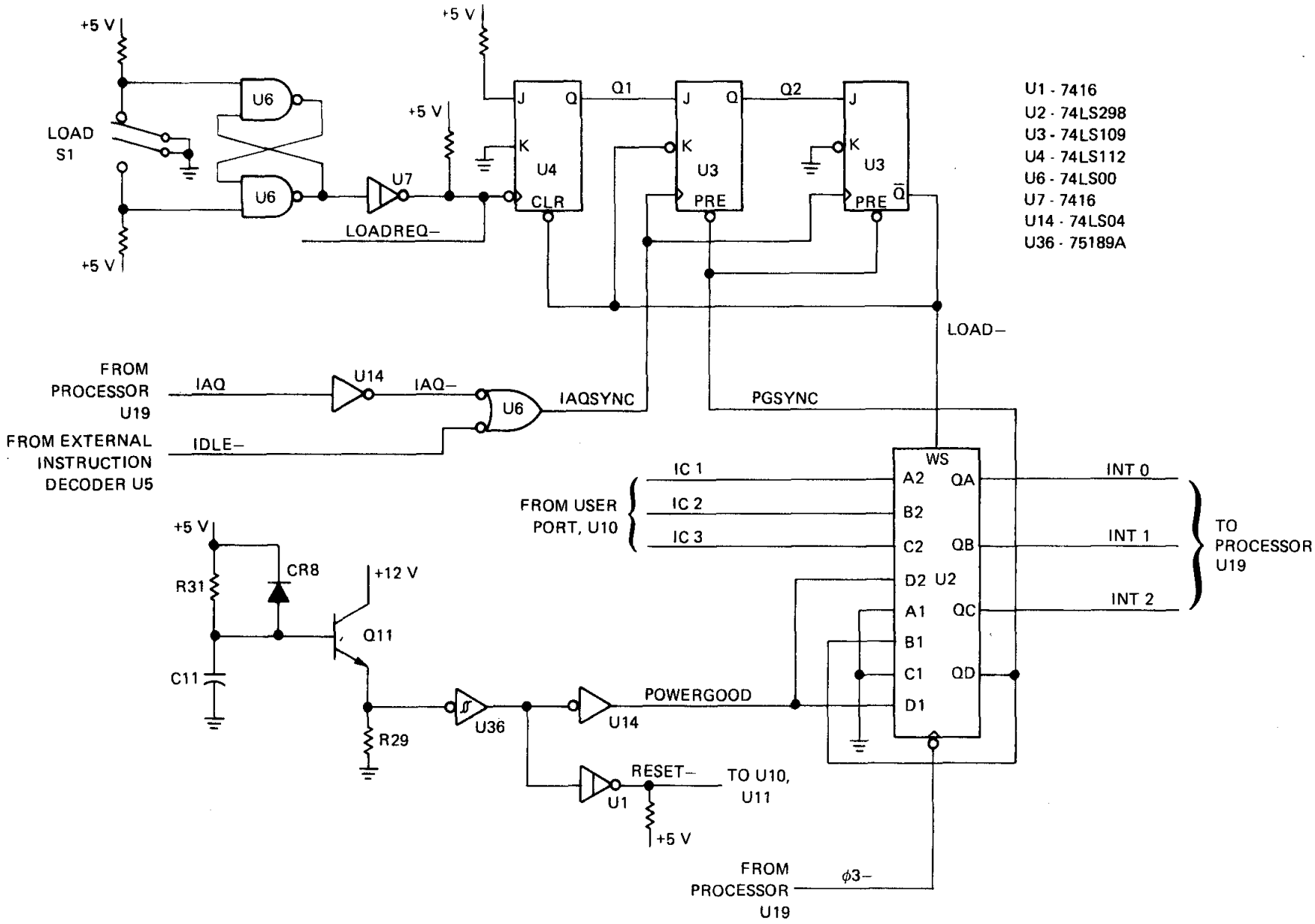
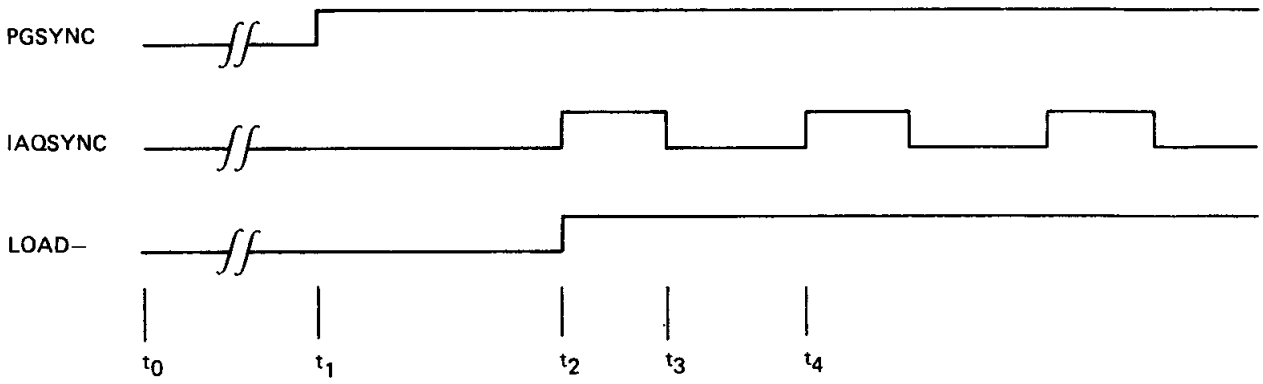
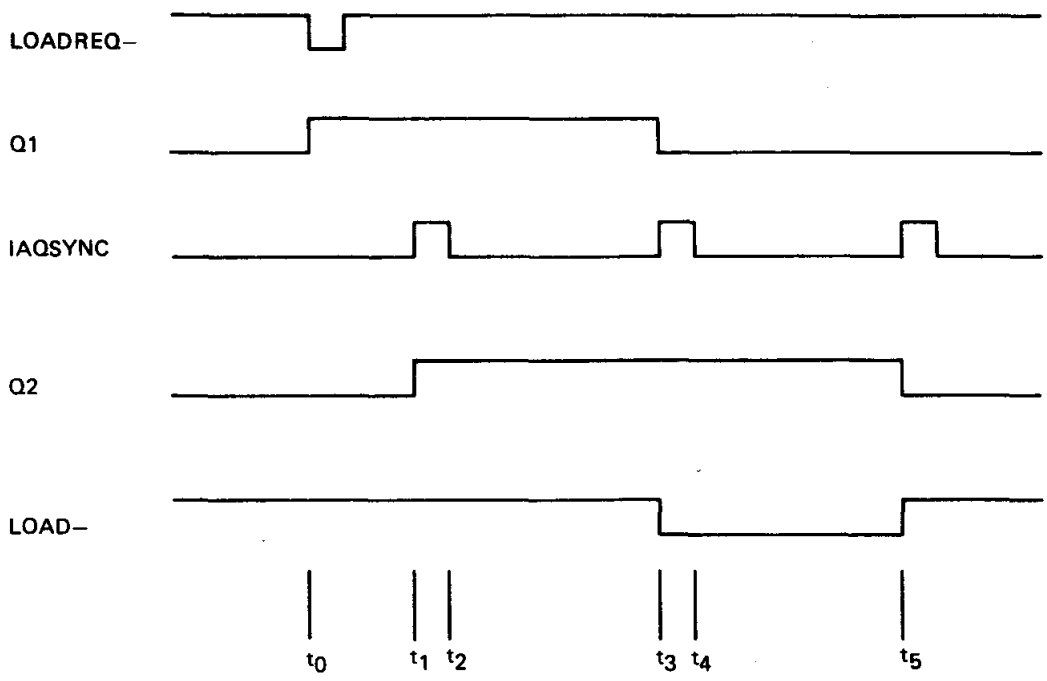


Figure 8-7. Load Interrupt Generator and Power Up Reset Circuit



TIME	INTERRUPT CODE	ACTION
t_0	UNSTABLE	POWER APPLIED
t_0-t_1	000	U3 PRESET, LOAD– =0 PROCESSOR WAITS FOR RESET CODE TO BECOME INACTIVE
t_1	CHANGING	PGSYNC FOLLOWS POWERGOOD HIGH
t_1-t_2	010	PROCESSOR GETS RESET WP AND PC, STORES PREVIOUS WP, PC, AND ST IN NEW WORKSPACE, SETS INTERRUPT PRIORITY =0, RECOGNIZES LOAD CODE, GETS LOAD WP AND PC, STORES RESET WP, PC, AND ST IN NEW WORKSPACE, SETS INTERRUPT PRIORITY =0.
t_2	CHANGING	IAQ PULSE CLOCKS U3, LOAD– GOES HIGH
t_2-t_3	IC1, IC2, IC3	PROCESSOR FETCHES FIRST INSTRUCTION IN POWER-UP ROUTINE
t_3		PROCESSOR COMPLETES FETCH.
t_3-t4	IC1, IC2, IC3	PROCESSOR EXECUTES FIRST INSTRUCTION.

Figure 8-8. Power-Up Sequence Timing



TIME	ACTION
t_0	EXECUTION OF LREX INSTRUCTION CAUSES LOADREQ– PULSE, U4 SETS, Q1 GOES HIGH.
t_0-t_1	PROCESSOR COMPLETES LREX INSTRUCTION,INCREMENTS PC FOR NEXT INSTRUCTION.
t_1	PROCESSOR STARTS INSTRUCTION ACQUISITION CYCLE. IAQSYNC PULSE CLOCKS U3, Q2 FOLLOWS Q1 HIGH.
t_1-t_2	PROCESSOR ACQUIRES AN RTWP INSTRUCTION.
t_2	INSTRUCTION ACQUISITION COMPLETE.
t_2-t_3	PROCESSOR EXECUTES RTWP; LOADS INTERNAL WP, PC, AND ST FROM MONITOR WORKSPACE REGISTERS 13, 14, AND 15, WHICH THE UNIBUG MONITOR HAS PREVIOUSLY LOADED.
t_3	PROCESSOR BEGINS TO ACQUIRE INSTRUCTION FROM USER PROGRAM. IAQSYNC CLOCKS Q2 INTO SECOND HALF OF U3, LOAD– GOES LOW, LOAD INTERRUPT CODE GENERATED BY U2, FED TO PROCESSOR. U4 CLEARS, Q1 GOES LOW.
t_3-t_4	PROCESSOR ACQUIRES INSTRUCTION FROM USER PROGRAM.
t_4	INSTRUCTION ACQUISITION COMPLETE.
t_4-t_5	PROCESSOR EXECUTES USER PROGRAM INSTRUCTION,INCRE– MENTS PC, RECOGNIZES LOAD CODE, GETS LOAD VECTOR, STORES WP, PC, AND ST IN MONITOR WORKSPACE.
t_5 and beyond	NOW BACK UNDER UNIBUG CONTROL, PROCESSOR DISPLAYS PC OF NEXT USER PROGRAM INSTRUCTION AND WAITS FOR NEXT COMMAND.

Figure 8-9. Load Timing

8.5 MEMORY

With fourteen address lines, the TMS 9980A can address up to 16,384 eight-bit memory locations. In this system, a total of 8192 locations are dedicated to onboard devices and the remaining locations are reserved for offboard functions. The system memory map, Figure 8-10 shows the organization of the memory space.

8.5.1 MEMORY ADDRESS DECODING

The memory address decoding circuit is shown in Figure 8-11. Chip enables for the various onboard memory devices are generated by the coincidence of certain control signals discussed below.

Address decoding is performed by U34 which divides the address space into four 4096 byte sectors identified by the control signals LOMEMENA— (low memory enable), DECODE 1, DECODE 2, and HIMEMENA— (high memory enable) as shown in Figure 8-12. As seen from the diagram, the 4K byte system ROM (U33) is enabled by ROMCE— whenever HIMEMENA— is low ($3000_{16} \leq$ ADDRESS $\leq 3FFF_{16}$) and DBIN is high (read cycle in progress).

In addition U34 subdivides each sector into 1024 byte blocks, identified by BLOCK 0— through BLOCK 3—. Thus system RAM (U20 and U22) is enabled by RAMCE— whenever LOMEMENA— is low (0000 $\leq$ ADDRESS $\leq$ 0FFF), BLOCK 0— is low (X000 $\leq$ ADDRESS $\leq$ X400), and DRE— is low.

DRE— (delayed RAM enable) is derived from DBIN and WE— and is used to prevent data bus conflicts. The RAM devices used in this system normally enable their data output buffers in response to a chip enable unless WE— goes low first. If the data buffers were enabled while the processor was outputting write data on the same lines, erroneous data may be written into the memory. DRE— avoids this problem by delaying RAM chip enables during write operations until WE— has gone low. RAM read and write cycle timing is shown in Figure 8-13 and 8-14.

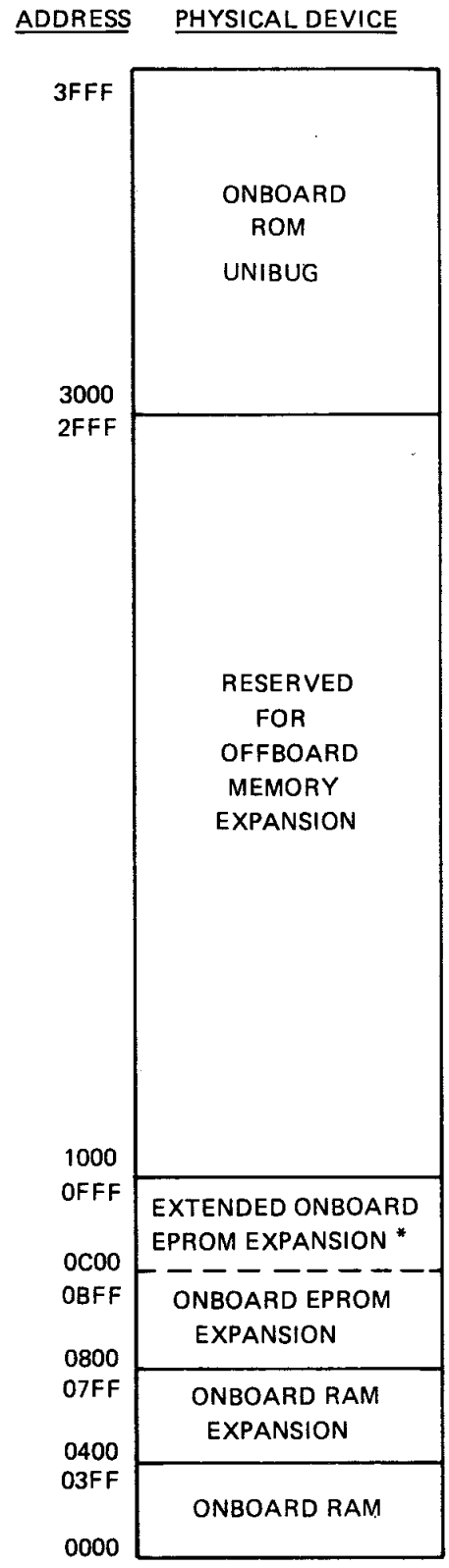
In a similar manner, onboard expansion RAM (U21 and U23) is enabled by EXPRAMCE— under the true conditions of LOMEMENA—, BLOCK 1—, and DRE—. The expansion ROM (U32) is enabled by EPROMCE— when LOMEMENA—, DBIN—, and BLOCK 2— are all low. A jumper option allows BLOCK 3— to be OR'ed with BLOCK 2— to increase the EPROM memory space to 2K bytes for a TMS 2716.

8.5.2 MEMORY EXPANSION

Offboard memory expansion capability is provided through the Bus Expansion Interface which generates buffered signals corresponding to A0 through A13 CRUOUT, D0 through D7, MEMEN—, WE—, DBIN, and HOLDA. READY and HOLD— inputs to the processor are also provided.

The direction of the expansion data bus buffers is controlled by the DIN— signal which is generated as shown in Figure 8-15. The buffers are directed inward whenever DECODE 1— or DECODE 2— is low ($1000_{16} \leq$ ADDRESS $\leq 2FFF_{16}$) and DBIN is high (read cycle in progress) to allow offboard memory data to be read by the processor. At all other times, the data buffers are directed outward, as are the address and control buffers.

With one exception, all expansion buffer outputs go to a passive high impedance state in response to the HOLDA signal from the processor. The exception is EXPHOLDA (expansion hold acknowledge) which always drives outward to inform offboard circuitry that the system has relinquished control of the expansion buses.



* USER STRAPPABLE OPTION (SEE SECTION 9)

Figure 8-10. System Memory Map

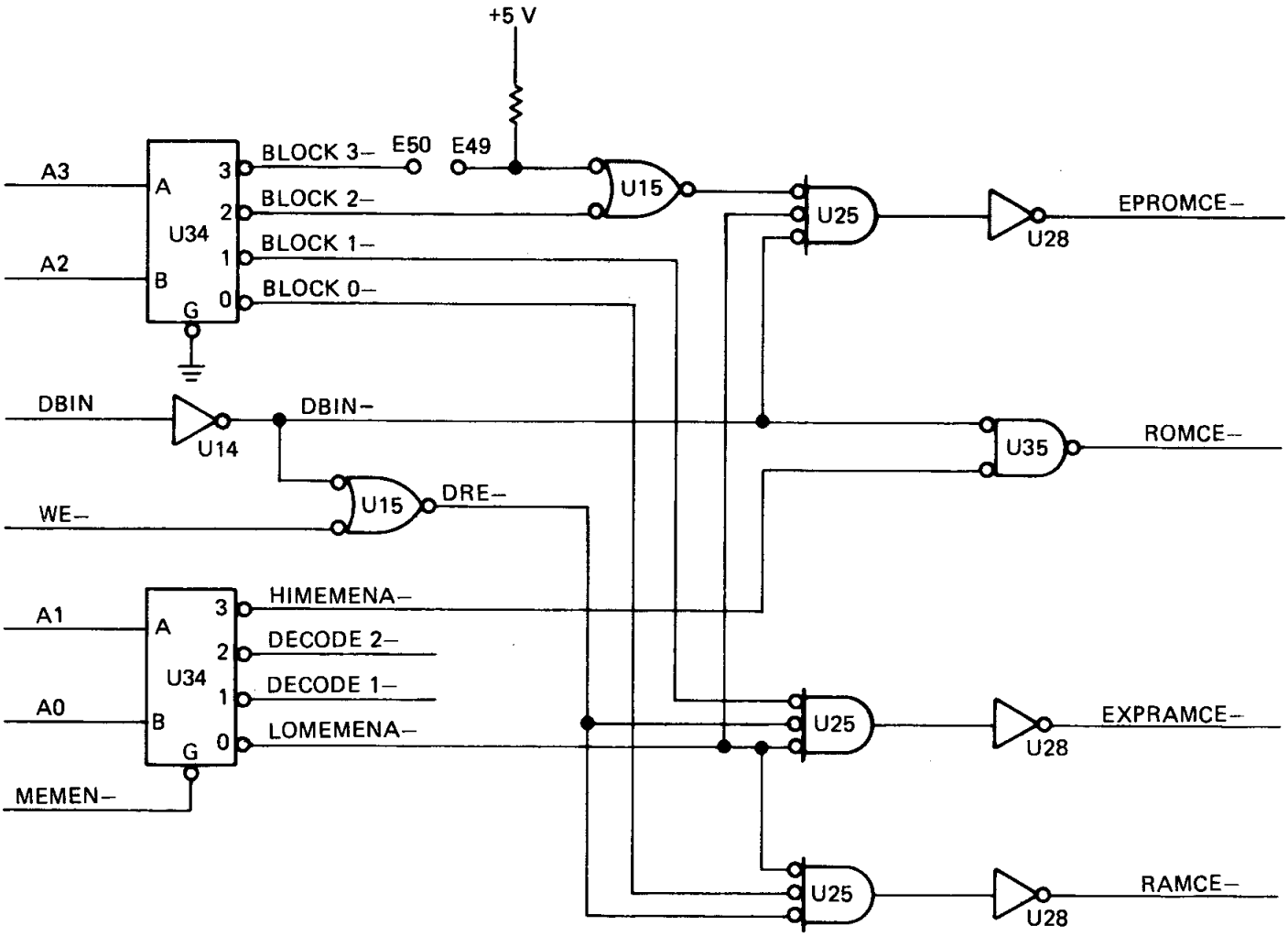


Figure 8-11. Memory Address Decoding

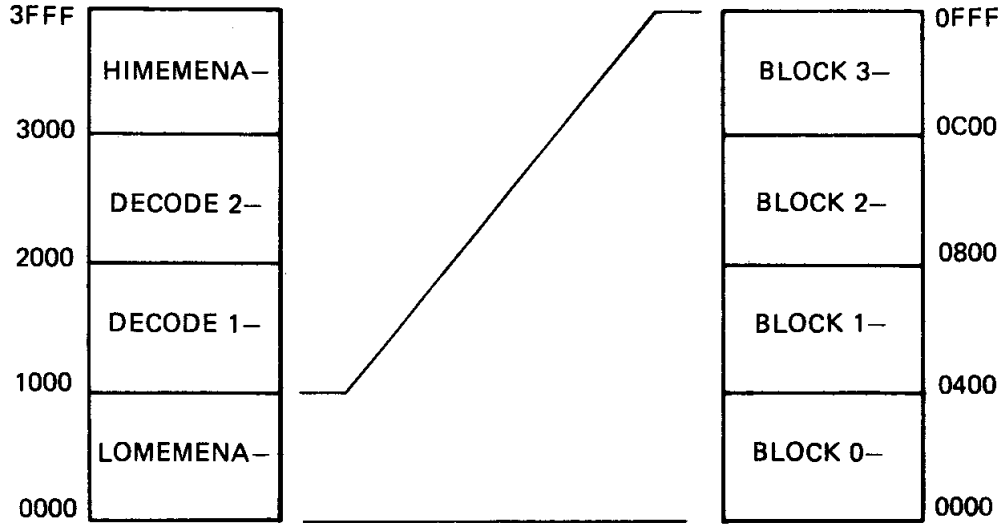


Figure 8-12. Memory Partitioning Signals

NOTE: PROPAGATION DELAYS EXAGGERATED TO SHOW SEQUENCE OF EVENTS.

Figure 8-13. RAM Read Cycle Timing

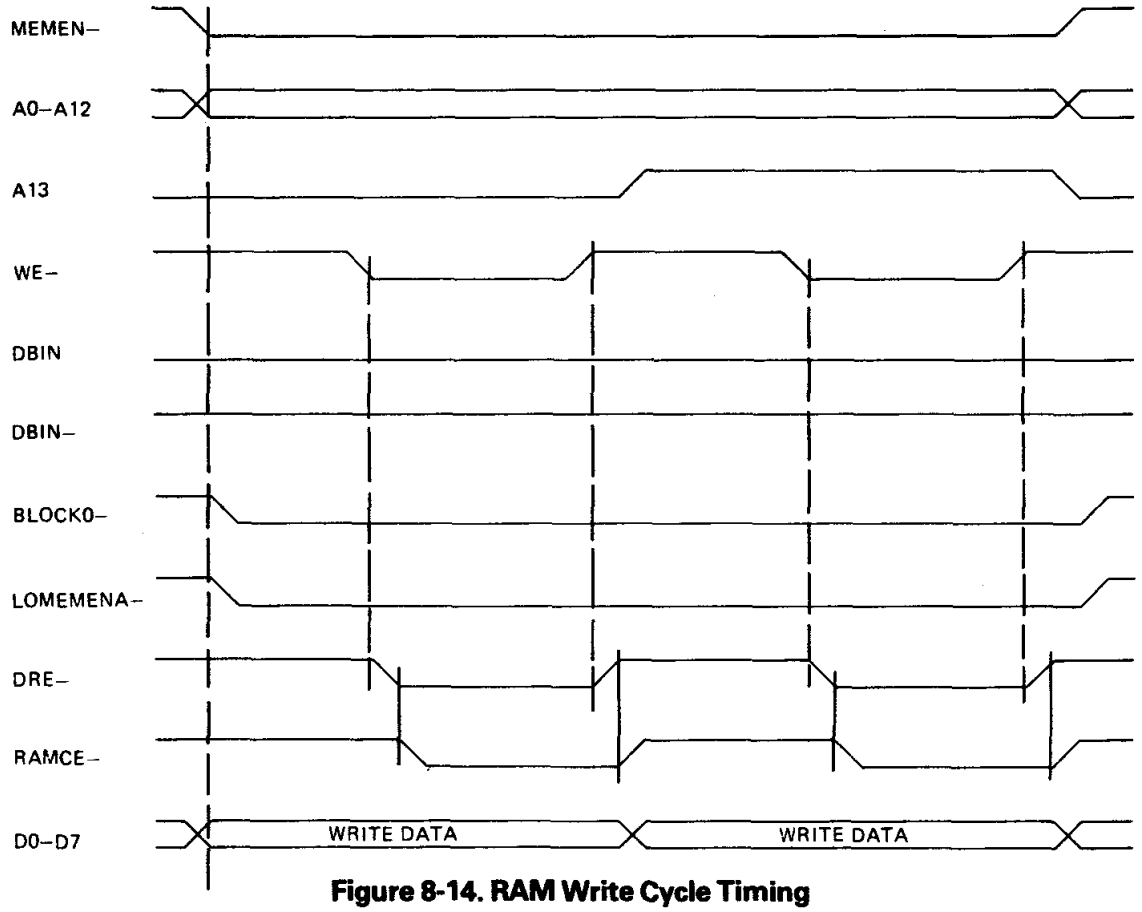


Figure 8-14. RAM Write Cycle Timing

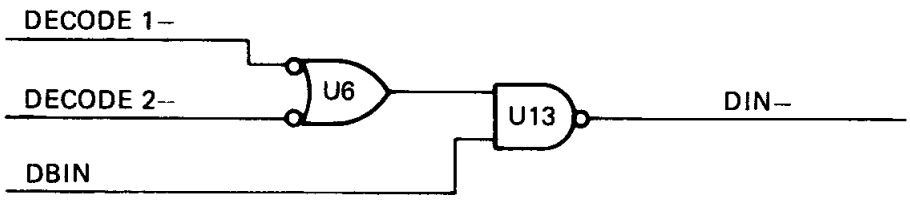


Figure 8-15. Expansion Data Bus Control Logic

8.6 INPUT-OUTPUT

Input-output operations are performed via the Communications Register Unit (CRU). The TMS 9980A CRU addresses a total of 2048 input-output bits effecting data transfers of from one to sixteen bits.

NOTE

In this write-up, reference is made to R12 CRU address and to CRU base address. CRU base address is the actual bit being addressed and is the value on address lines A_2 to A_{12} during a CRU operation, as explained in Section 7. The R12 CRU address is the contents of register 12 during a CRU operation and is used by the processor to determine the address bus contents during the CRU operation. The CRU base address is the R12 CRU address multiplied by two (or the R12 CRU base address shifted left one bit) as described in Section 7.

Onboard I/O for the TM 990/189 is defined as four blocks of 32 bits as shown in the System CRU Map, Figure 8-16. Each block (or port) is enabled by a separate signal (USERIOCE—, KBIOCE—, COMIOCE—, or EXTCRUENA) decoded from the address bus as shown in Figure 8-17. CRUENA— insures that each 32-bit block is unique (i.e., appears only once in the CRU map) by requiring that address lines A4 through A7 be low for CRU operations. A8 through A12 route to the individual CRU devices to select the individual bit being referenced while A2 and A3 are decoded to BLOCK 0— through BLOCK 3— and gated with CRUENA— to generate the four enable signals.

8.6.1 USER I/O PORT

The User I/O Port extends from R12 CRU address 000_{16} to $03E_{16}$ and consists of a TMS 9901 (U10) which provides 16 bidirectional signal lines available to the user through connector P5. The low order four bits (R12 CRU addresses 020 through 026) are also connected to the drivers for light emitting diodes CR1 through CR4 which illuminate in response to a logic one input. The individual bit assignments of the User Port are shown in Table 8-5.

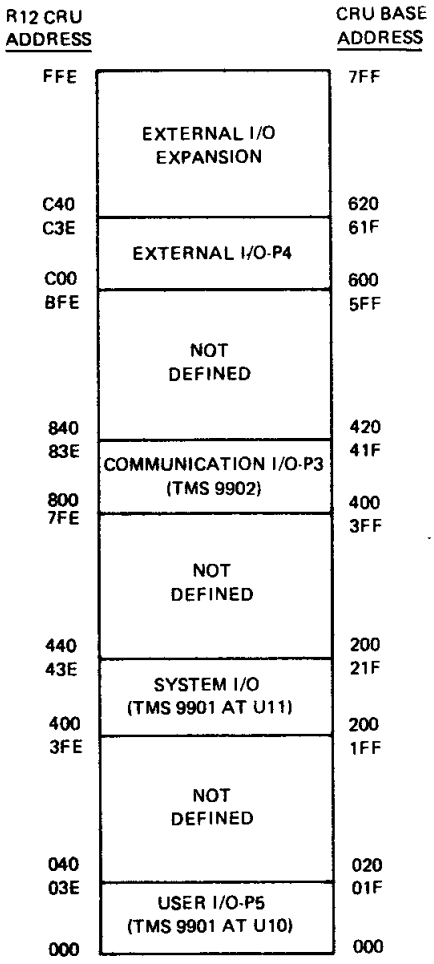


Figure 8-16. System CRU Map

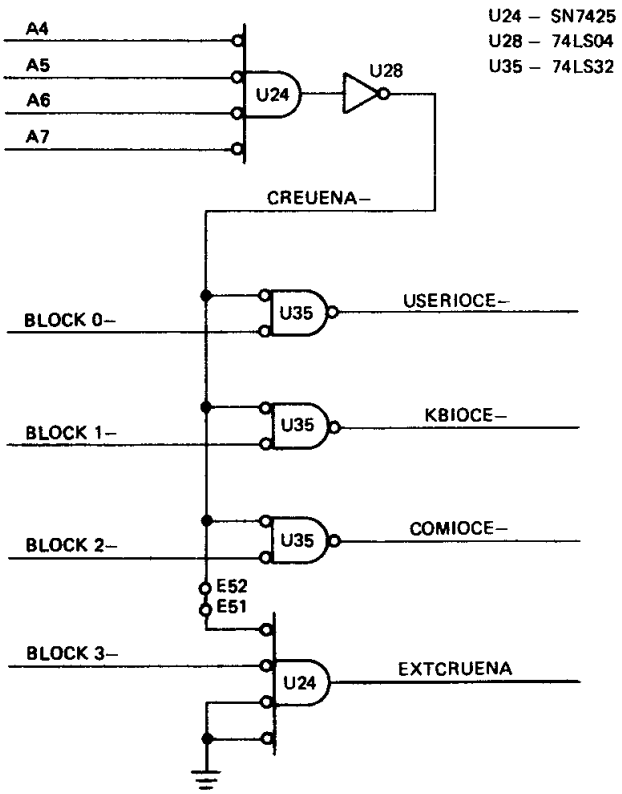


Figure 8-17. CRU Address Decoding Logic

TABLE 8-5. USER PORT I/O MAP

R12 CRU ADDRESS	TMS 9901 BIT ASSIGNMENT			SIGNAL LINE AFFECTED
	BIT ADDRESSED	INPUT	OUTPUT	
000	0	CONTROL BIT	CONTROL BIT	
002	1	INT1—, CLK 1	MASK 1, CLK 1	UINT 1—
004	2	INT2—, CLK 2	MASK 2, CLK 2	UINT 2—
006	3	INT3—, CLK 3	MASK 3, CLK 3	UINT 3—
008	4	INT4—, CLK 4	MASK 4, CLK 4	UINT 4—
00A	5	INT5—, CLK 5	MASK 5, CLK 5	UINT 5—
00C	6	INT6—, CLK 6	MASK 6, CLK 6	KBINT—
00E	7	INT7—, CLK 7	MASK 7, CLK 7	USER P15
010	8	INT8—, CLK 8	MASK 8, CLK 8	USER P14
012	9	INT9—, CLK 9	MASK 9, CLK 9	USER P13
014	10	INT10—, CLK 10	MASK 10, CLK 10	USER P12
016	11	INT11—, CLK 11	MASK 11, CLK 11	USER P11
018	12	INT12—, CLK 12	MASK 12, CLK 12	USER P10
01A	13	INT13—, CLK 13	MASK 13, CLK 13	USER P9
01C	14	INT14—, CLK 14	MASK 14, CLK 14	USER P8
01E	15	INT15—, INTREQ—	MASK 15, RST 2—	USER P7
020	16	P0 INPUT	P0 OUTPUT	USER P0
022	17	P1 INPUT	P1 OUTPUT	USER P1
024	18	P2 INPUT	P2 OUTPUT	USER P2
026	19	P3 INPUT	P3 OUTPUT	USER P3
028	20	P4 INPUT	P4 OUTPUT	USER P4
02A	21	P5 INPUT	P5 OUTPUT	USER P5
02C	22	P6 INPUT	P6 OUTPUT	USER P6
02E	23	P7 INPUT	P7 OUTPUT	USER P7
030	24	P8 INPUT	P8 OUTPUT	USER P8
032	25	P9 INPUT	P9 OUTPUT	USER P9
034	26	P10 INPUT	P10 OUTPUT	USER P10
036	27	P11 INPUT	P11 OUTPUT	USER P11
038	28	P12 INPUT	P12 OUTPUT	USER P12
03A	29	P13 INPUT	P13 OUTPUT	USER P13
03C	30	P14 INPUT	P14 OUTPUT	USER P14
03E	31	P15 INPUT	P15 OUTPUT	USER P15

8.6.2 SYSTEM I/O PORT

The System I/O Port occupies CRU addresses 400_{16} through $43E_{16}$ and consists of another TMS 9901 (U11) dedicated to onboard devices such as the keyboard, display, sound disc, and the audio cassette interface. Individual bit assignments for the system port are shown in Table 8-6.

8.6.2.1 Keyboard and Display Interface

Signal flow between the TMS 9901 and keyboard and display is diagrammed in Figure 8-18. UNIBUG software routines scan both the keyboard and display thereby minimizing the hardware required to drive the 80 display segments and read the 45 keyswitches.

The display is a twelve digit common cathode seven segment L.E.D. type of which the middle ten digits are used. Used digits are numbered left to right from 1 to 10. Segments within a digit are designated as shown in Figure 8-19.

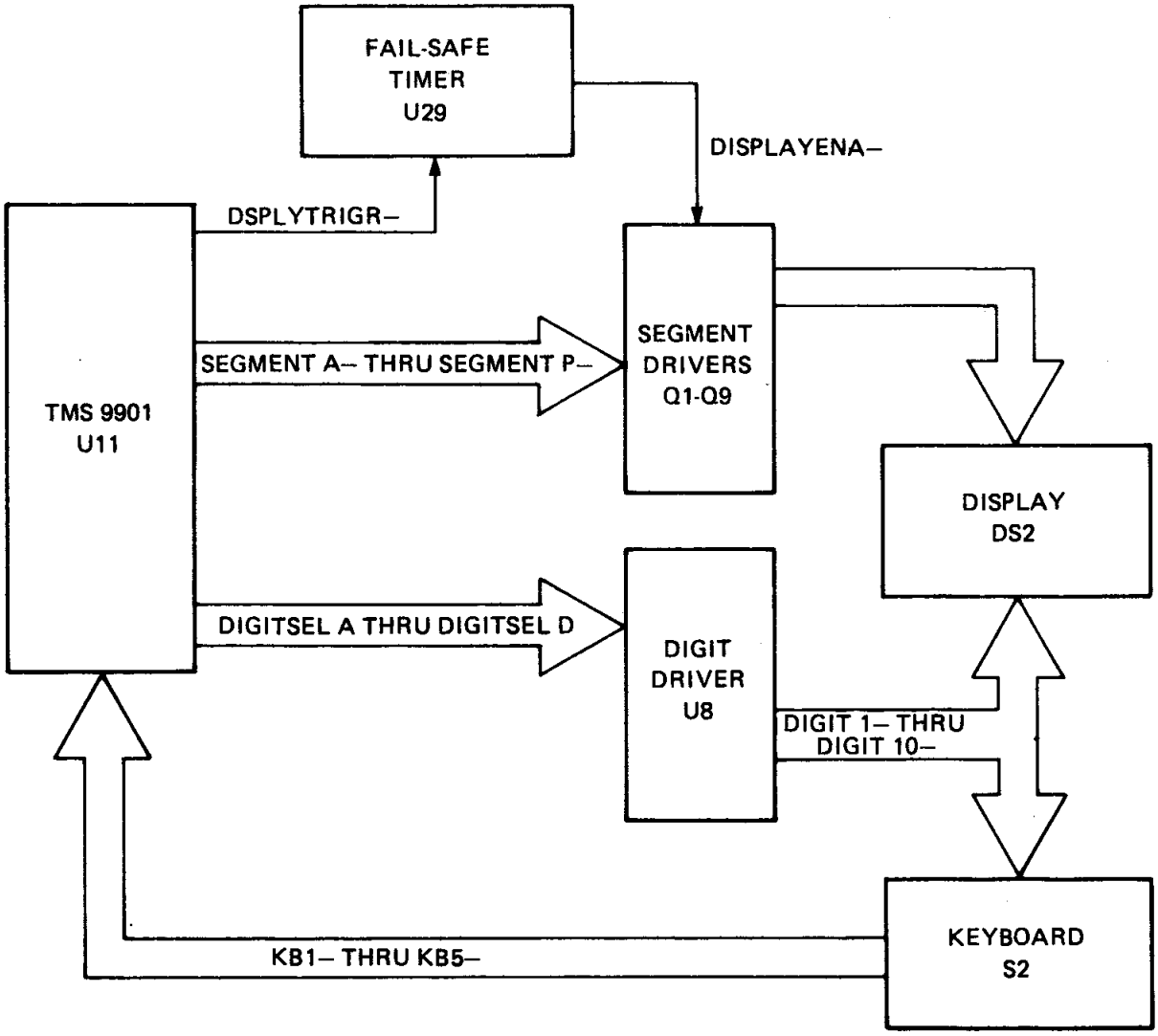


Figure 8-18. Keyboard and Display Interface Block Diagram

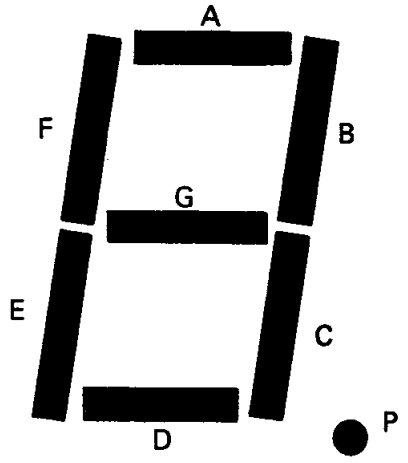


Figure 8-19. Display Segment Designation

TABLE 8-6. SYSTEM PORT I/O MAP

R12 CRU ADDRESS	TMS 9901 BIT ASSIGNMENT			SIGNAL LINE AFFECTED
	BIT ADDRESSED	INPUT	OUTPUT	
400	0	CONTROL BIT	CONTROL BIT	
402	1	INT1—, CLK 1	MASK 1, CLK 1	KB1—
404	2	INT2—, CLK 2	MASK 2, CLK 2	KB2—
406	3	INT3—, CLK 3	MASK 3, CLK 3	KB3—
408	4	INT4—, CLK 4	MASK 4, CLK 4	KB4—
40A	5	INT5—, CLK 5	MASK 5, CLK 5	KB5—
40C	6	INT6—, CLK 6	MASK 6, CLK 6	RDATA
40E	7	INT7—, CLK 7	MASK 7, CLK 7	
410	8	INT8—, CLK 8	MASK 8, CLK 8	
412	9	INT9—, CLK 9	MASK 9, CLK 9	
414	10	INT10—, CLK 10	MASK 10, CLK 10	
416	11	INT11—, CLK 11	MASK 11, CLK 11	
418	12	INT12—, CLK 12	MASK 12, CLK 12	
41A	13	INT13—, CLK 13	MASK 13, CLK 13	
41C	14	INT14—, CLK 14	MASK 14, CLK 14	
41E	15	INT15—, INTREQ—	MASK 15, RST 2—	
420	16	P0 INPUT	P0 OUTPUT	DIGITSEL A
422	17	P1 INUT	P1 OUTPUT	DIGITSEL B
424	18	P2 INPUT	P2 OUTPUT	DIGITSEL C
426	19	P3 INPUT	P3 OUTPUT	DIGITSEL D
428	20	P4 INPUT	P4 OUTPUT	SEGMENT A—
42A	21	P5 INPUT	P5 OUTPUT	SEGMENT B—
42C	22	P6 INPUT	P6 OUTPUT	SEGMENT C—
42E	23	P7 INPUT	P7 OUTPUT	SEGMENT D—
430	24	P8 INPUT	P8 OUTPUT	SEGMENT E—
432	25	P9 INPUT	P9 OUTPUT	SEGMENT F—
434	26	P10 INPUT	P10 OUTPUT	SEGMENT G—
436	27	P11 INPUT	P11 OUTPUT	SEGMENT P—
438	28	P12 INPUT	P12 OUTPUT	DSPLYTRIGR—
43A	29	P13 INPUT	P13 OUTPUT	SHIFTLIGHT
43C	30	P14 INPUT	P14 OUTPUT	SPKRDRIVE
43E	31	P15 INPUT	P15 OUTPUT	WDATA

Display segment and digit driver circuits are shown in Figure 8-20. Current is sourced to all A segment anodes by pulling SEGMENT A— low, provided DISPLAYENA— is also low. Current is sunk from all digit 1 cathodes by pulling DIGIT 1— low, which is accomplished through U8 (a 74LS145, one of ten decoder) by placing an all zero code on DIGITSEL A through DIGITSEL B. The remaining segments and digits are driven in a similar manner. All L.E.D.'s between sourced anode connections and sunk cathode connections will be illuminated.

While scanning the display does reduce the amount of display drive circuitry needed, it also reduces each segment's duty cycle, and hence, its apparent brightness. To counteract this effect, the segment currents are set much higher than normally required for continuous operation. The resulting high-intensity pulse is perceived by the eye to be of normal brightness. The increased segment current does not harm the display because the average power dissipated in each segment is still within normal limits. However, should normal program control ever be lost, it is possible that segments could remain illuminated for extended periods resulting in display damage. The fail-safe timer (U29) guards against that possibility by requiring constant reassurance from the scan routine that program control is being maintained. That reassurance is provided by periodic pulsing of the DSPLYTRIGR— line. Loss of that drive signal allows the timer to expire and drives DISPLAYENA— high disabling the entire display.

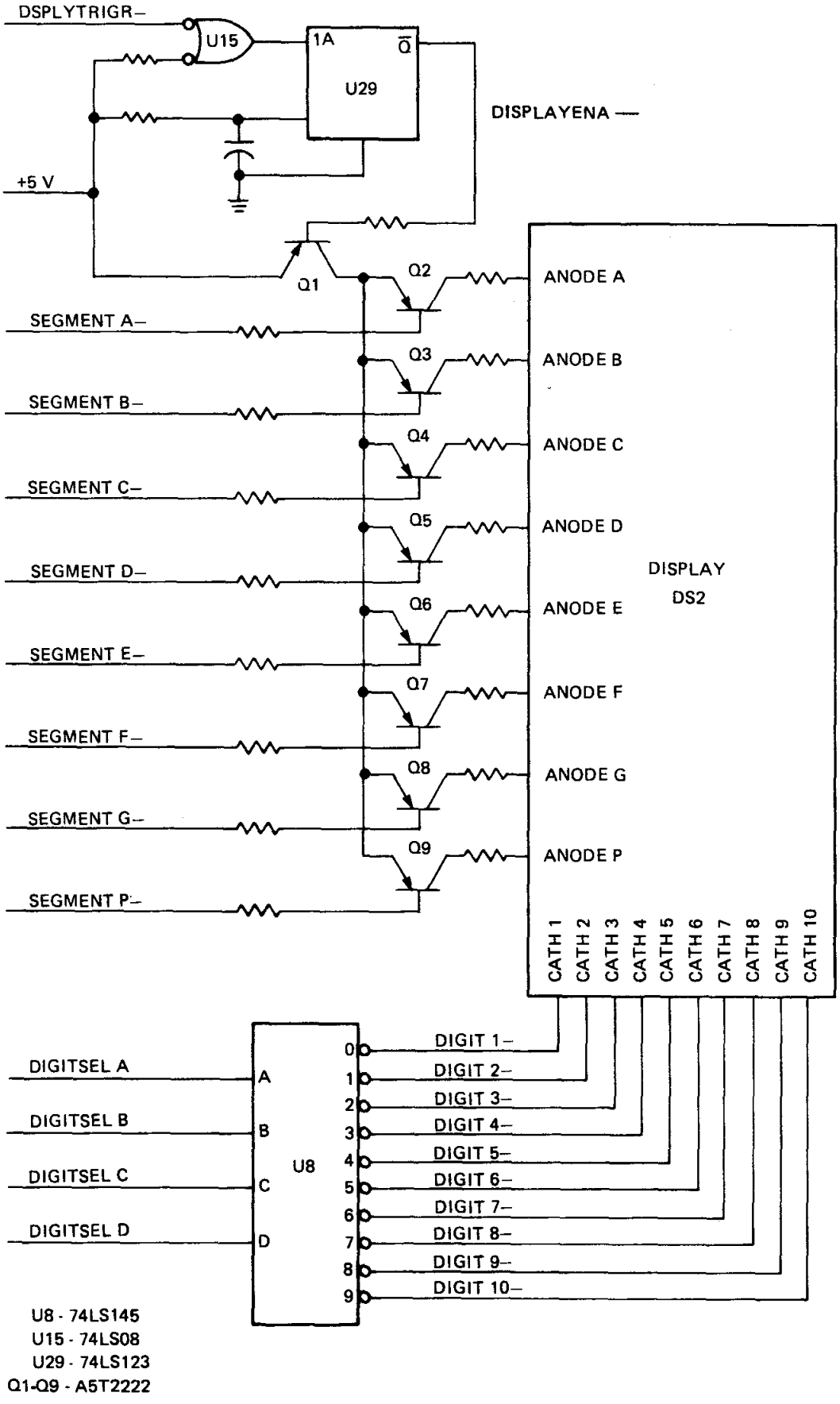


Figure 8-20. Display Driver Circuitry

The keyboard is composed of 45 normally open switches connected in a matrix between 9 rows and 5 columns as shown in Figure 8-21. Keyboard scanning is normally performed in conjunction with the display scan. As mentioned above, digits are enabled by outputting a four bit select code on DIGITSEL A through DIGITSEL D which is decoded to ten active low enable lines. Nine of those lines (DIGIT 1— through DIGIT 9—) are routed to the keyboard row lines. Once a display digit is selected and the corresponding keyboard row line enabled, key closures on that row are detected by reading the state of the keyboard column lines KB1— through KB5—. Since the column lines have pull-up resistors and therefore normally float at a logic one, a low level on a column line indicates a key closure to the enabled row line.

The column lines KB1— through KB5— are read through U11 using another unique feature of the TMS 9901. Since the device contains an interrupt mask register, activity on the interrupt inputs may be prevented from causing interrupt code generation. However, the actual state of the interrupt input lines may still be read using the TB and STCR instructions. Therefore, with interrupts masked, the dedicated interrupt request pins (INT 1— through INT 6—) may be used as ordinary input pins for reading data. The state of the keyboard column lines is then read by interrogating the INT 1— through INT 5— bits in U11.

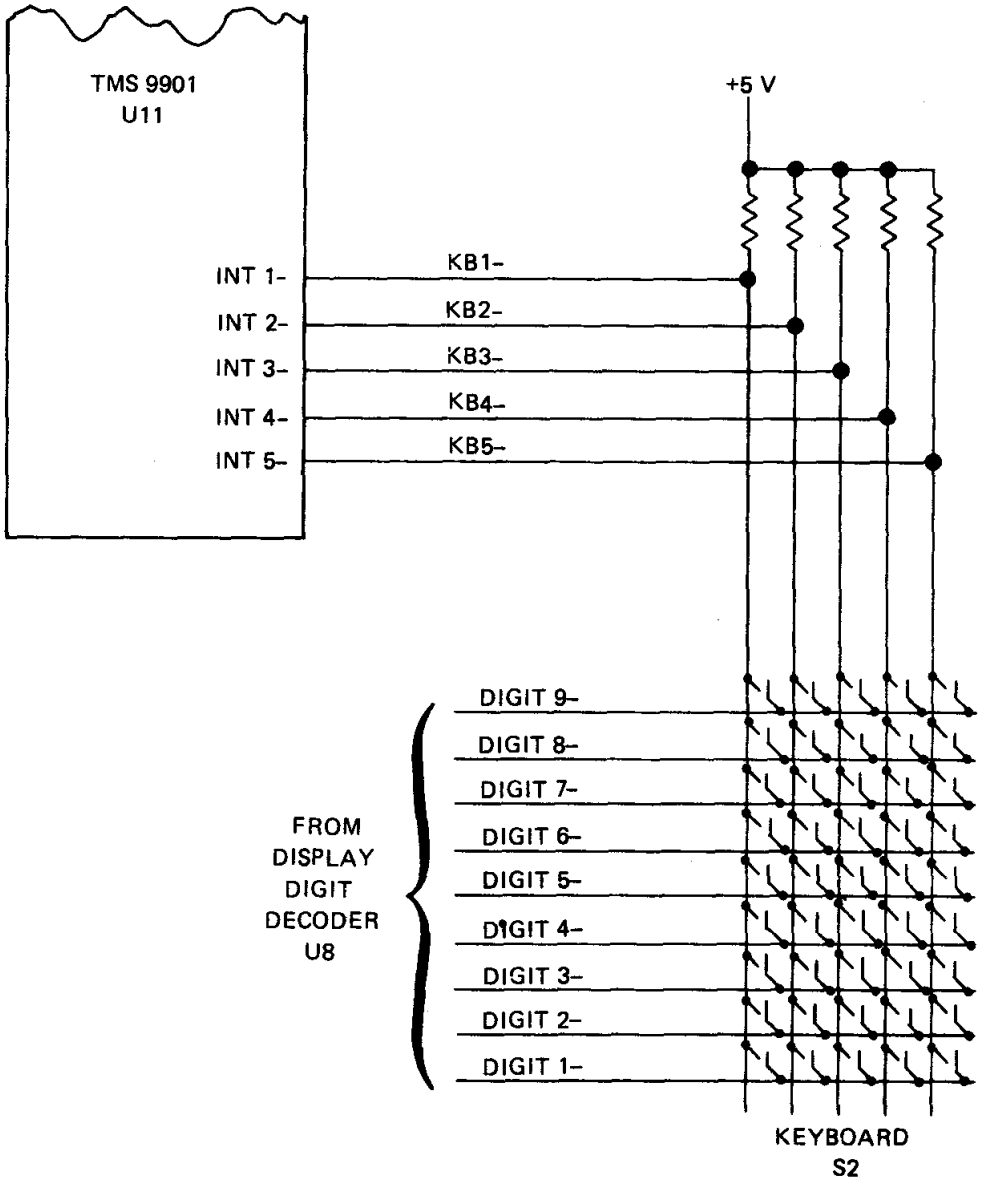


Figure 8-21. Keyboard Interface

8.6.2.2 Audio Cassette Interface

The Audio Cassette Interface allows the user to read and write digital data on a voice-quality cassette or reel recorder. The purpose of the interface is to translate the TTL level data signals to recorder-compatible signal levels, and vice versa. The actual generation and interpretation of the data streams is done by the processor under software control.

The UNIBUG monitor encodes data on tape using a frequency-shift keying (FSK) technique where a logic zero is represented by two complete cycles at 1200 Hz and a logic one is represented by two cycles at 2400 Hz. UNIBUG computes the required signal frequency based on the data bit to be written and outputs an approximate square wave of that frequency on WDATA (P15 of U11).

The WDATA signal is then shifted to tape-compatible levels by the write circuit shown in Figure 8-22. Buffered by open-collector gate U7, the signal is then attenuated by the voltage divider composed of R45 and R46 to approximately 500 millivolts peak to peak. A low pass filter consisting of R47 and C10 then removes the higher frequency components and presents the rounded square wave signal to the recorder. Notice that the logical inversion of the signal by U7 is unimportant since the information is contained in the frequency of the signal and not its magnitude.

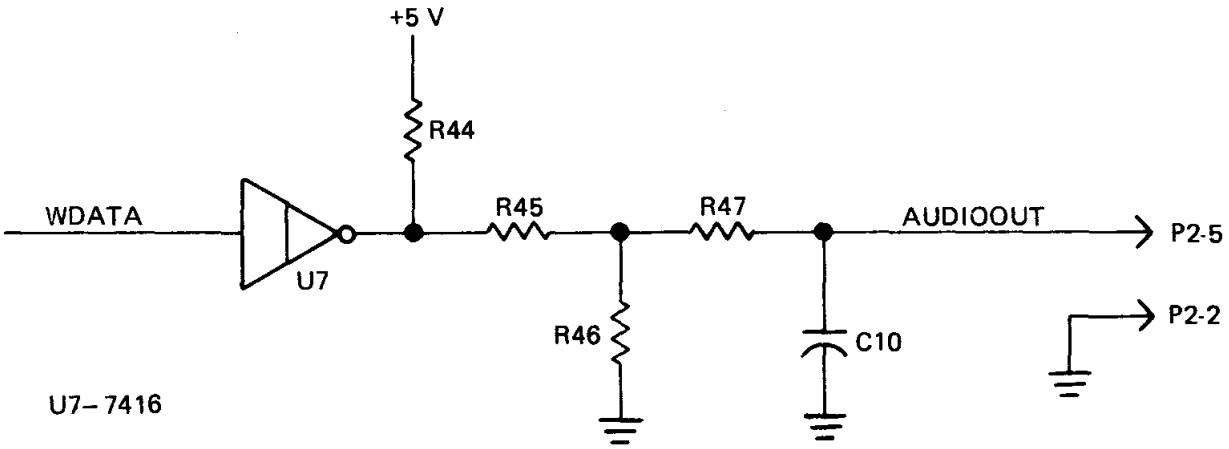


Figure 8-22. Cassette Write Circuit

Playback signals from the recorder are conditioned by the read circuit shown in Figure 8-23. Resistor R42 places a light load on the AUDIOIN signal line to reduce effects from induced noise while C9 blocks any DC component of the audio signal. A magnitude limiting stage composed of R43, U38, CR10, and CR11 then clips the signal to a peak level of one diode drop, or approximately 600 millivolts. It is then amplified by a gain stage consisting of the remainder of U38, R40, and R41 and squared by Schmitt trigger U36 to produce RDATA which is read by U11. R30 is used to slightly modify the normal threshold voltages of the device to center its hysteresis characteristics more about zero volts.

Observe that due to the double inversion of the signal, once by the limiter and again by the Schmitt trigger, RDATA has the same voltage polarity as AUDIOIN. R50 is used to supply a small amount of positive feedback to the input of the limiter and give the entire circuit the characteristics of a Schmitt trigger. R51, together with R50, sets the overall circuit's threshold voltages and determines its hysteresis.

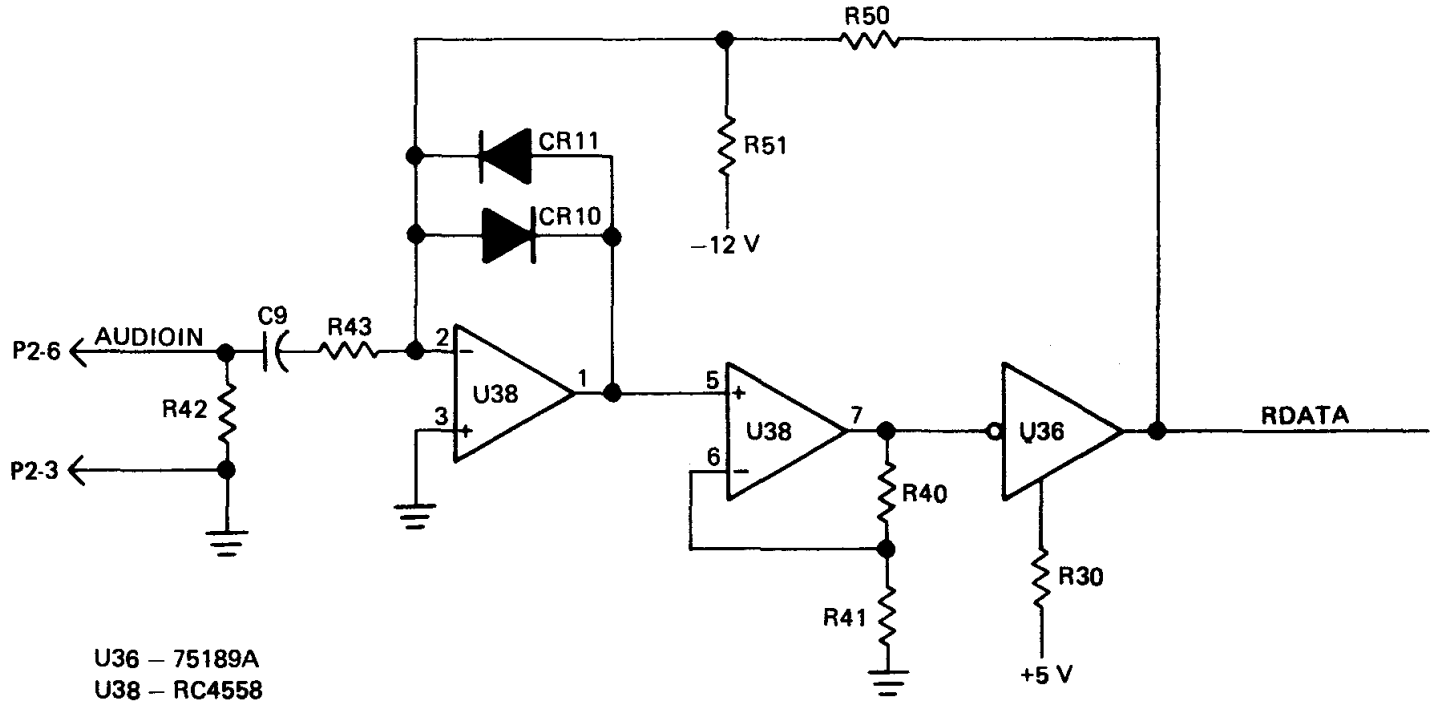


Figure 8-23. Cassette Read Circuit

8.6.2.3 *Sound Disc*

The Sound Disc (DS1) is a piezoelectric crystal which flexes when an electric potential is applied across it. The SPKRDRIVE signal (P14 of U11) is amplified by Q10 and applied to the disc which generates a tone of frequency equal to the SPKRDRIVE signal.

8.6.3 SERIAL COMMUNICATIONS PORT

The Serial Communications Port occupies CRU addresses 800_{16} through $83E_{16}$ and consists of a TMS 9902 Asynchronous Communications Controller and interface devices for conversion of signals to RS-232-C and 20 milliampere current loop levels. The TMS 9902 receives data from the CR4 and generates the timing and handshaking signals necessary to send the data to a data terminal such as a printer or video display. In addition, the TMS 9902 receives data from the terminal and buffers it for presentation to the CRU. The bit assignments of the Serial Communications Port are shown in Table 8-7. These signals can be brought out to an optional 25-pin EIA connector installed by the user at P3.

TABLE 8-7. COMMUNICATIONS I/O MAP

R12 CRU ADDRESS	BIT ADDRESSED	TMS 9901 BIT ASSIGNMENT INPUT	OUTPUT	SIGNAL LINE AFFECTED
800	0	RBR 0	*	
802	1	RBR 1	*	
804	2	RBR 2	*	
806	3	RBR 3	*	
808	4	RBR 4	*	
80A	5	RBR 5	*	
80C	6	RBR 6	*	
80E	7	RBR 7	*	
810	8	0	*	
812	9	RCVERR	*	
814	10	RPER	*	
816	11	ROVER	LXDR	
818	12	RFER	LRDR	
81A	13	RFBD	LDIR	
81C	14	RSBD	LDCTRL	
81E	15	RIN	TSTMD	RCVDATA
820	16	RBINT	RTSON	RST—
822	17	XBINT	BRKON	XMTDATA
824	18	0	RIENB	
826	19	TIMINT	XRIENB	
828	20	DSCINT	TIMENB	
82A	21	RBRL	DSCENB	
82C	22	XBRE		
82E	23	XSRE		
830	24	TIMERR		
832	25	TIMELP		
834	26	RTS		RTS—
836	27	DSR		DSR—
838	28	CTS		RTS—
83A	29	DSCH		
83C	30	FLAG		
83E	31	INT	RESET	

*Control, Interval, Receive Data Rate, Transmit Data Rate, and Transmit Buffer Registers depending on Register Load Control Flags (bits 14-11). See TMS 9902 Data Manual in Appendix F.

8.6.4 EXTERNAL I/O PORT

The External I/O Port allows the user to expand his input-output capability offboard. Signals provided through the Bus Expansion Interface include the address lines, A13 CRUOUT, CRUIN, IOCLK, and EXPCRUENA and are available at connector P4. Signal $\phi 3-$ is alos provided to facilitate I/O expansion using TMS 9900 family I/O devices. P inputs from external CR4 devices are enabled by EXPCRUENA (see Figure 8-24) at R12 CRU addresses $C00_{16}$. A jumper option is provided to allow expansion of external I/O to R12 CRU addresses $C00_{16}$ through FFE_{16}.

Outputs to external CRU devices are determined by decode logic constructed by the user, typically being a function of EXPCRUENA and address lines A8 through A12.

The external CRU bus outputs, like the address and data bus lines, enter a high impedance state in response to the HOLDA signal from the processor.

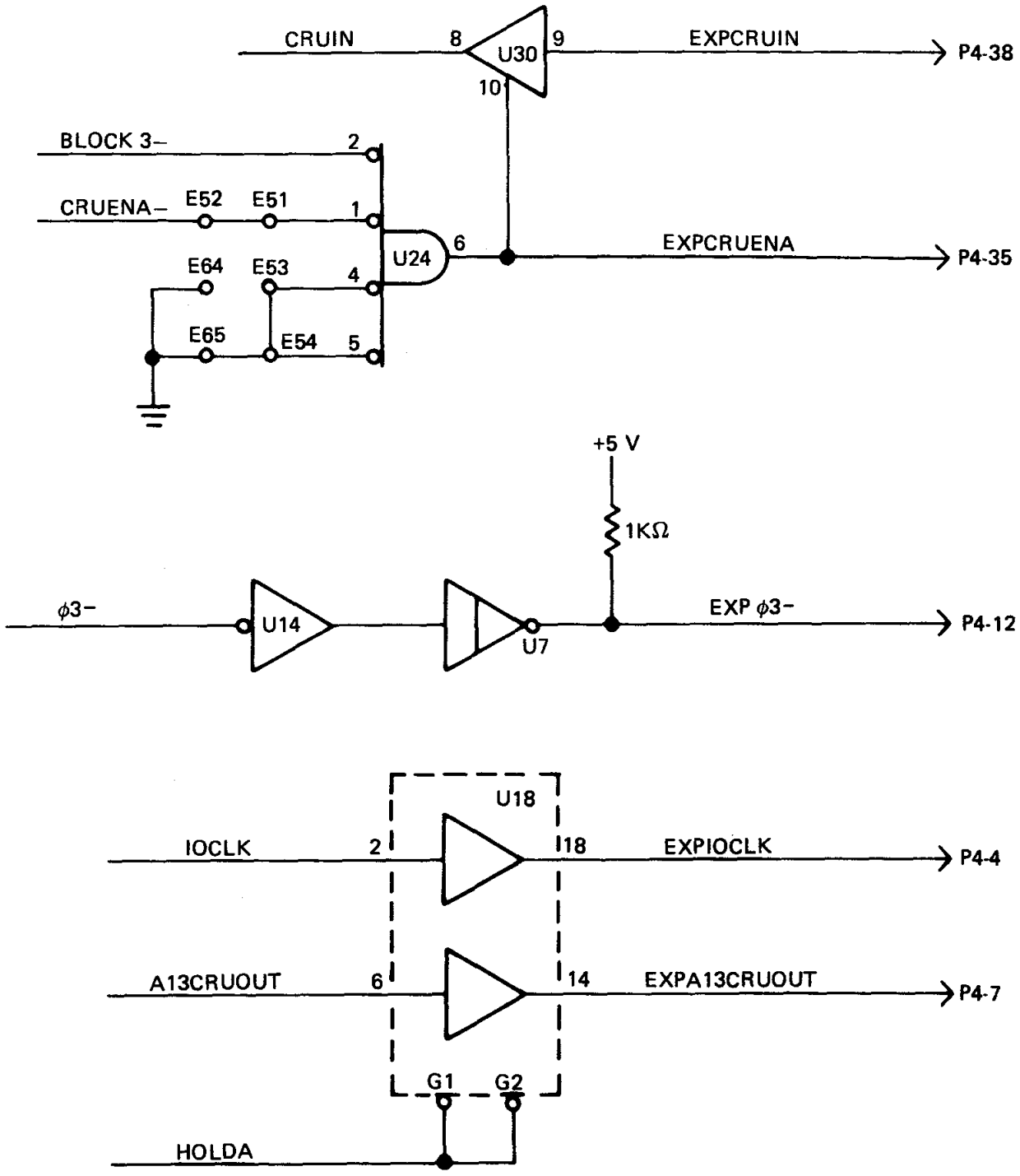


Figure 8-24. External I/O Port

SECTION 9

USER OPTIONS

9.1 INTRODUCTION

Users wishing to expand the capability of their TM 990/189 are encouraged to make use of the numerous user options already provided on the board. This section describes those options and details procedures for their incorporation. Figure 9-1 shows the layout of the TM 990/189 board.

9.2 MEMORY OPTIONS

9.2.1 ON-BOARD RAM EXPANSION

On-board RAM may be expanded to 2048 bytes by installing memory devices as shown below. User RAM will then extend from memory addresses >0000 to >07FF.

QTY	DESCRIPTION	INSTALL AT
2	TMS 4014 OR EQUIVALENT (TMS 4045) *	U21, U23

9.2.2 ON-BOARD EPROM, 1K BYTES

1024 bytes of EPROM may be added to the board by installing the device listed below. Circuitry is already provided to map the EPROM at memory addresses >0800 to >0BFF.

QTY	DESCRIPTION	INSTALL AT
1	TMS 2708	U32

9.2.3 ON-BOARD EPROM, 2 K BYTES

2048 bytes of EPROM may be added to the board by modifying the board per the procedure below and installing the device below. Following the modification, the EPROM will be mapped at memory addresses >0800 to >0FFF.

QTY	DESCRIPTION	INSTALL AT
1	TMS 2716	U32

Modification Procedure:

a. Cut the conductor trace between E38 and E39.

b. Cut the trace between E40 and E41.

c. Add a jumper wire between E39 and E40.

d. Add a jumper between E41 and E42.

e. Add a jumper between E49 and E50.

*The TMS 4045 RAM (which can tolerate up to 10 percent power supply noise) can be substituted for the TMS 4014 RAM (which can tolerate up to 5 percent power supply noise).

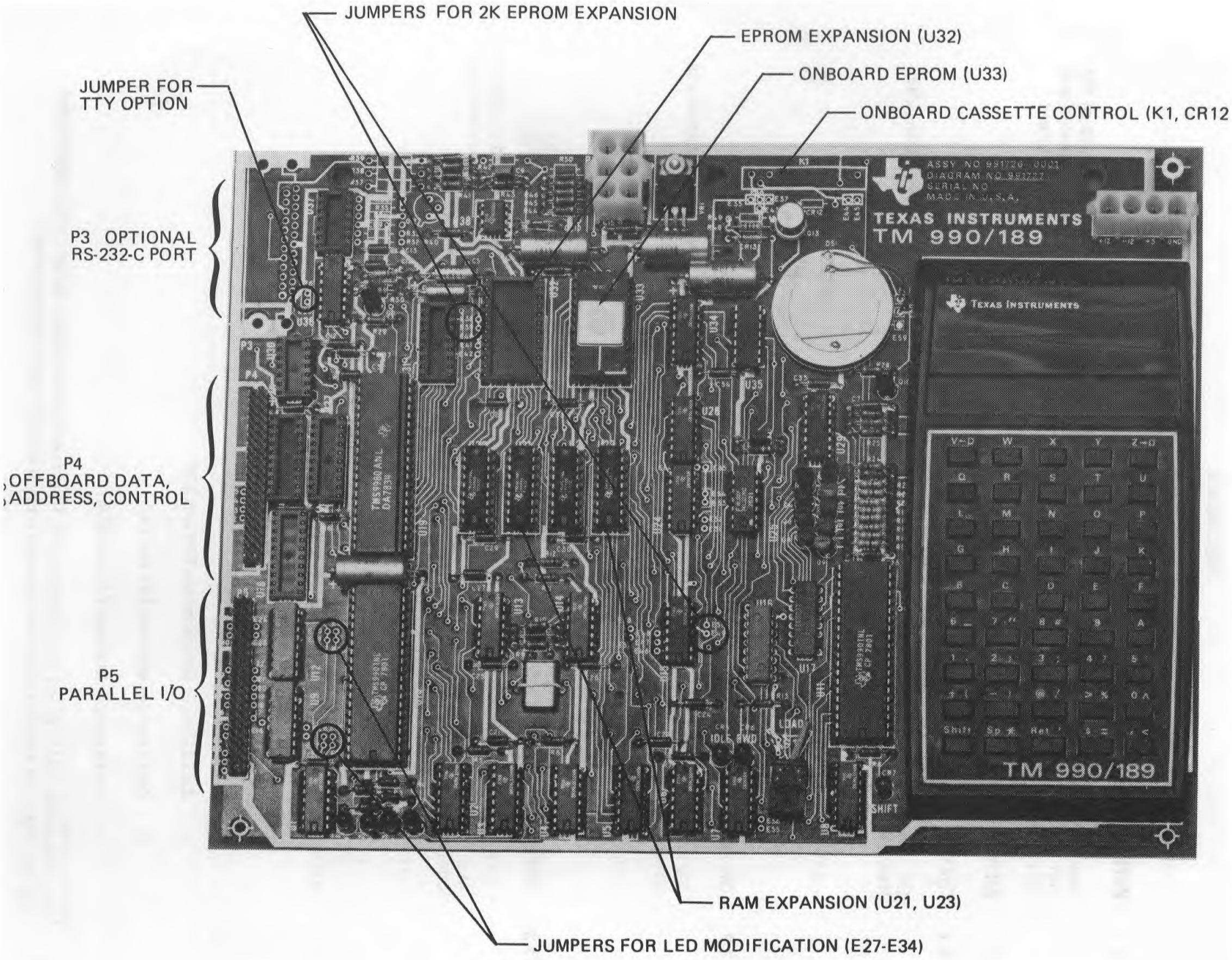


Figure 9-1. TM 990/189 Board Layout

9.2.4 ON-BOARD EPROM, 4K BYTES

4096 bytes of EPROM may be substituted for the system ROM by removing the device at U33 and replacing it with EPROM listed below:

QTY	DESCRIPTION	INSTALL AT
1	TMS 2532	U33

While sacrificing the use of the UNIBUG monitor and assembler, this substitution provides the capability for installing resident user programs of up to 4096 bytes on the board. This capability may be expanded to 6144 bytes by installing an additional 2K byte EPROM per Section 9.1.3. Typical applications of this option include user-written operating systems and applications programs. The 4K byte EPROM is mapped at memory addresses >3000 to >3FFF.

9.2.5 OFF-BOARD MEMORY EXPANSION

Up to 8192 bytes of user-defined off-board memory or memory-mapped I/O may be used, in addition to the on-board memory described above, by installing the Bus Expansion Interface components per Table 9.1. Signals provided through the interface include the address bus, data bus, and memory control signals. These signals are available at connector port P4 as shown on sheet 11 of the schematics in Appendix A.

TABLE 9-1. LIST OF MATERIALS, OFFBOARD MEMORY EXPANSION

ITEM	QTY	DESCRIPTION	INSTALL AT
1	1	74LS126	U30
2	2	74LS244	U18, U26
3	1	74LS245	U27
4	1	RES, 10K, 5%, 0.25W	R34

9.3 INPUT-OUTPUT OPTIONS

9.3.1 ASYNCHRONOUS COMMUNICATIONS, EIA

The capability to perform asynchronous serial communications with devices meeting the requirements of RS-232-C may be added to the board by installing Serial Communication Interface components per Table 9-2. The interface occupies CRU base addresses >0800 through >083E. When installed, signals are available at 25-pin connector P3 as shown in sheet 10 of the schematics (Appendix A). This sheet also shows the interconnection of the components in Table 9-2.

TABLE 9-2. LIST OF MATERIAL, EIA OPTION

ITEM	QTY	DESCRIPTION	INSTALL AT
1	1	TMS 9902	U31
2	1	75188	U37
3	1	RES, 10KΩ, 10%, 0.25W	R34
4	2	RES, 3.3KΩ, 10%, 0.25W	R35, R36
5	1	Connector, AMP 206584-2 or CANNON DBP-25S-AA	P3

9.3.2 ASYNCHRONOUS COMMUNICATIONS, TTY

The capability to perform asynchronous serial communications with devices having a 20 mA current loop interface may be added by modifying the board as described below and installing Serial Communications Interface components per Table 9-3. When installed, signals will be available at 25-pin connector P3 as shown on sheet 10 of the schematics in Appendix A. This sheet also shows the components in Table 9-3. The serial interface occupies R12 CRU base addresses >0800 through >083E.

Modification Procedure:

Add a jumper wire between E25 and E26.

TABLE 9-3. LIST OF MATERIALS, TTY OPTION

ITEM	QTY	DESCRIPTION	INSTALL AT
1	1	TMS 9902	U31
2	1	75188	U37
3	1	2N2905	Q12
4	1	1N914B	CR9
5	1	RES, 33KΩ, 10%, 0.25W	R32
6	1	RES, 3.3KΩ 10%, 0.25W	R33
7	1	RES, 10KΩ, 10%, 0.25W	R34
8	1	RES, 330Ω, 10%, 0.5W	R37
9	1	RES, 560Ω, 10%, 0.5W	R38
10	1	RES, 2.7KΩ 10%, 0.25W	R39
11	1	CONNECTOR, AMP 206584-2 or CANNON DBP-25S-AA	P3

9.3.3 ON-BOARD CASSETTE RELAY

Control of the cassette deck motors may be performed by the TM 990/189 through the installation of a on-board relay and transient protection diode as listed below, and connection of P2 pins 1 and 4 to the deck's "REMOTE" jack.

ITEM	QTY	DESCRIPTION	INSTALL AT
1	1	1N914B	CR12
2	1	RELAY, COSAR 1112-12-1A*	K1

*Cosar, 3121 Benton St., Garland, Texas 75042

CAUTION

Operation with cassette models other than those listed in paragraph 2.7.1 may not yield reliable data transfers and certain models of recorders can *damage* the TM 990/189 control relay due to excessive inrush currents.

9.3.4 OFF-BOARD CASSETTE RELAY

If the user wishes to control the cassette motors by a means other than the on-board relay described in Section 9.3.3, jumper options make available the open-collector drive signal DECKCONTROL— along with connections to +12 volts, +5 volts, and GROUND. Refer to page 12 of the logic diagram in Appendix A for connection details. Care should be exercised to avoid exceeding the maximum output voltage and current ratings of the 7416 which drives the DECKCONTROL— signal. Those ratings are +25V and 40 mA respectively.

9.3.5 OFF-BOARD CRU EXPANSION

In addition to the three on-board devices comprising the User I/O Port, keyboard I/O Port, and Serial Communications Interface, CRU devices may be expanded off-board through the Bus Expansion Interface by installing the components listed below.

ITEM	QTY	DESCRIPTION	INSTALL AT
1	1	74LS126	U30
2	2	74LS244	U18, U26

This CRU expansion uses the buffered outputs of address lines A2 through A12 and the A13 CRUOUT line with a 74LS244 installed in U18 and U26. This is shown in sheet 11 of the schematics in Appendix A.

CRU output operations may be performed at any CRU base (bit) address from $>000_{16}$ through $XEFF_{16}$ (R12 CRU address 000_{16} through FFE_{16}) by externally decoding address bus lines A_2 to A_{12} and writing the data present on the EXPA13CRUOUT line at that location. The user should be aware that output operations to offboard devices at R12 CRU address >000 through 03E, >400 through >43E, and >800 through >83E will also cause transfers to the onboard devices at those same locations which may result in unexpected actions being taken. Figure 8-12 is a map of the CRU addressing scheme.

CRU input operations from off-board devices are controlled by the EXPCRUENA signal which defines a block of 32 bits extending from R12 CRU address>C00 through>C3E. Derivation of this signal is shown on sheets 2 and 11 of the schematics.

Capability to input from off-board devices may be expanded to a block of 512 bits extending from R12 CRU address >C00 through >FFE. Expansion is accomplished by performing the modification procedure below.

Modification Procedure for expanding off-board CRU input capability to 512 bits:

a. Cut the conductor trace between E51 and E52.

b. Add a jumper wire between E51 and E65.

After installation, data bus, address bus, and control signals are available at port P4 as shown on sheet 11 of the schematics in Appendix A.

9.3.6 ON-BOARD LED DISCONNECTION

Each TM 990/189 is supplied with LED indicators CR1 through CR4 driven by a 7416 open-collector inverter connected to signals USERP3 through USERP0 respectively, the least significant four bits of the User I/O Port. Thus, each signal USERP3 through USERP0 already drives one TTL load. This is shown on sheet 7 of the schematics, Appendix A.

To avoid exceeding the output drive capability of U10 (TMS 9901) when connecting the User I/O Port to offboard circuitry, the user may find it necessary to disconnect the one TTL load established by the LED driver. Disconnection is accomplished by performing the modifications described in Table 9-4.

TABLE 9-4. LED DISCONNECTION MODIFICATION

TO REMOVE LOAD ON	CUT TRACE BETWEEN
User P0	E27, E28
User P1	E31, E32
User P2	E33, E34
User P2	E33, E34
User P3	E29, E30

9.4 PROCESSOR OPTIONS

9.4.1 COMMUNICATIONS INTERRUPTS

Provision is made to allow the user to connect the interrupt request signal from the TMS 9902 communications controller (COMINT—) to one of five interrupt inputs to the TMS 9901 system interrupt port (INT1— to INT5—). This connection allows the user to utilize the TMS 9902's interrupt in various applications including interrupt-driven I/O routines. Connection is made by performing the modification procedure described in Table 9-5. Interconnections are shown on sheet 7 of the schematics in Appendix A.

