As you are now the owner of this document which should have come to you for free, please consider making a donation of £1 or more for the upkeep of the (Radar) website which holds this document. I give my time for free, but it costs me money to bring this document to you. You can donate here https://blunham.com/Misc/Texas

Many thanks.

Please do not upload this copyright pdf document to any other website. Breach of copyright may result in a criminal conviction.

This Acrobat document was generated by me, Colin Hinson, from a document held by me. I requested permission to publish this from Texas Instruments (twice) but received no reply. It is presented here (for free) and this pdf version of the document is my copyright in much the same way as a photograph would be. If you believe the document to be under other copyright, please contact me.

The document should have been downloaded from my website https://blunham.com/, or any mirror site named on that site. If you downloaded it from elsewhere, please let me know (particularly if you were charged for it). You can contact me via my Genuki email page: https://www.genuki.org.uk/big/eng/YKS/various?recipient=colin

You may not copy the file for onward transmission of the data nor attempt to make monetary gain by the use of these files. If you want someone else to have a copy of the file, point them at the website. (https://blunham.com/Misc/Texas). **Please do not point them at the file itself as it may move or the site may be updated.**

It should be noted that most of the pages are identifiable as having been processed by me.

I put a lot of time into producing these files which is why you are met with this page when you open the file.

If you find missing pages, pages in the wrong order, anything else wrong with the file or simply want to make a comment, please drop me a line (see above).

It is my hope that you find the file of use to you.

Colin Hinson
In the village of Blunham, Bedfordshire.

TEXAS INSTRUMENTS

TM 990

TM 990/102
Extended Memory CPU

MICROPROCESSOR SERIES™

User's Guide

SECTION 1

INTRODUCTION

1.1 GENERAL

The TM 990/102 microcomputer board provides a maximum of 128-K byte dynamic RAM memory along with memory mapping capabilities that allow an address map of up to 1 megabyte. It is designed around TI's 16-bit TMS 9900 microprocessor. Figure 1-1 shows major components and Figure 1-2 shows board dimensions. Board features include:

- TMS 9900 microprocessor instruction set is software compatible with other members of TIs 990 family.
- Fully compatible with the TM 990 microcomputer product line.
- Dynamic RAM (DRAM) is user-selectable; 16 chips provide the following memory space:

  - 64 K bytes (32 K words) using TMS 4532s (1 by 32 K bits)
  - 128 K bytes (64 K words) using TMS 4164s (1 by 64 K bits)

- Eraseable programmable read-only memory (EPROM) is user-selectable; the following memory configurations can be populated:

  - 0 K bytes (The EPROM can be turned off)
  - 4 K bytes using TMS 2516s
  - 8 K bytes using TMS 2532s
  - 16 K bytes using TMS 2564s

  An optional TM 990/404 monitor is available on two EPROMs.

- A total of 1 megabyte of memory can be addressed via the 19-line address bus using the memory mapper under software control. The 8 MSBs of a 19-bit address bus are developed using the mapper's 16 programmable registers.
- 3 MHz crystal-controlled clock.
- Two onboard event and interval timers; can be programmed in polled or interrupt mode.
- Seventeen interrupts; priority programmable at interrupt handler as well as at microprocessor. Fifteen are maskable. Two unmaskable interrupts are RESET (powerup, or hardware actuated) and LOAD (software or hardware actuated).
- Two LEDs, one user-programmable, one indicating memory mapper mode.
- One EIA RS-232-C serial terminal interface.
- Onboard memory is accessible to a secondary processor.

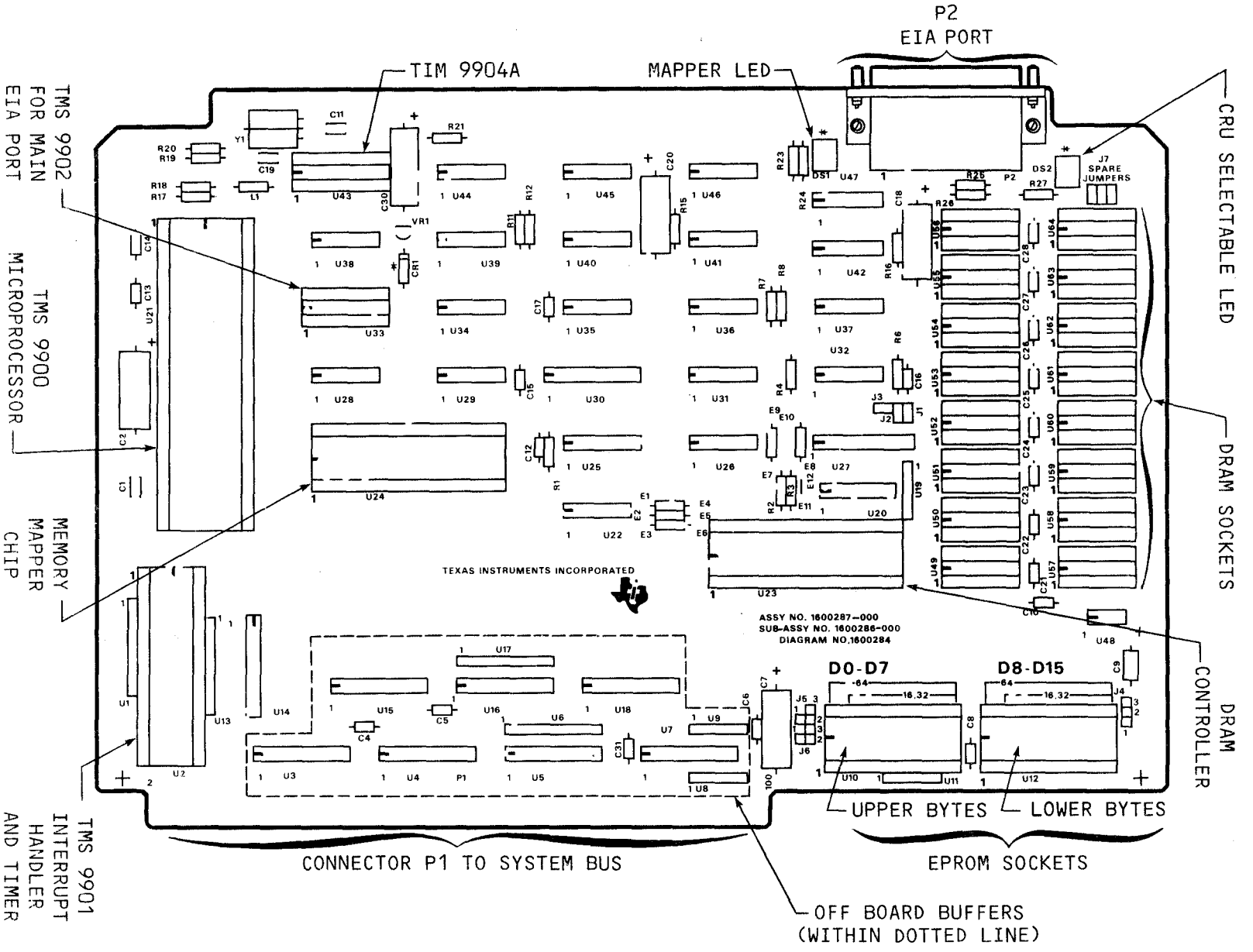


FIGURE 1-1. TM 990/102 MAJOR COMPONENTS

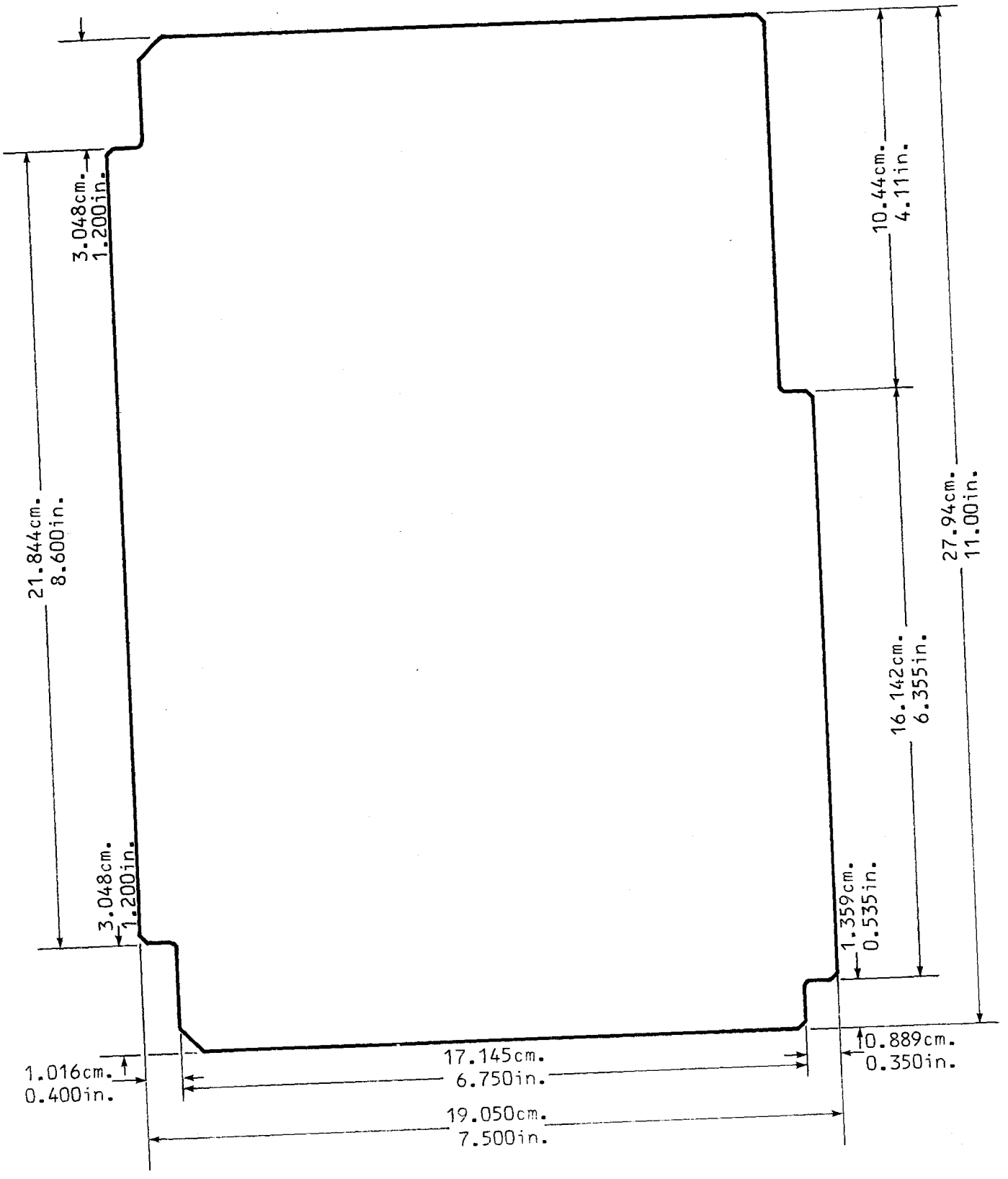


FIGURE 1-2. TM 990/102 DIMENSIONS

1.2 MANUAL ORGANIZATION

This manual is organized into the following sections and appendices:

Section 1, Introduction, contains general information such as board features, specifications.

Section 2, Installation and Operation, helps the user set up and begin board operation. Jumpers and other features are covered.

Section 3, Memory Mapping, shows how to configure and use the memory mapper and its 19 address lines.

Section 4, TMS 9901 Operation, shows how to program the interrupt handler, the 9901s interval timer, and user LED.

Section 5, TMS 9902 Operation, shows how to program the EIA port controller and interval timer.

Section 6, Theory of Operation, blocks off the board by functions and explains their theory.

Section 7, Instruction Set, gives brief descriptions of TMS 9900 assembly language code.

Appendix A. Schematics
Appendix B. EIA RS-232-C Cabling
Appendix C. ASCII Code
Appendix D. Binary, Decimal, and Hexadecimal Numbering
Appendix E. Parts List
Appendix F. P1 and P2 Pin Assignments
Appendix G. 990 Object Code
Appendix H. TMS 74LS612 Memory Mapper Data Sheet

1.3 PRODUCT INDEX

The TM 990/102 microcomputer is available is three different configurations which are specified by a dash number appended to the product name; e.g., TM 990/102-1. These configurations are listed in Table 1-1.

TABLE 1-1. PRODUCT INDEX

Model No.	Memory	
	Socketed	Populated
TM 990/102-1	0 to 16 K bytes EPROM	No RAM
TM 990/102-2	0 to 16 K bytes EPROM	64 K bytes DRAM
TM 990/102-3	0 to 16 K bytes EPROM	128 K bytes DRAM

1.4 BOARD CHARACTERISTICS

Figure 1-1 shows the major portions and components of the TM 990/102 microcomputer. The system bus connector is P1, which is a 100-pin (50 each side)

PC board edge connector spaced on 0.125 inch centers. Connector P2 is the main serial port. The P2 connector is a standard 25-position female jack used in RS232-C communications. Figure 1-2 shows the PC board silkscreen markings which detail the various components on the board as well as the board dimensions and tolerances.

1.5 GENERAL SPECIFICATIONS

Power Consumption:

	+5V		+12V		-12V		
	Typ	Max	Typ	Max	Typ	Max	Units
TM 990/102-1	1.7	2.0	0.3	0.5	0.25	0.40	A
TM 990/102-2	1.9	3.0	0.3	0.5	0.25	0.40	A
TM 990/102-3	1.9	3.0	0.3	0.5	0.25	0.40	A

Clock Rate: 3 MHz

Baud Rates: 110, 300, 600, 1200, 2400, 4800, 9600, 19200

Total Memory Capacity:

DRAM: Sixteen TMS 4532s (32 K X 1 bit each) for TM 990/102-2
Sixteen TMS 4164s (64 K X 1 bit each) for TM 990/102-3
EPROM: Two TMS 2516s (2 K X 8 bits each) or
Two TMS 2532s (4 K x 8 bits each) or
Two TMS 2564s (8 K x 8 bits each)

Board Dimensions: See Figure 1-2.

Environmental Specifications
Humidity: 5 to 95 percent, noncondensing
Storage temperature: -40° to 85° C
Shock: 2-inch vertical drop
Vibration: 1 G @ 5 to 80 Hz
0.3 G @ 80 to 500 Hz

1.3 REFERENCE DOCUMENTS

The following documents provide supplementary information for the TM 990/102 User's Manual and are included in the documentation kit.

- TMS 9900 Microprocessor Data Manual
- TMS 9901 Programmable Systems Interface Data Manual
- TMS 9902 Asynchronous Communication Controller Data Manual

Other manuals containing supplementary information for the TM 990/102 but which are not included with the documentation kit are:

- TMS 990 Computer, TMS 9900 Microprocessor Assembly Language Programmer's Guide (P/N 943441-9701)
- TM 990/404 MAPBUG Monitor User's guide and Listing (P/N 1602172-9701)
- TM 990/502 Cable Assembly (EIA RS-232-C)
- TM 990/510A, 520A, and 530 Card Cages (P/N 1602126-9701)
- TM 990 System Specification (P/N 1602059-9701)
- Memory Mapping Using SN54/LS610 thru SN54/74LS613 Application Report (Bulletin SCA-205)

1.7 GLOSSARY

The following are definitions of terms used with the TM 990/102. Applicable areas in this manual are given in parentheses.

Absolute Address: The actual memory address in quantity of bytes. Memory addressing is usually represented in hexadecimal from 0000_{16} to $FFFF_{16}$ for the TMS 9900 address bus. The TM 990/102 has extended addressing capability to $FFFFF_{16}$ (megabyte).

Alphanumeric Character: Letters, numbers, and associated symbols.

ASCII Code: a seven-bit code used to represent alphanumeric characters and control (Appendix C).

Assembler: Program that translates assembly language source statements into object code.

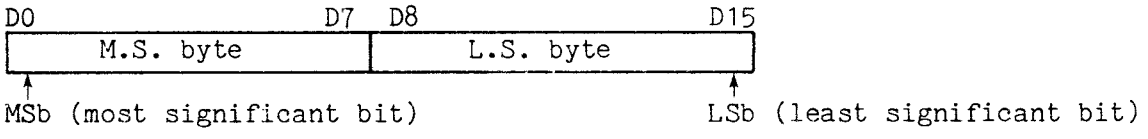
Assembly Language: Mnemonics which can be interpreted by an assembler and translated into an object program (section 7.6).

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.

BUSCLK- Bus Clock: This signal is generated on the designated primary processor. This processor should be in chassis slot 1. BUSCLK- is distributed throughout the system for synchronization of memory and bus arbitration functions. Synchronization is to the LOW-to-HIGH transition (rising edge) of BUSCLK-.

Byte: Eight bits of binary data or half a word.



Carry: A carry occurs when the most significant bit is carried out in an arithmetic operation (i.e., result cannot be contained in only 16 bits), (section 7.3.3.4).

Central Processing Unit (CPU): The "heart" of the computer: responsibilites include instruction access and interpretation, arithmetic functions, I/O memory access. The TMS 9900 is the CPU of the 102. A CPU can be either a primary or secondary processor.

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

CRU (Communications Register Unit): The TMS 9900's general purpose, command-driven input/output interface. The CRU provides up to 4096 directly addressable input and output bits (section 7.7).

CRUCLK-: The CRU clock is a tri-state line output from the current bus master during CRU output operations. CRU output modules connected to the system bus can clock the data from the CRUOUT line on the LOW to HIGH transition of CRUCLK-.

CRUIN: The serial data input line to the current bus master. During CRU input operations the processor samples the CRUIN line to read the data bit whose source is identified by the 12-bit address field (A3 to A14). This is a tri-state line to which all CRU input modules are connected.

CRUOUT: The serial tri-state output data line from the current bus master. During CRU output operations the processor puts the data on this line which is to be placed into the destination address.

DBIN Data Bus In: This tri-state signal is driven active HIGH when the current bus master is doing a READ memory access, and active LOW when the current bus master is doing a WRITE memory access. A primary processor may drive the data bus with DBIN HIGH (in order to allow an external processor to decode an external instruction) when doing an access to on-board memory. DBIN is used to control data buffer direction on memory modules.

DRAM: Dynamic Random Access Memory.

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

EIA Port: An I/O port that operates according to the EIA RS-232-C specification.

EPROM: Erasable Programmable Read Only Memory. See Read Only Memory.

Hexadecimal: Numerical notation in the base 16 (Appendix D).

HOLD- Hold: This open collector signal is driven LOW to request control of the bus from the primary processor.

HOLDA Hold Acknowledge: This normally inactive (LOW) tri-state signal is driven HIGH by the primary processor to indicate that system bus is under control of a secondary processor or a DMA module.

IAQ Instruction Acquisition: When active (HIGH), this signal indicates that the primary processor is performing an instruction fetch memory cycle. IAQ may be used as a diagnostic aid to detect illegal codes. IAQ is an optional signal, and may only be generated by primary processors.

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 (section 7.5.5).

Indirect Addressing: The effective address is the contents of a register (section 7.5.2).

INT1- through INT15- Level n Interrupt: When active (LOW), INT1- through INT15- indicate that the corresponding interrupt is active. Interrupts are driven by open-collector devices. The interrupt levels are ranked with level 1 having the highest priority and level 15 having the lowest priority. Interrupts must remain active until acknowledged by the processor under software control.

Interrupt: A signal used to gain the processor's attention.

Interrupt Disabled: A level of interrupt that is not allowed by the interrupt mask to be accepted by the processor.

Interrupt Enabled: A level of interrupt that is allowed by the interrupt mask to be accepted by the processor.

Interrupt Mask: A value which determines what levels of interrupts the processor will accept or ignore. The interrupt mask value is placed in bits 12 through 15 of the TMS 9900's Status Register (a LIMI instruction can be used to load this register with a value).

I/O: The input/output lines are the signals which connect an external device to the data lines of the TMS 9900.

IORST- I/O Reset: This signal is generated by the designated primary processor. When active (LOW), it indicates a system reset and that all I/O modules should be reset to their initial or idle condition. This signal is active for at least two cycles of REFCLK-. IORST- is driven LOW as a result of either PRES- line being LOW or a reset instruction (RSET).

Kilo: (K) - 1,024.

Least Significant Bit (LSB): Bit having the smallest value (smallest power of base 2): represented by the right-most bit.

LED: Light Emitting Diode.

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

Load: Transfer control to operating system using the equivalent of a BLWP instruction to vectors in upper memory ($FFFC_{16}$ and $FFFE_{16}$). See Reset.

Loader: Program that places one or more absolute or relocatable object programs into memory (Appendix G explains object code).

Machine Language: Binary code that can be interpreted by the CPU (Table 7-4).

Mapper Off Mode: When the TM 990/102 operates in this mode, the Mapper Chip passes the address from the processor to the memory bus and does not extend the addressing capability.

Mapper On Mode: When the TM 990/102 operates in this mode, the Mapper Chip drives the extended address (XA0 to A3) lines to extend the processor's addressing capability.

Mapper Registers: Registers which reside in the memory space of the processor. They are loaded with 4 K-byte memory page addresses to be used to drive extended addresses.

Mapper Register Access: The Mapper Registers are accessible to the user. This accessibility can be programmed to coincide with the Mapper Off or Mapper On mode.

Maskable Interrupt: Interrupt level the processor can be programmed to ignore.

Mega: (M) - 1,048,576 or 2^{20}.

MEMCYC- Memory Cycle: This tri-state signal is driven LOW by the current bus master to indicate that a memory cycle is in progress. The rising edge of MEMCYC- indicates that this memory cycle is complete. In a Single Busclock Memory Access (SBMA), the rising edge of MEMCYC- occurs at the next rising edge of busclock. MEMCYC- may be used to control output data buffer enable on memory modules.

MEMEN- Memory Enable: This tri-state signal is driven LOW by the current bus master to indicate that a memory (or memory-mapped I/O) access is taking place. This signal may remain LOW for several consecutive memory cycles. It is inactive (HIGH) during non-memory cycles.

Memory Mapping: A method by which the processor's addressing capability can be increased.

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; the left-hand bit representing the highest power of base 2. This bit is often used to show sign with a 1 indicating negative and a 0 indicating positive.

Nibble: 4-bits of binary data.

Nonmaskable Interrupt: An interrupt level that the processor cannot be programmed to ignore.

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 the instruction (section 7.5).

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), (section 7.3.3.5).

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 (section 7.3.3.6).

PC Board (Printed Circuit Board): A copper-coated fiberglass or phenolic board on which areas of copper are selectively etched away, leaving conductor paths forming a circuit. Various other processes such as soldermasking and silkscreen markings are added to higher quality PC boards.

PRES- Power On Reset: Normally generated by the system power supply during power up. This open-collector signal, when active (LOW), resets the designated primary processor which, in turn, issues an IORST- to the system. PRES- must remain active for 100 nanoseconds after all power voltages are stable. This asynchronous signal is intended for system initialization or full system reset if activated after initialization.

Primary Processor: A CPU module which does not contain bus arbitration logic (it controls the bus), but which does include the capability to enter a "hold" state to allow for DMA operations on the bus. A primary processor may or may not be resident in any given system configuration but only one may be resident in a given system.

Program Counter (PC): Hardware register that points to the next instruction to be executed or next word to be interpreted (section 7.3.1).

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 burner. Some PROM's can be erased (EPROM's) by exposure to ultraviolet light.

READY: This normally HIGH open-collector signal can be pulled LOW during an Acknowledged Ready Memory Access (ARMA) by the addressed memory (or memory-mapped I/O) module to allow that module additional time to process the other bus signals on-board. The current bus master enters a wait state until the addressed module releases READY.

REFCLK- Reference Clock: This 3 MHz $\pm$ 0.05% signal is generated on the designated primary processor. REFCLK- is distributed throughout the system as a timing reference for I/O devices, timers, and counters.

Reset: Transfer control to operating system using the equivalent of a BLWP instruction to vectors in lower memory (0000_{16} and 0002_{16}). See Load.

RESTART-: When active (LOW), this open-collector signal causes the processor to execute a nonmaskable interrupt (NMI) trap. No resets are generated. RESTART- is an asynchronous system input signal and must remain active for a minimum of 100 nanoseconds.

Secondary Processor: A CPU module which includes bus arbitration logic. Multiple secondary processors may be resident in any given system, either with or without a primary processor.

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 (section 7.3.3).
Triggered Interrupt: An interrupt that has gone active LOW.

Vectors: A two-word block of memory that contains the work space pointer and the program counter values to be loaded when the microprocessor needs to execute the related code.

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 (section 7.4).

Workspace Pointer (WP): Hardware register that contains the memory address of the beginning (register 0) of the workspace area (section 7.3.2).

SECTION 2

INSTALLATION AND OPERATION

2.1 GENERAL

This section explains procedures for unpacking and setting up the TM 990/102 board for operation. This section assumes (1) the TM 990/404 monitor has been purchased and is resident in EPROMs on the board, and (2) that a terminal suitable for connection to the main communcations port is used with proper cable assembly.

CAUTION

Be sure that the correct cable assembly is used with your data terminal. Refer to Appendix B for the signal configuration used by the main port. Most RS-232-C compatible terminals, such as a Lear Siegler ADM-1, will require the TM 990/502A cable, or equivalent. A TI 743 or 745 must use a TM 990/503 cable, or equivalent because of the connector on the terminal end of the cable. A TI 733 requires the use of a TM 990/505 cable, or equivalent. Many RS-232-C compatible terminals come with their own cables, and therefore will probably work with no problem. The TM 990/102 does not operate with a TTY-type terminal.

2.2 REQUIRED EQUIPMENT

The basic equipment required, along with appropriate options, is explained in the following paragraphs.

2.2.1 Power Supply

A power supply capable of meeting at least the following specification is required for one TM 990/102 board. A heavy-duty supply is recommended, if possible, especially for supplying the +5 voltage.

Voltage	Regulation	Current
+5V	3%	3.0 A
-12V	3%	0.4 A
+12V	3%	0.5 A

2.2.2 Power Cable/Card Cage

The use of a TM 990/510A, 520A, or 530 card cage facilitates operation and setup. Alternately, one of the following 100-pin, 0.125 inch (center to center) PCB edge connectors may be used to interface with connector P1, such as with wire-wrap modules:

- TI H431111-50 (TI H3211-50)
- Amphenol 225-804-50
- Viking 3VH50/9CND5
- Elco 00-6064-100-061-001

2.2.3 Miscellaneous Equipment

A Volt/Ohmmeter is needed to measure completed/open connections and to verify power supply voltages and connections.

If any custom connections are required, a soldering iron (25-45 watt), rosin core solder, and wire are needed. Suggested wire sizes are 18 AWG insulated stranded wire for power connections, 24 AWG insulated stranded wire for I/O connections.

2.3 UNPACKING

Lift the TM 990/102 board from its carton and remove the protective wrapping. Check the board for shipping damages. If any discrepancy is found, notify your TI distributor.

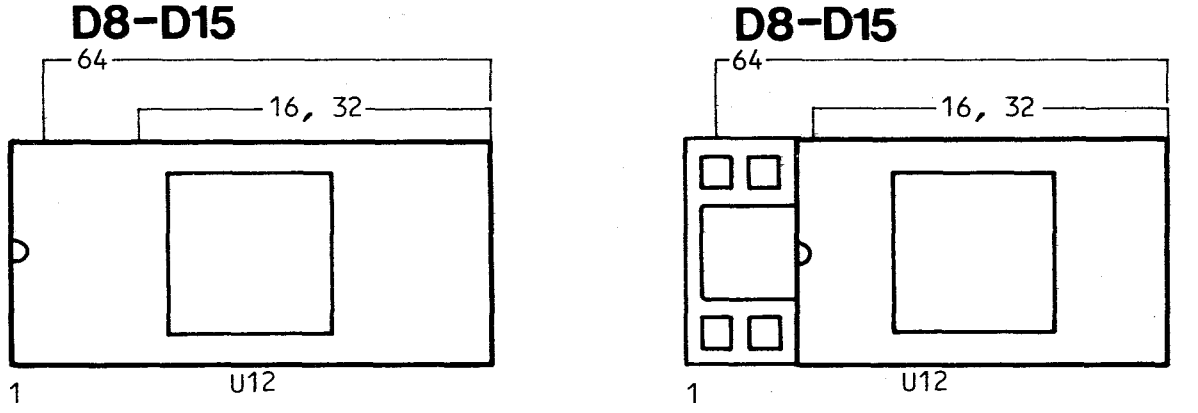
Verify that data manuals for the following devices are included:

- TMS 9900 Microprocessor
- TMS 9901 Interrupt Controller (Programmable Systems Interface)
- TMS 9902 EIA Controller (Asynchronous Communications Controller)

Appenndix H contains the data sheet for the 74LS612 memory mapper.

2.4 EPROM INSTALLATION

Figure 2-1 shows the correct configuration for inserting different ERPOMs into sockets U10 and U12. Note that in both examples shown, socket pin 1 is in the lower left corner; however, EPROM pin 1 does not go into this pin socket for the 24-pin EPROMs (TMS 2516s or TMS 2532s). As shown in (b) of this figure, the 24-pin EPROMs leave pins 1 and 2 blank. As shown in Figure 2-1, all EPROMs (including the 28-pin TMS 2564) fit onto the socket with their rightmost pins inserted in the rightmost pin receptors of the socket (as seen with connector P1 on the bottom).



(a) TMS 2564 INSERTED

(b) TMS 2516 OR TMS 2532 INSERTED

FIGURE 2-1. EPROM INSERTION IN SOCKETS U10 AND U12

TABLE 2-1. EPROM JUMPER SETTINGS

EPROM Configuration	Jumper Configuration					
	J1	J2	J3	J4	J5	J6
TMS 2516	NC	C	C	1-2	2-3	2-3
TMS 2532	NC	NC	C	1-2	2-3	1-2
TMS 2564	NC	NC	NC	2-3	1-2	1-2
All EPROM off	C	(other jumpers not applicable)				

C = connected; NC = not connected

2.5 JUMPER SELECTION, POWER AND TERMINAL HOOKUP

These procedures assume the TM 990/404 monitor is resident in the required address space (0 - FFF_{16}), and that a terminal and cable are properly connected. The monitor's lower-byte EPROM is placed in U12 and the upper-byte EPROM is placed in U10. Correct placement of EPROMs is covered in section 2.4.

CAUTION

Be very careful to apply correct voltage levels to the TM 990/102. Texas Instruments assumes no responsibility for damage caused by improper wiring or incorrect voltage application by the user.

2.5.1 Jumper Selection

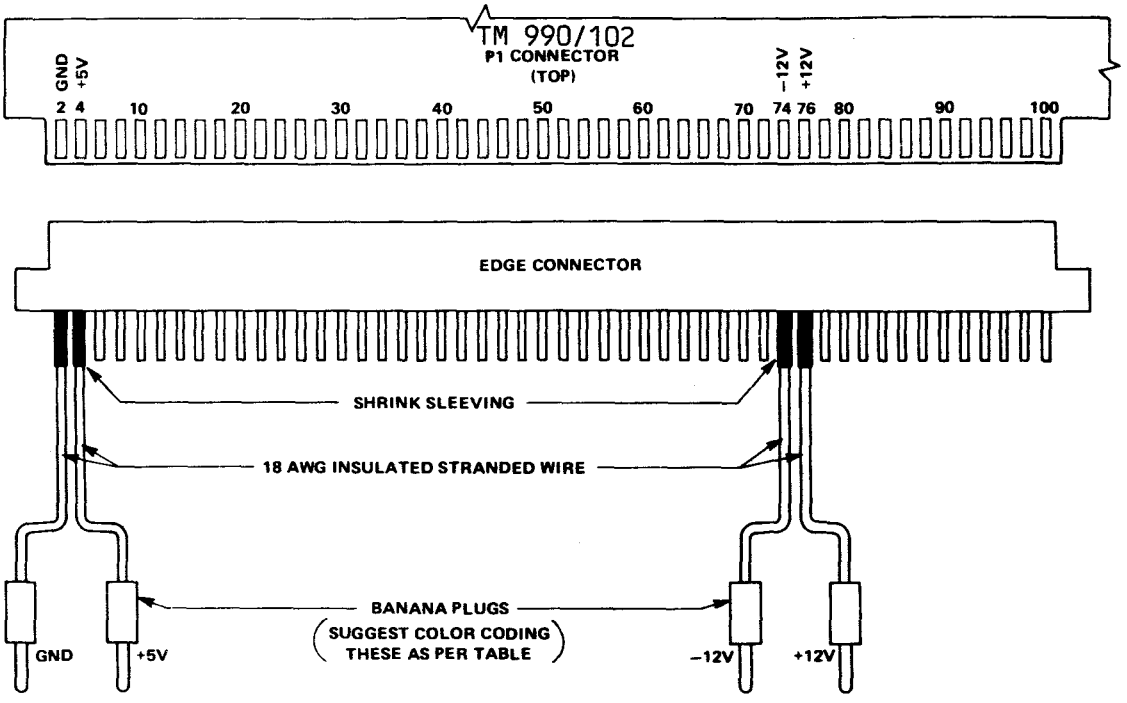
Verify that jumpers are correctly set as shown in Table 2-1.

2.5.2 Power Supply Connections

Figure 2-2 shows how the power supply is connected to the TM 990/102 through connector P1, using a 100-pin edge connector. Be careful to use the correct pins as numbered on the board; these pin numbers may not correspond to the numbers on the particular edge connector used. Check connections with an ohmmeter before applying power if there is any doubt about the quality or location of a connection.

The table in Figure 2-2 shows suggested color coding for the power supply plugs. To prevent incorrect connection, label the top side of the edge connector "TOP" and the bottom "TURN OVER".

Figure 2-3 shows how to correctly place the TM 990/102 in the TM 990/510A card chassis. The processor should be in slot 1 of the chassis unless there is a secondary processor that will have a heavy access of the bus, in which case the secondary processor should have the first slot and the processor the next. This is because the termination resisters for the control bus signals are at the opposite end of the backplane, according to transmission line concepts. Slide the microcomputer into the slot, following the guides. Be sure the P1 connector is correctly aligned in the socket on the backplane, then gently but firmly push the board edge into the edge connector socket.



VOLTAGE	P1 PIN*	SUGGESTED PLUG COLORS
+5V	3, 4, 97, 98	RED
+12V	75, 76	BLUE
−12V	73, 74	GREEN
GND	1, 2, 99, 100	BLACK

*ON BOARD, ODD-NUMBERED PADS ARE DIRECTLY BENEATH EVEN-NUMBERED PADS.

NOTES

1. A TM 990/510A Card Cage or its equivalent provides power supply connections.

2. If you want to make your own cable, be aware that the connecting plugs of various vendors, including TI, do not necessarily use the numbering schemes on the board edge connector. ALWAYS refer to the board edge when wiring a connector.

FIGURE 2-2. POWER SUPPLY HOOKUP

Looking on the backside of the backplane, find the connections for each of the supply voltages and connect them to the power supply.

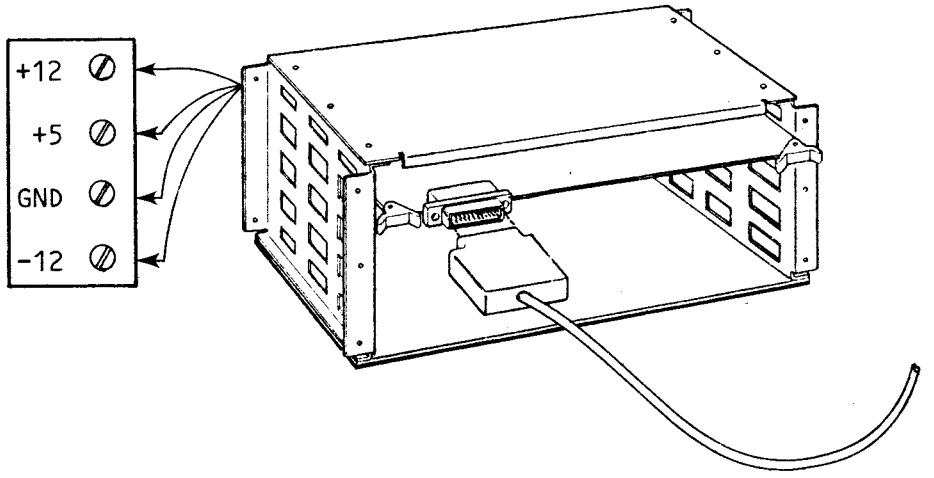


FIGURE 2-3. TM 990/102 BOARD IN A TM 990/510A CHASSIS

CAUTION

BEFORE connecting the power supply to the microcomputer, use a volt-ohmmeter to verify that correct voltages are present at the power supply. After verification, switch the power supply OFF, and then make connections to the chassis as shown in Figure 2-3.