TABLE 9-5. COMMUNICATIONS INTERRUPT MODIFICATION

TO CONNECT SIGNALS COMINT— AND	ADD JUMPER BETWEEN E16 AND
INT1—	E23
INT 2—	E24
INT 3—	E22
INT 4—	E19
INT 5—	E18

9.4.2 SYSTEM CLOCK FREQUENCY

Some users may wish to increase the system clock frequency above 2.0 MHz to increase processor throughput. This may be accomplished by removing the MP9529 processor (U19) and oscillator crystal (Y1) and installing the components listed in Table 9-6. The replacement crystal should be a series resonant fundamental-mode type with a frequency four times that of the desired new system frequency. Interconnections are shown on sheet 2 of the schematics in Appendix A. The user should be reminded that timing for the keyboard, display, and communications routines is dependent on the system frequency and may require modification for use at the new frequency.

TABLE 9-6. LIST OF MATERIALS, SYSTEM FREQUENCY MODIFICATION

ITEM	QTY	DESCRIPTION	INSTALL AT
1	1	TMS 9980A	U19
2	1	CRYSTAL, SEE TEXT	Y1
3	1	RES, 10Ω, 10%, 0.25W	R49

SECTION 10

TROUBLESHOOTING SUGGESTIONS

10.1 INTRODUCTION

This section outlines a suggested procedure for troubleshooting malfunctioning boards. It is assumed that the user has access to the following test equipment:

- Oscilloscope, preferably dual-trace, triggered sweep
- 10X oscilloscope probes
- VOM

Additional equipment which the user may find helpful includes:

- Logic probe
- Logic analyzer
- Magnifying glass

It is suggested that the user review the theory of operation of the board in Section 8 before proceeding with the troubleshooting procedures.

10.2 TROUBLESHOOTING PROCEDURE

10.2.1 VISUAL CHECKS

Probably the greatest source of board problems is shorts between signals caused by foreign objects and solder bridges between adjacent solder joints. Inspect both sides of the board carefully and remove any shorts observed. Also, brush both sides of the board with a soft dry brush, such as a drafting brush, to sweep away any loose objects which were not spotted in the visual inspection.

10.2.2 STATIC CHECKS

With power applied to the board, measure the three primary supply voltages and compare the measured values to the operational limits listed in Table10-1. A convenient place to access those voltages is at the power supply bypass capacitors (C3, C4 and C5) near P2 (see Table 10-1).

If the primary supplies are within their specified limits, then check the secondary supplies, −5V and VDD, which are derived onboard from the primary supply voltages. See note 8 on sheet 1 of the logic diagram for comments regarding the VDD supply. Operational limits for the secondary supplies are also listed in Table 10-1.

TABLE 10-1. SUPPLY VOLTAGE OPERATIONAL LIMITS

SUPPLY	LIMITS MIN	LIMITS MAX	CHECK AT	COMMENTS
+5V	4.75	5.25	+end, C5	
+12V	11.4	12.6	+end, C3	
−12V	−11.4	−12.6	−end, C4	
−5V	−4.75	−5.25	U19 pin 21	
VDD	8.75	9.95	U19 pin 36	See Note 8, Logic Diagram, Page 1

10.2.3 DYNAMIC CHECKS

If the visual and static checks have not revealed the source of the problem, then proceed with the dynamic checks below. Figures 10-2 through 10-10 show various signal waveforms present on the board under normal conditions. Many waveforms show clock phase $\phi 3-$ at the top of the photo to provide a relative timing reference. The user should keep in mind that the waveforms shown are typical and may vary slightly from board to board due to component variations and instructions being executed at that time. Figure 10-1 provides information to be used in the interpretation of the photographs. The following procedure may be followed to help guide the user's investigation.

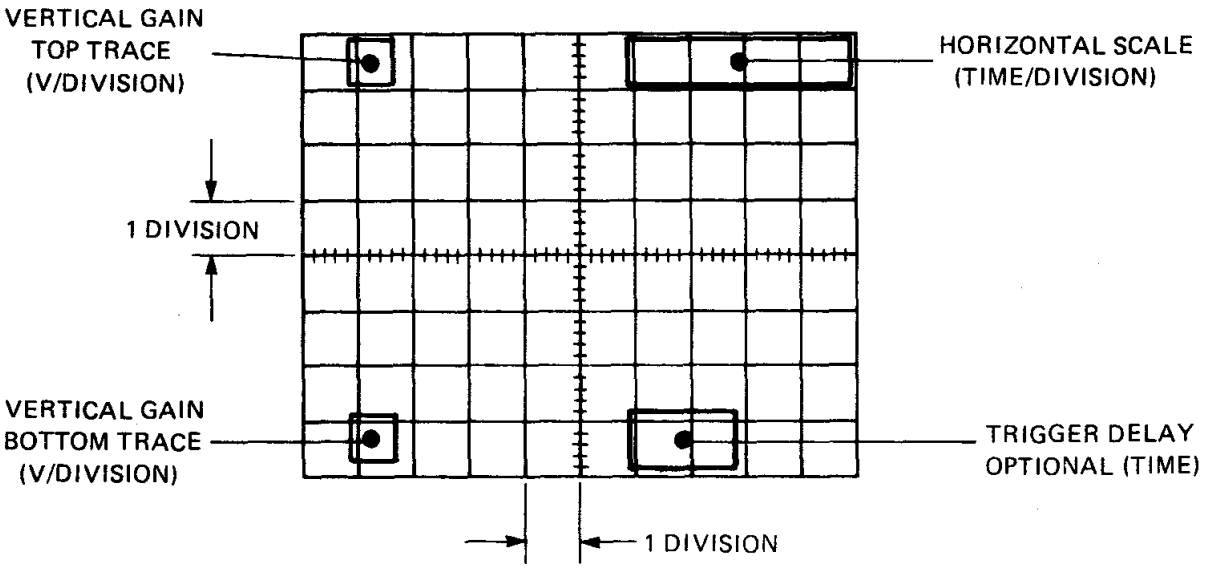


Figure 10-1. Photo Interpretation Guide

a. Refer to sheet 12 of the logic diagrams. Observe the waveform of the voltage across C11 in the power-up reset circuit at application and immediately following application of power to the board. Also observe the waveform of the RESET— and POWERGOOD signals and compare with Figure 10-2. POWERGOOD is not shown in the figure but should follow RESET— high 400 to 1400 milliseconds after power is applied.

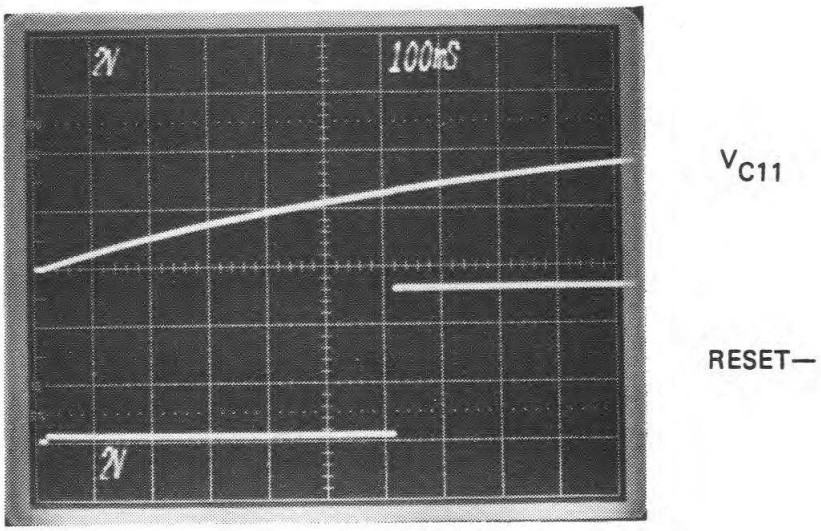


Figure 10-2. V_{C11} vs. RESET—

b. Refer to sheet 2 of the logic diagram. Observe the waveforms of CKIN and clock phase $\phi 3-$ at the microprocessor and compare with Figure 10-3.

NOTE

When making high frequency measurements such as above, use the shortest possible scope probe ground lead. The increased inductance and stray capacitance associated with longer ground leads can introduce resonant stages in the equivalent circuit of the scope probe which are not negligible at the signal frequencies being observed. The effect will be observed as severe waveform ringing following signal transitions, and greatly degrades the quality of the information obtainable from the waveform.

c. Refer to sheet 2 of the logic diagrams. Observe the waveforms of MEMEN—, DBIN, IAQ, WE—, and CRUCLK at the microprocessor. Compare with Figures 10-4 through 10-8.

d. For problems associated with the audio cassette interface, refer to sheets 8 and 12 of the logic diagrams. Observe the waveforms of AUDIOIN, RDATA, WDATA, and AUDIOOUT with the recorder connected and operating in the appropriate playback or record mode.

10.3 TECHNICAL ASSISTANCE

Users experiencing difficulty with the operation or debug of their board may refer their questions to the Microprocessor Customer Support Line at (713) 776-6632 or address them to:

Texas Instruments Incorporated
8600 Commerce Park Drive
Suite 700, M/S 6404
Houston, Texas 77036

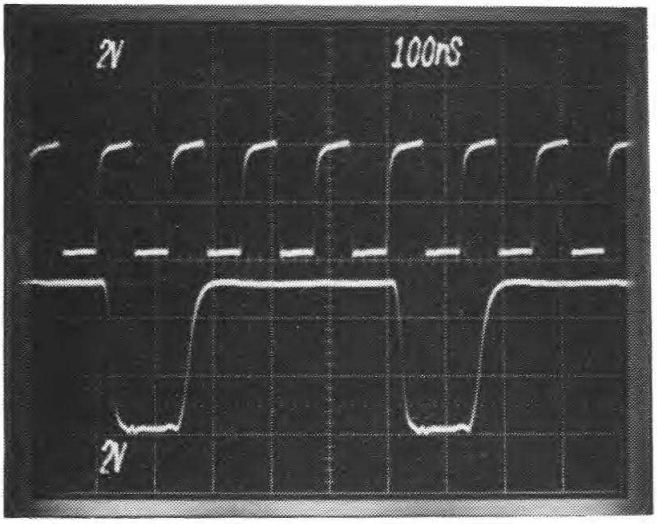


Figure 10-3. CKIN vs. Clock Phase ϕ3 –

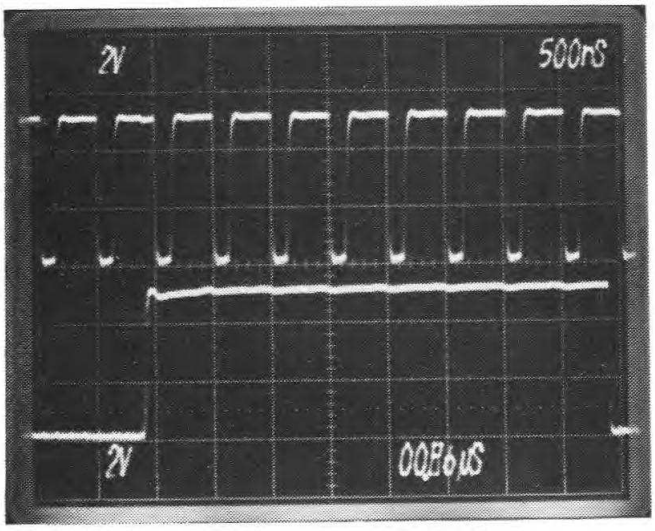
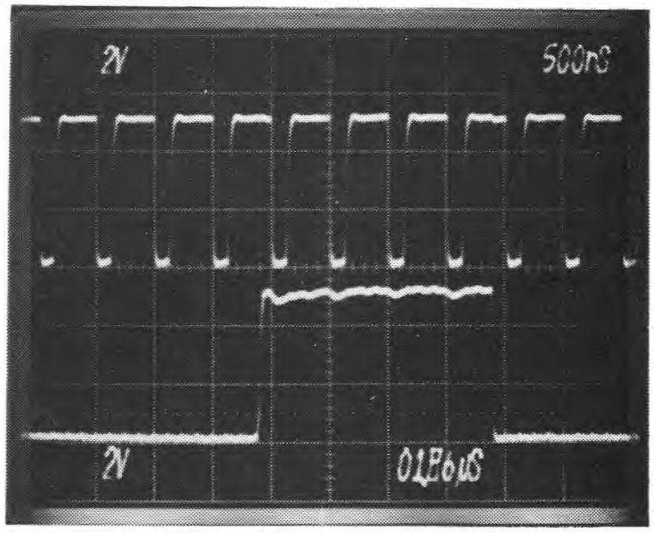


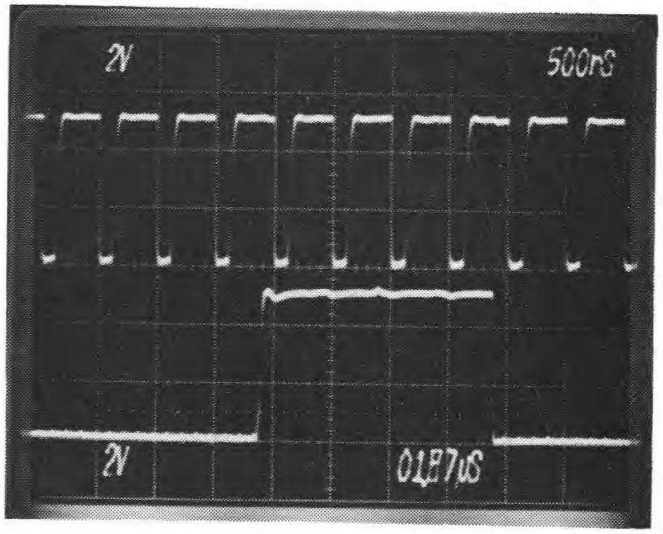
Figure 10-4. Clock Phase ϕ3— vs, MEMEN—



$\phi 3-$

DBIN

Figure 10-5. Clock Phase $\phi 3-$ vs. DBIN



$\phi 3-$

IAQ

Figure 10-6. Clock Phase $\phi 3-$ vs. IAQ

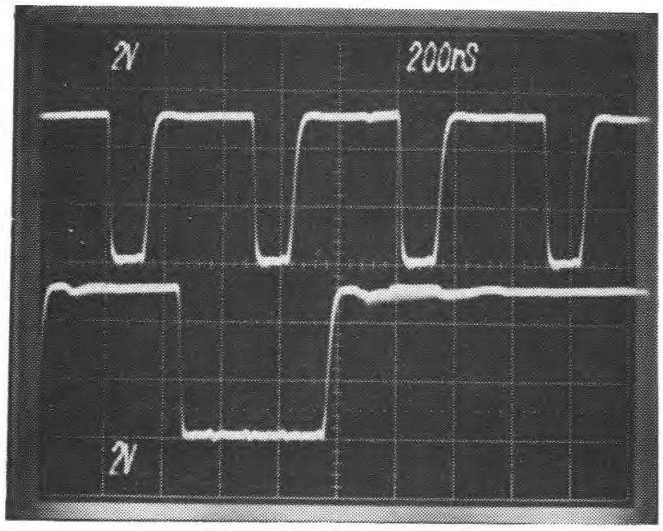


Figure 10-7. Clock Phase ϕ3— vs. WE—

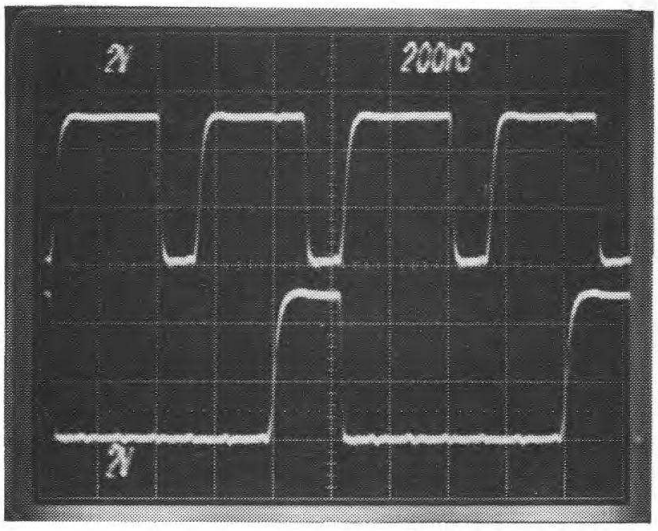


Figure 10-8. Clock Phase ϕ3— vs. CRUCLK

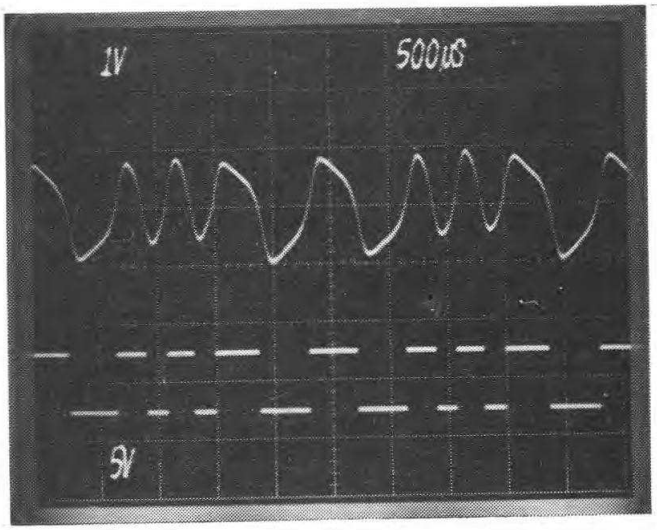


Figure 10-9. AUDIOIN vs. RDATA

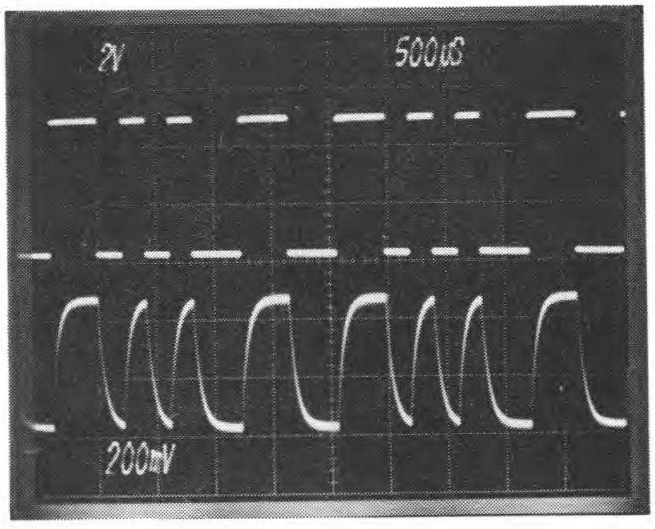


Figure 10-10. WDATA vs. AUDIO OUT

APPENDIX A

SCHEMATICS

NOTES: UNLESS OTHERWISE SPECIFIED:

1. CAPACITANCE VALUES ARE IN MICROFARADS
2. RESISTANCE VALUES ARE IN OHMS

[3] EXPANSION RAM'S (U21 AND U23) INSTALLED BY USER.

[4] EXPANSION EPROM (U32) INSTALLED BY USER.

[5] SERIAL COMMUNICATION INTERFACE COMPONENTS (U31, U37, R32 THRU R39, Q12, CR9, AND P3) INSTALLED BY USER.

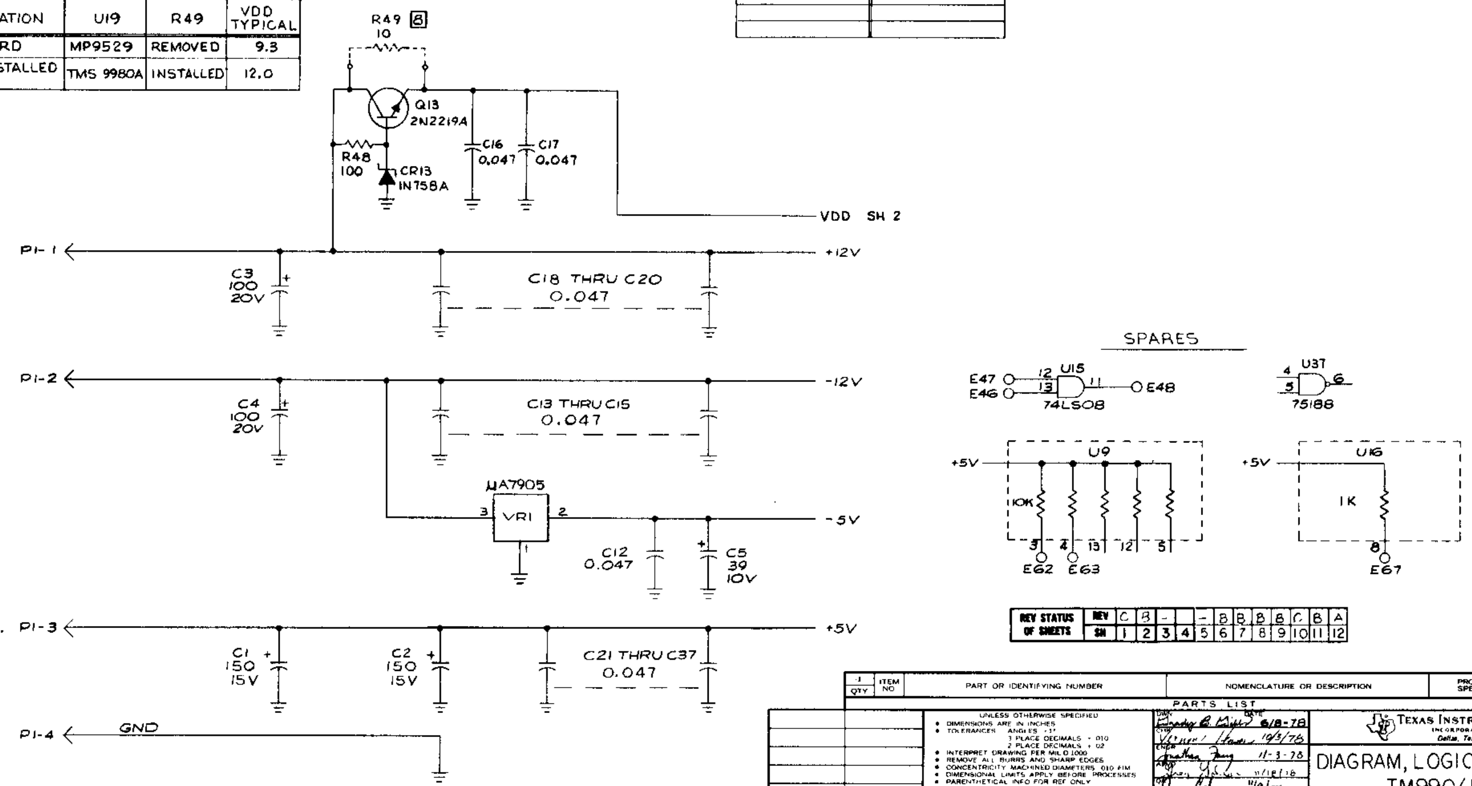
[6] BUS EXPANSION INTERFACE COMPONENTS (U18, U26, U27, U30, AND R34) INSTALLED BY USER.

[7] DECK CONTROL COMPONENTS (K1 AND CR12) INSTALLED BY USER

[8]

CONFIGURATION	U19	R49	VDD TYPICAL
STANDARD	MP9529	REMOVED	9.3
USER-INSTALLED OPTION	TMS 9980A	INSTALLED	12.0

REFERENCE DESIGNATORS	
USED	NOT USED
C1 - C37	
CR1 - CR12	
DS1, DS2	
K1	
P1 - P5	
Q1 - Q13	
R1 - R51	
S1, S2	
U1 - U38	
VR1	
Y1	



REV STATUS OF SHEETS	REV	C	B	-	-	-	B	B	B	B	C	B	A
	SH	1	2	3	4	5	6	7	8	9	10	11	12

REVISIONS			
REV	DESCRIPTION	DATE	APPROVED
A	CN 446938	10/26/78	
B	CN446959	11/20/78	
C	CN447300	1/19/79	

DWG NO 991727 SH 1

PARTS LIST

QTY	ITEM NO	PART OR IDENTIFYING NUMBER	NOMENCLATURE OR DESCRIPTION	PROCUREMENT SPECIFICATION	NOTES

UNLESS OTHERWISE SPECIFIED
- DIMENSIONS ARE IN INCHES
- TOLERANCES: ANGLES ±1°, 3 PLACE DECIMALS ±.010, 2 PLACE DECIMALS ±.02
- INTERPRET DRAWING PER MIL-D-1000
- REMOVE ALL BURRS AND SHARP EDGES
- CONCENTRICITY MACHINED DIAMETERS .010 FIM
- DIMENSIONAL LIMITS APPLY BEFORE PROCESSES
- PARENTHETICAL INFO FOR REF ONLY

HOLE TOLERANCE

NEXT ASSY	USED ON
991726	8117

APPLICATION

TEXAS INSTRUMENTS INCORPORATED Dallas, Texas

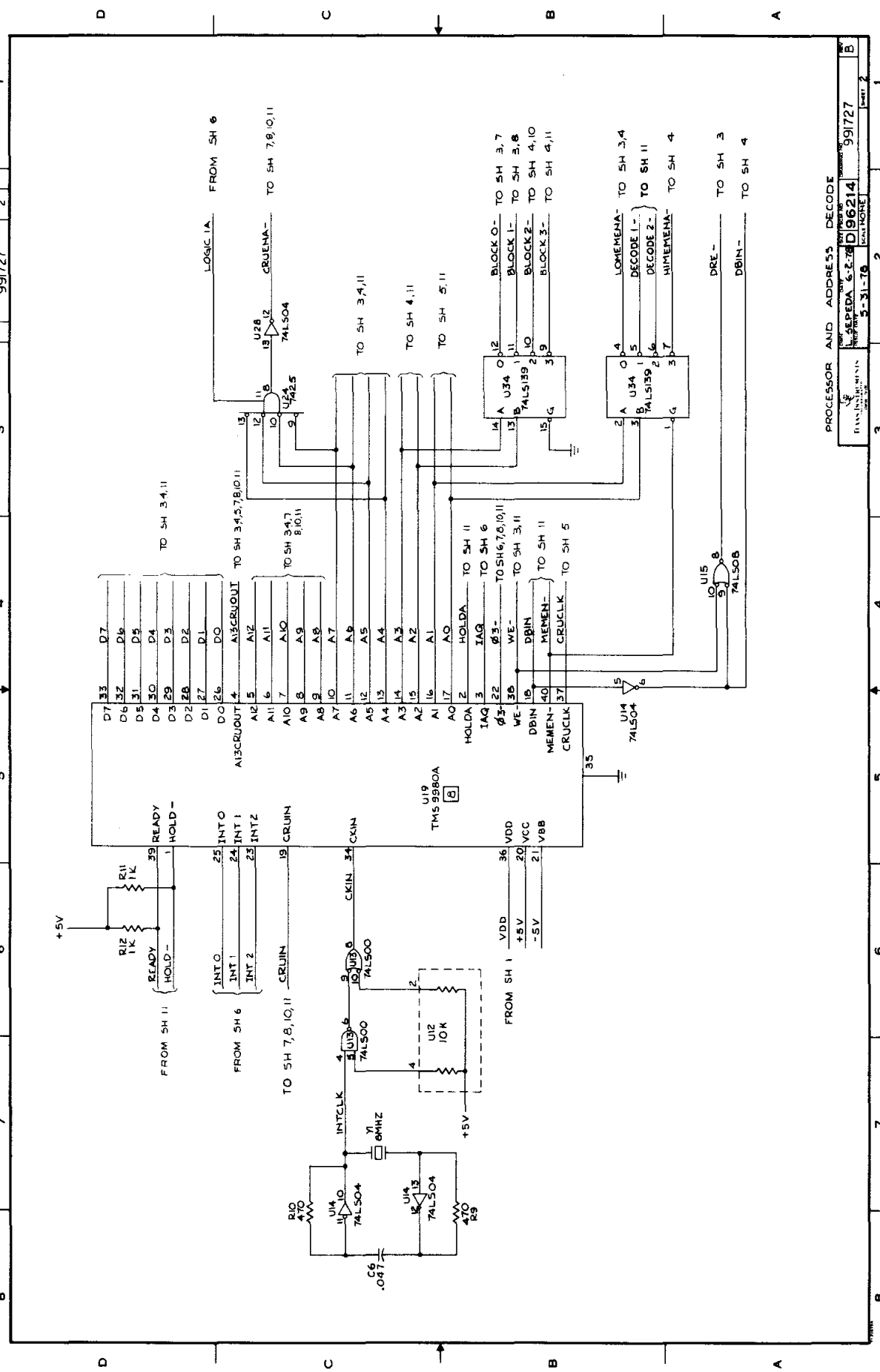
DIAGRAM, LOGIC, DETAILED TM990/189

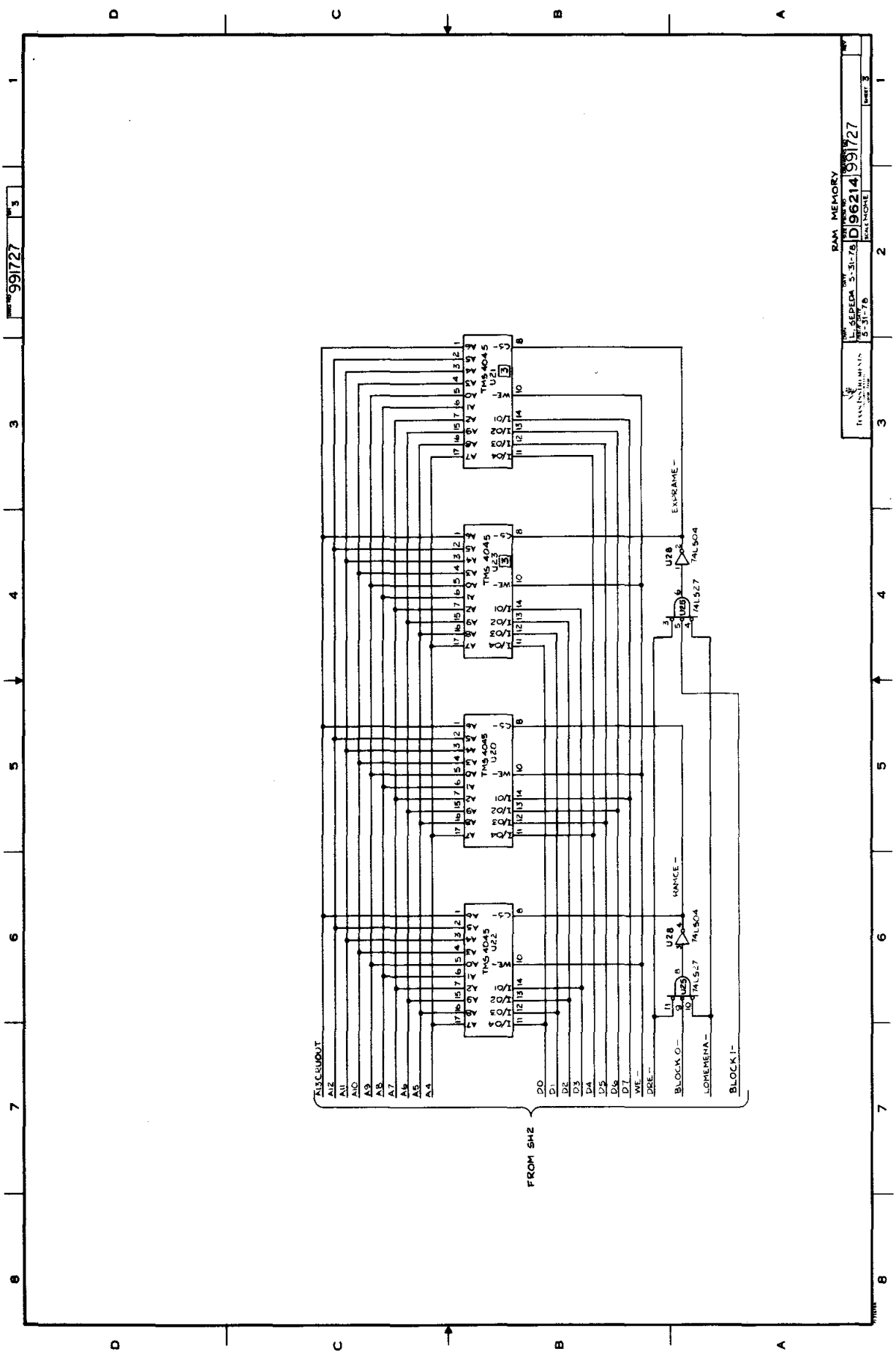
SIZE D FSCM NO 96214 DRAWING NO 991727

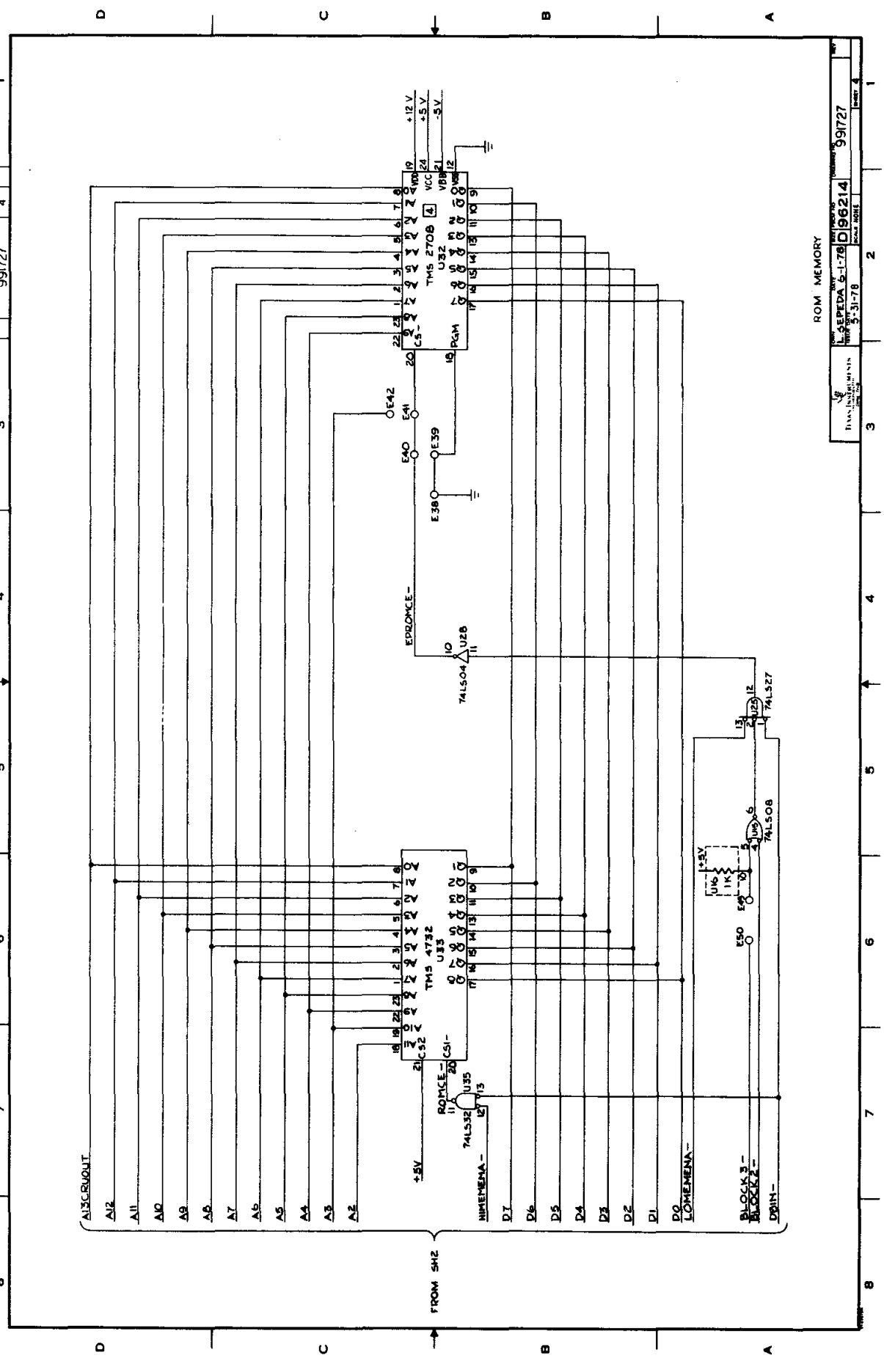
SCALE NONE SHEET 1 OF 12

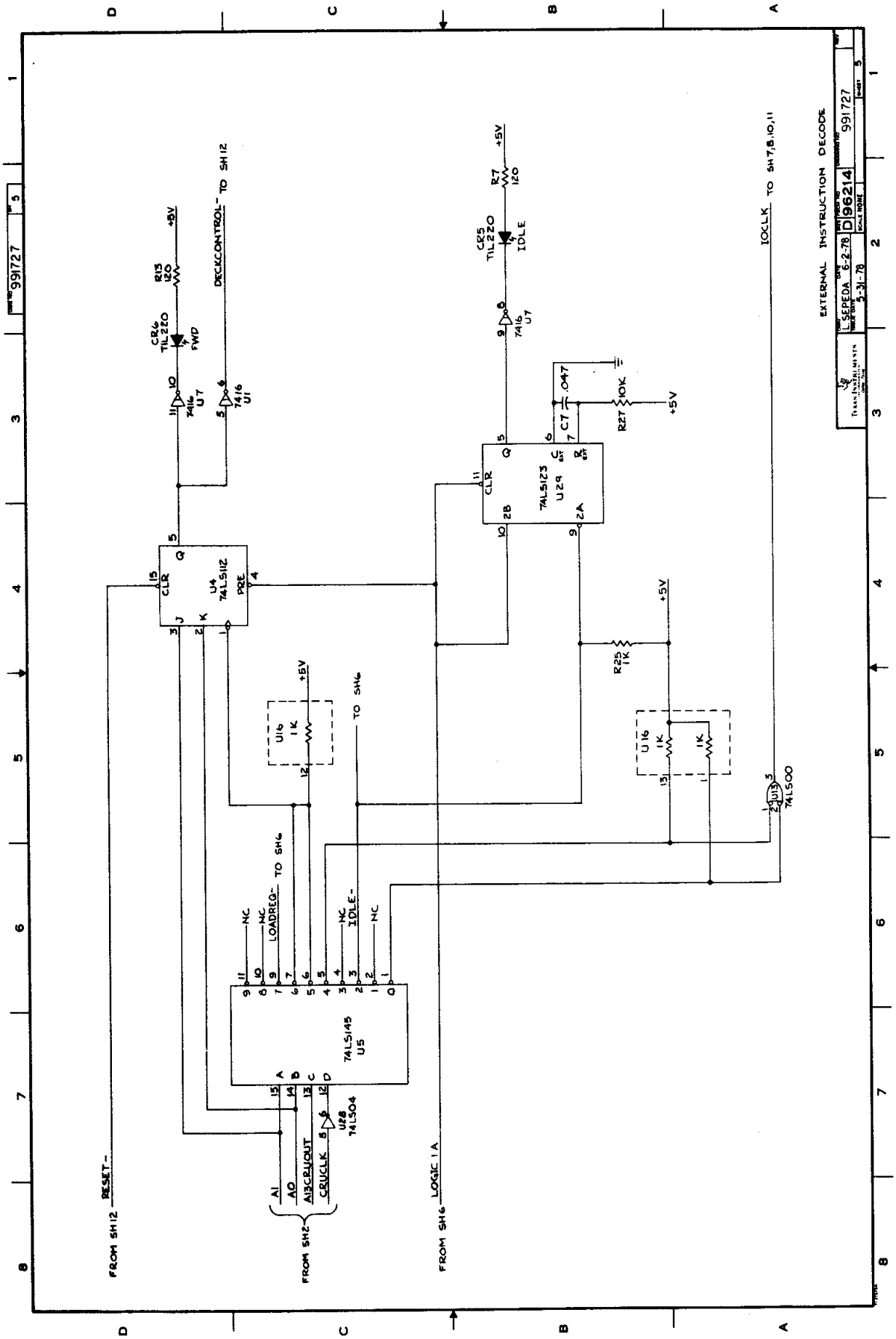
SEQ NO	IDENT	F-SPEC	NO	ADDITIONAL	NOTES
	PROCESS			CLASSIFICATION	

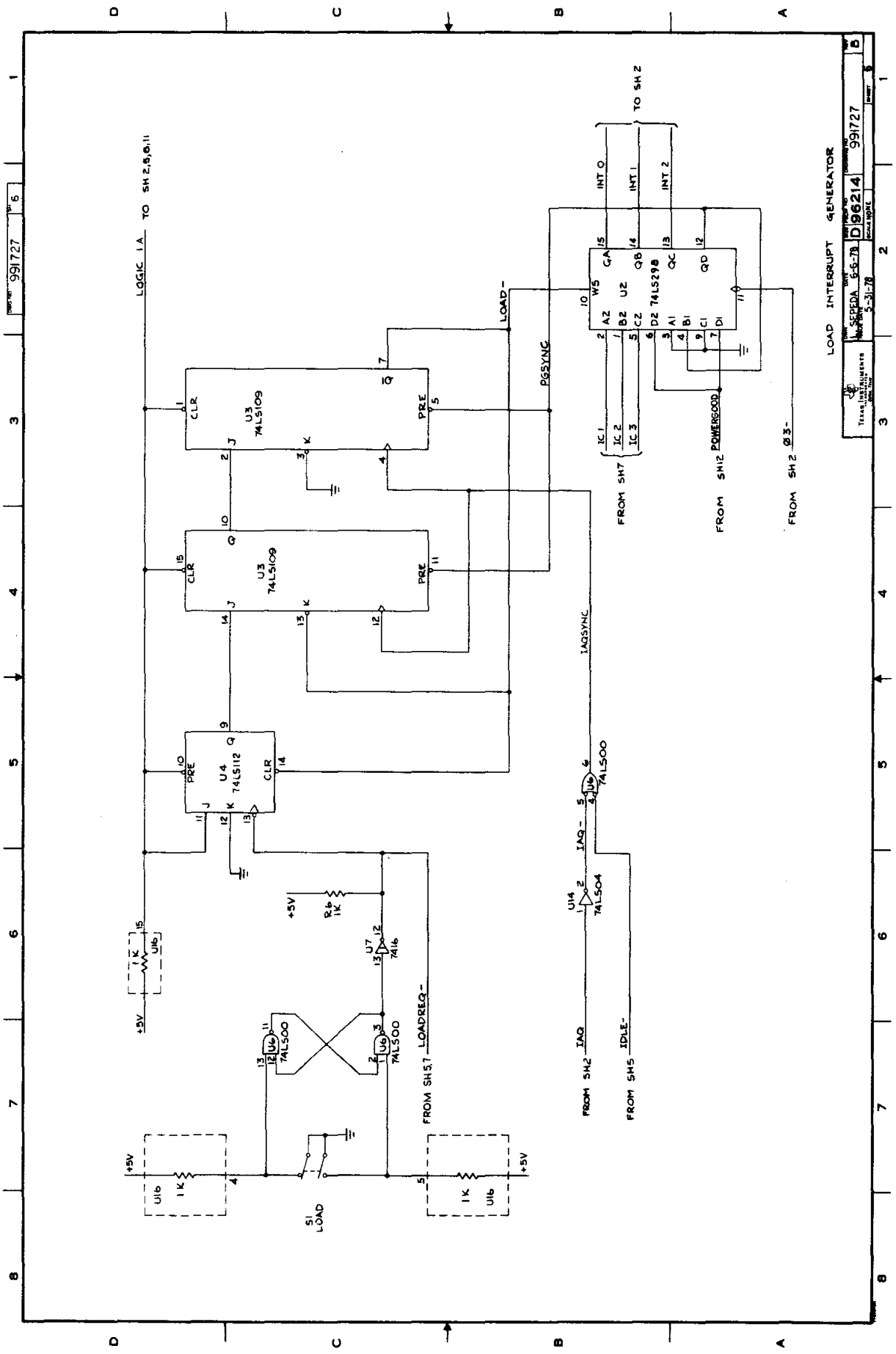
PROCESSES — FOR CORRELATION TO GOVT/IND SPECIFICATIONS SEE TI DRAWING 729467

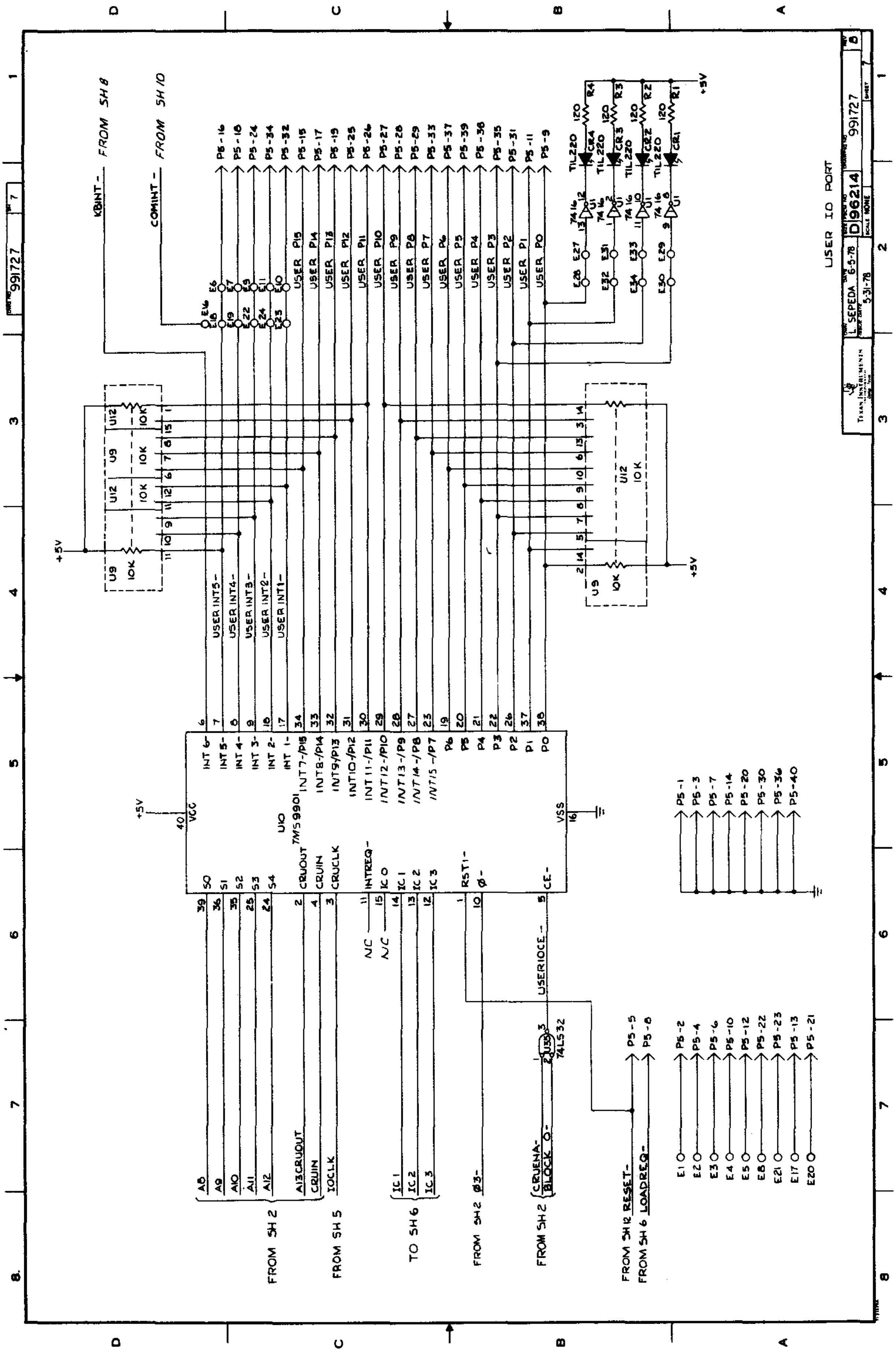

PROCESSOR AND ADDRESS DECODE
991727
D96214
5-31-79
TMS 9980A
TO SH 3,4,11
TO SH 4,11
TO SH 5,11
FROM SH 6
LOGIC IA
CRUENA-
BLOCK 0-
BLOCK 1-
BLOCK 2-
BLOCK 3-
LOMEMENA-
DECODE 1-
DECODE 2-
HIMEMENA-
DRE-
DBIN-
READY
HOLD-
INT 0
INT 1
INT 2
CRUIN
INTCLK
6MHZ
+5V
-5V
VDD
FROM SH 1
FROM SH 11
FROM SH 6
TO SH 7,8,10,11
TO SH 3,4,5,7,8,10,11
TO SH 3,4,7,8,10,11
TO SH 6
TO SH 11
TO SH 3,11
TO SH 5
TO SH 3,7
TO SH 3,8
TO SH 4,10
TO SH 4,11
TO SH 3,4
TO SH 11
TO SH 4
TO SH 3
TO SH 4

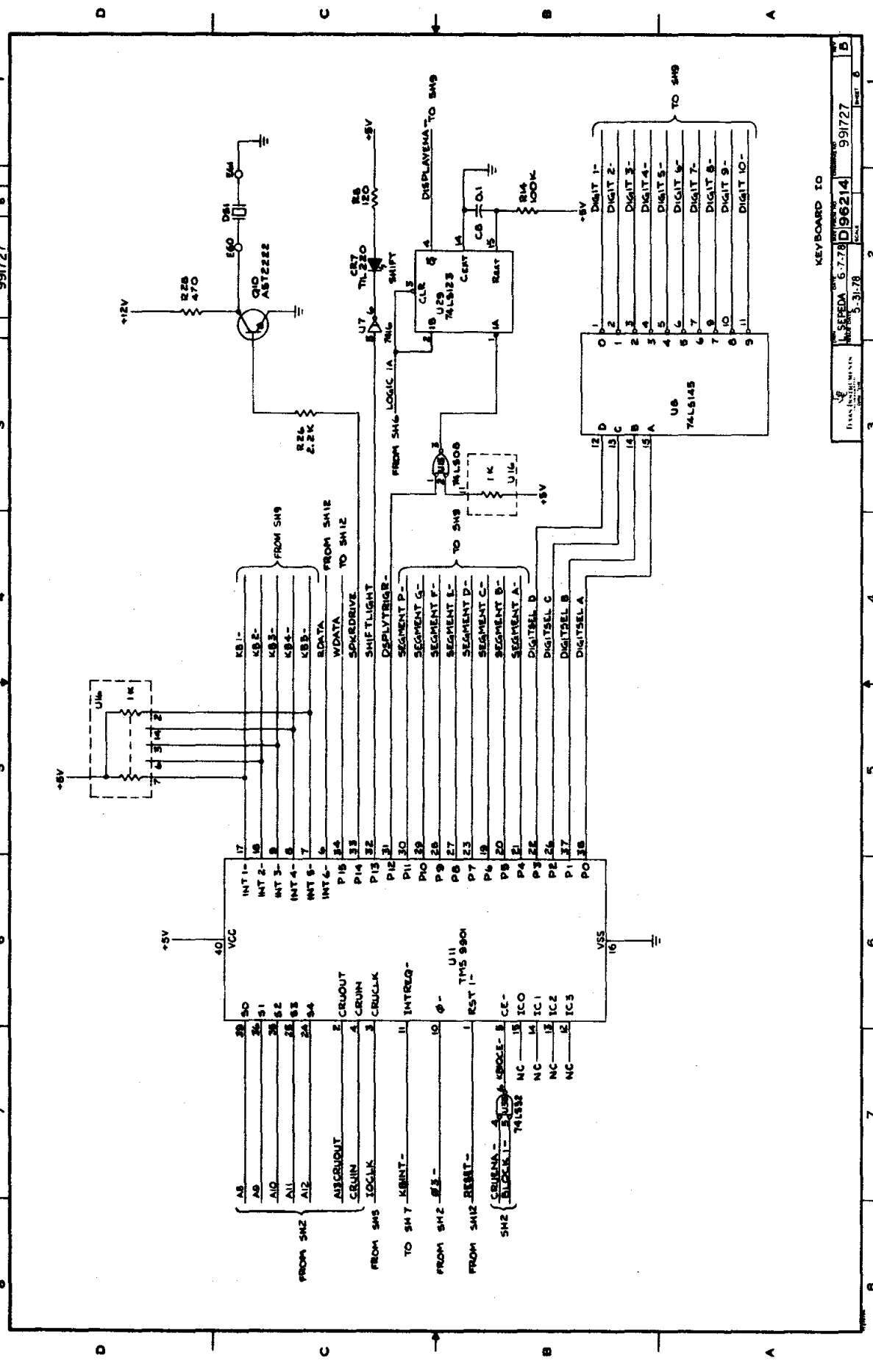

RAM MEMORY
99727
D96214
TMS4045
U21
U22
U23
U20
74LS04
74LS27
U28
U25
A13CLKOUT
A12
A11
A10
A9
A8
A7
A6
A5
A4
D0
D1
D2
D3
D4
D5
D6
D7
WE –
DRE –
BLOCK 0 –
LOMEMENA –
BLOCK 1 –
EXRAME –
RAMCE –
FROM SH2

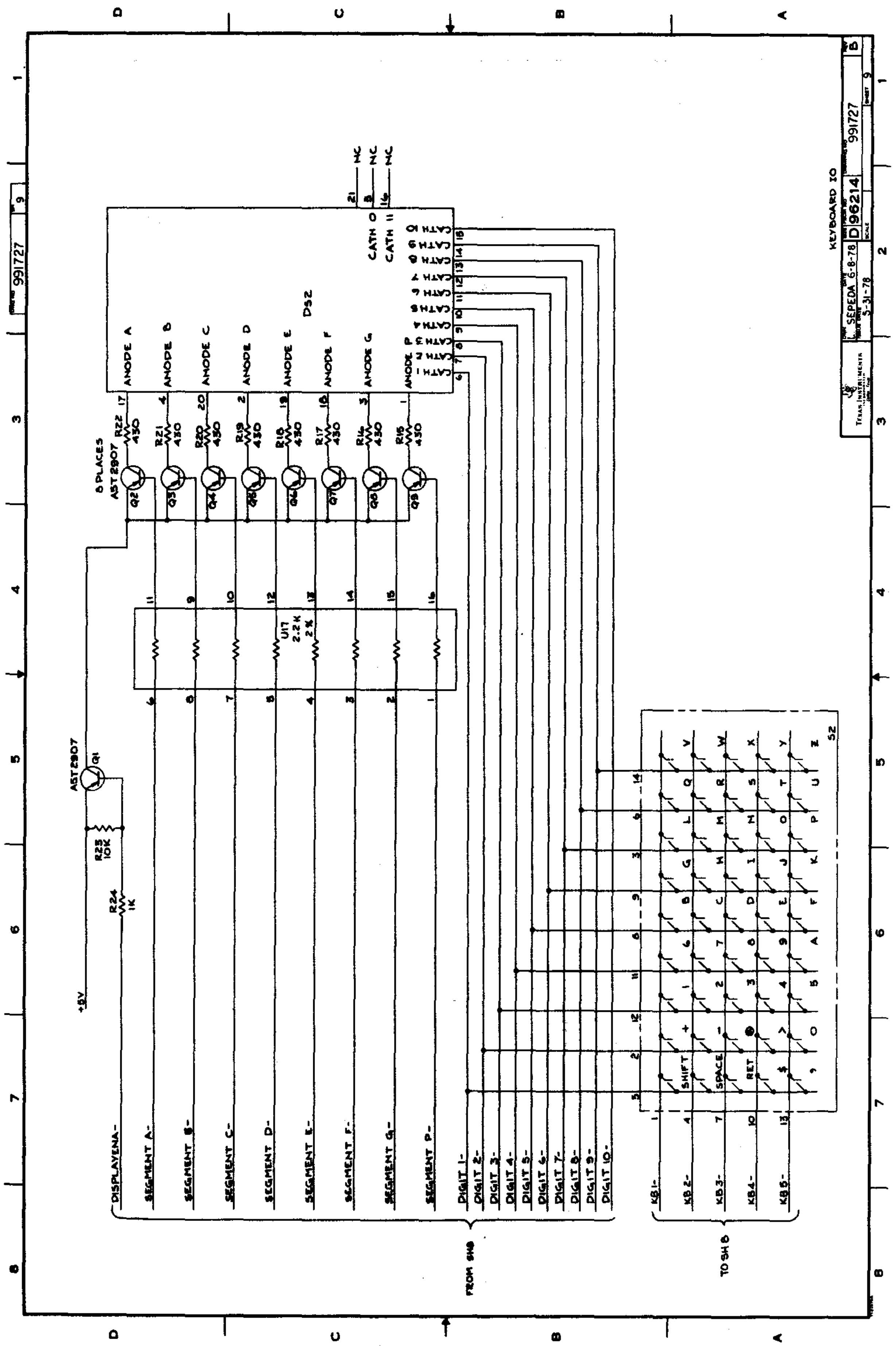

ROM MEMORY
99727
D96214
L. SEPEDA 6-1-78
5-31-78
TMS 2708
U32
TMS 4732
U33
EPROMCE-
ROMCE-
CS1-
CS2-
741S04
U28
741S32
U35
741S08
741S27
+12 V
+5 V
-5 V
+5V
1K
A13CRYOUT
A12
A11
A10
A9
A8
A7
A6
A5
A4
A3
A2
FROM SH2
D7
D6
D5
D4
D3
D2
D1
D0
HIMEMENA-
LOMEMENA-
BLOCK3-
BLOCK2-
DBIN-

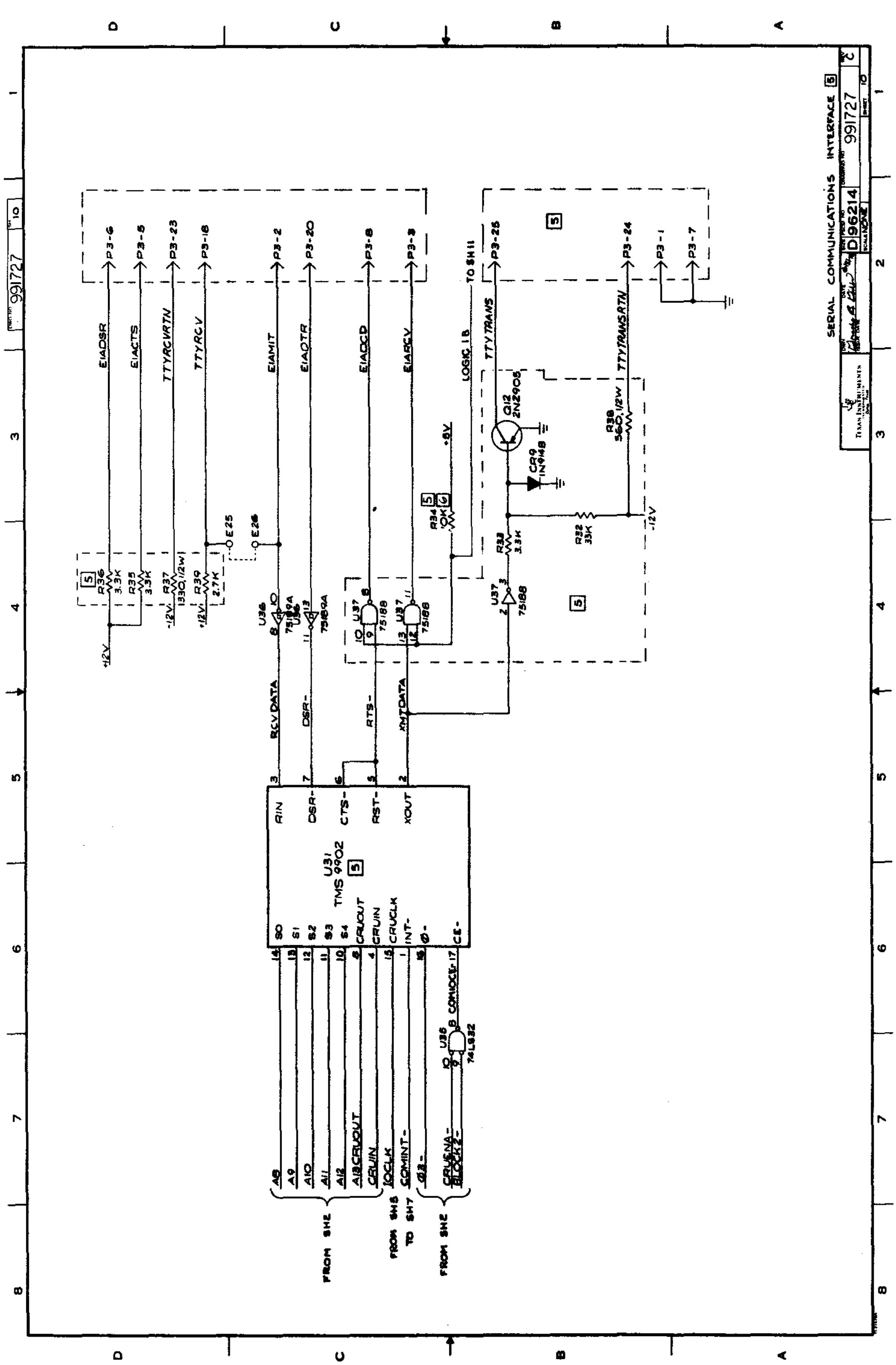

991727
EXTERNAL INSTRUCTION DECODE
L.SEPEDA 6-2-78
D96214
5-31-78
991727
DECKCONTROL- TO SH12
IOCLK TO SH7,8,10,11
R13 120
R7 120
CR4 TIL220 FWD
CR5 TIL220 IDLE
+5V
7416 U7
7416 U1
U4 74LS112
U29 74LS123
R27 10K
.047
R25 1K
U16 1K
U13 74LS00
U5 74LS145
U28 74LS04
NC
LOADREG- TO SH6
IDLE- TO SH6
TO SH6
RESET- FROM SH12
A1
A0
A13CRUOUT
CRUCLK
FROM SH12
LOGIC 1 A
FROM SH6

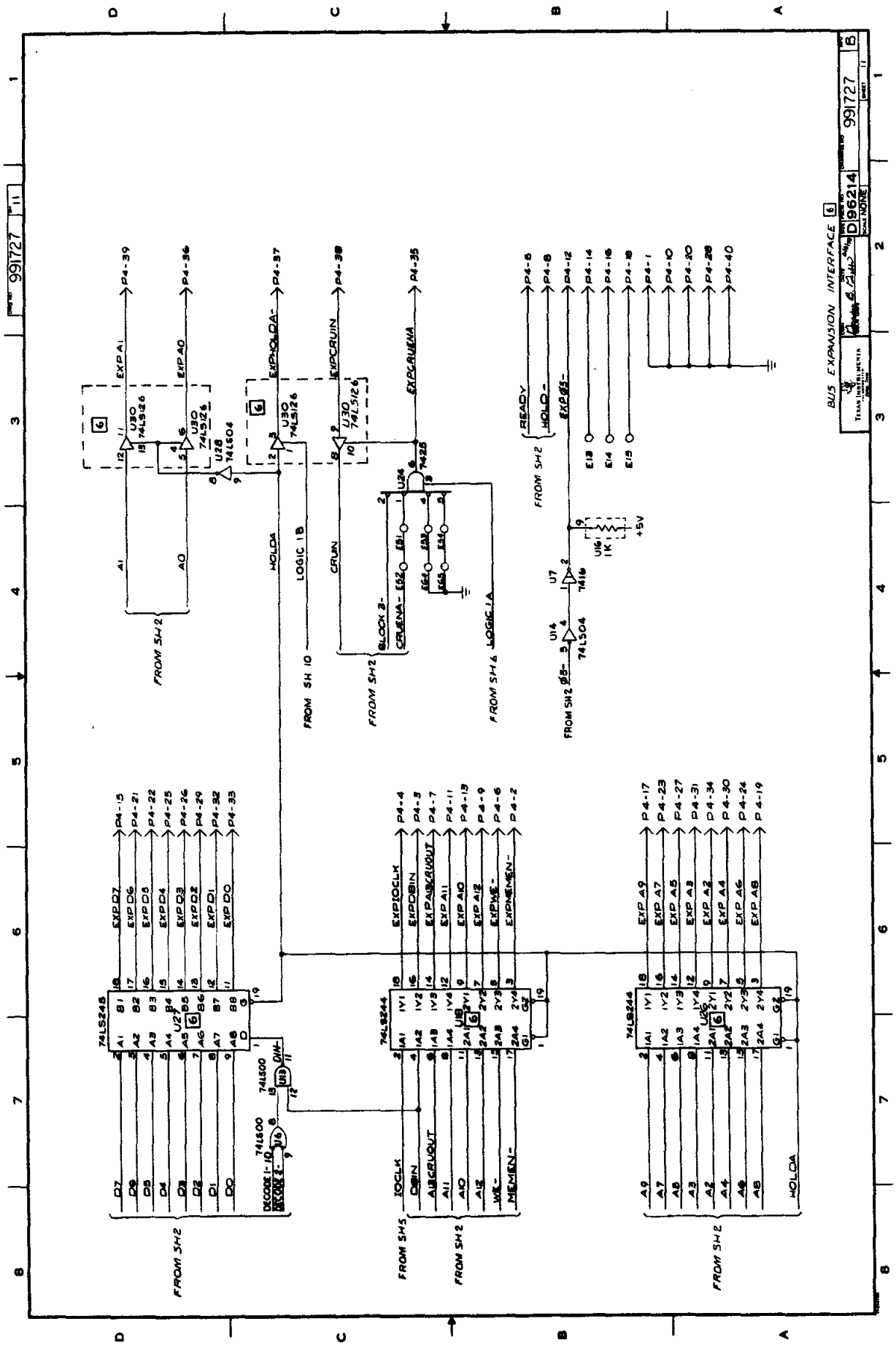

LOAD INTERRUPT GENERATOR
D96214
991727
LOGIC 1A TO SH 2,8,10,11
TO SH 2
INT 0
INT 1
INT 2
U2 74LS298
U3 74LS109
U4 74LS112
U3 74LS109
LOAD-
PGSYNC
IAQSYNC
IAQ-
IAQ
U14 74LS04
U6 74LS00
U7 7416
R4 1K
+5V
1K U16
U6 74LS00
U6 74LS00
LOADREQ-
FROM SH 5,7
S1 LOAD
IDLE-
FROM SH 5
FROM SH 2
FROM SH 7
IC 1
IC 2
IC 3
FROM SH12 POWERGOOD
FROM SH 2 Ø3-

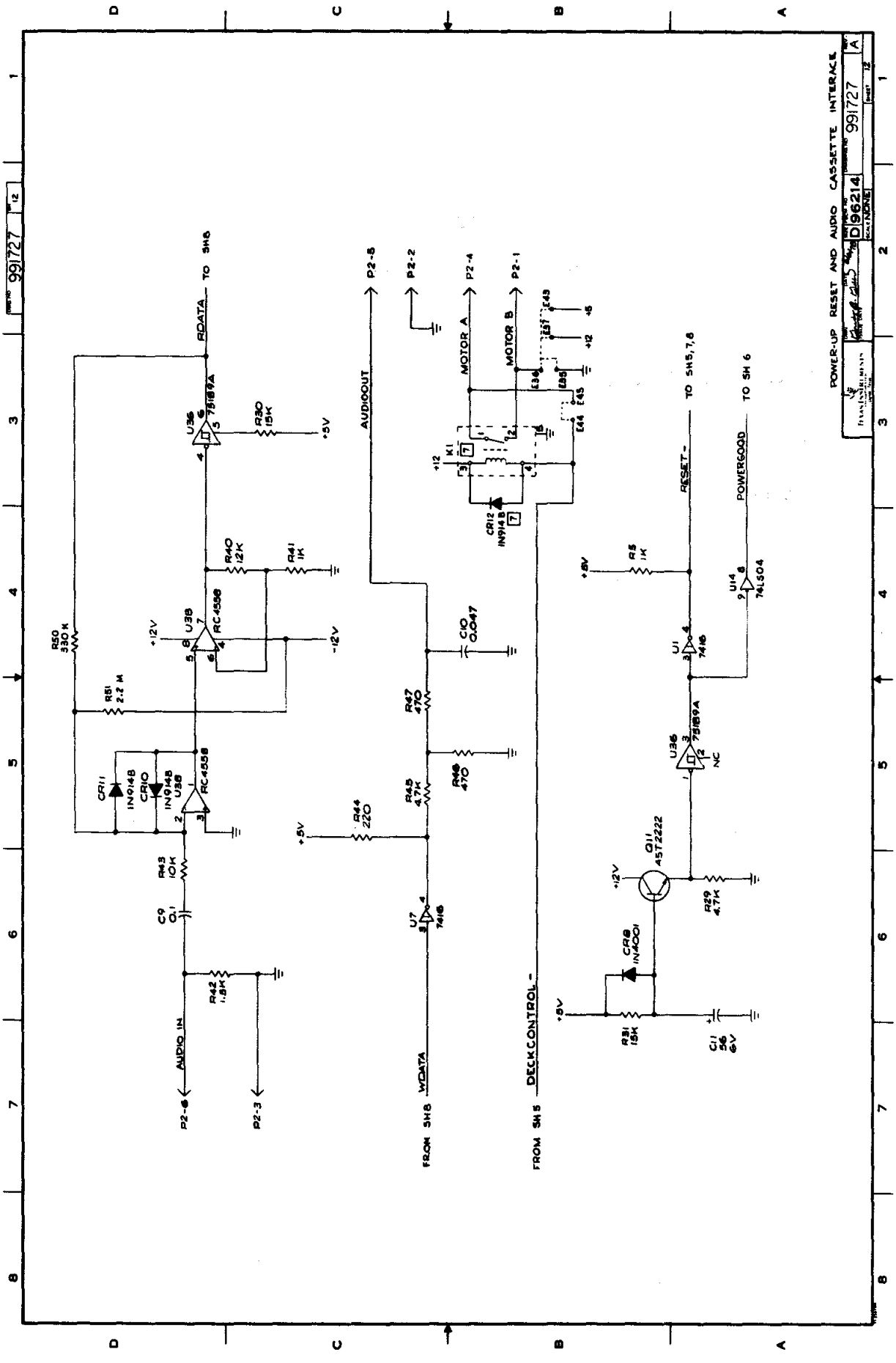

USER IO PORT
991727
D 96214
L SEPEDA 6-5-78
5-31-78
TMS 9901
74LS32


KEYBOARD IO
99727
D96214
SEREDA 6-7-78
5-31-78
DISPLAYENA- TO SH9
FROM SH9
TO SH9
FROM SH12
TO SH12
FROM SH6 LOGIC 1A
U8 74LS145
U11 TMS 9901
U7 74LS86
U29 74LS123
U16 74LS08
Q10 AST2222
CR7 TIL220
+5V
+12V
DIGIT 1-
DIGIT 2-
DIGIT 3-
DIGIT 4-
DIGIT 5-
DIGIT 6-
DIGIT 7-
DIGIT 8-
DIGIT 9-
DIGIT 10-
KB1-
KB2-
KB3-
KB4-
KB5-
SDATA
WDATA
SPKRDRIVE
SHIFTLIGHT
DSPLYTRIGR-
SEGMENT G-
SEGMENT F-
SEGMENT E-
SEGMENT D-
SEGMENT C-
SEGMENT B-
SEGMENT A-
DIGITSEL D
DIGITSEL C
DIGITSEL B
DIGITSEL A
CRUOUT
CRUIN
CRUCLK
INTREQ-
RESET-
CRUENA-
BLOCK 1-
SHIFT


KEYBOARD IO
991727
D 96214
SEPEDA 6-8-78
5-31-78
TEXAS INSTRUMENTS
DS2
S2
U17 2.2K 2%
8 PLACES AST 2907
AST 2907
R23 10K
R24 1K
+5V
DISPLAYENA-
SEGMENT A-
SEGMENT B-
SEGMENT C-
SEGMENT D-
SEGMENT E-
SEGMENT F-
SEGMENT G-
SEGMENT P-
DIGIT 1-
DIGIT 2-
DIGIT 3-
DIGIT 4-
DIGIT 5-
DIGIT 6-
DIGIT 7-
DIGIT 8-
DIGIT 9-
DIGIT 10-
KB1-
KB2-
KB3-
KB4-
KB5-
FROM SHB
TO SHB


SERIAL COMMUNICATIONS INTERFACE
991727
D96214
U31
TMS 9902
RIN
DSR-
CTS-
RST-
XOUT
S0
S1
S2
S3
S4
CRUOUT
CRUIN
CRUCLK
INT-
CE-
RCV DATA
DSR-
RTS-
XMT DATA
EIADSR
EIACTS
TTYRCVRTN
TTYRCV
EIAMIT
EIADTR
EIADCD
EIARCV
LOGIC 1B
TO SH11
TTYTRANS
TTYTRANSRTN
P3-6
P3-5
P3-23
P3-18
P3-2
P3-20
P3-8
P3-3
P3-25
P3-24
P3-1
P3-7
Q12
2N2905
CR9
1N914B
R38
560, 1/2W
R32
33K
R33
3.3K
R34
10K
R36
3.3K
R35
3.3K
R37
1330, 1/2W
R39
2.7K
E25
E26
U36
75189A
U37
75188
+5V
+12V
-12V
U35
74LS32
A8
A9
A10
A11
A12
A13CRUOUT
CRUIN
IOCLK
COMINT-
CRUENA-
BLOCK2-
FROM SH2
FROM SH5
TO SH7


BUS EXPANSION INTERFACE
991727
D96214
TEXAS INSTRUMENTS
FROM SH2
FROM SH 10
FROM SH6
LOGIC 1A
LOGIC 1B
HOLDA
CRUN
READY
HOLD-
74LS245
74LS244
74LS04
74LS00
74LS126
7416
7425
EXPD7
EXPD6
EXPD5
EXPD4
EXPD3
EXPD2
EXPD1
EXPD0
EXPIOCLK
EXPIOBIN
EXPABGRNTOUT
EXPA11
EXPA10
EXPA12
EXPWE-
EXPMEMEN-
EXPA9
EXPA7
EXPA5
EXPA3
EXPA2
EXPA4
EXPA6
EXPA8
EXPA1
EXPA0
EXPHOLDA-
EXPBRUIN
EXPBRUENA
EXPBRQ-
+5V


POWER-UP RESET AND AUDIO CASSETTE INTERFACE
991727
D19614
RDATA
TO SH6
AUDIOOUT
P2-8
P2-2
MOTOR A
P2-4
MOTOR B
P2-1
RESET-
TO SH5,7,8
POWERGOOD
TO SH 6
AUDIO IN
P2-6
P2-3
WDATA
FROM SH8
DECKCONTROL-
FROM SH5
U36 75189A
U38 RC4558
U7 7416
U1 7416
U14 74LS04
Q11 4S72222
K1
CR11 IN914B
CR10 IN914B
CR12 IN914B
CR8 IN4001
R50 330K
R51 2.2 M
R40 12K
R41 1K
R30 15K
R43 10K
R42 1.8K
R44 220
R45 4.7K
R46 470
R47 470
R5 1K
R29 4.7K
R31 15K
C9 0.1
C10 0.047
C11 56 6V
+12V
-12V
+5V
+12
+5

APPENDIX B

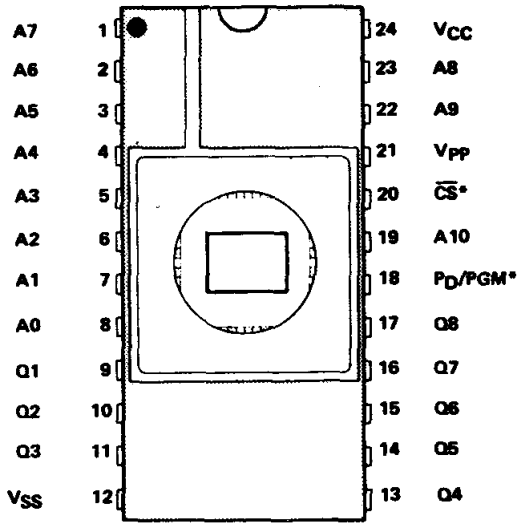
MEMORY DATA SHEETS

TMS 2516 JL AND TMS 2532 JL
16K AND 32K EPROMs

APRIL 1978

- Organization:
  - TMS 2516 . . . 2K X 8
  - TMS 2532 . . . 4K X 8
- Single +5 V Power Supply
- Pin Compatible with Existing ROMs and EPROMs (8 K, 16 K, 32 K, and 64 K)
- JEDEC Standard Pinouts
- All Inputs/Outputs Fully TTL Compatible
- Static Operation (No Clocks, No Refresh)
- Max Access/Min Cycle Time . . . 450 ns
- 8-Bit Output for Use in Microprocessor-Based Systems
- N-Channel Silicon-Gate Technology
- 3-State Output Buffers
- Low Power
  - Active:
    - TMS 2516 . . . 285 mW Typical
    - TMS 2532 . . . 400 mW Typical
  - Standby . . . 50 mW Typical
- Guaranteed dc Noise Immunity with Standard TTL Loads
- No Pull-Up Resistors Required

TMS 2516
24-PIN CERAMIC
DUAL-IN-LINE PACKAGE
(TOP VIEW)



*FOR TMS 2532:
PIN 18 . . . A11
PIN 20 . . . . PD/$\overline{PGM}$

PIN NOMENCLATURE	
A(N)	Address inputs
$\overline{CS}$	Chip Select
PD/PGM PD/$\overline{PGM}$	Power Down/Program
Q(N)	Input/Output
V_{CC}	+5 V Power Supply
V_{PP}	+25 V Power Supply
V_{SS}	0 V Ground

description

The TMS 2516 JL and TMS 2532 JL are 16,384-bit and 32,768-bit, ultraviolet light erasable, electrically programmable read-only memories. These devices are fabricated using N-channel silicon-gate technology for high speed and simple interface with MOS and Bipolar circuits. All inputs (including program data inputs) can be driven by Series 74 TTL circuits without the use of external pull-up resistors, and each output can drive one Series 74 TTL circuit without external resistors. The data outputs are three-state for OR-tying multiple devices on a common bus. The TMS 2516 is upward pin-compatible with the TMS 2532 and the TMS 2532 is plug-in compatible with the TMS 4732 32K ROM.