2.5.3 Terminal Hookup

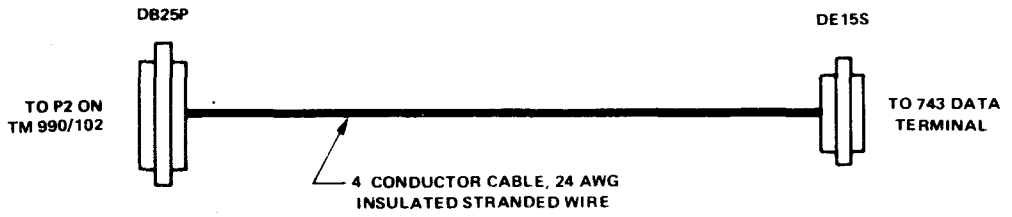
Figure 2-4 shows a cable to connect the TM 990/102 to the TI 743 KSR terminal through connector P2. The DE15S connector attaches to the terminal; a DB25P connector attaches to P2 on the board. A table of point-to-point connections between the connectors is shown in the figure. Figure 2-5 shows a TI 733 connected to a TM 990/102 inserted in a TM 990/510A card cage. Table 2-2 lists available cables for different terminals.

NOTE

The TM 990/102 does not operate with a TTY-type terminal.

TABLE 2-2. TERMINAL CABLES

Cable Model	Terminal
TM 990/502	Most RS-232-C types
TM 990/503A	TI 743 or TI 745
TM 990/505	TI 733 ASR



CONNECTIONS		
PIN ON DE15S	PIN ON DB25P	SIGNAL
13	2	XMIT
12	3	RECV
11	8	DCD
1	7	GND

FIGURE 2-4. 743 KSR TERMINAL CABLE

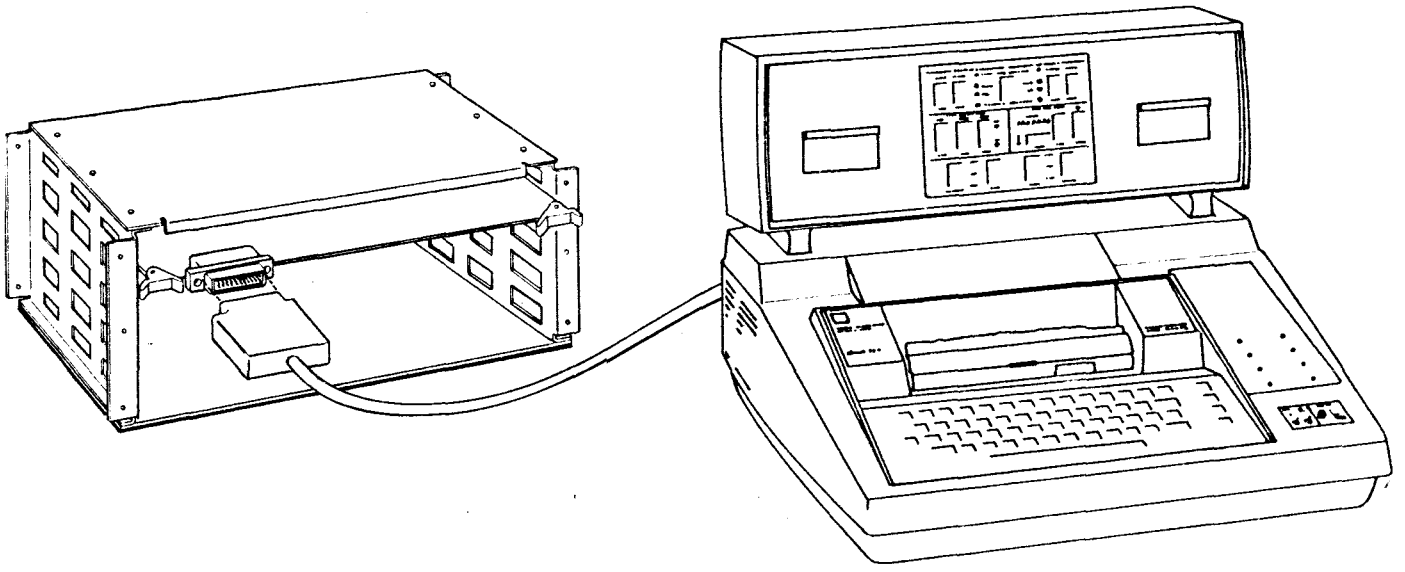


FIGURE 2-5. CONNECTOR P2 CONNECTED TO MODEL 733 ASR

All terminals connected to the microcomputer will have a similar hookup procedure and point-to-point configuration. Terminals for communicating directly with the TM 990/404 monitor must be connected to the main communications port (connector P2) at the corner of the board.

The TM 990/404 monitor operates the I/O port at one of the following baud rates:

110, 300, 600, 1200, 2400, 4800, 9600, or 19200 baud.

There is a 200 ms delay following a carriage return for all baud rates at or below 1200 baud. The delay allows for the printhead travel.

The TMS 9902 asynchronous communication controller is initialized by the TM 990/404 monitor for a seven-bit ASCII character, even parity, and two stop bits (for compatibility with all terminals). At the terminal, set the baud rate of the terminal to one of the above speeds.

The TM 990/404 monitor also uses conversational mode full-duplex communication. Set the communications mode of your terminal to FULL-DUPLEX, and set the OFF/ON LINE switch to ON LINE, or the functional equivalents.

2.5.4 RESET Switch

It is necessary to connect a reset switch to the backplane to cause a full system reset to your application. If you are using a TM 990 card cage there is connection on the barrier strip for this purpose. A momentary switch should be wired to the barrier strip such that one end is connected to the screw marked PRES.B and the other to GND. If you are not using a TM 990 card cage, the momentary switch should be wired between pin 94 on the P1 connector and the board's ground.

2.6 OPERATION

2.6.1 Verification

Verify the following conditions before applying power:

- Power connected to correct pins on P1 connector.
- Terminal cable between P2 connector and terminal.
- Jumpers in correct positions (see Table 2-1).
- Baud rate and communications mode are correctly set at the terminal, and terminal is ON LINE and in FULL DUPLEX.
- TM 990/404 monitor installed at sockets U10 and U12

2.6.2 Power-Up/Reset

a. Apply power to the board and the data terminal.

b. Press the "A" key on the terminal (it may be convenient to press the carriage return key instead; this is also acceptable). The TM 990/404 monitor measures the time of the start bit and determines the baud rate. A carriage return time delay of 200 ms will be provided for all baud rates at or slower than 1200 baud.

c. The monitor prints the banner and, on a new line, a question mark. This is a request to input a command to the monitor's command scanner. These commands are explained in the TM 990/404 monitor's user manual.

NOTE

If control is lost during operation, return control to the monitor by cycling the power, or if a reset switch has been installed, use it and retype the letter "A."

2.7 DEBUG CHECKLIST

If the microcomputer does not respond correctly, turn the power OFF. Do not turn power ON again until you are reasonably sure the problem has been found. The following is a checklist of points to verify.

- Check Power Circuits:

  Check for proper power supply voltages and current capacity. Check for proper connections from the power supply to the P1 edge connnector. Check pin numbers on P1. Check plug positions at your power supply. Look for short circuits. Look for broken connections. Make sure the

board is seated in the chassis or edge connector socket correctly. Be certain that the edge connector socket (if used) is not upside down.

- Check Terminal Circuits:
  - Check for proper cable hookup between the P2 connector and terminal.
  - Check for power at terminal.
  - Check that the terminal is ON LINE.
  - Check that the terminal is in FULL DUPLEX mode.
  - Check for valid baud rate set at terminal.

- Check jumper positions against Table 2-1.

- Check EPROMS for correct placement and bent pins.

- Check orientation of all socketed chips and check for bent pins on the chips.

If the cause of inoperation cannot be found, turn power OFF and call your TI distributor. Before calling, please be sure that your power supply, terminal, and all connectors (use a volt-ohmmeter) are working properly.

2.8 EXAMPLE PROGRAM

The following program blinks the LED at DS2 on a regular basis. It can be loaded into memory using the memory inspect/change command of the TM 990/404 monitor. Enter the object code at the address shown; then load 4000_{16} in the Program Counter and execute with the monitor execute command.

```
Address  Object
 4000     02E0          LWPI  >4020       SET WORKSPACE AFTER PROGRAM
 4002     4020
 4004     020C          LI    R12,>120    CRU ADDRESS OF USER LED
 4006     0120
 4008     1D00  LOOP    SBO   0           TURN ON LED
 400A     06A0          BL    @TIME       TIME DELAY AFTER LIT
 400C     4016
 400E     1E00          SBZ   0           TURN OFF LED
 4010     06A0          BL    @TIME       TIME DELAY AFTER OUT
 4012     4016
 4014     10F9          JMP   LOOP        REPEAT
 4016     04C1  TIME    CLR   R1          SET R1 TO ALL ZEROES
 4018     0601          DEC   R1          DECREMENT BY ONE
 401A     16FE          JNE   $-2         DECREMENT UNTIL R1 = 0
 401C     045B          B     *R11        RETURN
```

SECTION 3

MEMORY MAPPING

3.1 GENERAL

This section explains the operation of the 74LS612 memory mapper. By using this IC, the 64-K byte addressing capability of the TMS 9900 microprocessor can be expanded to 1 megabyte (i.e., the TMS 9900 microprocessor address bus contains lines A0 to A14; the memory mapper allows adding four address lines, XA0 to XA3, to make a 19-bit extended address).

The TM 990/102 memory architecture depends upon the quantity of DRAM populated and the type of EPROM populated. Onboard memory begins at address 0000_{16} and extends up to the highest address populated. If the board is fully populated with 128 K bytes of DRAM, memory extends to address $1FFFF_{16}$, which necessitates using address line XA3 as well as A0 to A14. Note that only A0 to A14 are provided by the TMS 9900 microprocessor; thus, an extended address scheme must be used for memory above 64 K bytes.

In memory access without the mapper, the processor's address lines, A0 to A14, identify a memory word. In the memory mapping mode, the processor's most-significant four address bits (A0 to A3) identify a 12-bit register in the mapper. In a memory access, an eight-bit value is taken from the register and is applied to memory address lines XA0 to XA3 and A0 to A3 (composing the eight MSBs of the 19-bit extended address). The rest of the extended memory address (A4 to A14) is the value on those same address lines at the processor. Thus, the extended address is composed of eight bits from one of the mapper registers (for XA0 to XA3 and A0 to A3), and the rest of the extended address (A4 to A14) is the same as that on the processor. Note that processor address lines A0 to A3 are not part of the extended address; they only select the mapper register. Figure 3-1(a) shows this addressing scheme.

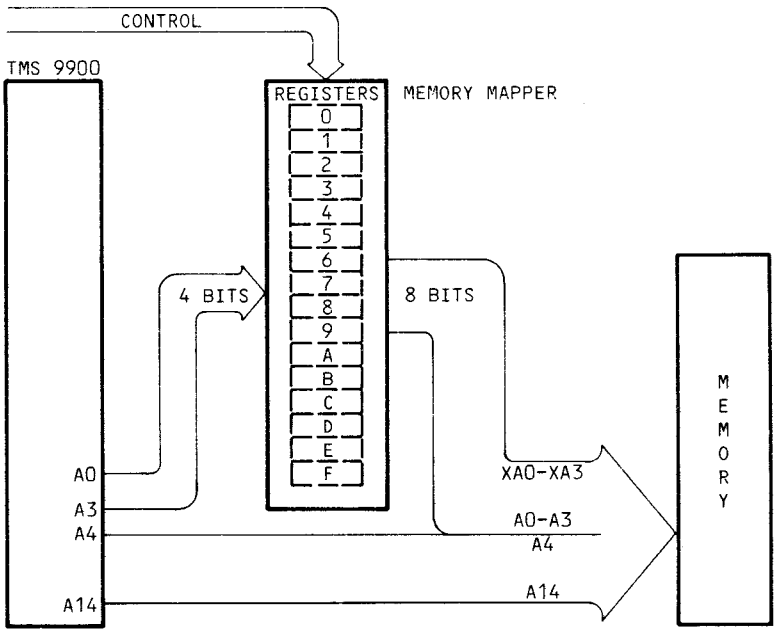
The TM 990/102 operates in one of two modes:

- Mapper-On Mode where a mapper register is used to create a 19-bit extended address; shown in Figure 3-1(a).
- Mapper-Off Mode where the processor's 15 address lines are used to address up to 64 K bytes of memory.

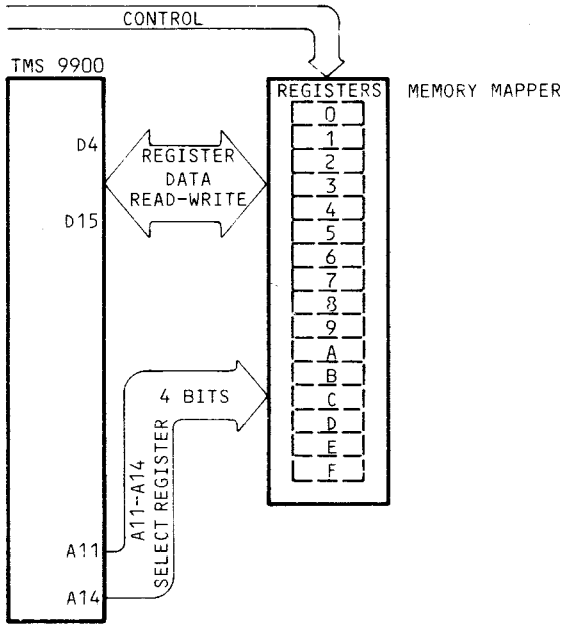
Note that in either of these two modes, addresses 80_{16} to $9E_{16}$ can be selected by software to be either user memory (in DRAM or EPROM) or mapper registers. The mapper's 16 registers can be written to or read; access is via memory mapping. This is depicted in Figure 3-1(b).

Four of the twelve mapper register bits can be used as programmable flags. These modes are further defined in section 3.2.

Figure 3-1(a) depicts the extended addressing. As shown, an extended address is built using eight bits from a mapper register. Because the 11 LSBs of a word address (A4 to A14) are independent of the mapper register contents, memory is accessed in 4-K byte blocks (only in mapper-on mode). Note that in the mapper-on mode, the 11 address LSBs are the same as on the Program Counter. When a new 4-K block is needed, new mapper register contents are needed. The eight MSBs of the extended address (XA0 to A3) select one of the 256 4-K blocks.



(a) Mapper-On Mode, Extended Address Operation



(b) Mapper-Register Access

FIGURE 3-1. BLOCK DIAGRAM USING THE MEMORY MAPPER

Each mapper register contains 12 bits. The extended address uses only the eight least-significant bits (LSBs) of each register; thus the four unused most-significant bits (of the register's 12 bits) can be used as flags.

In summary, during an extended address sequence, the TMS 9900 places a 15-bit address value on its address bus; the four MSBs (A0 to A3) select one of the 16 mapper registers. The eight-bit value in the register becomes the value on address lines XA0 to XA3 (extended address lines) and on A0 to A3 (the four MSBs of the regular TMS 9900's address bus which selected the register). The processor's remaining address lines, A4 to A14, make up the remainder of the extended address.

3.2 MEMORY MAPPER MODES

The following is a summary of the two memory-mapper modes. The first is used to load the mapper registers; the second and third are used to turn off or turn on the mapper depending upon whether or not a mapper register is needed to develop an extended address. A detailed explanation of each is provided in sections that follow.

NOTE

If mapper register access has been granted, memory addresses 80_{16} to $9E_{16}$ will remain as mapper registers regardless of the mapper operating in one of these two modes. When not in either of these two modes (via a RSET instruction or powerup), user memory addresses will reside in this address space (instead of mapper registers).

- Mapper-On Mode: Mapper is used for extended addressing (19 address lines). Eight-bit values from a mapper register are applied to memory address lines XA0 to XA3 and A0 to A3, providing a 19-bit address. This allows addressing of memory from 00000_{16} to $FFFFF_{16}$. This mode is entered by executing the CKON instruction. When entered, the four MSBs of the 15-bit processor's address bus select the desired register in the mapper, and the eight LSBs of the register are applied to the eight memory address lines XA0 to XA3 and A0 to A3.

- Mapper-Off Mode: Mapper is not used. Addressing is via the proocessor's 15 address lines (A0 to A14), allowing addressing of memory from 0000_{16} to $FFFF_{16}$. This mode is entered by executing the RSET instruction or by enabling IORST- (such as by an external RESET or RESTART at pins P1-93 or P1-94 respectively). The mapper comes up in this mode from a powerup or a hardware RESET. In this mode the mapper registers are not accessible, and addresses 0080_{16} to $009F_{16}$ will be memory addresses (not mapper register addresses).

3.2.1 Mapper-Register Access

A primary purpose for mapper register access is to set a register to a specified value. Figure 3-2 shows the 16 mapper registers with register A set to a value of $1BD_{16}$. Note that only the eight least-significant bits (LSBs) of the register (BD_{16}) are to be applied to the eight address lines XA0 to A3 (i.e., the four MSBs -- value of 1 in this example -- are ignored for addressing). This is further shown in Figure 3-3.

However, the four most-significant bits of the register's 12 bits can be used as flags since these registers can be easily set and read (this is further explained in section 3.2.3). The following is example code to load mapper register A with the value shown in Figure 3-2. Note that the register memory address is equal to: 80_{16} + (2 x register no.)$_{16}$.

```
LI      R1,>1BD   PLACE MAP REGISTER CONTENTS IN R1
CKOF              ALLOW ACCESSING MAPPER REGISTERS
MOV     R1,@>94   MOVE R1 CONTENTS TO MAPPER-REGISTER A
```

The CKOF external instruction is decoded by onboard hardware to gain mapper-register access. The MOV instruction destination is memory address 0094_{16}, this is the memory-mapped address of mapper-register A. This access is disabled by an RSET external instruction.

CAUTION

Mapper-register access can be disabled by executing RSET. However, this access is not disabled by entering the mapper-on mode (via CKON instruction). If the mapper-on mode is entered while already accessing mapper registers, the mapper registers can still be written to and read. This capability can possibly result in erroneously changing mapper register contents.

3.2.2 Mapper-On and Mapper-Off Modes

If extended addressing is not used, the four MSB's of the processor's address bus (A0 to A3) are used to select contiguous blocks of memory from 0000_{16} to $F000_{16}$. However, in the mapper-on mode, these four bits select a register in the mapper, and the eight LSBs of this register are used to build the eight MSBs of a 19-bit extended address.

With a value in the desired mapper register (e.g., register A in the example in Figure 3-2), an extended address can be used. As shown in Figure 3-2, when mapper register A is used, the eight MSBs of the extended address will have its eight MSBs equal to BD_{16}. Mapper register A will be called by entering the mapper-on mode and using an address (15-bit) in which the address's four MSBs are the value A_{16} (e.g., $A122_{16}$). To enter the mapper-on mode, execute the external instruction CKON. The following code clears (to all zeroes) extended memory address $BD122_{16}$. (For this to work, the code in section 3.2.1, above, must first have been executed to load mapper-register A as shown in Figure 3-2.)

```
CKON              ENTER MAPPER-ON MODE
CLR     @>A122    CLEAR ADDRESS HEX BD122
RSET              EXIT MAPPER-ON MODE
```

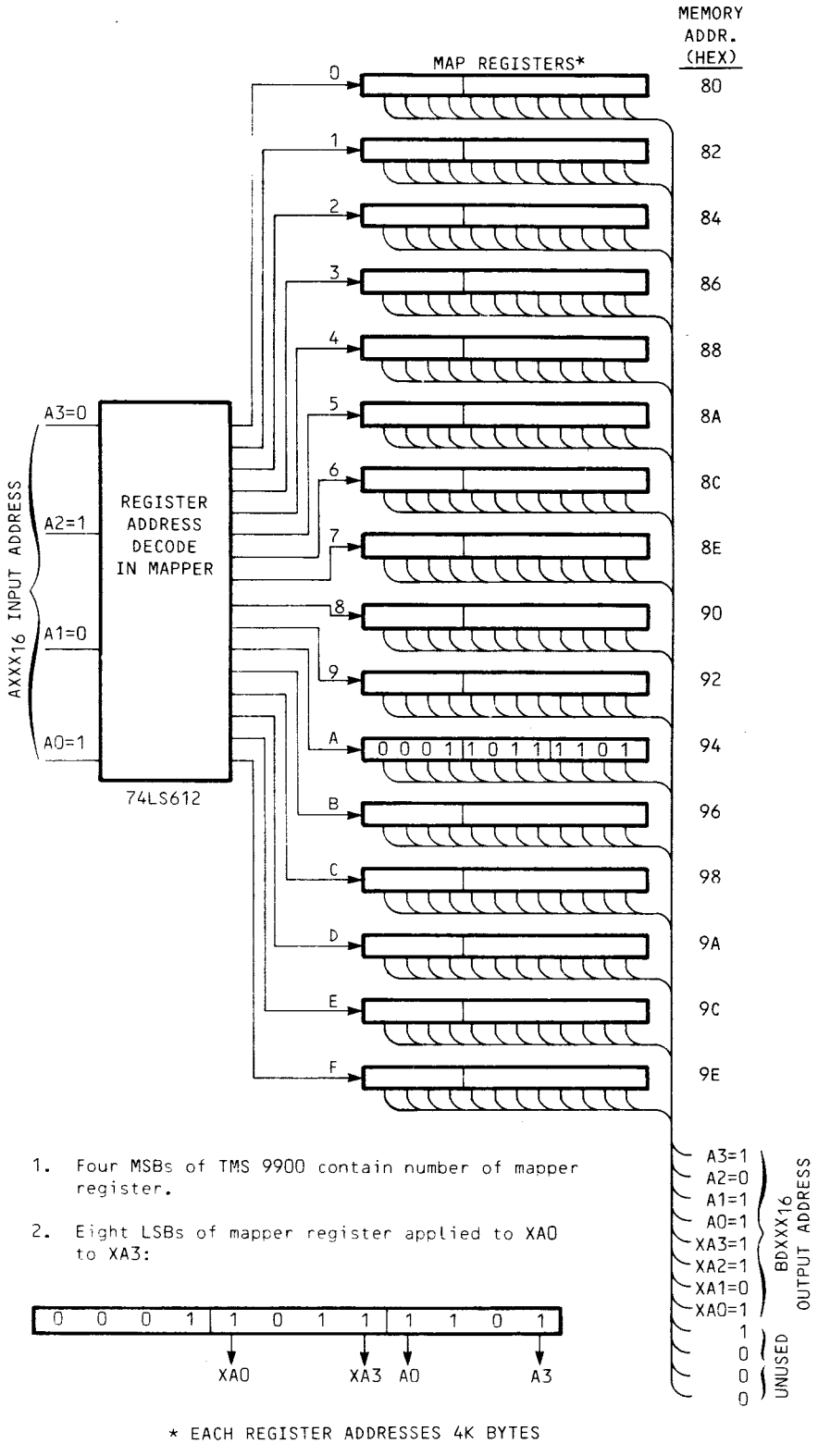
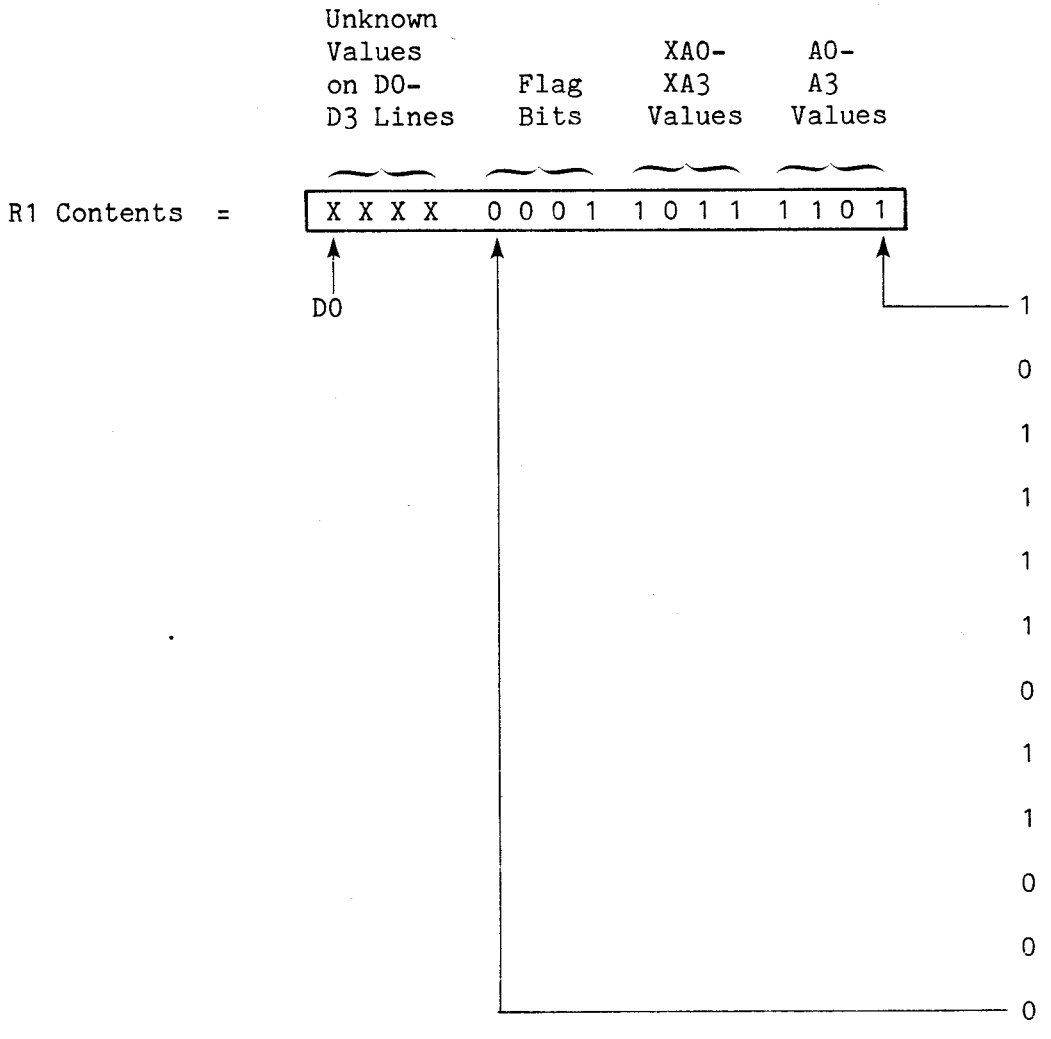


FIGURE 3-2. MAPPER REGISTER SELECTION

```
CKOF                    ENTER MAPPER-REGISTER MODE
MOV    @>0094,R1        READ VALUE IN MAPPER REGISTER 0
RSET                    EXIT MAPPER-REGISTER MODE
```



Mapper Register A (Address 0094_{16})

FIGURE 3-3. READ MAPPER REGISTER ZERO CONTENTS

The first instruction, CKON at bottom of page 3-4, is interpreted by hardware to place the mapper chip in the mapper-on mode. In the second instruction, the first four MSBs of the destination address (A_{16}) selects register A of the mapper (this only happens while in the mapper-on mode). As shown in Figure 3-4, this results in an extended address with a value of $BD122_{16}$ applied to memory address lines XA0 to XA3 and A0 to A14.

NOTE

Two different address buses come into use in the mapper-on mode, the microprocessor's address lines and the memory address bus. First, there is the address on the microprocessor's address lines. The four MSBs of these lines go to the mapper chip in the mapper-on mode. The output of the chip is eight bits which are XA0 to A3 of the memory address bus. The remaining least-significant address lines of the microprocessor are address lines A4 to A14.

```
CKON              ENTER MAPPER-ON MODE
CLR     @>A122    CLEAR ADDRESS HEX BD122
RSET              EXIT MAPPER-ON MODE
```

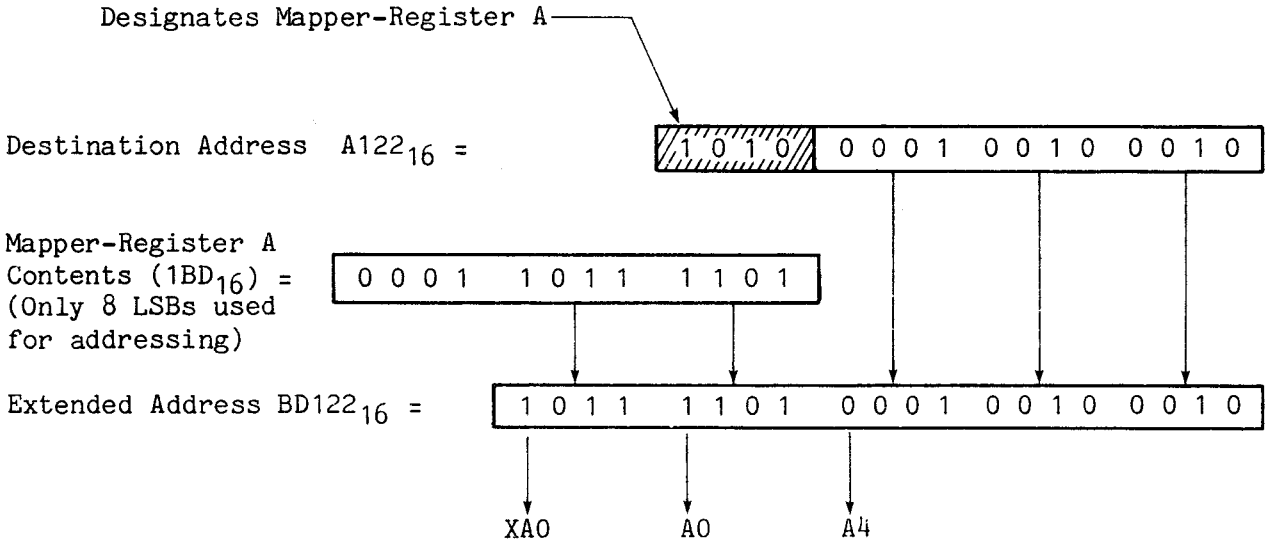


FIGURE 3-4. DERIVING EXTENDED ADDRESS

CAUTION

The RSET external instruction causes the mapper-on mode to be exited, and it disables access to the 16 mapper registers. The mapper-on mode can also be exited with the CKOF command; however, access to the mapper registers continues with this instruction. If access to the mapper registers remains, register contents at memory addresses 0080_{16} to $009F_{16}$ can be changed if they are written to intentionally or accidently. This would not happen if RSET is used to exit the mapper-on mode. However, the RSET instruction also causes IORST.B- to be active at system bus pin P1-88 which can reset or clear I/O devices elsewhere in the system.

3.2.3 Use Mapper Registers as Flags

Only the eight least-signifcant bits (LSBs) of each 12-bit mapper register are used to build the extended address. This allows the four most-significant bits of each register to be used as programmable flags. Each time the flags are read or written to, the mapper must be in the mapper-register mode.

The four bits can be accessed via a move word (M) or move byte (MB) instruction. In either case, the four most-significant bits will be unusable as they will be the floating values on data lines D0 through D3. Data read on data lines D4 to D7 contain flag information while data lines D8 to D15 contain extended address values for XA0 to A3. This is shown in Figure 3-3.

```
CKOF                    ENTER THE MAPPER-REGISTER MODE
MOV     @>0094,R1       READ MAPPER REGISTER A
ANDI    R1,>0F00        SET ALL BUT FLAG BITS TO ZERO
SWPB    R1              FLAG BIT VALUE COMPRISES ALL OF R1
JEQ     EXIT            IF ALL BITS = ZERO, EXIT
  .
  .
  .
```

3.3 CONSIDERATIONS

3.3.1 Mapper-On Mode Creates 4-K Byte Boundaries.

When in the mapper-on mode, memory is divided into blocks of 4-K bytes each. Selection of a particular block is via the first four bits of the program counter which selects the mapper register. This allows different blocks of memory to be dedicated to different functions; each block accessible by a mapper register value.

For an example, the least significant four bits of mapper registers 4 to 7 contain the following:

- mapper register 4: 04
- mapper register 5: 05
- mapper register 6: 06
- mapper register 7: 07

Branches to program counter values with the first four bits a 4, 5, 6, or 7 will cause a branch to a dedicated program beginning at respective memory addresses of $04XXX_{16}$, $05XXX_{16}$, $06XXX_{16}$, or $07XXX_{16}$ (the XXX represents what values are on the TMS 9900's address lines A4 to A14).

In this example, register numbers correspond to the extended-address MSB value in their least-significant byte. This allows direct transfer from one block to another as code crosses a 4-K block boundry. This is because the program counter will naturally progress from (hexadecimal) 04FFE to 05000, or 06FFE to 07000, etc. Another way to understand this is to place a value in the next mapper register that is different than the register number. For example, set mapper register 4 to 04 but set mapper register 5 to 09. In this case, when the program counter progresses from $04FFE_{16}$ to 05000_{16}, the 5 in the latter address will call up register 5 which places 09000_{16} on the memory bus. Thus, program continuity goes from 04FFE to 09000.

3.3.2 RSET to Exit Mapper Modes

The RSET command can be used to exit the mapper-on and mapper-off modes. However, an added result of using RSET is the activation of IORST.B- at system bus pin P1-88 which can reset or clear I/O devices elsewhere in the system.

3.3.3 Interrupts

Several considerations must be taken into account when using interrupts with the TM 990/102:

- The processor goes to memory for interrupt vectors. If the mapper is in the mapper-on mode when the interrupt occurs, a mapper register will be used to determine the address where the vectors are located. Because the processor's address of these vectors always begins with a hexadecimal zero (all interrupt vectors are within the address space of 0000_{16} to $003E_{16}$), interrupts occurring in the mapper-on mode will seek an address using mapper register zero for the extended address.

- If the interrupting program changes the mapper's mode, then the mapper must be returned to that mode before control is returned back to the interrupted program. For example, if a program is running with extended addressing in the mapper-on mode, and the interrupt handler changes this to the mapper-off mode, the mapper on mode must be re-entered before returning to the interrupted program.

3.3.3.1 Interrupt Vectors in Mapper-On Mode

Interrupt vectors can be placed in extended memory; however, this requires that the mapper always be in the mapper-on mode when the interrupt happens. If the mapper is in the mapper-on mode when an interrupt occurs, the contents of mapper register zero will be used to define the interrupt-vector address. This is because the TMS 9900 has a zero in its most-significant four bits in all interrupt vector addresses (0000_{16} to $003E_{16}$). Vector addressing would be simplified by using addressing in lower memory (non-extended). In case interrupts occur while in the mapper mode, mapper register zero could be set to zero to effect this addressing scheme.

Another scheme would be to have interrupt vectors be flexible to match the program in use. In this case, while operating in the mapper-on mode, mapper register zero can be configured to the desired contents.

3.3.3.2 Restore Former Mapper Mode on Return

Possibly, when a program is interrupted, the interrupt service routine could possibly change the mapper register mode from that used by the interrupted program. The system must maintain knowledge of the mapper's previous state so that control can be correctly returned to the interrupted program.

For example, if an interrupted program was operating without extended addressing, the mapper must be in the mapper-off mode when control is returned to that program. Otherwise, control could return to the wrong address.

One scheme to solve this is to dedicate a register in each program as a mapper-mode indicator. The operating program will set this indicator to the mode in which it is operating. This indicator register could be accessed through the workspace return address in register 13 of the new program. Then, before returning control to the interrupted program, the presently executing program can set the mapper to the correct mode. Note that it must be known if the workspace is or is not in extended addressing. To simplify, locate all workspaces without extended addresses.