Since these EPROMs operate from a single +5 V supply (in the read mode), they are ideal for use in microprocessor systems. One other (+25 V) supply is needed for programming but all programming signals are TTL level, requiring a single 50 ms pulse. For programming outside of the system, existing EPROM programmers can be used. Locations may be programmed singly, in blocks, or at random. Total programming time for all bits for the TMS 2516 is 100 seconds; 200 seconds for the TMS 2532.

PRELIMINARY DATA SHEET:
Supplementary data will be published at a later date.

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 5012 • DALLAS, TEXAS 75222

operation

DEVICE		MODE										
FUNCTION (PINS)		Read		Output Disable		Power Down		Start Programming		Inhibit Programming		Program Verification
TMS 2516	TMS 2532	TMS 2516	TMS 2532	TMS 2516	TMS 2532	TMS 2516	TMS 2532	TMS 2516	TMS 2532	TMS 2516	TMS 2532	TMS 2516
PD/PGM (18)	PD/$\overline{PGM}$ (20)	V_{IL}	V_{IL}	Don't Care	V_{IH}	V_{IH}	V_{IH}	Pulsed V_{IL} to V_{IH}	Pulsed V_{IH} to V_{IL}	V_{IL}	V_{IH}	V_{IL}
$\overline{CS}$ (20)	Use PD/$\overline{PGM}$ as chip select	V_{IL}	N/A	V_{IH}	N/A	Don't Care	N/A	V_{IH}	N/A	V_{IH}	N/A	V_{IL}
V_{PP} (21)	V_{PP} (21)	+5	+5	+5	+5	+5	+5	+25	+25	+25	+25	+25 (or +5)
V_{CC} (24)	V_{CC} (24)	+5	+5	+5	+5	+5	+5	+5	+5	+5	+5	+5
Q (9 to 11, 13 to 17)	Q (9 to 11, 13 to 17)	Q	Q	HI-Z	HI-Z	HI-Z	HI-Z	D	D	HI-Z	HI-Z	Q

read/output disable

When the outputs of two or more TMS 2516's and/or TMS 2532's are commoned on the same bus, the output of any particular device in the circuit can be read with no interference from the competing outputs of the other devices. If the device whose output is to be read is a TMS 2516, it should have a low-level TTL signal applied to the $\overline{CS}$ and PD/PGM pins. If it is a TMS 2532, the low-level signal is applied to the PD/$\overline{PGM}$ pin. All other devices in the circuit should have their outputs disabled by applying a high-level signal to these same pins. (PD/PGM on the TMS 2516, can be left low, but it may be advantageous to power down the device during output disable.) Output data is accessed at pins Q1 to Q8. Data can be accessed in 450 ns = $t_{a(A)}$. (On the TMS 2516 access time from $\overline{CS}$ is 150 ns = $t_{a(\overline{CS})}$, once the addresses are stable.)

power down

Active power dissipation can be cut by 80% by applying a high TTL signal to the PD/PGM (PD/$\overline{PGM}$ for the TMS 2532) pin. In this mode all outputs are in a high-impedance state.

erasure

Before programming, the TMS 2516 or TMS 2532 is erased by exposing the chip through the transparent lid to high intensity ultraviolet light (wavelength 2537 angstroms). The recommended minimum exposure dose (= UV intensity X exposure time) is fifteen watt-seconds per square centimeter. A typical 12 milliwatt per square centimeter, filterless UV lamp will erase the device in about 21 minutes. The lamp should be located about 2.5 centimeters above the chip during erasure. After erasure, all bits are in the "1" state.

start programming

After erasure (all bits in logic "1" state), logic "0's" are programmed into the desired locations. A "0" can be erased only by ultraviolet light. The programming mode is achieved when V_{PP} is 25 V and $\overline{CS}$ (for TMS 2516 only) is at V_{IN}. Data is presented in parallel (8 bits) on pins Q1 to Q8. Once addresses and data are stable, a 50 millisecond high TTL pulse (low for the TMS 2532) should be applied to the PGM pin at each address location to be programmed. Maximum pulse width is 55 milliseconds. Locations can be programmed in any order. More than one TMS 2516 or TMS 2532 can be programmed when the devices are connected in parallel.

inhibit programming

When two or more devices (either TMS 2516 or TMS 2532, or a combination of both) are connected in parallel, data can be programmed into all devices or only chosen devices. TMS 2516's not intended to be programmed (i.e., inhibited) should have a low level applied to the PD/PGM pin and a high-level applied to the $\overline{CS}$ pin. TMS 2532's not intended to be programmed should have a high level applied to PD/$\overline{PGM}$.

program verification

A verify is done to see if the device was programmed correctly. A verify can be done at any time. It can be done on each location immediately after that location ia programmed. To do a verify on the TMS 2516 Vpp may be kept at + 25 V. (Verify on the TMS 2532 is the read operation.)

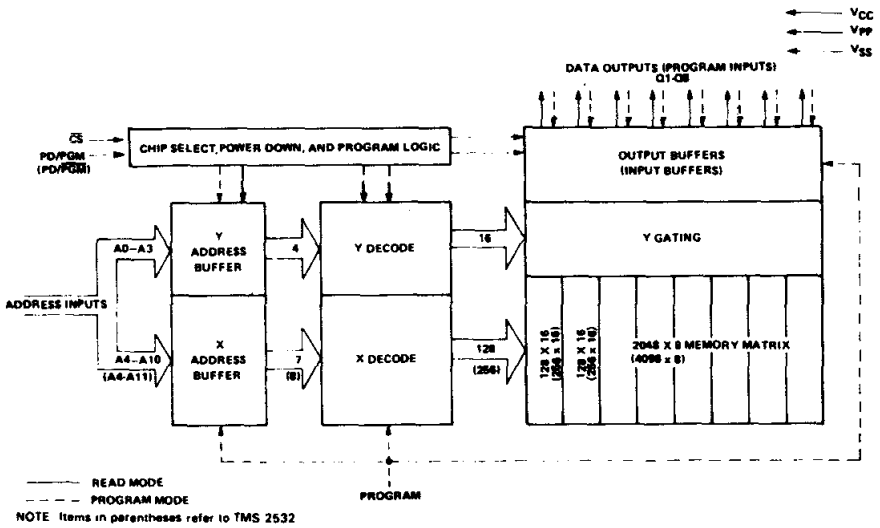
absolute maximum ratings over operating free-air temperature range (unless otherwise noted)*

Supply voltage, V_{CC} (see Note 1)	−0.3 to 6 V
Supply voltage, V_{PP} (see Note 1)	−0.3 to 28 V
All input voltages (see Note 1)	−0.3 to 6 V
Output voltage (operating with respect to V_{SS})	−0.3 to 6 V
Operating free-air temperature range	0°C to 70°C
Storage temperature range	−55°C to 125°C

NOTE 1: Under absolute maximum ratings, voltage values are with respect to the most-negative supply voltage, V_{SS} (substrate).

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

functional block diagram



recommended operating conditions

PARAMETER	TMS 2516			TMS 2532			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC} (see Note 2)	4.75	5	5.25	4.75	5	5.25	V
Supply voltage, V_{PP} (see Note 3)	V_{CC} 0.6	V_{CC}	V_{CC}+0.6	V_{CC} 0.6	V_{CC}	V_{CC}+0.6	V
Supply voltage, V_{SS}		0			0		V
High-level input voltage, V_{IH}	2.0		V_{CC}+1	2.2		V_{CC}+1	V
Low-level input voltage, V_{IL}	−0.1		0.8	−0.1		0.65	V
Read cycle time, $t_{c(rd)}$	450			450			ns
Operating free-air temperature, T_A	0		70	0		70	°C

NOTES: 2. V_{CC} must be applied before or at the same time as V_{PP} and removed after or at the same time as V_{PP}. The device must not be inserted into or removed from the board when V_{PP} is applied so that the device is not damaged.

3. V_{PP} can be connected to V_{CC} directly (except in the program mode). V_{CC} supply current in this case would be $I_{CC} + I_{PP}$. Tolerance of ± .6 volts enables the V_{PP} pin to be switched from V_{CC} (read) to 25 volts (programming) using a drive circuit. During programming, V_{PP} must be maintained at 25V (± 1V).

electrical characteristics over full ranges of recommended operating conditions

PARAMETER			TEST CONDITIONS	TMS 2516			TMS 2532			UNIT
				MIN	TYP†	MAX	MIN	TYP†	MAX	
V_{OH}	High-level output voltage		I_{OH} = −400 μA	2.4			2.4			V
V_{OL}	Low-level output voltage		I_{OL} = 2.1 mA			0.45			0.45	V
I_I	Input current (leakage)		V_I = 5.25V			10			10	μA
I_O	Output current (leakage)		V_O = 5.25V			10			10	μA
I_{PP1}	V_{PP} supply current	TMS 2516	V_{PP} = 5.85V, PD/PGM = V_{IL}			6			12	mA
		TMS 2532	V_{PP} = 5.85V, PD/$\overline{PGM}$ = V_{IL}							
I_{PP2}	V_{PP} supply current (during program pulse)	TMS 2516	PD/PGM = V_{IH}			30			30	mA
		TMS 2532	PD/$\overline{PGM}$ = V_{IL}							
I_{CC1}	V_{CC} supply current (standby)	TMS 2516	PD/PGM = V_{IH}		10	25		10	25	mA
		TMS 2532	PD/$\overline{PGM}$ = V_{IH}							
I_{CC2}	V_{CC} supply current (active)	TMS 2516	$\overline{CS}$ = PD/PGM = V_{IL}		57	100		80	160	mA
		TMS 2532	PD/$\overline{PGM}$ = V_{IL}							

†Typical values are at T_A = 25°C and nominal voltages.

capacitance over recommended supply voltage and operating free-air temperature range f = 1 MHz

PARAMETER		TEST CONDITIONS	TYP†	MAX	UNIT
C_i	Input capacitance	V_I = 0 V, f = 1 MHz	4	6	pF
C_o	Output capacitance	V_O = 0 V, f = 1 MHz	8	12	pF

†All typical values are T_A = 25°C and nominal voltage.

switching characteristics over full ranges of recommended operating conditions, (unless otherwise noted)

PARAMETER		TEST CONDITIONS (SEE NOTES 4 AND 5)	MIN	TYP†	MAX	UNIT
$t_{a(A)}$	Access time from address	C_L = 100 pF, 1 Series 74 TTL load, $t_r \leqslant 20$ ns, $t_f \leqslant 20$ ns		280	450	ns
$t_{a(\overline{CS})}$	Access time from chip select (TMS 2516 only)				120	ns
$t_{a(PR)}$	Access time from PD/PGM (PD/$\overline{PGM}$ for TMS 2532)			280	450	ns
t_{PVX}	Output not valid from address change		0			ns
t_{PXZ}	Output disable time from chip deselect during read only		0		100	ns
t_{PXZ}	Output disable time from chip deselect during program and program verify				120	ns
t_{PXZ}	Output disable time from PD/PGM (PD/$\overline{PGM}$ for TMS 2532) during standby		0		100	ns

†All typical values are at T_A = 25°C and nominal voltages.

recommended timing requirements for programming T_A = 25°C (see Note 4)

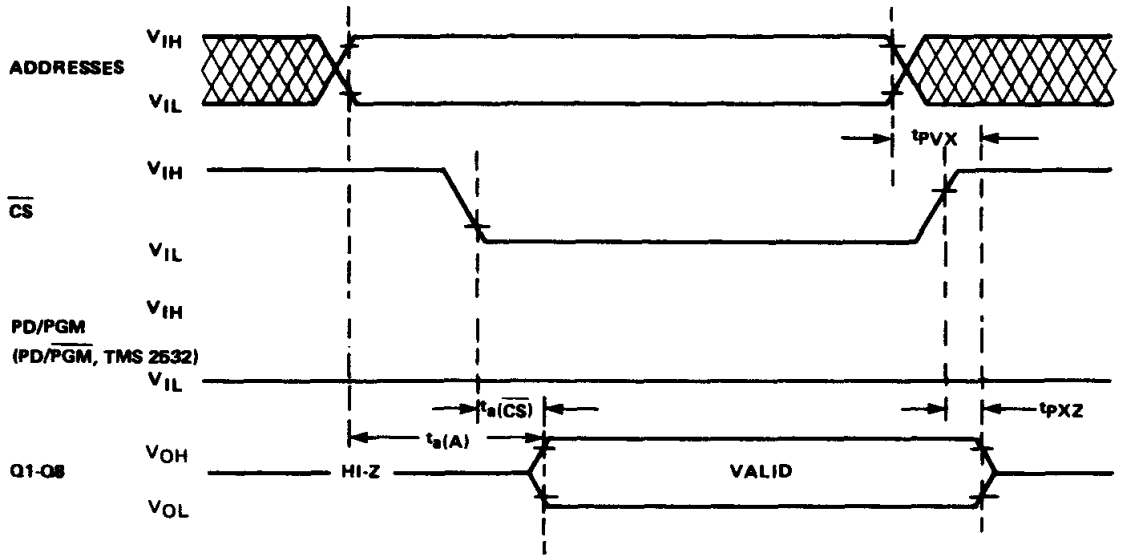
PARAMETER		MIN	TYP†	MAX	UNIT
$t_{w(PR)}$	Pulse width, program pulse	45	50	55	ms
$t_{r(PR)}$	Rise time, program pulse	5			ns
$t_{f(PR)}$	Fall time, program pulse	5			ns
$t_{su(A)}$	Address setup time	2			μs
$t_{su(\overline{CS})}$	Chip-select setup time	2			μs
$t_{su(D)}$	Data setup time	2			μs
$t_{su(VPP)}$	Setup time from Vpp	0			ns
$t_{h(A)}$	Address hold time	2			μs
$t_{h(\overline{CS})}$	Chip-select hold time (TMS 2516 only)	2			μs
$t_{h(D)}$	Data hold time	2			μs
$t_{h(PR)}$	Program pulse hold time (TMS 2532 only)	0			ns
$t_{h(VPP)}$	Vpp hold time (TMS 2532 only)	0			ns

†Typical values are at nominal voltages.

NOTES: 4. For all switching characteristics and timing measurements, input pulse levels are 0.65 V to 2.2 V and V_{PP} = 25 V ± 1 V during programming.

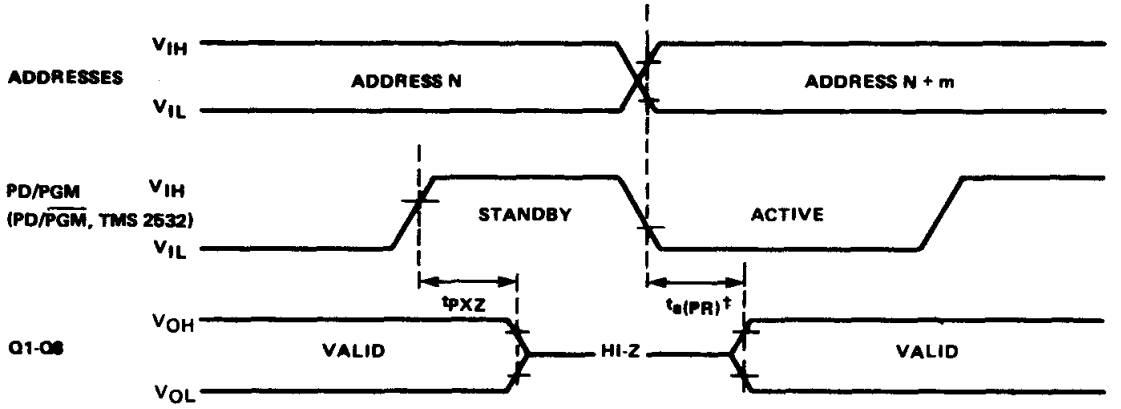
5. Common test conditions apply for t_{PXZ} except during programming. For $t_{a(A)}$, $t_{a(\overline{CS})}$, and t_{PXZ}, PD/PGM = $\overline{CS}$ = V_{IL} for the TMS 2516 and PD/$\overline{PGM}$ = V_{IL} for the TMS 2532.

read cycle timing



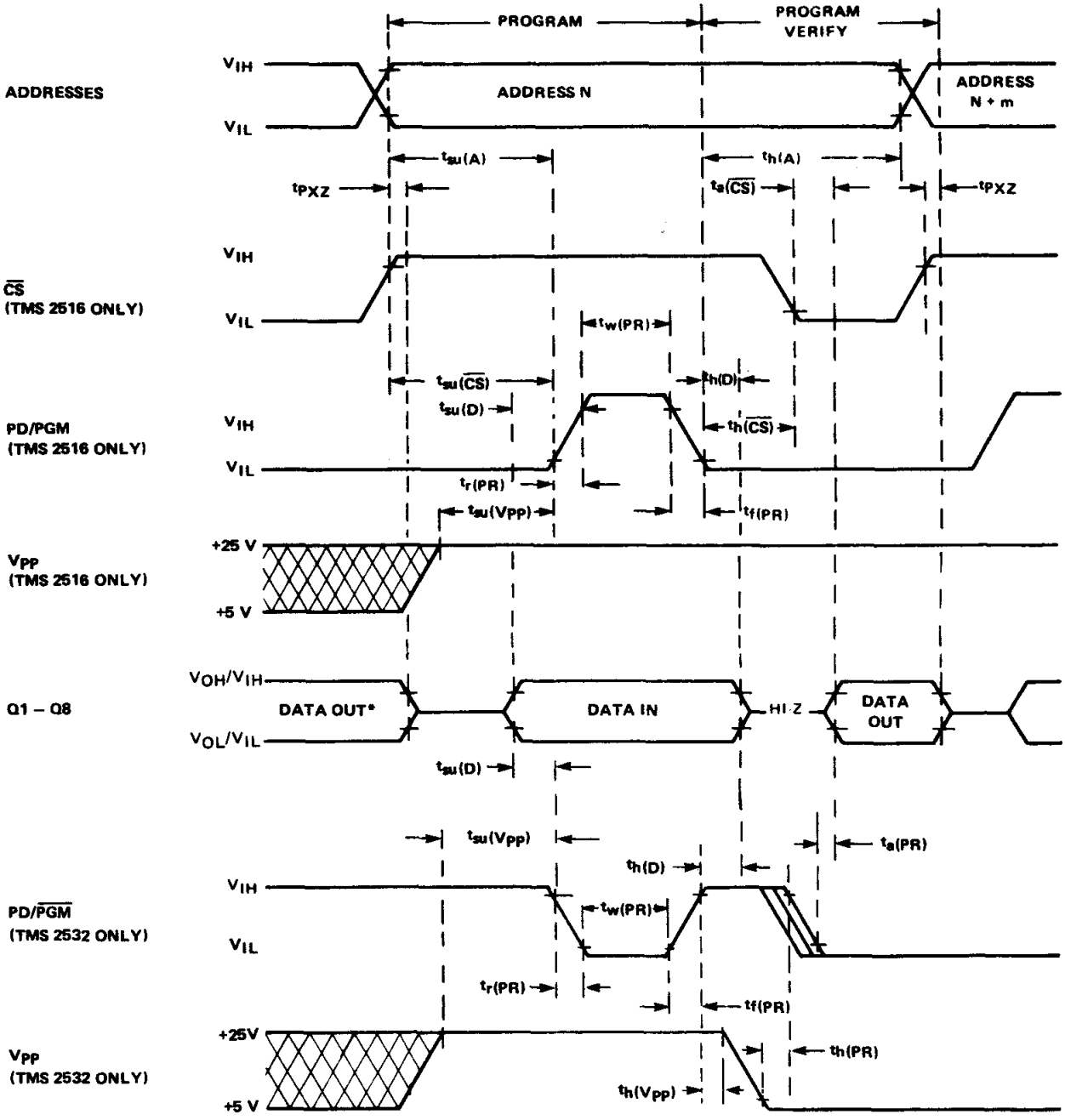
NOTE: There is no chip select pin on the TMS 2532.
The chip-select function is incorporated in the power-down mode.

standby mode



NOTE: $\overline{CS}$ (TMS 2516) must be in low state during Active Mode, "Don't Care" otherwise.
† $t_{a(PR)}$ referenced to PD/PGM (PD/$\overline{PGM}$ for TMS 2532) or the address, whichever occurs last.

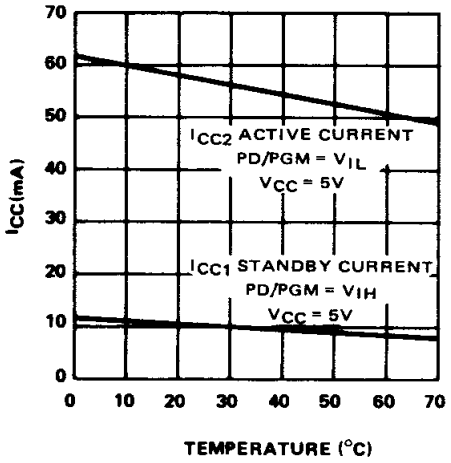
program cycle timing



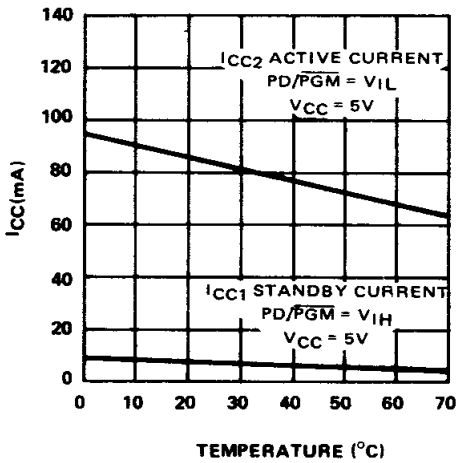
NOTE: There is no chip select pin on the TMS 2532. Chip select is incorporated in the power down mode.
CS (TMS 2516) is in "don't care" state.
*HI-Z for the TMS 2532.

typical device characteristics (read mode)

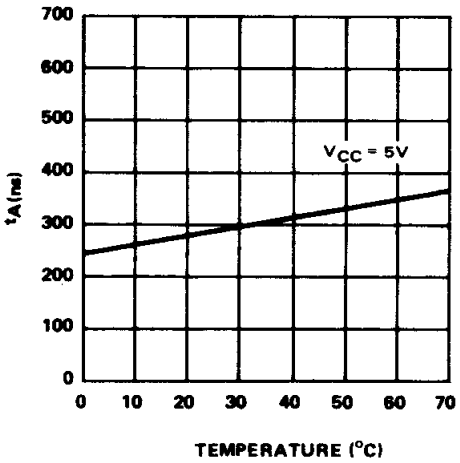
TMS 2516
I_{CC} CURRENT
vs.
TEMPERATURE



TMS 2532
I_{CC} CURRENT
vs.
TEMPERATURE



TMS 2516 and TMS 2532
ACCESS TIME
vs.
TEMPERATURE

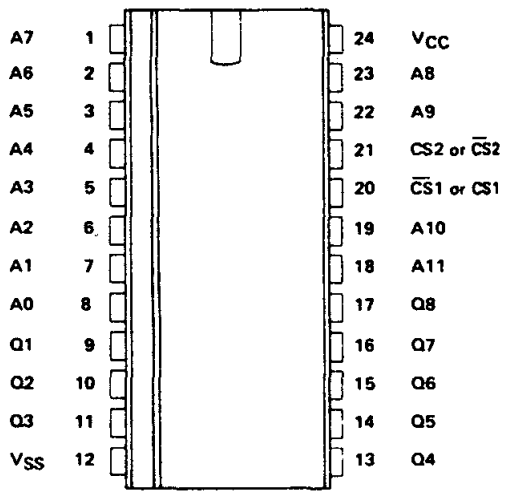


TMS 4732 JL, NL
4096-WORD BY 8-BIT READ-ONLY MEMORY

MAY 1977

- 4096 x 8 Organization
- All Inputs and Outputs TTL-Compatible
- Fully Static (No Clocks, No Refresh)
- Single 5-V Power Supply
- Maximum Access Time . . . 450 ns
- Minimum Cycle Time . . . 450 ns
- Typical Power Dissipation . . . 580 mW
- 3-State Outputs for OR-Ties
- Pin-Compatible with TMS 4700, TMS 2708 and Intel 8316B
- Two Output Enable Controls for Chip Select Flexibility
- N-Channel Silicon-Gate Technology

24-PIN CERAMIC AND PLASTIC
DUAL-IN-LINE PACKAGES
(TOP VIEW)



description

The TMS 4732 is a 32,768-bit read-only memory organized as 4096 words of 8-bit length. This makes the TMS 4732 ideal for microprocessor based systems. The device is fabricated using N-channel silicon-gate technology for high speed and simple interface with bipolar circuits.

All inputs can be driven directly by Series 74 TTL circuits without the use of any external pull-up resistor. Each output can drive one Series 74 or 74S load without external resistors. The data outputs are three-state for OR-tieing multiple devices on a common bus. Two chip select controls allow data to be read. These controls are programmable, providing additional system decode flexibility. The data is always available; it is not dependent on external CE clocking.

The TMS 4732 is designed for high-density fixed-memory applications such as logic function generation and microprogramming. Systems utilizing the TMS 4700 1024 x 8-bit ROM or the TMS 2708 1024 x 8-bit EPROM can expand to the 4096 x 8-bit TMS 4732 with changes only to pins 18, 19, and 21. To upgrade from the 8316B, simply replace CS2 with A11 on pin 18.

This ROM is supplied in 24-pin dual-in-line plastic (NL suffix) or ceramic (JL suffix) packages designed for insertion in mounting-hold rows on 600-mil centers. The device is designed for operation from 0°C to 70°C.

operation

address (A0–A11)

The address-valid interval determines the device cycle time. The 12-bit positive-logic address is decoded on-chip to select one of 4096 words of 8-bit length in the memory array. A0 is the least-significant bit and A11 the most-significant bit of the word address.

chip select (CS1 and CS2)

Each chip select control can be programmed during mask fabrication to be active with either a high or a low level input. When both chip select signals are active, all eight outputs are enabled and the eight-bit addressed word can be read. When either chip select is not active, all eight outputs are in a high-impedance state.

PRELIMINARY DATA SHEET:
Supplementary data will be published at a later date.

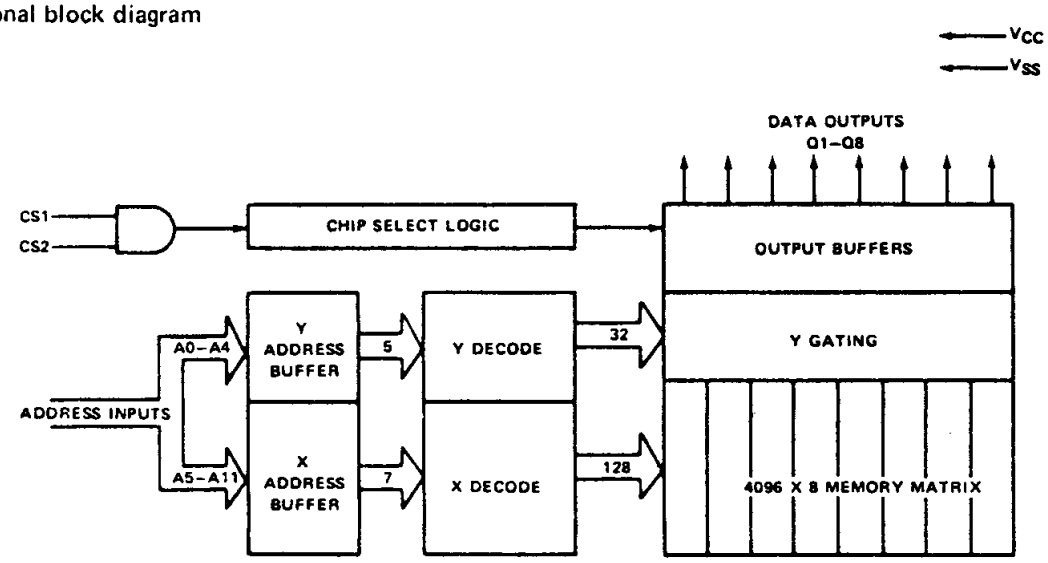
TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 5012 • DALLAS, TEXAS 75222

data out (Q1–Q8)

The eight outputs must be enabled by both chip select controls before the output word can be read. Data will remain valid until the address is changed or the outputs are disabled (chip deselected). When disabled, the three-state outputs are in a high-impedance state. Q1 is considered the least-significant bit, Q8 the most-significant bit.

The outputs will drive TTL circuits without external components.

functional block diagram



absolute maximum ratings

Supply voltage to ground potential (see Note 1) . . . . . −0.5 to 7 V
Applied output voltage (see Note 1) . . . . . −0.5 to 7 V
Applied input voltage (see Note 1) . . . . . −0.5 to 7 V
Power dissipation . . . . . 1 W
Ambient operating temperature . . . . . 0°C to 150°C
Storage temperature . . . . . −55°C to 150°C

Note 1: Voltage values are with respect to V_{SS}.

recommended operating conditions

PARAMETER	MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}	4.75	5	5.25	V
High-level input voltage, V_{IH}	2	2.4	V_{CC}	V
Low-level input voltage, V_{IL}	V_{SS}	0.5	0.65	V
Read cycle time, $t_{c(rd)}$	450			ns
Operating free-air temperature, T_A	0		70	°C

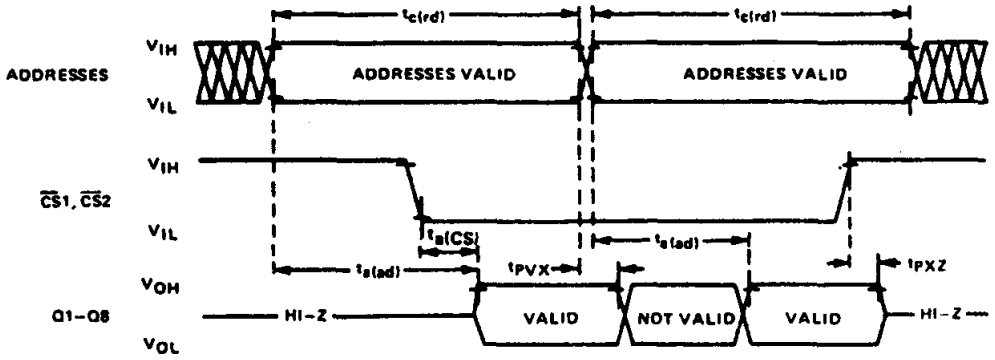
electrical characteristics, TA = 0°C to 70°C, VDD = 5 V +5% (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	MAX	UNIT
V_{OH}	High-level output voltage	V_{CC} = 4.75 V,	I_{OH} = −200 μA	2.4	V_{CC}	V
V_{OL}	Low-level output voltage	V_{CC} = 4.75 V,	I_{OL} = 2 mA		0.4	V
I_I	Input current	V_{CC} = 5.25 V,	0V ≤ V_{IN} ≤ 5.25 V		10	μA
I_{OZ}	Output leakage current	V_O = 0.4 V to V_{CC},	Outputs disabled		±10	μA
I_{CC}	Supply current from V_{CC}	V_{CC} = 5.25 V,	V_I = V_{CC} output not loaded		150	mA
C_i	Input capacitance	V_O = 0 V, f = 1 MHz	T_A = 25°C,		7	pF
C_o	Output capacitance	V_O = 0 V, f = 1 MHz	T_A = 25°C,		10	pF

switching characteristics, T_A = 0°C to 70°C, V_{CC} = 5 V +5%, 1 series 74 TTL load, C_L = 100 pF

PARAMETER		MIN	MAX	UNIT
$t_{a(ad)}$	Access time from address		450	ns
$t_{a(CS)}$	Access time from chip select		200	ns
t_{PVX}	Previous output data valid after address change		450	ns
t_{PXZ}	Output disable time from chip select		200	ns

read cycle timing



PROGRAMMING DATA

PROGRAMMING REQUIREMENTS: The TMS 4732JL, NL is a fixed program memory in which the programming is performed by TI at the factory during manufacturing cycle to the specific customer inputs supplied in the format below. The device is organized as 4096 8-bit words with address locations numbered 0 to 4095. The 8-bit words can be coded as a 2-digit hexadecimal number between 00 and FF. All data words and addresses in the following format are coded in hexadecimal numbers. In coding, all binary words must be in positive logic before conversion to hexadecimal. Q1 is considered the least significant bit and Q8 the most significant bit. For addresses, A0 is least significant bit and A11 is the most significant bit.

Every card should include the TI Custom Device Number in the form ZAXXXX (4 digit number to be assigned by TI) in columns 75 through 80.

PROGRAMMABLE CHIP SELECTS: The chip select inputs shall be programmed according to the data punched in columns 73 and 74. Every card should include in column 73 a 1 if the output is to be enabled with a high level at CS2 or a 0 (zero) to enable the output with a low level at CS2. The column 74 entry is the same for programming CS1.

PROGRAMMED DATA FORMAT: The format for the cards to be supplied to TI to specify the data to be programmed is provided below. The card deck for each device consists of 128 cards with each card containing data for 32 memory locations.

CARD COLUMN	HEXADECIMAL FORMAT
1 to 3	Hexadecimal address of first word on the card
4	Blank
5 to 68	Data. Each 8-bit memory byte is represented by two ASCII characters to represent a hexadecimal value of '00' to 'FF'.
69, 70	Checksum. The checksum is the negative of the sum of all 8-bit bytes in the record from columns 1 to 68, evaluate modulo 256 (carry from high order bit ignored). (For purposes of calculating the checksum, the value of Column 4 is defined to be zero). Adding together, modulo 256, all 8-bit bytes from Column 1 to 68 (Column 4 = 0), then adding the checksum, results in zero.
71, 72	Blank
73	One (1) or zero (0) for CS2
74	One (1) or zero (0) for CS1
75, 76	ZA
77 to 80	XXXX (4 digit number assigned by TI)

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 5012 • DALLAS, TEXAS 75222

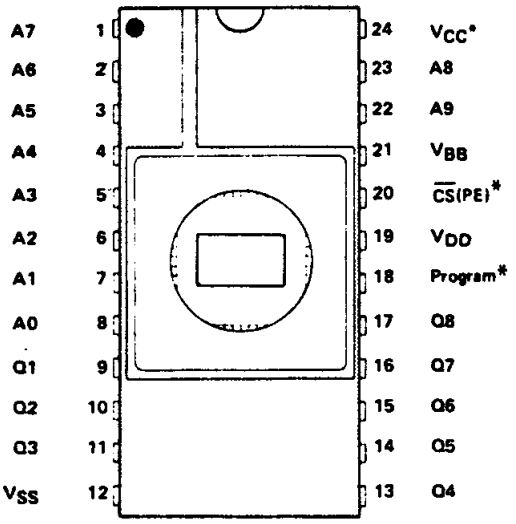
MOS
LSI

TMS 2708 JL, TMS 27L08 AND TMS 2716 JL
8K AND 16K ERASABLE
PROGRAMMABLE READ-ONLY MEMORIES

OCTOBER 1977

- 2708 JL and 27L08 JL – 1024 X 8 Organization
- 2716 JL 2048 X 8 Organization
- All Inputs and Outputs Fully TTL-Compatible
- Static Operation (No Clocks, No Refresh)
- Maximum Access Time . . . 450 ns
- Minimum Cycle Time . . . 450 ns
- 3-State Outputs for OR-Ties
- N-Channel Silicon-Gate Technology
- 8-Bit Output for Use in Microprocessor-Based Systems
- Low Power
  - TMS 27L08 . . . 245 mW (Typical)
  - TMS 2716 . . . 315 mW (Typical)
- 10% Power Supply Tolerance (TMS 27L08 Only)
- Plug-Compatible Pin-Outs Allowing Interchangeability/Upgrade to 16K With Minimum Board Change

24-PIN CERAMIC
DUAL-IN-LINE PACKAGE
(TOP VIEW)



*For 2716 JL Only: Pin:
18 $\overline{CS}$ (Program)
20 A10
24 V_{CC} (PE)

description

The TMS 2708 JL, TMS 27L08 JL, and TMS 2716 JL are ultra-violet light-erasable, electrically programmable read only memories. The TMS 2708 JL and TMS 27L08 JL have 8,192 bits organized as 1024 words of 8-bit length. The TMS 2716 JL has 16,384 bits organized as 2048 words of 8-bit length. The devices are fabricated using N-channel silicon-gate technology for high speed and simple interface with MOS and bipolar circuits. All inputs (including program data inputs) can be driven by Series 74 TTL circuits without the use of external pull-up resistors and each output can drive one Series 74 TTL circuit without external resistors. The TMS 27L08 guarantees 200 mV dc noise immunity in the high state and 250 mV in the low state. It will also directly drive one Series 74, 74S, or 74LS TTL circuit. TMS 2716 also guarantees 250 mV dc noise immunity in the low state. The data outputs for all three circuits are three-state for OR-tying multiple devices on a common bus. The TMS 2716 is plug-in compatible with the TMS 2708 and the TMS 27L08. Pin compatible mask programmed ROMs are available for large volume requirements.

These EPROMs are designed for high-density fixed-memory applications where fast turn arounds and/or program changes are required. They are supplied in a 24-pin dual-in-line ceramic (JL suffix) packages designed for insertion in mounting-hole rows on 600-mil centers. They are designed for operation from 0°C to 70°C.

operation (read mode)

address (A0-A9/A10)
The address-valid interval determines the device cycle time. The 10-bit (11 bit TMS 2716) positive-logic address is decoded on-chip to select one of 1024 words (2048 words TMS 2716) of 8-bit length in the memory array. A0 is the least-significant bit and A9 (A10 TMS 2716) most-significant bit of the word address.

chip select, program enable [$\overline{CS}$ (PE)] and chip select, program [$\overline{CS}$ (Program)]
When the chip select is low, all eight outputs are enabled and the eight-bit addressed word can be read. When the chip select is high, all eight outputs are in a high-impedance state.

PRELIMINARY DATA SHEET:
Supplementary data will be published at a later date.

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 5012 • DALLAS, TEXAS 75222

TMS 2708, TMS 27L08
When the chip select program enable is brought to V_{DD}, the outputs become inputs and the EPROM is ready for programming.

TMS 2716
In the program mode, the chip select feature does not function as pin 18 inputs only the program pulse. The program mode is selected by the V_{CC}(PE) pin. Either 0 V or +12 V on this pin will cause the TMS 2716 to assume program cycle.

data out (Q1-Q8)
The chip must be selected before the eight-bit output word can be read. Data will remain valid until the address is changed or the chip is deselected. When deselected, the three-state outputs are in a high-impedance state. The outputs will drive TTL circuits without external components.

TMS 2708, TMS 27L08
The program pin must be held below V_{CC} in the read mode.

operation (program mode)

erase
Before programming, the TMS 2708, TMS 27L08 and TMS 2716 are erased by exposing the chip through the transparent lid to high-intensity ultraviolet light (wavelength 2537 angstroms). The recommended exposure is ten watt-seconds per square centimeter. This can be obtained by, for instance, 20 to 30 minutes exposure of a filterless Model S52 short wave UV lamp about 2.5 centimeters above the EPROM. After erasure all bits are in the high state.

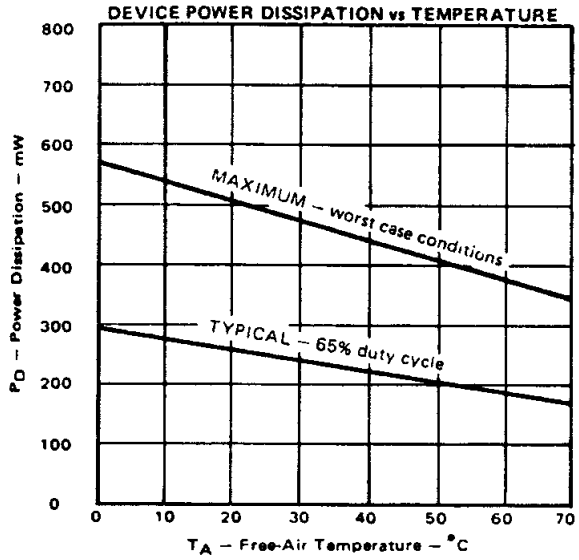
programming
Programming consists of successively depositing a small amount of charge to a selected memory cell that is to be changed from the erased high state to the low state. A low can be changed to a high only by erasure. Programming is normally accomplished on a PROM or EPROM Programmer, an example of which is TI's Universal PROM Programming Module in conjunction with the 990 prototyping system. Programming must be done at room temperature (25°C) only.

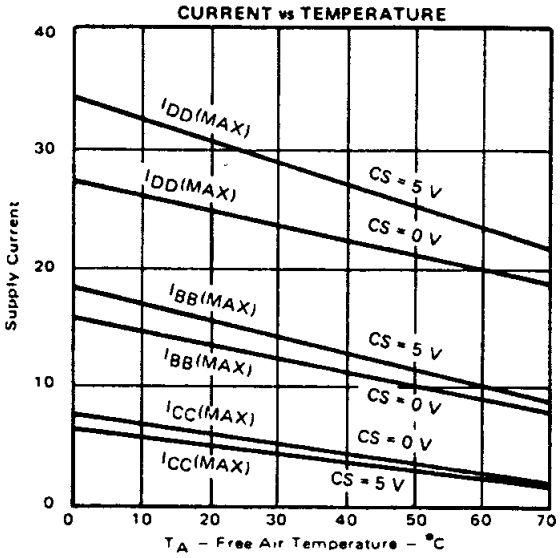
to start programming (see program cycle timing diagram on page 148)
First bring the $\overline{CS}$ (PE) pin (for the TMS 2708, TMS 27L08) to +12 V or the V_{CC}(PE) (for the TMS 2716) to +12 V or 0 V to disable the outputs and convert them to inputs. This pin is held high for the duration of the programming sequence. The first word to be programmed is addressed (it is customary to begin with the "0" address) and the data to be stored is placed on the Q1-Q8 program inputs. Then a +26-V program pulse is applied to the program pin. After 0.1 to 1.0 milliseconds the program pin is brought back to 0 V. After at least one microsecond the word address is sequentially changed to the next location, the new data is set up and the program pulse is applied.

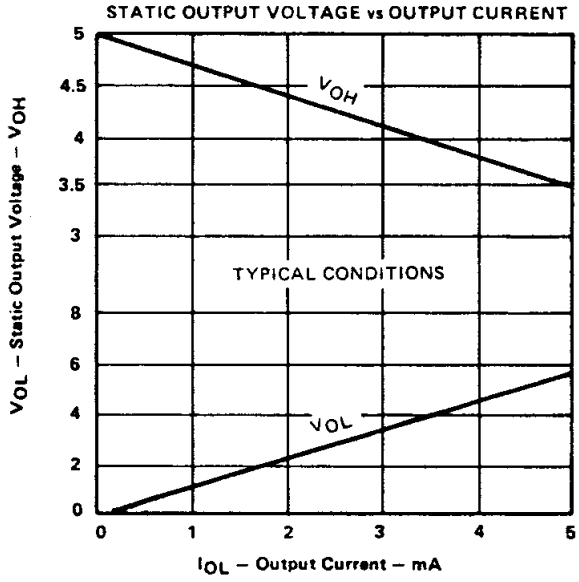
Programming continues in this manner until all words have been programmed. This constitutes one of N program loops. The entire sequence is then repeated N times with N x $t_{w(PR)} \geq 100$ ms. Thus, if $t_{w(PR)} = 1$ ms; then N = 100, the minimum number of program loops required to program the EPROM.

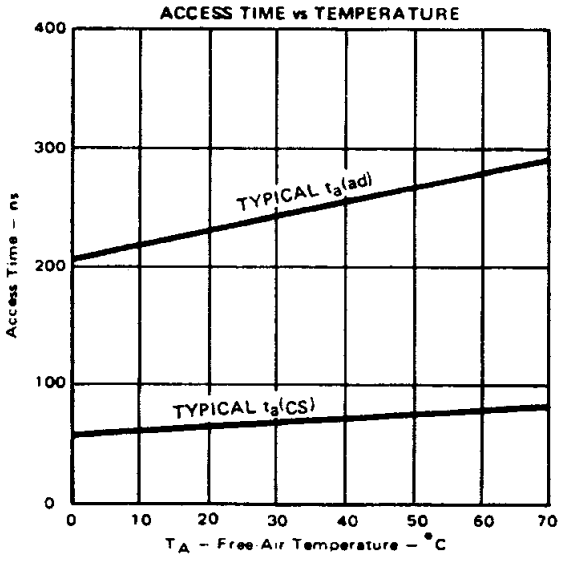
to stop programming
After cycling through the N program loops, the last program pulse is brought to 0 V, then for the TMS 2708 and TMS 27L08, Program Enable [CS(PE)] is brought to V_{IL} (for the TMS 2716 Program Enable [V_{CC}(PE)] is brought back to ±5 volts) which takes the device out of the program mode. The data supplied by the programmer must be removed before the address is changed since the program inputs are now data outputs and a change of address could cause a voltage conflict on the output buffer. Q1-Q8 outputs are invalid up to 10 microseconds after the program enable pin is brought from $V_{IH(PE)}$ to V_{IL} [$V_{IL(PE)}$ on TMS 2716].

TYPICAL TMS 27L08 JL CHARACTERISTICS









capacitance over recommended supply voltage range and operating free-air temperature range, f = 1 MHz

	PARAMETER	TYP†	MAX	UNIT
C_i	Input capacitance [except $\overline{CS}$ (Program) for the TMS 2716]	4	6	pF
$C_{i(CS)}$	$\overline{CS}$ (Program) input capacitance for TMS 2716 only	20	30	pF
C_o	Output capacitance	8	12	pF

†All typical values are at T_A = 25°C and nominal voltages.

switching characteristics over recommended supply voltage range and operating free-air temperature range

	PARAMETER	TEST CONDITIONS	MIN	MAX	UNIT
$t_{a(ad)}$	Access time from address	C_L = 100 pF 1 Series 74 TTL Load $t_{f(CS)}$, $t_{f(ad)}$ = 20 ns		450	ns
$t_{a(CS)}$	Access time from $\overline{CS}$			120	ns
t_{PVX}	Output invalid from address change		0		ns
t_{PXZ}	Output disable time		0	120	ns
$t_{c(rd)}$	Read Cycle time		450		ns

T_A = 25°C program characteristics over recommended supply voltage range

	PARAMETER	MIN	MAX	UNIT
$t_{w(PR)}$	Pulse width, program pulse	0.1	1	ms
t_T	Transition times (except program pulse)		20	ns
$t_{T(PR)}$	Transition times, program pulse	50	2000	ns
$t_{su(ad)}$	Address setup time	10		µs
$t_{su(da)}$	Data setup time	10		µs
$t_{su(PE)}$	Program enable setup time	10		µs
$t_{h(ad)}$	Address hold time	1000		ns
$t_{h(ad,da\ R)}$	Address hold time after program input data stopped	0		ns
$t_{h(da)}$	Data hold time	1000		ns
$t_{h(PE)}$	Program enable hold time	500		ns
$t_{CL,adX}$	Delay time, $\overline{CS}$(Program) low to address change	0		ns

recommended operating conditions

PARAMETER	TMS 2708, TMS 2716			TMS 27L08			UNIT
	MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{BB}	−4.75	−5	−5.25	−4.5	−5	−5.5	V
Supply voltage, V_{CC}	4.75	5	5.25	4.5	5	5.5	V
Supply voltage, V_{DD}	11.4	12	12.6	10.8	12	13.2	V
Supply voltage, V_{SS}		0			0		V
High-level input voltage, V_{IH} (except program and program enable)	2.4		V_{CC}+1	2.2		V_{CC}+1	V
High-level program enable input voltage, $V_{IH(PE)}$	11.4	12	12.6	10.8	12	13.2	V
High-level program input voltage, $V_{IH(PR)}$	25	26	27	25	26	27	V
Low-level input voltage, V_{IL} (except program)	V_{SS}		0.65	V_{SS}		0.65	V
Low-level program input voltage, $V_{IL(PR)}$ Note: $V_{IL(PR)}$ max $\leq V_{IH(PR)}$ −25 V	V_{SS}		1	V_{SS}		1	V
High-level program pulse input current (sink), $I_{IH(PR)}$			40			40	mA
Low-level program pulse input current (source), $I_{IL(PR)}$			3			3	mA
Operating free-air temperature, T_A	0		70	0		70	°C

electrical characteristics over full ranges of recommended operating conditions (unless otherwise noted)

PARAMETER		TEST CONDITIONS		TMS 2708			TMS 27L08			TMS 2716			UNITS
				MIN	TYP†	MAX	MIN	TYP†	MAX	MIN	TYP†	MAX	
V_{OH}	High-level output voltage	I_{OH} = −100 μA		3.7			3.7			3.7			V
		I_{OH} = −1 mA		2.4			2.4			2.4			
V_{OL}	Low-level output voltage	I_{OL} = 1.6 mA				0.45			0.40			0.45	V
I_I	Input current (leakage)	V_I = 0 V to 5.25 V			1	10		1	10		1	10	μA
I_O	Output current (leakage)	$\overline{CS}$(PE) = 5 V	TMS 2708, 27L08		1	10		1	10		1	10	μA
		$\overline{CS}$ (Program) = 5 V	TMS 2716										
I_{BB}	Supply current from V_{BB}	All inputs high			30	45		9	18		10	20	mA
I_{CC}	Supply current from V_{CC}	$\overline{CS}$(PE) = 5 V	TMS 2708, 27L08		6	10		9	6		1	8	mA
		$\overline{CS}$ (Program) = 5 V	TMS 2716										
I_{DD}	Supply current from V_{DD}	For I_{DD} MAX,	T_A = 0°C (worst case)		50	65		20	34		26	45	mA
I_{PE}	Supply current from PE on V_{CC} Pin	V_{PE} = V_{DD}	TMS 2716 only								2	4	mA
$P_{D(AV)}$	Power Dissipation	T_A = 70°C				800			350			540	mW
		T_A = 0°C	$\overline{CS}$ = 0V					245	475		315	595	
		T_A = 0°C	$\overline{CS}$ = +5V					290	580		375	720	

†All typical values are at T_A = 25°C and nominal voltages.

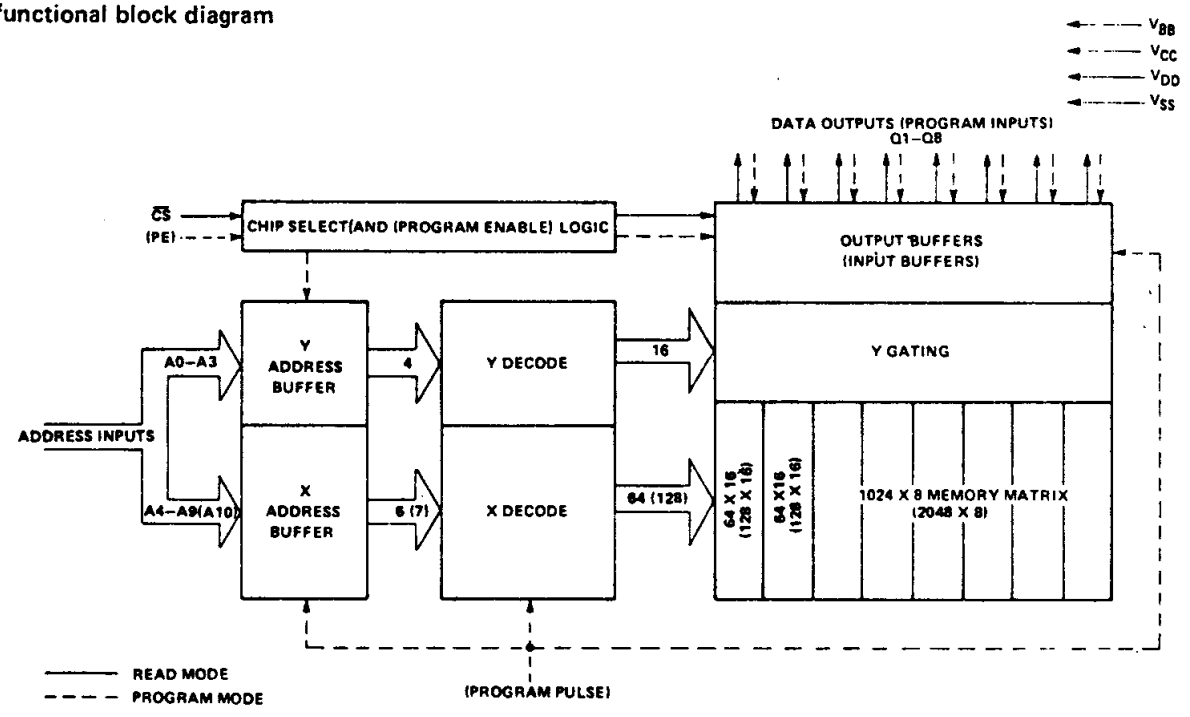
absolute maximum ratings over operating free-air temperature range (unless otherwise noted)*

Supply voltage, V_{CC} (see Note 1)	−0.3 to 15 V
Supply voltage, V_{DD} (see Note 1)	−0.3 to 20 V
Supply voltage, V_{SS} (see Note 1)	−0.3 to 15 V
All input voltage (except program) (see Note 1)	−0.3 to 20 V
Program Input (see Note 1)	−0.3 to 35 V
Output voltage (operating, with respect to V_{SS})	−2 to 7 V
Operating free-air temperature range	0°C to 70°C
Storage temperature range	−55°C to 125°C

NOTE 1: Under absolute maximum ratings, voltage values are with respect to the most-negative supply voltage, V_{BB} (substrate), unless otherwise noted. Throughout the remainder of this data sheet, voltage values are with respect to V_{SS}.

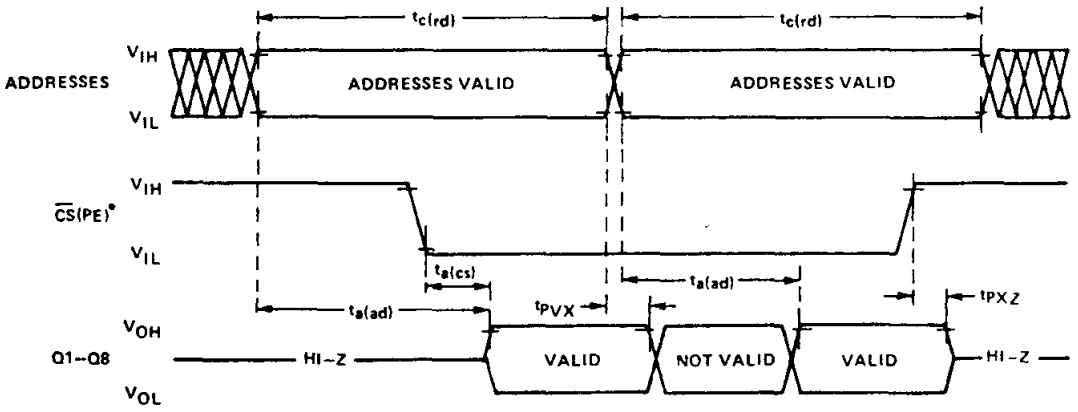
*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

functional block diagram



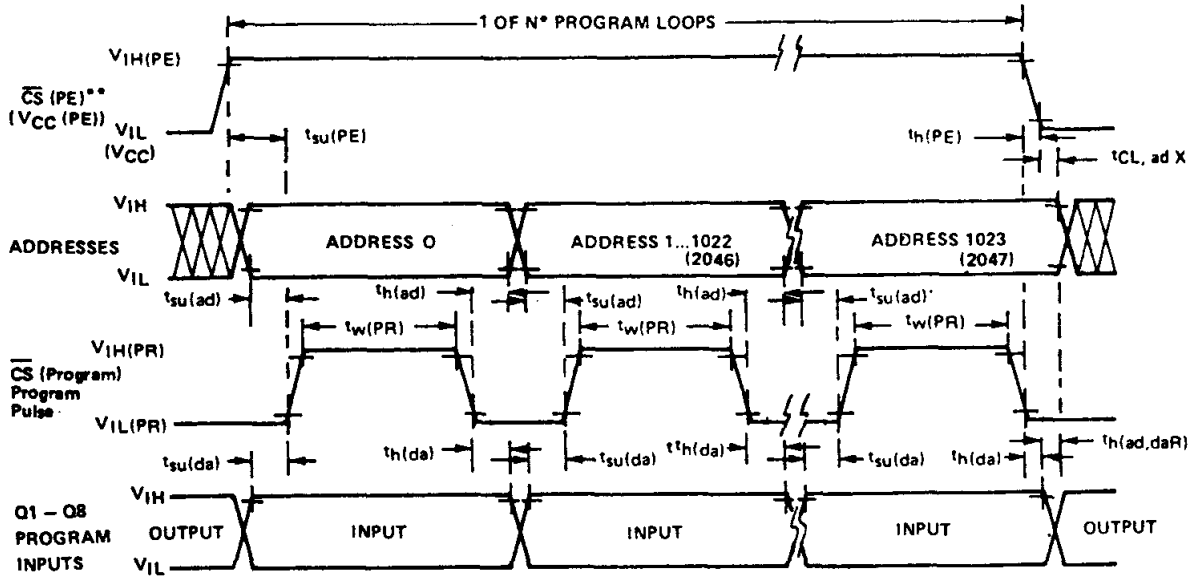
NOTE: Numbers in parentheses refer to TMS 2716

read cycle timing



* $\overline{CS}$ (Program) TMS 2716

program cycle timing



**CS (PE) is at +12 V for the TMS 2708 and 27L08 and V_{CC} (PE) is at 0 V or +12 V for the TMS 2716 through N program loops where N ⩾ 100 ms/tw (PR).

NOTE: Q1-Q8 outputs are invalid up to 10 μsec after programming [CS(PE) (V_{CC}(PE) for TMS 2716) goes low].

NOTE: Items in parentheses refer to TMS 2716.

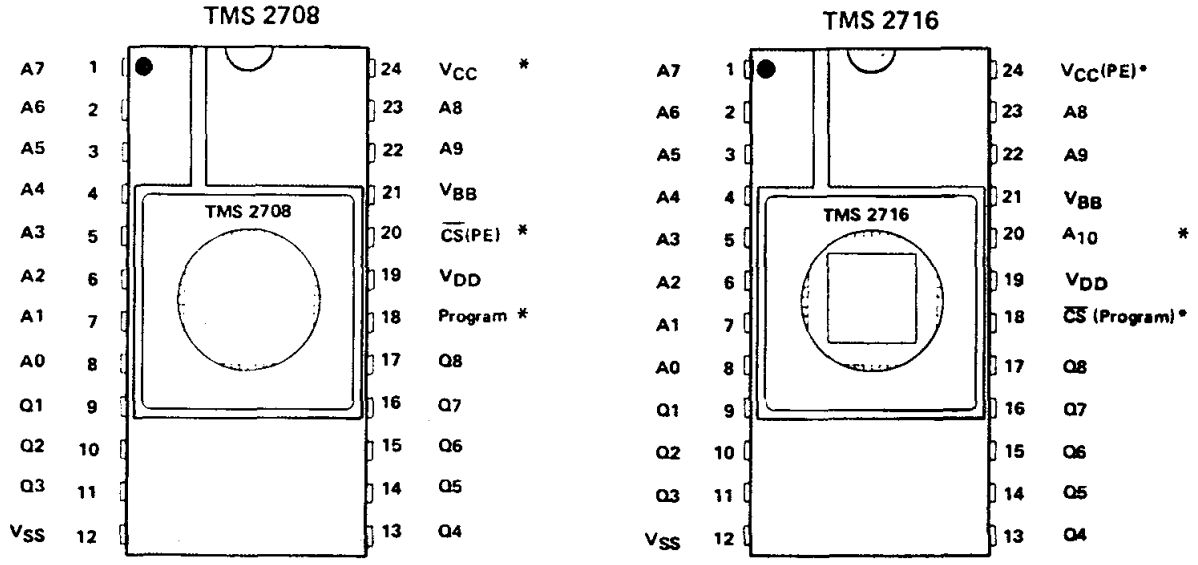
APPLICATIONS INFORMATION

Ease of Conversion From TMS 2708 To TMS 2716

A. The TMS 2716 and TMS 2708 have compatible timing, voltage and current parameters in both modes.

B. The TMS 2716 requires less power than the TMS 2708.

C. The pinouts are compatible. (See below.)



As can be seen from the above diagrams, only three pins* are modified in going from TMS 2708 to TMS 2716:

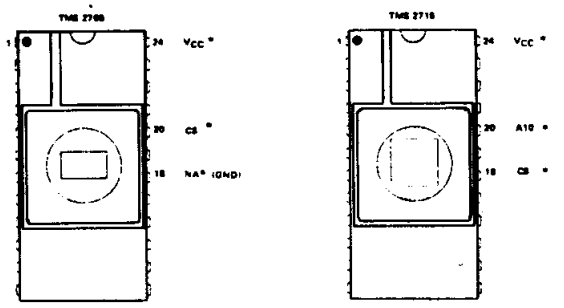
1. The additional address pin required for the 16K EPROM is located on pin 20 which displaces the $\overline{CS}$/PE functions on the TMS 2708.

2. Since V_{CC} is not required during programming, the PE function shares pin 24 with V_{CC} on the TMS 2716.

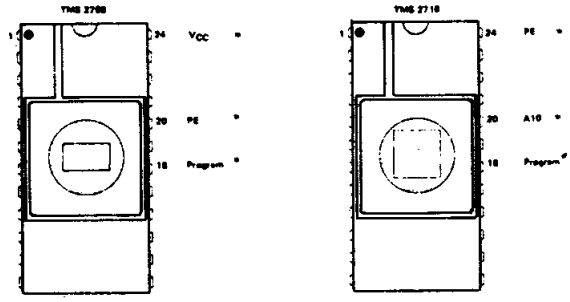
3. The $\overline{CS}$ function and program function are mutually exclusive during normal read mode (and are self-actuated complementary during the program/verify mode) and share pin 18 on the TMS 2716.

The diagrams below show how these three pins are actually utilized in the read mode and in the program mode. Only pins 18, 20, and 24 need to be shown, as all other pin connections are identical.

Read Mode



Program (Write) Mode



TMS 2716 — Easy Programmability On Existing 2708 Programmers

Existing EPROM Programmers — Upgrading To The TMS 2716

Most of the EPROM manufacturers have implemented field upgrade modifications to allow TMS 2716 programming on current EPROM programmers. This is greatly simplified because the TMS 2716 and the TMS 2708 are programmed in an identical manner. A slight modification to the socket card, an additional 1K x 8 of RAM, and an extra address signal (A10) are all that is required. All timing and voltage parameters are identical, so the upgrade is easily accomplished. Programmer manufacturers contacted to date on the TMS 2716 include: Data I/O, PRO LOG, Texas Instruments, Technico, CramerKit, Shepardson Micro Systems, Cromenco, MicroPro, Ramtek, Oliver Audio, Inc., etc. Ultraviolet Erasure lights and fixtures are available from Ultraviolet Products, Turner Designs, and others.

NOTE: Information on EPROM programmers and erasurers are provided only for user convenience and do not indicate any preference by TI.

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

MOS
LSI

TMS 4045 JL, JDL, NL; TMS 40L45 JL, JDL, NL; TMS 4047 JL, JDL, NL; TMS 40L47 JL, JDL, NL 1024-WORD BY 4-BIT STATIC RAMs

SEPTEMBER 1978

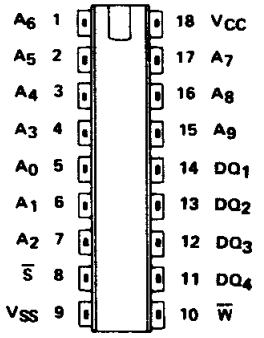
- **1024 × 4 Organization**
- **Single +5 V Supply (±10% Tolerance)**
- **High Density 300-mil (7.62 mm) 18- and 20-Pin Packages**
- **Fully Static Operation (No Clocks, No Refresh, No Timing Strobe)**
- **4 Performance Ranges:**

	ACCESS READ OR WRITE TIME (MAX)	CYCLE (MIN)
TMS 4045/L45-45, TMS 4047/L47-45	450 ns	450 ns
TMS 4045/L45-25, TMS 4047/L47-25	250 ns	250 ns
TMS 4045/L45-20, TMS 4047/L47-20	200 ns	200 ns
TMS 4045-15, TMS 4047-15	150 ns	150 ns

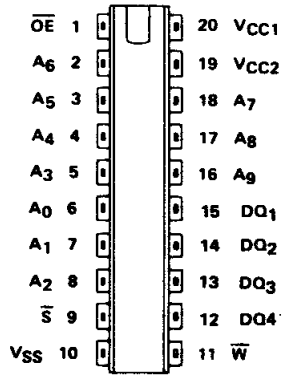
- **400 mV Guaranteed DC Noise Immunity with Standard TTL Loads — No Pull-Up Resistors Required**
- **Common I/O**
- **3-State Outputs and Chip Select Control for OR-Tie Capability**
- **Fan-Out to 2 Series 74, 1 Series 74S, or 8 Series 74LS TTL Loads**
- **Low Power Dissipation**

	MAX (OPERATING)	MAX (STANDBY)
TMS 4045	550 mW	170 mW
TMS 40L45	330 mW	110 mW
TMS 4047	550 mW	13 mW
TMS 40L47	330 mW	13 mW

TMS 4045/TMS 40L45
18-PIN CERAMIC AND PLASTIC
DUAL-IN-LINE PACKAGES
(TOP VIEW)



TMS 4047/TMS 40L47
20-PIN CERAMIC AND PLASTIC
DUAL-IN-LINE PACKAGES
(TOP VIEW)



description

This series of static random-access memories is organized as 1024 words of 4 bits each. Static design results in reducing overhead costs by elimination of refresh-clocking circuitry and by simplification of timing requirements. Because this series is fully static, chip select may be tied low to further simplify system timing. Output data is always available during a read cycle.

All inputs and outputs are fully compatible with Series 74, 74S or 74LS TTL. No pull-up resistors are required. This 4K Static RAM series is manufactured using TI's reliable N-channel silicon-gate technology to optimize the cost/performance relationship. All versions are characterized to retain data at V_{CC} = 2.4 V to reduce power dissipation.

PIN NAMES

A_0 - A_9	Addresses
DQ	Data In/Data Out
$\overline{S}$	Chip Select
V_{CC} (TMS 4045/L45)	+5 V Supply
V_{CC1} (TMS 4047/L47)	+5 V Supply (array only)
V_{CC2} (TMS 4047/L47)	+5 V Supply (periphery only)
V_{SS}	Ground
$\overline{W}$	Write Enable

PRELIMINARY DATA SHEET:
Supplementary data will be published at a later date.

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

Furthermore, for applications such as battery back-up, the TMS 4047 and TMS 40L47 have separate V_{CC} pins for the array and periphery, and data will be retained if power to the array alone is maintained.

The TMS 4045/40L45 series and the TMS 4047/40L47 series are offered respectively in 18-pin and 20-pin dual-in-line cerdip (JL suffix), sidebraze (JDL suffix), and plastic (NL suffix) packages designed for insertion in mounting-hole rows on 300-mil (7.62 mm) centers. The series is guaranteed for operation from 0°C to 70°C.

operation

addresses (A0-A11)
The ten address inputs select one of the 1024 4-bit words in the RAM. The address inputs must be stable for the duration of a write cycle. The address inputs can be driven directly from standard Series 54/74 TTL with no external pull-up resistors.

output enable ($\overline{OE}$)
The output enable terminal, which can be driven directly from standard TTL circuits, affects only the data-in/data-out terminals. When output enable is at a logic high level, the DQ terminals are disabled to the high-impedance state. Output enable provides greater output control flexibility, simplifying data bus design.

chip select ($\overline{S}$)
The chip-select terminal, which can be driven directly from standard TTL circuits, affects the data-in and data-out terminals. When chip select is at a logic low level, both terminals are enabled. When chip select is high, data-in is inhibited and data-out is in the floating or high-impedance state.

write enable ($\overline{W}$)
The read or write mode is selected through the write enable terminal. A logic high selects the read mode; a logic low selects the write mode. $\overline{W}$ or $\overline{S}$ must be high when changing addresses to prevent erroneously writing data into a memory location. The $\overline{W}$ input can be driven directly from standard TTL circuits.

data-in/data-out (DQ_1-DQ_4)
Data can be written into a selected device when the write enable input is low. The DQ terminal can be driven directly from standard TTL circuits. The three-state output buffer provides direct TTL compatibility with a fan-out of two Series 74 TTL gates, one Series 74S TTL gate, or eight Series 74LS TTL gates. The DQ terminals are in the high impedance state when chip select ($\overline{S}$) is high or whenever a write operation is being performed. Data-out is the same polarity as data-in.

standby operation

There are two basic standby modes available to retain data when operating the TMS 4045/40L45/4047/40L47 series:

1. Reduce the V_{CC} supply to 2.4 V
2. Supply power to the array only (TMS 4047 and TMS 40L47 only).

Combining modes 1 and 2 on the TMS 4047 or TMS 40L47 will produce the lowest possible standby power while retaining data.

DEVICE	SUPPLY	OPERATING	STANDBY	
TMS 4045, TMS 40L45	V_{CC}	+5 V	+2.4 V	
TMS 4047, TMS 40L47	V_{CC1}	+5 V	+5 V	+2.4 V
	V_{CC2}	+5 V	0 V	0 V

(nominal supply values)

During standby operation, data can not be read or written into the memory. When resuming normal operation, five cycle times must be allowed after normal supplies are returned for the memory to resume steady state operating conditions.

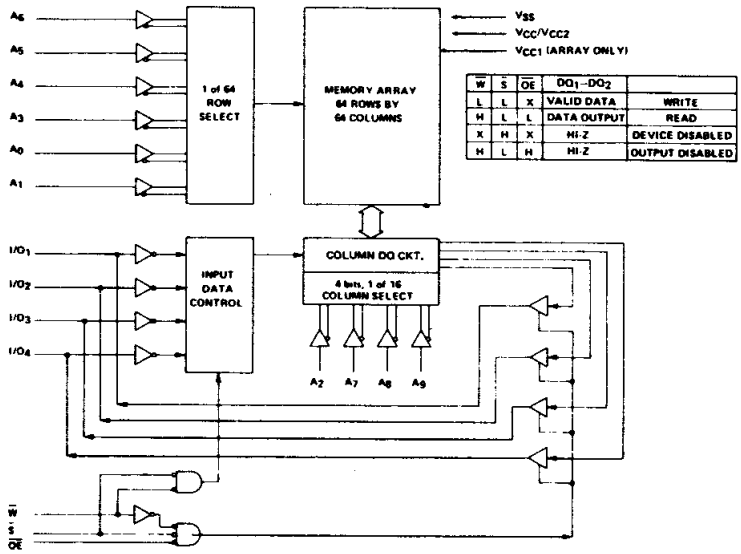
absolute maximum ratings over operating free-air temperature (unless otherwise noted)*

Supply voltage, V_{CC} (see Note 1) ..... −0.5 to 7 V
Input voltage (any input) (see note 1) ..... −1 to 7 V
Continuous power dissipation ..... 1 W
Operating free-air temperature range ..... 0°C to 70°C
Storage temperature range ..... −55°C to 150°C

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: Voltage values are with respect to the ground material.

functional block diagram



W	S	$\overline{OE}$	DO_1–DO_2	
L	L	X	VALID DATA	WRITE
H	L	L	DATA OUTPUT	READ
X	H	X	HI-Z	DEVICE DISABLED
H	L	H	HI-Z	OUTPUT DISABLED

recommended operating conditions

PARAMETER		MIN	NOM	MAX	UNIT
Supply voltage, V_{CC} (TMS 4045, TMS 40L45)	Operating	4.5	5	5.5	V
	Standby	2.4	5	5.5	
Supply voltage (array only), V_{CC1} (TMS 4047, TMS 40L47)	Operating	4.5	5	5.5	V
	Standby	2.4	5	5.5	
Supply voltage (periphery only), V_{CC2} (TMS 4047, TMS 40L47)	Operating	4.5	5	5.5	V
	Standby	0	0	.5.5	
Supply voltage, V_{SS}			0		V
High-level input voltage, V_{IH}		2.0		5.5	V
Low-level input voltage, V_{IL}		−1.0		0.8	V
Operating free-air temperature, T_A		0		70	°C

electrical characteristics over recommended operating free-air temperature ranges (unless otherwise noted)

PARAMETER		TEST CONDITIONS			MIN	TYP†	MAX	UNIT
V_{OH}	High level voltage	I_{OH} = −1.0 mA, V_{CC} = 4.5 V			2.4			V
V_{OL}	Low level voltage	I_{OL} = 3.2 mA, V_{CC} = 4.5 V					0.4	V
I_I	Input current	V_I = 0 V to 5.5 V					10	μA
I_{OZ}	Off-state output current	S at 2 V or W at 0.3 V, V_O = 0 V to 5.5 V			−10		10	μA
I_{CC}	Supply current from V_{CC}	I_O = 0 mA, T_A = 0°C (worst case)	TMS 4045	V_{CC} = 5.5 V		90	100	mA
				V_{CC} = 2.4 V		60	70	
			TMS 40L45	V_{CC} = 5.5 V		50	60	
				V_{CC} = 2.4 V		35	45	
I_{CC1}	Supply current from V_{CC1} (array only) (TMS 4047/L47 only)	I_O = 0 mA, T_A = 70°C (worst case)		V_{CC} = 5.5 V		5	11	mA
				V_{CC} = 2.4 V		2.5	5.5	
I_{CC2}	Supply current from V_{CC2} (periphery only) (TMS 4047/L47 only)	I_O = 0 mA, V_{CC} = 5.5 V, T_A = 0°C (worst case)	TMS 4047			90	99	mA
			TMS 40L47			50	59	
C_i	Input capacitance	V_I = 0 V, f = 1 MHz					8	pF
C_o	Output capacitance	V_O = 0 V, f = 1 MHz					8	pF

†All typical values are at V_{CC} = 5 V, T_A = 25°C.

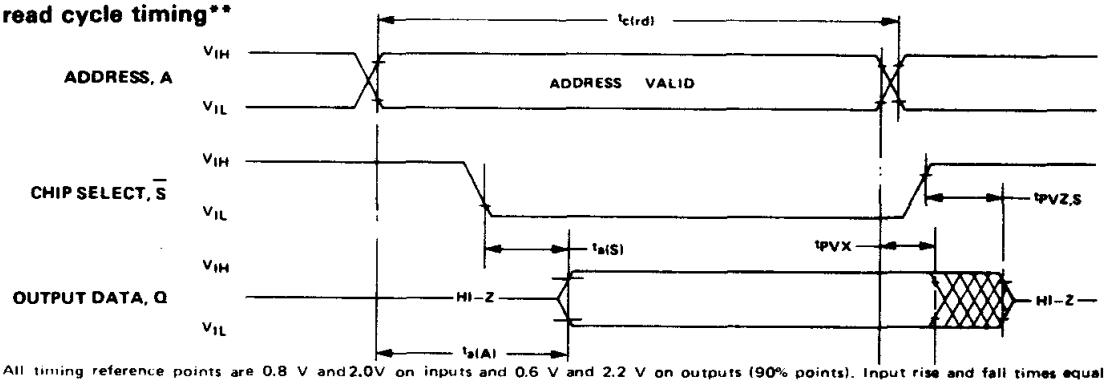
timing requirements over recommended supply voltage range, T_A = 0°C to 70°C 1 Series 74 TTL load C_L = 100 pF

PARAMETER		TMS 4045-15 TMS 4047-15		TMS 4045/L45-20 TMS 4047/L47-20		TMS 4045/L45-25 TMS 4047/L47-25		TMS 4045/L45-45 TMS 4047/L47-45		UNIT
		MIN	MAX	MIN	MAX	MIN	MAX	MIN	MAX	
$t_{c(rd)}$	Read cycle time	150		200		250		450		ns
$t_{c(wr)}$	Write cycle time	150		200		250		450		ns
$t_{w(W)}$	Write pulse width	80		100		100		200		ns
$t_{su(A)}$	Address set up time	0		0		0		0		ns
$t_{su(S)}$	Chip select set up time	80		100		100		200		ns
$t_{su(D)}$	Data set up time	80		100		100		200		ns
$t_{h(D)}$	Data hold time	0		0		0		0		ns
$t_{h(A)}$	Address hold time	0		0		0		0		ns

switching characteristics over recommended voltage range, t_A = 0°C to 70°C, 1 Series 74 TTL load, C_L = 100 pF

PARAMETER		TMS 4045-15 TMS 4047-15 MIN	TMS 4045-15 TMS 4047-15 MAX	TMS 4045/L45-20 TMS 4047/L47-20 MIN	TMS 4045/L45-20 TMS 4047/L47-20 MAX	TMS 4045/L45-25 TMS 4047/L47-25 MIN	TMS 4045/L45-25 TMS 4047/L47-25 MAX	TMS 4045/L45-45 TMS 4047-L47-45 MIN	TMS 4045/L45-45 TMS 4047-L47-45 MAX	UNIT
$t_{a(A)}$	Access time from address		150		200		250		450	ns
$t_{a(S)}$	Access time from chip select (or output enable) low		70		85		100		120	ns
$t_{a(W)}$	Access time from write enable high		70		85		100		120	ns
t_{PVX}	Output data valid after address change	20		20		20		20		ns
$t_{PVZ,S}$	Output disable time after chip select (or output enable) high		50		60		60		100	ns
$t_{PVZ,W}$	Output disable time after write enable low		50		60		60		100	ns

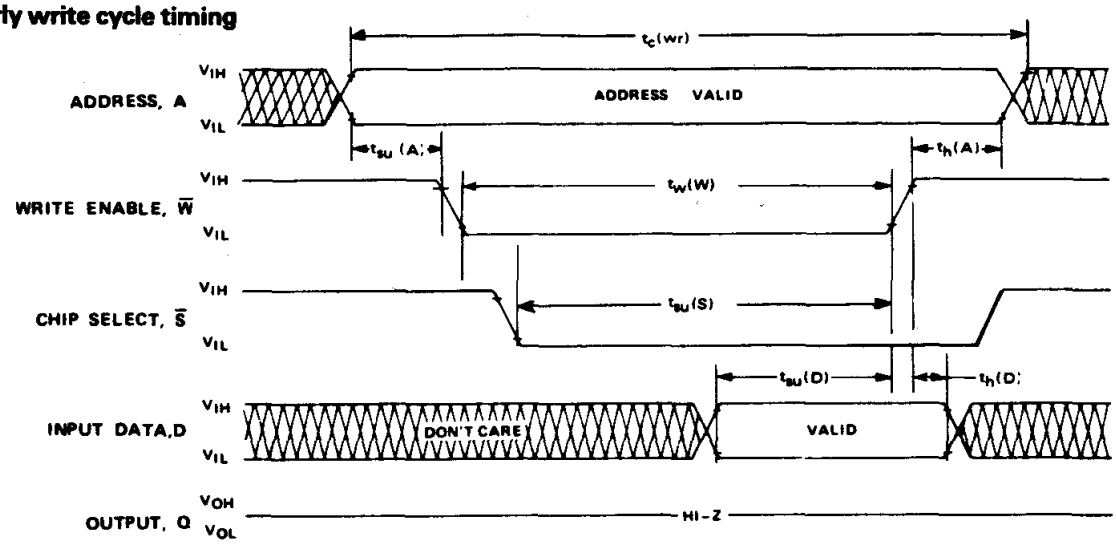
read cycle timing**



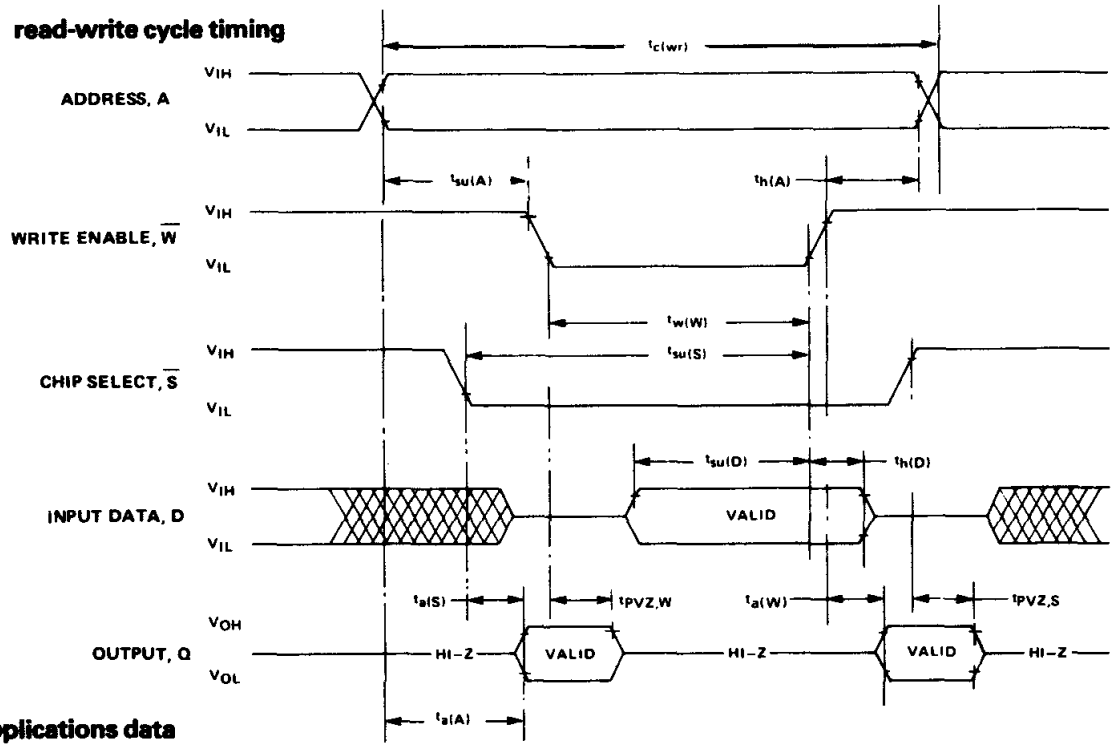
All timing reference points are 0.8 V and 2.0V on inputs and 0.6 V and 2.2 V on outputs (90% points). Input rise and fall times equal 10 nanoseconds.