For example, if workspace register 1 is the mapper-mode indicator, this can be set to a value reflecting the mode being used; such as:

- 1 = Mapper-off mode
- 2 = Mapper-on mode
- 3 = Mapper-register mode

3.3.4 Initialize Mapper Registers

It is good programming practice to initialize the mapper registers to known values. For example, code below 3-5 sets mapper register 0 (sets to 000_{16}) to point to the first 4 K memory block, register 1 (sets to 001_{16}) to point to the second 4 K memory block, etc.

```
CLR  R1          SET REGISTER 1 TO ZERO TO LOAD IN MAP REGISTER 0
LI   R2,>80      R2 POINTS TO FIRST MEMORY MAPPER REGISTER
CKOF             ACCESS MAPPER REGISTERS
MOV  R1,*R2      LOAD MAPPER REGISTER WITH VALUE
INC  R1          ADD ONE TO VALUE TO BE LOADED IN MAPPER REGISTER
INCT R2          R2 POINTS TO NEXT MAPPER REGISTER
CI   R2,>A0      R2 INCREMENTED TO A0?
JNE  LOOP        NO, LOOP UNITL ALL 16 REGISTERS ARE LOADED
RSET             YES, EXIT ACCESS TO MAPPER REGISTERS
RTWP             EXIT ROUTINE
```

The following code (see next page) also initializes the mapper registers; however, the intiial register values have been designated by specific values in DATA statements.

```
       IDT  'SETMAP'
*           THIS ROUTINE LOADS THE MAP REGISTERS WITH
*           THE ADDRESS SET UP IN A TABLE
*
START  EQU  $
       CKOF                    MAKE MAP REGISTERS AVAILABLE
       LI   R0,TABLE           GET TABLE ADDRESS

       LI   R1,>0080           LOAD R1 WITH ADDRESS OF MAP R

LOOP   MOV  *R0+,*R1+          LOAD MAP REGISTER
       CI   R1,>A0             FINISHED LOADING MAP REGISTER

       JNE  LOOP               NO, CONTINUE LOADING
END    EQU  $
*
*
*
TABLE  EQU  $                  SET MAP REGISTERS TO ACCESS
       DATA >20                THIRD 64K-BYTE PAGE IN
       DATA >21                1M-BYTE MEMORY SPACE
       DATA >22
       DATA >23
       DATA >24
       DATA >25
       DATA >26
       DATA >27
       DATA >28
       DATA >29
       DATA >2A
       DATA >2B
       DATA >2C
       DATA >2D
       DATA >2E
       DATA >2F
*
       END  START
```

FIGURE 3-5. CODE TO LOAD SPECIFIC VALUES INTO MAPPER REGISTERS

SECTION 4

PROGRAMMING THE TMS 9901 INTERRUPT CONTROLLER

4.1 GENERAL

The TMS 9901 is the controller for the 15 maskable interrupts on the TM 990/102. It also contains an interval timer that can be programmed (with a 3 MHz clock) in increments of 21.3 microseconds with a total countdown time length of 349.5 milliseconds. The TMS 9901 also controls (via its output port P0) the user-programmable light emitting diode (LED) DS2.

The 15 maskable interrupts are INT1- to INT15-, numerically prioritized with INT1- being the highest priority.

NOTES

1. If you are not familiar with using the Communications Register Unit (CRU) for I/O, please read section 7.7 before proceeding.
2. There is a level zero interrupt, RESET, which is nonmaskable and not monitored by the TMS 9901. RESET is used for powerup initialization and general board initialization with its WP (workspace pointer) and PC (program counter) vectors obtained from addresses 0000_{16} and 0002_{16} respectively, these addresses usually being in EPROM. Another interrupt not monitored by the TMS 9901 is LOAD which obtains WP and PC vectors from addresses $FFFC_{16}$ and $FFFE_{16}$. These addresses are in DRAM so they can be programmed as desired.
3. As stated in the TMS 9901 data book, nine of the interrupt input pins can be programmed as I/O ports. However, the TMS 9901 inputs come from dedicated bus interrupt input lines; thus, the user is cautioned to write only to TMS 9901 bits 0 to 16 (decimal). Writing to bits 17 to 31 is not recommended, and writing to bits 23 to 31 can cause damage. Figure 4-1 shows TMS 9901 connections.

4.2 CRU STRUCTURE

Programming is through the Communications Register Unit (CRU) which is the serial input and output port on the TMS 9900 microprocessor. CRU instructions and addressing are explained in detail in section 7.7 (CRU Instructions).

In general, there are five CRU instructions, three single bit (SBZ, SBO, and TB) and two multibit (LDCR and STCR) CRU instructions. When executed, CRU instructions place an address on the address lines and then either (1) send one (or more) bits over the CRUOUT line of the microprocessor or (2) receive one (or more) bits at the CRUIN line or test the binary value at the CRUIN line. Table 4-1 lists CRU addresses for the board. Table 4-2 lists CRU addresses for the TMS 9901.

In CRU programming, the desired address is first placed in register 12 before CRU instruction execution. When a CRU instruction is executed, bits 3 to 14 of R12 are placed on address lines A3 to A14. An address-line decoder (shown in Figure 6-15) then enables the TMS 9901 so that it can be accessed through the CRUIN (in to processor) or CRUOUT (out from processor) lines. The value received from R12 is usually modified by the CRU instruction adding a signed displacement to the binary values in bits 3 to 14 of the register (bit 15 of register 12 is ignored), and then applying this sum to address lines A3 to A14 of the TM 990/102 to access a particular TMS 9901 bit.

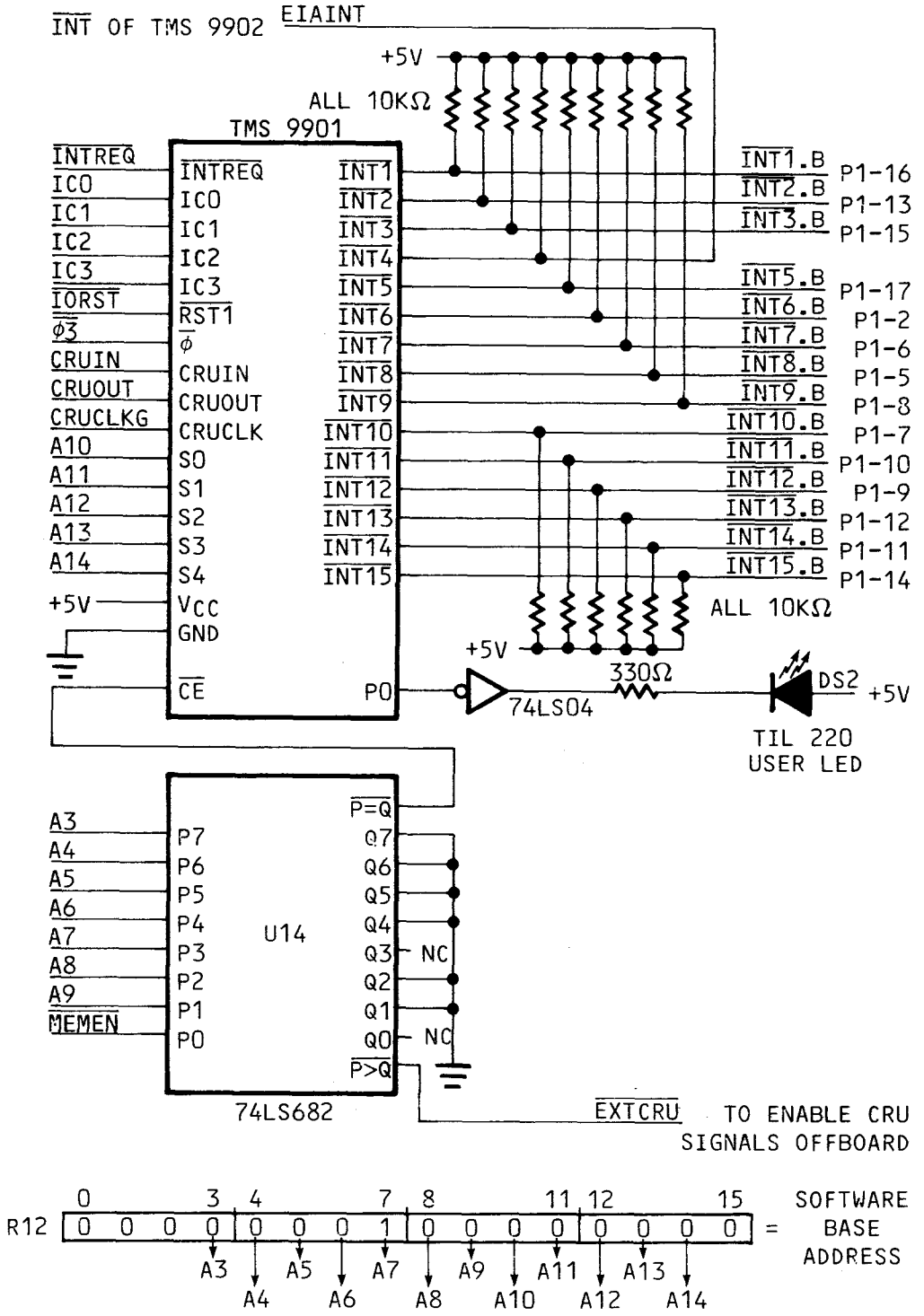


FIGURE 4-1. TMS 9901 CONNECTIONS

TABLE 4-1. CRU ADDRESSES FOR TM 990/102

	Hardware Base Address (Hex)		Software Base Address (Hex)	
	From	To	From	To
Reserved	0000	003F	0000	007E
TMS 9902	0040	005F	0080	00BE
Reserved	0060	007F	00C0	00FE
TMS 9901	0080	009F	0100	013E
User Defined	00A0	0FFF	0140	1FFE

NOTE

Some existing software uses software base address >0180 to >01BE for a second EIA port.

TABLE 4-2. CRU BITS FOR THE TMS 9901

Displacement (Bit No.)		Interrupt Mode [1]Control Bit = 0		Clock Mode [1]Control Bit = 1	
Decimal	Hex	Write	Read	Write	Read
0	0	Zero	Zero	One	One
1	1	INT1- mask	INT1- pin	[2]CLK1	CLK1
2	2	INT2- mask	INT2- pin	CLK2	CLK2
3	3	INT3- mask	[3]INT3- pin	CLK3	CLK3
4	4	[4]INT4- mask	EIAINT- (9902)	CLK4	CLK4
5	5	INT5- mask	INT5- pin	CLK5	CLK5
6	6	INT6- mask	INT6- pin	CLK6	CLK6
7	7	INT7- mask	INT7- pin	CLK7	CLK7
8	8	INT8- mask	INT8- pin	CLK8	CLK8
9	9	INT9- mask	INT9- pin	CLK9	CLK9
10	A	INT10- mask	INT10- pin	CLK10	CLK10
11	B	INT11- mask	INT11- pin	CLK11	CLK11
12	C	INT12- mask	INT12- pin	CLK12	CLK12
13	D	INT13- mask	INT13- pin	CLK13	CLK13
14	E	INT14- mask	INT14- pin	CLK14	CLK14
15	F	INT15- mask	INT15- pin	RST2-	INTREQ
16	10	[1]LED DS2			

NOTES: 1. Control bit designates the use of CRU bits 1 to 15 only (not 16).

2. Write a non-zero value to CLK14 (MSB) to CLK1 (LSB) to set the clock register value, or read these bits to obtain the clock read register value. Registers are set to all zeroes by a hardware reset (IORST- active), or by writing all zeroes to the clock register (not by setting RST2- low at bit 15).
3. Interrupt level 3 is the clock interrupt. Reading this bit reads the value on external pin INT3-, not a clock interrupt status bit.
4. INT4- comes from the TMS 9902 only (no connection to system bus). INT4- can be activated by one of four different events:
   - Receipt of a character at P2 (Receive Buffer full)
   - Character transmitted at P2
   - TMS 9902's counter counts down to zero
   - Data change at TMS 9902's DSR- or CTS- inputs

4.3 INTERRUPTS

4.3.1 Definitions

- Interrupt Enabled: The interrupt mask is set to 1, meaning the mask is enabled to allow an interrupt should priority conditions also permit it.

- Interrupt Disabled: The interrupt mask is set to 0; if an interrupt request occurs, no interrupt will occur.

- Interrupt Active: An interrupt request (low input) is applied to the (external) interrupt pin of the TMS 9901 or the TMS 9901 clock has decremented to zero.

4.3.2 General Interrupt Operation

Interrupts are used to shift a microprocessor's (uP's) capabilities to another event that needs immediate attention. The interrupt is a request for the uP to halt its present work, and take a look at another process that has reached a point requiring uP intervention. One example is a system of four terminals, with each keyboards monitored by its own uP. Instead of dedicating four uPs to listening to four terminals, wasting much computing time between keystrokes, interrupts allow one uP to monitor all the terminals, with each keystroke causing an interrupt that momentarily shifts the uP's attention to one terminal, then another.

Interrupts can be generated onboard or can come to the TM 990/102 via the bus interrupt lines at connector P1 (INT1- to INT15-). Onboard interrupts include the timer at the TMS 9901 and interrupt outputs from TMS 9902; these are dedicated to interrupt levels 3 and 4 respectively.

Interrupts can make their presence known in one of two ways: (1) enabling interrupt-input logic to the uP or (2) the uP periodically checking (polling) interrupt lines for events requiring its attention. The TMS 9901 allows approaching interrupts in either of these ways.

In the first method above, the TMS 9901 verifies priorities before sending the request to the TMS 9900. This logic includes the following input pins on the processor:

- IC0 to IC3 which contain the binary value of the interrupt

- INTREQ- which signals that an interrupt is being requested.

If the microprocessor polls the TMS 9901, any number of the interrupt levels 1 to 15 may be active at the TMS 9901 at one time; however, the uP does the selection as to which request to answer. The uP can poll each bit to see if a request is active, then select which to acknowledge. Table 4-2 is a summary of the 15 levels of interrupts; these can be enabled, disabled, or polled using CRU bits 1 to 15.

Note that there are two special dedicated interrupts:

- INT3- which can be programmed as an external interrupt (from system bus at P1) or as the TMS 9901's clock interrupt which becomes active when the clock counts down to zero. When the clock is counting, this interrupt level ignores inputs from P1.

- INT4- which is the EIAINT- input from the TMS 9902. This interrupt is not connected to connector P1; it connects only to the INT output of the TMS 9902. When active, one of four conditions is indicated at the TMS 9902:

  - Receipt of a character at P2 (Receive Buffer full)
  - Character transmitted at P2
  - TMS 9902's interval counter counts down to zero
  - Data change at TMS 9902's DSR- or CTS- inputs

  These are further explained in Section 5.

When an interrupt request is enable and active, the TMS 9901 INTREQ- signal becomes active, going to the INTREQ- input to the TMS 9900 along with a binary value of the interrupt (1 to 15) to IC0 (MSB) to IC3 of the microprocessor.

Several steps must occur to make the interrupt process work. It is important to understand that the interrupt must be enabled at both the TMS 9900 and the TMS 9901. In order, these are (only step 2 applies to polling):

1) Interrupt enabled at the TMS 9901 via CRU (i.e., its interrupt mask at TMS 9901 was set to one while the TMS 9901 was in the interrupt mode).

2) Interrupt must be active at TMS 9901 by one of the following:
   - A low input to a P1 interrupt pin (including a low to INT4- by the TMS 9902), or
   - The TMS 9901 clock has reached a countdown to zero (INT3-).

3) No higher-priority interrupt is active and enabled at the TMS 9901.

4) The microprocessor interrupt mask value is equal to or a higher value than the interrupt value being requested (e.g., if an interrupt 4 is active at the TMS 9901, the TMS 9900's Status Register interrupt value (4 LSBs) must be any number from 4 to 15). INT1- is the highest-priority maskable interrupt; INT15- is the lowest priority.

5) Interrupt vectors programmed with the correct WP and PC vectors. Interrupt vectors are shown in Figure 4-2.

4.3.3 Enabling, Disabling, and Polling Interrupts

Except for INT3-, external inputs to pins INT1- to INT15- can be polled for activity. An activating low input to the TMS 9901 pin is read as a low (zero) through the CRU. As long as the external interrupt input (active low) to the TMS 9901 remains active, an interrupt can be sensed by polling. CRU bits (sensed low) are shown in Table 4-2. When the external interrupt input is released, the polled signal goes high due to pullup resistors on the interrupt input lines. The external input to the interrupt can be polled even though the interrupt mask for the interrupt is not enabled (not set to one) at the TMS 9901.

To enable an interrupt at the TMS 9901, do the following:

- Enter the interrupt mode by setting the control bit (bit 0) to zero (setting to one enters the clock mode)

MEMORY ADDRESS		
0000 0002	WP VECTOR, INTERRUPT 0 PC VECTOR, INTERRUPT 0	RESET FUNCTION UNMASKABLE
0004 0006	WP VECTOR, INTERRUPT 1 PC VECTOR, INTERRUPT 1	
0008 000A	WP VECTOR, INTERRUPT 2 PC VECTOR, INTERRUPT 2	
000C 000E	WP VECTOR, INTERRUPT 3 PC VECTOR, INTERRUPT 3	
0010 0012	WP VECTOR, INTERRUPT 4 PC VECTOR, INTERRUPT 4	
0014 0016	WP VECTOR, INTERRUPT 5 PC VECTOR, INTERRUPT 5	
0018 001A	WP VECTOR, INTERRUPT 6 PC VECTOR, INTERRUPT 6	
001C 001E	WP VECTOR, INTERRUPT 7 PC VECTOR, INTERRUPT 7	
0020 0022	WP VECTOR, INTERRUPT 8 PC VECTOR, INTERRUPT 8	
0024 0026	WP VECTOR, INTERRUPT 9 PC VECTOR, INTERRUPT 9	
0028 002A	WP VECTOR, INTERRUPT 10 PC VECTOR, INTERRUPT 10	
002C 002E	WP VECTOR, INTERRUPT 11 PC VECTOR, INTERRUPT 11	
0030 0032	WP VECTOR, INTERRUPT 12 PC VECTOR, INTERRUPT 12	
0034 0036	WP VECTOR, INTERRUPT 13 PC VECTOR, INTERRUPT 13	
0038 003A	WP VECTOR, INTERRUPT 14 PC VECTOR, INTERRUPT 14	
003C 003E	WP VECTOR, INTERRUPT 15 PC VECTOR, INTERRUPT 15	

FIGURE 4-2. INTERRUPT TRAP LOCATIONS

- Write a one to the bit (1 to 15 to enable interrupt levels 1 to 15).

For example:

```
LI    R12,CRUADR        SET R12 TO (HEX) 0100
SBZ   0                 CONTROL BIT TO ZERO = INTERRUPT MODE
SBO   3                 ENABLE INTERRUPT 3
SBO   4                 ENABLE INTERRUPT 4
```

To disable (mask off) an interrupt at the TMS 9901, do the following:

- Enter the interrupt mode by setting the control bit (bit 0) to zero.
- Write a zero to the mask (bit 1 to 15 to disable interrupt levels 1 to 15).

NOTES

1. It is important to disable an answered interrupt (i.e. LIMI 0) to prevent further bus-request cycles. Also, at least one instruction should be executed after disabling the interrupt at the TMS 9901 and before executing an RTWP (which would return to the original interrupt level). This ensures the interrupt request to the uP is disabled.
2. An interrupt will remain enabled until reprogrammed.
3. An interrupt request will remain active only as long as a low logic level is applied to the external pin.
4. Note that if the clock counts down to zero and its interrupt mask is set to one, it causes INTREQ- to become active; however, polling bit 3 will obtain the value on its external pin; thus, the clock cannot be polled for an interrupt. Also, after countdown, the clock interrupt must be re-enabled by writing a one to its interrupt mask; otherwise, the next countdown will not result in issuing an interrupt.

In order to disable an interrupt, write a zero to its mask. For example, the following will mask off the INT2- interrupt:

```
LI    R12,CRUADR      9901 Control CRU Address (>100)
SBZ   0               ENTER INTERRUPT MODE
SBZ   2               MASK OFF INTERRUPT REQUEST REQ2(-)
```

The clock interrupt is unique in that its mask must be written to each time after a countdown in order for the next countdown to also cause an interrupt. The clock is further covered in section 4.4.

The TMS 9901 can be polled for the value on the input pins to the interrupts. A low indicates an interrupt request is active (low applied to external pin). To poll for active interrupts:

- First enter the interrupt mode (set bit 0 to zero).
- Test bits 1 to 15 for the logic level at pins INT1- to INT15-. A low indicates an interrupt request; a high indicates no interrupt request.

The following is an example to poll for interrupts 4, 5, and 6.

```
LI      R12,CRUADR      PLACE HEX 0100 IN R12
SBZ     0               ENTER INTERRUPT MODE
TB      4               IS INT4- ACTIVE?
JNE     INT4            YES, LOW FOUND
TB      5               NO, IS INT5- ACTIVE?
JNE     INT5            YES, LOW FOUND
TB      6               NO, IS INT6- ACTIVE?
JNE     INT6            YES, LOW FOUND
RT                      NO, RETURN TO PROGRAM
```

The TB (Test Bit) instruction reads the value at the bit addressed by the contents of register 12 and the signed displacement and places the value found in EQ (equal) bit of the Status Register. If an active interrupt is found, a zero is placed in the EQ bit (zero found at interrupt bit). With EQ low, the JNE instruction will execute (jump on not equal). Refer to section 7.3.3 for more information on the Status Register.

4.4 INTERVAL AND EVENT TIMER

The TMS 9901 has an internal clock that can be programmed via the CRU. As shown in Table 4-2, there are 14 clock bits with bit CLK1 the LSB and bit CLK14 the MSB of the clock register. A fully loaded clock register (all ones) will contain the decimal value of 16,383; this value is counted down at the rate of 46,875 counts per second with a system clock of 3.0 MHz (the clock decrements at a rate of 3 MHz/64). Each count is 21.3333 microseconds -- computed by 1/(3 MHz/64). When triggered, the clock will count down to zero and an interrupt will be active. Also, if enabled, an interrupt will be sent to the processor.

NOTE

A system clock of 3.0 MHz will be used in this writeup.

A fully loaded clock register will count down in 349.5 milliseconds. The following are examples of countdown times vs. clock register values:

Clock Register Value		Countdown Time
Decimal	Hexadecimal	at 3 MHz
16,383	3FFF	349.5 ms
11,719	2DC7	250.0 ms
5,859	16E3	125.0 ms
4,687	124F	100.0 ms
2,344	0928	50.0 ms

4.4.1 Entering and Exiting the Clock Mode

All access to the clock must be done while in the clock mode. Enter this mode by writing a one to CRU address bit 0 (i.e., set control bit to one). To exit the clock mode, write a zero to CRU address bit 0 (enter interrupt mode).

4.4.2 Starting Clock Countdown

Start the clock by entering the clock mode and writing a timer value to the clock register. When the last bit is entered in the register, the clock begins counting down:

```
LI      R12,CRUADR+2        BIT 1 ADDRESS (START OF CLOCK REG.(>102)) IN R12
LI      R1,11719            CLOCK COUNT FOR 250 MS IN R1
SBO     -1                  CONTROL BIT = 1 = ENTER CLOCK MODE
LDCR    R1,14               APPLY 11719 TO CLOCK REGISTER
```

The above code is straightforward in nature. Actually, this code can be more efficient. Set the software base address to the value represented by CRUADR (0100_{16}). Then, instead of using one instruction to set the control bit and another instruction to load the clock register, both these functions can be done with one LDCR instruction. In this instruction, the first bit (LSB) of the LDCR source address is a zero to set the clock mode. This is followed by 14 bits to set the clock register. This shortened source code is shown in the three source lines in Figure 4-3.

```
LI      R12,>100    CRU ADDRESS OF TMS 9901
LI      R1,>5B8F    CLOCK,>2DC7 COUNTS, AND SET CLOCK MODE BIT
LDCR    R1,15       SET CLOCK VALUE AT CLOCK REGISTER
```

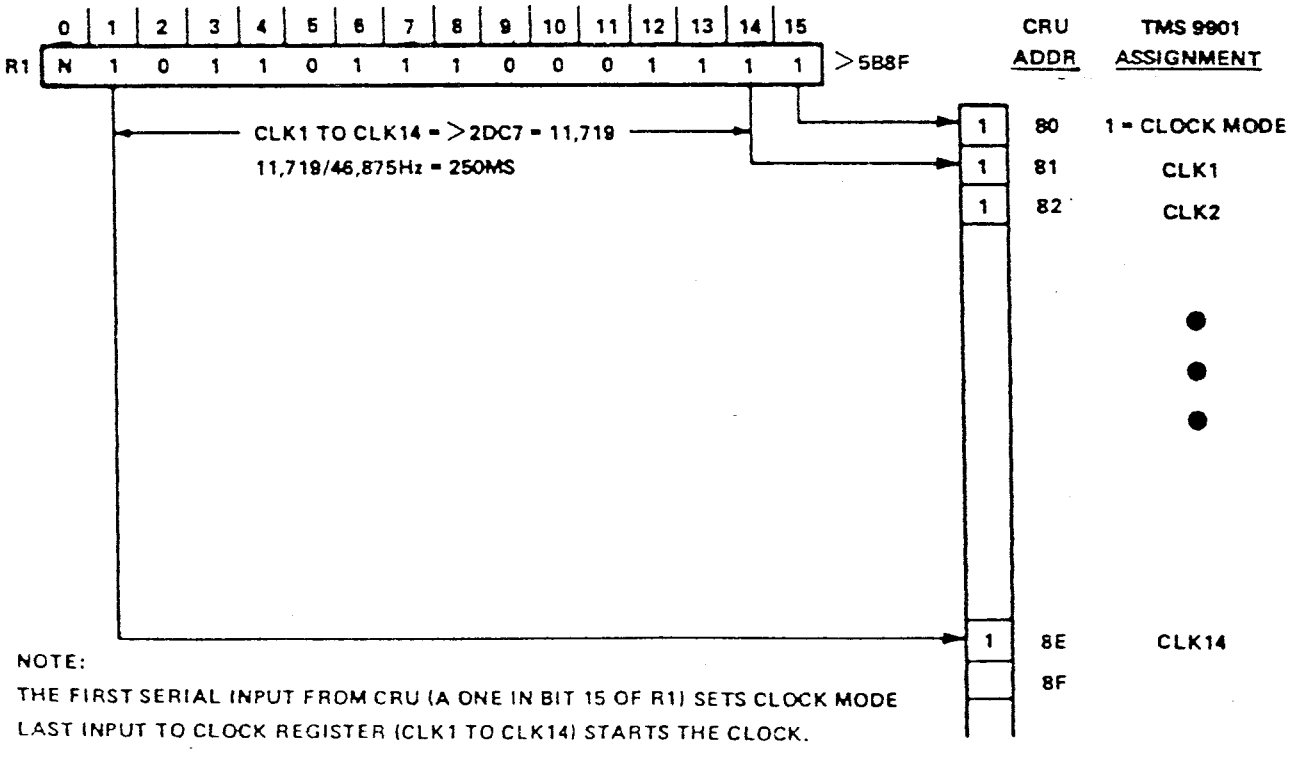


FIGURE 4-3. CODE TO ENABLE AND TRIGGER THE CLOCK ON THE TMS 9901

When the clock counts down to zero, the initial value set in the clock register is reloaded into the clock register and the countdown begins again. After countdown, the clock interrupt must be re-enabled before the next countdown to zero can cause another interrupt. To do this, enter the interrupt mode and write a one to CRU bit 3:

```
LI   R12,CRUADR      TMS 9901 CRU ADDRESS (>100)
SBZ  0               ENTER INTERRUPT MODE
SBO  3               RE-ENABLE CLOCK INTERRUPT
```

If the clock interrupt is not re-enabled after a countdown, the clock decrementer continues counting down but no interrupt is issued.

To disable the clock interrupt, either set the clock interrupt mask to zero, or set the clock register to all zeroes, or cause a hardware reset (e.g., execute RSET to cause IORST- to be active, activating the external RST1- pin). A software reset (CRU bit RST2- active; see Table 4-2, note 2) does not disable the clock.

4.4.3 Use as Interval Timer

To use the TMS 9901 clock as an interval timer, initialize the clock with a known value, such as $3FFF_{16}$ (fully loaded clock register), then exit the clock mode. Later, when a interval value is wanted, re-enter the clock mode and read the clock read register. When the clock mode was re-entered, the register contents were latched into the clock read register; these contents are not updated again until the clock mode is exited and then re-entered. Thus, do not re-enter the clock mode until immediately before reading the clock contents. Subtract the new value from the initialized value to determine the counts elapsed (applies if a countdown to zero did not occur in which case the decrementer would have been re-initialized to initial clock register value and the countdown restarted).

There are two methods to exit the clock mode:

- write a zero to the control bit of the TMS 9901 (bit 0)
- address the LED (i.e., read or write to a bit outside the first 16 CRU bits of the TMS 9901)

In summary, to obtain the current clock register value, exit the clock mode; then, immediately before wanting to read the current clock register value, re-enter the clock mode and read the read register. If the clock mode had not been exited, the read register will not have been updated to the latest count even though the countdown had continued.

4.4.4 Example Timer Program

Figure 4-3 represents the applying bits to the CRU to initialize the TMS 9901 timer. Figure 4-4 is an example program that uses the TMS 9901 to issue an interrupt every time the clock counts down. The interrupt service routine sends a message to a terminal every 15 seconds. The basic configuration consists of a TM 990/102 microcomputer with DRAM at least from $E000_{16}$ to $E520_{16}$ and for eight bytes beginning at the PC vector location for interrupt 3 PC. This program can be executed by using loading the object code shown in Figure 4-4's third column into the memory address in the second column. Then load $E000_{16}$ in the program counter and execute the program. These steps and the applicable monitor commands are discussed in the opening comments of the program.

```
0003                *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
0004                *     THIS PROGRAM CAUSES AN INTERRUPT THROUUGH INT3          *
0005   M            *     EVERY 15 SECONDS USING THE INTERVAL TIMER IN THE        *
0006   e            *     TMS 9901. THIS CODE MAY BE ENTERED THROUGH THE          *
0007   m            *     LINE-BY-LINE ASSEMBLER OR THROUGH DIRECT OBJECT         *
0008   o            *     CODE ( THIRD COLUMN ) ENTRY WITH THE "IM"               *
0009   r     0      *     MONITOR COMMAND. THE PROGRAM SHOULD BE ENTERED          *
0010   y     b      *     AT THE MEMORY LOCATION SHOWN IN THE SECOND              *
0011         j       *     COLUMN OF THE LISTING.                                  *
0012   A     e      *     THE PROGRAM MAY BE EXECUTED BY SETTING THE              *
0013   d     c      *     PC TO MEMORY LOCATION  E000 WITH THE "IR"               *
0014   d     t      *     MONITOR COMMAND. AND THEN USING THE EXECUTE             *
0015   r            *     COMMAND "EX" TO START THE PROGRAM.                      *
0016   e     C      *     AN "S" ANSWER TO "CONTINUE OR STOP (C/S)?"              *
0017   s     o      *     CAUSES EXIT TO THE MONITOR.                             *
0018   s     d      *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *  *
0019         e      *
0020                *  PROGRAM CALLING THE INTERRUPT
0021                *
0022 E000                      AORG >E000          BEGINNING LOCATION OF MAIN PGM
0023 E000 02E0                 LWPI >E500          DEFINE THE WORKSPACE ADDRESS
     E002 E500
0024 E004 C060                 MOV  @>000C,R1      R1 POINTS TO TOP OF INT3 WORKSPACE
     E006 000C
0025 E008 C0A0                 MOV  @>000E,R2      R2 POINTS TO INT3 PC
     E00A 000E
0026 E00C 0203                 LI   R3,VE          R3 POINTS TO THE VECTOR TABLE
     E00E E06E
0027                *
0028                *   SET UP INTERRUPT BRANCH VECTOR AREA
0029                *
0030 E010 CCB3  LP             MOV  *R3+,*R2+      MOVE VECTOR TABLE TO INT3 PC ADDR
0031 E012 0283                 CI   R3,M1          CK IF ALL OF TABLE TRANSFERRED
     E014 E076
0032 E016 16FC                 JNE  LP             IF NOT GO BACK
0033                *
0034 E018 020C  TP             LI   R12,>0100      TMS 9901 CRU BASE ADDRESS IN R12
     E01A 0100
0035 E01C 04D1                 CLR  *R1            CLEAR INTERRUPT 3 REGISTER 0
0036 E01E 1E00                 SBZ  0              SET 9901 TO INTERRUPT MODE
0037 E020 1D03                 SBO  3              ENABLE INTERRUPT 3 ON 9901
0038 E022 0300                 LIMI 3              ENABLE CPU FOR INT3
     E024 0003
0039 E026 0204                 LI   R4,3           R4 = CLOCK VALUE OF >0001
     E028 0003
0040 E02A 33C4                 LDCR R4,15          ENTER CLOCK MODE, ENABLE, ENTER
0041                *                              CLOCK VALUE OF >0001
0042 E02C 04C0                 CLR  R0             CLEAR THE INT3 FLAG
0043 E02E C000  WT             MOV  R0,R0          LOOP HERE, WAIT FOR THE INTERRUPT
0044 E030 13FE                 JEQ  WT             INTERRUPT ROUTINE WILL SET R0=FFFF
0045 E032 2FA0                 XOP  @M1,14         ISSUE USER PROMPT MESSAGE
     E034 E076
```

FIGURE 4-4. EXAMPLE CODE TO USE THE INTERVAL TIMER (SHEET 1 OF 2)

```
0046 E036 2EC5          XOP  R5,11         ECHO USER KEYBOARD ENTRY
0047 E038 0206          LI   R6,'S'        R6 = 0053
     E03A 0053
0048 E03C 06C6          SWPB R6            R6 = 5300 ..XOP LEFT JUSTIFIED
0049 E03E 8185          C    R5,R6         IF AN "S" GO TO MONITOR
0050 E040 16EB          JNE  TP            ELSE START ROUTINE OVER
0051 E042 0460          B    @>A0          GO TO MONITOR TOP
     E044 00A0
0052                *
0053                *   INTERRUPT SUBROUTINE "SB" ENTERED MAIN PROGRAM.
0054                *   WILL RETURN TO THE MAIN PROGRAM AFTER 15 SECONDS
0055                *
0056 E046 020C  SB      LI   R12,>0100     TMS 9901 CRU BASE ADDRESS IN R12
     E048 0100
0057 E04A 0201          LI   R1,>5B8F      R1=CLOCK COUNT 11,719 = 250 MSEC
     E04C 5B8F
0058 E04E 33C1          LDCR R1,15         START CLOCK BY SENDING COUNT
0059 E050 1E00  TS      SBZ  0            SET 9901 TO INTERRUPT MODE
0060 E052 1D03          SBO  3             CLEAR INTERRUPT 3 ON 9901
0061 E054 1D00          SBO  0             RETURN TO CLOCK MODE
0062                *
0063                *   POLL INTREQ ( BIT 15 ) WAITING FOR CLOCK TO FINISH
0064                *
0065 E056 1F0F  PO      TB   15            HAS CLOCK SENT INTERRUPT
0066 E058 16FE          JNE  PO            NO THEN LOOP HERE
0067 E05A 0580          INC  R0            INCREMENT TIMER COUNTER
0068 E05C 0280          CI   R0,60         COUNT=60 =  15 SECONDS ELAPSED
     E05E 003C
0069 E060 16F7          JNE  TS            NO THEN GO BACK FOR ANOTHER
0070 E062 1E03          SBZ  3             OTHERWISE DISABLE 9901 INTERRUPT 3
0071 E064 2FA0          XOP  @M2,14        PRINT TIME MESSAGE
     E066 E092
0072 E068 04CF          CLR  R15           CLEAR THE MAIN STATUS REG BEFORE
0073                *                      WE RETURN  DISABLE ALL INTERRUPTS
0074 E06A 071D          SETO *R13          SET MAIN REG 0 TO SHOW INT OCCURED
0075 E06C 0380          RTWP               AND THEN RETURN TO THE MAIN PGM
0076                *
0077                *    DATA VALUES AND MESSAGES
0078                *
0079 E06E 0300  VE      DATA >0300         LIMI INSTRUCTION OP CODE
0080 E070 0000          DATA >0000         LIMI MASK VALUE
0081 E072 0460          DATA >0460         BRANCH INSTRUCTION OP CODE
0082 E074 E046          DATA SB            INTERRUPT SUBROUTINE ADDRESS
0083 E076 0D0A  M1      DATA >0D0A         CARRIAGE RETURN LINE FEED
0084 E078   43          TEXT 'CONTINUE OR STOP (C/S)?'
0085 E090 0000          DATA 0
0086 E092 0D0A  M2      DATA >0D0A         CR,LF
0087 E094   31          TEXT '15 SECONDS HAVE ELAPSED..'
0088 E0AE 0000          DATA 0
0089                    END
```

FIGURE 4-4. EXAMPLE CODE TO USE THE INTERVAL TIMER (SHEET 2 OF 2)

4.5 USER-PROGRAMMABLE LED DS2

LED DS2 can be programmed by writing a one or a zero to bit 16 of the TMS 9901:

- Bit 16 a one turns on the LED
- Bit 16 a zero turns off the LED
- Executing a sofware RST2- by writing a zero to bit 15 while in the clock mode turns on the LED
- IORST- active turns on the LED

The latter is caused by a RSET instruction, or either a hardware RESET or RESTART (pins P1-94 or P1-93 held to system ground).

CAUTION

Do not write to bits 17 to 31 of the TMS 9901. These bits were intended to be I/O ports with nine of them sharing external pins with interrupts INT7- to INT15-. If written to, they become programmed in the output mode, and the TMS 9901 can be damaged if one of these pins is written to while in the output mode (e.g., an interrupt input goes high at one of the shared pins). These shared CRU bits are bits 23 to 31 of the TMS 9901 which share external pins with INT15- to INT7- respectively. An exception is the clearing of parity on the TM 990/203 by outputting INT15.B-. However, this can be avoided by clearing parity through the CRU.