**Write enable is high for a read cycle.

early write cycle timing



read-write cycle timing



applications data

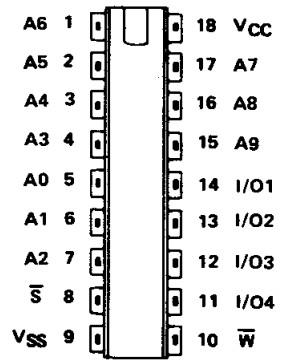
Early write cycle avoids DQ conflicts by controlling the write time with $\overline{S}$. On the diagram above, the write operation will be controlled by the leading edge of $\overline{S}$, not $\overline{W}$. Data can only be written when both $\overline{S}$ and $\overline{W}$ are low. Either $\overline{S}$ or $\overline{W}$ being high inhibits the write operation. To prevent erroneous data being written into the array, the addresses must be stable during the write cycle as defined by $t_{su(A)}$, $t_{w(W)}$, and $t_{h(A)}$.

TMS 4014 JDL, NL
1024 WORD BY 4-BIT STATIC RAM

FEBRUARY 1979

- 1024 X 4 Organization
- Single 5% Tolerance +5 V Supply
- High Density, 300-mil (7,62 mm), 18-Pin Package
- Fully Static Operation (No Clocks, no refresh, no timing strobe)
- Maximum Access Time . . . 450 ns
- Minimum Cycle (Read or Write) Time . . . 450 ns
- 400 mV Guaranteed dc Noise Immunity With Standard TTL Loads — No Pull-Up Resistors Required
- Common I/O With Three-State Outputs and Chip Select Control for OR-Tie Capability
- Fan-Out to 1 Series 74 or 74S TTL Load — No Pull-Up Resistors Required
- Low Power Dissipation
  - 330 mW Typical
  - 525 mW Maximum

TMS 4014
18-PIN CERAMIC AND PLASTIC
DUAL-IN-LINE PACKAGES
(TOP VIEW)



description

The TMS 4014 static random-access memory is organized as 1024 words of 4 bits. Static design results in reduced overhead costs by elimination of refresh-clocking circuitry and by simplification of timing requirements. Because this series is fully static, chip select may be tied low to further simplify system timing. Output data is always available during a read cycle.

All inputs and outputs are fully compatible with Series 74 or 74S TTL. No pull-up resistors are required. The TMS 4014 is manufactured using TI's reliable N-channel silicon-gate technology to optimize the cost/performance relationship. The TMS 4014 is characterized to retain data at V_{CC} = 2.4 V to reduce power dissipation. The TMS 4014 is offered in 18-pin dual-in-line sidebraze (JDL suffix) and plastic (NL suffix) packages designed for insertion in mounting-hole rows on 300-mil (7,62-mm) centers. The TMS 4014 is designed for operation from 0°C to 70°C.

PIN NAMES	
A0-A9	Addresses
I/O1-I/O4	Data input/output
$\overline{OE}$	Output Enable
$\overline{S}$	Chip Select
V_{CC}	+5 V Supply
V_{SS}	Ground
$\overline{W}$	Write Enable

PRELIMINARY DATA SHEET:
Supplementary data will be published at a later date.

operation

addresses (A0-A9)
The ten address inputs select one of the 1024 four-bit words stored in the RAM. The address-inputs must be stable for the duration of a write cycle. The address inputs can be driven directly from standard Series 54/74 TTL with no external pull-up resistors.

chip select ($\overline{S}$)
The chip-select terminal, which can be driven directly from standard TTL circuits, affects only the data-in/data-out terminals. When chip select and output enable are a logic low, the I/O terminals are enabled. When chip select is a logic high, the I/O terminals are in the floating or high-impedance state and the input is inhibited.

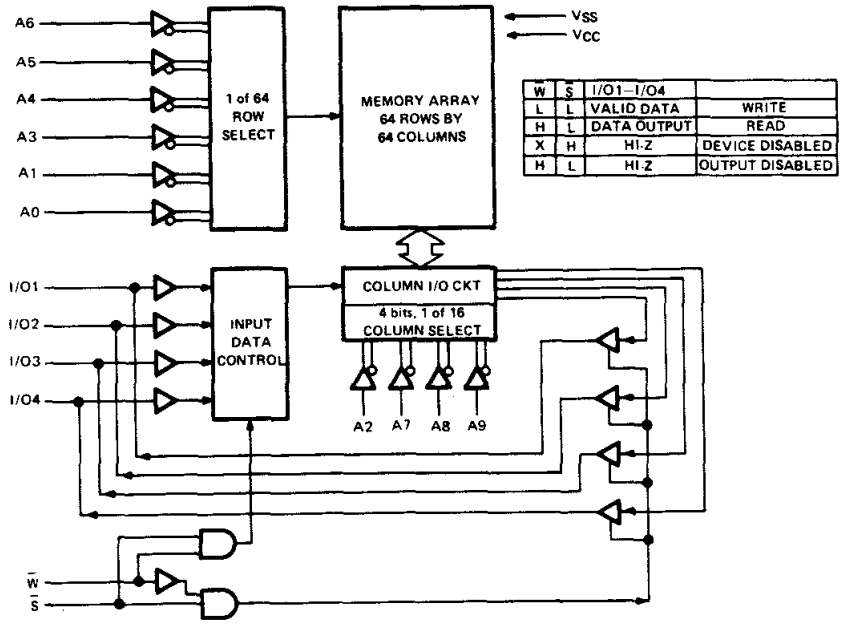
write enable ($\overline{W}$)
The read or write mode is selected through the write enable terminal. A logic high selects the read mode; a logic low selects the write mode. W must be a logic high when changing addresses to prevent erroneously writing data into a memory location. The W input can be driven directly from standard TTL circuits.

data-in/data-out (I/O1-I/O4)
Data can be written into a selected device when the write enable input is a logic low. The I/O terminal can be driven directly from standard TTL circuits. The three-state output buffer provides direct TTL compatibility with a fan-out of one Series 74 TTL gate or one Series 74S TTL gate. The I/O terminals are in the high-impedance state when chip select (S) is high or whenever a write operation is being performed. Data-out is the same polarity as data-in.

standby operation

The standby mode, available to retain data while operating the TMS 4014, is attained by reducing V_{CC} from 5 V to 2.4 V. Before, during, and immediately after standby, $\overline{S}$ and $\overline{W}$ must be in the high state. During standby operation, data cannot be read or written into the memory. When resuming normal operation, five cycle times must be allowed after normal power is restored for the memory to resume steady-state operating conditions.

functional block diagram



W̄	S̄	I/O1–I/O4	
L	L	VALID DATA	WRITE
H	L	DATA OUTPUT	READ
X	H	HI-Z	DEVICE DISABLED
H	L	HI-Z	OUTPUT DISABLED

absolute maximum ratings over operating free-air temperature range (unless otherwise noted)*

Supply voltage, V_{CC} (see Note 1) . . . . . −0.5 to 7 V
Input voltage (any input) (see Note 1) . . . . . −0.5 to 7 V
Continuous power dissipation . . . . . 1 W
Operating free-air temperature range . . . . . 0°C to 70°C
Storage temperature range . . . . . −55°C to 150°C

NOTE 1: Voltage values are with respect to the ground terminal.

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

recommended operating conditions

PARAMETER		MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}	Operating	4.75	5	5.25	V
	Standby	2.4	5	5.25	
Supply voltage, V_{SS}			0		V
High-level input voltage, V_{IH}		2.0		5.25	V
Low-level input voltage, V_{IL}		−0.3		0.8	V
Operating free-air temperature, T_A		0		70	°C

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER		TEST CONDITIONS		MIN	TYP†	MAX	UNIT
V_{OH}	High-level voltage	I_{OH} = −200 μA,	V_{CC} = 4.75 V	2.4			V
V_{OL}	Low-level voltage	I_{OL} = 2 mA,	V_{CC} = 4.75 V			0.4	V
I_I	Input current	V_I = 0 to 5.25 V				10	μA
I_{OZ}	Off-state output current	$\overline{S}$ @ 2 V or $\overline{W}$ at 0.8 V,	V_O = 0 to 5.25 V			±10	μA
I_{CC}	Supply current from V_{CC}	I_O = 0 mA, T_A = 0°C (worst case)	V_{CC} = 5.25 V		90	100	mA
			V_{CC} = 2.4 V		60	70	mA
C_i	Input capacitance	V_I = 0 V,	f = 1 MHz			8	pF
C_o	Output capacitance	V_O = 0 V,	f = 1 MHz			12	pF

†All typical values are at V_{CC} = 5 V, T_A = 25°C.

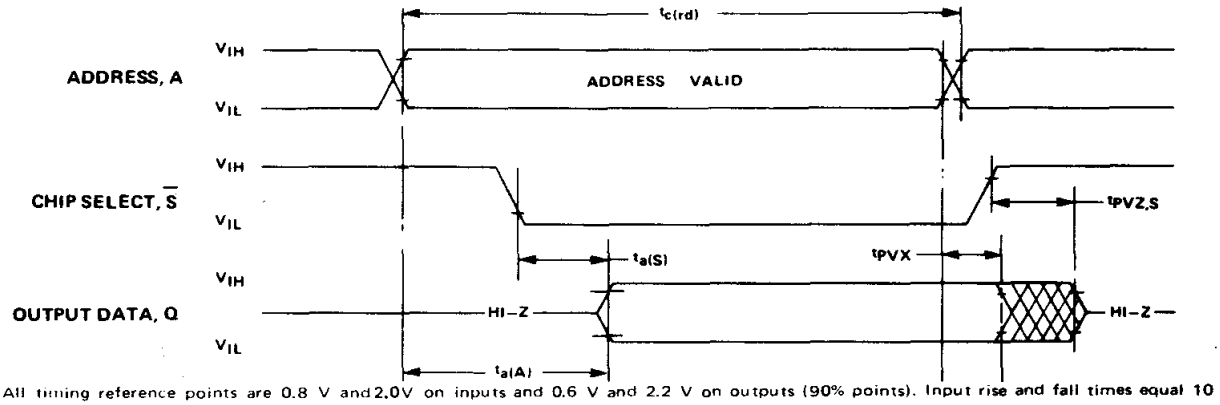
timing requirements over recommended supply voltage range and operating free-air temperature range

	PARAMETER	MIN	MAX	UNIT
$t_{c(rd)}$	Read cycle time	450		ns
$t_{c(wr)}$	Write cycle time	450		ns
$t_{w(W)}$	Write pulse width	200		ns
$t_{su(A)}$	Address set up time	0		ns
$t_{su(S)}$	Chip select set up time	200		ns
$t_{su(D)}$	Data set up time	200		ns
$t_{h(D)}$	Data hold time	0		ns
$t_{h(A)}$	Address hold time	20		ns
$t_{T(A)}$	Address transition time	5	200	ns

switching characteristics over recommended voltage range, T_A = 0°C to 70°C, 1 series 74 TTL load, C_L = 100 pF

	PARAMETER	MIN	NOM	MAX	UNIT
$t_{a(A)}$	Access time from address			450	ns
$t_{a(S)}$	Access time from chip select (or output enable) low			120	ns
$t_{a(W)}$	Access time from write enable high			120	ns
t_{PVX}	Output data valid after address change	10			ns
$t_{PVZ,S}$	Output disable time after chip select (or output enable) high			100	ns
$t_{PVZ,W}$	Output disable time after write enable high			100	ns

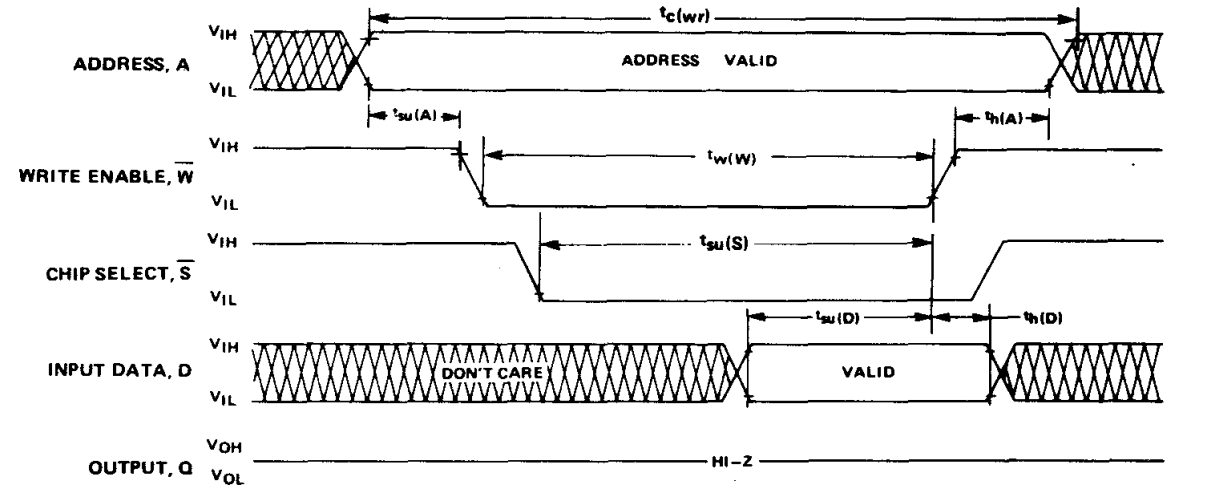
read cycle timing**



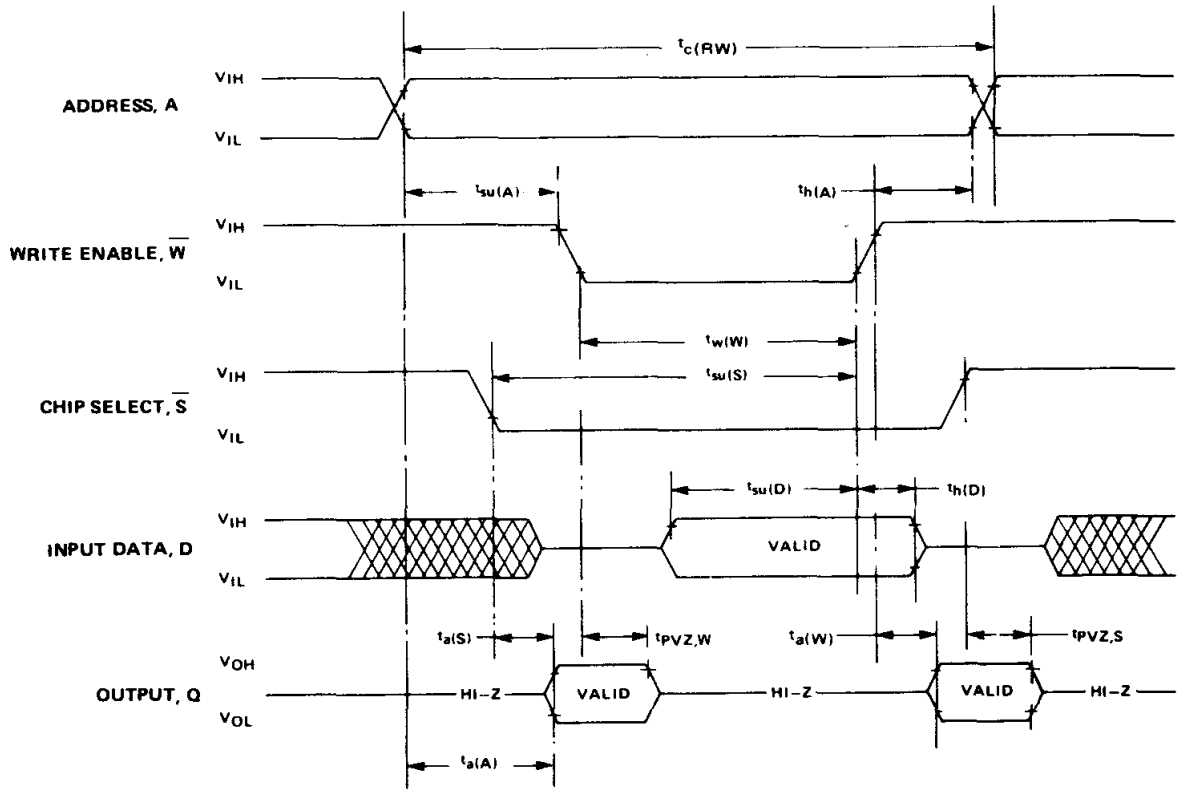
All timing reference points are 0.8 V and 2.0 V on inputs and 0.6 V and 2.2 V on outputs (90% points). Input rise and fall times equal 10 nanoseconds.

**Write enable is high for a read cycle.

early write cycle timing



read-write cycle timing



applications data

Early write cycle avoids I/O conflicts by controlling the write time with $\overline{S}$. In the diagram above the write operation will be controlled by the leading edge of $\overline{S}$, not $\overline{W}$. Data can only be written when both $\overline{S}$ and $\overline{W}$ are logic low. Either $\overline{S}$ or $\overline{W}$ being logic high inhibits the write operation, and data stored will not be affected by the address. To prevent erroneous data from being written into the array, the addresses must be stable during the write cycle as defined by $t_{su(A)}$, $t_{w(W)}$, and $t_{h(A)}$.

TEXAS INSTRUMENTS
INCORPORATED
POST OFFICE BOX 225012 • DALLAS, TEXAS 75265

APPENDIX C

ASCII CODE

TABLE C-1. *ASCII CONTROL CODES

CONTROL	BINARY CODE	HEXADECIMAL CODE
NUL – Null	000 0000	00
SOH – Start of heading	000 0001	01
STX – Start of text	000 0010	02
ETX – End of text	000 0011	03
EOT – End of transmission	000 0100	04
ENQ – Enquiry	000 0101	05
ACK – Acknowledge	000 0110	06
BEL – Bell	000 0111	07
BS – Backspace	000 1000	08
HT – Horizontal tabulation	000 1001	09
LF – Line feed	000 1010	0A
VT – Vertical tab	000 1011	0B
FF – Form feed	000 1100	0C
CR – Carriage return	000 1101	0D
SO – Shift out	000 1110	0E
SI – Shift in	000 1111	0F
DLE – Data link escape	001 0000	10
DC1 – Device control 1	001 0001	11
DC2 – Device control 2	001 0010	12
DC3 – Device control 3	001 0011	13
DC4 – Device control 4 (stop)	001 0100	14
NAK – Negative acknowledge	001 0101	15
SYN – Synchronous idle	001 0110	16
ETB – End of transmission block	001 0111	17
CAN – Cancel	001 1000	18
EM – End of medium	001 1001	19
SUB – Substitute	001 1010	1A
ESC – Escape	001 1011	1B
FS – File separator	001 1100	1C
GS – Group separator	001 1101	1D
RS – Record separator	001 1110	1E
US – Unit separator	001 1111	1F
DEL – Delete, rubout	111 1111	7F

*American Standards Institute Publication X3.4-1968

NOTE

Hexadecimal codes 01 to 1F can be generated using most keyboard devices with the CONTROL (SHIFT) key pressed prior to pressing another keyboard key. For example, hexadecimal codes 01 to 19 can be generated on the TM 990/189 using the SHIFT key and keys A through Y respectively with the exception of keys V and X which have shift functions dedicated to display right and cancel respectively.

TABLE C-2. *ASCII CHARACTER CODE

CHARACTER	BINARY CODE	HEXADECIMAL CODE	CHARACTER	BINARY CODE	HEXADECIMAL CODE
Space	010 0000	20	P	101 0000	50
!	010 0001	21	Q	101 0001	51
" (dbl. quote)	010 0010	22	R	101 0010	52
#	010 0011	23	S	101 0011	53
$	010 0100	24	T	101 0100	54
%	010 0101	25	U	101 0101	55
&	010 0110	26	V	101 0110	56
' (sgl. quote)	010 0111	27	W	101 0111	57
(	010 1000	28	X	101 1000	58
)	010 1001	29	Y	101 1001	59
* (asterisk)	010 1010	2A	Z	101 1010	5A
+	010 1011	2B	[	101 1011	5B
, (comma)	010 1100	2C	\	101 1100	5C
- (minus)	010 1101	2D	]	101 1101	5D
. (period)	010 1110	2E	^	101 1110	5E
/	010 1111	2F	_ (underline)	101 1111	5F
0	011 0000	30	`	110 0000	60
1	011 0001	31	a	110 0001	61
2	011 0010	32	b	110 0010	62
3	011 0011	33	c	110 0011	63
4	011 0100	34	d	110 0100	64
5	011 0101	35	e	110 0101	65
6	011 0110	36	f	110 0110	66
7	011 0111	37	g	110 0111	67
8	011 1000	38	h	110 1000	68
9	011 1001	39	i	110 1001	69
:	011 1010	3A	j	110 1010	6A
;	011 1011	3B	k	110 1011	6B
<	011 1100	3C	l	110 1100	6C
=	011 1101	3D	m	110 1101	6D
>	011 1110	3E	n	110 1110	6E
?	011 1111	3F	o	110 1111	6F
@	100 0000	40	p	111 0000	70
A	100 0001	41	q	111 0001	71
B	100 0010	42	r	111 0010	72
C	100 0011	43	s	111 0011	73
D	100 0100	44	t	111 0100	74
E	100 0101	45	u	111 0101	75
F	100 0110	46	v	111 0110	76
G	100 0111	47	w	111 0111	77
H	100 1000	48	x	111 1000	78
I	100 1001	49	y	111 1001	79
J	100 1010	4A	z	111 1010	7A
K	100 1011	4B	{	111 1011	7B
L	100 1100	4C	|	111 1100	7C
M	100 1101	4D	}	111 1101	7D
N	100 1110	4E	~	111 1110	7E
O	100 1111	4F			

*American Standards Institute Publication X3.4-1968

APPENDIX D

BINARY, DECIMAL AND HEXADECIMAL NUMBERING

D-1 GENERAL

This appendix covers numbering systems to three bases (2, 10, and 16) which are used throughout this manual.

D-2 POSITIVE NUMBERS

D-2.1 DECIMAL (BASE 10). When a numerical quantity is viewed from right to left, the right-most digit represents the base number to the exponent 0. The next digit represents the base number to the exponent 1, the next to the exponent 2, then exponent 3, etc. For example, using the base 10 (decimal):

$$10^6\ \ 10^5\ \ 10^4\ \ 10^3\ \ 10^2\ \ 10^1\ \ 10^0$$
$$X,\quad X\quad X\ \ X,\quad X\quad X\quad X$$

or

1,000,000
100,000
10,000
1000 100 10 1
X , X X X , X X X

For example, 75,264 can be broken down as follows:

75, 264

$4 \times 10^0 = 4 \times 1 = 4$

$6 \times 10^1 = 6 \times 10 = 60$

$2 \times 10^2 = 2 \times 100 = 200$

$5 \times 10^3 = 5 \times 1000 = 5000$

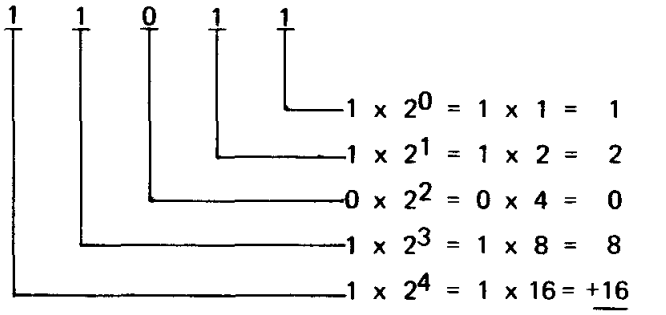
$7 \times 10^4 = 7 \times 10{,}000 = +70000$

75264_{10}

D-2.2 BINARY (BASE 2). As base 10 numbers use ten digits, base 2 numbers use only 0 and 1. When viewed from right to left, they each represent the number 2 to the powers 0, 1, 2, etc., respectively as shown below:

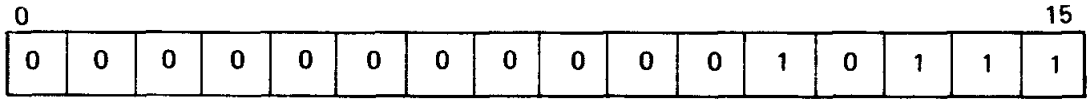
2^{15}		2^6	2^5	2^4	2^3	2^2	2^1	2^0
(32,768)	• • •	(64)	(32)	(16)	(8)	(4)	(2)	(1)
X	• • •	X	X	X	X	X	X	X

For example, 11011_2 can be translated into base 10 as follows:



or 11011_2 equals 27_{10}.

Binary is the language of the digital computer. For example, to place the decimal quantity 23 (23_{10}) into a 16-bit memory cell, set the bits to the following:



which is $1 + 2 + 4 + 16 = 23_{10}$.

D-2.3 HEXADECIMAL (BASE 16). Whereas binary uses two digits and decimal uses ten digits, hexadecimal uses 16 (0 to 9, A, B, C, D, E, and F).

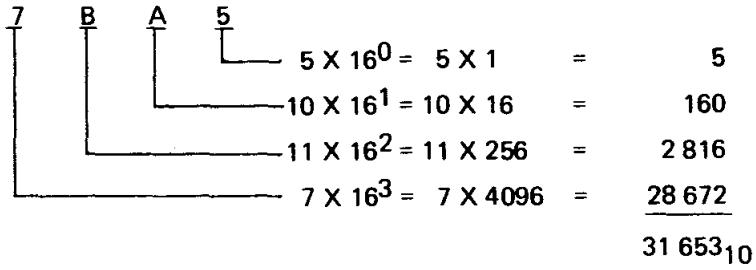
The letters A through F are used to represent the decimal numbers 10 through 15 as shown on the following page.

N_{10}	N_{16}	N_{10}	N_{16}
0	0	8	8
1	1	9	9
2	2	10	A
3	3	11	B
4	4	12	C
5	5	13	D
6	6	14	E
7	7	15	F

When viewed from right to left, each digit in a hexadecimal number is a multiplier of 16 to the powers 0, 1, 2, 3, etc., as shown below:

16^3	16^2	16^1	16^0
(4096)	(256)	(16)	(1)
X	X	X	X

For example, 7 B A 5_{16} can be translated into base 10 as follows:



or 7 B A 5_{16} equals $31{,}653_{10}$.

Because it would be awkward to write out 16-digit binary numbers to show the contents of a 16-bit memory word, hexadecimal is used instead. Thus

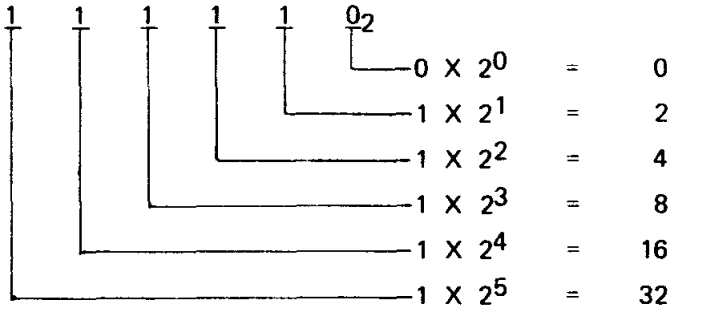
$003E_{16}$ or > 003E (> indicates hexadecimal)

is used instead of

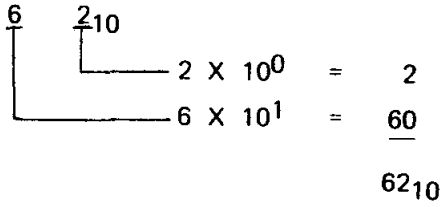
$0000\ 0000\ 0011\ 1110_2$

to represent 62_{10} as computed below:

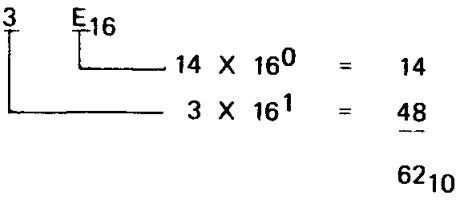
BASE 2



BASE 10



BASE 16



Note that separating the 16 binary bits into four-bit parts facilitates recognition and translation into hexadecimal.

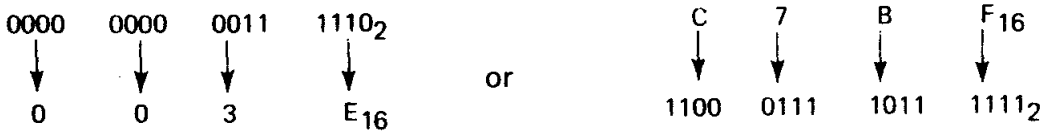


Table D-1 is a conversion chart for converting decimal to hexadecimal and vice versa. Table D-2 shows binary, decimal and hexadecimal equivalents for numbers 0 to 15. Note that Table D-1 is divided into four parts, each part representing four of the 16-bits of a memory cell or word (bits 0 to 15) with bit 0 being the most significant bit (MSB) and bit 15 being the least significant bit (LSB). Note that the MSB is on the left and represents the highest power of 2 and the LSB on the right represents the 0 power of 2 ($2^0 = 1$). As explained later, the MSB can also be used to signify number polarity (+ or −).

NOTE

To convert a binary number to decimal or hexadecimal, convert the *positive* binary value as described in Section D-4.

TABLE D-1 HEXADECIMAL/DECIMAL CONVERSION CHART

	MSB												LSB			
	16^3				16^2				16^1				16^0			
BITS	0	1	2	3	4	5	6	7	8	7	8	11	12	13	14	15
	HEX		DEC		HEX		DEC		HEX		DEC		HEX		DEC	
	0		0		0		0		0		0		0		0	
	1		4 096		1		256		1		16		1		1	
	2		8 192		2		512		2		32		2		2	
	3		12 288		3		768		3		48		3		3	
	4		16 384		4		1 024		4		64		4		4	
	5		20 480		5		1 280		5		80		5		5	
	6		24 576		6		1 536		6		96		6		6	
	7		28 672		7		1 792		7		112		7		7	
	8		32 768		8		2 048		8		128		8		8	
	9		36 864		9		2 304		9		144		9		9	
	A		40 960		A		2 560		A		160		A		10	
	B		45 056		B		2 816		B		176		B		11	
	C		49 152		C		3 072		C		192		C		12	
	D		53 248		D		3 328		D		208		D		13	
	E		57 344		E		3 584		E		224		E		14	
	F		61 440		F		3 840		F		240		F		15	

To convert a number from hexadecimal, add the decimal equivalents for each hexadecimal digit. For example, $7A82_{16}$ would equal in decimal 28,672 + 2,560 + 128 + 2. To convert hexadecimal to decimal, find the nearest decimal number in the above table less than or equal to the number being converted. Set down the hexadecimal equivalent then subtract this number from the nearest decimal number. Using the remainder(s), repeat this process. For example:

$31{,}362_{10} = 7000_{16} + 2690_{10}$
$2{,}690_{10} = A00_{16} + 130_{10}$
$130_{10} = 80_{16} + 2_{10}$
$2_{10} = 2_{16}$

7000
A00
80
2
$7A82_{16}$

TABLE D-2. BINARY, DECIMAL, AND HEXADECIMAL EQUIVALENTS

BINARY (N_2)	DECIMAL (N_{10})	HEXADECIMAL (N_{16})
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	10	A
1011	11	B
1100	12	C
1101	13	D
1110	14	E
1111	15	F
10000	16	10
10001	17	11
10010	18	12
10011	19	13
10100	20	14
10101	21	15
10110	22	16
10111	23	17
11000	24	18
11001	25	19
11010	26	1A
11011	27	1B
11100	28	1C
11101	29	1D
11110	30	1E
11111	31	1F
100000	32	20

D-3 ADDING AND SUBTRACTING BINARY

Adding and subtracting in binary uses the same conventions for decimal: carrying over in addition and borrowing in subtraction.

Basically,

```
  0       1                                          10
+ 1     + 1                                         - 1
---     ---                                         ---
  1      10   (the carry, 1, is carried to the left)  01   (1 is borrowed from
                                                             top left)
```

```
   1 }                                                    11
   1 }  = 0 + carry 1                                      1
 + 1    = 0 (from above) + 1 = 1                         + 1
 ---                                                     ---
  11                                                     101
  \__carry                          carry 1 + 1 = 10 ___/
```

```
   1 }                         1000 }                  {  1
   1 }  = 0 + 1 carry           - 1 }  Borrow the 1    { 0110
   1 }                         ---- }                  {  - 1
 + 1 }  = 0 + 1 carry          0111 }                  { ----
 ---                                                   { 0111
 100
 | |__0 + 0 = 0
 |____carry 1 + carry 1
```

D-4 **POSITIVE/NEGATIVE CONVERSION (BINARY).** To compute the negative equivalent of a positive binary or hexadecimal number, or interpret a binary or hexadecimal negative number (determine its positive equivalent) use the two's complement of the binary number.

NOTE

To convert a binary number to decimal, convert the *positive* binary value (*not* the negative binary value) and add the sign.

Two's complementing a binary number includes two simple steps:

a. Obtain one's complement of the number (1's become 0's, 0's becomes 1's) (invert bits).

b. Add 1 to the one's complement.

For example, with the MSB (left-most bit) being a sign bit:

010	$(+2_2)$	111	(-1_2)	110	(-2_2)	101	(-3_2)
101	Invert	000	Invert	001	Invert	010	Invert
+ 1	Add 1	+ 1	Add 1	+ 1	Add 1	+ 1	
110	(-2_2)	001	$(+1_2)$	010	$(+2_2)$	011	$(+3_2)$

This can be expanded to 16-bit positive numbers:

$(=39F6_{16})$	0011	1001	1111	0110	$(39F6_{16}$	=	$+14{,}838_{10})$	
	1100	0110	0000	1001	Invert			
				+1	Add 1			
$(=C60A_{16})$	1100	0110	0000	1010	$(C60A_{16}$	=	$-14{,}838_{10})$	Two's Complement

SIGN BIT(−)

And to 16-bit negative numbers:

$(=C60A_{16})$	1100	0110	0000	1010	$(C60A_{16}$	=	$-14{,}838_{10})$	
	0011	1001	1111	0101	Invert			
				+1	Add 1			
$(=39F6_{16})$	0011	1001	1111	0110	$(39F6_{16}$	=	$+14{,}838_{10})$	Two's Complement

SIGN BIT(+)

APPENDIX E

The Engineering Staff of
TEXAS INSTRUMENTS INCORPORATED
Semiconductor Group

TMS 9901 PROGRAMMABLE SYSTEMS INTERFACE

JULY 1978

TEXAS INSTRUMENTS
INCORPORATED

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

1.1 DESCRIPTION

The TMS 9901 Programmable Systems Interface (PSI) is a multifunctional component designed to provide low cost interrupt and I/O ports and an interval timer for TMS 9900-family microprocessor systems. The TMS 9901 is fabricated using N-channel silicon-gate MOS technology. The TMS 9901 is TTL-compatible on all inputs and outputs, including the power supply (+5 V) and single-phase clock.

1.2 KEY FEATURES

- Low Cost
- 9900-Family Peripheral
- Performs Interrupt and I/O Interface functions:
  - — Six Dedicated Interrupt Lines
  - — Seven Dedicated I/O Lines
  - — Nine Programmable Lines as I/O or Interrupt
  - — Up to 15 Interrupt Lines
  - — Up to 22 Input Lines
  - — Up to 16 Output Lines
- Easily Cascaded for Expansion
- Interval or Event Timer
- Single 5 V Power Supply
- All Inputs and Outputs TTL-Compatible
- Standard 40-Pin Plastic or Ceramic Package
- N-Channel Silicon-Gate MOS Technology.

1.3 APPLICATION OVERVIEW

The following example of a typical application may help introduce the user to the TMS 9901 PSI. Figure 1 is a block diagram of a typical application. Each of the ideas presented below is described in more detail in later sections of this manual.

The TMS 9901 PSI interfaces to the CPU through the Communications Register Unit (CRU) and the interrupt control lines as shown in Figure 1. The TMS 9901 occupies 32 bits of CRU input and output space. The five least significant bits of address bus are connected to the S lines of the PSI to address one of the 32 CRU bits of the TMS 9901. The most significant bits of the address bus are decoded on CRU cycles to select the PSI by taking its chip enable ($\overline{CE}$) line active (LOW).

Interrupt inputs to the TMS 9901 PSI are synchronized with $\bar{\phi}$, inverted, and then ANDed with the appropriate mask bit. Once every $\bar{\phi}$ clock time, the prioritizer looks at the 15 interrupt input AND gates and generates the interrupt control code. The interrupt control code and the interrupt request line constitute the interrupt interface to the CPU.

After reset all I/O ports are programmed as inputs. By writing to any I/O port, that port will be programmed as an output port until another reset occurs, either software or hardware. Data at the input pins is buffered on to the TMS 9901. Data to the output ports is latched and then buffered off-chip by the PSI's MOS-to-TTL buffers.

The interval timer on the TMS 9901 is accessed by writing a ONE to select bit zero (control bit), which puts the PSI CRU interface in the clock mode. Once in the clock mode the 14-bit clock contents can be read or written. Writing to the clock register will reinitialize the clock and cause it to start decrementing,. When the clock counts to zero, it will cause an interrupt and reload to its initial value. Reading the clock contents permits the user to see the decrementer contents at that point in time just before entering the clock mode. The clock read register is not updated when the PSI is in the clock mode.

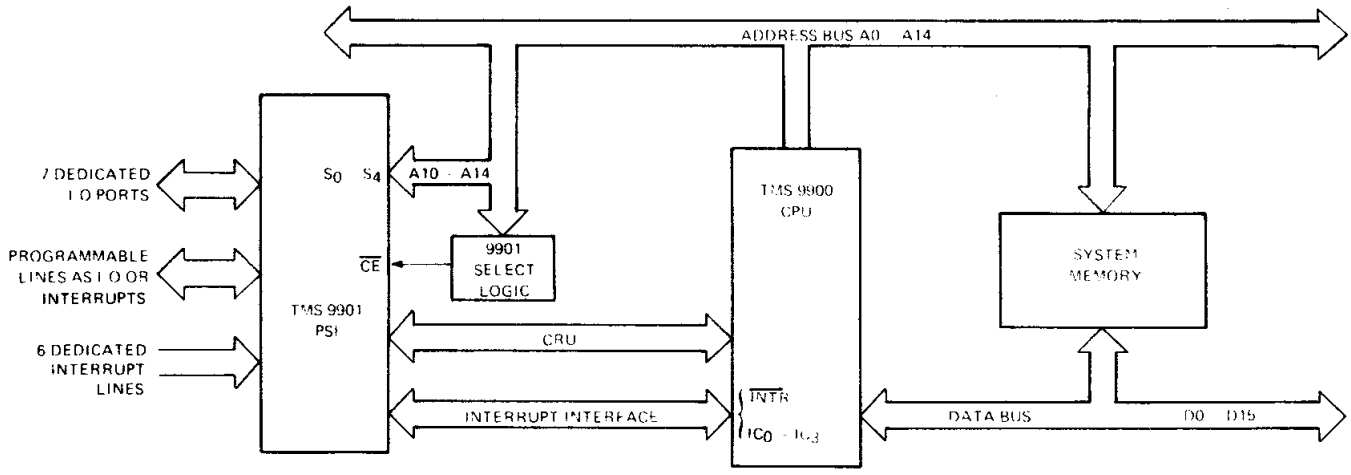


FIGURE 1– TYPICAL TMS 9901 PROGRAMMABLE SYSTEM INTERFACE (PSI) APPLICATION

2. ARCHITECTURE

The architecture of the TMS 9901 Programmable Systems Interface (PSI) is designed to provide the user maximum flexibility when designating system I/O ports and interrupts. The TMS 9901 can be divided into four subsystems: CRU interface, interrupt interface, input/output interface, and interval timer. Figure 2 is a general block diagram of the TMS 9901 internal architecture. Each of the subsystems of the PSI is discussed in detail in subsequent paragraphs.

2.1 CRU Interface

The CPU communicates with the TMS 9901 PSI via the CRU. The TMS 9901 occupies 32 bits in CRU read space and 32 bits in CRU write space. Table 1 shows the mapping for CRU bit addresses to TMS 9901 functions.

The CRU interface consists of five address select lines (S0-S4), chip enable ($\overline{CE}$), and the three CRU lines (CRUIN, CRUOUT, CRUCLK). The select lines (S0-S4) are connected to the five least significant bits of the address bus; for a TMS 9900 system S0-S4 are connected to A10-A14, respectively. Chip enable ($\overline{CE}$) is generated by decoding the most significant bits of the address bus on CRU cycles; for a 9900 based system address bits 0-9 would be decoded. When $\overline{CE}$ goes active (LOW), the five select lines point to the CRU bit being accessed. When $\overline{CE}$ is inactive (HIGH), the PSI's CRU interface is disabled.

NOTE

When $\overline{CE}$ is inactive (HIGH) the 9901 sets its CRUIN pin to high impedance and disables CRUCLK from coming on chip. This means that CRUIN can be used as an OR tied bus. When $\overline{CE}$ is high the 9901 will still see the select lines, but no command action is taken.

In the case of a write operation, the TMS 9901 strobes data off the CRUOUT line with CRUCLK. For a read operation, the data is sent to the CPU on the CRUIN line.

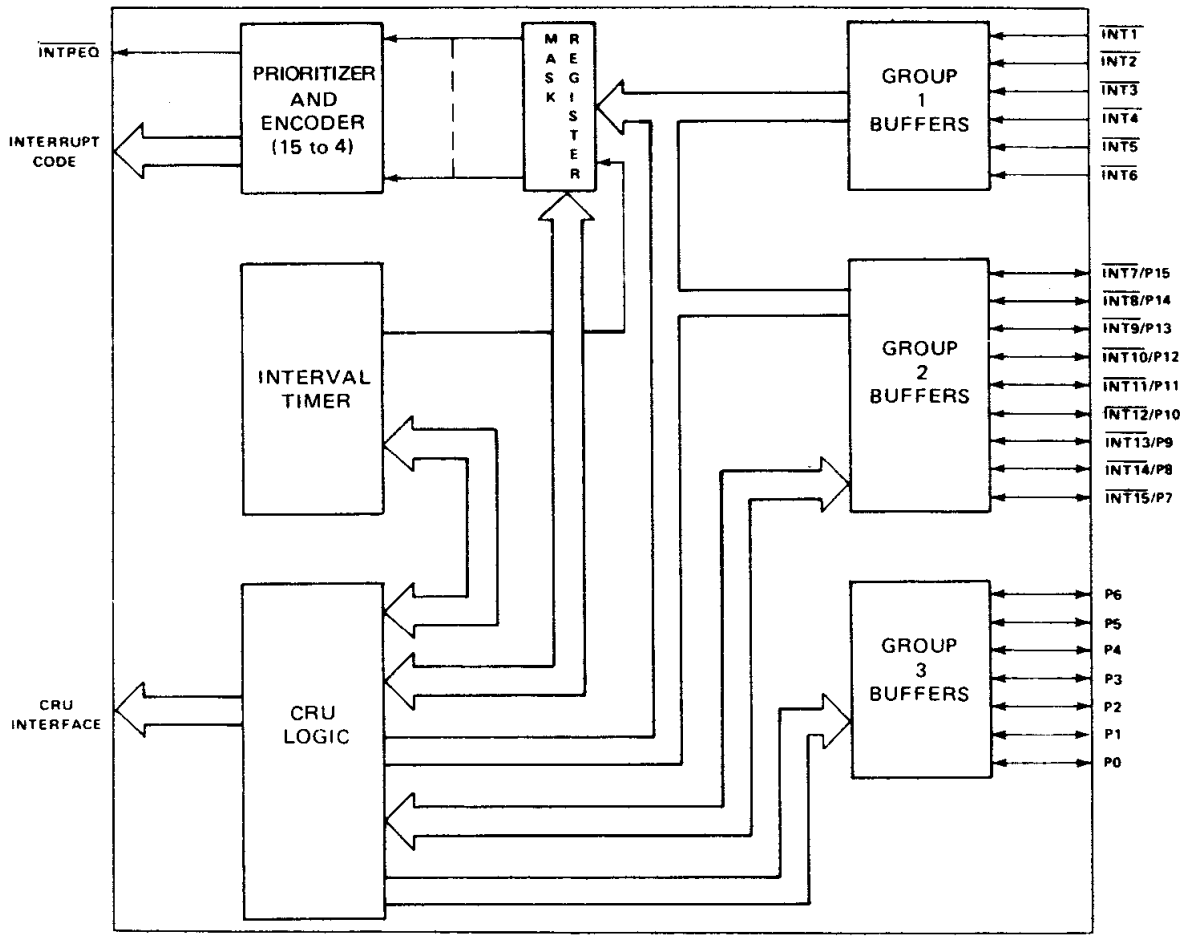


FIGURE 2–TMS 9901 PSI BLOCK DIAGRAM

Several TMS 9901 devices may be cascaded to expand I/O and interrupt handling capability simply by connecting all CRU and address select lines in parallel and providing each device with a unique chip enable signal: the chip enable ($\overline{CE}$) is generated by decoding the high-order address bits (A0-A9) on CRU cycles.

For those unfamiliar with the CRU concept, the following is a discussion of how to build a CRU interface. The CRU is a bit addressable (4096 bits), synchronous, serial interface over which a single instruction can transfer between one and 16 bits serially. Each one of the 4096 bits of the CRU space has a unique address and can be read and written to. During multi-bit CRU transfers, the CRU address is incremented at the beginning of each CRU cycle to point to the next consecutive CRU bit.

TABLE 1

CRU SELECT BIT ASSIGNMENTS

CRU Bit	S_0 S_1 S_2 S_3 S_4	CRU Read Data	CRU Write Data
0	0 0 0 0 0	CONTROL BIT(1)	CONTROL BIT(1)
1	0 0 0 0 1	$\overline{INT}$1/CLK1(2)	Mask 1/CLK1(3)
2	0 0 0 1 0	$\overline{INT}$2/CLK2	Mask 2/CLK2
3	0 0 0 1 1	$\overline{INT}$3/CLK3	Mask 3/CLK3
4	0 0 1 0 0	$\overline{INT}$4/CLK4	Mask 4/CLK4
5	0 0 1 0 1	$\overline{INT}$5/CLK5	Mask 5/CLK5
6	0 0 1 1 0	$\overline{INT}$6/CLK6	Mask 6/CLK6
7	0 0 1 1 1	$\overline{INT}$7/CLK7	Mask 7/CLK7
8	0 1 0 0 0	$\overline{INT}$8/CLK8	Mask 8/CLK8
9	0 1 0 0 1	$\overline{INT}$9/CLK9	Mask 9/CLK9
10	0 1 0 1 0	$\overline{INT}$10/CLK10	Mask 10/CLK10
11	0 1 0 1 1	$\overline{INT}$11/CLK11	Mask 11/CLK11
12	0 1 1 0 0	$\overline{INT}$12/CLK12	Mask 12/CLK12
13	0 1 1 0 1	$\overline{INT}$13/CLK13	Mask 13/CLK13
14	0 1 1 1 0	$\overline{INT}$14/CLK14	Mask 14/CLK14
15	0 1 1 1 1	$\overline{INT}$15/INTREQ(7)	Mask 15/$\overline{RST2}$(4)
16	1 0 0 0 0	PO Input(5)	PO Output(6)
17	1 0 0 0 1	P1 Input	P1 Output
18	1 0 0 1 0	P2 Input	P2 Output
19	1 0 0 1 1	P3 Input	P3 Output
20	1 0 1 0 0	P4 Input	P4 Output
21	1 0 1 0 1	P5 Input	P5 Output
22	1 0 1 1 0	P6 Input	P6 Output
23	1 0 1 1 1	P7 Input	P7 Output
24	1 1 0 0 0	P8 Input	P8 Output
25	1 1 0 0 1	P9 Input	P9 Output
26	1 1 0 1 0	P10 Input	P10 Output
27	1 1 0 1 1	P11 Input	P11 Output
28	1 1 1 0 0	P12 Input	P12 Output
29	1 1 1 0 1	P13 Input	P13 Output
30	1 1 1 1 0	P14 Input	P14 Output
31	1 1 1 1 1	P15 Input	P15 Output

NOTES:

(1) 0 = Interrupt Mode 1 = Clock Mode

(2) Data present on $\overline{INT}$ input pin (or clock value) will be read regardless of mask value.

(3) While in the Interrupt Mode (Control Bit = 0) writing a "1" into mask will enable interrupt; a "0" will disable.

(4) Writing a zero to bit 15 while in the clock mode (Control Bit = 1) executes a software reset of the I/O pins.

(5) Data present on the pin will be read. Output data can be read without affecting the data.

(6) Writing data to the port will program the port to the output mode and output the data.

(7) INTREQ is the inverted status of the $\overline{INTREQ}$ pin.

When a 99XX CPU executes a CRU Instruction, the processor uses the contents of workspace register 12 as a base address. (Refer to the *9900 Microprocessor Data Manual* for a complete discussion on how CRU addresses are derived.) The CRU address is brought out on the 15-bit address bus; this means that the least significant bit of R12 is not brought out of the CPU. During CRU cycles, the memory control lines ($\overline{\text{MEMEN}}$, $\overline{\text{WE}}$, and DBIN) are all inactive; $\overline{\text{MEMEN}}$ being inactive (HIGH) indicates the address is not a memory address and therefore is a CRU address or external instruction code. Also, when $\overline{\text{MEMEN}}$ is inactive (HIGH) and a valid address is present, address bits A0-A2 must all be zero to constitute a valid CRU address; if address bits A0-A2 are other than all zeros, they are indicating an external instruction code. In summary, address bits A3-A14 contain the CRU address to be decoded, address bits A0-A2 must be zero and $\overline{\text{MEMEN}}$ must be inactive (HIGH) to indicate a CRU cycle.

2.2 Interrupt Interface

A block diagram of the interrupt control section is shown in Figure 3. The interrupt inputs (six dedicated, $\overline{\text{INT1}}$-$\overline{\text{INT6}}$, and nine programmable) are sampled on the falling edge of $\overline{\phi}$ and latched onto the chip for one $\overline{\phi}$ time by the SYNC LATCH, each $\overline{\phi}$ time. The output of the sync latch is inverted (interrupts are LOW active) and ANDed with its respective mask bit (MASK = 1, INTERRUPT ENABLED). On the rising edge of $\overline{\phi}$, the prioritizer and encoder senses the masked interrupts and produces a four-bit encoding of the highest priority interrupt present (see Tables 2 and 3). The four-bit prioritized code and $\overline{\text{INTREQ}}$ are latched off-chip with a sync latch on the falling edge of the next $\overline{\phi}$, which ensures proper synchronization to the processor.

Once an interrupt goes active (LOW), it should stay active until the appropriate interrupt service routine explicitly turns off the interrupt. If an interrupt is allowed to go inactive before the interrupt service routine is entered, an erroneous interrupt code could be sent to the processor. A total of five clock cycles occur between the time the CPU samples the $\overline{\text{INTREQ}}$ line and the time it samples the IC0-IC3 lines. For example, if an interrupt is active and the CPU recognizes that an interrupt is pending, *but before the CPU can sample the interrupt control lines the interrupt goes inactive,* the interrupt control lines will contain an incorrect code.

The interrupt mask bits on the TMS 9901 PSI are individually set or reset under software control. Any unused interrupt line should have its associated mask disabled to avoid false interrupts: To do this, the control bit (CRU bit zero), is first set to a zero for interrupt mode operation. Writing to TMS 9901 CRU bits 1-15 will enable or disable interrupts 1-15, respectively. Writing a one to an interrupt mask will *enable* that interrupt; writing a zero will *disable* that interrupt. Upon application of $\overline{\text{RST1}}$ (power-up reset), all mask bits are reset (LOW), the interrupt code is forced to all zeros, and $\overline{\text{INTREQ}}$ is held HIGH. Reading TMS 9901 CRU bits 1-15 indicates the status of the respective interrupt inputs; thus, the designer can employ the unused (disabled) interrupt input lines as data inputs (true data in).

2.3 Input/Output Interface

A block diagram of the TMS 9901 I/O interface is shown in Figure 4. Up to 16 individually controlled, I/O ports are available (seven dedicated, P0-P6, and nine programmable) and, as discussed above, the unused dedicated interrupt lines also can be used as input lines (true data in). Thus the 9901 can be configured to have more than 16 inputs. $\overline{\text{RST1}}$ (power-up reset) will program all I/O ports to input mode. Writing data to a port will automatically switch that port to the output mode. Once programmed as an output, a port will remain in output mode until $\overline{\text{RST1}}$ or $\overline{\text{RST2}}$ (command bit) is executed. An output port can be read and indicates the present state of the pin. A pin programmed to the output mode *cannot* be used as an input pin: *Applying an input current to an output pin may cause damage to the TMS 9901.* The TMS 9901 outputs are latched and buffered off-chip, and inputs are buffered onto the chip. The output buffers are MOS-to-TTL buffers and can drive two standard TTL loads.

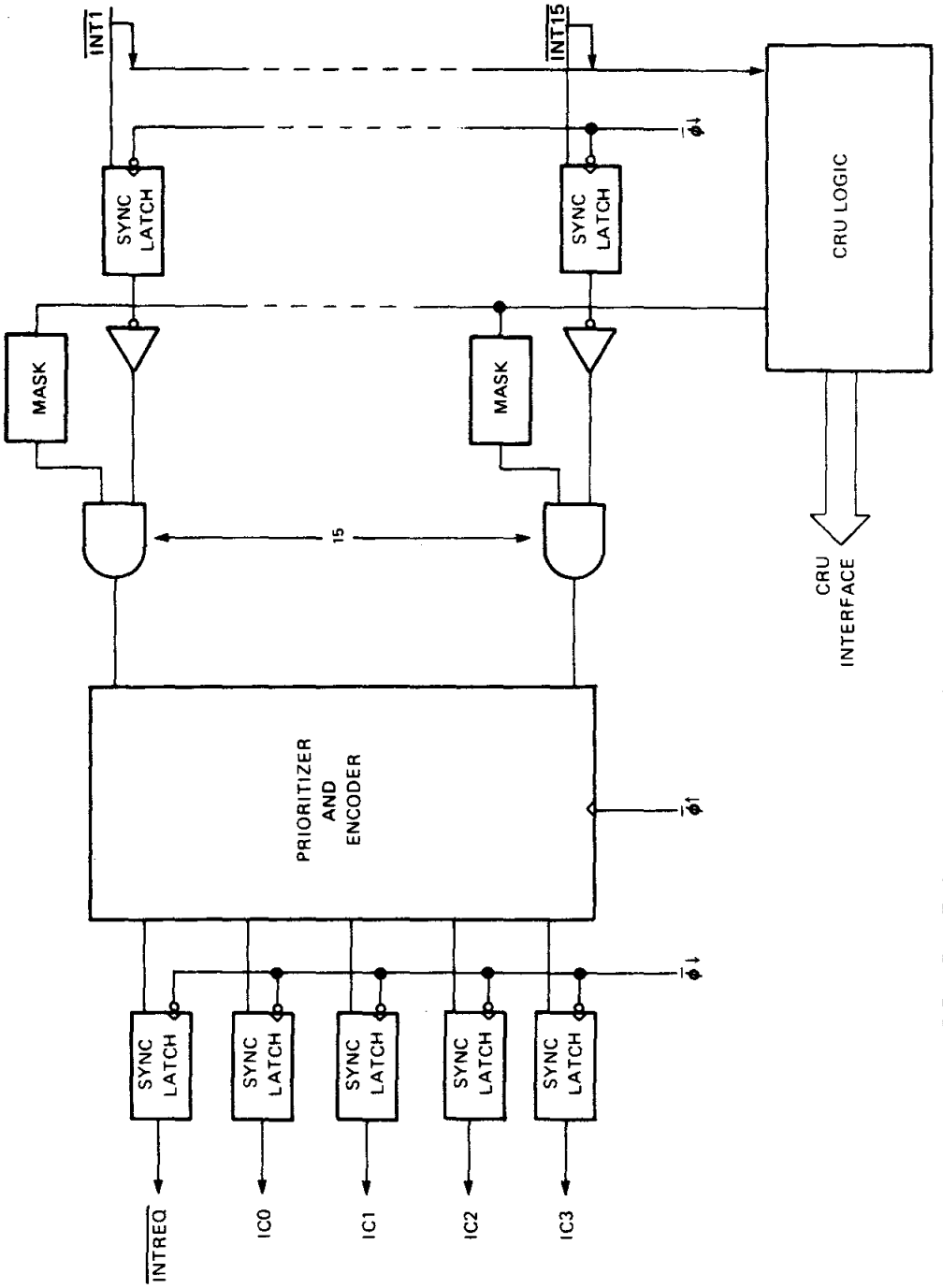


FIGURE 3– TMS 9901 PSI INTERRUPT CONTROL SECTION BLOCK DIAGRAM

TABLE 2
INTERRUPT CODE GENERATION

INTERRUPT/STATE	PRIORITY	I_{C0}	I_{C1}	I_{C2}	I_{C3}	$\overline{INTREQ}$
$\overline{RST}$ 1	–	0	0	0	0	1
$\overline{INT}$ 1	1 (HIGHEST)	0	0	0	1	0
$\overline{INT}$ 2	2	0	0	1	0	0
$\overline{INT}$ 3/CLOCK	3	0	0	1	1	0
$\overline{INT}$ 4	4	0	1	0	0	0
$\overline{INT}$ 5	5	0	1	0	1	0
$\overline{INT}$ 6	6	0	1	1	0	0
$\overline{INT}$ 7	7	0	1	1	1	0
$\overline{INT}$ 8	8	1	0	0	0	0
$\overline{INT}$ 9	9	1	0	0	1	0
$\overline{INT}$ 10	10	1	0	1	0	0
$\overline{INT}$ 11	11	1	0	1	1	0
$\overline{INT}$ 12	12	1	1	0	0	0
$\overline{INT}$ 13	13	1	1	0	1	0
$\overline{INT}$ 14	14	1	1	1	0	0
$\overline{INT}$ 15	15 (LOWEST)	1	1	1	1	0
NO INTERRUPT	–	1	1	1	1	1

TABLE 3
TMS 9980A OR TMS 9981 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	------	------	

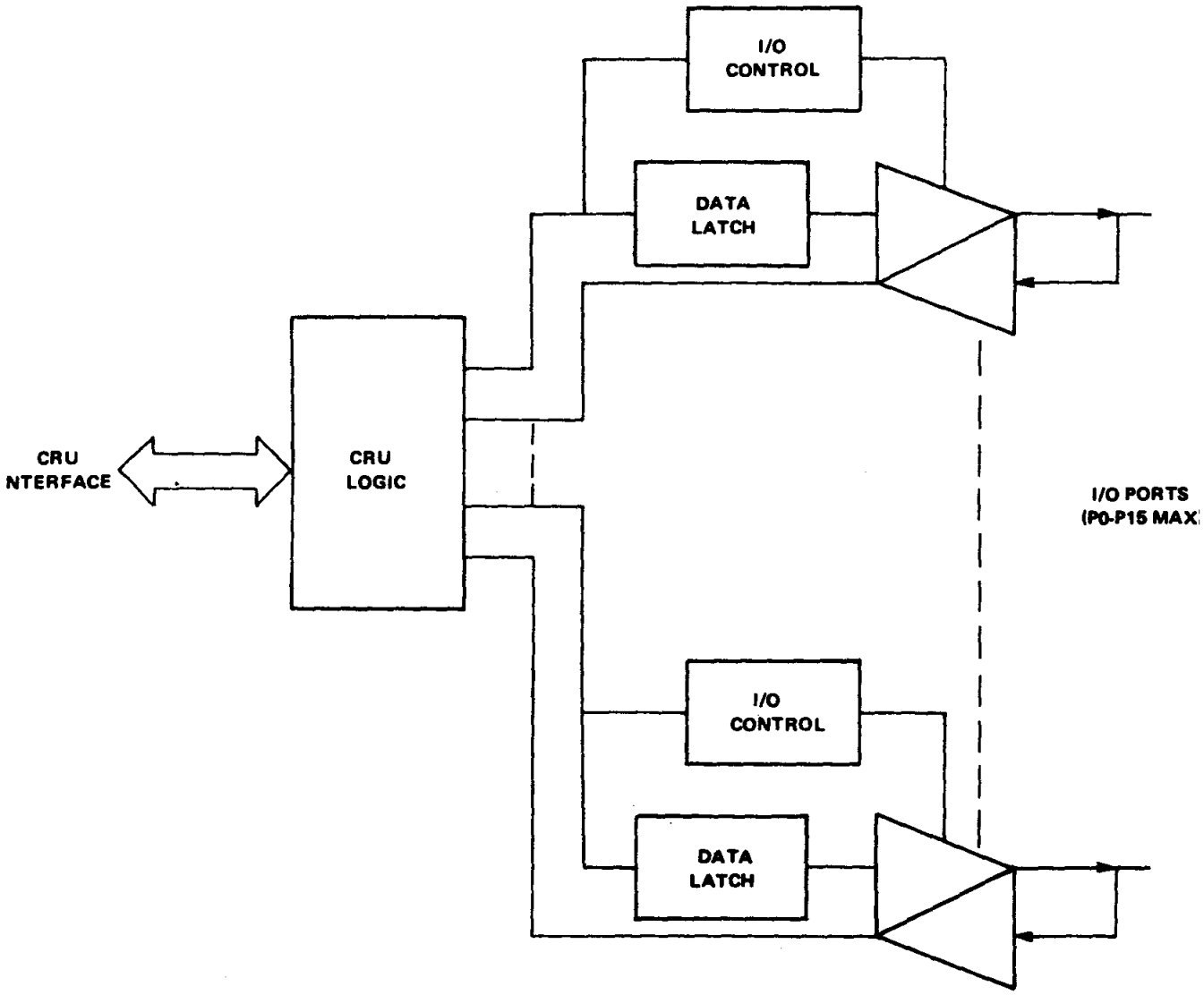
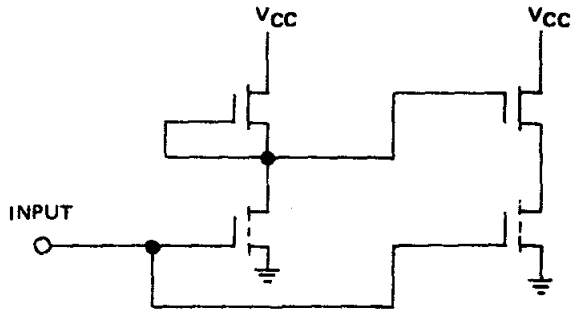


FIGURE 4—TMS 9901 I/O INTERFACE SECTION

EQUIVALENT OF I/O INPUTS



EQUIVALENT OF I/O OUTPUTS

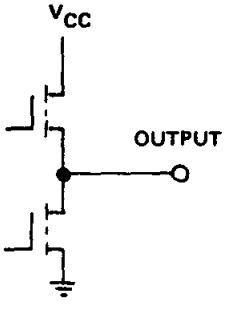


FIGURE 5 – INPUT AND OUTPUT EQUIVALENTS

2.4 Programmable Ports

A total of nine pins ($\overline{INT7}$/P15-$\overline{INT15}$/P7) on the TMS 9901 are user-programmable as either I/O ports or interrupts. These pins will assume all characteristics of the type pin they are programmed to be (as described in Sections 2.2 and 2.3). Any pin which is not being used for interrupt should have the appropriate interrupt mask disabled (mask = 0) to avoid erroneous interrupts to the CPU. To program one of the pins as an interrupt, its interrupt mask simply is enabled and the line may be used as if it were one of the dedicated interrupt lines. To program a pin as an I/O port, disable the interrupt mask and use that pin as if it were one of the dedicated I/O ports.

2.5 Interval Timer

Figure 6 is a block diagram of the TMS 9901 interval timer section. The clock consists of a 14-bit counter that decrements at a rate of $f(\overline{\phi})/64$ (at 3 MHz this results in a maximum interval of 349 milliseconds with a resolution of 21.3 microseconds). The clock can be used as either an interval timer or an event timer. To access the clock, select bit zero (control bit) must be set to a one. The clock is enabled to cause interrupts by writing a nonzero value to it and is then disabled from interrupting by writing zero to it or by a $\overline{RST1}$. The clock starts operating at no more than two $\overline{\phi}$ times after it is loaded. When the clock decrementer is running, it will decrement down to zero and issue a level-3 interrupt. The decrementer, when it becomes zero, will also be reloaded from the clock register and decrementing will start again. (The zero state is counted as any other decrementer state.) The decrementer always runs, but it will not issue interrupts unless enabled; of course, the contents of the unenabled clock read register are meaningless.

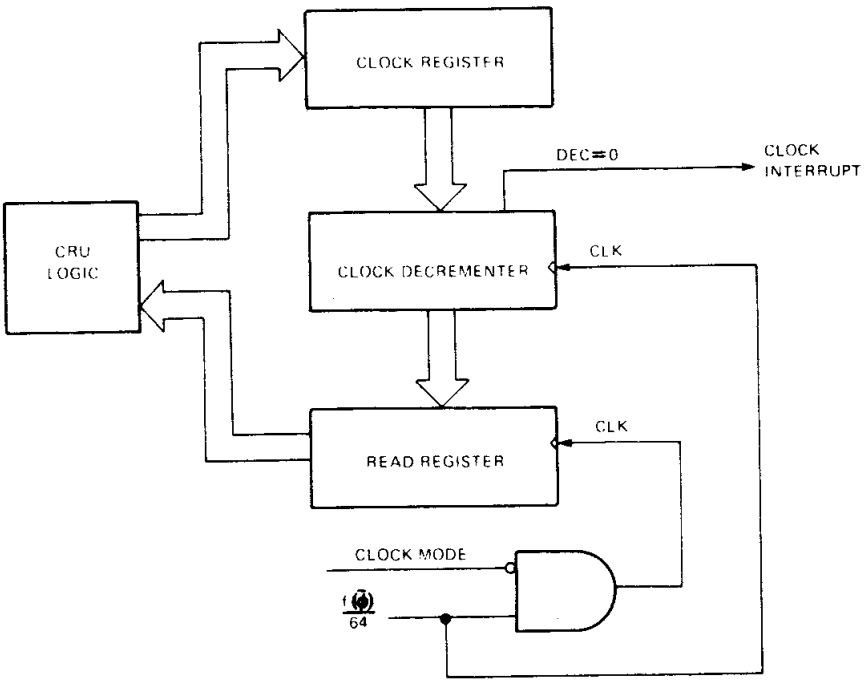


FIGURE 6–TMS 9901 INTERVAL TIMER SECTION

The clock is accessed by writing a one into the control bit (TMS 9901 CRU bit zero) to force CRU bits 1-15 to clock mode. Writing a nonzero value into the clock register then enables the clock and sets its time period. When the clock is enabled, it interrupts on level 3 and external level-3 interrupts are disabled. The mask for level 3 in the PSI must be set to a one so that the processor will see the clock interrupt. When the clock interrupt is active, the clock mask (mask bit 3) must be written into with either a one or zero to clear the interrupt; writing a zero also disables further interrupts.

If a new clock value is required, a new 14-bit clock start value can be programmed by executing a CRU write operation to the clock register. During programming, the decrementer is restarted with the current start value after each start value bit is written. A timer restart is easily implemented by writing a single bit to any of the clock bits. The clock is disabled by $\overline{RST}1$ (power up reset) or by writing a zero value into the clock register; $\overline{RST}2$ does not affect the clock.

The clock read register is updated every time the decrementer decrements when the TMS 9901 is not in clock mode. There are two methods to leave the clock mode : first, a zero is written to the control bit; or second, a TMS 9901 select bit greater than15 is accessed. Note that when $\overline{CE}$ is inactive(HIGH),the PSI is not disabled from seeing the select lines. As the CPU is addressing memory, A10-A14 could very easily have a value of 15 or greater — A10-A14 are connected to the select lines; therefore, the TMS 9901 interval timer section can "think" it is out of clock mode and update the clock read register. Very simply, this means that a value cannot be locked into the clock read register by writing a one to CRU select bit zero (the control bit). The 9901 must be out of clock mode for at least one timer period to ensure that the contents of the clock read register has been updated. This means that to read the most recent contents of the decrementer, just before reading, the TMS 9901 *must* not be in the clock mode. The only sure way to manipulate clock mode is to use the control bit (select bit zero). When clock mode is reentered to access the clock read register, updating of the read register will cease. This is done so that the contents of the clock read register will not change while it is being accessed.

2.6 Power-Up Considerations

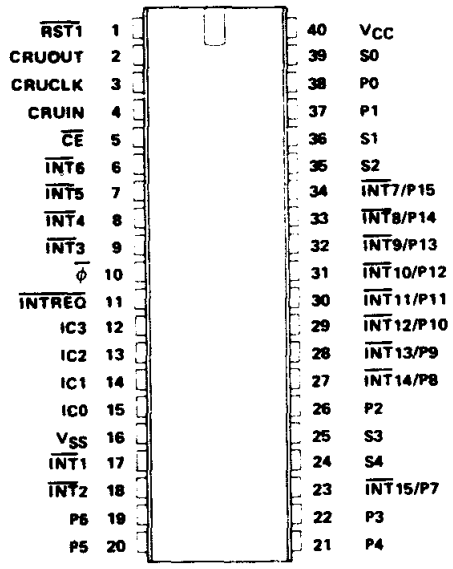
During hardware reset, $\overline{RST}1$ must be active (LOW) for a minimum of two clock cycles to force the TMS 9901 into a known state. $\overline{RST}1$ will disable all interrupts, disable the clock, program all I/O ports to the input mode, and force IC0-IC3 to all zeros with $\overline{INTREQ}$ held HIGH. The system software must enable the appropriate interrupts, program the clock, and configure the I/O ports as required. After initial power-up the TMS 9901 is accessed only as needed to service the clock, enable (disable) interrupts, or read (write) data to the I/O ports. The I/O ports can be reconfigured by use of the $\overline{RST}2$ software reset command bit.

2.7 Pin Descriptions

Table 4 defines the TMS 9901 pin assignments and describes the function of each pin.

TABLE 4
TMS 9901 PIN ASSIGNMENTS AND FUNCTIONS

SIGNATURE	PIN	I/O	DESCRIPTION
$\overline{\text{INTREQ}}$	11	OUT	INTERRUPT Request. When active (low) $\overline{\text{INTREQ}}$ indicates that an enabled interrupt has been received. $\overline{\text{INTREQ}}$ will stay active until all enabled interrupt inputs are removed.
IC0 (MSB)	15	OUT	Interrupt Code lines. IC0-IC3 output the binary code corresponding to the highest priority enabled interrupt. If no enabled interrupts are active IC0–IC3 = (1,1,1,1).
IC1	14	OUT	
IC2	13	OUT	
IC3 (LSB)	12	OUT	
$\overline{\text{CE}}$	5	IN	Chip Enable. When active (low) data may be transferred through the CRU interface to the CPU. $\overline{\text{CE}}$ has no effect on the interrupt control section.
S0	39	IN	Address select lines. The data bit being accessed by the CRU interface is specified by the 5-bit code appearing on S0-S4.
S1	36	IN	
S2	35	IN	
S3	25	IN	
S4	24	IN	
CRUIN	4	OUT	CRU data in (to CPU). Data specified by S0-S4 is transmitted to the CPU by CRUIN. When $\overline{\text{CE}}$ is not active CRUIN is in a high-impedance state.
CRUOUT	2	IN	CRU data out (from CPU). When $\overline{\text{CE}}$ is active, data present on the CRUOUT input will be sampled during CRUCLK and written into the command bit specified by S0-S4.
CRUCLK	3	IN	CRU Clock (from CPU). CRUCLK specifies that valid data is present on the CRUOUT line.
$\overline{\text{RST1}}$	1	IN	Power Up Reset. When active (low) $\overline{\text{RST1}}$ resets all interrupt masks to "0", resets IC0 – IC3 = (0, 0, 0, 0), $\overline{\text{INTREQ}}$ = 1, disables the clock, and programs all I/O ports to inputs. $\overline{\text{RST1}}$ has a Schmitt-triger input to allow implementation with an RC circuit as shown in Figure 7.
V_{CC}	40		Supply Voltage. +5 V nominal.
V_{SS}	16		Ground Reference
$\overline{\phi}$	10	IN	System clock ($\overline{\phi}$3 in TMS 9900 system, $\overline{\text{CKOUT}}$ in TMS 9980 system).
$\overline{\text{INT1}}$	17	IN	Group 1, interrupt inputs. When active (Low) the signal is ANDed with its corresponding mask bit and if enabled sent to the interrupt control section. $\overline{\text{INT1}}$ has highest priority.
$\overline{\text{INT2}}$	18	IN	
$\overline{\text{INT3}}$	9	IN	
$\overline{\text{INT4}}$	8	IN	
$\overline{\text{INT5}}$	7	IN	
$\overline{\text{INT6}}$	6	IN	
$\overline{\text{INT7}}$/ P15	34	I/O	Group 2, programmable interrupt (active low) or I/O pins (true logic). Each pin is individually programmable as an interrupt, an input port, or an output port.
$\overline{\text{INT8}}$/ P14	33	I/O	
$\overline{\text{INT9}}$/ P13	32	I/O	
$\overline{\text{INT10}}$/P12	31	I/O	
$\overline{\text{INT11}}$/P11	30	I/O	
$\overline{\text{INT12}}$/P10	29	I/O	
$\overline{\text{INT13}}$/P9	28	I/O	
$\overline{\text{INT14}}$/P8	27	I/O	
$\overline{\text{INT15}}$/P7	23	I/O	
P0	38	I/O	Group 3, I/O ports (true logic). Each pin is individually programmable as an input port or an output port.
P1	37	I/O	
P2	26	I/O	
P3	22	I/O	
P4	21	I/O	
P5	20	I/O	
P6	19	I/O	



3. APPLICATIONS

3.1 Hardware Interface

Figure 7 illustrates the use of a TMS 9901 PSI in a TMS 9900 system. The TIM 9904 clock generator/driver syncs the $\overline{\text{RESET}}$ for both the TMS 9901 and the CPU. The RC circuit on the TIM 9904 provides the power-up and pushbutton $\overline{\text{RESET}}$ input to the clock chip. Address lines A0-A9 are decoded on CRU cycles to select the TMS 9901. Address lines A10-A14 are sent directly to PSI select lines S0-S4, respectively, to select which TMS 9901 CRU bit is to be accessed.

Figure 8 illustrates the use of a TMS 9901 with a TMS 9981 CPU. No TIM 9904 is needed with the TMS 9981, so the reset circuitry is connected directly to the system reset line. The clock ($\overline{\phi}$) then comes from the TMS 9981. All other circuitry is identical to the TMS 9900 system.

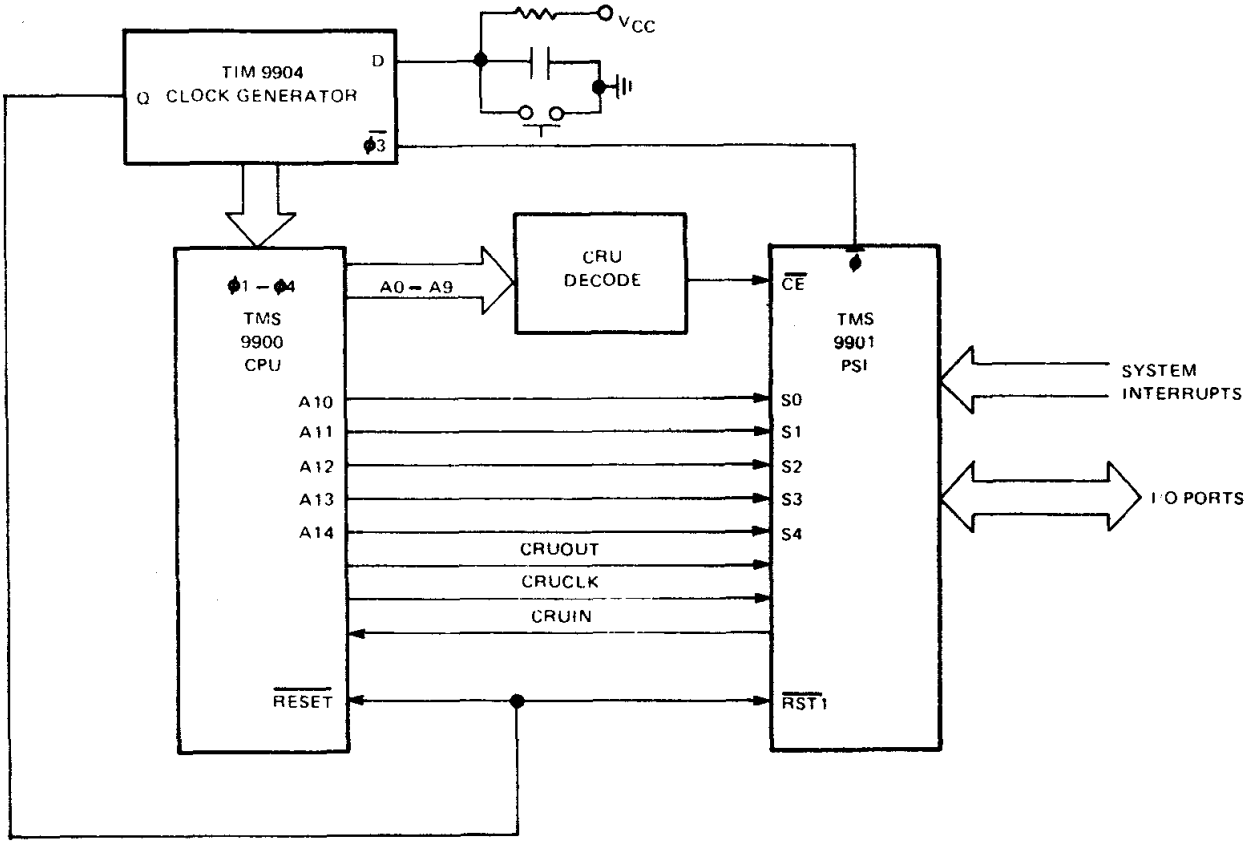


FIGURE 7–TMS 9900/TMS 9901 INTERFACE

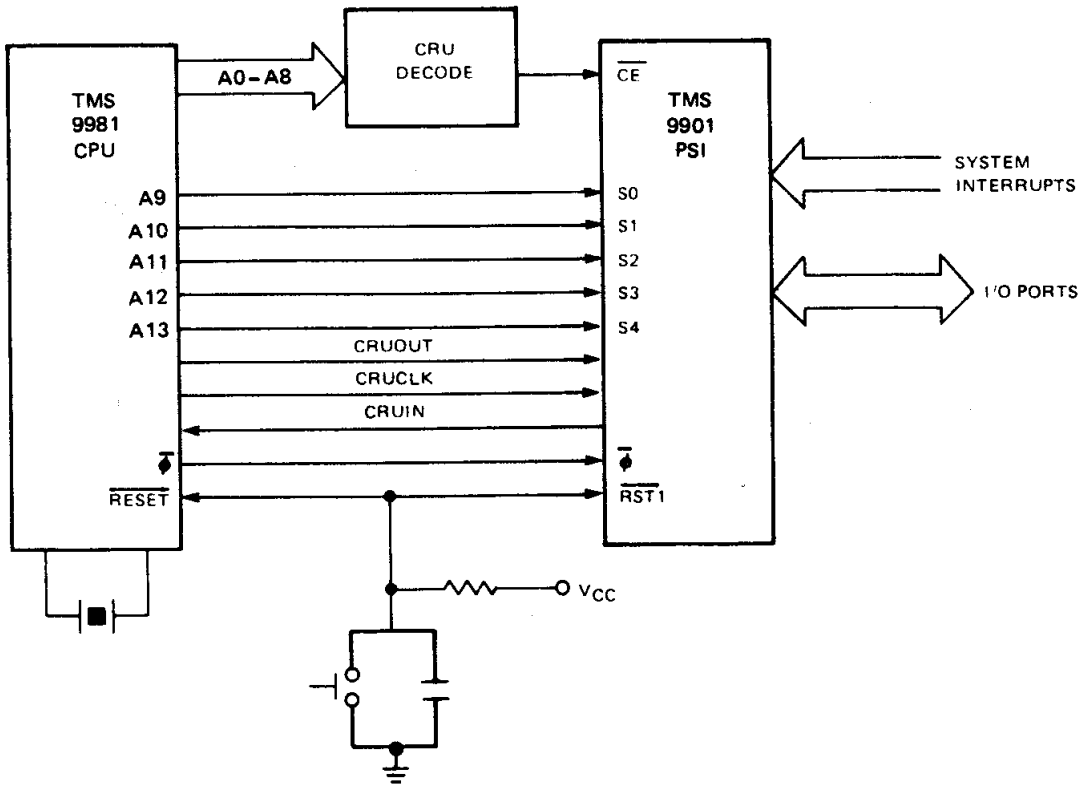


FIGURE 8–TMS 9981/TMS 9901 INTERFACE

3.2 Software Interface

Figure 9 lists the TMS 9900 code needed to control the TMS 9901 PSI. The code initializes the PSI to an eight-bit input port, an eight-bit output port, and enables interrupt levels 1-6. The six dedicated interrupt pins are all used for interrupts; their mask bits are set ON. The nine programmable pins are all used as I/O ports; mask bits 7-15 remain reset. P0-P7 are programmed as an eight-bit output port, and P8-P15 are programmed as an eight-bit input port.

Some code is added to read the contents of the clock read-register. The SBZ instruction takes the TMS 9901 out of clock mode long enough for the clock read register to be updated with the most recent decrementer value. When clock mode is reentered, the decrementer will cease updating the clock read-register so that the contents of the register will not be changing during a read operation.

The second section of code is typical code found in a clock interrupt service routine. All interrupts initially are disabled by the routine. These functions are not necessary, but are usually done to ensure system integrity. The interrupt mask should be restored as soon as the sensitive processing is complete. The interrupt is counted in the variable COUNT and is then cleared by writing a one to mask bit 3. If a zero is written to mask bit 3 to clear the interrupt, clock interrupt will be disabled from that point onward, but the clock will continue to run.

ASSUMPTION:

- System uses clock at maximum interval (349 msec @ 3MHz)
- Interrupts 1-6 are used
- Eight bits are used as an output port , P0 –P7
- Eight bits are used as an input port , P8 – P15
- $\overline{\text{RST1}}$ (power-up reset) has been applied
- The most significant byte of R1 contains data to be output.

```
        LI      R12, PSIBAS     Set up CRU base to point to 9901
        LDCR    @CLKSET, 0      16-bit transfer, set clock to max interval

        LDCR    @INTSET, 7      Enter interrupt mode and enable interrupts 1 – 6

        LI      R12, PSIBAS+32  Set CRU base to I/O ports – output
        LDCR    R1, 8           Output byte from R1, program ports 0 – 7 as output

        LI      R12, PSIBAS+48  Set CRU base to I/O ports – input
        STCR    R2, 8           Store a byte from input port into MSBT of R2

        LI      R12, PSIBAS     Set CRU base to 9901
        SBZ     0               Leave clock mode so decremented contents can be latched
        INCT    R12             Set CRU base to clock read register
        SBO     -1              Enter clock mode
        STCR    R3, 14          Read 14-bit clock read register contents into R3

CLKSET  DATA    >FFFF
INTSET  BYTE    >7E


CLKINT  $                       Clock interrupt service routine – level 3
        LIMI    0               Disable interrupts at CPU
        INC     @COUNT          Count the clock interrupt
        LI      R12, PSIBAS     Set CRU base to point to 9901
        SBZ     0               Enter interrupt mode
        SBO     3               Clear clock interrupt
```

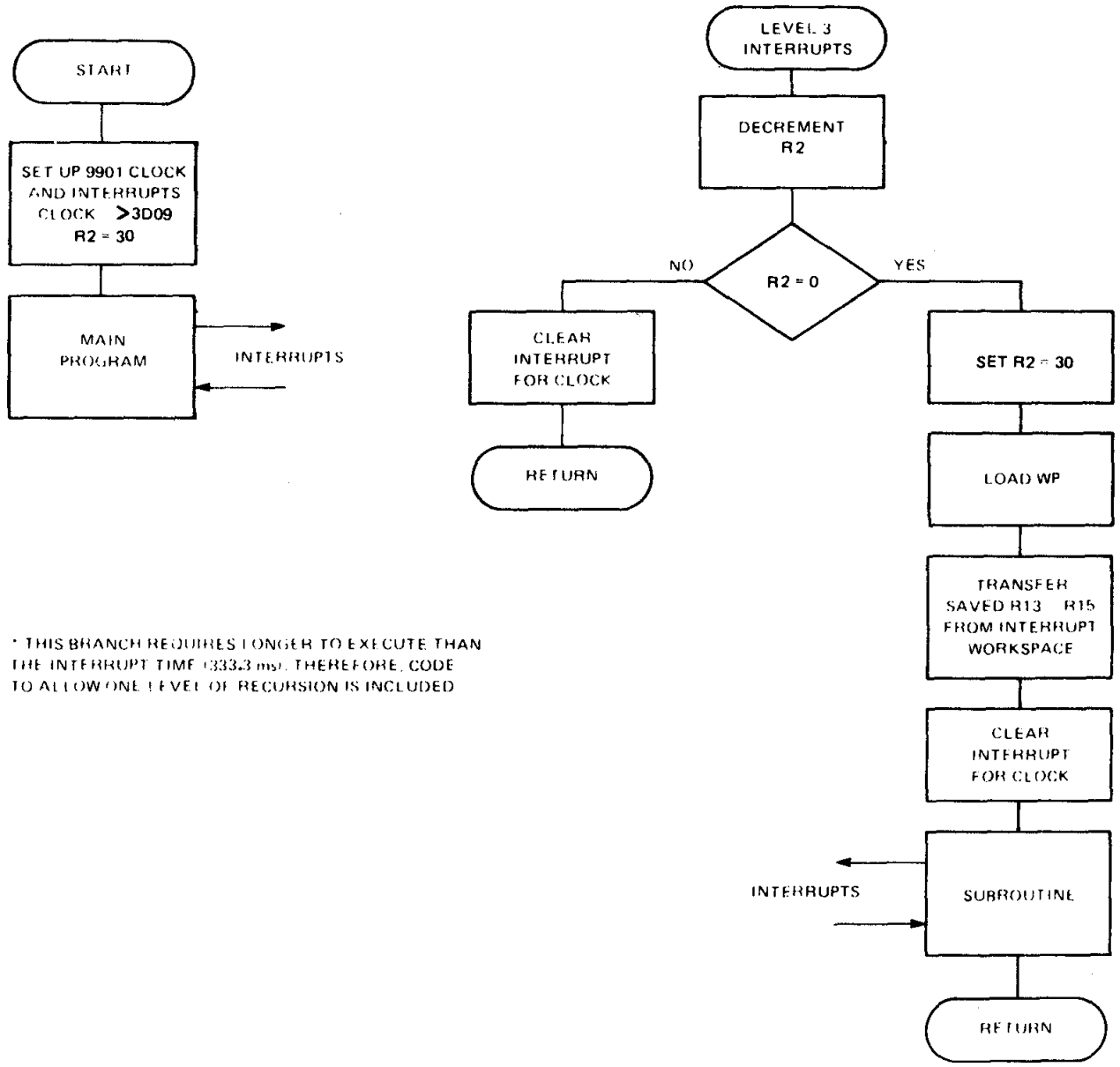
FIGURE 9 – TMS 9900 SAMPLE SOFTWARE TO CONTROL THE TMS 9901

3.3 Interval Timer Application

A TM 990/100M microcomputer board application in which every 10 seconds a specific task must be performed is described below. The TMS 9901 clock is set to interrupt every 333.33 milliseconds. This is accomplished by programming the 14-bit clock register to $3D09_{16}$ ($15{,}625_{10}$). The TM 990/100M microcomputer board system clock runs at 3 MHz, giving a clock resolution of 21.33 microseconds. A decrementer period of 21.33 microseconds multiplied by 15,625 periods until interrupt gives 333.33 milliseconds between interrupts. The interrupt service routine must count 30 interrupts before 10 seconds elapses:

$$f_{(\,DEC\,)} = \frac{f_{(\phi)}}{64}, \qquad T_{(\,DEC\,)} = \frac{1}{f_{(\,DEC\,)}} = \frac{64}{3{,}000{,}000} = 21.3333\ \mu s$$

Figure 10 is a flowchart of the software required to perform the above application, and Figure 11 is a listing of the code. Following the flowchart, the main routine sets up all initial conditions for the 9901 and clock service routine. The interrupt service routine decrements a counter in R2 which was initialized to 30. When the counter in R2 decrements to zero, 10 seconds have elapsed, and the work portion of the service routine is entered. Note carefully that the work portion of the service routine takes longer than 333.33 ms which is the time between clock interrupts from the 9901. Therefore, recursive interrupts are going to occur and some facility must be provided to handle them. Loading a new workspace pointer and transferring the saved WP, PC, and ST (R13-R15) from the interrupt workspace to the new workspace allows one level of recursion.



* THIS BRANCH REQUIRES LONGER TO EXECUTE THAN THE INTERRUPT TIME (333.3 ms). THEREFORE, CODE TO ALLOW ONE LEVEL OF RECURSION IS INCLUDED

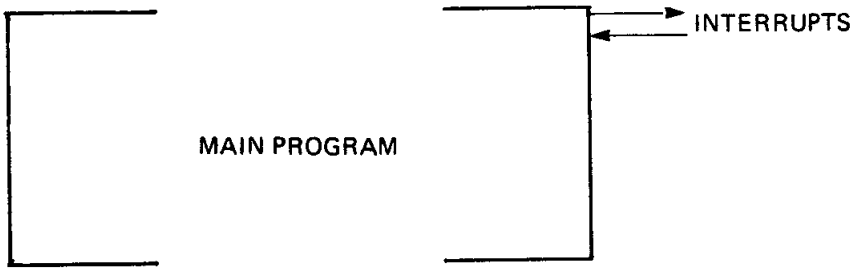
FIGURE 10—TMS 9901 INTERVAL TIMER APPLICATION FLOWCHART

DEVICE INITIALIZATION

```
FE00 02E0 LWPI >FF20
FE02 FF20
FE04 020C LI R12,>100    9901 CRU BASE ADDRESS
FE06 0100
FE08 02E0 LWPI >FF68     INTERRUPT 3 WORKSPACE
FE0A FF68
FE0C 0201 LI R1,>7A13    DATA FOR 333.33MS CLOCK
FE0E 7A13
FE10 0202 LI R2,30       30 X 333.33MS = 10SEC
FE12 001E
FE14 020C LI R12,>100    9901 CRU BASE ADDRESS
FE16 0100
FE18 33C1 LDCR R1,15     LOAD 9901 CLOCK
FE1A 1E00 SBZ 0          SET 9901 TO INTERRUPT MODE
FE1C 1D03 SBO 3          UNMASK INTERRUPT 3
```

MAIN PROGRAM

```
FD00 02E0 LWPI >FF00     MAIN PROGRAM WORKSPACE
FD02 FF00
FD04 0300 LIMI 3         ENABLE INT 0-3
FD06 0003
```



NOTE: This code was assembled using the TM 990/402 line-by-line assembler.

FIGURE 11–INTERVAL TIMER

INTERRUPT 3 SERVICE ROUTINE
(WP = FF68)

```
FD80 0602 DEC R2          COUNT DOWN 30 IN R2
FD82 1302 JEQ >FD88       IF ZERO THEN JUMP
FD84 1D03 SBO 3           CLEAR 9901 CLOCK INTERRUPT
FD86 0380 RTWP            RETURN TO INTERRUPTED ROUTINE
FD88 0202 LI R2,30        RELOAD R2 FOR 10 SEC COUNT DOWN
FD8A 001E
FD8C 0460 B @>FC80        BRANCH TO SUBROUTINE
FD8E FC80
```

ROUTINE TO BE PERFORMED EVERY 10 SECONDS, IT TAKES
LONGER THAN 333.33 MS WHICH IS 9901 CLOCK PERIOD'

```
FC80 02E0 LWPI >FF20      WORKSPACE FOR SUBROUTINE
FC82 FF20
FC84 C360 MOV @>FF82,R13  TRANSFER SAVED WP,PC,ST FROM
FC86 FF82
FC88 C3A0 MOV @>FF84,R14  INT 3 WORKSPACE
FC8A FF84
FC8C C3E0 MOV @>FF86,R15
FC8E FF86
FC90 1D03 SBO 3           CLEAR 9901 CLOCK INTERRUPT
FC92 0300 LIMI 3          ENABLE INT 0-3
FC94 0003
```


```
FFC0        RTWP
```

FIGURE 11–(CONCLUDED)

4. TMS 9901 ELECTRICAL SPECIFICATIONS

4.1 Absolute Maximum Ratings Over Operating Free Air Temperature Range (Unless Otherwise Noted)*

Supply voltage, V_{CC} ..... −0.3 V to 10 V
All inputs and output voltages ..... −0.3 V to 10 V
Continuous power dissipation ..... 0.85 W
Operating free-air temperature range ..... 0°C to 70°C
Storage temperature range ..... −65°C to 150°C

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

4.2 Recommended Operating Conditions*

PARAMETER	MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}	4.75	5.0	5.25	V
Supply voltage, V_{SS}		0		V
High-level input voltage, V_{IH}	2.0		V_{CC}	V
Low-level input voltage, V_{IL}	V_{SS}−.3		0.8	V
Operating free-air temperature, T_A	0		70	°C

4.3 Electrical Characteristics Over Full Range of Recommended Operating Conditions (Unless Otherwise Noted)*

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH}	High level output voltage	I_{OH} = −100 μA	2.4		V_{CC}	V
		I_{OH} = −200 μA	2.2		V_{CC}	V
V_{OL}	Low level output voltage	I_{OL} = 3.2 mA	V_{SS}		0.4	V
I_I	Input current (any input)	V_I = 0 V to V_{CC}			±100	μA
$I_{CC(av)}$	Average supply current from V_{CC}	$t_{c(\phi)}$ = 330 ns, T_A = 70°C			150	mA
C_I	Small signal input capacitance, any input	f = 1 MHz			15	pF

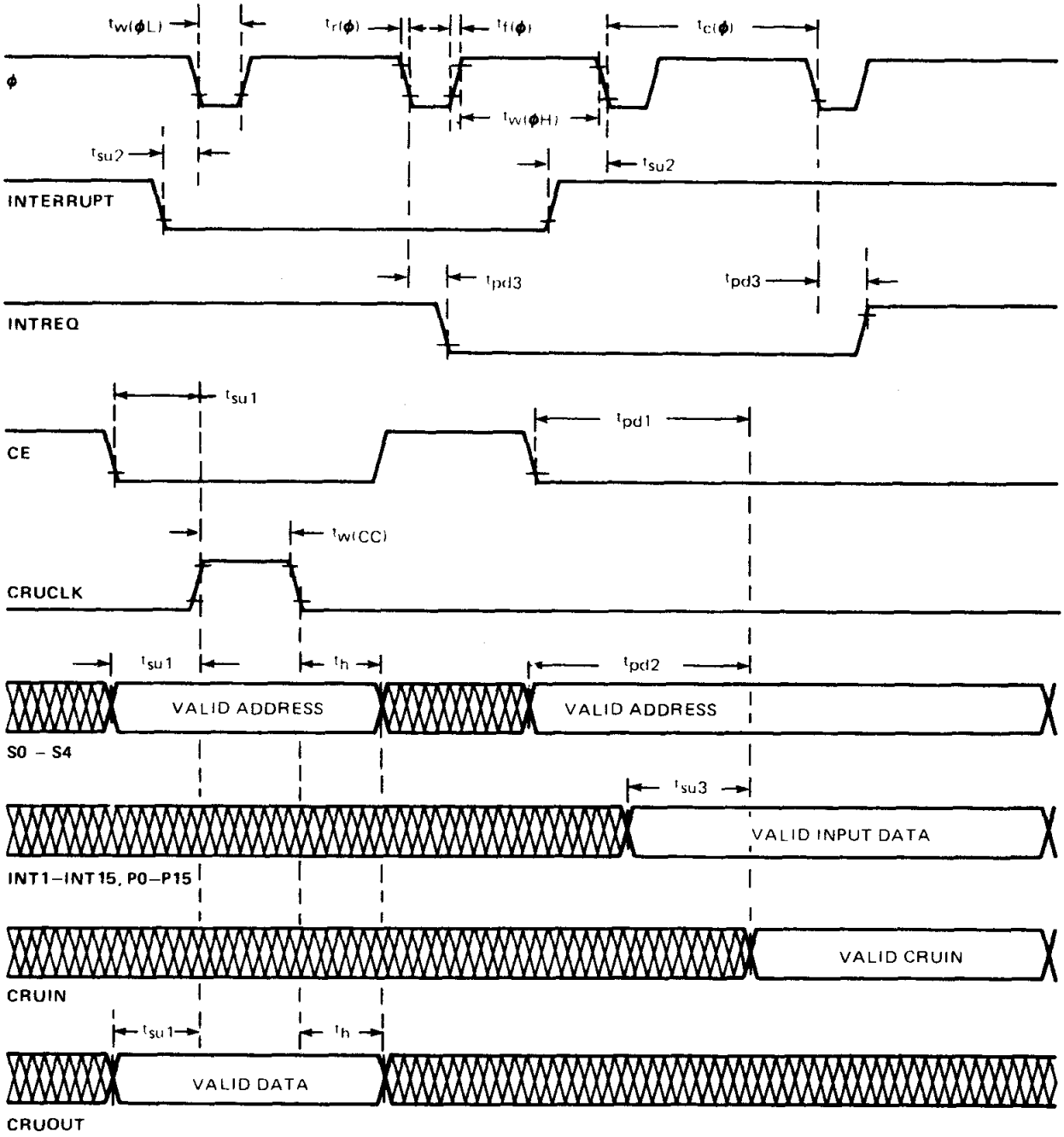
4.4 Timing Requirements Over Full Range of Operating Conditions

PARAMETER		MIN	TYP	MAX	UNIT
$t_{c(\phi)}$	Clock cycle time	300	333	2000	ns
$t_{r(\phi)}$	Clock rise time	5		40	ns
$t_{f(\phi)}$	Clock fall time	10		40	ns
$t_{w(\phi H)}$	Clock pulse width (high level)	225			ns
$t_{w(\phi L)}$	Clock pulse width (low level)	45		300	ns
$t_{w(CC)}$	CRUCLK pulse width	100	185		ns
t_{su1}	Setup time for $\overline{CE}$, S0-S4, or CRUOUT before CRUCLK	100			ns
t_{su2}	Setup time for interrupt before $\overline{\phi}$ low	60			ns
t_{su3}	Setup time for inputs before valid CRUIN	200			ns
t_h	Hold time for $\overline{CE}$, S0-S4, or CRUOUT after CRUCLK	60			ns

*NOTE: All voltage values are referenced to V_{SS}.

4.5 Switching Characteristics Over Full Range of Recommended Operating Conditions

PARAMETER		TEST CONDITION	MIN	TYP	MAX	UNIT
t_{pd1}	Propagation delay, $\overline{CE}$ to valid CRUIN	C_L = 100 pF			300	ns
t_{pd2}	Propagation delay, S0-S4 to valid CRUIN	C_L = 100 pF			320	ns
t_{pd3}	Propagation delay, ϕ low to valid $\overline{INTREQ}$, IC0-IC3	C_L = 100 pF			110	ns
t_{pd}	Propagation delay, $\overline{CRUCLK}$ to valid data out (P0-P15)	C_L = 100 pF			300	ns



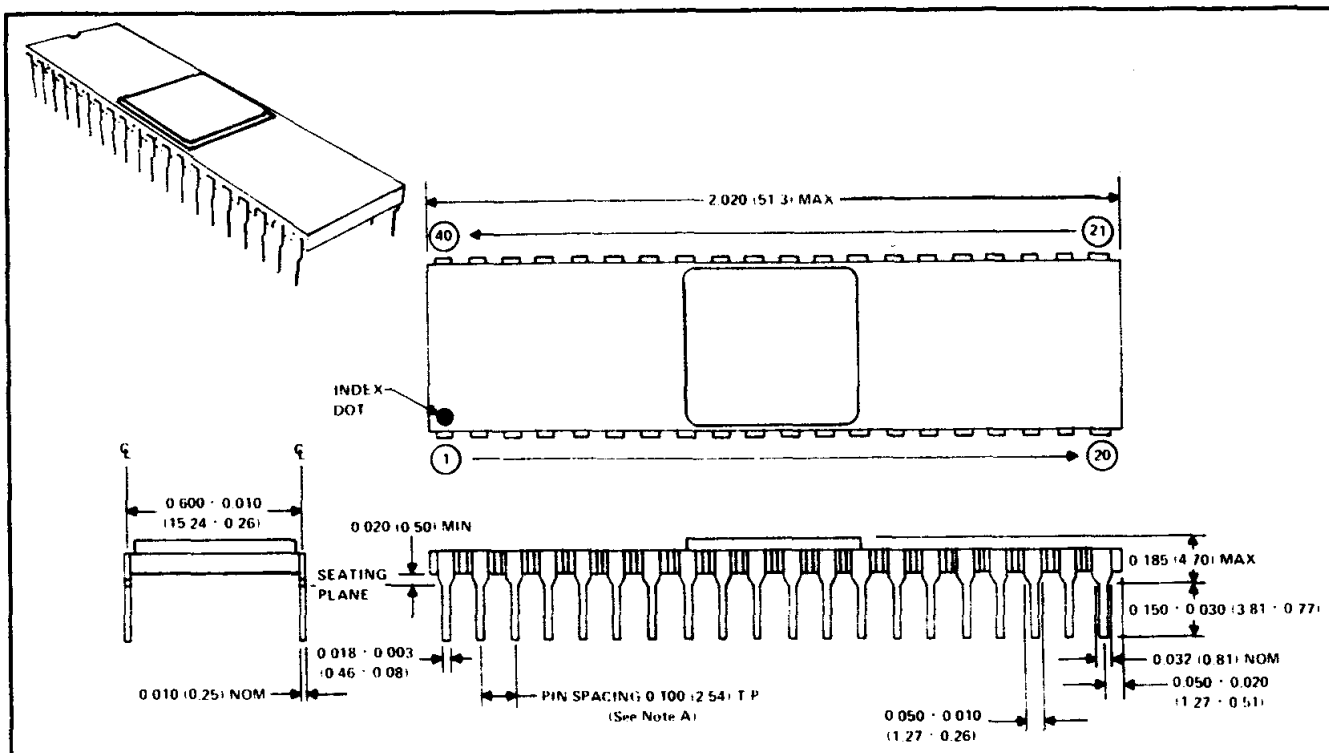
NOTE 1: ALL TIMING MEASUREMENTS ARE FROM 10% and 90% POINTS

FIGURE 12–SWITCHING CHARACTERISTICS

5. MECHANICAL DATA

5.1 TMS 9901 JL — 40 Pin Ceramic Package

ceramic packages with side-brazed leads and metal or epoxy or glass lid seal



NOTES: a. Each pin centerline is located within 0.010 (2,54) of its true longitudinal position.
b. All linear dimensions are in inches and parenthetically in millimeters. Inch dimensions govern.
c. The index is placed in this area to identify pin 1 and to provide other information as follows:

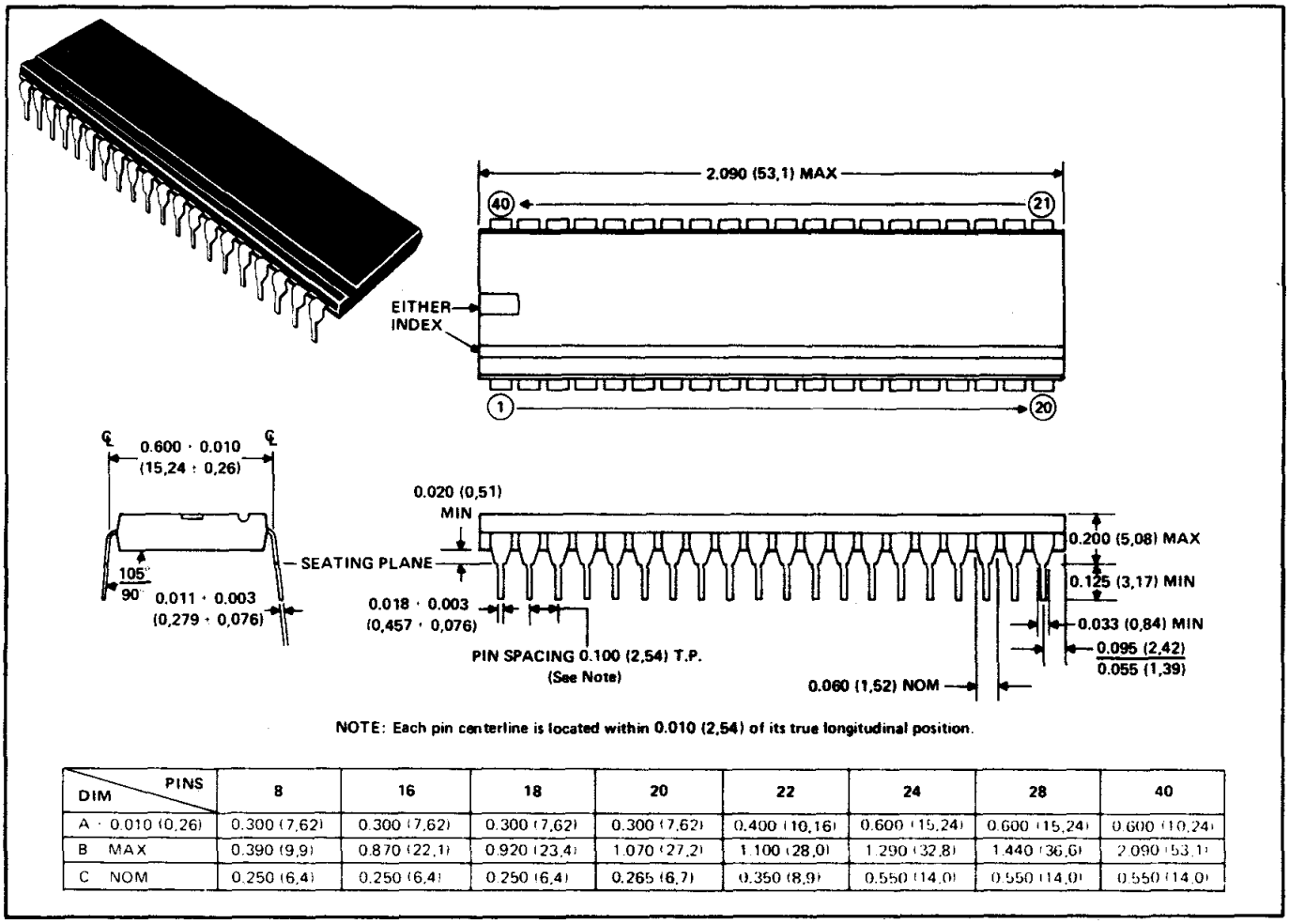
1	Pin 1 connected to chip-mounting pad.
ΔXX	Pin XX connected to chip-mounting pad.
•	No connection to chip-mounting pad.

Other symbols may indicate any combination of up to 4 pins connected to the chip-mounting pad.

DIM \ PINS	16	18	20	22	24	28	40
A ± 0.010 (0.26)	0.300 (7.62)	0.300 (7.62)	0.300 (7.62)	0.400 (10.16)	0.600 (15.24)	0.600 (15.24)	0.600 (15.24)
B MAX	0.840 (21.4)	0.910 (23.1)	1.020 (25.9)	1.100 (28.0)	1.290 (32.8)	1.410 (35.8)	2.020 (51.3)
C NOM	0.290 (7.4)	0.290 (7.4)	0.290 (7.4)	0.290 (7.4)	0.590 (15.0)	0.590 (15.0)	0.590 (15.0)

5.2 TMS 9901 NL – 40 Pin Plastic Package

plastic packages



NOTE: Each pin centerline is located within 0.010 (2,54) of its true longitudinal position.

DIM \ PINS	8	16	18	20	22	24	28	40
A · 0.010 (0,26)	0.300 (7,62)	0.300 (7,62)	0.300 (7,62)	0.300 (7,62)	0.400 (10,16)	0.600 (15,24)	0.600 (15,24)	0.600 (15,24)
B MAX	0.390 (9,9)	0.870 (22,1)	0.920 (23,4)	1.070 (27,2)	1.100 (28,0)	1.290 (32,8)	1.440 (36,6)	2.090 (53,1)
C NOM	0.250 (6,4)	0.250 (6,4)	0.250 (6,4)	0.265 (6,7)	0.350 (8,9)	0.550 (14,0)	0.550 (14,0)	0.550 (14,0)

APPENDIX F

The Engineering Staff of
TEXAS INSTRUMENTS INCORPORATED
Semiconductor Group

TMS 9902 ASYNCHRONOUS COMMUNICATIONS CONTROLLER DATA MANUAL

JULY 1978

TEXAS INSTRUMENTS
INCORPORATED

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

1.1 DESCRIPTION

The TMS 9902 Asynchronous Communications Controller (ACC) is a peripheral device designed for use with the Texas Instruments 9900 family of microprocessors. The TMS 9902 is fabricated using N-channel, silicon gate, MOS technology. The TMS 9902 is TTL-compatible on all inputs and outputs, including the power supply (+5 V) and single-phase clock. The TMS 9902 ACC provides an interface between a microprocessor and a serial, asynchronous, communications channel. The ACC performs the timing and data serialization and deserialization functions, facilitating microprocessor control of the asynchronous channel. The TMS 9902 ACC accepts *EIA Standard RS-232-C* protocol.

1.2 KEY FEATURES

- Low Cost, Serial, Asynchronous Interface
- Programmable, Five- to Eight-Bit, I/O Character Length
- Programmable 1, 1½, and 2 Stop Bits
- Even, Odd, or No Parity
- Fully Programmable Data Rate Generation
- Interval Timer with Resolution from 64 to 16,320 Microseconds
- TTL-Compatibility, Including Power Supply
- Standard 18-Pin Plastic or Ceramic Package
- N-Channel, Silicon Gate Technology

1.3 TYPICAL APPLICATION

Figure 1 shows a general block diagram of a system incorporating a TMS 9902 ACC. Following is a tutorial discussion of this application. Subsequent sections of this Data Manual detail most aspects of TMS 9902 use.

The TMS 9902 interfaces with the CPU through the *communications register unit* (CRU). The CRU interface consists of five address select lines (S0-S4), chip enable ($\overline{CE}$), and three CRU lines (CRUIN, CRUOUT, CRUCLK). An additional input to the CPU is the ACC interrupt line ($\overline{INT}$). The TMS 9902 occupies 32 bits of CRU space; each of the 32 bits are selected individually by processor address lines A10-A14 which are connected to the ACC select lines S0-S4, respectively. Chip enable ($\overline{CE}$) is generated by decoding address lines A0-A9 for CRU cycles. Under certain conditions the TMS 9902 causes interrupts. The interrupt logic shown in Figure 1 can be a TMS 9901.

The ACC interfaces to the asynchronous communications channel on five lines: request to send ($\overline{RTS}$), data set ready ($\overline{DSR}$), clear to send ($\overline{CTS}$), serial transmit data (XOUT), and serial receive data (RIN). The request to send ($\overline{RTS}$) goes active (LOW) whenever the transmitter is activated. However, before data transmission begins, the clear to send ($\overline{CTS}$) input must be active. The data set ready ($\overline{DSR}$) input does not affect the receiver or transmitter. When $\overline{DSR}$ or $\overline{CTS}$ changes level, an interrupt is generated.

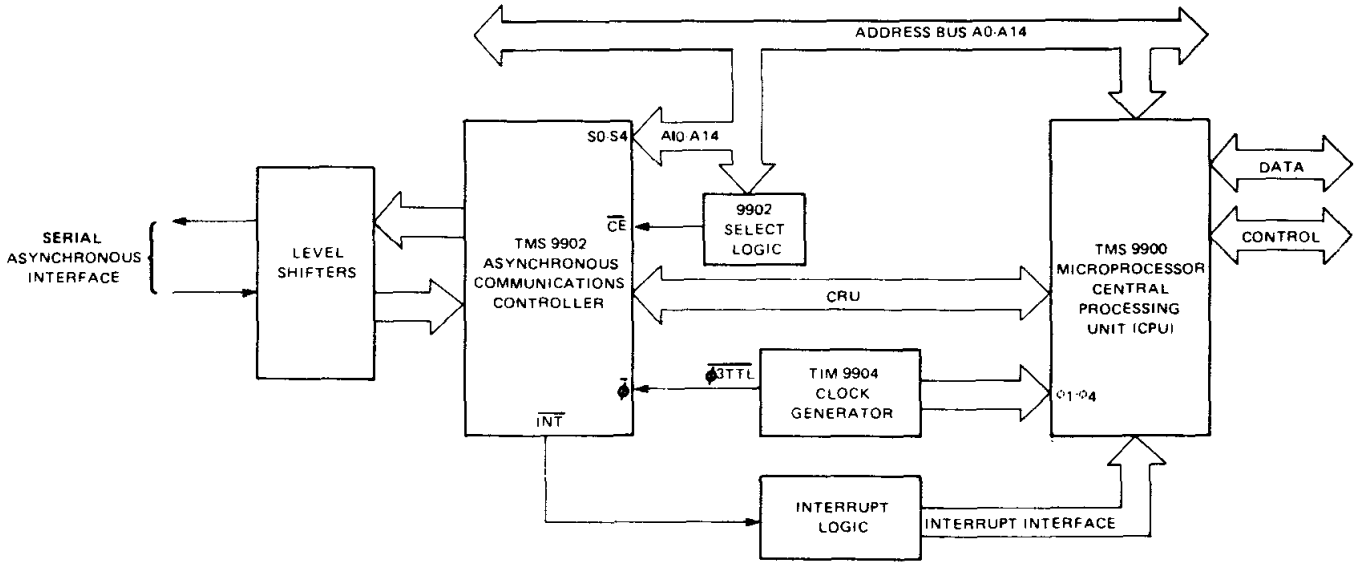


FIGURE 1. TYPICAL APPLICATION, TMS 9902 ASYNCHRONOUS COMMUNICATION CONTROLLER (ACC)

2. ARCHITECTURE

The TMS 9902 asynchronous communications controller (ACC) is designed to provide a low cost, serial, asynchronous interface to the 9900 family of microprocessors. The TMS 9902 ACC is diagrammed in Figure 2. The ACC has five main subsections: CRU interface, transmitter section, receiver section, interval timer, and interrupt section.

2.1 CRU INTERFACE

The communications register unit (CRU) is the means by which the CPU communicates with the TMS 9902 ACC. The ACC occupies 32 bits of CRU read and write space. Figure 3 illustrates the CRU interface between a TMS 9902 and a TMS 9900 CPU; Figure 4 illustrates the CRU Interface for a TMS 9980A or 9981 CPU. The CRU lines are tied directly to each other as shown in Figures 3 and 4. The least significant bits of the address bus are connected to the select lines. In a TMS 9900 CPU system A14-A10 are connected to S4-S0 respectively. The most significant address bits are decoded to select the TMS 9902 via the chip enable ($\overline{CE}$) signal. When $\overline{CE}$ is inactive (HIGH), the CRU interface of the 9902 is disabled.

NOTE

When $\overline{CE}$ is inactive (HIGH) the 9902 sets its CRUIN pin to high impedance and disables CRUCLK from coming on chip. This means the CRUIN line can be used as an OR-tied bus. The 9902 is still able to see the select lines even when $\overline{CE}$ is high.

For those unfamiliar with the CRU concept, the following is a discussion of how to build a CRU interface. The CRU is a bit addressable (4096 bits), synchronous, serial interface over which a single instruction can transfer between one and 16 bits serially. Each one of the 4096 bits of the CRU space has a unique address and can be read and written to. During multi-bit CRU transfers, the CRU address is incremented at the beginning of each CRU cycle to point to the next consecutive CRU bit.

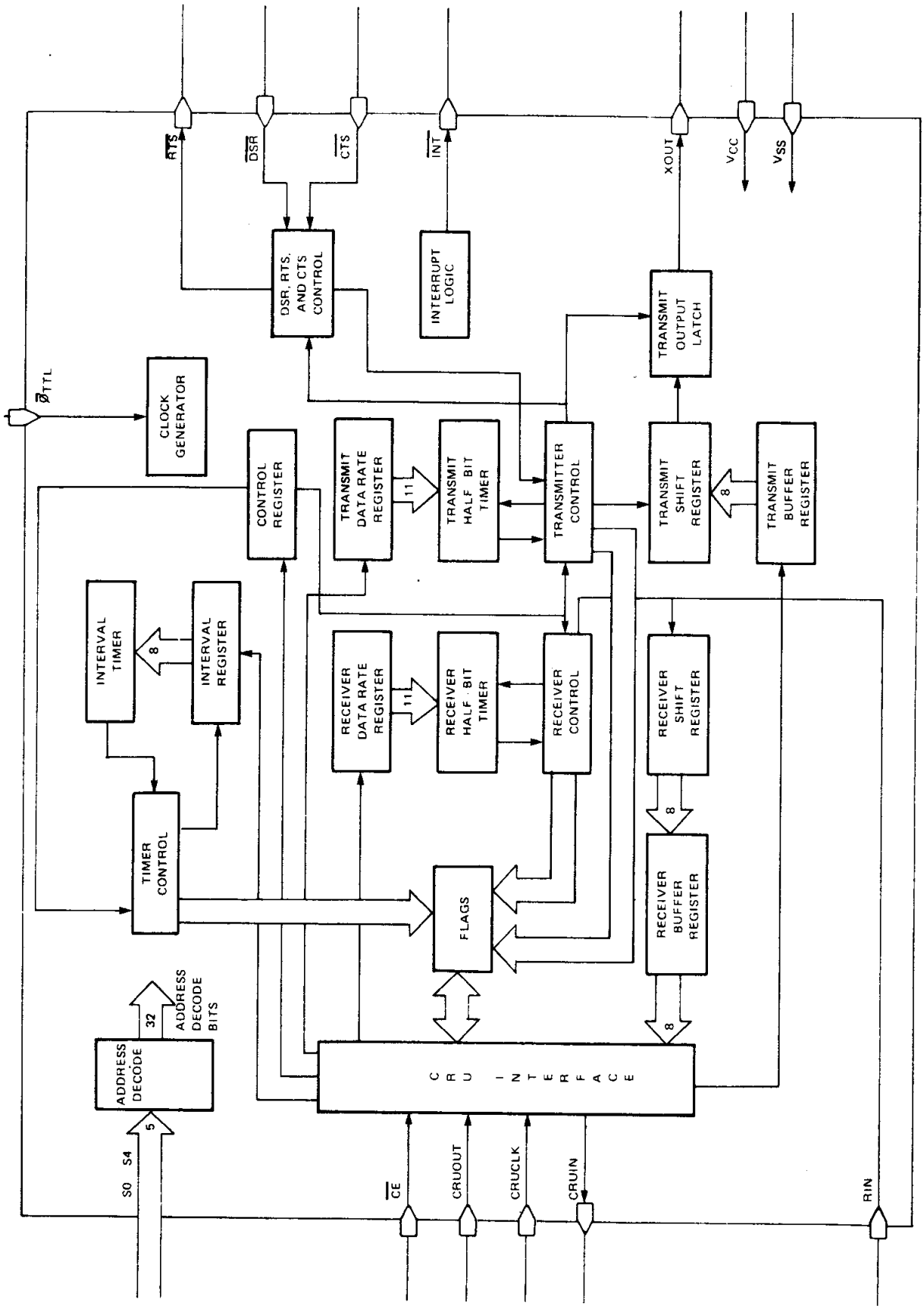


FIGURE 2. TMS 9902 ASYNCHRONOUS COMMUNICATIONS CONTROLLER (ACC) BLOCK DIAGRAM

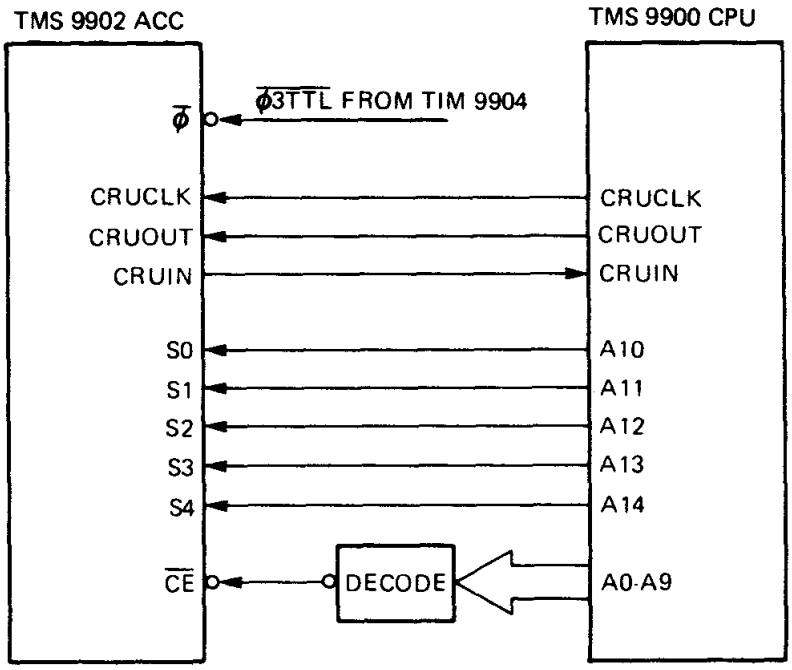


FIGURE 3. TMS 9902 – TMS 9900 CRU INTERFACE

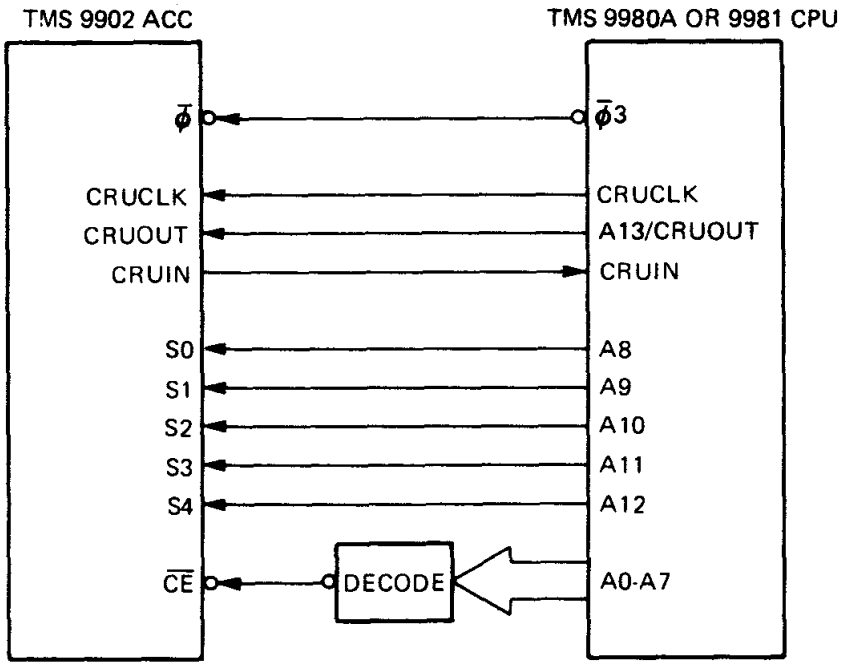


FIGURE 4. TMS 9902 – TMS 9980A OR 9981 CRU INTERFACE

When a 99XX CPU executes a CRU Instruction, the processor uses the contents of workspace register 12 as a base address. (Refer to the 9900 Microprocessor Data Manual for a complete discussion on how CRU addresses are derived.) The CRU address is brought out on the 15-bit address bus; this means that the least significant bit of R12 is not brought out of the CPU. During CRU cycles, the memory control lines ($\overline{MEMEN}$, $\overline{WE}$, and DBIN) are all inactive; $\overline{MEMEN}$ being inactive (HIGH) indicates the address is not a memory address and therefore is a CRU address or external instruction code. Also, when $\overline{MEMEN}$ is inactive (HIGH) and a valid address is present, address bits A0-A2 must all be zero to constitute a valid CRU address; if address bits A0-A2 are other than all zeros, they are indicating an external instruction code. In summary, address bits A3-A14 contain the CRU address to be decoded, address bits A0-A2 must be zero and $\overline{MEMEN}$ must be inactive (HIGH) to indicate a CRU cycle.

2.1.1 CPU OUTPUT FOR CRU

The TMS 9902 ACC occupies 32 bits of output CRU space, of which 23 bits are used: 31 and 21-0. These 23 bits are employed by the CPU to communicate command and control information to the TMS 9902. Table 1 shows the mapping between CRU address select (S lines) and ACC functions. Each CRU addressable output bit on the TMS 9902 is described in detail following Table 1.

TABLE 1
TMS 9902 ACC OUTPUT SELECT BIT ASSIGNMENTS

$ADDRESS_2$ S0	S1	S2	S3	S4	$ADDRESS_{10}$	NAME	DESCRIPTION
1	1	1	1	1	31	RESET	Reset device.
					30-22		Not used.
1	0	1	0	1	21	DSCENB	Data Set Status Change Interrupt Enable.
1	0	1	0	0	20	TIMENB	Timer Interrupt Enable
1	0	0	1	1	19	XBIENB	Transmitter Interrupt Enable
1	0	0	1	0	18	RIENB	Receiver Interrupt Enable
1	0	0	0	1	17	BRKON	Break On
1	0	0	0	0	16	RTSON	Request to Send On
0	1	1	1	1	15	TSTMD	Test Mode
0	1	1	1	0	14	LDCTRL	Load Control Register
0	1	1	0	1	13	LDIR	Load Interval Register
0	1	1	0	0	12	LRDR	Load Receiver Data Rate Register
0	1	0	1	1	11	LXDR	Load Transmit Data Rate Register
					10-0		Control, Interval, Receive Data Rate, Transmit Data Rate, and Transmit Buffer Registers

Bit 31 (RESET) — **Reset.** Writing a one or zero to bit 31 causes the device to reset, consequently disabling all interrupts, initializing the transmitter and receiver, setting $\overline{RTS}$ inactive (HIGH), setting all register load control flags (LDCTRL, LDIR, LRDR, and LXDR) to a logic one level, and resetting the BREAK flag. No other input or output operations should be performed for 11 $\overline{\phi}$ clock cycles after issuing the RESET command.

Bit 30-Bit 22 — Not used.

INTERRUPT ENABLE	SELECT BIT	INTERRUPT FLAG	INTERRUPT ENABLED
DSCENB	21	DSCH	DSCINT
TIMENB	20	TIMELP	TIMINT
XIENB	19	XBRE	XINT
RIENB	18	RBRL	RINT

Bit 21 (DSCENB) — **Data Set Change Interrupt Enable.** Writing a one to bit 21 causes the $\overline{INT}$ output to be active (LOW) whenever DSCH (Data Set Status Change) is a logic one. Writing a zero to bit 21 causes DSCH interrupts to be disabled. Writing either a one or zero to bit 21 causes DSCH to reset. (Refer also to Section 2.5).

Bit 20 (TIMENB) — **Timer Interrupt Enable.** Writing a one to bit 20 causes the $\overline{INT}$ output to be active whenever TIMELP (Timer Elapsed) is a logic one. Writing a zero to bit 20 causes TIMELP interrupts to be disabled. Writing either a one or zero to bit 20 causes TIMELP and TIMERR (Timer Error) to reset. (Refer also to Sections 2.4 and 2.5.)

Bit 19 (XBIENB) — **Transmit Buffer Interrupt Enable.** Writing a one to bit 19 causes the $\overline{INT}$ output to be active whenever XBRE (Transmit Buffer Register Empty) is a logic one. Writing a zero to bit 19 causes XBRE interrupts to be disabled. The state of XBRE is not affected by writing to bit 19. (Refer also to Sections 2.2 and 2.5.)

Bit 18 (RIENB) — **Receiver Interrupt Enable.** Writing a one to bit 18 causes the $\overline{INT}$ output to be active whenever RBRL (Receiver Buffer Register Loaded) is a logic one. Writing a zero to bit 18 disables RBRL interrupts. Writing either a one or zero to bit 18 causes RBRL to reset. (Refer also to Sections 2.3 and 2.5.)

Bit 17 (BRKON) — **Break On.** Writing a one to bit 17 causes the XOUT (Transmitter Serial Data Output) to go to a logic zero whenever the transmitter is active and the Transmit Buffer Register (XBR) and the Transmit Shift Register (XSR) are empty. While BRKON is set, loading of characters into the XBR is inhibited. Writing a zero to bit 17 causes BRKON to reset and the transmitter to resume normal operation.

Bit 16 (RTSON) — **Request To Send On.** Writing a one to bit 16 causes the $\overline{RTS}$ output to be active (LOW). Writing a zero to bit 16 causes $\overline{RTS}$ to go to a logic one after the XSR (Transmit Shift Register) and XBR (Transmit Buffer Register) are empty, and BRKON is reset. Thus, the $\overline{RTS}$ output does not become inactive (HIGH) until *after* character transmission is completed.

Bit 15 (TSTMD) — **Test Mode.** Writing a one to bit 15 causes $\overline{RTS}$ to be internally connected to $\overline{CTS}$, XOUT to be internally connected to RIN, $\overline{DSR}$ to be internally held LOW, and the Interval Timer to operate 32 times its normal rate. Writing a zero to bit 15 re-enables normal device operation. There seldom is reason to enter the test mode under normal circumstances, but this function is useful for diagnostic and inspection purposes.

Bits 14-11 — **Register Load Control Flags.** Output bits 14-11 control which of the five registers are loaded when writing to bits 10-0. The flags are prioritized as shown in Table 2.

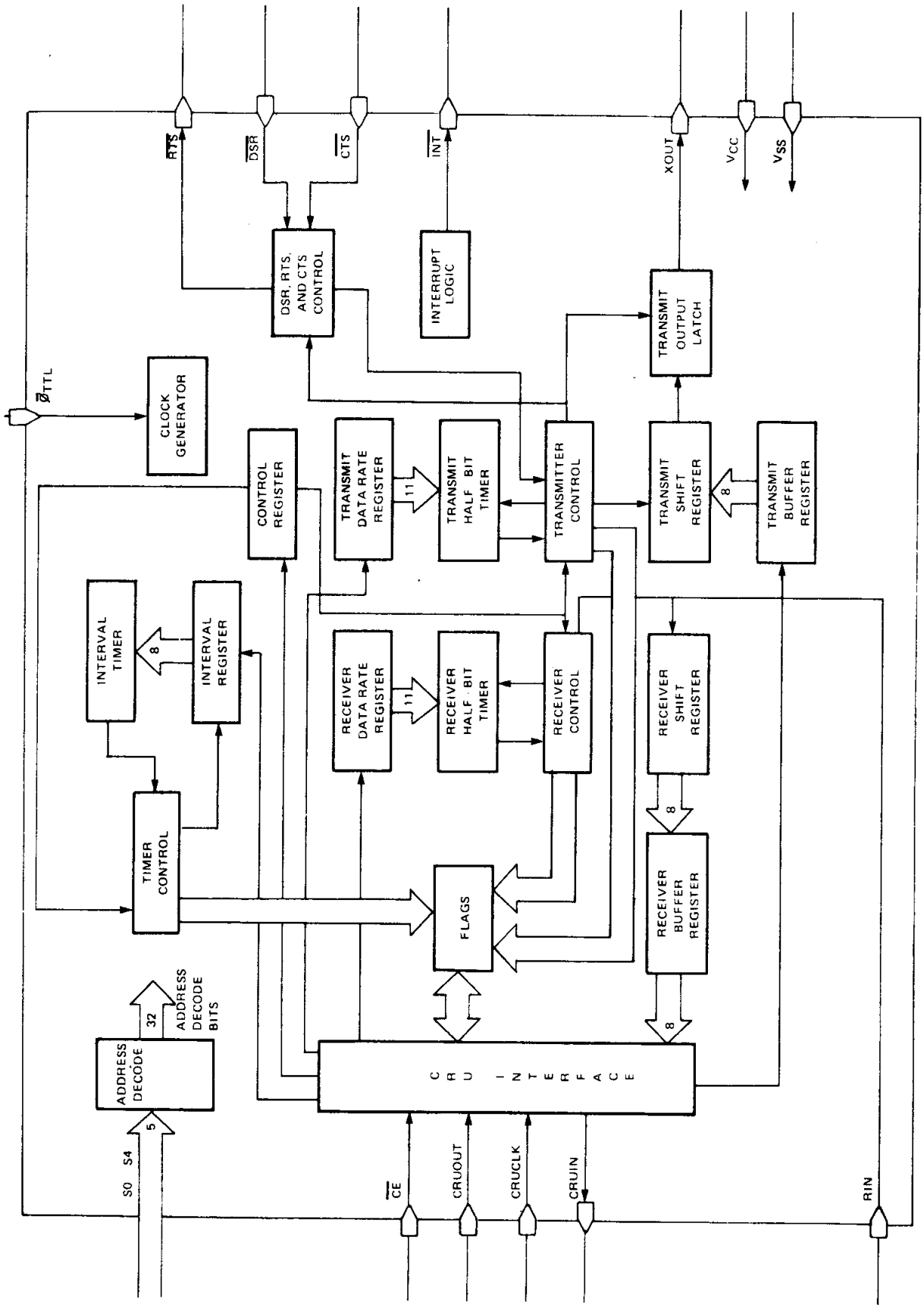


FIGURE 2. TMS 9902 ASYNCHRONOUS COMMUNICATIONS CONTROLLER (ACC) BLOCK DIAGRAM

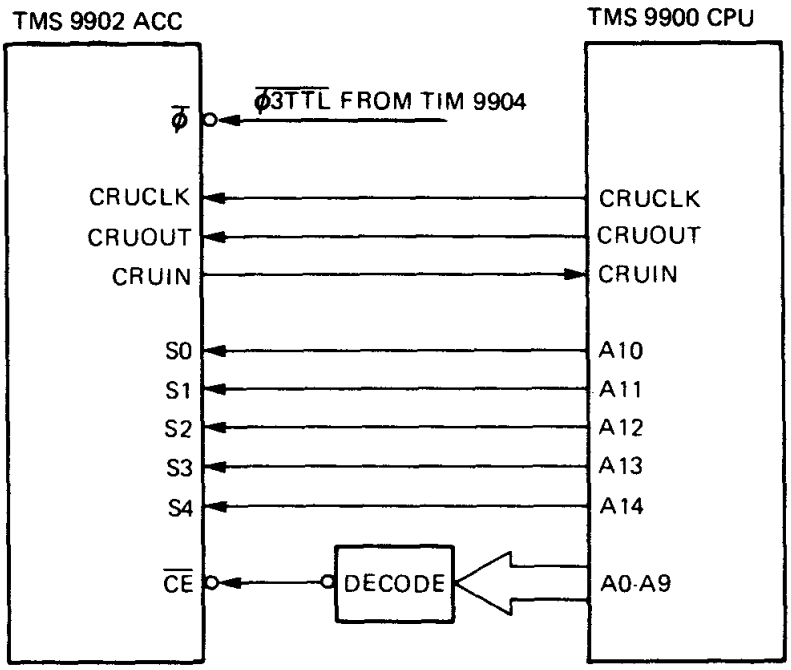


FIGURE 3. TMS 9902 – TMS 9900 CRU INTERFACE

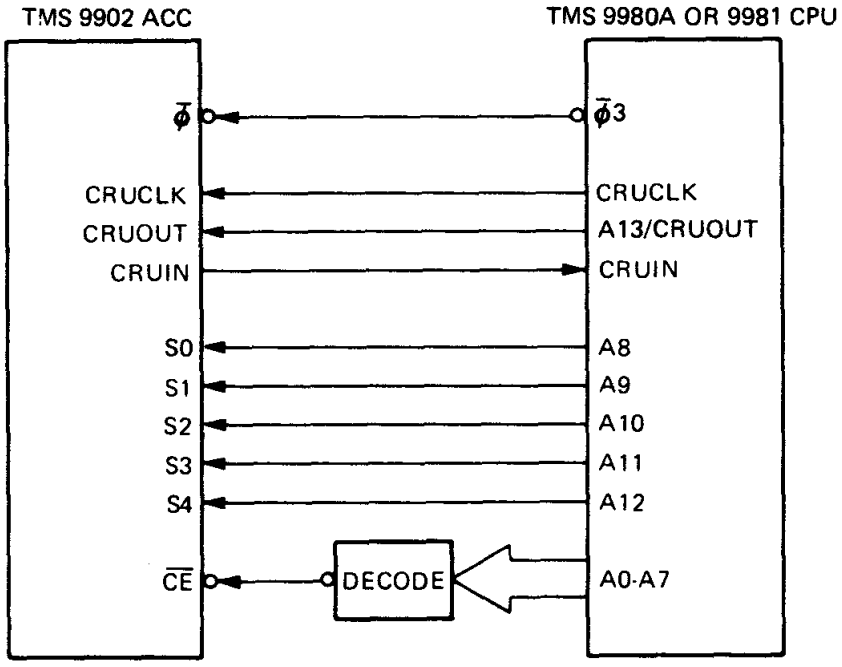


FIGURE 4. TMS 9902 – TMS 9980A OR 9981 CRU INTERFACE

TABLE 2

TMS 9902 ACC REGISTER LOAD SELECTION

REGISTER LOAD CONTROL FLAG STATUS				REGISTER ENABLED
LDCTRL	LDIR	LRDR	LXDR	
1	X	X	X	Control Register
0	1	X	X	Interval Register
0	0	1	X	Receive Data Rate Register *
0	0	X	1	Transmit Data Rate Register *
0	0	0	0	Transmit Buffer Register

*If both LRDR and LXDR bits are set, both registers are loaded, assuming LDCTRL and LDIR are disabled; if only one of these registers is to be loaded, only that register bit is set, and the other register bit reset.

Bit 14 (LDCTRL) — **Load Control Register.** Writing a one to bit 14 causes LDCTRL to be set to a logic one. When LDCTRL = 1, any data written to bits 0-7 is directed to the Control Register. Note that LDCTRL is also set to a logic one when a one or zero is written to bit 31 (RESET). Writing a zero to bit 14 causes LDCTRL to reset to a logic zero, disabling loading of the Control Register. LDCTRL is also automatically reset to logic zero when a datum is written to bit 7 of the Control Register, reset normally occurs as the last bit is written when loading the Control Register with a LDCR instruction.

Bit 13 (LDIR) — **Load Interval Register.** Writing a one to bit 13 causes LDIR to set to a logic one. When LDIR = 1 and LDCTRL = 0, any data written to bits 0-7 is directed to the Interval Register. Note that LDIR is also set to a logic one when a datum is written to bit 31 (RESET); however, Interval Register loading is not enabled until LDCTRL is set to a logic zero. Writing a zero to bit 13 causes LDIR to be reset to logic zero, disabling loading of the Interval Register. LDIR is also automatically reset to logic zero when a datum is written to bit 7 of the Interval Register; reset normally occurs as the last bit is written when loading the Interval Register with a LDCR instruction.

Bit 12 (LRDR) — **Load Receive Data Rate Register.** Writing a one to bit 12 causes LRDR to set to a logic one. When LRDR = 1, LDIR = 0, and LDCTRL = 0, any data written to bits 0-10 is directed to the Receive Data Rate Register. Note that LRDR is also set to a logic one when a datum is written to bit 31 (RESET); however, Receive Data Rate Register loading is not enabled until LDCTRL and LDIR are set to a logic zero. Writing a zero bit to 12 causes LRDR to reset to a logic zero, disabling loading of the Receive Data Rate Register. LRDR is also automatically reset to logic zero when a datum is written to bit 10 of the Receive Data Rate Register; reset normally occurs as the last bit is written when loading the Receive Data Rate Register with a LDCR instruction.

Bit 11 (LXDR) — **Load Transmit Data Rate Register.** Writing a one to bit 11 causes LXDR to set to a logic one. When LXDR = 1, LDIR = 0, and LDCTRL = 0, any data written to bits 0-10 is directed to the Transmit Data Rate Register. Note that loading of both the Receive and Transmit Data Rate Registers is enabled when LDCTRL = 0, LDIR = 0, LRDR = 1, and LXDR = 1; thus these two registers may be loaded simultaneously when data is received and transmitted at the same rate. LXDR is also set to a logic one when a datum is written to bit 31 (RESET); however, Transmit Data Rate Register loading is not enabled until LDCTRL and LDIR are to logic zero. Writing a zero to bit 11 causes LXDR to reset to logic zero, consequently disabling loading of the Transmit Data Rate Register. Since bit 11 is the next bit addressed after loading the Transmit Data Rate Register, the register may be loaded and the LXDR flag reset with a single LDCR instruction where 12 bits (Bits 0-11) are written and a zero is written to Bit 11.

Bits 14-11 (All Zeros) — **Load Transmit Buffer Register.** See Section 2.1.2.5.

Bits 10-0 (Data) — **Data.** Information written to bits 10-0 is loaded into the controlling registers as indicated by LDCTRL, LDIR, LRDR, and LXDR (see Table 2). The different register bits are described in Section 2.1.2 below.

2.1.2 REGISTERS

2.1.2.1 Control Register

The Control Register is loaded to select character length, device clock operation, parity, and the number of stop bits for the transmitter; control register loading occurs when LDCTRL is active (see Table 2). Table 3 shows the bit address assignments for the Control Register.

TABLE 3
CONTROL REGISTER BIT ADDRESS ASSIGNMENTS

$ADDRESS_{10}$	NAME	DESCRIPTION
7	SBS1	← Stop Bit Select
6	SBS2	
5	PENB	Parity Enable
4	PODD	Odd Parity Select
3	CLK4M	$\overline{\phi}$ Input Divide Select
2	—	Not Used
1	RCL1	← Character Length Select
0	RCL0	

7	6	5	4	3	2	1	0
SBSI	SBS2	PENB	PODD	CLK4M	NOT USED	RCL1	RCL0
MSB							LSB

Bits 7 and 6 (SBS1 and SBS2) — **Stop Bit Selection.** The number of stop bits to be appended to each transmitter character is selected by bits 7 and 6 of the Control Register as shown below. The receiver only tests for a single stop bit, regardless of the status of bits 7 and 6.

STOP BIT SELECTION

SBS1 BIT 7	SBS2 BIT 6	NUMBER OF TRANSMITTED STOP BITS
0	0	1½
0	1	2
1	0	1
1	1	1

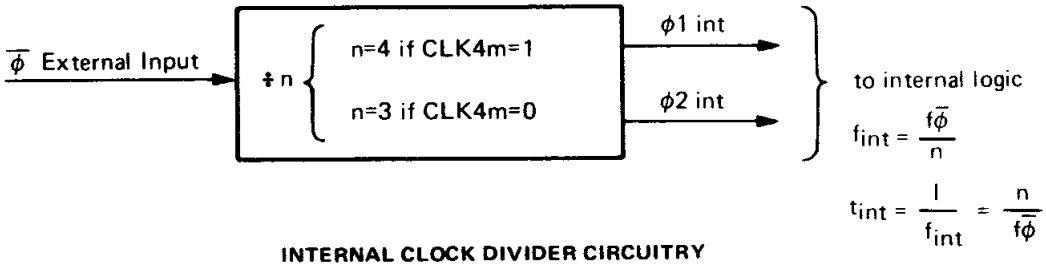
Bits 5 and 4 (PENB and PODD) — **Parity Selection.** The type of parity generated for transmission and detected for reception is selected by bits 5 and 4 of the Control Register as shown below. When parity is enabled (PENB = 1), the parity bit is transmitted and received in addition to the number of bits selected for the character length. Odd parity is such that the total number of ones in the character and parity bit, exclusive of stop bit(s), will be odd. For even parity, the total number of ones will be even.

PARITY SELECTION

PENB BIT 5	PODD BIT 4	PARITY
0	0	None
0	1	None
1	0	Even
1	1	Odd

Bit 3 (CLK4M) —

$\overline{\phi}$ Input Divide Select. The $\overline{\phi}$ input to the TMS 9902 ACC is used to generate internal dynamic logic clocking and to establish the time base for the Interval Timer, Transmitter, and Receiver. The $\overline{\phi}$ input is internally divided by either 3 or 4 to generate the two-phase internal clocks required for MOS logic, and to establish the basic internal operating frequency (f_{int}) and internal clock period (t_{int}). When bit 3 of the Control Register is set to a logic one (CLK4M = 1), $\overline{\phi}$ is internally divided by 4, and when CLK4M = 0, $\overline{\phi}$ is divided by 3. For example, when $f\overline{\phi}$ = 3 MHz, as in a standard 3 MHz TMS 9900 system, and CLK4M = 0, $\overline{\phi}$ is internally divided by 3 to generate an internal clock period t_{int} of 1 μs. The figure below shows the operation of the internal clock divider circuitry. The internal clock frequency should be no greater than 1.1 MHz; thus, when $f\overline{\phi}$ > 3.3 MHz, CLK4M should be set to a logic one.



INTERNAL CLOCK DIVIDER CIRCUITRY

Bits 1 and 0 (RCL1 and RCL0) —

Character Length Select. The number of data bits in each transmitted and received character is determined by bits 1 and 0 of the Control Register as shown below:

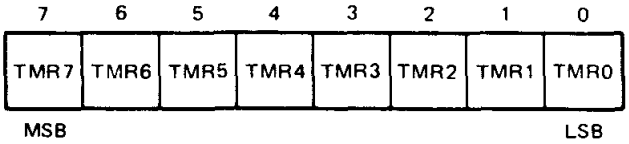
CHARACTER LENGTH SELECTION

RCL1 BIT 1	RCL0 BIT 0	CHARACTER LENGTH
0	0	5 Bits
0	1	6 Bits
1	0	7 Bits
1	1	8 Bits

NOTE: $f\phi$ denotes frequency of ϕ

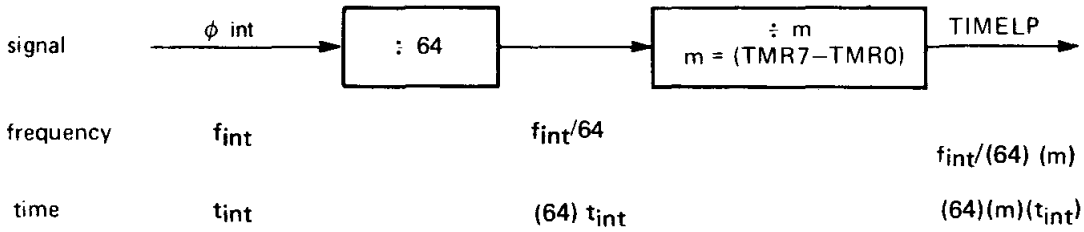
2.1.2.2 Interval Register

The Interval Register is enabled for loading when LDCTRL = 0 and LDIR = 1 (see Table 2). The Interval Register is used to select the rate at which interrupts are generated by the TMS 9902 Interval Timer. The figure below shows the bit assignments for the Interval Register when enabling for loading.



INTERVAL REGISTER BIT ADDRESS ASSIGNMENTS

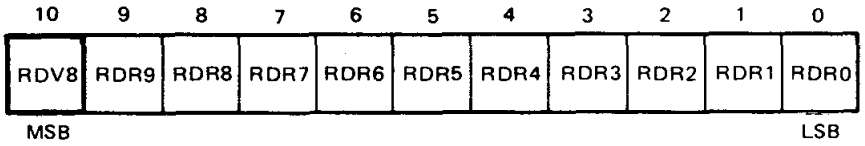
The figure below illustrates the establishment of the interval for the Interval Timer. For example, if the Interval Register is loaded with a value of 80_{16} (128_{10}) the interval at which Timer Interrupts are generated is $t_{itvl} = t_{int} \cdot 64 \cdot M = (1\ \mu s)(64)(128) = 8.192$ ms when $t_{int} = 1\ \mu s$. $t_{int} = n/f\overline{\phi}$ where n = 4 if CLK4M = 1, 3 if CLK4M = 0.



TIME INTERVAL SELECTION

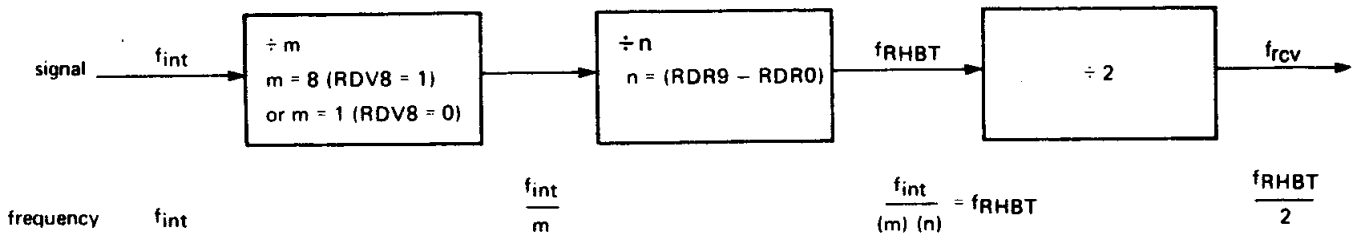
2.1.2.3 Receive Data Rate Register

The Receive Data Rate Register (RDR) is enabled for loading when LDCTRL = 0, LDIR = 0, and LRDR = 1 (see Table 2). The Receive Data Rate Register is used to select the bit rate at which data is received. The diagram shows the bit address assignments for the Receive Data Rate Register when enabled for loading.



RECEIVE DATA RATE REGISTER BIT ADDRESS ASSIGNMENTS

The diagram below illustrates the manner in which the receive data rate is established. Basically, two programmable counters are used to determine the interval for half the bit period of receive data. The first counter divides the internal system clock frequency (f_{int}) by either 8 (RDV8 = 1) or 1 (RDV8 = 0). The second counter has ten stages and may be programmed to divide its input signal by any value from 1 (RDR9 – RDR0 = 0000000001) to 1023 (RDR9 – RDR0 = 1111111111). The frequency of the output of the second counter (f_{rhbt}) is double the receive-data rate. For example, assume the Receive Data Rate Register is loaded with a value of 1100111000; RDV8 = 1, and RDR9 – RDR0 = $1000111000 = 238_{16} = 568_{10}$. Thus, for f_{int} = 1 MHz, (see Control Register, bit 3) the receive data rate = $f_{rcv} = [(1 \times 10^6 \div 8) \div 568] \div 2 = 110.04$ bits per second.



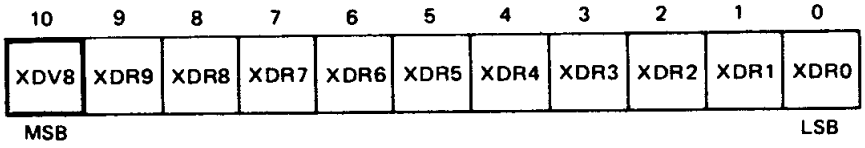
RECEIVE DATA RATE SELECTION

Quantitatively, the receive-data rate f_{RCV} is described by the following algebraic expression:

$$f_{rcv} = \frac{f_{RHBT}}{2} = \frac{f_{int}}{(2)\ (m)\ (n)} = \frac{f_{int}}{(2)\ (8^{RDV8})\ (RDR9 - RDR0)}$$

2.1.2.4 Transmit Data Rate Register

The Transmit Data Rate Register (XDR) is enabled for loading when LDCTRL = 0, LDIR = 0, and LXDR = 1 (see Table 2). The Transmit Data Rate Register is used to select the data for the transmitter. The figure below shows the bit address assignments for the Transmit Data Rate Register when enabled for loading.



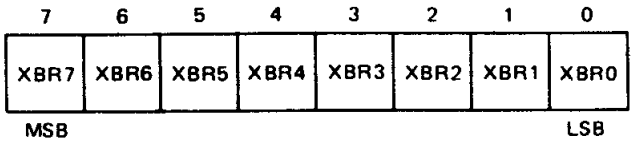
The transmit data rate is selected with the Transmit Data Rate Register in the same manner the receive data rate is selected with the Receive Data Rate Register. The algebraic Expression for the Transmit Data Rate f_{xmt} is

$$f_{xmt} = \frac{f_{XHBT}}{2} = \frac{f_{int}}{(2)\ (8^{XDV8})\ (XDR9\text{-}XDR0)}$$

For example, if the Transmit Data Rate Register is loaded with a value of 00110100001; XDV8 = 0, and XDR9 – XDR0 = $1A1_{16} = 417_{10}$, if f_{int} = 1 MHz the transmit data rate = f_{xmt} = [(1 × 10^6 ÷ 1) ÷ 417] ÷ 2 = 1199.0 bits per second.