4.6 SOFTWARE RESET RST2

A software RST2- does the following:

- Turns on LED DS2.
- Resets bits 16 to 31 to the input mode (see CAUTION above).

A software RST2- is caused by entering the clock mode (bit 0 = one), then writing a zero to bit 15.

SECTION 5

PROGRAMMING THE TMS 9902 EIA PORT CONTROLLER

5.1 GENERAL

Connector P2, the EIA port of the TM 990/102 is controlled by a TMS 9902 asynchronous communications controller. TMS 9902 connections are shown in Figure 5-1. It can be programmed for the following:

- Read characters coming in to the TMS 9902 from connector P2.
- Transmit characters via the TMS 9902 and connector P2.
- Set baud rates, parity, character length, and other transmission specifications at the TMS 9902.
- Be an interval timer, interrupt driven or polled.

5.2 CRU STRUCTURE

Programming is through the Communications Register Unit (CRU) which is the serial input and output port on the TMS 9900 microprocessor. CRU instructions and addressing are explained in detail in section 7.7 (CRU Instructions).

NOTE

If not familiar with the CRU, read section 7.7 before proceeding.

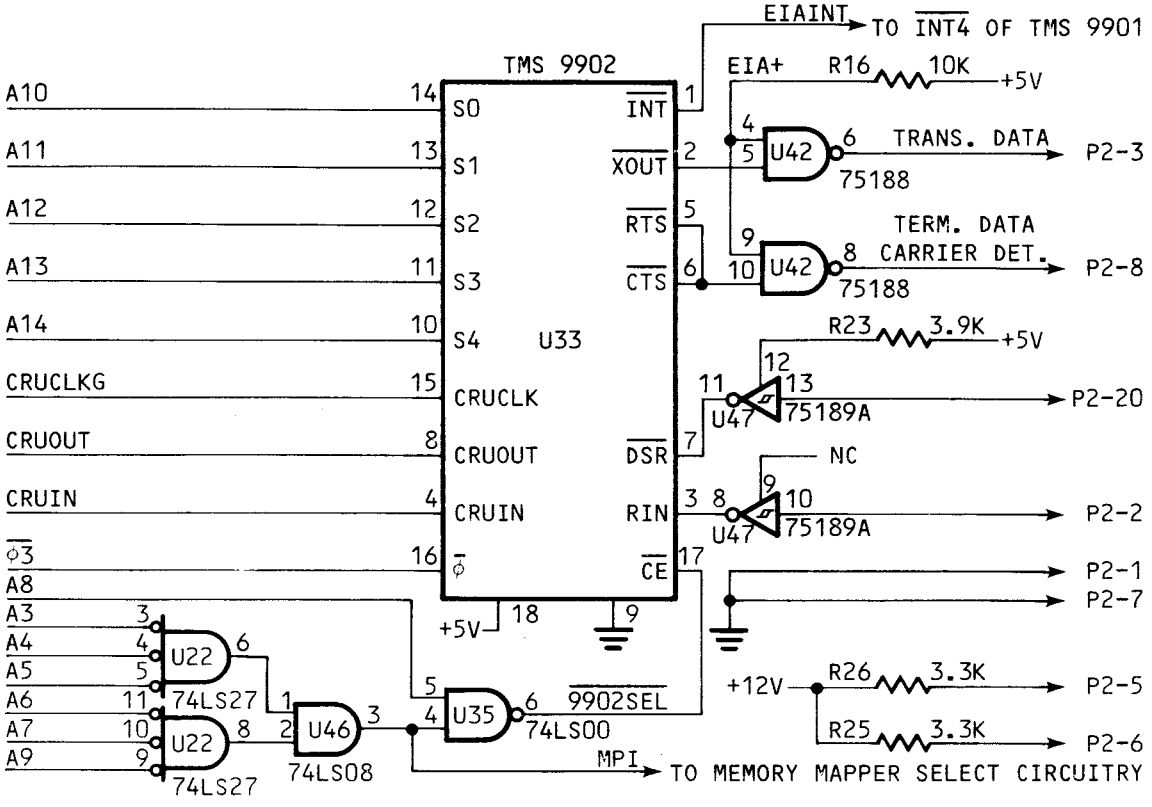
In general, there are five CRU instructions, three single bit (SBZ, SBO, and TB) and two multibit (LDCR and STCR) CRU instructions. When executed, CRU instructions place an address on the address lines and then either (1) send one (or more) bits over the CRUOUT line of the microprocessor or (2) receive one (or more) bits at the CRUIN line or (3) test the binary value at the CRUIN line. Table 5-1 lists CRU addresses for the board. Table 5-2 lists CRU addresses for the TMS 9902.

In CRU programming, the desired address is first placed in register 12 before CRU instruction execution; an address-line decoder (shown in Figure 6-15 in the Theory of Operation section) then enables the TMS 9902 so that it can be accessed through the CRU input (CRUIN) or output (CRUOUT) lines. Note that the CRU bit address is the resulting value on address lines A3 to A14 of the TM 990/102. This address line value is derived by adding a signed displacement to the binary values in bits 3 to 14 of register 12 (bit 15 of register 12 is ignored), and then applying this sum to address lines A3 to A14 of the TM 990/102.

TABLE 5-1. CRU ADDRESSES FOR TM 990/102

	Hardware Base Address (Hex)		Software Base Address (Hex)	
	From	To	From	To
Reserved	0000	003F	0000	007E
TMS 9902	0040	005F	0080	00BE
Reserved	0060	007F	00C0	00FE
TMS 9901	0080	009F	0100	013E
User Defined	00A0	0FFF	0140	1FFE

Note that some existing software uses software base address 0180_{16} to $01BE_{16}$ for a second EIA port.



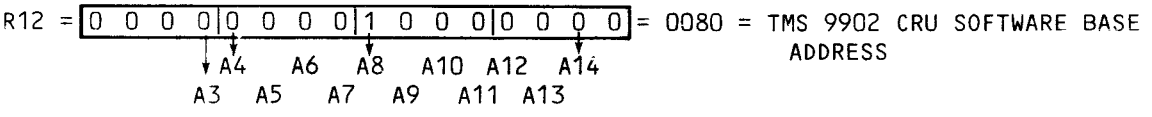


FIGURE 5-1. TMS 9902 ASYNCHRONOUS COMMUNICATION CONTROLLER

TABLE 5-2. CRU ADDRESSING FOR TMS 9902

Values on Address Lines (Hex) (Note 1) A3 to A14	Displacement (Base 10)	Device and Device Signals			
		Input Description (Read)		Output Description (Write)	
40	00	RBR0	Rec buffer data	REG0	Register data bit 0
41	01	RBR1		REG1	
42	02	RBR2		REG2	
43	03	RBR3		REG3	
44	04	RBR4		REG4	
45	05	RBR5		REG5	(Note 2)
46	06	RBR6		REG6	
47	07	RBR7	Rec buffer data	REG7	
48	08	0		REG8	
49	09	RCVERR	Receive error	REG9	
4A	10	RPER	Receive parity error	REG10	Register data bit 10
4B	11	ROVER	Receive overun error	LXDR	Load transmit data reg
4C	12	RFER	Receive framing error	LRDR	Load receive data reg
4D	13	RFBD	Receive full bit	LDIR	Load interval register
4E	14	RSBD	Receive start bit	LDCTRL	Load control register
4F	15	RIN	RIN pin status	TSTMD	Test mode
50	16	RBINT	Receive interrupt	RTSON	Request to send is on
51	17	XBINT	Transmit intrrupt	BRKON	Break on
52	18	0		RIENB	Receive interrupt enbl
53	19	TIMINT	Timer interrupt	XBIENB	Xmit interrupt enable
54	20	DSCINT	Data set change intrp	TIMENB	Timer interrupt enable
55	21	RBRL	Rcv buffer reg loaded	DSCENB	Data set chg intr enbl
56	22	XBRE	Trnsmt buffer empty		
57	23	XSRE	Trnsmt shft reg empty		
58	24	TIMERR	Timer error		
59	25	TIMELP	Timer elapsed		
5A	26	RTS	Request to send		
5B	27	DTR	Data term ready		
5C	28	CTS	Clear to send		
5D	29	DSCH	Data set status chng		
5E	30	FLAG	Register load/break		
5F	31	INT	Interrupt	RESET	Reset TMS 9902

NOTES

1. The values on address lines A10 to A14 are also the displacement in hexadecimal from the CRU hardware base address.
2. The TMS 9902 Registers, loaded via data bits 1 - 10, include the following internal registers:
   - Control Register (character length, timer time base, parity, and number of stop bits),
   - Interval Register (interval timer setting),
   - Receive Data Rate Register (set receive baud rate), and
   - Transmit Data Rate Register (set baud rate if different from that in the Receive Rate Register).

5.3 PROGRAMMING EXAMPLES

NOTE

In this subsection, mention is made of the registers and CRU bit functions of the TMS 9902. It is presumed that the reader is familiar with the TMS 9902; if not, the TMS 9902 data book is provided with this board for reference. Other programming examples can be found in the data book.

In the coding examples in this section, label TM9902 represents the software base address (R12 contents of 0080_{16}) for the TMS 9902.

5.3.1 Initialize TMS 9902 (Figures 5-2 and 5-3)

TMS 9902 initialization requires setting one or more internal registers:

- Control Register: Number of stop bits, parity, timer time base, and character length.
- Interval Register: Length of timer countdown to zero.
- Receive Data Rate Register: Receive baud rate.
- Transmit Data Rate Register: Transmit baud rate.

As explained in the TMS 9902 data manual, these registers are loaded by writing to bits 0-7 or 0-10 at the TMS 9902 CRU base address. The first part of the register-load operation will be to enable the register desired; this is done by setting CRU bits 11 to 14 as shown in Table 5-3, then writing the desired value to bits 0-7 or 0-10 as shown in the table. Note that these register-enable bits are set to all ones by a RESET; after loading, the register-enable bit is reset to zero. This allows setting the registers in the order shown in the table, top to bottom (e.g., after loading the control register, bit 14 is automatically set to zero; this sets up for loading the next register, etc.). The registers occupy CRU bits 0-7 or bits 0-10, depending on their size. (Figure 5-3 is an example of a RESET followed by four LDCRs which load the four registers; also see 3.1 in the TMS 9902 data manual.

CAUTION

After writing a one to bit 31 for a RESET, do not read or write to the TMS 9902 for 11 clock cycles.

TABLE 5-3. CRU-BIT LOGIC TO ENABLE TMS 9902 REGISTERS

Register(s) to be Loaded	CRU Bit Setting (Bit number is the displacement from CRU Base Address)				
	Bit 14	Bit 13	Bit 12	Bit 11	Register Occupies Bits
Control Register	1	X	X	X	0 to 7
Interval Register	0	1	X	X	0 to 7
Both Receive and Transmit D. R. Regs*	0	0	1	1	0 to 10
Receive Data Rate Register only	0	0	1	0	0 to 10
Transmit Data Rate Register only	0	0	0	1	0 to 10

*Loading the Receive and Transmit Data Rate Registers at the same time loads both with the same baud rate (this can save a programming step).
X = don't care

NOTE

After a register is loaded, its enabling bit is automatically reset to zero. This feature allows programming consecutive registers, beginning with the Control Register, without having to reset the enabling bit of the just-loaded register to obtain the logic patterns shown in Table 5-3 (i.e., load the Control Register, then the Interval Register, then the Data Rate Registers in this order for programming ease).

Figure 5-2 shows example code to load the Control Register. Figure 5-3 shows example code to load all four registers. The enabling bits for these registers will be set to one upon a RESET (CRU bit 31 set to one); this allows writing to the register using the logic shown in Table 5-3 and used in Figure 5-3.

Control Register. Stop bits, parity, clock division, and character length are set by writing to the TMS 9902 Control Register. See Figures 5-2 and 5-3.

The clock divide value allows choosing to divide the external clock input at pin 16 by a value of 3 or 4 using the following:

- CLK4M a one means divide by 4 (f_{int} = 1 MHz @ 4 MHz clock)
- CLK4M a zero means divide by 3 (f_{int} = 1 MHz @ 3 MHz clock) - This board uses a 3 MHz clock.

Interval Register. The Interval Register contains the binary value to be counted down for timing. When the countdown reaches zero, an interrupt is issued if enabled by writing a one to CRU bit 20 (TIMENB). If f_{int} = 1 MHz (see Control Register), each count lasts 64 usec; i.e., 64 x ($1/f_{int}$). An example of timer programming is shown in section 5.3.4. Also see Figure 5-3.

Receive Data Rate Register. This register is set to the desired baud rate constant of the incoming data. For the Receive Data Rate and Transmit Data Rate Registers, CRU bits 0 to 10 are used to load the register value. If f_{int} = 1 MHz (see Control Register) then the baud rate is equal to 1 Mbps/(register value x 2) if bit 10 is a zero, or 1 Mbps/(register value x 2 x 8) if bit 10 is a one. Figure 5-3(c) depicts loading the Data Rate Register for 110 baud.

Transmit Data Rate Register. This register is set to the desired baud rate constant of the data to be transmitted (see Receive Data Rate Register above).

5.3.2 Receive Character Through TMS 9902 (Figure 5-4)

A high-to-low transition on external line RIN (pin 3) activates the receiver circuitry of the TMS 9902. Figure 5-4 shows code to poll for a character received. In the third line of code in Figure 5-4, bit 21 (RBRL) is polled to see that the Receive Buffer Register is loaded with a full character via the TMS 9902 RIN line. Writing to bit 18 (RIENB) resets bit 21 so that it can be checked again for the next character received. Note that after resetting the TMS 9902 via the RESET bit (31), no input operation should occur for 11 clock cycles.

Figure 5-4 shows checking for character received by polling bit 21; however, if enabled, receipt of a character can cause an interrupt to INT4- of the TMS 9901. To cause this interrupt, set bit 16 (RBINT) to a one. Of course, the TMS 9901 and TMS 9900 must also be programmed to answer the interrupt (interrupts at the TMS 9901 are further covered in Section 4)

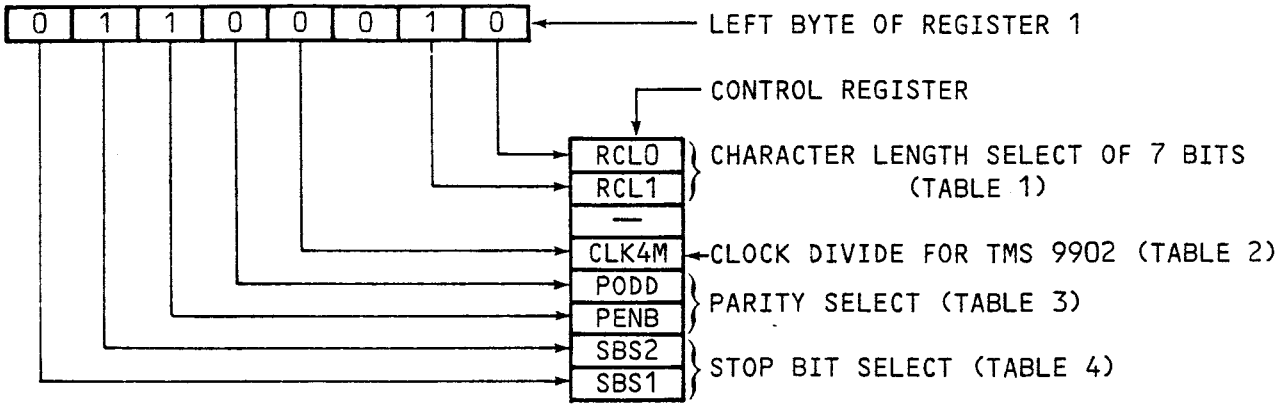


TABLE 1 CHARACTER LENGTH SELECTION

RCL1 BIT 1	RCL0 BIT 0	CHARACTER LENGTH
0	0	5 BITS
0	1	6 BITS
1	0	7 BITS
1	1	8 BITS

TABLE 2 CLOCK PERIOD DIVIDER

CLK4M	INTERNAL CLOCK PERIOD (t_{int}) AT 3 MHz CLOCK
0	3 MHz/3 = 1 μs
1	3 MHz/4 = 0.75μs

TABLE 3 PARITY SELECTION

PENB BIT 5	PODD BIT 4	PARITY
0	0	NONE
0	1	NONE
1	0	EVEN
1	1	ODD

TABLE 4 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

```
LI    R1,>6200      LEFT BYTE CONTAINS CONTROL BITS
LI    R12,CRUADR    SET PORT CRU SOFTWARE BASE ADDRESS (0080₁₆)
SBO   31            RESET TMS 9902
LDCR  R1,8          APPLY BITS TO CONTROL REGISTER
```

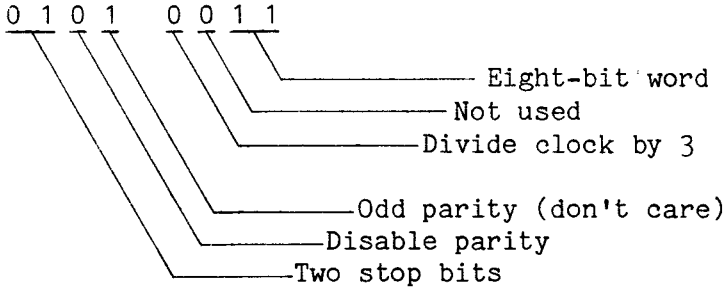
FIGURE 5-2. LOADING THE TMS 9902 CONTROL REGISTER

```
        LI    R12,TM9902    CRU BASE ADDRESS IN R12 (>0080)
*RESET THE TMS 9902 (FOLLOW WITH NO-OP FOR STABILIZATION)
        SBO   31            RESET TMS 9902, ALL REG. LOAD BITS = 1
        NOP                 NO-OP TO USE 11 CYCLES OF 9902 CLOCK
*LOAD THE CONTROL REGISTER (REG. CONTENTS EXPLAINED BELOW)
        LI    R1,>5300      SET UP CONTROL REGISTER CONTENTS
        LDCR  R1,8          APPLY TO CONTROL REGISTER
** THE LOAD CONTROL REGISTER BIT AUTOMATICALLY RESET TO ZERO
*LOAD THE INTERVAL REGISTER
        LI    R1,>9C00      >9C YIELDS 9.984 MILLISECONDS
        LDCR  R1,8          APPLY TO INTERVAL REGISTER
** THE INTERVAL REGISTER BIT WAS AUTOMATICALLY RESET TO ZERO
*LOAD THE RECEIVE DATA RATE REGISTER (REG. EXPLAINED BELOW)
        LI    R1,>638       >0638 YIELDS 110.04 BITS PER SECOND
        LDCR  R1,11         APPLY TO RECEIVE DATA RATE REGISTER
** THE RCV DATA RATE REGISTER BIT WAS AUTOMATICALLY RESET TO ZERO
*LOAD THE TRANSMIT DATA RATE REGISTER
        LI    R1,>01A1      >01A1 YIELDS 1200 BITS PER SECOND
        LDCR  R1,11         APPLY TO TRANSMIT DATA RATE REGISTER
** ELEVENTH BIT CLEARS TRANSMIT DATA RATE BIT, READIES FOR
** LOADING THE TRANSMIT BUFFER REGISTER WITH A CHARACTER
```

NOTE: If the same baud rate is needed for transmit as well as receive, delete the Load Receive Data Rate Register code above; then the Load Transmit Rate Register code also loads the Receive Data Rate Register as both CRU bits 11 and 12 are ones.