2.1.2.5 Transmit Buffer Register

The Transmit Buffer Register (XBR) is enabled for loading when LDCTRL = 0, LDIR = 0, LRDR = 0, LXDR = 0, and BRKON = 0 (see Table 2). The Transmit Buffer Register is used to store the next character to be transmitted. When the transmitter is active, the contents of the Transmit Buffer Register are transferred to the Transmit Shift Register (XSR) each time the previous character has been completely transmitted (XSR becomes empty). The bit address assignments for the Transmit Buffer Register are shown below:



TRANSMIT BUFFER REGISTER BIT ADDRESS ASSIGNMENTS

TABLE 4. CRU OUTPUT BIT ADDRESS ASSIGNMENTS

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RESET	NOT USED									DSCENB	TIMENB	XBIENB	RIENB	BRKON	RTSON

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TSTMD	LDCTRL	LDIR	LRDR	LXDR	CONTROL, INTERVAL, RECEIVE DATA RATE, TRANSMIT DATA RATE, AND TRANSMIT BUFFER REGISTERS										

CONTROL REGISTER

14	13	12	11	7	6	5	4	3	2	1	0
1	X	X	X	SBS1	SBS2	PENB	PODD	CLK4M	–	RCL1	RCL0

Stop Bits		Parity		f_{int} =	Character Length	
00	1-1/2	0X	none		00	5
01	2	10	even	$f\overline{\phi}/(3+CLK4M)$	01	6
1X	1	11	odd		10	7
					11	8

INTERVAL REGISTER

14	13	12	11	7	6	5	4	3	2	1	0
0	1	X	X	TMR7	TMR 6	TMR5	TMR4	TMR3	TMR2	TMR1	TMR0

TMR

$T_{ITVL} = t_{int}$ X 64 X TMR

RECEIVE DATA RATE REGISTER

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	1	X	RDV8	RDR9	RDR8	RDR7	RDR6	RDR5	RDR4	RDR3	RDR2	RDR1	RDR0

RDR

$f_{rcv} = f_{int} \div 8$ RDV8 $\div$ RDR $\div 2$

TRANSMIT DATA RATE REGISTER

14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	X	1	XDV8	XDR9	XDR8	XDR7	XDR6	XDR5	XDR4	XDR3	XDR2	XDR1	XDR0

XDR

$f_{xmt} = f_{int} \div 8$ XDV8 $\div$ XDR $\div 2$

TRANSMIT BUFFER REGISTER

14	13	12	11	7	6	5	4	3	2	1	0
0	0	0	0	XBR7	XBR6	XBR5	XBR4	XBR3	XBR2	XBR1	XBR0

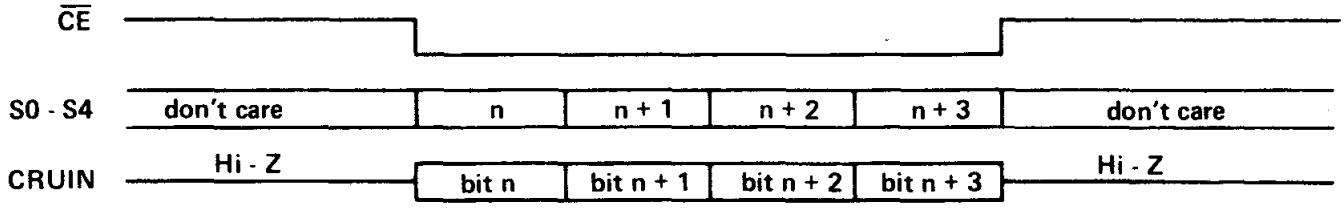
NOTE 1 LOADING OF THE BIT INDICATED BY ▨ CAUSES THE LOAD CONTROL FLAG FOR THAT REGISTER TO RESET AUTOMATICALLY.

All eight bits should be transferred into the register, regardless of the selected character length. The extraneous high order bits will be ignored for transmission purposes; however, loading of bit 7 is internally detected which causes the Transmit Buffer Register Empty (XBRE) status flag to reset.

2.1.3 INPUT TO CPU FOR CRU

The TMS 9902 ACC occupies 32 bits of input CRU space. The CPU reads the 32 bits from the ACC to sense the status of the device. Table 5 shows the mapping between CRU bit address and TMS 9902 read data. Each CRU addressable read bit is described following Table 5.

Status and data information is read from the ACC using $\overline{CE}$, S0-S4, and CRUIN. The following figure illustrates the relationship of the signals used to access four bits of data from the ACC.



ACC DATA ACCESS SIGNAL TIMING

TABLE 5
TMS 9902 ACC INPUT SELECT BIT ASSIGNMENTS

$ADDRESS_2$					$ADDRESS_{10}$	NAME	DESCRIPTION
S0	S1	S2	S3	S4			
1	1	1	1	1	31	INT	Interrupt
1	1	1	1	0	30	FLAG	Register Load Control Flag Set
1	1	1	0	1	29	DSCH	Data Set Status Change
1	1	1	0	0	28	CTS	Clear to Send
1	1	0	1	1	27	DSR	Data Set Ready
1	1	0	1	0	26	RTS	Request to Send
1	1	0	0	1	25	TIMELP	Timer Elapsed
1	1	0	0	0	24	TIMERR	Timer Error
1	0	1	1	1	23	XSRE	Transmit Shift Register Empty
1	0	1	1	0	22	XBRE	Transmit Buffer Register Empty
1	0	1	0	1	21	RBRL	Receive Buffer Register Loaded
1	0	1	0	0	20	DSCINT	Data Set Status Change Interrupt (DSCH • DSCENB)
1	0	0	1	1	19	TIMINT	Timer Interrupt (TIMELP • TIMENB)
1	0	0	1	0	18	–	Not Used (always = 0)
1	0	0	0	1	17	XBINT	Transmitter Interrupt (XBRE • XBIENB)
1	0	0	0	0	16	RBINT	Receiver Interrupt (RBRL • RIENB)
0	1	1	1	1	15	RIN	Receive Input
0	1	1	1	0	14	RSBD	Receive Start Bit Detect
0	1	1	0	1	13	RFBD	Receive Full Bit Detect
0	1	1	0	0	12	RFER	Receive Framing Error
0	1	0	1	1	11	ROVER	Receive Overrun Error
0	1	0	1	0	10	RPER	Receive Parity Error
0	1	0	0	1	9	RCVERR	Receive Error
0	1	0	0	0	8	–	Not Used (always = 0)
					7-0	RBR7 - RBR0	Receive Buffer Register (Received Data)

Bit 31 (INT) — INT = DSCINT (Data Set Status Change Interrupt) + TIMINT (Timer Interrupt) + XBINT (Transmitter Interrupt) + RBINT (Receiver Interrupt). The interrupt output ($\overline{INT}$) is active (LOW) when this status signal is a logic one. (Refer also to Section 2.6.)

Bit 30 (FLAG) — FLAG = LDCTRL + LDIR + LRDR + LXDR + BRKON. When any of the register load control flags or BRKON is set, FLAG = 1 (see Section 2.1.1).

Bit 29 (DSCH) — **Data Set Status Change.** DSCH is set when the $\overline{DSR}$ or $\overline{CTS}$ input changes state. To ensure recognition of the state change, $\overline{DSR}$ or $\overline{CTS}$ must remain stable in its new state for a minimum of two internal clock cycles. DSCH is reset by an output to bit 21 (DSCENB).

Bit 28 (CTS) — **Clear To Send.** The CTS signal indicates the inverted status of the $\overline{CTS}$ device input.

Bit 27 (DSR) — **Data Set Ready.** The DSR signal indicates the inverted status of the $\overline{DSR}$ device input.

Bit 26 (RTS) — **Request To Send.** The RTS signal indicates the inverted status of the $\overline{RTS}$ device output.

Bit 25 (TIMELP) — **Timer Elapsed.** TIMELP is set each time the Interval Timer decrements to 0. TIMELP is reset by an output to bit 20 (TIMENB).

Bit 24 (TIMERR) — **Timer Error.** TIMERR is set whenever the Interval Timer decrements to 0 and TIMELP (Timer Elapsed) is already set, indicating that the occurrence of TIMELP was not recognized and cleared by the CPU before subsequent intervals elapsed. TIMERR is reset by an output to bit 20 (TIMENB, Timer Interrupt Enable).

Bit 23 (XSRE) — **Transmit Shift Register Empty.** When XSRE = 1, no data is currently being transmitted and the XOUT output is at logic one unless BRKON (see Section 2.1.1) is set. When XSRE = 0, transmission of data is in progress.

Bit 22 (XBRE) — **Transmit Buffer Register Empty.** When XBRE = 1, the transmit buffer register does not contain the next character to be transmitted. XBRE is set each time the contents of the transmit buffer register are transferred to the transmit shift register, XBRE is reset by an output to bit 7 of the transmit buffer register (XBR7), indicating that a character has been loaded.

Bit 21 (RBRL) — **Receive Buffer Register Loaded.** RBRL is set when a complete character has been assembled in the receive shift register, and the character is transferred to the receive buffer register. RBRL is reset by an output to bit 18 (RIENB, Receiver Interrupt Enable).

Bit 20 (DSCINT) — **Data Set Status Change Interrupt.** DSCINT = DSCH (Data Set Status Change)AND DSCENB(Data Set Status Change Interrupt Enable). DSCINT indicates the presence of an enabled interrupt caused by the changing of state of $\overline{DSR}$ or $\overline{CTS}$.

Bit 19 (TIMINT) — **Timer Interrupt.** TIMINT = TIMELP (Timer Elapsed)AND TIMENB (Timer Interrupt Enable). TIMINT indicates the presence of an enabled interrupt caused by the interval timer.

Bit 17 (XBINT) — **Transmitter Interrupt.** XBINT = XBRE (Transmit Buffer Register Empty) AND XBIENB (Transmit Buffer Interrupt Enable). XBINT indicates the presence of an enabled interrupt caused by the transmitter.

Bit 16 (RBINT) — **Receiver Interrupt.** RBINT = RBRL (Receive Buffer Register Loaded) AND RIENB (Receiver Interrupt Enable). RBINT indicates the presence of an enabled interrupt caused by the receiver.

Bit 15 (RIN) — **Receive Input.** RIN indicates the status of the RIN input to the device.

Bit 14 (RSBD) — **Receive Start Bit Detect.** RSBD is set a half bit time after the 1-to-0 transition of RIN, indicating the start bit of a character. If RIN is not still 0 at such time, RSBD is reset. Otherwise, RSBD remains true until the complete character has been received. This bit is normally used only for testing purposes.

Bit 13 (RFBD) — **Receive Full Bit Detect.** RFBD is set one bit time after RSBD is set to indicate the sample point for the first data bit of the received character. RSBD is reset when the character has been completely received. This bit is normally used only for testing purposes.

Bit 12 (RFER) — **Receive Framing Error.** RFER is set when a character is received in which the stop bit, which should be a logic one, is a logic zero. RFER should only be read when RBRL (Receive Buffer Register Loaded) is a one. RFER is reset when a character with the correct stop bit is received.

Bit 11 (ROVER) — **Receive Overrun Error.** ROVER is set when a new character is received before the RBRL (Receive Buffer Register Loaded) flag is reset, indicating that the CPU failed to read the previous character and reset RBRL before the present character is completely received. ROVER is reset when a character is received and RBRL is 0 when the character is transferred to the receive buffer register.

Bit 10 (RPER) — **Receive Parity Error.** RPER is set when a character is received in which the parity is incorrect. RPER is reset when a character with correct parity is received.

Bit 9 (RCVERR) — **Receive Error.** RCVERR = RFER (Receive Framing Error) + ROVER (Receiver Overrun Error) + RPER (Receive Parity Error). The RCVERR signal indicates the presence of an error in the most recently received character.

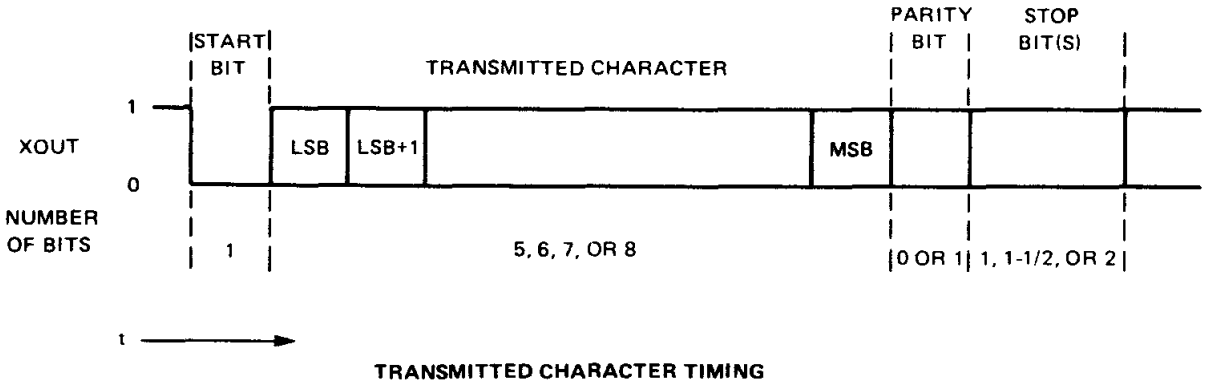
Bit 7-Bit 0 (RBR7-RBR0) — **Receive Buffer Register.** The Receive Buffer Register contains the most recently received character. For character lengths of fewer than eight bits, the character is right-justified, with unused most significant bit(s) all zero(es). The presence of valid data in the Receive Buffer Register is indicated when RBRL (Receive Buffer Register Loaded) is a logic one.

2.2 TRANSMITTER OPERATION

The operation of the transmitter is diagrammed in Figure 5. The transmitter is initialized by issuing the RESET command (output to bit 31), which causes the internal signals XSRE (Transmit Shift Register Empty) and XBRE (Transmit Buffer Register Empty) to set, and BRKON to reset. Device outputs $\overline{RTS}$ and XOUT are set, placing the transmitter in its idle state. When RTSON (Request-to-Send On) is set by the CPU, the $\overline{RTS}$ output becomes active (LOW) and the transmitter becomes active when the $\overline{CTS}$ input goes LOW.

2.2.1 Data Transmission

If the Transmit Buffer Register contains a character, transmission begins. The contents of the Transmit Buffer Register are transferred to the Transmit Shift Register, causing XSRE to reset and XBRE to set. The first bit transmitted (start bit) is always a logic zero. Subsequently, the character is shifted out, LSB first. Only the number of bits specified by RCL1 and RCL0 (character length select) of the Control Register are shifted. If parity is enabled, the correct parity bit is next transmitted. Finally the stop bit(s) selected by SBS1 and SBS0 of the Control Register are transmitted. Stop bits are always logic one. XSRE is set to indicate that no transmission is in progress, and the transmitter again tests XBRE to determine if the CPU has yet loaded the next character. The timing for a transmitted character is shown below.



TRANSMITTED CHARACTER TIMING

2.2.2 BREAK Transmission

The BREAK message is transmitted only if XBRE = 1, $\overline{CTS}$ = 0, and BRKON = 1. After transmission of the BREAK message begins, loading of the Transmit Buffer Register is inhibited and XOUT is reset. When BRKON is reset by the CPU, XOUT is set and normal operation continues. It is important to note that characters loaded into the Transmit Buffer Register are transmitted prior to the BREAK message, regardless of whether or not the character has been loaded into the Transmit Shift Register before BRKON is set. Any character to be transmitted subsequent to transmission of the BREAK message may not be loaded into the Transmit Buffer Register until after BRKON is reset.

2.2.3 Transmission Termination

Whenever XSRE = 1 and BRKON = 0 the transmitter is idle, with XOUT set to one. If RTSON is reset at this time, the $\overline{RTS}$ device output will go inactive (HIGH), disabling further data transmission until RTSON is again set. $\overline{RTS}$ will not go inactive, however, until any characters loaded into the Transmit Buffer Register prior to resetting RTSON are transmitted and BRKON = 0.

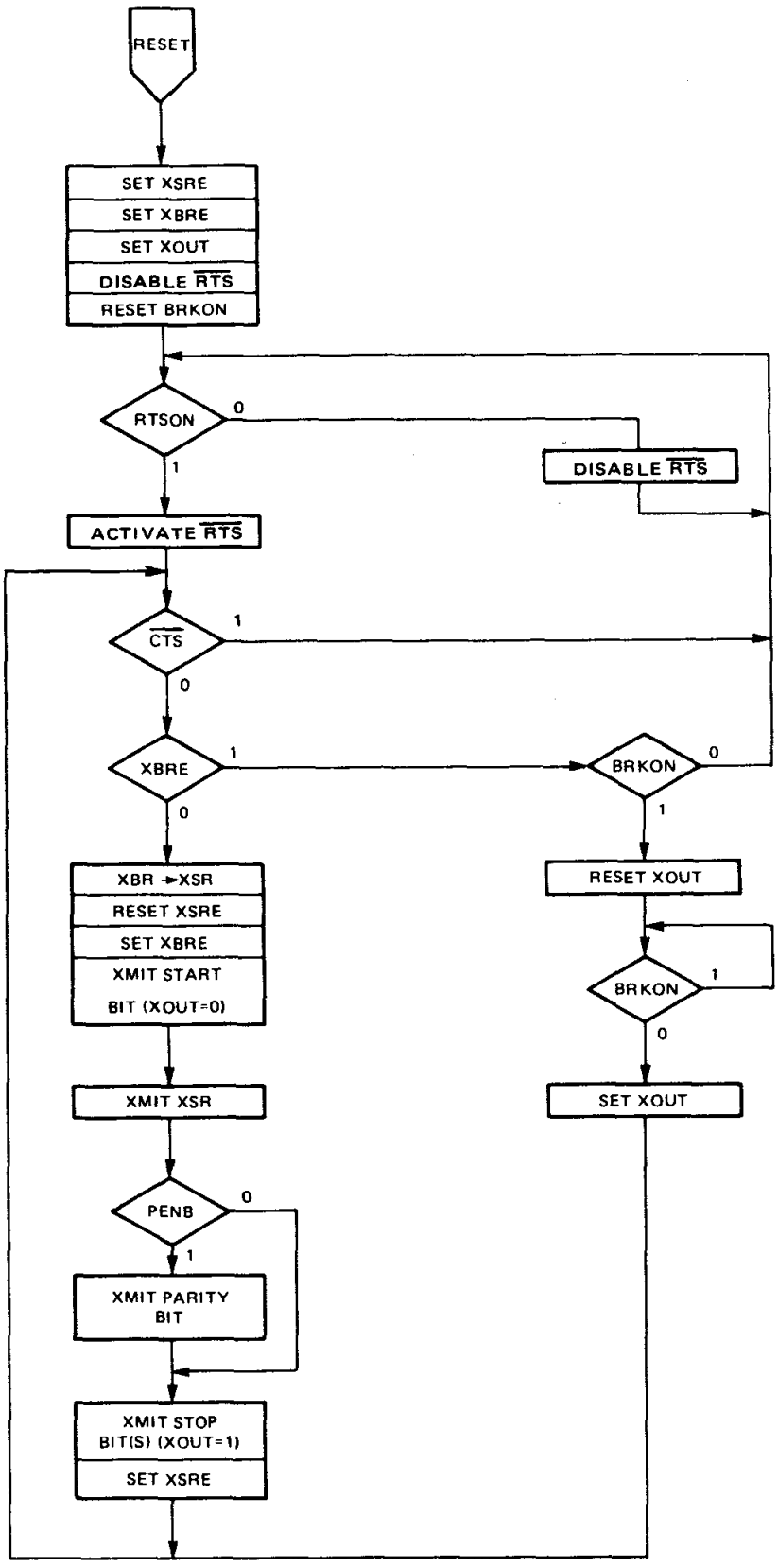


FIGURE 5. TMS 9902 TRANSMITTER OPERATION

2.3 RECEIVER OPERATION

2.3.1 Receiver Initialization

Operation of the TMS 9902 receiver is diagrammed in Figure 6. The receiver is initialized whenever the CPU issues the RESET command. The RBRL (Receive Buffer Register Loaded) flag is reset to indicate that no character is currently in the Receive Buffer Register, and the RSBD (Receive Start Bit Detect) and RFBD (Receive Full Bit Detect) flags are reset. The receiver remains in the inactive state until a one to zero transition is detected on the RIN device input.

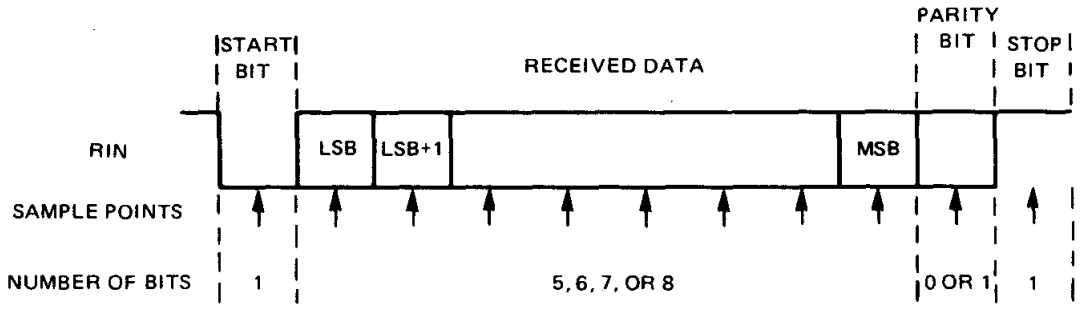
2.3.2 Start Bit Detection

The receiver delays a half bit time and again samples RIN to ensure that a valid start bit has been detected. If RIN = 0 after the half-bit delay, RSBD is set and data reception begins. If RIN = 1, no data reception occurs.

2.3.3 Data Reception

In addition to verifying the valid start bit, the half-bit delay after the one-to-zero transition also establishes the sample point for all subsequent data bits in a valid received character. Theoretically, the sample point is in the center of each bit cell, thus maximizing the limits of acceptable distortion of data cells. After the first full bit delay the least significant data bit is received and RFBD is set. The receiver continues to delay one-bit intervals and sample RIN until the selected number of bits are received. If parity is enabled, one additional bit is read for parity. After an additional bit delay, the received character is transferred to the Receive Buffer Register, RBRL is set, ROVER (Receive Overrun Error) and RPER (Receive Parity Error) are loaded with appropriate values, and RIN is tested for a valid stop bit. If RIN = 1, the stop bit is valid. RFER (Receive Framing Error), RSBD, and RFBD are reset, and the receiver waits for the next start bit to begin reception of the next character.

If RIN = 0 when the stop bit is sampled, RFER is set to indicate the occurrence of a framing error. RSBD and RFBD are reset, but sampling for the start bit of the next character does not begin until RIN = 1. The timing for a received character is depicted below.



RECEIVED CHARACTER TIMING

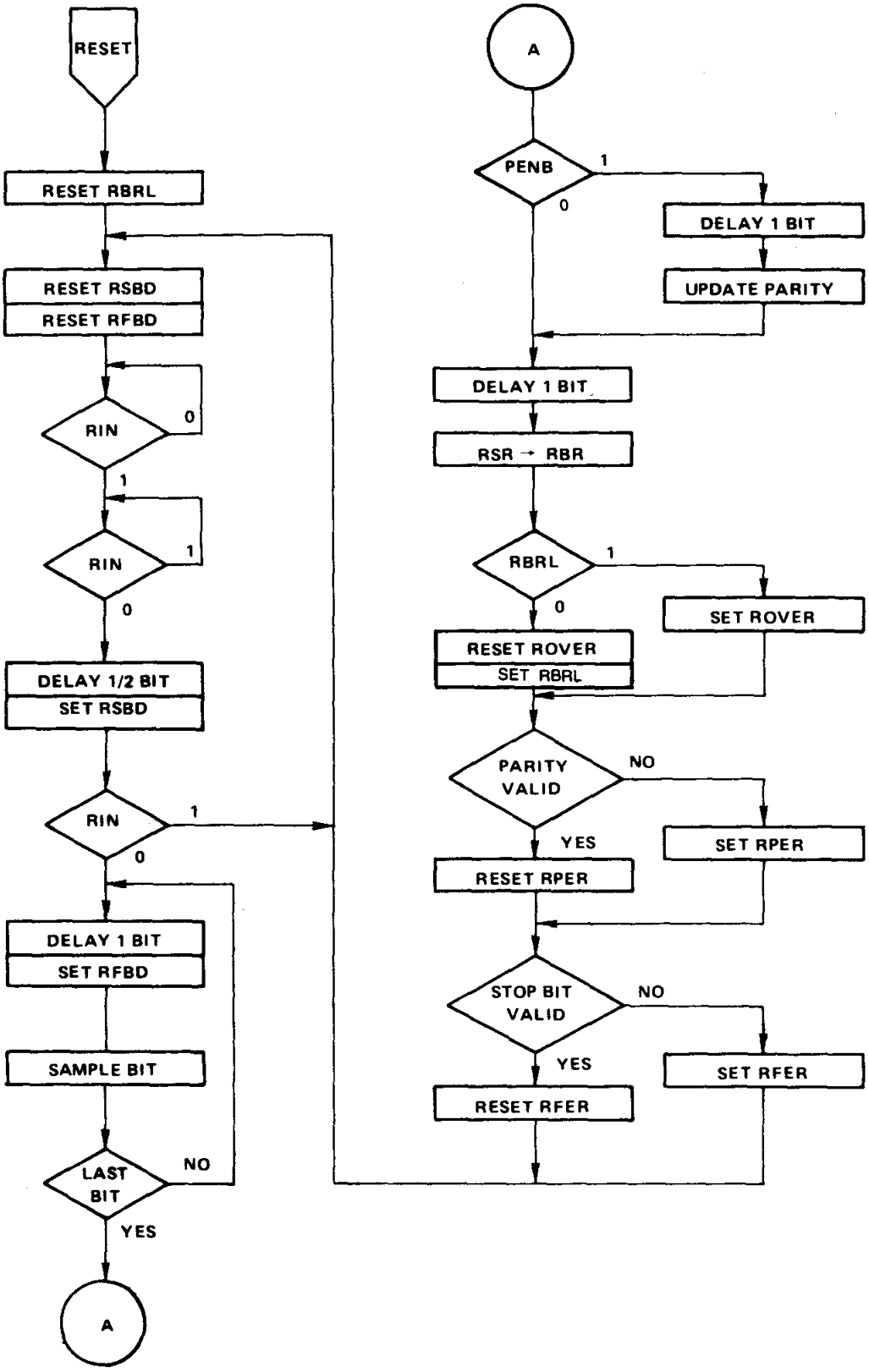


FIGURE 6. TMS 9902 RECEIVER OPERATION

2.4 INTERVAL TIMER OPERATION

A flowchart of the operation of the Interval Timer is shown in Figure 7. Execution of the RESET command by the CPU causes TIMELP (Timer Elapsed) and TIMERR (Timer Error) to reset and LDIR (Load Interval Register) to set. Resetting LDIR causes the contents of the Interval Register to be loaded into the Interval Timer, thus beginning the selected time interval. The timer is decremented every 64 internal clock cycles (every two internal clock cycles when in Test Mode) until it reaches zero, at which time the Interval Timer is reloaded by the Interval Register and TIMELP is set. If TIMELP was already set, TIMERR is set to indicate that TIMELP was not cleared by the CPU before the next time period elapsed. Each time LDIR is reset, the contents of the Interval Register are loaded into the Interval Timer, thus restarting the timer (refer also to Section 2.1.2.2).

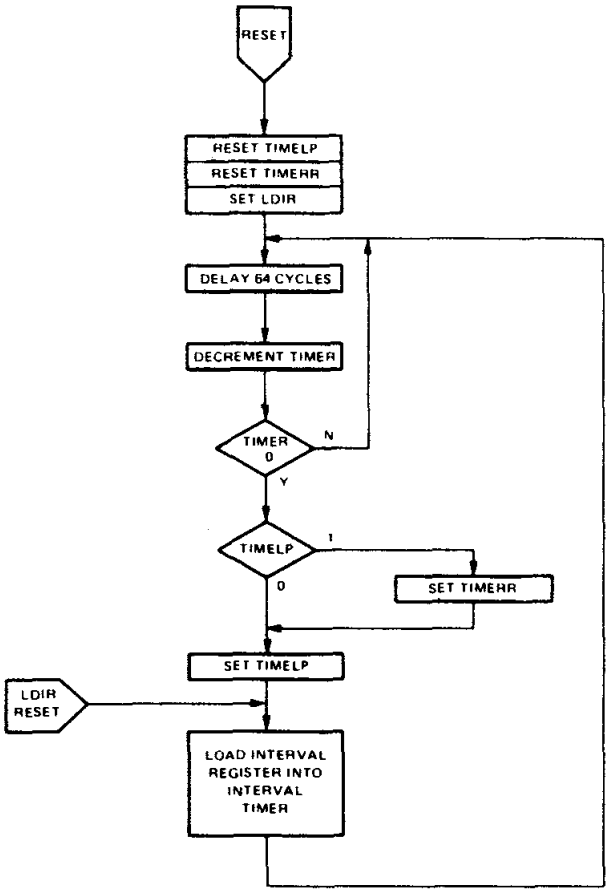


FIGURE 7. TMS 9902 INTERVAL TIMER OPERATION

INTERVAL TIMER SELECTION

2.5 INTERRUPTS

The interrupt output ($\overline{INT}$) is active (LOW) when any of the following conditions occurs and the corresponding interrupt has been enabled on the TMS 9902 by the CPU:

(1) $\overline{DSR}$ or $\overline{CTS}$ changes levels (DSCH = 1);

(2) a character has been received and stored in the Receive Buffer Register (RBRL = 1);

(3) the Transmit Buffer Register is empty (XBRE = 1); or

(4) the selected time interval has elapsed (TIMELP = 1).

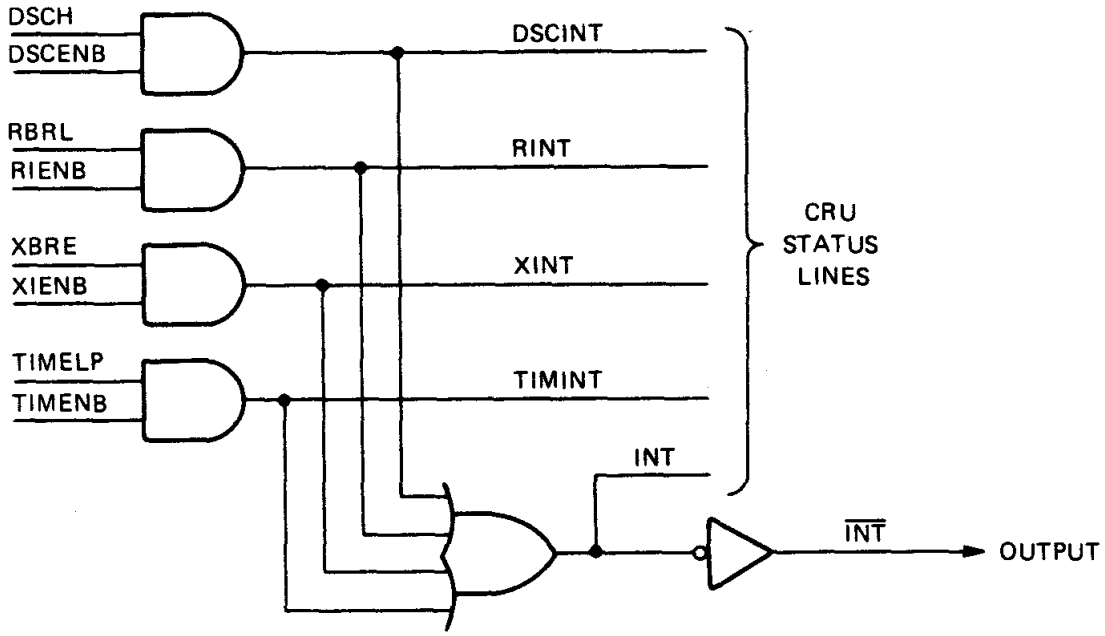


FIGURE 8. $\overline{INT}$ OUTPUT GENERATION

Figure 8 illustrates the logical equivalent of the ACC interrupt section. Table 6 lists the actions necessary to clear those conditions of the TMS 9902 that cause interrupts.

TABLE 6
TMS 9902 INTERRUPT CLEARING

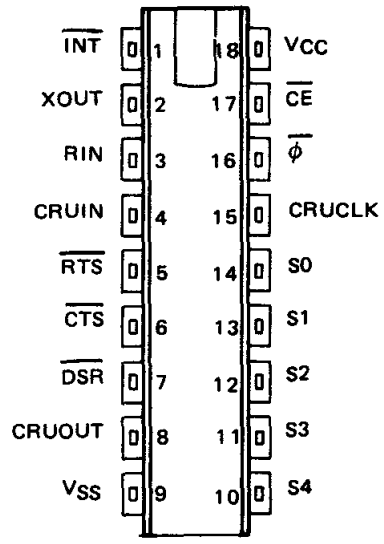
MNEMONIC	CAUSE	ACTION TO RESET
DSCINT	$\overline{CTS}$ or $\overline{DSR}$ change state	Write a bit to DSCENB (bit 21)*
RINT	Recieve Buffer Full	Write a bit to RIENB (bit 18)*
XINT	Transmit Buffer Register Empty	Load Transmit Buffer
TIMINT	Timer Elapsed	Write a bit to TIMENB (bit 20)*

*Writing a zero to clear the interrupt will clear the interrupt and disable further interrupts.

2.6 TMS 9902 TERMINAL ASSIGNMENTS AND FUNCTIONS

SIGNATURE	PIN	I/O	DESCRIPTION
$\overline{INT}$	1	O	Interrupt — when active (LOW), the $\overline{INT}$ output indicates that at least one of the interrupt conditions has occurred.
XOUT	2	O	Transmitter Serial Data Output line — XOUT, remains inactive (HIGH) when TMS 9902 is not transmitting.
RIN	3	I	Receiver Serial Data Input Line — RCV must be held in the inactive (HIGH) state when not receiving data. A transition from HIGH to LOW activates the receiver circuitry.
CRUIN	4	O	Serial data output pin from TMS 9902 to CRUIN input pin of the CPU.
$\overline{RTS}$	5	O	Request-to-Send output from TMS 9902 to modem. $\overline{RTS}$ is enabled by the CPU and remains active (LOW) during transmission from the TMS 9902.
$\overline{CTS}$	6	I	Clear-to-Send input from modem to TMS 9902. When active (LOW), it enables the transmitter section of TMS 9902.
$\overline{DSR}$	7	I	Data Set Ready input from modem to TMS 9902. $\overline{DSR}$ generates an interrupt when it changes state.
CRUOUT	8	I	Serial data input line to TMS 9902 from CRUOUT line of the CPU.
V_{SS}	9	I	Ground reference voltage.
S4 (LSB)	10	I	Address Select Lines. The data bit being accessed by the CPU interface is specified by the 5-bit code appearing on S0-S4.
S3	11	I	
S2	12	I	
S1	13	I	
S0	14	I	
CRUCLK	15	I	CRU Clock. When active (HIGH), indicates valid data on the CRUOUT line for the 9902.
$\overline{\phi}$	16	I	TTL Clock.
$\overline{CE}$	17	I	Chip Enable — when $\overline{CE}$ is inactive (HIGH), TMS 9902 CRU interface is disabled. CRUIN remains at high-impedance when $\overline{CE}$ is inactive (HIGH).
V_{CC}	18	I	Supply voltage (+5 V nominal).

TMS 9902
18-PIN PACKAGE



3. DEVICE APPLICATION

This section describes the software interface between the CPU and the TMS 9902 ACC and discusses some of the design considerations in the use of this device for asynchronous communications applications.

3.1 DEVICE INITIALIZATION

The ACC is initialized by the RESET command from the CPU (output bit 31), followed by loading the Control, Interval, Receive Data Rate, and Transmit Data Rate registers. Assume that the value to be loaded into the CRU Base Register (register 12) in order to point to bit 0 is 0040_{16}. In this application characters have seven bits of data plus even parity and one stop bit. The $\overline{\phi}$ input to the ACC is a 3 MHz signal. The ACC divides this signal frequency by three to generate an internal clock frequency of 1 MHz. An interrupt is generated by the Interval Timer every 1.6 milliseconds when timer interrupts are enabled. The transmitter operates at a data rate of 300 bits per second, and the receiver operates at 1200 bits per second.

NOTE

To operate both the transmitter and receiver at 300 bits per second, delete the "LDCR @RDR,11" instruction (see below), and the "LDCR @XDR,12" instruction will cause both data rate registers to be loaded and LRDR and LXDR to reset.

3.1.1 Initialization Program

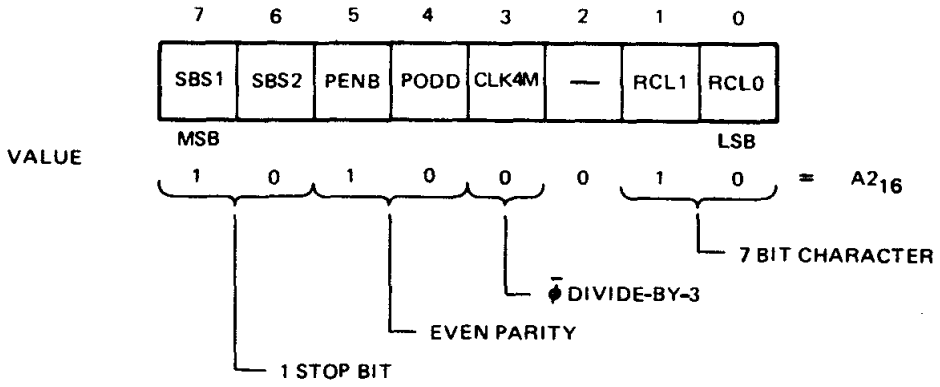
The initialization program for the configuration described above is shown below. The RESET command disables all interrupts, initializes all controllers, and sets the four register load control flags (LDCTRL, LDIR, LRDR, and LXDR). Loading the last bit of each of the registers causes the load control flag to reset automatically.

```
        LI      R12,>40     INITIALIZE CRU BASE
        SBO     31          RESET COMMAND
        LDCR    @ CNTRL,8   LOAD CONTROL AND RESET LDCTRL
        LDCR    @ INTVL,8   LOAD INTERVAL AND RESET LDIR
        LDCR    @ RDR,11    LOAD RDR AND RESET LRDR
        LDCR    @ XDR,12    LOAD XDR AND RESET LXDR
          .
          .
          .
CNTRL   BYTE    >A2
INTVL   BYTE    1600/64
RDR     DATA    >1A1
XDR     DATA    >4D0
```

The RESET command initializes all subcontrollers, disables interrupts, and sets LDCTRL, LDIR, LRDR, and LXDR, enabling loading of the control register.

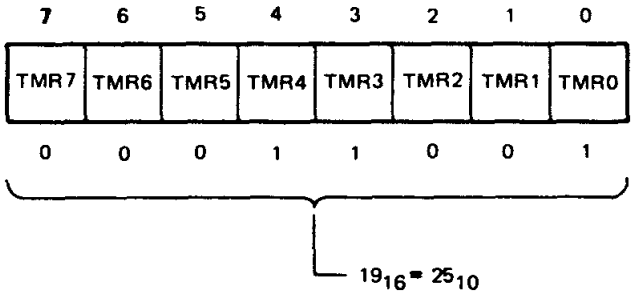
3.1.2 Control Register

The options listed in Table 3 in Section 2.1.2.1 are selected by loading the value shown below.



3.1.3 Interval Register

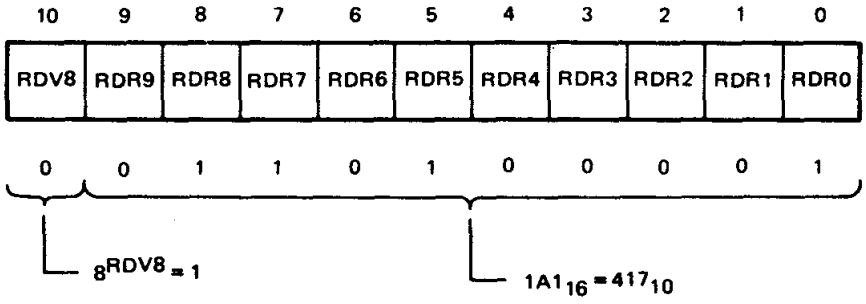
To set up the interval register to generate an interrupt every 1.6 milliseconds, load the value into the interval register to specify the number of 64-microsecond increments in the total interval desired.



25 X 64 MICROSECONDS = 1.6 MILLISECONDS

3.1.4 Receive Data Rate Register

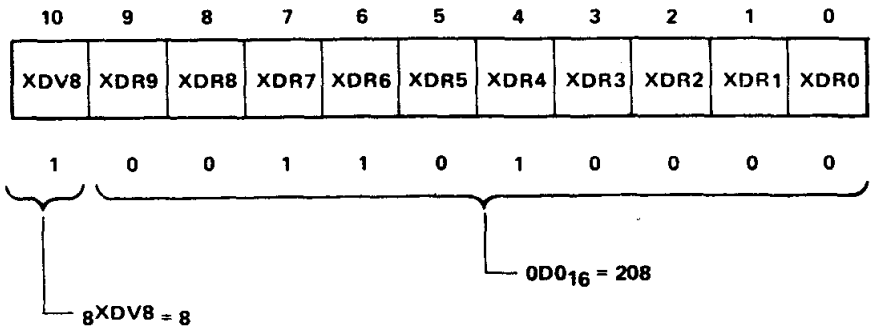
To set the data rate for the receiver to 1200 bits per second, load the value into the Receive Data Rate register as shown below:



$10^6 \div 1 \div 417 \div 2 = 1199.04$ BITS PER SECOND

3.1.5 Transmit Data Rate Register

To program the data rate for the transmitter for 300 bits per second, load the following value into the Transmit Data Rate register:



$1 \times 10^6 \div 8 \div 208 \div 2 = 300.48$ BITS PER SECOND

3.2 DATA TRANSMISSION

The subroutine shown below demonstrates a simple loop for transmitting a block of data.

```
        LI      R0, LISTAD      INITIALIZE LIST POINTER
        LI      R1, COUNT       INITIALIZE BLOCK COUNT
        LI      R12, CRUBAS     INITIALIZE CRU BASE
        SBO     16              TURN ON TRANSMITTER
XMTLP   TB      22              WAIT FOR XBRE = 1
        JNE     XMTLP
        LDCR    *R0+,8          LOAD CHARACTER INCREMENT POINTER
                                    RESET XBRE
        DEC     R1              DECREMENT COUNT
        JNE     XMTLP           LOOP IF NOT COMPLETE
        SBZ     16              TURN OFF TRANSMITTER
```

After initializing the list pointer, block count, and CRU base address, RTSON is set to cause the transmitter and the $\overline{RTS}$ output to become active. Data transmission does not begin, however, until the $\overline{CTS}$ input becomes active. After the final character is loaded into the Transmit Buffer register, RTSON is reset. The transmitter and the $\overline{RTS}$ output do not become inactive until the final character is transmitted.

3.3 DATA RECEPTION

The following software will cause a block of data to be received and stored in memory.

```
CARRET   BYTE   >0D
RCVBLK   LI     R2, RCVLST      INITIALIZE LIST COUNT
         LI     R3, MXRCNT      INITIALIZE MAX COUNT
         LI     R4, CARRET      SET UP END OF BLOCK CHARACTER
RCVLP    TB     21              WAIT FOR RBRL = 1
         JNE    RCVLP
         STCR   *R2,8           STORE CHARACTER
         SBZ    18              RESET RBRL
         DEC    R3              DECREMENT COUNT
         JEQ    RCVEND          END IF COUNT = 0
         CB     *R2+,R4         COMPARE TO EOB CHARACTER, INCREMENT
                                    POINTER
         JNE    RCVLP           LOOP IF NOT COMPLETE
RCVEND   RT                     END OF SUBROUTINE
```

3.4 REGISTER LOADING AFTER INITIALIZATION

The Control, Interval, and Data Rate registers may be reloaded after initialization. For example, it may be desirable to change the interval of the timer. Assume that the interval is to be changed to 10.24 milliseconds; the instruction sequence is:

```
         SBO    13              SET LOAD CONTROL FLAG
         LDCR   @ INTVL2,8      LOAD REGISTER, RESET FLAG
           .
           .
           .
INTVL2   BYTE   10240/64
```

When transmitter interrupts are enabled, caution should be exercised to ensure that a transmitter interrupt does not occur while the load control flag is set. For example, if a transmitter interrupt occurs between execution of the "SBO 13" and the next instruction, the transmit buffer is not enabled for loading when the Transmitter Interrupt service routine is entered because the LDIR flag is set. This situation may be avoided by the following sequence:

```
         BLWP   @ITVCHG         CALL SUBROUTINE
           .
           .
           .
ITVCPC   LIMI   0               MASK ALL INTERRUPTS
         MOV    @ 24(R13),R12   LOAD CRU BASE ADDRESS
         SBO    13              SET FLAG
         LDCR   @ INTVL2,8      LOAD REGISTER AND RESET FLAG
         RTWP                   RESTORE MASK AND RETURN
           .
           .
           .
           .
ITVCHG   DATA   ACCWP, ITVCPC
INTVL2   BYTE   10240/64
```

In this case all interrupts are masked, ensuring that all interrupts are disabled while the load control flag is set.

3.5 INTERFACE TO A DATA TERMINAL

Following is a discussion of the TMS 9902 interface to a TI Model 733 data terminal as implemented on the TM 990/100M microcomputer module. Figure 9 diagrams the hardware interface, and Table 7 lists the software interface. The 733 data terminal is an ASCII-code, serial, asynchronous, EIA device equipped with a keyboard, thermal printer, and digital cassette tape.

3.5.1 Hardware Interface

The hardware interface between the TMS 9902 and the 733 data terminal is shown in Figure 9. The asynchronous communication conforms to *EIA Standard RS-232-C*. The 75188 and 75189 performs the necessary level shifting between TTL levels and RS-232-C levels. The ACC chip enable ($\overline{9902SEL}$) signal comes from decode circuitry which looks at A0-A9 on CRU cycles. The interrupt output ($\overline{INT}$) of the TMS 9902 is sent to the TMS 9901 for prioritization and encoding. When the 9902 is communicating with a terminal, the $\overline{RTS}$ pin can be connected to the $\overline{CTS}$ pin because the terminal will always be in the clear-to-send ($\overline{CTS}$) condition.

3.5.2 Software

The software required to initialize, read from, and write to the TMS 9902 ACC is listed in Table 7. These routines are taken directly from TIBUG (TM 990/402-1) which is the monitor that runs on the TM 990/100M boards. The coding shown is part of a routine entered because of a power-up reset. Before this section of code was entered, not shown, R12 is set to the correct value of the TMS 9902 CRU base address. The baud rate is detected by measuring the start bit length when an "A" is entered via the keyboard. The variable COUNT is incremented every time the SPLOOP loop is executed. When a zero is seen at 9902 bit 15 (RIN) the start bits are finished being received. The value of COUNT is then compared against a table of known values in TABLE to determine the baud rate.

TIBUG assumes that all 1200-baud data terminals are TI Model 733 data terminals. The TI Model 733 communicates at 1200 baud, but prints at 300 baud; this means that bits travel the communications line at 1200 baud, but the spacing between characters is 300 baud. A wait loop is included in the write character routine to handle this spacing requirement. The TIBUG T command is used to indicate that a 1200 baud terminal is true 1200 baud; i.e., not a TI 733.

This code is taken from the middle of TIBUG; thus constructs and symbols are used which are not defned here. Lines 261 and 262 of the code contain XOP calls. The READ opcode is really a call to XOP 13 and the MESG opcode is a call to XOP 14, which in turn calls XOP 12. This can be figured out if the assembled code for these opcodes is examined. Following is a list of EQU statements that appear at the beginning of TIBUG, but are not shown here:

```
COUNT     EQU     3
POINT     EQU     7
LINK      EQU     11
CRUBAS    EQU     12
```

Once again, these values could easily be obtained by looking at the assembled code for the statement in which the symbol is used.

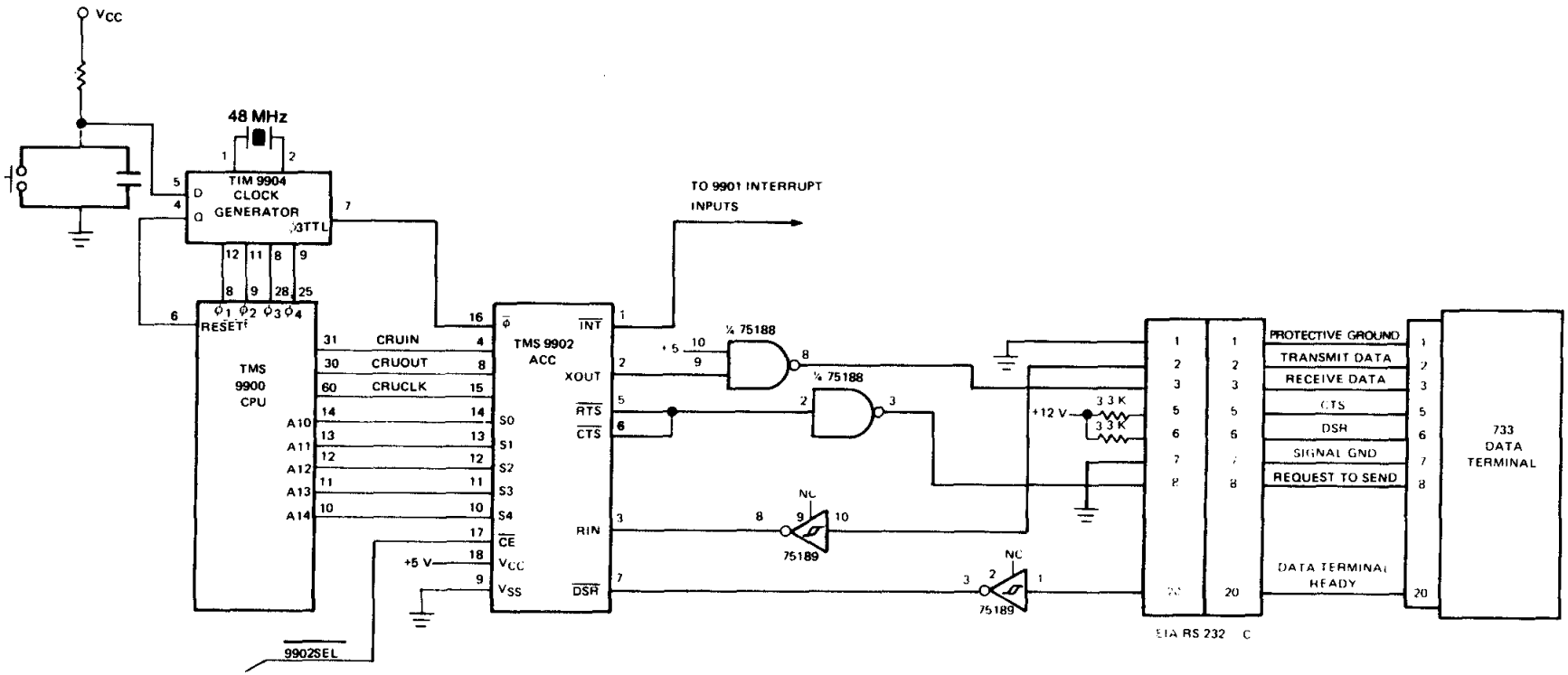


FIGURE 9. INTERFACE TO A 733 DATA TERMINAL

```
TIBUG
***COMMAND SEARCH AND SYSTEM INZ***

STATEMENT NO.  ADDRESS (HEX)  ASSEMBLED CODE

0231                          *
0232                          * INITIALIZE TMS9902 FOR: *BAUD RATE
0233                          *                        *7 BITS/CHARACTER
0234                          *                        *EVEN PARITY
0235                          *                        *2 STOP BITS
0236                          *                        *POLLED OPERATION
0237                          *
0238 015E 1D1F                SBO  31                  RESET TMS9902 UART

0239 0160 3220                LDCR @CR,8               INITIALIZE TMS9902 CONTROL REG
     0162 01A4'
0240 0164 1E0D                SBZ  13                  DO NOT INT INTERVAL REG
0241 0166 04C3                CLR  COUNT               RESET LOOP COUNT
0242 0168 1F0F  TSTSP         TB   15                  SPACE?
0243 016A 13FE                JEQ  TSTSP               NO, JUMP BACK
0244 016C 0583  SPLOOP        INC  COUNT               TIME THE START BIT
0245 016E 1F0F                TB   15                  FALL OUT ON A MARK
0246 0170 16FD                JNE  SPLOOP
0247                          *
0248                          * TABLE SEARCH FOR BAUD RATE
0249                          *
0250 0172 0207                LI   POINT,TABLE         SET POINTER TO TABLE
     0174 0194'
0251 0176 8DC3  BDLOOP        C    COUNT,*POINT+       MATCH?
0252 0178 1202                JLE  MATCH               YES, SET BAUD RATE
0253 017A 05C7                INCT POINT               NO, UPDATE POINTER
0254 017C 10FC                JMP  BDLOOP
0255      017E' MATCH         EQU  $
0256 017E 3317                LDCR *POINT,12           INT. REC./XMT. DATA RATE
0257 0180 C1D7                MOV  *POINT,POINT
0258 0182 0287                CI   POINT,>1A0          1200 BAUD ?
     0184 01A0
0259 0186 1602                JNE  BANNER              LEAVE ASR FLAG ALONE
0260 0188 0720                SETO @ASR                SET 733ASR FLAG
     018A FFF4
0261 018C 2F45  BANNER        READ CHAR
0262 018E 2FA0                MESG @LOGON              PRINT LOG ON MESSAGE
     0190 022B'
0263 0192 10DC                JMP  JMMONT              TO TOP OF MONITOR
0264 0194 0040  TABLE         DATA >40,>D0             2400 BAUD
     0196 00D0
0265 0198 0070                DATA >70,>1A0            1200 BAUD
     019A 01A0
0266 019C 0200                DATA >200,>4D0           300 BAUD
     019E 04D0
0267 01A0 0400                DATA >400,>638           110 BAUD
     01A2 0638
0268 01A4   62  CR            BYTE >62
```

TABLE 7 (Continued)

```
*** WRITE CHARACTER ***

0290                    ***********************************************
0291                    * WRITE CHARACTER -- XOP  R,12
0292                    *                              ---      NORMAL RETURN
0293                    *
0294                    * TRANSMIT THE CHARACTER IN THE LEFT BYTE OF
0295                    * USER REGISTER R. IF THE CHARACTER IS A
0296                    * CARRIAGE RETURN, THE ROUTINE WAITS 200 MSEC FOR
0297                    * THE CARRIAGE TO RETURN. IF THE TERMINAL IS
0298                    * A 733ASR AS DENOTED IN THE T COMMAND, EACH
0299                    * CHARACTER IS PADDED WITH 25 MSEC TO REDUCE
0300                    * THE TRANSFER RATE TO 300 BAUD.
0301                    ***********************************************
0302 01B6 020A  WENTRY LI   R10,3750
     01B8 0EA6
0303 01BA 020C         LI   CRUBAS,>80    SET CRU BASE REG.
     01BC 0080
0304 01BE 1D10         SBO  16            SET RTSON
0305 01C0 1F16         TB   22            TRANSMIT BUFFER REG. EMPTY?
0306 01C2 16F9         JNE  WENTRY        NO, WAIT UNTIL IT IS
0307 01C4 321B         LDCR *LINK,8       CHARACTER TO UART
0308 01C6 D2DB         MOVB *LINK,LINK
0309 01C8 1E10         SBZ  16            RESET RTSON
0310 01CA 098B         SRL  LINK,8
0311 01CC 028B         CI   LINK,>000D    CARRIAGE RETURN
     01CE 000D
0312 01D0 1608         JNE  ASR733        NO, SKIP
0313 01D2 0A3A         SLA  R10,3
0314 01D4 1F16  WLOOP1 TB   22            WAIT FOR XMISSION TO END
0315 01D6 16FE         JNE  WLOOP1
0316 01D8 1F17         TB   23
0317 01DA 16FC         JNE  WLOOP1
0318 01DC 060A  WLOOP2 DEC  R10           WAIT LOOP
0319 01DE 16FE         JNE  WLOOP2
0320 01E0 0380         RTWP
0321 01E2 C2E0  ASR733 MOV  @DUMPFG,LINK IN DUMP ROUTINE ?
     01E4 FFF6
0322 01E6 1303         JEQ  WEXIT         YES,IGNORE ASR FLAG
0323 01E8 C2E0         MOV  @ASR,LINK     ASR733 ?
     01EA FFF4
0324 01EC 16F3         JNE  WLOOP1        YES, WAIT 3 NULLS
0325 01EE 0380  WEXIT  RTWP

TIBUG
*** READ CHARACTER ***

0271                    ************************************************
0272                    * READ CHARACTER -- XOP  R,13
0273                    *                           --      NORMAL RETURN
0274                    *
0275                    * READ WAITS FOR A CHARACTER TO BE ASSEMBLED IN
0276                    * THE UART. THE CHARACTER IS PLACED IN THE LEFT
0277                    * BYTE OF USER REGISTER R. THE RIGHT BYTE IS
0278                    * ZEROED. ALL ERRORS ARE IGNORED.
0279                    ************************************************
0280                    *
0281 01A6 020C  RENTRY LI   CRUBAS,>80    SET CRU BASE REG.
     01A8 0080
0282 01AA 1F15         TB   21            RECEIVE BUFFER REG. FULL?
0283 01AC 16FC         JNE  RENTRY        NO, LOOP
0284 01AE 04DB         CLR  *LINK
0285 01B0 361B         STCR *LINK,8
0286 01B2 1E12         SBZ  18
0287 01B4 0380         RTWP
```

4. TMS 9902 ELECTRICAL SPECIFICATIONS

4.1 Absolute Maximum Ratings Over Operating Free Air Temperature Range (Unless Otherwise Noted) *

Supply voltage, V_{CC} ..... −0.3 V to 10 V
All inputs and output voltages ..... −0.3 V to 10 V
Continuous power dissipation ..... 0.55 W
Operating free-air temperature range ..... 0°C to 70°C
Storage temperature range ..... −65°C to 150°C

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute maximum rated conditions for extended periods may affect device reliability.

4.2 Recommended Operating Conditions *

PARAMETER	MIN	NOM	MAX	UNIT
Supply voltage, V_{CC}	4.75	5.0	5.25	V
Supply voltage, V_{SS}		0		V
High-level input voltage, V_{IH}	2.0		V_{CC}	V
Low-level input voltage, V_{IL}	V_{SS}−.3		0.8	V
Operating free-air temperature, T_A	0		70	°C

4.3 Electrical Characteristics Over Full Range of Recommended Operating Conditions (Unless Otherwise Noted) *

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
V_{OH}	High level output voltage	I_{OH} = −100 μA	2.4		V_{CC}	V
		I_{OH} = −200 μA	2.2		V_{CC}	V
V_{OL}	Low level output voltage	I_{OL} = 3.2 mA	V_{SS}		0.4	V
I_I	Input current (any input)	V_I = 0 V to V_{CC}			±10	μA
$I_{CC(av)}$	Average supply current from V_{CC}	$t_{c(\phi)}$ = 330 ns, T_A = 70°C			100	mA
C_i	Small signal input capacitance, any input	f = 1 MHz			15	pF

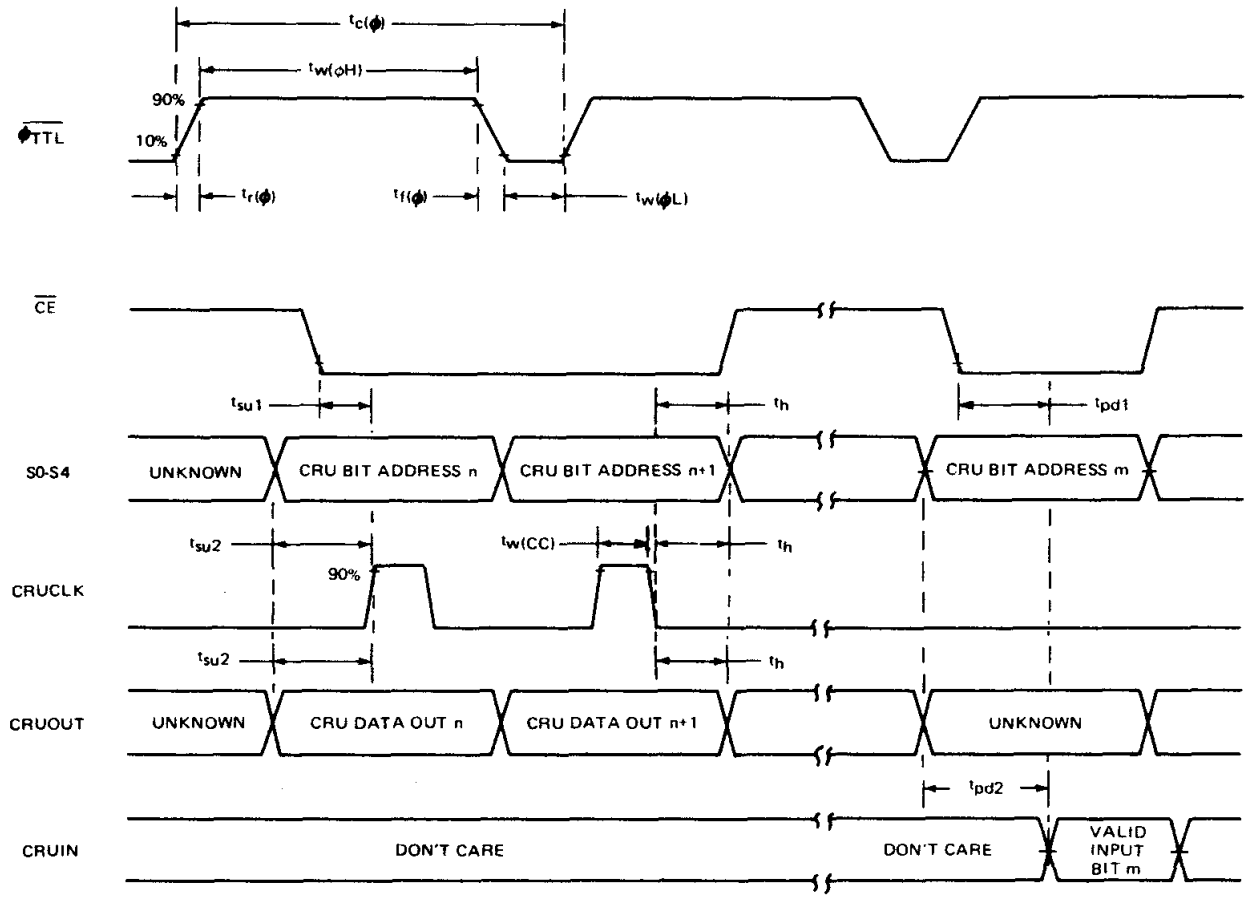
4.4 Timing Requirements Over Full Range of Operating Conditions

PARAMETER		MIN	TYP	MAX	UNIT
$t_{c(\phi)}$	Clock cycle time	300	333	667	ns
$t_{r(\phi)}$	Clock rise time	5		40	ns
$t_{f(\phi)}$	Clock fall time	10		40	ns
$t_{w(\phi H)}$	Clock pulse width (high level)	225			ns
$t_{w(\phi L)}$	Clock pulse width (low level)	45			ns
$t_{w(CC)}$	CRUCLK pulse width	100	185		ns
t_{su1}	Setup time for $\overline{CE}$ before CRUCLK	150			ns
t_{su2}	Setup time for S0-S4, or CRUOUT before CRUCLK	180			ns
t_h	Hold time for $\overline{CE}$, S0-S4, or CRUOUT after CRUCLK	60			ns

*NOTE: All voltage values are referenced to V_{SS}.

4.5 Switching Characteristics Over Full Range of Recommended Operating Conditions

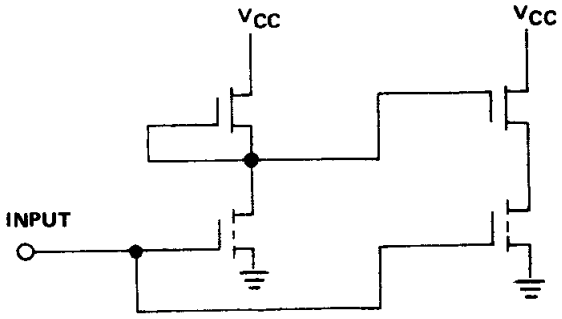
	PARAMETER	TEST CONDITION	MIN	TYP	MAX	UNIT
t_{pd1}	Propagation delay, $\overline{CE}$ to valid CRUIN	CL = 100pF			300	ns
t_{pd2}	Propagation delay, S0-S4 to valid CRUIN	CL = 100pF			320	ns



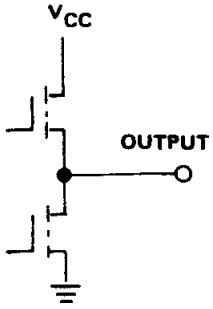
SWITCHING CHARACTERISTICS

NOTE: ALL SWITCHING TIMES ARE ASSUMED TO BE AT 10% OR 90% VALUES.

EQUIVALENT OF I/O INPUTS



EQUIVALENT OF I/O OUTPUTS

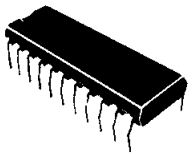


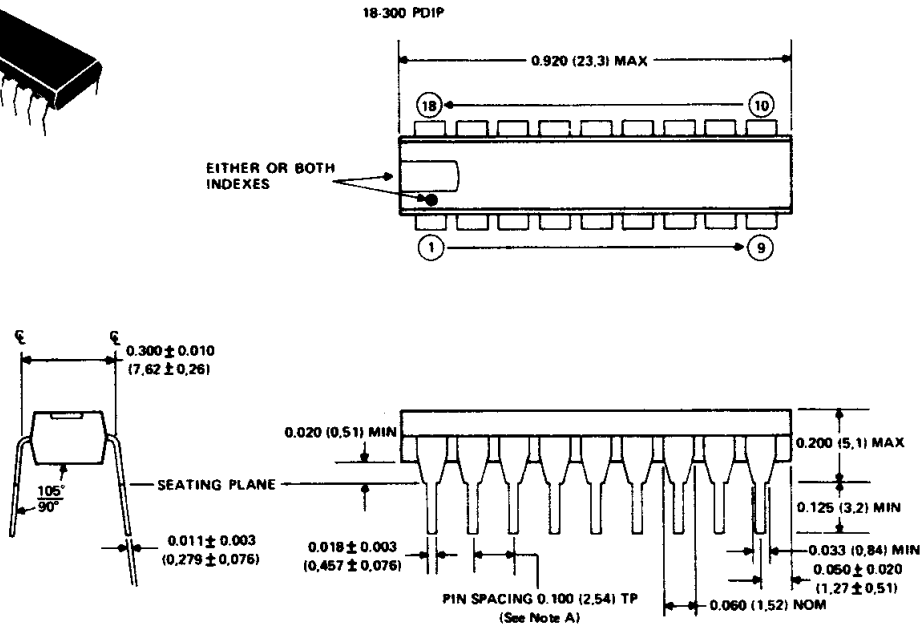
INPUT AND OUTPUT EQUIVALENTS

5. MECHANICAL SPECIFICATIONS

TMS 9902 NL

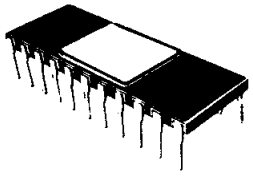
18 pin plastic packages

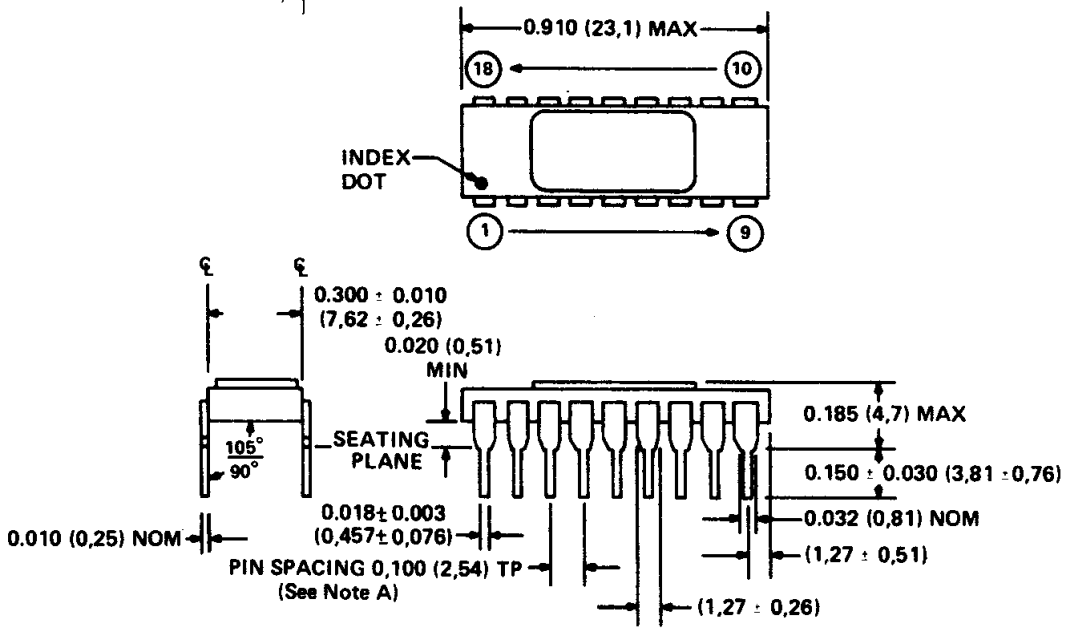




NOTES: A. Each pin centerline is located within 0.010 (0,26) of its true longitudinal position.
B. All linear dimensions are in inches and parenthetically in millimeters. Inch dimensions govern.

18-pin ceramic packages with side-brazed leads and metal or epoxy or glass lid seal





NOTES: A. Each pin centerline is located within 0.010 (0,26) of its true longitudinal position.
B. All linear dimensions are in inches and parenthetically in millimeters. Inch dimensions govern.

APPENDIX G

The Engineering Staff of
TEXAS INSTRUMENTS INCORPORATED
Semiconductor Group

TMS 9980A/ TMS 9981 Microprocessor Data Manual

NOVEMBER 1977

TEXAS INSTRUMENTS
INCORPORATED

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

1.1 DESCRIPTION

The TMS 9980A/TMS 9981 is a software-compatible member of TI's 9900 family of microprocessors. Designed to minimize the system cost for smaller systems, the TMS 9980A/TMS 9981 is a single-chip 16-bit central processing unit (CPU) which has an 8-bit data bus, on-chip clock, and is packaged in a 40-pin package (see Figure 1). The instruction set of the TMS 9980A/TMS 9981 includes the capabilities offered by full minicomputers and is exactly the same as the 9900's. The unique memory-to-memory architecture features multiple register files, resident in memory, which allow faster response to interrupts and increased programming flexibility. The separate bus structure simplifies the system design effort. Texas Instruments provides a compatible set of MOS and TTL memory and logic function circuits to be used with a TMS 9980A/TMS 9981 system.

1.2 KEY FEATURES

- 16-Bit Instruction Word
- Full Minicomputer Instruction Set Capability Including Multiply and Divide
- Up to 16,384 Bytes of Memory
- 8-Bit Memory Data Bus
- Advanced Memory-to-Memory Architecture
- Separate Memory, I/O, and Interrupt-Bus Structures
- 16 General Registers
- 4 Prioritized Interrupts
- Programmed and DMA I/O Capability
- On-Chip 4-Phase Clock Generator
- 40-Pin Package
- N-Channel Silicon-Gate Technology

1.3 TMS 9980A/TMS 9981 DIFFERENCES

The TMS 9980A and the TMS 9981 although very similar, have several differences which user should be aware.

1. The TMS 9980A requires a V_{BB} supply (pin 21) while the TMS 9981 has an internal charge pump to generate V_{BB} from V_{CC} and V_{DD}.
2. The TMS 9981 has an optional on-chip crystal oscillator in addition to the external clock mode of the TMS 9980A.
3. The pin-outs are not compatible for D0-D7, INT0-INT2, and $\overline{\phi}3$.

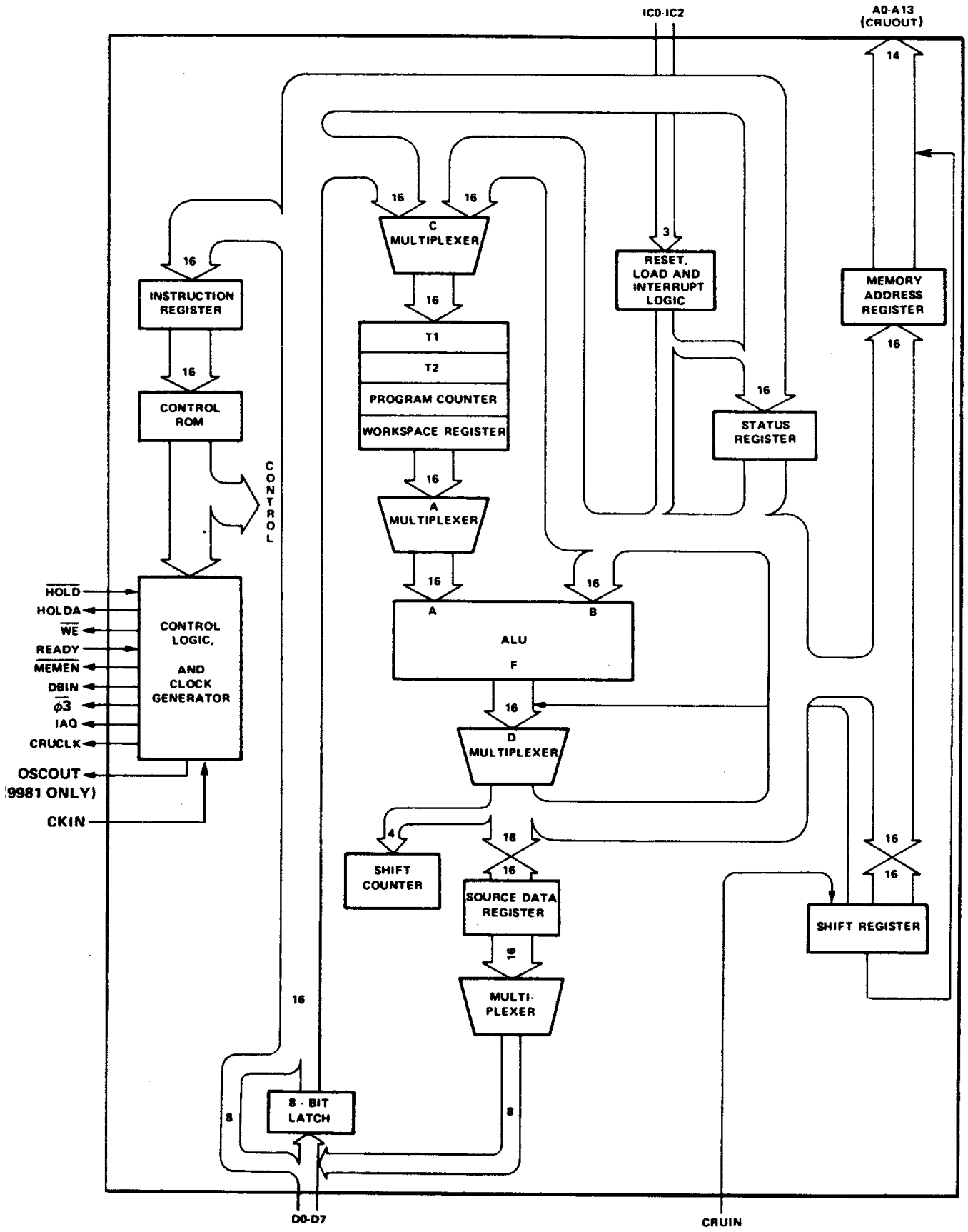
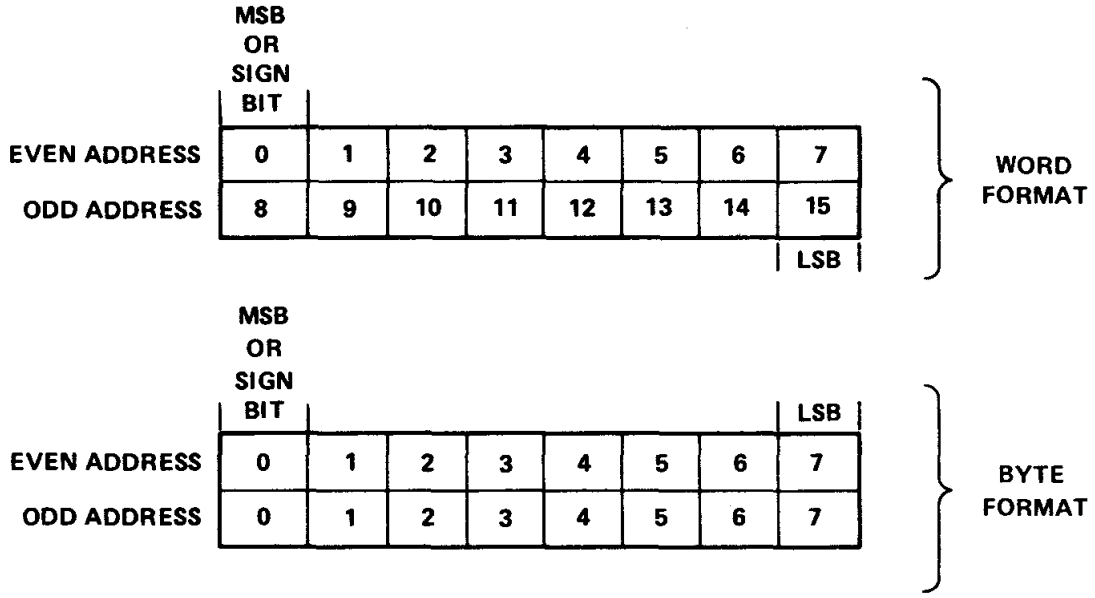


FIGURE 1 – ARCHITECTURE

2. ARCHITECTURE

The memory for the TMS 9980A/TMS 9981 is addressable in 8-bit bytes. A word is defined as 16 bits or 2 consecutive bytes in memory. The words are restricted to be on even address boundaries, i.e., the most-significant half (8 bits) resides at even address and the least-significant half resides at the subsequent odd address. A byte can reside at even or odd address. The word and byte formats are shown below.



2.1 REGISTERS AND MEMORY

The TMS 9980A/TMS 9981 employs an advanced memory-to-memory architecture. Blocks of memory designated as workspace replace internal hardware registers with program-data registers. The TMS 9980A/TMS 9981 memory map is shown in Figure 2. The first two words (4 bytes) are used for RESET trap vector. Addresses 0004_{16} through 0013_{16} are used for interrupt vectors. Addresses 0040 through 007F are used for the extended operation (XOP) instruction trap vectors. The last four bytes at address $3FFC_{16}$ to 3FFF are used for trap vector for the LOAD function.

The remaining memory is available for programs, data, and workspace registers. If desired, any of the special areas may also be used as general memory.

Three internal registers are accessible to the user. The program counter (PC) contains the address of the instruction following the current instruction being executed. This address is referenced by the processor to fetch the next instruction from memory and is then automatically incremented. The status register (ST) contains the present state of the processor and will be further defined in Section 3.4. The workspace pointer (WP) contains the address of the first word in the currently active set of workspace registers.

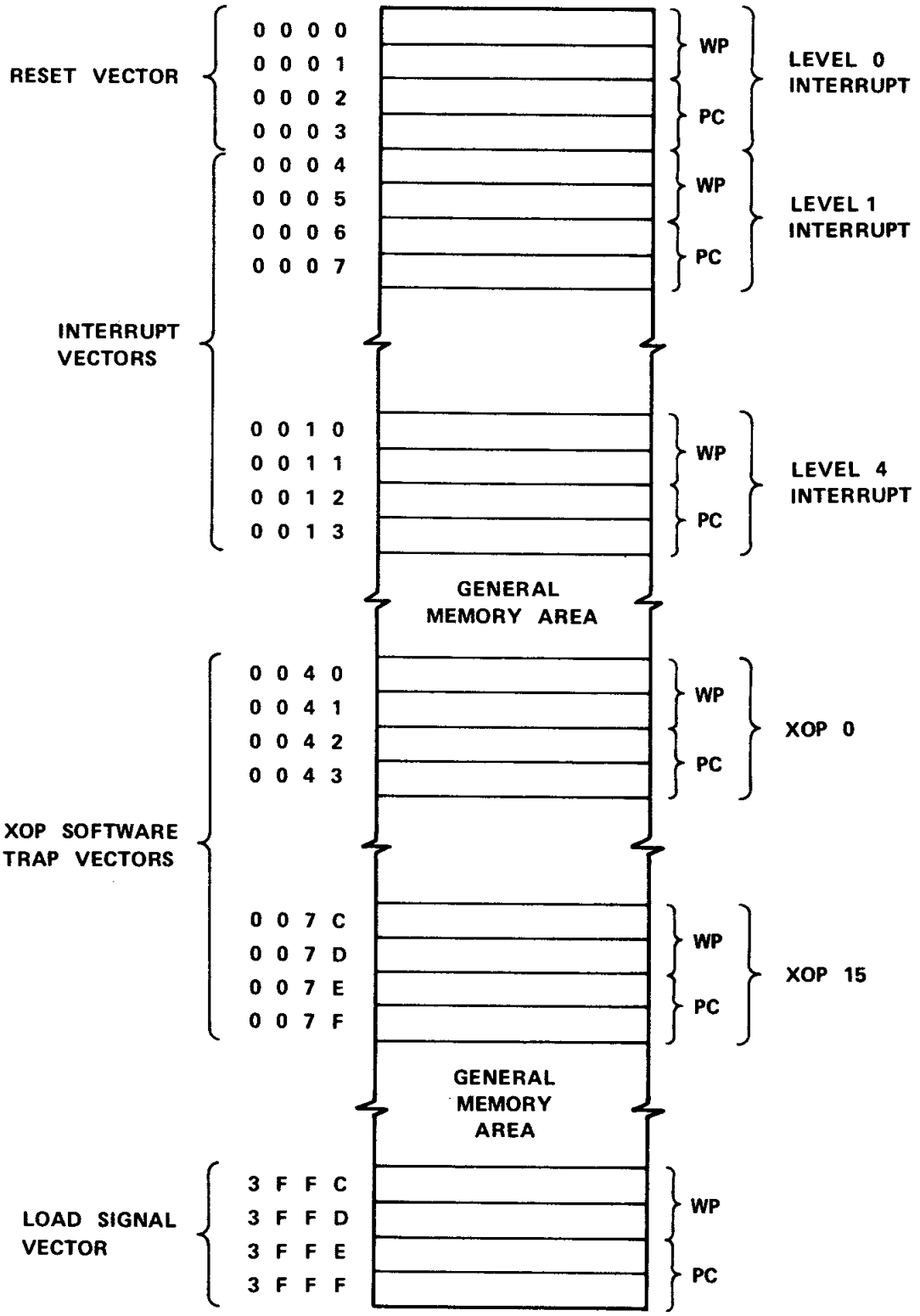
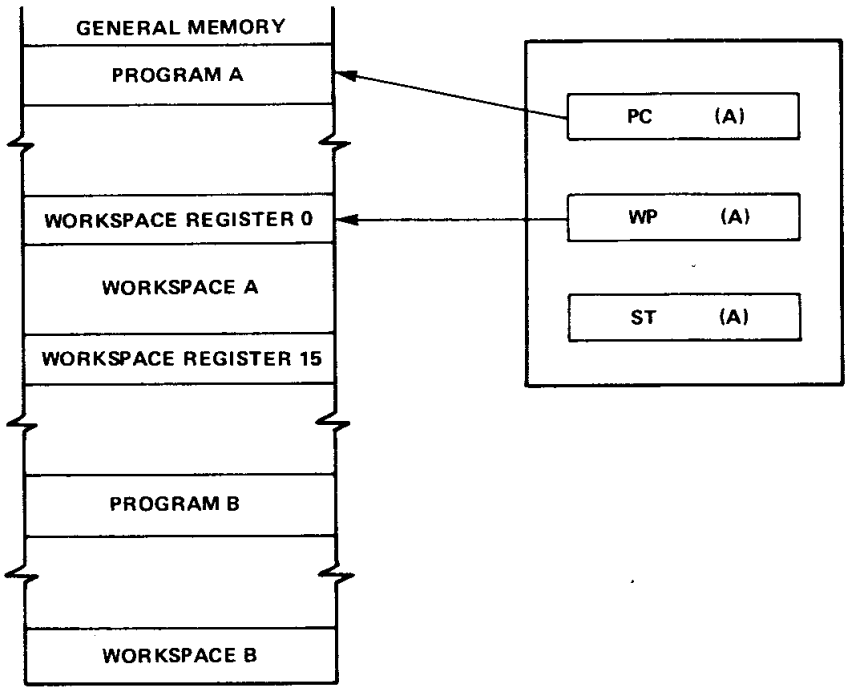


FIGURE 2 – MEMORY MAP

A workspace-register file occupies 16 contiguous memory words in the general memory area. Each workspace register may hold data or addresses and function as operand registers, accumulators, address registers, or index registers. During instruction execution, the processor addresses any register in the workspace by adding the register number to the contents of the workspace pointer and initiating a memory request for the word. The relationship between the workspace pointer and its corresponding workspace is shown below.



The workspace concept is particularly valuable during operations that require a context switch, which is a change from one program environment to another (as in the case of an interrupt or call to a subroutine). Such an operation, using a conventional multi-register arrangement, requires that at least part of the contents of the register file be stored and reloaded. A memory cycle is required to store or fetch each word. By exchanging the program counter, status register, and workspace pointer, the TMS 9980A/TMS 9981 accomplishes a complete context switch with only six store cycles and six fetch cycles. After the switch the workspace pointer contains the starting address of a new 16-word workspace in memory for use in the new routine. A corresponding time saving occurs when the original context is restored. Instructions in the TMS 9980A/TMS 9981 that result in a context switch include:

1. Branch and Load Workspace Pointer (BLWP)
2. Return from Subroutine (RTWP)
3. Extended Operation (XOP)

Device interrupts, $\overline{\text{RESET}}$, and $\overline{\text{LOAD}}$ also cause a context switch by forcing the processor to trap to a service subroutine.

2.2 INTERRUPTS

The architecture of the 9900 family allows vectoring of 16 interrupts. These interrupts are assigned levels from 0 to 15. The interrupt at level 0 has the highest priority and the interrupt at level 15 has the lowest priority. The TMS 9900 implements all 16 interrupt levels. The TMS 9980A/TMS 9981 implements only 5 levels (level 0 and levels 1 through 4). Level 0 is reserved for $\overline{\text{RESET}}$ function.

Levels 1 through 4 may be used for external devices. The external levels may also be shared by several device interrupts, depending upon system requirements. The TMS 9980A/TMS 9981 continuously compares the interrupt code (IC0 through IC2) with the interrupt mask contained in status-register bits 12 through 15. When the level of the pending interrupt is less than or equal to the enabling mask level (higher or equal priority interrupt), the processor recognizes the interrupt and initiates a context switch following completion of the currently executing instruction. The processor fetches the new context WP and PC from the interrupt vector locations. Then, the previous context WP, PC, and ST are stored in workspace registers 13, 14, and 15, respectively, of the new workspace. The TMS 9980A/TMS 9981 then forces the interrupt mask to a value that is one less than the level of the interrupt being serviced. This allows only interrupts of higher priority to interrupt a service routine. The processor also inhibits interrupts until the first instruction of the service routine has been executed to allow modification of interrupt mask if needed (to mask out certain interrupts). All interrupt requests should remain active until recognized by the processor in the device-service routine. The individual service routines must reset the interrupt requests before the routine is complete. The interrupt code (IC0-IC2) may change asynchronously within the constraints specified in Section 2.10.4.

If a higher priority interrupt occurs, a second context switch occurs to service the higher-priority interrupt. When that routine is complete, a return instruction (RTWP) restores the first service routine parameters to the processor to complete processing of the lower-priority interrupt. All interrupt subroutines should terminate with the return instruction to restore original program parameters. The interrupt-vector locations, device assignment, enabling mask value and the interrupt code are shown in Table 1.

TABLE 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	-----	-----	-----

Note that $\overline{\text{RESET}}$ and $\overline{\text{LOAD}}$ functions are also encoded on the interrupt code input lines. Figure 3 illustrates some of the possible configurations. To realize $\overline{\text{RESET}}$ and one interrupt no external component is needed. If $\overline{\text{LOAD}}$ is also needed, a three input AND gate is wired as shown. If the system requires more than one interrupt, a single SN74148 (TIM 9907) is required.

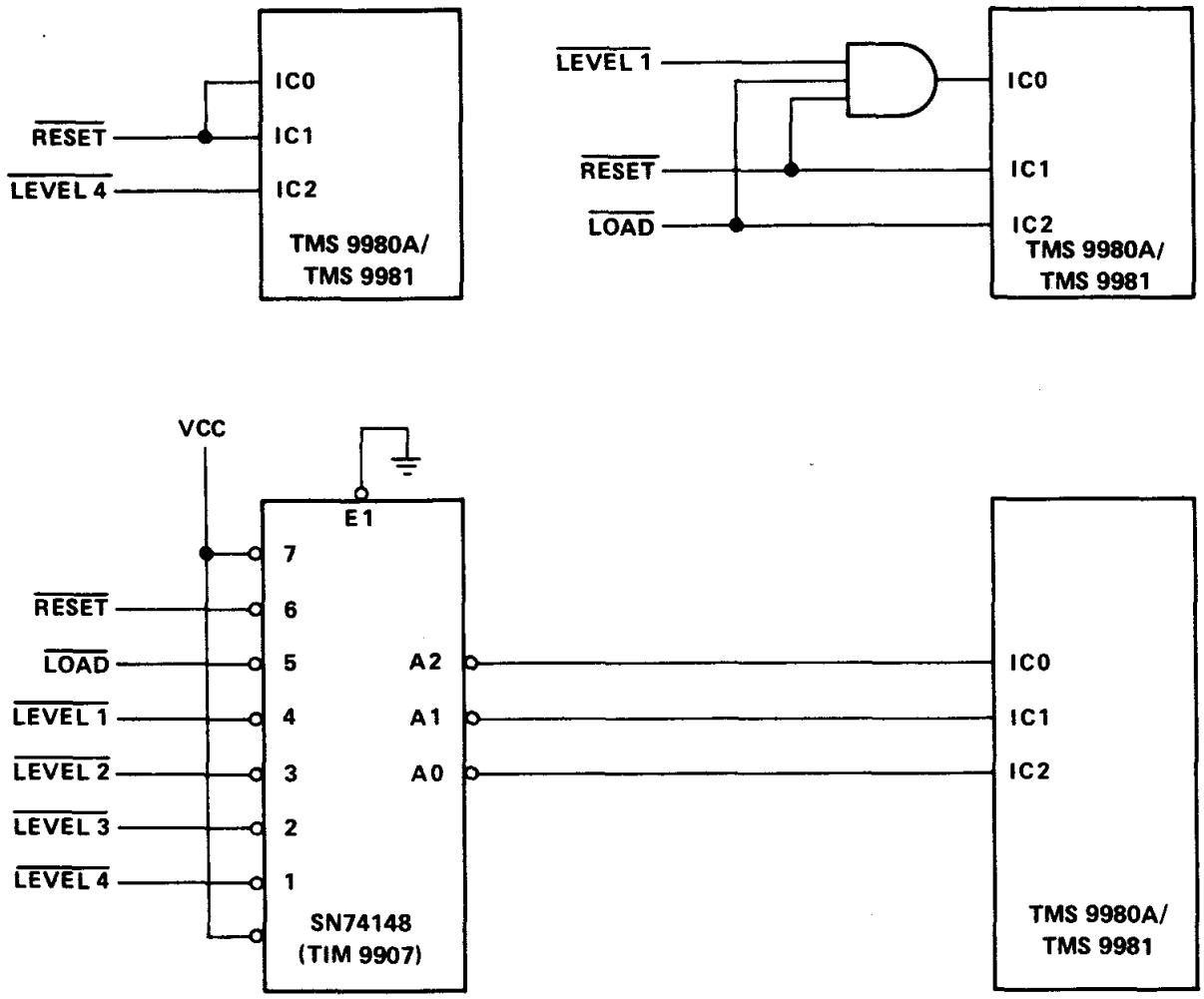


FIGURE 3 – TMS 9980A/TMS 9981 INTERRUPT INTERFACE

2.3 INPUT/OUTPUT

The TMS 9980A/TMS 9981 utilizes a versatile direct command-driven I/O interface designated as the communications-register unit (CRU). The CRU provides up to 2,048 directly addressable output bits. Both input and output bits can be addressed individually or in fields of from 1 to 16 bits. The TMS 9980A/TMS 9981 employs CRUIN, CRUCLK, and A13 (for CRUOUT) and 11 bits (A2-A12) of the address bus to interface with the CRU system. The processor instructions that drive the CRU interface can set, reset, or test any bit in the CRU array or move between memory and CRU data fields.

2.4 SINGLE-BIT CRU OPERATIONS

The TMS 9980A/TMS 9981 performs three single-bit CRU functions: test bit (TB), set bit to one (SBO), and set bit to zero (SBZ). To identify the bit to be operated upon, the TMS 9980A/TMS 9981 develops a CRU-bit address and places it on the address bus, A2 to A12.

For the two output operations (SBO and SBZ), the processor also generates a CRUCLK pulse, indicating an output operation to the CRU device and places bit 7 of the instruction word on the A13 line to accomplish the specified

operation (bit 7 is a one for SBO and a zero for SBZ). A test-bit instruction transfers the addressed CRU bit from the CRUIN input line to bit 2 of the status register (EQUAL).

The TMS 9980A/TMS 9981 develops a CRU-bit address for the single-bit operations from the CRU-base address contained in workspace register 12 and the signed displacement count contained in bits 8 through 15 of the instruction. The displacement allows two's complement addressing from base minus 128 bits through base plus 127 bits. The base address from W12 is added to the signed displacement specified in the instruction and the result is loaded into the address bus. Figure 4 illustrates the development of a single-bit CRU address.

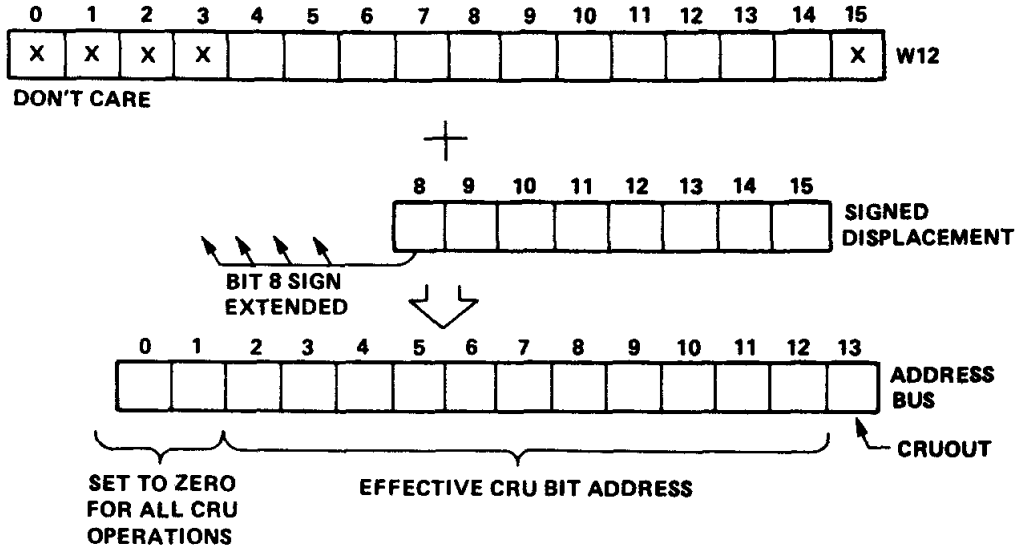


FIGURE 4 – TMS 9980A/TMS 9981 SINGLE-BIT CRU ADDRESS DEVELOPMENT

2.5 MULTIPLE-BIT CRU OPERATIONS

The TMS 9980A/TMS 9981 performs two multiple-bit CRU operations: store communications register (STCR) and load communications register (LDCR). Both operations perform a data transfer from the CRU-to-memory or from memory-to-CRU as illustrated in Figure 5. Although the figure illustrates a full 16-bit transfer operation, any number of bits from 1 through 16 may be involved. The LDCR instruction fetches a word from memory and right-shifts it to serially transfer it to CRU output bits. If the LDCR involves eight or fewer bits, those bits come from the right-justified field within the addressed byte of the memory word. If the LDCR involves nine or more bits, those bits come from the right-justified field within the whole memory word. When transferred to the CRU interface, each successive bit receives an address that is sequentially greater than the address for the previous bit. This addressing mechanism results in an order reversal of the bits; that is, bit 15 of the memory word (or bit 7) becomes the lowest addressed bit in the CRU and bit 0 becomes the highest addressed bit in the CRU field.

An STCR instruction transfers data from the CRU to memory. If the operation involves a byte or less transfer, the transferred data will be stored right-justified in the memory byte with leading bits set to zero. If the operation involves from nine to 16 bits, the transferred data is stored right-justified in the memory word with leading bits set to zero.

When the input from the CRU device is complete, the first bit from the CRU is the least-significant-bit position in the memory word or byte.

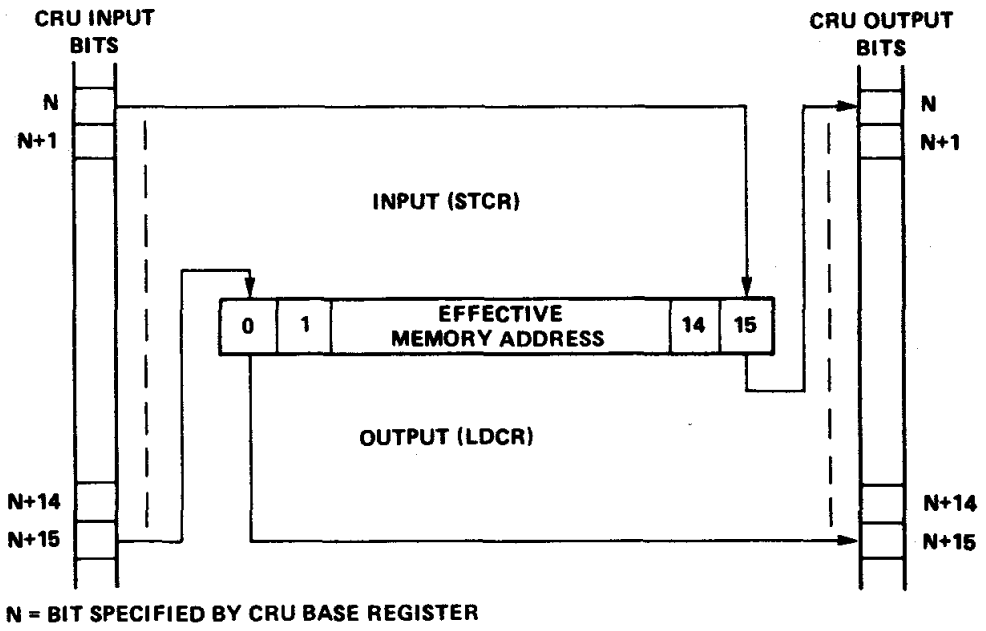


FIGURE 5 – TMS 9980A/TMS 9981 LDCR/STCR DATA TRANSFERS

Figure 6 illustrates how to implement a 16-bit input and a 16-bit output register in the CRU interface. CRU addresses are decoded as needed to implement up to 128 such 16-bit interface registers. In system application, however, only the exact number of interface bits needed to interface specific peripheral devices are implemented. It is not necessary to have a 16-bit interface register to interface an 8-bit device.

2.6 EXTERNAL INSTRUCTIONS

The TMS 9980A/TMS 9981 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/TMS 9981 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/TMS 9981, a unique 3-bit code appears on the address bus, bits A13, A0, and A1, along with a CRUCLK pulse. When the TMS 9980A/TMS 9981 is in an idle state, the 3-bit code and CRUCLK pulses occur repeatedly until the idle state is terminated. The codes are:

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

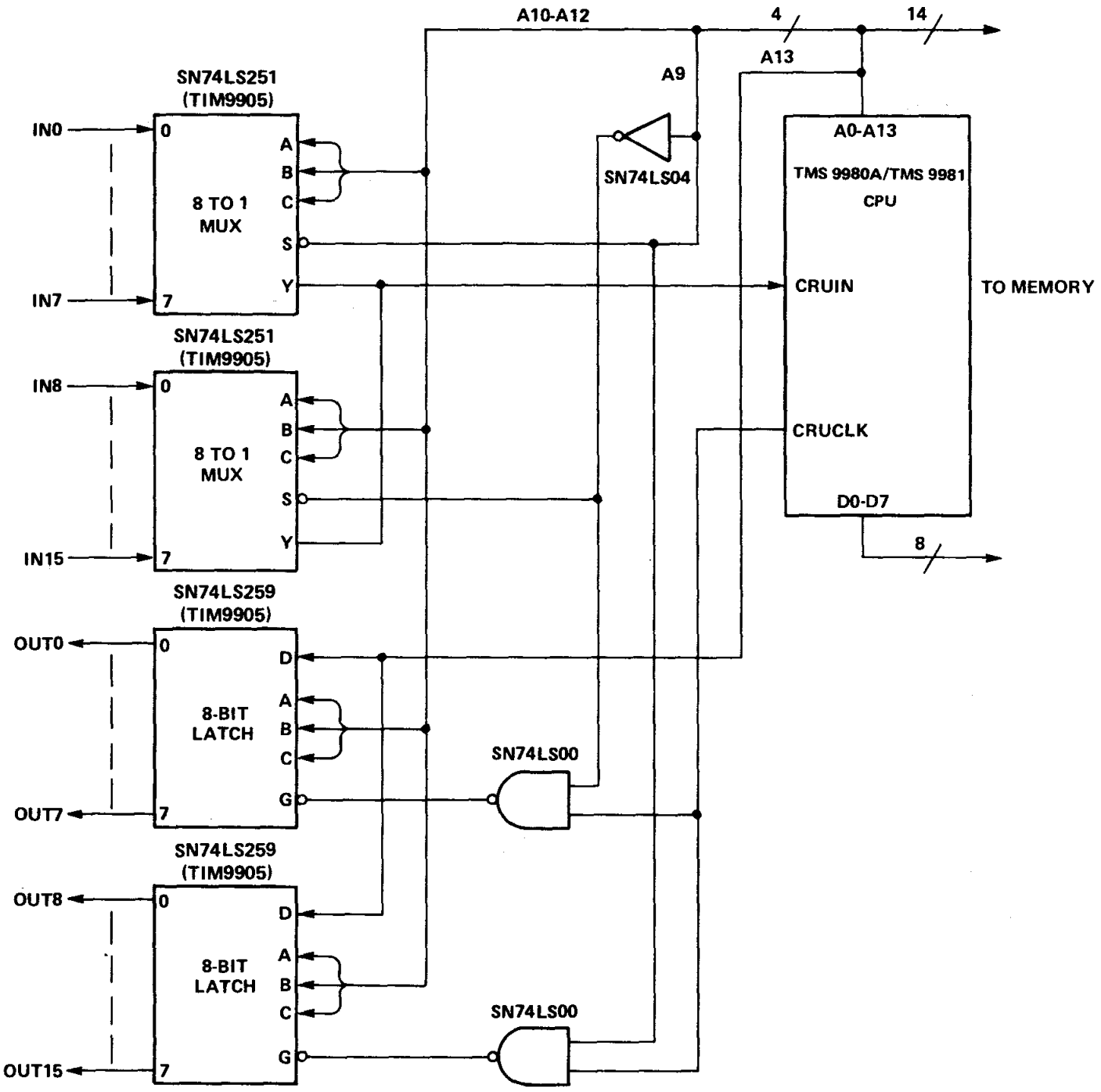


FIGURE 6 – TMS 9980A/9981 16-BIT INPUT/OUTPUT INTERFACE

Note that during external instructions bits (A2-A12) of the address bus may have any of the possible binary patterns. Since these bits (A2-A12) are used as CRU addresses, CRUCLK to the CRU must be gated with a decode of 0 on A0 and A1 to avoid erroneous strobe to CRU bits during external instruction execution.

Figure 7 illustrates typical external decode logic to implement these instructions. Note CRUCLK to the CRU is inhibited during external instructions.

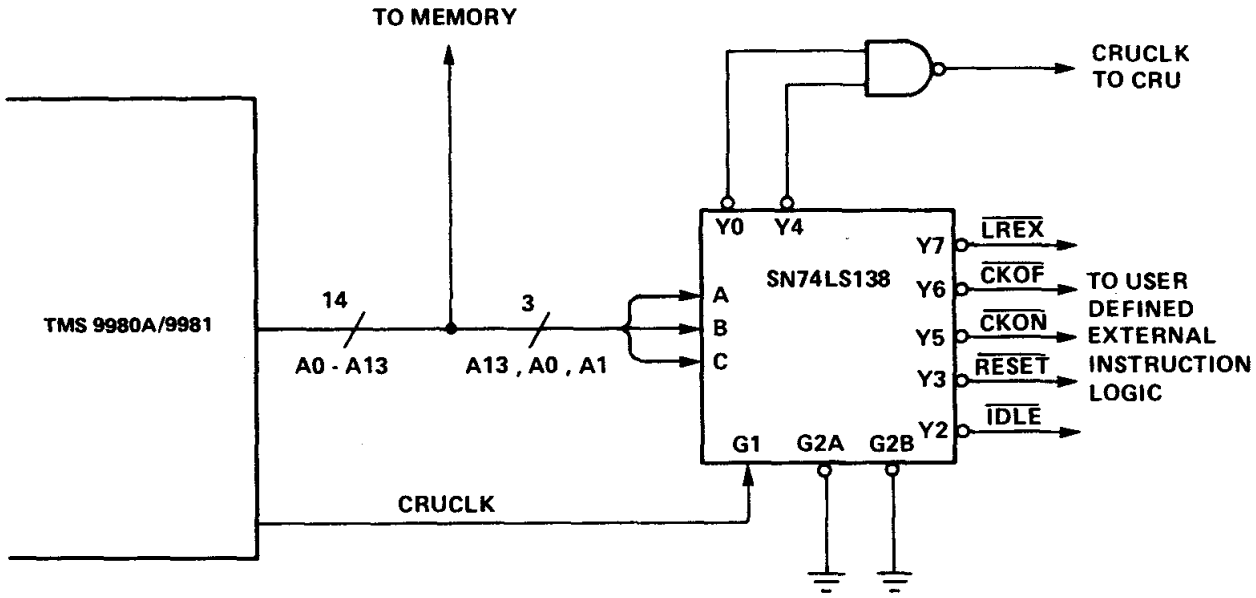


FIGURE 7 – EXTERNAL INSTRUCTION DECODE LOGIC

2.7 NON-MASKABLE INTERRUPTS

2.7.1 LOAD Function

The LOAD stimulus is an unmaskable interrupt that allows cold-start ROM loaders and front panels to be implemented for the TMS 9980A/TMS 9981. When the TMS 9980A/TMS 9981 decodes LOAD on IC0-IC2 lines, it initiates an interrupt sequence immediately following the instruction being executed. Memory location 3FFC is used to obtain the vector (WP and PC). The old PC, WP, and ST are loaded into the new workspace and the interrupt mask is set to 0000. Then the program execution resumes using the new PC and WP. Recognition of LOAD by the processor will also terminate the idle condition. External stimulus for LOAD must be held active (on IC0-IC2) for one instruction period by using IAQ signal.

2.7.2 RESET

When the TMS 9980A/TMS 9981 recognizes a RESET on IC0-IC2, it resets and inhibits $\overline{WE}$ and CRUCLK. Upon removal of the RESET code, the TMS 9980A/TMS 9981 initiates a level-zero interrupt sequence that acquires WP and PC from location 0000 and 0002, sets all status register bits to zero and starts execution. Recognition of RESET by the processor will also terminate an idle state. External stimulus for RESET must be held active for a minimum of three clock cycles.

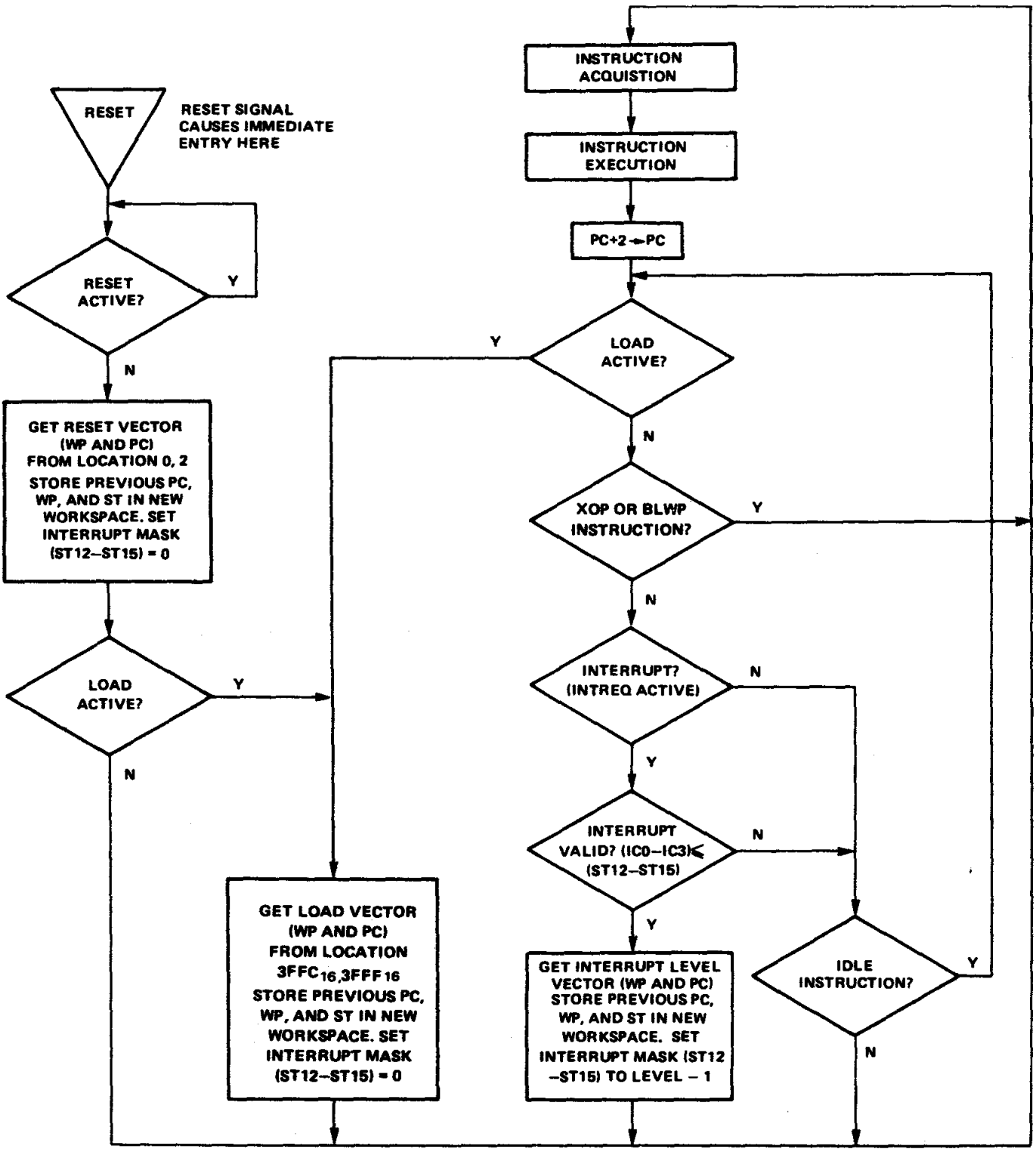


FIGURE 8 – TMS 9980A/TMS 9981 CPU FLOW CHART

2.8 TMS 9980A PIN DESCRIPTION

Table 2 defines the TMS 9980A pin assignments and describes the function of each pin.

TABLE 2
TMS 9980A PIN ASSIGNMENTS AND FUNCTIONS

SIGNATURE	PIN	I/O	DESCRIPTION
			ADDRESS BUS
A0 (MSB)	17	OUT	A0 through A13 comprise the address bus. This 3-state bus provides the memory-address vector to the external-memory system when MEMEN is active and I/O-bit addresses and external-instruction addresses to the I/O system when $\overline{\text{MEMEN}}$ is inactive. The address bus assumes the high-impedance state when HOLDA is active.
A1	16	OUT	
A2	15	OUT	
A3	14	OUT	
A4	13	OUT	
A5	12	OUT	
A6	11	OUT	
A7	10	OUT	
A8	9	OUT	
A9	8	OUT	
A10	7	OUT	
A11	6	OUT	
A12	5	OUT	
A13/CRUOUT	4	OUT	**CRUOUT** Serial I/O data appears on A13 when an LDCR, SBZ and SBO instruction is executed. This data should be sampled by the I/O interface logic when CRUCLK goes active (high). One bit of the external instruction code appears on A13 during external instruction execution.
			DATA BUS
D0 (MSB)	26	I/O	D0 through D7 comprise the bidirectional 3-state data bus. This bus transfers memory data to (when writing) and from (when reading) the external-memory system when $\overline{\text{MEMEN}}$ is active. The data bus assumes the high-impedance state when HOLDA is active.
D1	27	I/O	
D2	28	I/O	
D3	29	I/O	
D4	30	I/O	
D5	31	I/O	
D6	32	I/O	
D7 (LSB)	33	I/O	
			POWER SUPPLIES
V_{BB}	21		Supply voltage (−5V NOM)
V_{CC}	20		Supply voltage (5 V NOM)
V_{DD}	36		Supply voltage (12 V NOM)
V_{SS}	35		Ground reference
			CLOCKS
CKIN	34	IN	Clock In. A TTL compatible input used to generate the internal 4-phase clock. CKIN frequency is 4 times the desired system frequency.
$\overline{\phi 3}$	22	OUT	Clock phase 3 ($\phi 3$) inverted; used as a timing reference.
			BUS CONTROL
DBIN	18	OUT	Data bus in. When active (high), DBIN indicates that the TMS 9980A has disabled its output buffers to allow the memory to place memory-read data on the data bus during MEMEN. DBIN remains low in all other cases except when HOLDA is active at which time it is in the high-impedance state.

TMS 9980A PIN ASSIGNMENTS

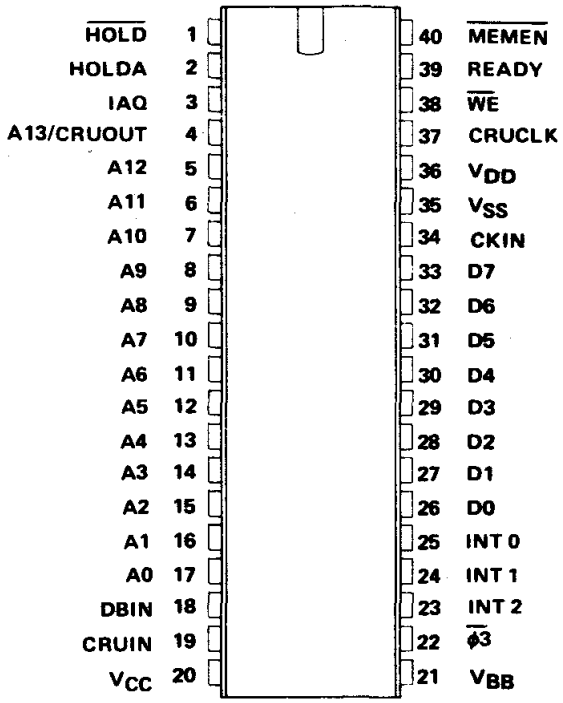


TABLE 2 (CONTINUED)

SIGNATURE	PIN	I/O	DESCRIPTION
$\overline{\text{MEMEN}}$	40	OUT	Memory enable. When active (low), $\overline{\text{MEMEN}}$ indicates that the address bus contains a memory address. When HOLDA is active, $\overline{\text{MEMEN}}$ is in the high impedance state.
$\overline{\text{WE}}$	38	OUT	Write enable. When active (low), $\overline{\text{WE}}$ indicates that memory-write data is available from the TMS 9980 to be written into memory. When HOLDA is active, $\overline{\text{WE}}$ is in the high-impedance state.
CRUCLK	37	OUT	CRU clock. When active (high), CRUCLK indicates that external interface logic should sample the output data on CRUOUT or should decode external instructions on A0, A1, A13.
CRUIN	19	IN	CRU data in. CRUIN, normally driven by 3-state or open-collector devices, receives input data from external interface logic. When the processor executes a STCR or TB instruction, it samples CRUIN for the level of the CRU input bit specified by the address bus (A2 through A12).
INT2	23	IN	Interrupt code. Refer to Section 2.2 for detailed description.
INT1	24	IN	
INT0	25	IN	
			MEMORY CONTROL
$\overline{\text{HOLD}}$	1	IN	Hold. When active (low), $\overline{\text{HOLD}}$ indicates to the processor that an external controller (e.g., DMA device) desires to utilize the address and data buses to transfer data to or from memory. The TMS 9980A enters the hold state following a hold signal when it has completed its present memory cycle.* The processor then places the address and data buses in the high-impedance state (along with $\overline{\text{WE}}$, $\overline{\text{MEMEN}}$, and DBIN) and responds with a hold-acknowledge signal (HOLDA). When $\overline{\text{HOLD}}$ is removed, the processor returns to normal operation.
HOLDA	2	OUT	Hold acknowledge. When active (high), HOLDA indicates that the processor is in the hold state and the address and data buses and memory control outputs ($\overline{\text{WE}}$, $\overline{\text{MEMEN}}$, and DBIN) are in the high-impedance state.
READY	39	IN	Ready. When active (high), READY indicates that memory will be ready to read or write during the next clock cycle. When not-ready is indicated during a memory operation, the TMS 9980A enters a wait state and suspends internal operation until the memory systems indicated ready.
			TIMING AND CONTROL
IAQ	3	OUT	Instruction acquisition. IAQ is active (high) during any memory cycle when the TMS 9980A is acquiring an instruction. IAQ can be used to detect illegal op codes. It may also be used to synchronize LOAD stimulus.

*If the cycle following the present memory cycle is also a memory cycle it, too, is completed before TMS 9980 enters hold state.

2.9 TMS 9981 PIN DESCRIPTION

Table 3 defines the TMS 9981 pin assignments and describes the function of each pin.

TABLE 3
TMS 9981 PIN ASSIGNMENTS AND FUNCTIONS

SIGNATURE	PIN	I/O	DESCRIPTION
			ADDRESS BUS
A0(MSB)	17	OUT	A0 through A13 comprise the address bus. This 3-state bus provides the memory-address vector to the external-memory system when MEMEN is active and I/O-bit addresses and external-instruction addresses to the I/O system when $\overline{\text{MEMEN}}$ is inactive. The address bus assumes the high-impedance state when HOLDA is active.
A1	16	OUT	
A2	15	OUT	
A3	14	OUT	
A4	13	OUT	
A5	12	OUT	
A6	11	OUT	
A7	10	OUT	
A8	9	OUT	
A9	8	OUT	
A10	7	OUT	
A11	6	OUT	
A12	5	OUT	
A13/CRUOUT	4	OUT	**CRUOUT** Serial I/O data appears on A13 when an LDCR, SBZ and SBO instruction is executed. This data should be sampled by the I/O interface logic when CRUCLK goes active (high). One bit of the external instruction code appears on A13 during external instruction execution.
			DATA BUS
D0 (MSB)	25	I/O	D0 through D7 comprise the bidirectional 3-state data bus. This bus transfers memory data to (when writing) and from (when reading) the external-memory system when $\overline{\text{MEMEN}}$ is active. The data bus assumes the high-impedance state when HOLDA is active.
D1	26	I/O	
D2	27	I/O	
D3	28	I/O	
D4	29	I/O	
D5	30	I/O	
D6	31	I/O	
D7 (LSB)	32	I/O	
			POWER SUPPLIES
V_{CC}	20		Supply voltage (5 V NOM)
V_{DD}	36		Supply voltage (12 V NOM)
V_{SS}	35		Ground reference
			CLOCKS
CKIN	34	IN	Clock In and Oscillator Out. These pins may be used in either of two modes to generate the internal 4-phase clock. In mode 1 a crystal of 4 times the desired system frequency is connected between CKIN and OSCOUT (see Figure 13). In mode 2 OSCOUT is left floating and CKIN is driven by a TTL compatible source whose frequency is 4 times the desired system frequency.
OSCOUT	33	OUT	
$\overline{\phi 3}$	21	OUT	Clock phase 3 ($\phi 3$) inverted; used as a timing reference.
			BUS CONTROL
DBIN	18	OUT	Data bus in. When active (high), DBIN indicates that the TMS 9981 has disabled its output buffers to allow the memory to place memory-read data on the data bus during $\overline{\text{MEMEN}}$. DBIN remains low in all other cases except when HOLDA is active at which time it is in the high-impedance state.

TMS 9981 PIN ASSIGNMENTS

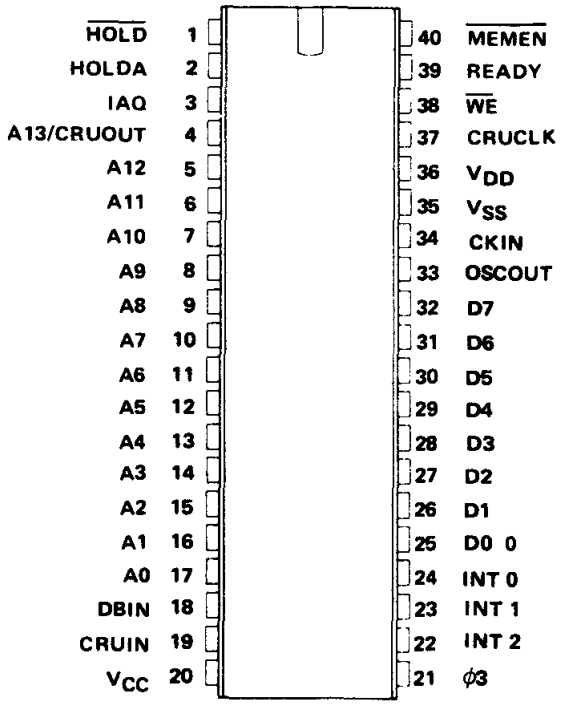


TABLE 3 (CONTINUED)

SIGNATURE	PIN	I/O	DESCRIPTION
$\overline{\text{MEMEN}}$	40	OUT	Memory enable. When active (low), $\overline{\text{MEMEN}}$ indicates that the address bus contains a memory address. When HOLDA is active, $\overline{\text{MEMEN}}$ is in the high-impedance state.
$\overline{\text{WE}}$	38	OUT	Write enable. When active (low), $\overline{\text{WE}}$ indicates that memory-write data is available from the TMS 9981 to be written into memory. When HOLDA is active, $\overline{\text{WE}}$ is in the high-impedance state.
CRUCLK	37	OUT	CRU clock. When active (high), CRUCLK indicates that external interface logic should sample the output data on CRUOUT or should decode external instructions on A0, A1, A13.
CRUIN	19	IN	CRU data in. CRUIN, normally driven by 3-state or open-collector devices, receives input data from external interface logic. When the processor executes a STCR or TB instruction, it samples CRUIN for the level of the CRU input bit specified by the address bus (A2 through A12).
INT2	22	IN	Interrupt code. Refer to Section 2.2 for detailed description.
INT1	23	IN	
INT0	24	IN	
$\overline{\text{HOLD}}$	1	IN	**MEMORY CONTROL** Hold. When active (low), HOLD indicates to the processor that an external controller (e.g., DMA device) desires to utilize the address and data buses to transfer data to or from memory. The TMS 9981 enters the hold state following a hold signal when it has completed its present memory cycle.* The processor then places the address and data buses in the high-impedance state (along with $\overline{\text{WE}}$, $\overline{\text{MEMEN}}$, and DBIN) and responds with a hold-acknowledge signal (HOLDA). When $\overline{\text{HOLD}}$ is removed, the processor returns to normal operation.
HOLDA	2	OUT	Hold acknowledge. When active (high), HOLDA indicates that the processor is in the hold state and the address and data buses and memory control outputs ($\overline{\text{WE}}$, $\overline{\text{MEMEN}}$, and DBIN) are in the high-impedance state.
READY	39	IN	Ready. When active (high), READY indicates that memory will be ready to read or write during the next clock cycle. When not-ready is indicated during a memory operation, the TMS 9981 enters a wait state and suspends internal operation until the memory systems indicated ready.
IAQ	3	OUT	**TIMING AND CONTROL** Instruction acquisition. IAQ is active (high) during any memory cycle when the TMS 9981 is acquiring an instruction. IAQ can be used to detect illegal op codes. It may also be used to synchronize LOAD stimulus.

*If the cycle following the present memory cycle is also a memory cycle it, too, is completed before TMS 9981 enters hold state.

2.10 TIMING

2.10.1 Memory

Basic memory read and write cycles are shown in Figures 9a and 9b. Figure 9a shows a read and a write cycle with no wait states while Figure 9b shows a read and a write cycle for a memory requiring one wait state.

$\overline{\text{MEMEN}}$ goes active (low) during each memory cycle. At the same time that $\overline{\text{MEMEN}}$ is active, the memory address appears on the address bits A0 through A13. Since the TMS 9980A/TMS 9981 has an 8-bit data bus, every memory operation consists of two consecutive memory cycles. Address bit A13 is 0 for the first of the two cycles and goes to 1 for the second. If the cycle is a memory-read cycle, DBIN will go active (high) at the same time $\overline{\text{MEMEN}}$ and A0 through A13 become valid. The memory-write ($\overline{\text{WE}}$) signal remains inactive during a read cycle.

The READY signal allows extended memory cycle as shown in Figure 9b.

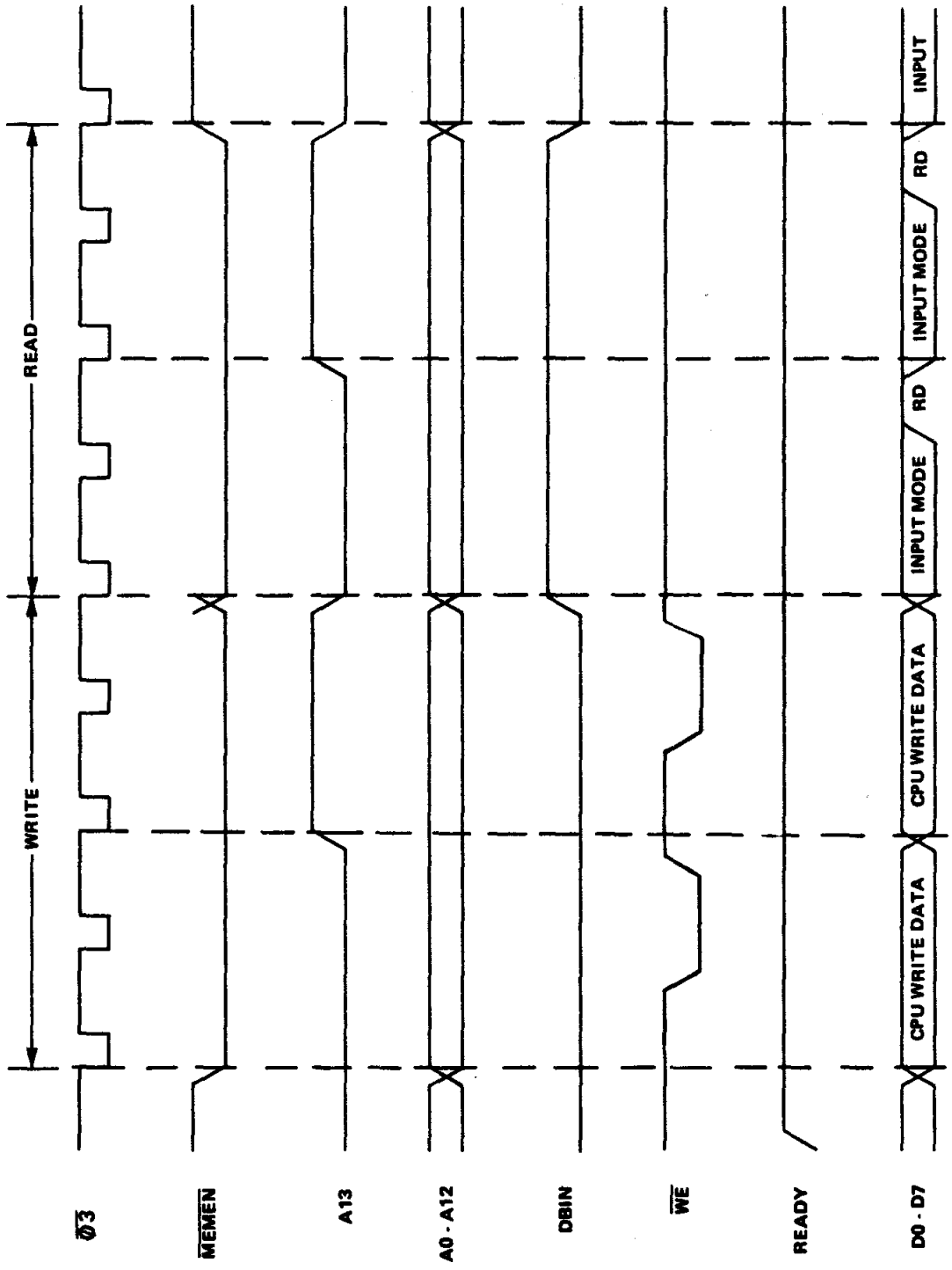


FIGURE 9a – TMS 9980A/TMS 9981 MEMORY BUS TIMING (NO WAIT STATES)

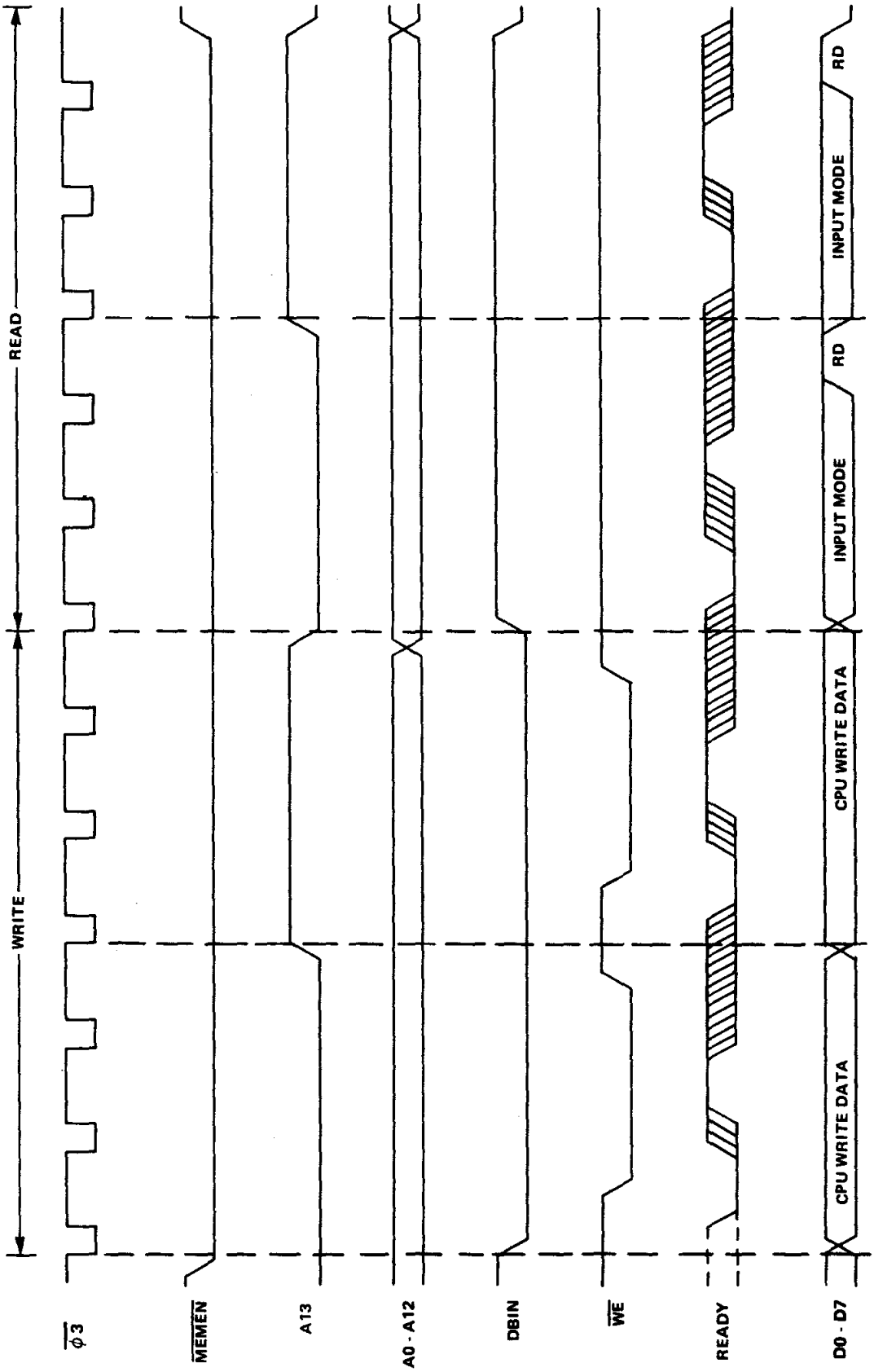


FIGURE 9b – TMS 9980A/TMS 9981 MEMORY BUS TIMING (ONE WAIT STATE)

At the end of the read cycle, $\overline{\text{MEMEN}}$ and DBIN go inactive (high and low respectively). The address bus also changes at this time, however, the data bus remains in the input mode for one clock cycle after the read cycle.

A write cycle is similar to read cycle except that $\overline{\text{WE}}$ goes active (low) as shown and valid write data appears on the data bus at the same time the address appears.

2.10.2 HOLD

Other interfaces may utilize the TMS 9980A/TMS 9981 memory bus by using the hold operation (illustrated in Figure 10) of the TMS 9980A/TMS 9981. When $\overline{\text{HOLD}}$ is active (low), the TMS 9980A/TMS 9981 enters the hold state at the next available non-memory cycle clock period. When the TMS 9980A/TMS 9981 has entered the hold state HOLDA goes active (high), A0 through A13, D0 through D7, DBIN, $\overline{\text{MEMEN}}$, and $\overline{\text{WE}}$ go into high-impedance state to allow other devices to use the memory buses. When $\overline{\text{HOLD}}$ goes inactive, TMS 9980A/TMS 9981 resumes processing as shown. Considering that there can be a maximum of 6 consecutive memory operations, the maximum delay between $\overline{\text{HOLD}}$ going active to HOLDA going active (high) could be $t_{c(\phi)}$ (for set up) + (12 + 6 W) $t_{c(\phi)}$ (delay for HOLDA), where W is the number of wait states per memory cycle and $t_{c(\phi)}$ is the clock cycle time. If hold occurs during a CRU operation, the TMS 9980A/TMS 9981 uses an extra clock cycle (after the removal of the $\overline{\text{HOLD}}$ signal) to reassert the CRU address providing the normal setup times for the CRU bit transfer that was interrupted.

2.10.3 CRU

CRU interface timing is shown in Figure 11. The timing for transferring two bits out and one bit in is shown. These transfers would occur during the execution of a CRU instruction. The other cycles of the instruction execution are not illustrated. To output a CRU bit, the CRU-bit address is placed on the address bus A2 through A12 and the actual bit data on A13. During the second clock cycle a CRU pulse is supplied by CRUCLK. This process is repeated until the number of bits specified by the instruction are completed.

The CRU input operation is similar in that the bit address appears on A2 through A12. During the subsequent cycle, the TMS 9980A/TMS 9981 accepts the bit input data as shown. No CRUCLK pulses occur during a CRU input operation.

2.10.4 Interrupt Code (IC0-IC2)

The TMS 9980A/TMS 9981 uses 4 phase clock (ϕ1, ϕ2, ϕ3, and ϕ4) for timing and control of the internal operations. IC0-IC2 are sampled during ϕ4 and then during ϕ2.

If these two successive samples are equal, the code is accepted and latched for internal use on the subsequent ϕ1. In systems with simple interrupt structures this allows the interrupt code to change asynchronously without the TMS 9980A/TMS 9981 accepting erroneous codes. Figure 3 shows systems with a single level of external interrupt implemented that would require no external timing. When implementing multiple external interrupts, as in the bottom diagram of Figure 3, external synchronization of interrupt requests is required. See Figure 12 for a timing diagram. In systems with more than one external interrupt, the interrupts should be synchronized with the $\overline{\phi 3}$ output of the TMS 9980A/TMS 9981 to avoid code transitions on successive sample cycles. This synchronization ensures that the TMS 9980A/TMS 9981 will service only the proper active interrupt level.

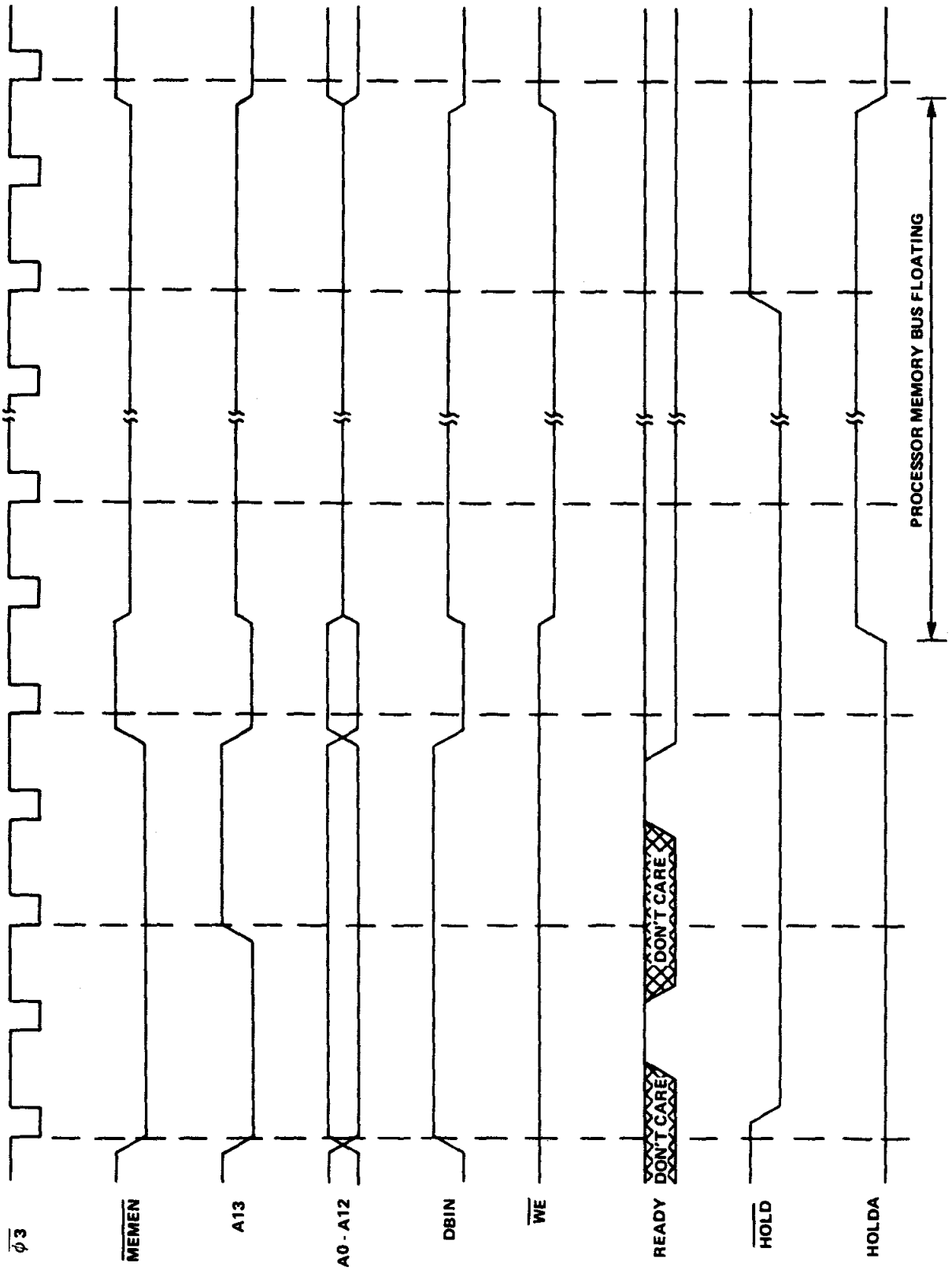


FIGURE 10 – TMS 9980A/TMS 9981 HOLD TIMING

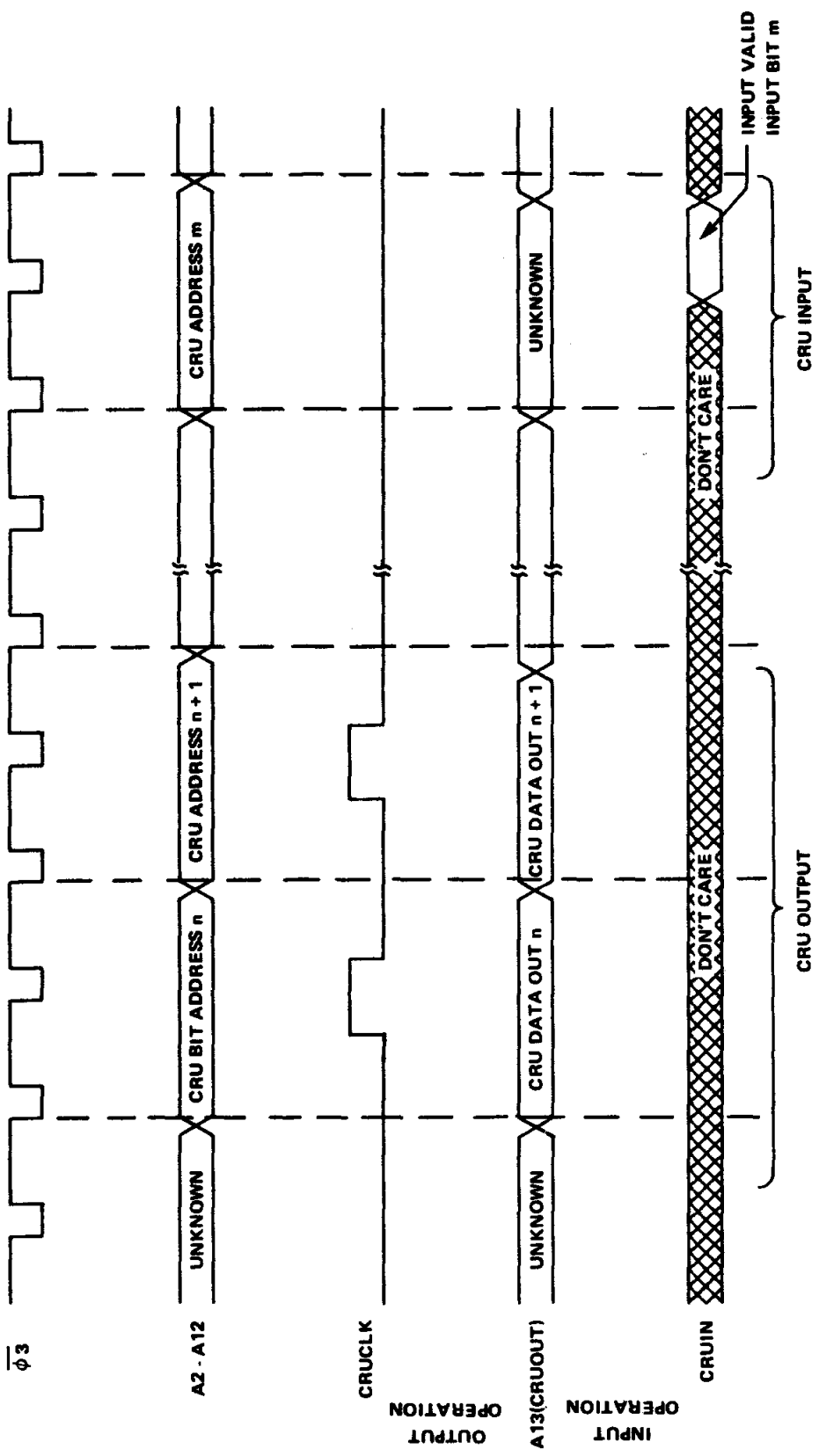


FIGURE 11 – TMS 9980A/TMS 9981 CRU INTERFACE TIMING

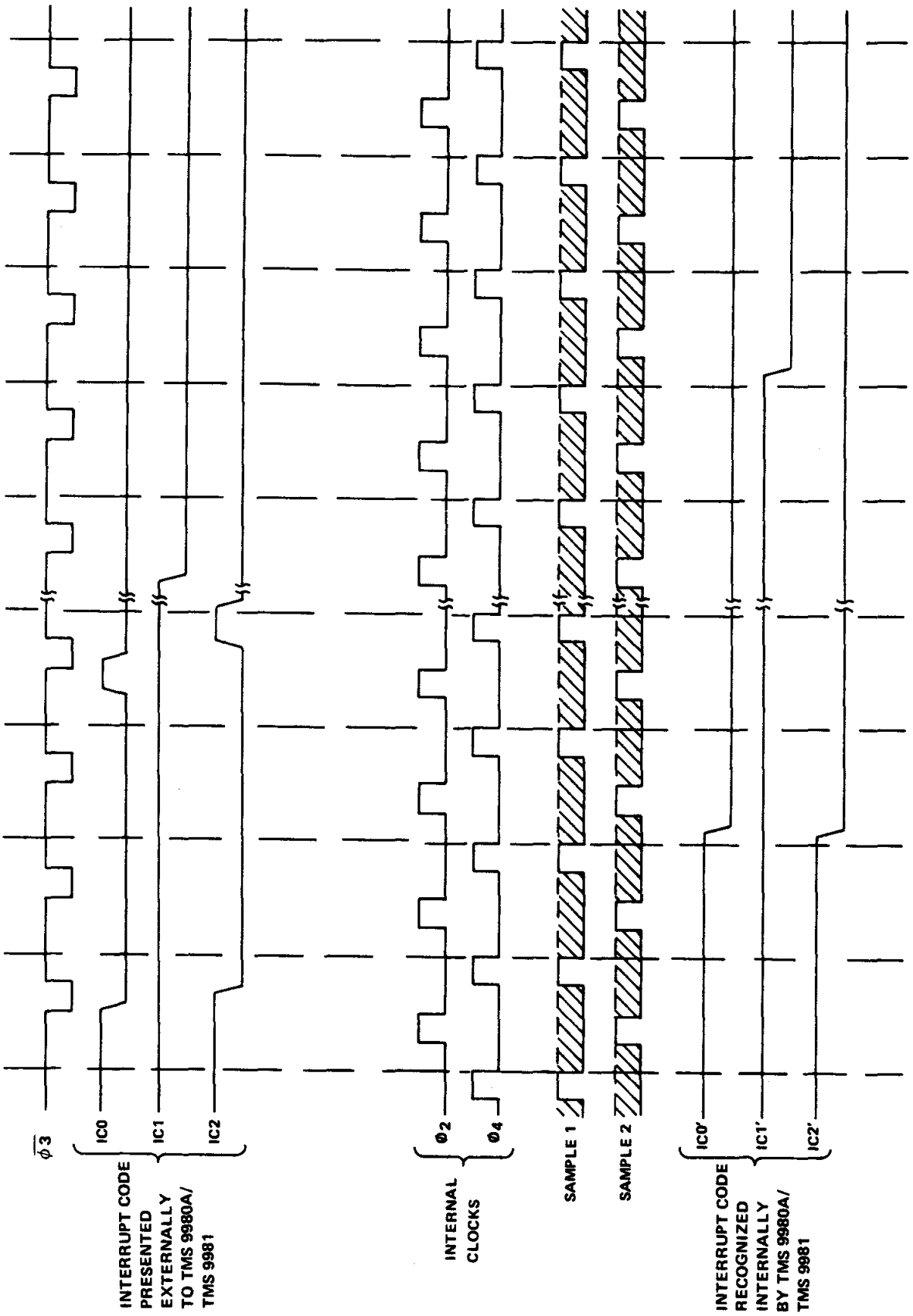


FIGURE 12 – INTERRUPT CODE TIMING

3. TMS 9980A/TMS 9981 INSTRUCTION SET

3.1 DEFINITION

The instruction set of the TMS 9980A/TMS 9981 is identical to that of TMS 9900. Each instruction of this set performs one of the following operations:

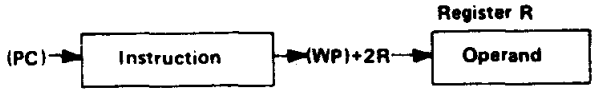
- Arithmetic, logical, comparison, or manipulation operations on data
- Loading or storage of internal registers (program counter, workspace pointer, or status)
- Data transfer between memory and external devices via the CRU
- Control functions.

3.2 ADDRESSING MODES

TMS 9980A/TMS 9981 instructions contain a variety of available modes for addressing random-memory data (e.g., program parameters and flags), or formatted memory data (character strings, data lists, etc.). The following figures graphically describe the derivation of the effective address for each addressing mode. The applicability of addressing modes to particular instructions is described in Section 3.5 along with the description of the operations performed by the instruction. The symbols following the names of the addressing modes [R, *R, *R+, @ LABEL, or @ TABLE (R)] are the general forms used by TMS 9980A/TMS 9981 assemblers to select the addressing mode for register R. Note that the TMS 9980A/TMS 9981 users use the same assembler and other software support packages as the ones used by the TMS 9900 users.

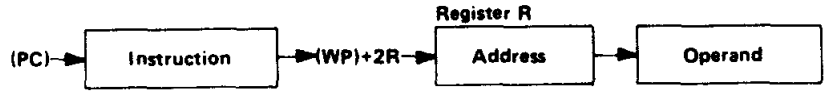
3.2.1 WORKSPACE REGISTER ADDRESSING R

Workspace Register R contains the operand.



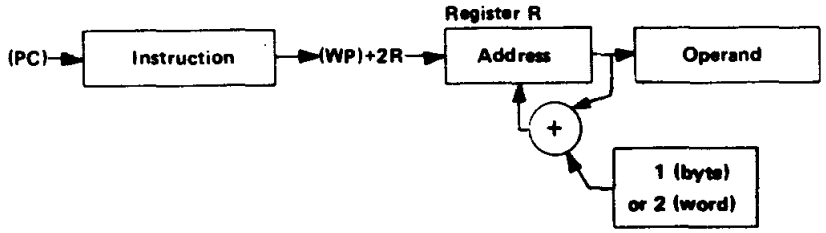
3.2.2 WORKSPACE REGISTER INDIRECT ADDRESSING *R

Workspace Register R contains the address of the operand.



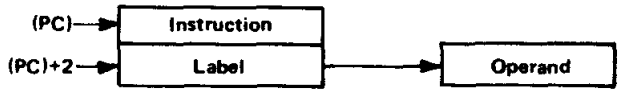
3.2.3 WORKSPACE REGISTER INDIRECT AUTO INCREMENT ADDRESSING *R+

Workspace Register R contains the address of the operand. After acquiring the operand, the contents of workspace register R are incremented.



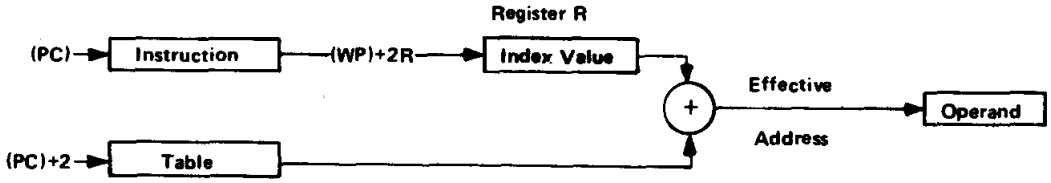
3.2.4 SYMBOLIC (DIRECT) ADDRESSING @LABEL

The word following the instruction contains the address of the operand.



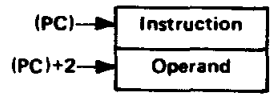
3.2.5 INDEXED ADDRESSING @ TABLE (R)

The word following the instruction contains the base address. Workspace register R contains the index value. The sum of the base address and the index value results in the effective address of the operand.



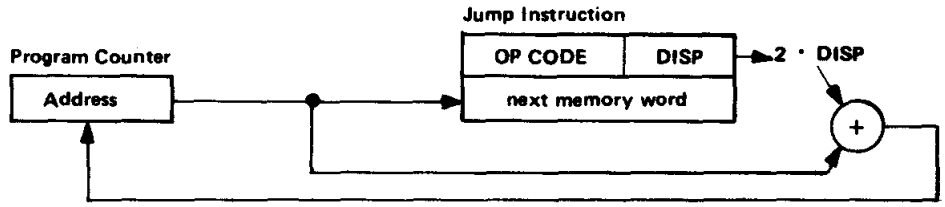
3.2.6 IMMEDIATE ADDRESSING

The word following the instruction contains the operand.



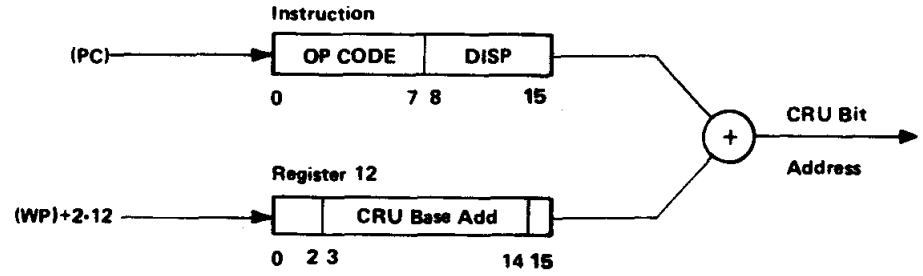
3.2.7 PROGRAM COUNTER RELATIVE ADDRESSING

The 8-bit signed displacement in the right byte (bits 8 through 15) of the instruction is multiplied by 2 and added to the updated contents of the program counter. The result is placed in the PC.



3.2.8 CRU RELATIVE ADDRESSING

The 8-bit signed displacement in the right byte of the instruction is added to the CRU base address (bits 3 through 14 of the workspace register 12). The result is the CRU address of the selected CRU bit.



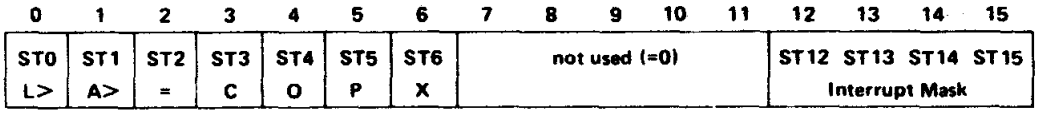
3.3 TERMS AND DEFINITIONS

The following terms are used in describing the instructions of the TMS 9980A/TMS 9981.

TERM	DEFINITION
B	Byte indicator (1=byte, 0 = word)
C	Bit count
D	Destination address register
DA	Destination address
IOP	Immediate operand
LSB(n)	Least significant (right most) bit of (n)
MSB(n)	Most significant (left most) bit of (n)
N	Don't care
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
T_D	Destination address modifier
T_S	Source address modifier
W	Workspace register
WRn	Workspace register n
(n)	Contents of n
a → b	a is transferred to b
|n|	Absolute value of n
+	Arithmetic addition
−	Arithmetic subtraction
AND	Logical AND
OR	Logical OR
⊕	Logical exclusive OR
$\overline{n}$	Logical complement of n

3.4 STATUS REGISTER

The status register contains the interrupt mask level and information pertaining to the instruction operation.



BIT	NAME	INSTRUCTION	CONDITION TO SET BIT TO 1
ST0	LOGICAL GREATER THAN	C,CB	If MSB(SA) = 1 and MSB(DA) = 0, or if MSB(SA) = MSB(DA) and MSB of [(DA) − (SA)] = 1
		CI	If MSB(W) = 1 and MSB of IOP = 0, or if MSB(W) = MSB of IOP and MSB of [IOP − (W)] = 1
		ABS	If (SA) ≠ 0
		All Others	If result ≠ 0
ST1	ARITHMETIC GREATER THAN	C,CB	If MSB(SA) = 0 and MSB(DA) = 1, or if MSB(SA) = MSB(DA) and MSB of [(DA) − (SA)] = 1
		CI	If MSB(W) = 0 and MSB of IOP = 1, or if MSB(W) = MSB of IOP and MSB of [IOP − (W)] = 1
		ABS	If MSB(SA) = 0 and (SA) ≠ 0
		All Others	If MSB of result = 0 and result ≠ 0

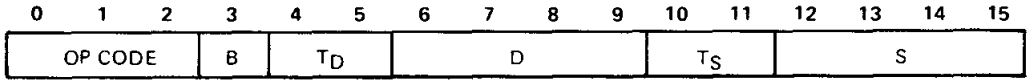
– Continued

BIT	NAME	INSTRUCTION	CONDITION TO SET BIT TO 1
ST2	EQUAL	C, CB	If (SA) = (DA)
		CI	If (W) = IOP
		COC	If (SA) and $\overline{(DA)}$ = 0
		CZC	If (SA) and (DA) = 0
		TB	If CRUIN = 1
		ABS	If (SA) = 0
		All others	If result = 0
ST3	CARRY	A, AB, ABS, AI, DEC, DECT, INC, INCT, NEG, S, SB	If CARRY OUT = 1
		SLA, SRA, SRC, SRL	If last bit shifted out = 1
ST4	OVERFLOW	A, AB	If MSB(SA) = MSB(DA) and MSB of result ≠ MSB(DA)
		AI	If MSB(W) = MSB of IOP and MSB of result ≠ MSB(W)
		S, SB	If MSB(SA) ≠ MSB(DA) and MSB of result ≠ MSB(DA)
		DEC, DECT	If MSB(SA) = 1 and MSB of result = 0
		INC, INCT	If MSB(SA) = 0 and MSB of result = 1
		SLA	If MSB changes during shift
		DIV	If MSB(SA) = 0 and MSB(DA) = 1, or if MSB(SA) = MSB(DA) and MSB of [(DA) − (SA)] = 0
		ABS, NEG	If (SA) = 8000_{16}
ST5	PARITY	CB, MOVB	If (SA) has odd number of 1's
		LDCR, STCR	If $1 \leq C \leq 8$ and (SA) has odd number of 1's
		AB, SB, SOCB, SZCB	If result has odd number of 1's
ST6	XOP	XOP	If XOP instruction is executed
ST12–ST15	INTERRUPT MASK	LIMI	If corresponding bit of IOP is 1
		RTWP	If corresponding bit of WR15 is 1

3.5 INSTRUCTIONS

3.5.1 Dual Operand Instructions with Multiple Addressing Modes for Source and Destination Operand

General format:



If B = 1 the operands are bytes and the operand addresses are byte addresses. If B = 0 the operands are words and the operand addresses are word addresses.

The addressing mode for each operand is determined by the T field of that operand.

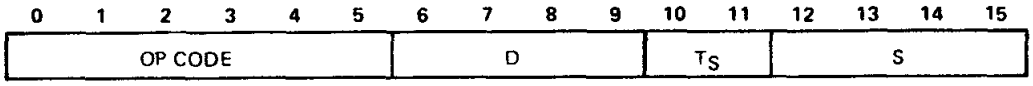
T_S OR T_D	S OR D	ADDRESSING MODE	NOTES
00	0, 1, . . . 15	Workspace register	1
01	0, 1, . . . 15	Workspace register indirect	
10	0	Symbolic	4
10	1, 2, . . . 15	Indexed	2,4
11	0, 1, . . . 15	Workspace register indirect auto-increment	3

NOTES:
1. When a workspace register is the operand of a byte instruction (bit 3 = 1), the left byte (bits 0 through 7) is the operand and the right byte (bits 8 through 15) is unchanged.
2. Workspace register 0 may not be used for indexing.
3. The workspace register is incremented by 1 for byte instructions (bit 3 = 1) and is incremented by 2 for word instructions (bit 3 = 0).
4. When $T_S = T_D = 10$, two words are required in addition to the instruction word. The first word is the source operand base address and the second word is the destination operand base address.

MNEMONIC	OP CODE 0 1 2	B 3	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
A	1 0 1	0	Add	Yes	0-4	(SA)+(DA) → (DA)
AB	1 0 1	1	Add bytes	Yes	0-5	(SA)+(DA) → (DA)
C	1 0 0	0	Compare	No	0-2	Compare (SA) to (DA) and set appropriate status bits
CB	1 0 0	1	Compare bytes	No	0-2,5	Compare (SA) to (DA) and set appropriate status bits
S	0 1 1	0	Subtract	Yes	0-4	(DA) – (SA) → (DA)
SB	0 1 1	1	Subtract bytes	Yes	0-5	(DA) – (SA) → (DA)
SOC	1 1 1	0	Set ones corresponding	Yes	0-2	(DA) OR (SA) → (DA)
SOCB	1 1 1	1	Set ones corresponding bytes	Yes	0-2,5	(DA) OR (SA) → (DA)
SZC	0 1 0	0	Set zeroes corresponding	Yes	0-2	(DA) AND ($\overline{SA}$) → (DA)
SZCB	0 1 0	1	Set zeroes corresponding bytes	Yes	0-2,5	(DA) AND ($\overline{SA}$) → (DA)
MOV	1 1 0	0	Move	Yes	0-2	(SA) → (DA)
MOVB	1 1 0	1	Move bytes	Yes	0-2,5	(SA) → (DA)

3.5.2 Dual Operand Instructions with Multiple Addressing Modes for the Source Operand and Workspace Register Addressing for the Destination

General format:



The addressing mode for the source operand is determined by the T_S field.

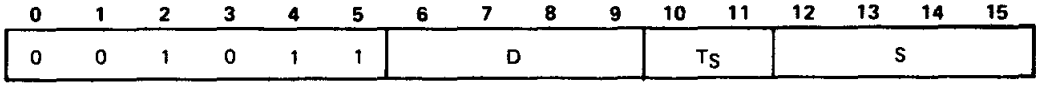
T_S	S	ADDRESSING MODE	NOTES
00	0, 1, . . . 15	Workspace register	
01	0, 1, . . . 15	Workspace register indirect	
10	0	Symbolic	
10	1, 2, . . . 15	Indexed	1
11	0, 1, . . . 15	Workspace register indirect auto increment	2

NOTES: 1. Workspace register 0 may not be used for indexing.
2. The workspace register is incremented by 2.

MNEMONIC	OP CODE 0 1 2 3 4 5	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
COC	0 0 1 0 0 0	Compare ones corresponding	No	2	Test (D) to determine if 1's are in each bit position where 1's are in (SA). If so, set ST2.
CZC	0 0 1 0 0 1	Compare zeros corresponding	No	2	Test (D) to determine if 0's are in each bit position where 1's are in (SA). If so, set ST2.
XOR	0 0 1 0 1 0	Exclusive OR	Yes	0-2	(D) ⊕ (SA) → (D)
MPY	0 0 1 1 1 0	Multiply	No		Multiply unsigned (D) by unsigned (SA) and place unsigned 32-bit product in D (most significant) and D+1 (least significant). If WR15 is D, the next word in memory after WR15 will be used for the least significant half of the product.
DIV	0 0 1 1 1 1	Divide	No	4	If unsigned (SA) is less than or equal to unsigned (D), perform no operation and set ST4. Otherwise, divide unsigned (D) and (D+1) by unsigned (SA). Quotient → (D), remainder → (D+1). If D = 15, the next word in memory after WR 15 will be used for the remainder.

3.5.3 Extended Operation (XOP) Instruction

General format:



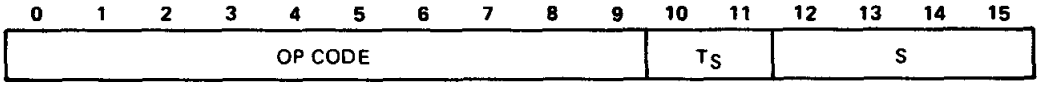
The T_S and S fields provide multiple mode addressing capability for the source operand. When the XOP is executed, ST6 is set and the following transfers occur:

$(40_{16} + 4D) \rightarrow (WP)$
$(42_{16} + 4D) \rightarrow (PC)$
SA → (new WR11)
(old WP) → (new WR13)
(old PC) → (new WR14)
(old ST) → (new WR15)

The TMS 9980A/TMS 9981 tests for reset and load but does not test for interrupt requests (INTREQ) upon completion of the XOP instruction.

3.5.4 Single Operand Instructions

General format:



The T_S and S fields provide multiple mode addressing capability for the source operand.

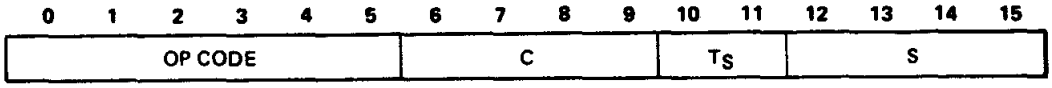
MNEMONIC	OP CODE 0 1 2 3 4 5 6 7 8 9	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
B	0 0 0 0 0 1 0 0 0 1	Branch	No	—	SA → (PC)
BL	0 0 0 0 0 1 1 0 1 0	Branch and link	No	—	(PC) → (WR11); SA → (PC)
BLWP	0 0 0 0 0 1 0 0 0 0	Branch and load workspace pointer	No	—	(SA) →(WP); (SA+2) →(PC); (old WP) → (new WR13); (old PC) → (new WR14); (old ST) → (new WR15); The TMS 9980A/TMS 9981 tests for reset and load, but does not test for interrupts upon completion of the BLWP instruction.
CLR	0 0 0 0 0 1 0 0 1 1	Clear operand	No	—	0 → (SA)
SETO	0 0 0 0 0 1 1 1 0 0	Set to ones	No	—	$FFFF_{16}$ → (SA)
INV	0 0 0 0 0 1 0 1 0 1	Invert	Yes	0-2	$\overline{(SA)}$ → (SA)
NEG	0 0 0 0 0 1 0 1 0 0	Negate	Yes	0-4	−(SA) → (SA)
ABS	0 0 0 0 0 1 1 1 0 1	Absolute value*	No	0-4	|(SA)| → (SA)
SWPB	0 0 0 0 0 1 1 0 1 1	Swap bytes	No	—	(SA), bits 0 thru 7 → (SA), bits 8 thru 15; (SA), bits 8 thru 15 → (SA), bits 0 thru 7.
INC	0 0 0 0 0 1 0 1 1 0	Increment	Yes	0-4	(SA) + 1 → (SA)
INCT	0 0 0 0 0 1 0 1 1 1	Increment by two	Yes	0-4	(SA) + 2 → (SA)
DEC	0 0 0 0 0 1 1 0 0 0	Decrement	Yes	0-4	(SA) − 1 → (SA)
DECT	0 0 0 0 0 1 1 0 0 1	Decrement by two	Yes	0-4	(SA) − 2 → (SA)
X†	0 0 0 0 0 1 0 0 1 0	Execute	No	—	Execute the instruction at SA.

*Operand is compared to zero for status bit.

†If additional memory words for the execute instruction are required to define the operands of the instruction located at SA, these words will be accessed from PC and the PC will be updated accordingly. The instruction acquisition signal (IAQ) will not be true when the TMS 9900 accesses the instruction at SA. Status bits are affected in the normal manner for the instruction executed.

3.5.5 CRU Multiple-Bit Instructions

General format:



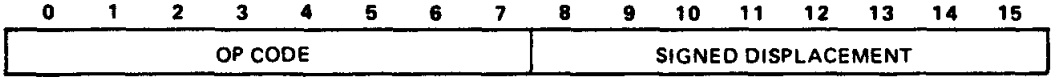
The C field specifies the number of bits to be transferred. If C = 0, 16 bits will be transferred. The CRU base registr (WR12, bits 4 through 14) defines the starting CRU bit address. The bits are transferred serially and the CRU address is incremented with each bit transfer, although the contents of WR12 is not affected. T_S and S provide multiple mode addressing capability for the source operand. If 8 or fewer bits are transferred (C = 1 through 8), the source address is a byte address. If 9 or more bits are transferred (C = 0, 9 through 15), the source address is a word address. If the source is addressed in the workspace register indirect auto increment mode, the workspace register is incremented by 1 if C = 1 through 8, and is incremented by 2 otherwise.

MNEMONIC	OP CODE 0 1 2 3 4 5	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
LDCR	0 0 1 1 0 0	Load communcation register	Yes	0-2,5†	Beginning with LSB of (SA), transfer the specified number of bits from (SA) to the CRU.
STCR	0 0 1 1 0 1	Store communcation register	Yes	0-2,5†	Beginning with LSB of (SA), transfer the specified number of bits from the CRU to (SA). Load unfilled bit positions with 0.

†ST5 is affected only if $1 \leq C \leq 8$.

3.5.6 CRU Single-Bit Instructions

General format:

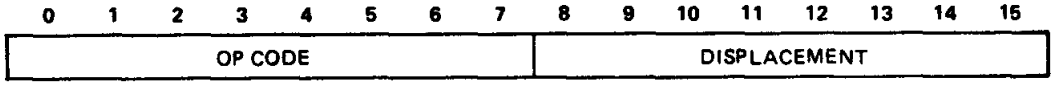


CRU relative addressing is used to address the selected CRU bit.

MNEMONIC	OP CODE 0 1 2 3 4 5 6 7	MEANING	STATUS BITS AFFECTED	DESCRIPTION
SBO	0 0 0 1 1 1 0 1	Set bit to one	–	Set the selected CRU output bit to 1.
SBZ	0 0 0 1 1 1 1 0	Set bit to zero	–	Set the selected CRU output bit to 0.
TB	0 0 0 1 1 1 1 1	Test bit	2	If the selected CRU input bit = 1, set ST2.

3.5.7 Jump Instructions

General format:

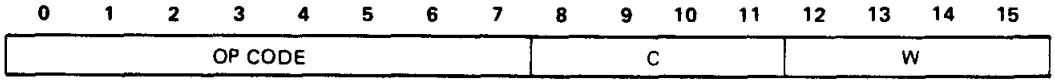


Jump instructions cause the PC to be loaded with the value selected by PC relative addressing if the bits of ST are at specified values. Otherwise, no operation occurs and the next instruction is executed since PC points to the next instruction. The displacement field is a word count to be added to PC. Thus, the jump instruction has a range of –128 to 127 words from memory-word address following the jump instruction. No ST bits are affected by jump instructions.

MNEMONIC	OP CODE 0 1 2 3 4 5 6 7	MEANING	ST CONDITION TO LOAD PC
JEQ	0 0 0 1 0 0 1 1	Jump equal	ST2 = 1
JGT	0 0 0 1 0 1 0 1	Jump greater than	ST1 = 1
JH	0 0 0 1 1 0 1 1	Jump high	ST0 = 1 and ST2 = 0
JHE	0 0 0 1 0 1 0 0	Jump high or equal	ST0 = 1 or ST2 = 1
JL	0 0 0 1 1 0 1 0	Jump low	ST0 = 0 and ST2 = 0
JLE	0 0 0 1 0 0 1 0	Jump low or equal	ST0 = 0 or ST2 = 1
JLT	0 0 0 1 0 0 0 1	Jump less than	ST1 = 0 and ST2 = 0
JMP	0 0 0 1 0 0 0 0	Jump unconditional	unconditional
JNC	0 0 0 1 0 1 1 1	Jump no carry	ST3 = 0
JNE	0 0 0 1 0 1 1 0	Jump not equal	ST2 = 0
JNO	0 0 0 1 1 0 0 1	Jump no overflow	ST4 = 0
JOC	0 0 0 1 1 0 0 0	Jump on carry	ST3 = 1
JOP	0 0 0 1 1 1 0 0	Jump odd parity	ST5 = 1

3.5.8 Shift Instructions

General format:

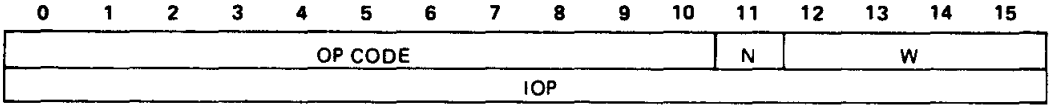


If C = 0, bits 12 through 15 of WR0 contain the shift count. If C = 0 and bits 12 through 15 of WR0 = 0, the shift count is 16.

MNEMONIC	OP CODE 0 1 2 3 4 5 6 7	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
SLA	0 0 0 0 1 0 1 0	Shift left arithmetic	Yes	0-4	Shift (W) left. Fill vacated bit positions with 0.
SRA	0 0 0 0 1 0 0 0	Shift right arithmetic	Yes	0-3	Shift (W) right. Fill vacated bit positions with original MSB of (W).
SRC	0 0 0 0 1 0 1 1	Shift right circular	Yes	0-3	Shift (W) right. Shift previous LSB into MSB.
SRL	0 0 0 0 1 0 0 1	Shift right logical	Yes	0-3	Shift (W) right. Fill vacated bit positions with 0's.

3.5.9 Immediate Register Instructions

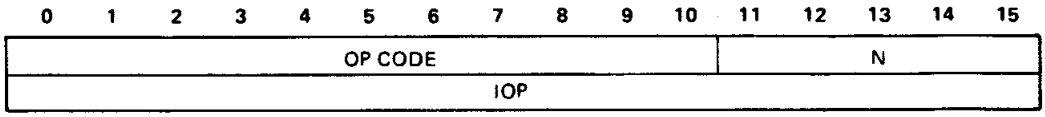
General format:



MNEMONIC	OP CODE 0 1 2 3 4 5 6 7 8 9 10	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
AI	0 0 0 0 0 0 1 0 0 0 1	Add immediate	Yes	0-4	(W) + IOP → (W)
ANDI	0 0 0 0 0 0 1 0 0 1 0	AND immediate	Yes	0-2	(W) AND IOP → (W)
CI	0 0 0 0 0 0 1 0 1 0 0	Compare immediate	Yes	0-2	Compare (W) to IOP and set appropriate status bits
LI	0 0 0 0 0 0 1 0 0 0 0	Load immediate	Yes	0-2	IOP → (W)
ORI	0 0 0 0 0 0 1 0 0 1 1	OR immediate	Yes	0-2	(W) OR IOP → (W)

3.5.10 Internal Register Instruction

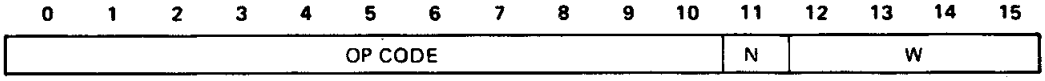
General format:



MNEMONIC	OP CODE 0 1 2 3 4 5 6 7 8 9 10	MEANING	DESCRIPTION
LWPI	0 0 0 0 0 0 1 0 1 1 1	Load workspace pointer immediate	IOP → (WP), no ST bits affected
LIMI	0 0 0 0 0 0 1 1 0 0 0	Load interrupt mask	IOP, bits 12 thru 15 → ST12 thru ST15

3.5.11 Internal Register Store Instructions

General format:

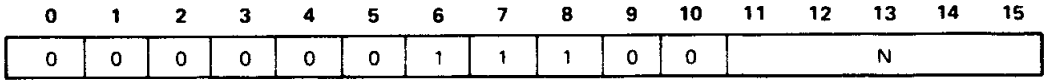


No ST bits are affected.

MNEMONIC	OP CODE 0 1 2 3 4 5 6 7 8 9 10	MEANING	DESCRIPTION
STST	0 0 0 0 0 0 1 0 1 1 0	Store status register	(ST) → (W)
STWP	0 0 0 0 0 0 1 0 1 0 1	Store workspace pointer	(WP) → (W)

3.5.12 Return Workspace Pointer (RTWP) Instruction

General format:

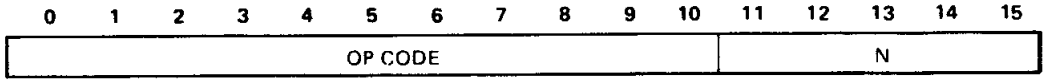


The RTWP instruction causes the following transfers to occur:

(WR15) → (ST)
(WR14) → (PC)
(WR13) → (WP)

3.5.13 External Instructions

General format:



External instructions cause the three address lines (A13; A0, A1) to be set to the below-described levels and the CRUCLK line to be pulsed, allowing external control functions to be initiated.

MNEMONIC	OP CODE 0 1 2 3 4 5 6 7 8 9 10	MEANING	STATUS BITS AFFECTED	DESCRIPTION	ADDRESS BUS A13	A0	A1
IDLE	0 0 0 0 0 0 1 1 0 1 0	Idle	—	Suspend TMS 9980/9981 instruction execution until an interrupt, $\overline{\text{LOAD}}$, or $\overline{\text{RESET}}$ occurs	L	H	L
RSET	0 0 0 0 0 0 1 1 0 1 1	Reset	12–15	0 → ST12 thru ST15	L	H	H
CKOF	0 0 0 0 0 0 1 1 1 1 0	User defined		---	H	H	L
CKON	0 0 0 0 0 0 1 1 1 0 1	User defined		---	H	L	H
LREX	0 0 0 0 0 0 1 1 1 1 1	User defined		---	H	H	H

3.6 TMS 9980A/TMS 9981 INSTRUCTION EXECUTION TIMES

Instruction execution times for the TMS 9980A/TMS 9981 are a function of:

1) Clock cycle time, $t_{c(\phi)}$

2) Addressing mode used where operands have multiple addressing mode capability

3) Number of wait states required per memory access.

Table 4 lists the number of clock cycles and memory accesses required to execute each TMS 9980A/TMS 9981 instruction. For instructions with multiple addressing modes for either or both operands, Table 4 lists the number of clock cycles and memory accesses with all operands addressed in the workspace-register mode. To determine the additional number of clock cycles and memory accesses required for modified addressing, add the appropriate values from the referenced tables. The total instruction-execution time for an instruction is:

$$T = t_{c(\phi)} (C+W \cdot M)$$

where:

T = total instruction time;

$t_{c(\phi)}$ = clock cycle time;

C = number of clock cycles for instruction execution plus address modification;

W = number of required wait states per memory access for instruction execution plus address modification;

M = number of memory accesses.

As an example, the instruction MOVB is used in a system with $t_{c(\phi)} = 0.400\mu s$ and no wait states are required to access memory. Both operands are addressed in the workspace register mode:

$$T = t_{c(\phi)} (C+W \cdot M) = 0.400\ (22+0 \cdot 8) = 8.8\ \mu s.$$

If two wait states per memory access were required, the execution time is:

$$T = 0.400\ (22 + 2 \cdot 8)\ \mu s = 15.2\ \mu s.$$

If the source operand was addressed in the symbolic mode and two wait states were required:

$$T = t_{c(\phi)} (C+W \cdot M)$$

$$C = 22 + 10 = 32$$

$$M = 8 + 2 = 10$$

$$T = 0.400\ (32 + 2 \cdot 10) = 20.8\ \mu s.$$

TABLE 4
INSTRUCTION EXECUTION TIMES

INSTRUCTION	CLOCK CYCLES C	MEMORY ACCESS M	ADDRESS MODIFICATION***	
			SOURCE	DESTINATION
A	22	8	A	A
AB	22	8	B	B
ABS (MSB = 0)	16	4	A	–
(MSB = 1)	20	6	A	–
AI	22	8	–	–
ANDI	22	8	–	–
B	12	4	A	–
BL	18	6	A	–
BLWP	38	12	A	–
C	20	6	A	A
CB	20	6	B	B
CI	20	6	–	–
CKOF	14	2	–	–
CKON	14	2	–	–
CLR	16	6	A	–
COC	20	6	A	–
CZC	20	6	A	–
DEC	16	6	A	–
DECT	16	6	A	–
DIV (ST4 is set)	22	6	A	–
DIV (ST4 is reset)*	104-136	12	A	–
IDLE	14	2	–	–
INC	16	6	A	–
INCT	16	6	A	–
INV	16	6	A	–
Jump (PC is changed)	12	2	–	–
(PC is not changed)	10	2	–	–
LDCR (C = 0)	58	6	A	–
(1<C<8)	26+2C	6	B	–
(9<C<15)	26+2C	6	A	–
LI	18	6	–	–
LIMI	22	6	–	–
LREX	14	2	–	–
LWPI	14	4	–	–
MOV	22	8	A	A
MOVB	22	8	B	B
MPY	62	10	A	–
NEG	18	6	A	–
ORI	22	8	–	–
RSET	14	2	–	–
RTWP	22	8	–	–
S	22	8	A	A
SB	22	8	B	B
SBO	16	4	–	–
SBZ	16	4	–	–
SETO	16	6	A	–
Shift (C ≠ 0)	18+2C	6	–	–
(C ≠0, Bits 12–15 of WRO = 0)	60	8	–	–
(C = 0, Bits 12–15 of WRP = N ≠ 0)	28+2N	8	–	–
SOC	22	8	A	A
SOCB	22	8	B	B
STCR (C = 0)	68	8	A	–
(1≤C≤7)	50	8	B	–
(C = 8)	52	8	B	–
(9≤C≤15)	66	8	A	–

*Execution time is dependent upon the partial quotient after each clock cycle during execution.

***The letters A and B refer to the respective tables that follow.

–CONTINUED–

TABLE 4 (CONTINUED)

INSTRUCTION	CLOCK CYCLES C	MEMORY ACCESS M	ADDRESS MODIFICATION***	
			SOURCE	DESTINATION
STST	12	4	–	–
STWP	12	4	–	–
SWPB	16	6	A	–
SZC	22	8	A	A
SZCB	22	8	B	B
TB	16	4	–	–
X**	12	4	A	–
XOP	52	16	A	–
XOR	22	8	A	–
RESET function	36	10	–	–
LOAD function	32	10	–	–
Interrupt context switch	32	10	–	–
Undefined op codes: 0000-01FF, 0320 033F, 0C00-0FFF, 0780-07FF	8	2	–	–

**Execution time is added to the execution time of the instruction located at the source address.
***The letters A and B refer to the respective tables that follow.

ADDRESS MODIFICATION – TABLE A

ADDRESSING MODE	CLOCK CYCLES C	MEMORY ACCESSES M
WR (T_S or T_D = 00)	0	0
WR indirect (T_S or T_D = 01)	6	2
WR indirect auto-increment (T_S or T_D = 11)	12	4
Symbolic (T_S or T_D = 10, S or D = 0)	10	2
Indexed (T_S or T_D = 10, S or D ≠ 0)	12	4

ADDRESS MODIFICATION – TABLE B

ADDRESSING MODE	CLOCK CYCLES C	MEMORY ACCESSES M
WR (T_S or T_D = 00)	0	0
WR indirect (T_S or T_D = 01)	6	2
WR indirect auto-increment (T_S or T_D = 11)	10	4
Symbolic (T_S or T_D = 10, S or D = 0)	10	2
Indexed (T_S or T_D = 10, S or D ≠ 0)	12	4

4. TMS 9980A/TMS 9981 ELECTRICAL AND MECHANICAL SPECIFICATIONS

4.1 ABSOLUTE MAXIMUM RATINGS OVER OPERATING FREE-AIR TEMPERATURE RANGE (UNLESS OTHERWISE NOTED)*

Supply voltage, V_{CC} (see Note 1) . . . . . −0.3 to 15 V
Supply voltage, V_{DD} (see Note 1) . . . . . −0.3 to 15 V
Supply voltage, V_{BB} (see Note 1) (9980A only) . . . . . −5.25 to 0 V
All input voltages (see Note 1) . . . . . −0.3 to 15 V
Output voltage (see Note 1) . . . . . −2 V to 7 V
Continuous power dissipation . . . . . 1.4 W
Operating free-air temperature range . . . . . 0°C to 70°C
Storage temperature range . . . . . −55°C to 150°C

*Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the "Recommended Operating Conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTE 1: Under absolute maximum ratings voltage values are with respect to V_{SS}.

4.2 RECOMMENDED OPERATING CONDITIONS

	MIN	NOM	MAX	UNIT
Supply voltage, V_{BB} (9980A only)	−5.25	−5	−4.75	V
Supply voltage, V_{CC}	4.75	5	5.25	V
Supply voltage, V_{DD}	11.4	12	12.6	V
Supply voltage, V_{SS}		0		V
High-level input voltage, V_{IH}	2.2	2.4	V_{CC}+1	V
Low-level input voltage, V_{IL}	−1	0.4	0.8	V
Operating free-air temperature, T_A	0	20	70	°C

4.3 ELECTRICAL CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS (UNLESS OTHERWISE NOTED)

PARAMETER			TEST CONDITIONS	MIN	TYP*	MAX	UNIT
I_I	Input current	Data bus during DBIN	$V_I = V_{SS}$ to V_{CC}			±75	μA
		$\overline{WE}$, $\overline{MEMEN}$, $\overline{DBIN}$ during HOLDA	$V_I = V_{SS}$ to V_{CC}			±75	
		Any other inputs	$V_I = V_{SS}$ to V_{CC}			±10	
V_{OH}	High-level output voltage		I_O = −0.4 mA	2.4			V
V_{OL}	Low-level output voltage		I_O = 2 mA			0.5	V
			I_O = 3.2 mA			0.65	
I_{BB}	Supply current from V_{BB} (9980A Only)					1	mA
I_{CC}	Supply current from V_{CC}		0°C		50	60	mA
			70°C		40	50	
I_{DD}	Supply current from V_{DD}		0°C		70	80	mA
			70°C		65	75	
C_I	Input capacitance (any inputs except data bus)		f = 1 MHz, unmeasured pins at V_{SS}		15		pF
C_{DB}	Data bus capacitance		f = 1 MHz, unmeasured pins at V_{SS}		25		pF
C_O	Output capacitance (any output except data bus)		f = 1 MHz, unmeasured pins at V_{SS}		15		pF

*All typical values are at T_A = 25°C and nominal voltages.

4.4 CLOCK CHARACTERISTICS

The TMS 9980A and TMS 9981 have an internal 4-phase clock generator/driver. This is driven by an external TTL compatible signal to control the phase generation. In addition, the TMS 9981 provides an output (OSCOUT) that in conjunction with CKIN forms an on-chip crystal oscillator. This oscillator requires an external crystal and two capacitors as shown in Figure 13. The external signal or crystal must be 4 times the desired system frequency.

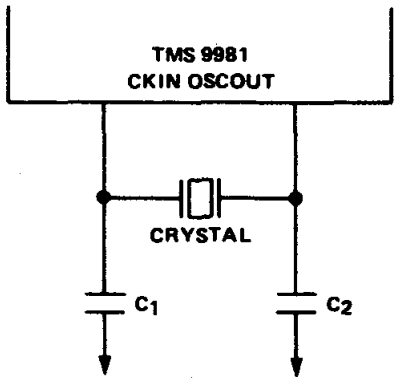


FIGURE 13 – CRYSTAL OSCILATOR CIRCUIT

4.4.1 Internal Crystal Oscillator (9981 Only)

The internal crystal oscillator is used as shown in Figure 13. The crystal should be a fundamental series resonant type. C_1 and C_2 represent the total capacitance on these pins including strays and parasitics.

PARAMETER	TEST CONDITIONS	MIN	TYP	MAX	UNIT
Crystal frequency	0°C-70°C	6		10	MHZ
C_1, C_2	0°C-70°C	10	15	25	pf

4.4.2 External Clock

The external clock on the TMS 9980A and optional on the TMS 9981, uses the CKIN pin. In this mode the OSCOUT pin of the TMS 9981 must be left floating. The external clock source must conform to the following specifications.

	PARAMETER	MIN	TYP	MAX	UNIT
f_{ext}	External source frequency*	6		10	MHz
V_H	External source high level	2.2			V
V_L	External source low level			0.8	V
T_r/T_f	External source rise/fall time		10		ns
T_{WH}	External source high level pulse width	40			ns
T_{WL}	External source low level pulse width	40			ns

*This allows a system speed of 1.5 MHz to 2 MHz.

4.5 SWITCHING CHARACTERISTICS OVER FULL RANGE OF RECOMMENDED OPERATING CONDITIONS

The timing of all the inputs and outputs are controlled by the internal 4 phase clock; thus all timings are based on the width of one phase of the internal clock. This is $1/f_{(CKIN)}$ (whether driven or from a crystal). This is also $\frac{1}{4}f_{system}$. In the following table this phase time is denoted t_w.

All external signals are with reference to $\phi 3$ (see Figure 14).

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_r(\phi 3)$	Rise time of $\phi 3$	t_w=1/f(CKIN) =¼fsystem C_L = 200pf	3	5	10	ns
$t_f(\phi 3)$	Fall time of $\phi 3$		5	7.5	15	ns
$t_w(\phi 3)$	Pulse width of $\phi 3$		t_w-15	t_w-10	t_w+10	ns
t_{su}	Data or control setup time*		t_w-30			ns
t_h	Data hold time*		$2t_w+10$			ns
t_{PHL} ($\overline{WE}$)	Propagation delay time WE high to low		t_w-10	t_w	t_w+20	ns
t_{PLH} ($\overline{WE}$)	Propagation delay time WE low to high		t_w	t_w+10	t_w+30	ns
t_{PHL}(CRUCLK)	Propagation delay time, CRUCLK high to low		−20	−10	+10	ns
t_{PLH}(CRUCLK)	Propagation delay time, CRUCLK low to high		$2t_w-10$	$2t_w$	$2t_w+20$	ns
t_{OV}	Delay time from output valid to $\phi 3$ low		t_w-50	t_w-30		ns
t_{OX}	Delay time from output invalid to $\phi 3$ low			t_w-20	t_w	ns

*All inputs except IC0-IC2 must be synchronized to meet these requirements. IC0-IC2 may change asynchronously. See section 2.10.4.

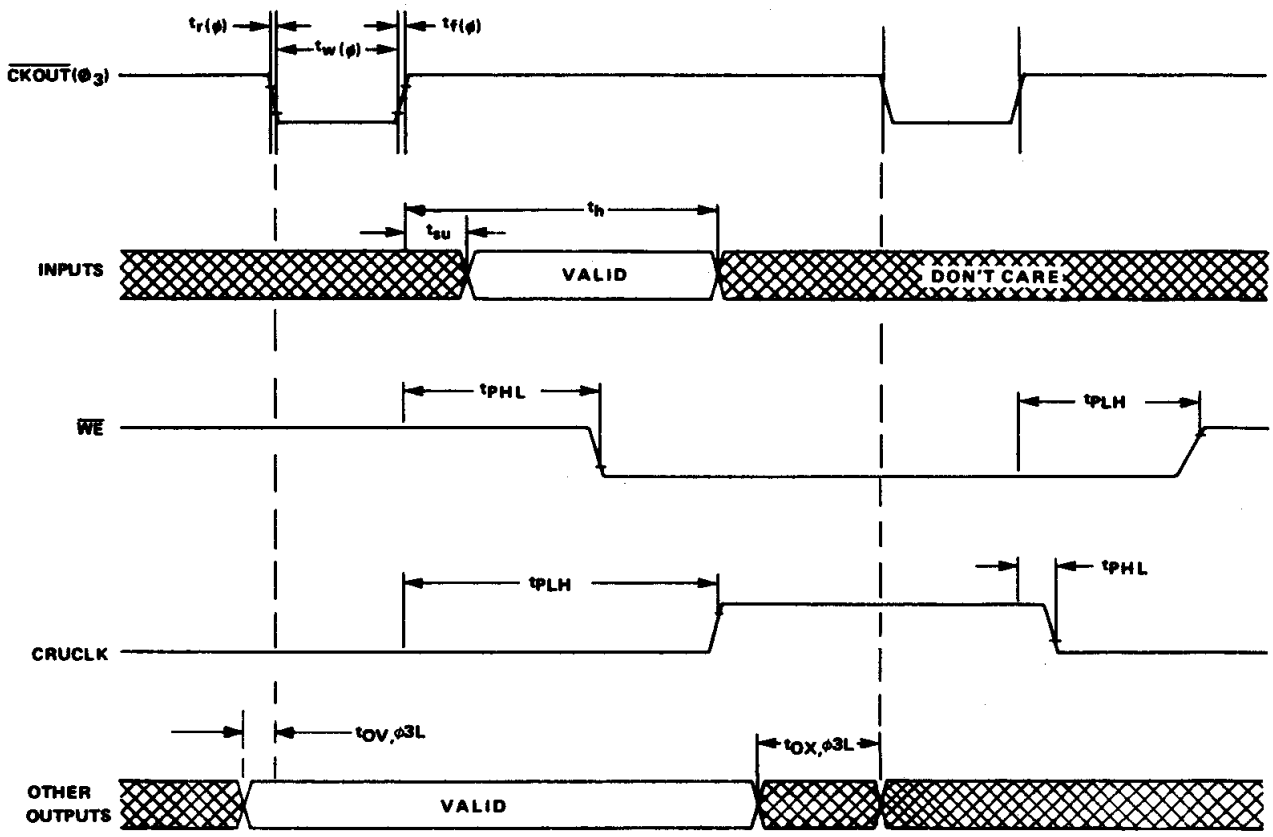


FIGURE 14 – EXTERNAL SIGNAL TIMING DIAGRAM

5. THE PROTOTYPING SYSTEM

5.1 HARDWARE

Because of the software compatibility, the users of TMS 9980A/TMS 9981 uses the TMS 9900 prototyping system to generate and debug software and to debug I/O controller interfaces. The prototyping system consists of:

- 990/4 computer with TMS 9900 microprocessor
- 1024 bytes of ROM containing the bootstrap loader for loading prototyping system software, the front-panel and maintenance utility, and the CPU self-testing feature
- 16,896 bytes of RAM with provisions for expansion up to 57,334 bytes of RAM
- Programmable-write-protect feature for RAM
- Interface for Texas Instruments Model 733 ASR* Electronic Data Terminal with provisions for up to five additional interface modules
- Available with Texas Instruments 733 ASR Electronic Data Terminal
- 7-inch-high table-top chassis
- Programmer's front panel with controls for run, halt, single-instruction execute, and entering and displaying memory or register contents
- Power supply with the following voltages:
  5 V dc @ 20 A
  12 V dc @ 2 A
  −12 V dc @ 1 A
  −5 V dc @ 0.1 A
- Complete hardware and software documentation.

5.2 SYSTEM CONSOLE

The system console for the prototyping system is the 733 ASR, which provides keyboard entry, 30-character-per-second thermal printer, and dual cassette drives for program loading and storage.

5.3 SOFTWARE

The following software is provided on cassette for loading into the prototyping system:

- Debug Monitor – Provides full control of the prototyping system during program development and includes single instruction, multiple breakpoints, and entry and display capability for register and memory contents for debugging user software user 733 ASR console control.
- One-Pass Assembler – Converts source code stored on cassette to relocatable object on cassette and generates program listing. (Object is upward compatible with other 990 series assemblers.)
- Linking Loader – Allows loading of absolute and relocatable object modules and links object modules as they are loaded.

* Requires remote device control and 1200 baud EIA interface option on 733 ASR.

- Source Editor – Enables user modifications of both source and object from cassette with resultant storage on cassette.

- Trace Routine – Allows user to monitor status of computer at completion of each instruction.

- PROM Programming/Documentation Facility – Provides documentation for ROM mask generation, or communicates directly with the optional PROM Programmer Unit.

5.4 OPTIONS

The following optional equipment is offered for the prototyping system:

- Battery-pack/standby-power supply
- PROM programming unit and adapter boards
- Universal wire-wrap modules
- Expansion RAM modules
- Expansion EPROM modules
- I/O modules and other interfaces
- Rack-mounted version
- International ac voltage option

6. SUPPORT CIRCUITS

DEVICE	ORGANIZATION/ FUNCTION	I/O STRUCTURE	PACKAGE	ACCESS TIME
		RAMS		
TMS 4036-2	64 x 8 Static	Common bus	20 Pin	450 ns MAX
TMS 4033	1024 x 1 Static	Dedicated bus	16 Pin	450 ns MAX
TMS 4039-2	256 x 4 Static	Dedicated bus	22 Pin	450 ns MAX
TMS 4042-2	256 x 4 Static	Common bus	18 Pin	450 ns MAX
TMS 4043-2	256 x 4 Static	Common bus	16 Pin	450 ns MAX
TMS 4050	4096 x 1 Dynamic	Common bus	18 Pin	300 ns MAX
TMS 4051	4096 x 1 Dynamic	Dedicated bus	18 Pin	300 ns MAX
TMS 4060	4096 x 1 Dynamic	Dedicated bus	22 Pin	300 ns MAX
		ROMS/PROMS		
TMS 2708	1024 x 8 EROM		24 Pin	450 ns MAX
TMS 4700	1024 x 8 ROM		24 Pin	450 ns MAX
SN74S371	256 x 8 ROM		20 Pin	70 ns MAX
SN74S471	256 x 8 ROM		20 Pin	70 ns MAX
SN74S472	512 x 8 PROM		20 Pin	55 ns TYP
		PERIPHERALS		
TMS 9901	PSI, Programmable System Interface		40 Pin	
TMS 9902	ACC, Asynchronous Communication Controller		18 Pin	
TMS 9903	SCC, Synchronous Communication Controller		20 Pin	
TIM 9905	Data multiplexer (SN74LS251)		16 Pin	
TIM 9906	Addressable latch (SN74LS259		16 Pin	
TIM 9907	Priority encoder (SN74148)		16 Pin	
SN74S412	8-bit I/O port		24 Pin	
TMS 6011	UART		40 Pin	
SN74S241	Bidirectional bus driver		20 Pin	

7. SYSTEM DESIGN EXAMPLES

Figure 15 illustrates a typical minimum TMS 9981 system. Eight bits of input and output interface are implemented. No interface circuits are used for interrupt code thus providing for reset and one interrupt only. CKIN and OSCOUT are tied to a 10 MHz crystal to use the on-chip crystal oscillator. The memory system contains 1024 x 8 ROM and 256 x 8 RAM. The package count for this system is 6 packages.

A maximum TMS 9980A/TMS 9981 system is illustrated in Figure 16. ROM and RAM are both shown for a total of 16,284 bytes of memory. The I/O interface support 2048 output bits and 2048 input bits. RESET, LOAD, and 4 interrupts are implemented on the interrupt interface lines. Optional external clock may be supplied on CKIN. Buss buffers, required for this maximally configured system, are indicated on the system buses.

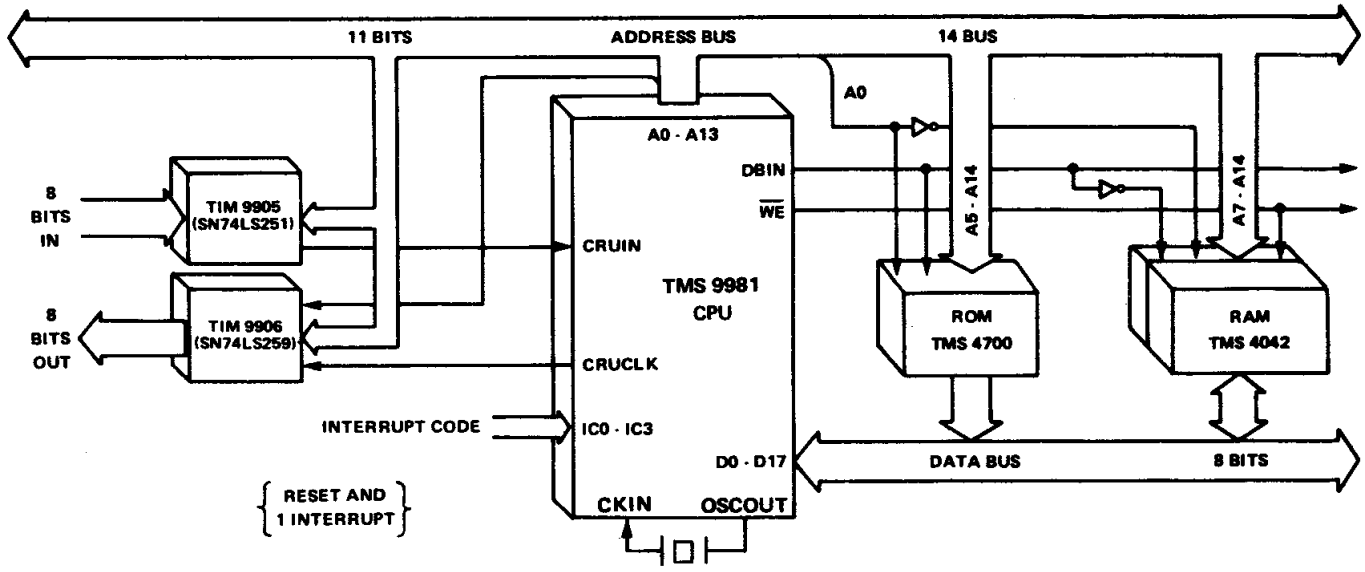


FIGURE 15 – MINIMUM TMS 9980 SYSTEM

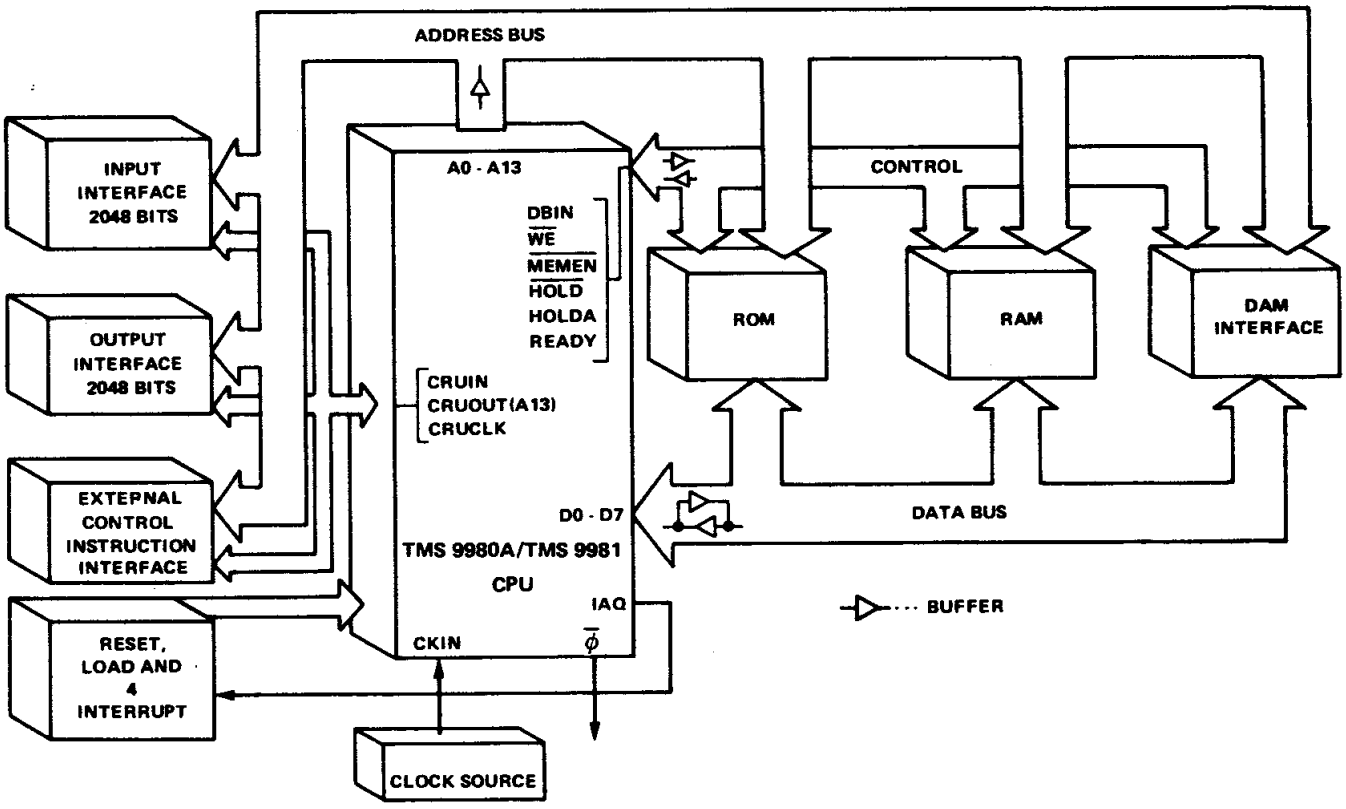
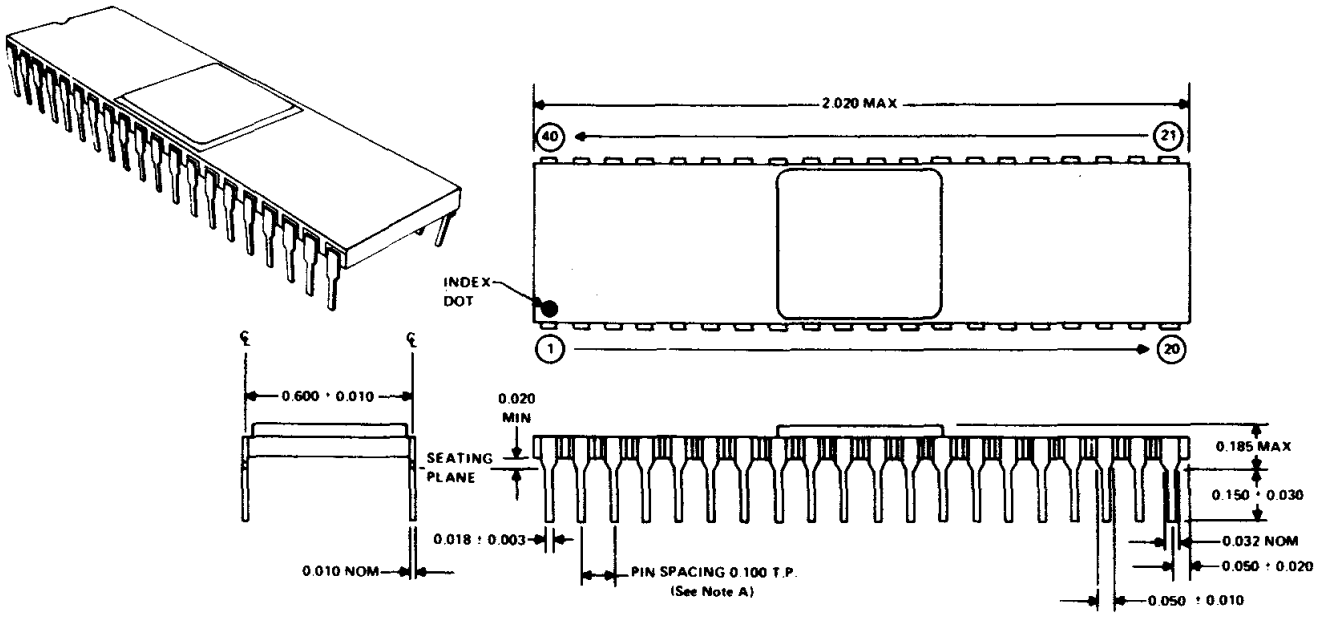


FIGURE 16 – MAXIMUM TMS 9980 SYSTEM

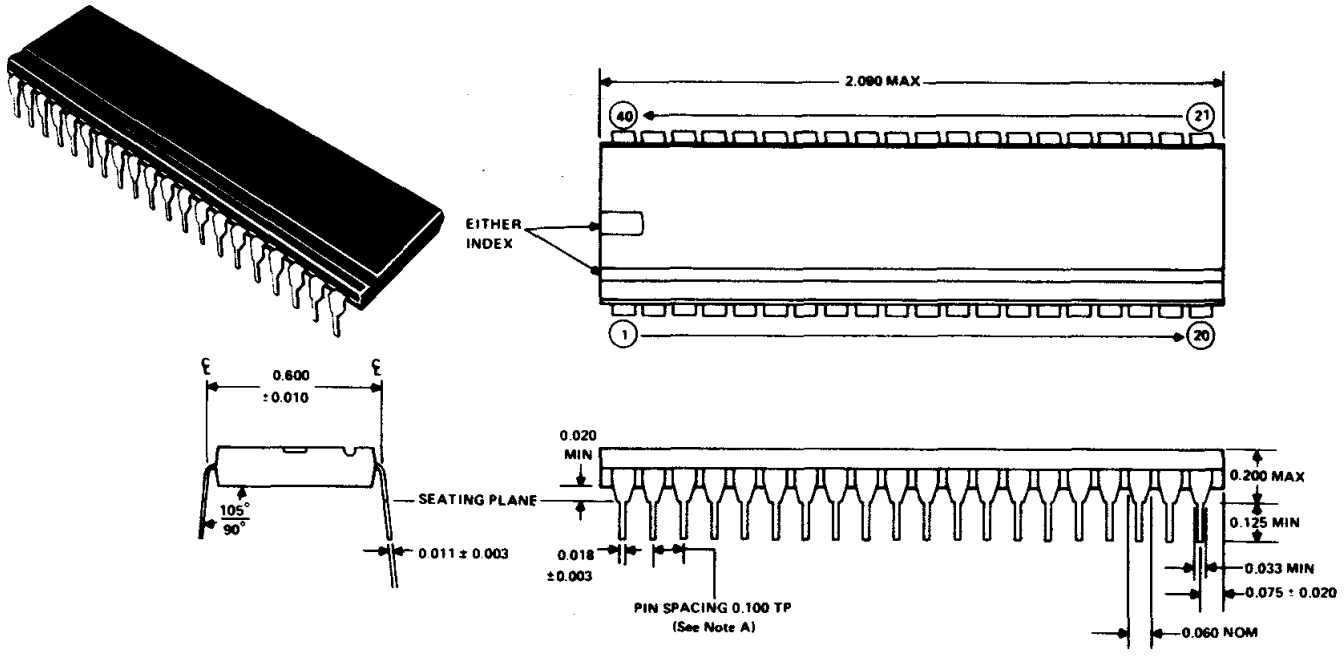
8. MECHANICAL DATA

8.1 TMS 9980A/TMS 9981 – 40-PIN CERAMIC PACKAGE



NOTES: A. Each pin centerline is located within 0.010 of its true longitudinal position.
B. All linear dimensions are in inches.

8.2 TMS 9980A/TMS 9981 – 40-PIN PLASTIC PACKAGE



NOTES: A. Each pin centerline is located within 0.010 of its true longitudinal position.
B. All linear dimensions are in inches.

APPENDIX H

TMS 9900 FAMILY MACHINE CYCLES

H.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.

H.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.

H.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

H.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.

H.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.

H.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.

H.4 Data Derivation Sequences

H.4.1 Workspace Register

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

H.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

H.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

H.4.4 Workspace Register Indirect Auto-Increment (Work 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

H.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

H.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

H.5 Instruction Execution Sequences

H.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.

H.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

H.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 fetc 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

H.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

H.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.

H.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

H.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.

H.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.

H.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.

H.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

H.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

H.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.

H.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

H.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

H.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

H.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 registe DB = Result
10+C	ALU	Increment PC AB = NC DB = SD

H.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

H.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

H.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

H.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

H.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

H.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)

H.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

H.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

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

H.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

H.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

H.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

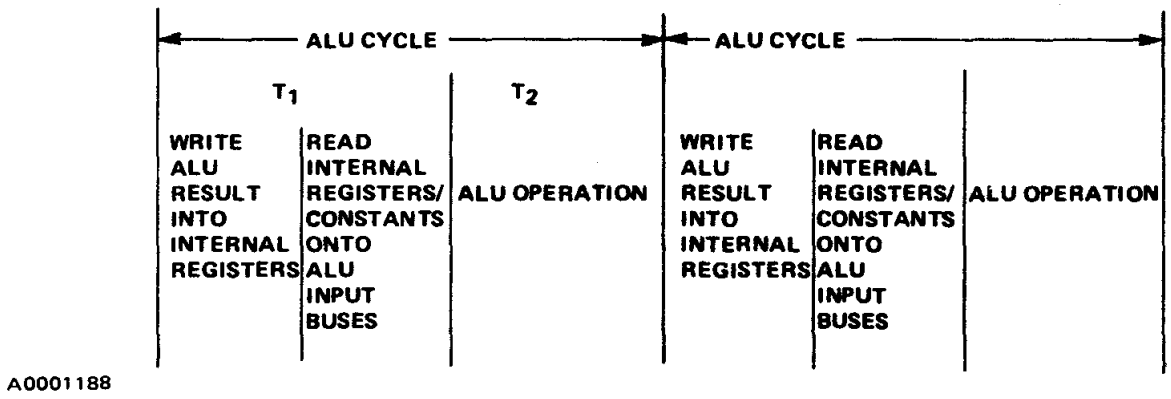


Figure H-1. ALU Cycle

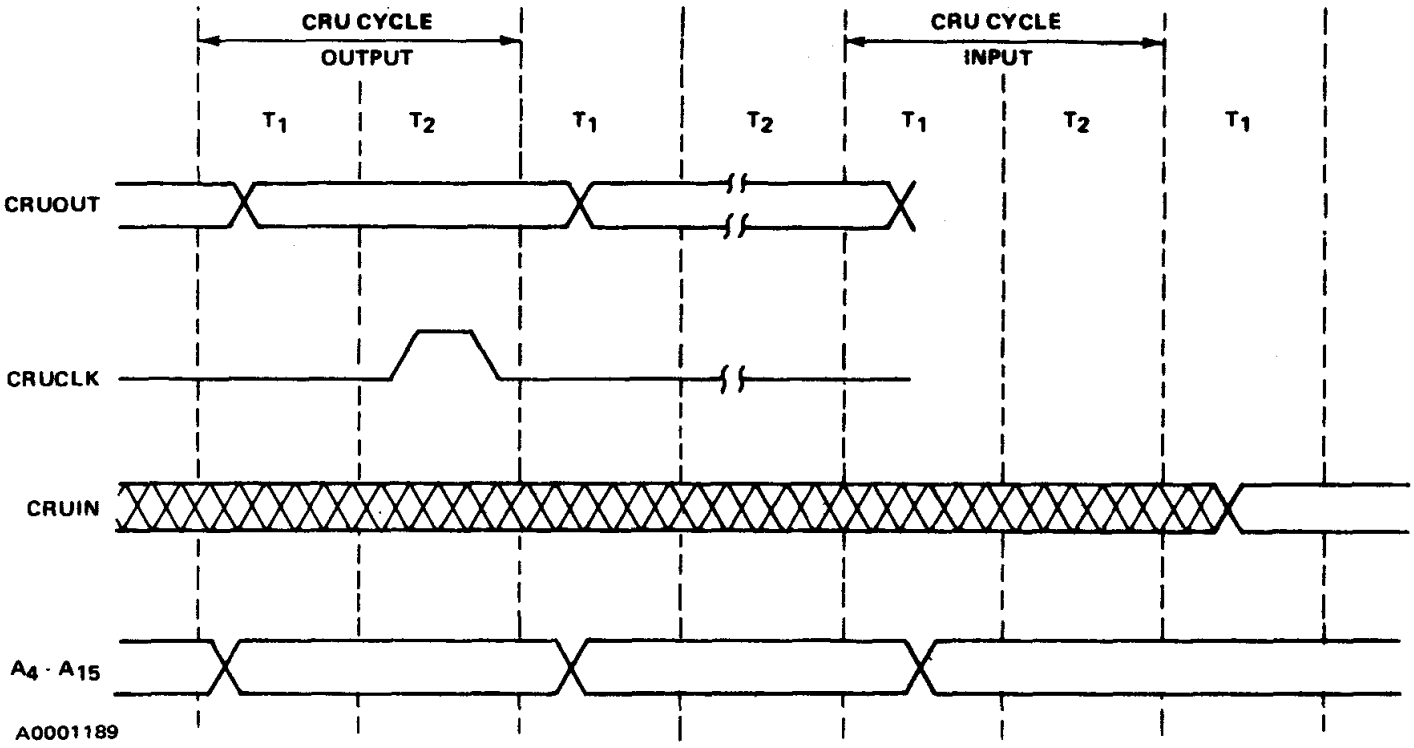


Figure H-2. CRU Cycle

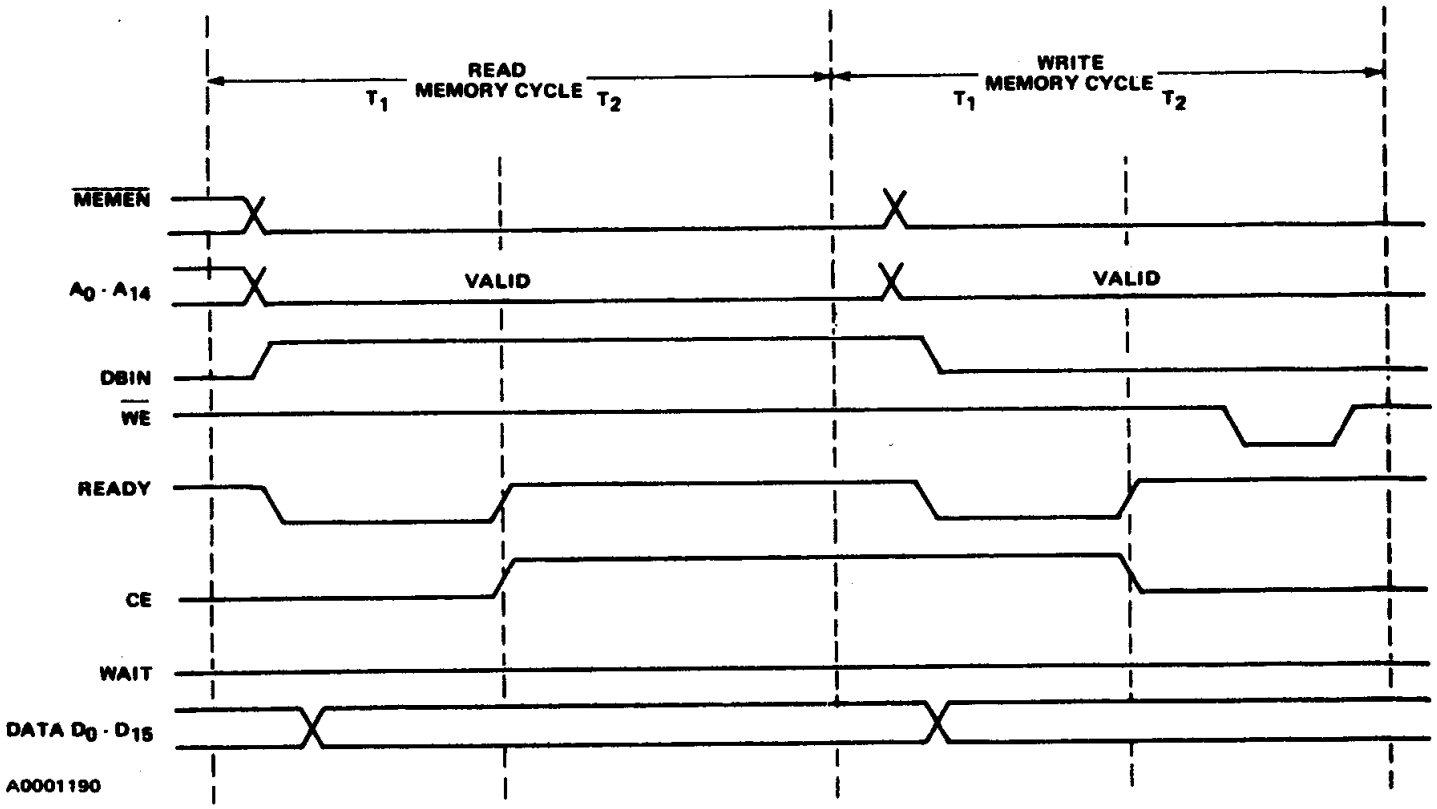


Figure H-3. TMS 9900 Memory Cycle (No Wait States)

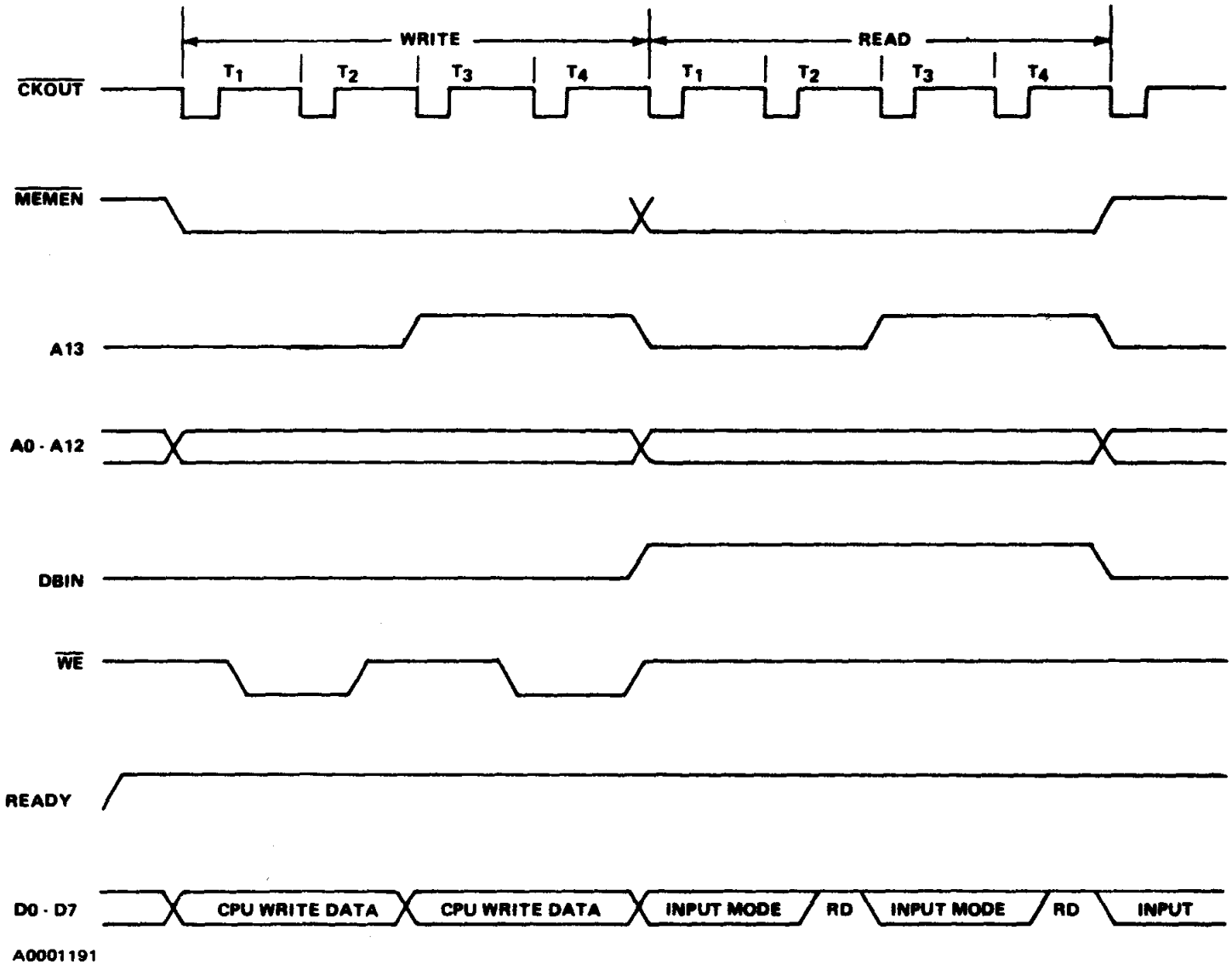


Figure H-4. TMS 9980A/81 Memory Cycle (No Wait States)

APPENDIX I

DATA TERMINAL HOOKUP

.1 TELETYPEWRITER TERMINAL STRIP CONNECTIONS

.2 EIA RS-232-C CABLING FOR 733 DATA TERMINAL

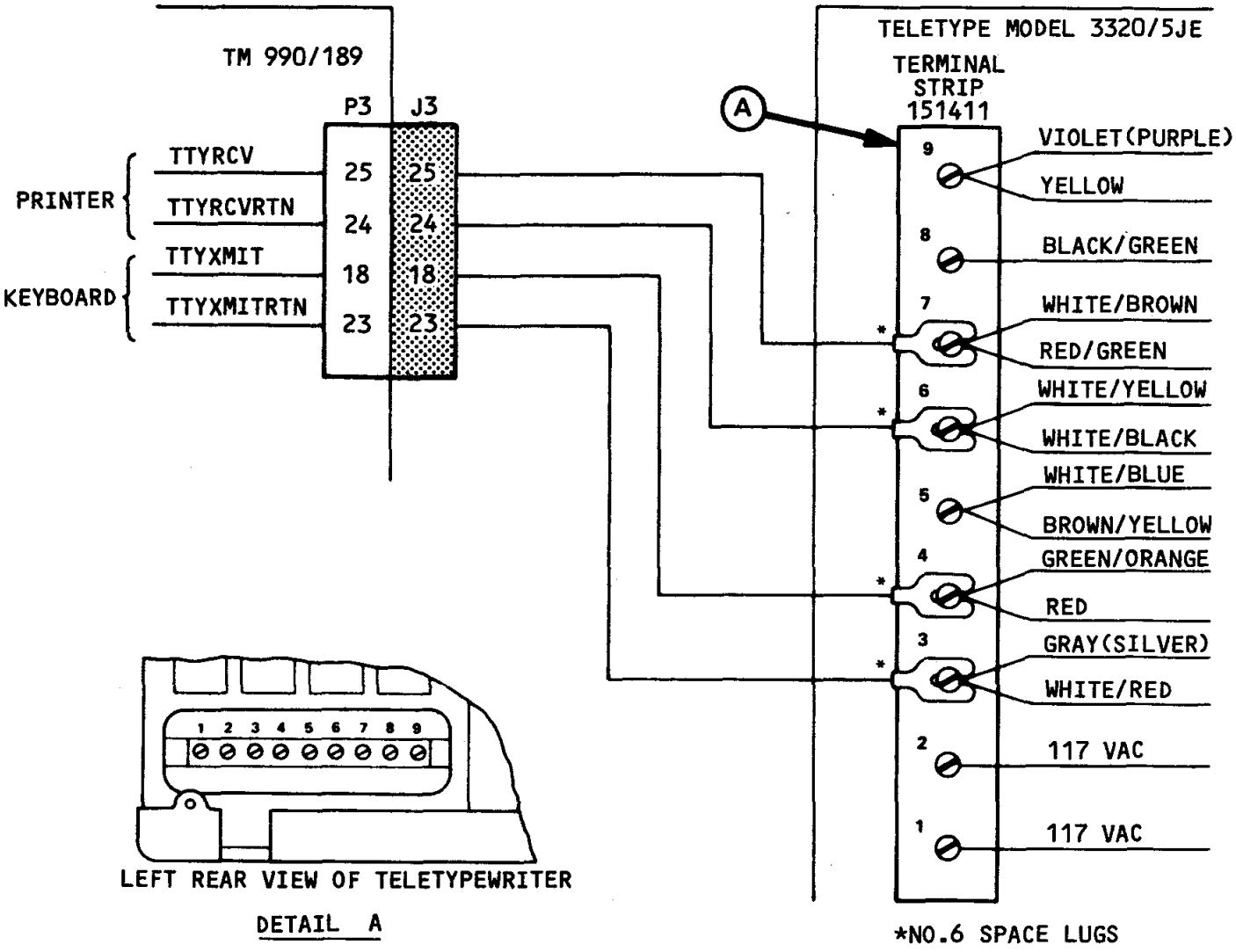


Figure I-1. Teletypewriter Terminal Strip Connections

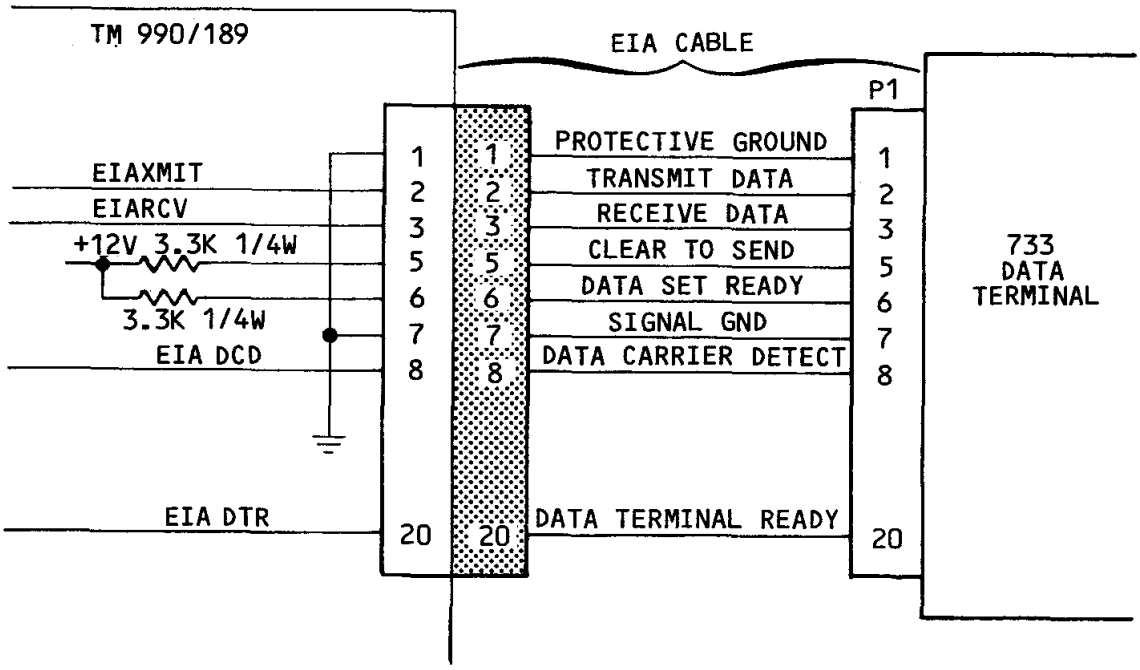


Figure I-2. EIA RS-232-C Cabling for 733 Data Terminal

APPENDIX J

I/O CABLES AND CONNECTORS

Connector	Description	Part Number
J1	4 contact plug	AMP 1-480702-0
	Contacts (take 18-24 AWG wire)	AMP 350689-1
J2	6 contact plug	AMP 1-480704-0
	Contacts (take 18-24 AWG wire)	AMP 350689-1
J3	25 contact cable connector	ITT Cannon or TRW Cinch DB-25P
J4 & J5	40-pin connectors for flat ribbon cable	1. Berg 65484-017 (Has strain relief) 2. Berg 65485-017 (No strain relief) 3. 3M 3417-5000 (Has strain relief)
	Individual conductors (Use #22 to #26 wires and Berg 47743, 47747, or 75691-013 pins)	Berg 65043-017
J4 & J5	40 conductor cable assemblies with connectors.	1. AP Products 924015-36-R 36", connector at one end 2. AP Products 924005-06-R 6", connectors at both ends 3. AP Products 924005-18-R 18", connectors at both ends 4. AP Products 924005-36-R 36", connectors at both ends

Amp Incorporated
Harrisburg, PA 17105

AP Products INC
Box 110-Q
Plainesville, Ohio 44077

3M Company
Electronic Products Division
Saint Paul, MN 55101

E. I. du Pont de Nemours & Co. INC.
Berg Electronics Division
New Cumberland, PA 17070

APPENDIX K

PARTS LIST FOR TM 990/189 MICROCOMPUTER

TM 990/189 PARTS LIST (Sheet 1 of 2)

Symbol	Description	Qty
C01, C02	Capacitor, fixed, tantilum, 150 uFd, 10%, 15 V	2
C03, C04	Capacitor, fixed, tantilum, 100 uFd, 10%, 20 V	2
C05	Capacitor, fixed, tantilum, 39 uFd, 10%, 10 V	1
C06, C07, C10 C12 to C37	Capacitor, fixed, axial lead, 0.047 uFd, +80%, -20%	29
C08, C09	Capacitor, fixed, ceramic, 0.100 uFd, 10%, 50 V	2
C11	Capacitor, fixed, tantilum, 56 uFd, 10%, 6 V	
CR01 to CR07	Optoelectronic device, TIL 220	7
CR08	Diode, 1N4001, rectifier, 1 A, 50 PIV	1
CR10, CR11	Diode, 1N914B, switching, 75 PIV, 75 mA	2
CR13	Diode, 1N758A, 10.0 V, 5%, voltage regulator	1
P1	Connector, universal PWB mounting, 4 circuit (AMP 350430-1)	1
P2	Connector, universal PWB mounting, 6 circuit (AMP 350431-1)	1
P4, P5	Header, 20 pin. straight double row	2
Q01 to Q09	Transistor, TI A5T2907, PNP, silicon	9
Q10, Q11	Transistor, TI A5T2222, NPN, silicon	2
Q13	Transistor, TI 2N2219A, NPN, general purpose, SW TO5	1
R01 to R04, R07, R08, R13	Resistor, fixed, 120 ohm, 5%, 0.25 W	7
R05, R06, R11, R12, R24 R25, R41	Resistor, fixed, 1.0 Kilohm, 5%, 0.25 W	7
R09, R10, R28, R46, R47	Resistor, fixed, 470 ohm, 5%, 0.25 W	5
R14	Resistor, fixed, 100 kilohm, 5%, 0.25 W	1
R15 to R22	Resistor, fixed, 430 ohm, 5%, 0.25 W	8
R23,R27,R43	Resistor, fixed, 10 kilohm, 5%, 0.25 W	3
R26	Resistor, fixed, 2.2 kilohm, 5%, 0.25 W	1
R29, R45	Resistor, fixed, 4.7 kilohm, 5%, 0.25 W	2
R30, R31	Resistor, fixed, 15 kilohm, 5%, 0.25 W	2
R40	Resistor, fixed, 12 kilohm, 5%, 0.25 W	1
R42	Resistor, fixed, 1.5 kilohm, 5%, 0.25 W	1
R44	Resistor, fixed, 220 ohm, 5%, 0.25 W	1
R48	Resistor, fixed, 100 ohm, 5%, 0.25 W	1
R50	Resistor, fixed, 330 kilohm, 5%, 0.25 W	1
R51	Resistor, fixed, 2.2 megohm, 5%, 0.25 W	1
S1	Toggle switch (Cutler-Hammer SF6TGX392)	1

Symbol	Description	Qty
U01, U07	Network, SN7416N	2
U02	IC, SN74LS2987N, quad 2-input multiplexer	1
U03	Network, SN74LS109N	1
U04	Network, SN74LS112N	1
U05, U08	Network, 74LS145	2
U06, U13	Network SN74LS00N	2
U09, U12	Resistor, 10.0 Kilohm, pullup, 16 pins, DIL	2
U10, U11	TMS 9901 Programmable system interface	2
U14, U28	Network, SN74LS04N	2
U15	Network, SN74LS08N	1
U16	Resistor, 1.0 kilohm, pullup, 16 pins, DIL	1
U17	Resistor network, 2.2 Kilohm, 2%	1
U19	Microprocessor, MP 9529	1
U20, U22	IC, 2114 1024 x 4-bit static RAM	2
U24	Network, SN7425N	1
U25	Network, SN74LS27N	1
U29	IC, SN74LS123N, monostable multivibrator	1
U33	ROM with UNIBUG monitor	1
U34	Network, SN74LS139N	1
U35	Network, SN74LS32N	1
U36	Network, SN75i89AN, quad line receiver (MC1489AL)	1
U38	IC, RC4558P operational amplifier	1
VR1	IC, voltage regulator (UA7905C, MC7905CP)	1
Y1	Crystal, 8.000 MHz, quartz (No. Eng. HC18/uNE-18)	1
	Sound disk: (P/N SK160: Gulton Indst., Box 4300, Fullerton, Cal. 92634 or P/N 60993: Vernitron, 232 Forbes Rd, Redford Ohio, 44146)	1