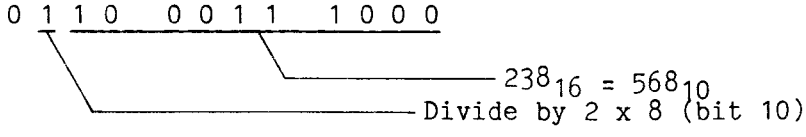
(a) Code to Initialize the TMS 9902

```
LI      R1,>5300
LDCR    R1,8
```



(b) Load Control Register Code

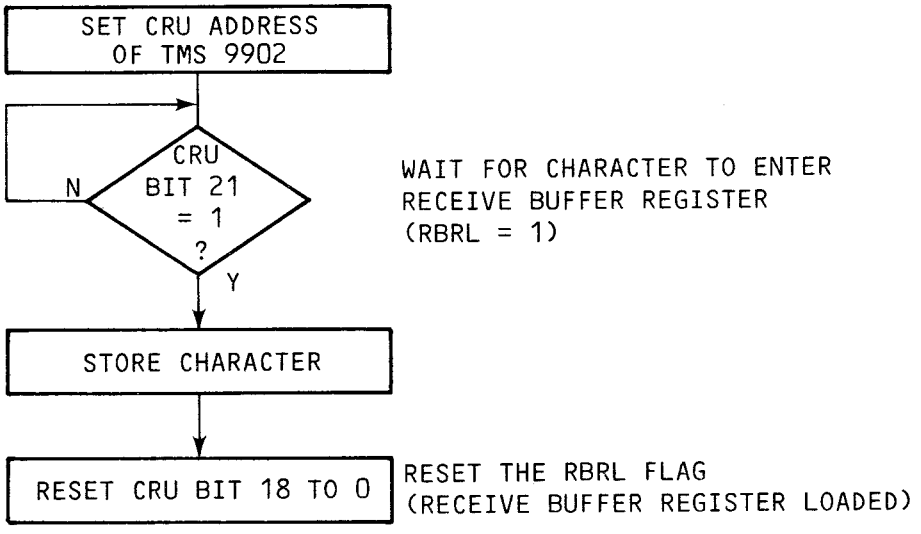
```
LI    R1,>638
LDCR  R1,11
```



1 M/(568 x 2 x 8) = 110.04 bps baud rate

(c) Load Receive Data Rate Register Code

FIGURE 5-3. LOADING THE FOUR REGISTERS ON THE TMS 9902




```
        LI    R12,TM9902    TMS 9902 CRU ADDRESS IN R12 (>0080)
        LI    R1,STORE      MEMORY ADDRESS TO STORE CHARACTERS
        .
        .
        .
LOOP    TB    21            HAS CHARACTER BEEN RECEIVED (RBRL=1?)
        JNE   LOOP          NO, LOOP TO WAIT FOR CHARACTER
        STCR  R2,8          BRING 8-BIT CHARACTER TO R2
        CI    R2,>0400      WAS CHARACTER AN EOT?
        JEQ   EXIT          YES, EXIT
        MOVB  R2,*R1+       STORE CHARACTER IN MEMORY
        SBZ   18            RESET RBRL FLAG (OUTPUT BIT 21)
        JMP   LOOP          GET NEXT CHARACTER
EXIT
        .
        .
        .
```

FIGURE 5-4. RECEIVE CHARACTER(S) BY POLLING

5.3.3 Transmit Character(s) Through TMS 9902 (Figure 5-5)

Writing a one to bit 16 (RTSON) enables the external RTS- (request to send) line (pin 5) low which is interpreted by the peripheral device. When the external CTS- (clear to send) line (pin 6) goes low, the transmitter is enabled. In a polled operation, bit 22 (XBRE) is tested to verify that the transmit buffer register is presently empty. Then the character is written to the TMS 9902 via the CRUOUT line of the host (to receive characters, the Transmit Buffer Register must be active; i.e., the other four TMS 9902 registers -- Control, Interval, and both Data Rate Registers -- must be disabled along with BREAKON also a zero). Bit 22 (XBRE) is then checked for a one to see that the character has exited the TMS 9902 Receive Buffer Register; this bit is reset when a character is sent to the transmit buffer register. Then bit 23 (XSRE) is checked for a one to see that the character has exited the Transmit Buffer Register so that (if required) a timing loop can be entered for a particular baud rate should a delay be needed (e.g., for a carriage return). Characters are transmitted from the TMS 9902 via its XOUT pin (pin 2). Note that after resetting the TMS 9902 via the RESET bit (31), no output operations should occur for 11 clock cycles.

5.3.4 Programming the Inverval Timer (Figure 5-6)

A flow chart for execution of the interval timer is shown in Figure 5-6. Numbers next to the various steps show user programming interaction with the TMS 9902; unnumbered steps are internal to the TMS 9902.

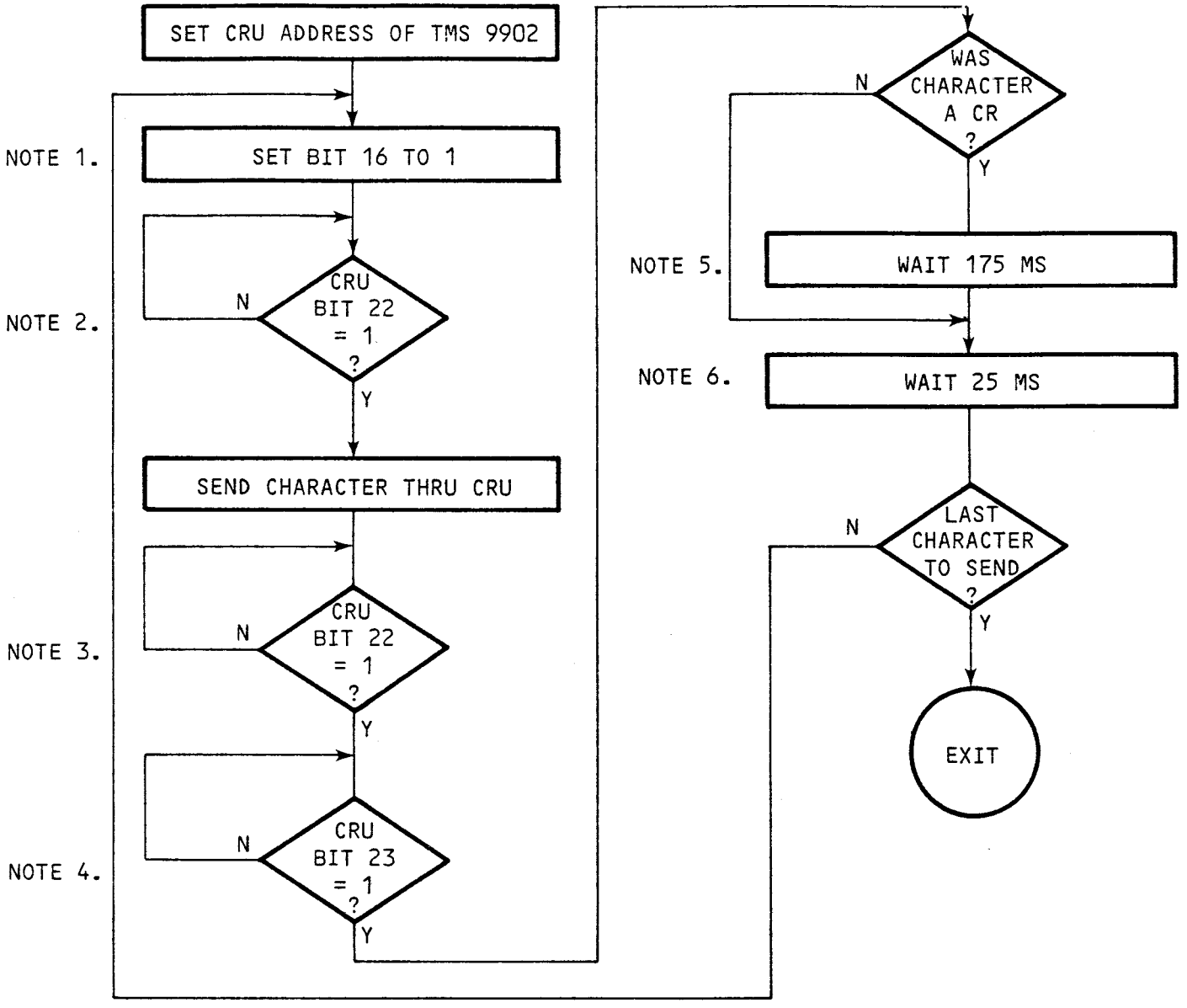
The code below sets up the TMS 9901 to receive an interrupt from the TMS 9902 and also sets up the TMS 9902 as follows:

- Reset the TMS 9902 which causes
  - TIMELP flag to be reset (timer elapsed, bit 25)
  - TIMERR flag to be reset (timer error, bit 24)
  - LDCTRL and LDIR to be set to ones (bits 14 and 13)
- Sets up interval value for 15 milliseconds in R2 (using 3 MHz clock)
- Sets TIMENB at bit 20 which allows an interrupt issued at pin INT- of the TMS 9902 when interval register counts down to zero.
- Load interval register which starts the countdown.

```
*RESET TMS 9902, SET COUNTER, AND ENABLE ISSUANCE OF INTERRUPT
*NOTE: INTERNAL CLOCK PERIOD SET AT CONTROL REGISTER FOR 1 US (3 MHZ CLOCK)
       LI    R12,TM9902        PLACE TMS 9902 SFTWR BASE ADDR IN R12 (>0080)
       SBO   31               RESET (SETS LDCTRL, LDIR; RESETS TIMELP, TIMERR)
       LI    R2,>EA00          15-MILLISECOND TIMER VALUE IN R2
       SBZ   14                RESET LDCTRL: SET UP FOR LOADING INTERVAL REG.
       SBO   20                SET TIMENB (ALLOWS INT- TO TMS 9901)
       LDCR  R2,8             8 MSB'S OF R2 TO INTERVAL REG.: START COUNTDOWN
*ENABLE INT4 AT TMS 9901 (INPUT OF SIGNAL EIAINT- FROM INT- OF TMS 9902)
       LI    R12,TM9901        PLACE TMS 9901 SFTWR BASE ADDR IN R12
       SBZ   0                 ENTER INTERRUPT MODE
       SBO   4                 ENABLE INTERRUPT 4
*ENABLE INTERRUPT 4 (OR HIGHER PRIORITY) AT TMS 9900
       LIMI  4                 ENABLE INTERRUPTS 1 TO 4 AT TMS 9900
```

When the countdown occurs, INT- from the TMS 9902 becomes active (low) to INT4- of the TMS 9901. Enabled at the TMS 9901 and TMS 9900, interrupt 4

vectors (at memory address >10, >12) are loaded into the WP and PC registers and a trap occurs to an interrupt service routine. Included in this routine should be code to disable the interrupt at the TMS 9901 and disable the timer at the TMS 9902.



NOTES:

1. ACTIVATE $\overline{\text{RTS}}$ (LOW)
2. WAIT FOR TMS 9902 TRANSMIT BUFFER REGISTER TO BE EMPTY (XBRE = 1 ?)
3. WAIT FOR TMS 9902 TRANSMIT BUFFER REGISTER TO BE EMPTY (XBRE = 1)
4. WAIT FOR TMS 9902 TRANSMIT SHIFT REGISTER TO BE EMPTY (XSRE = 1)
5. A CARRIAGE RETURN ON A 733 ASR/KSR REQUIRES 200 MS DELAY FOR PRINTHEAD TRAVEL
6. A CHARACTER ON A 733 ASR/KSR REQUIRES 25 MS DELAY FOR 300 BAUD OPERATION

(a) Flow Chart

FIGURE 5-5. TRANSMIT CHARACTER(S) BY POLLING (SHEET 1 OF 2)

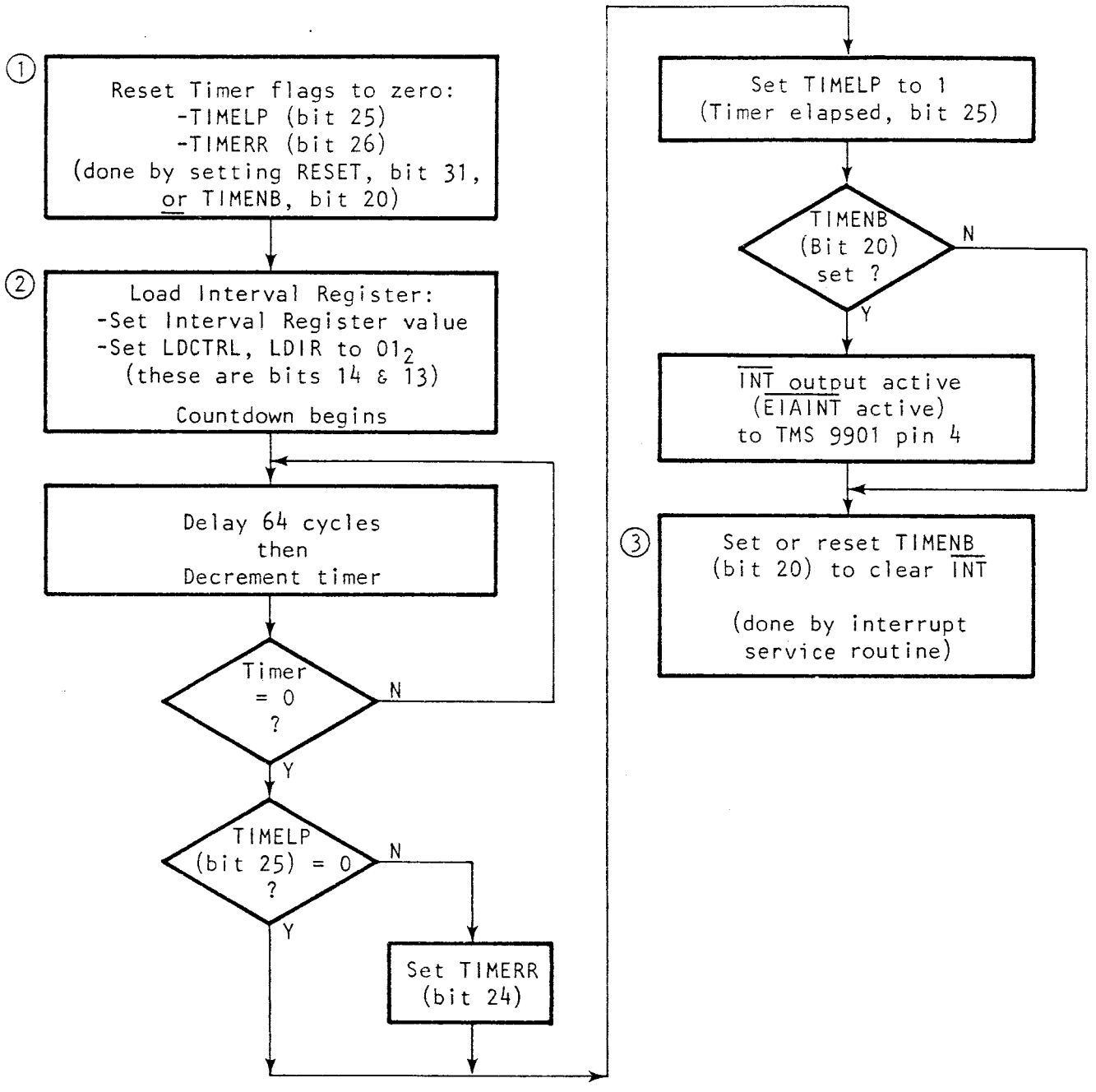
```
*INITIALIZE REGISTERS
        CLR   R2             INITIALIZE TEMPORARY STORAGE
        LI    R12,>0080      CRU SOFTWARE BASE ADDRESS
        LI    R1,CHARS       STORAGE FOR CHARACTERS TO BE SENT
        .
        .
        .

LOOP6   LI    R3,26253       LOOP COUNT FOR 175 MILLISECOND DELAY
        LI    R4,3750        LOOP COUNT FOR  25 MILLISECOND DELAY
        SBO   16             SET RTS TO ONE
LOOP1   TB    22             IS TRANSMIT BUFFER EMPTY?
        JNE   LOOP1          NO, WAIT UNTIL EMPTY
        LDCR  *R1,8          CHARACTER TO TMS 9902 VIA CRUOUT
LOOP2   TB    22             IS TRANSMIT BUFFER EMPTY?
        JNE   LOOP2          NO, WAIT UNTIL CHARACTER MOVED
LOOP3   TB    23             IS TRANSMIT SHIFT REGISTER EMPTY?
        JNE   LOOP3          NO, WAIT UNTIL CHARACTER SENT
        MOVB  *R1+,R2        MOVE CHARACTER TO R2
        CI    R2,>0D00       WAS IT A CARRIAGE RETURN?
        JNE   LOOP5          NO, SKIP EXTRA 175 MS DELAY
LOOP4   DEC   R3             DECREMENT COUNTER FOR 175 MS
        JNE   LOOP4          LOOP FOR 175 MS
LOOP5   DEC   R4             DECREMENT COUNTER FOR 25 MS
        JNE   LOOP5          LOOP FOR 25 MS
        CI    R2,0           WAS THAT LAST CHARACTER = 00?
        JNE   LOOP6          NO, GET NEXT CHARACTER
         .
         .
         .
```

NOTE

Above values are accurate for a 3 MHz system clock

(b) Assembly Language Coding

FIGURE 5-5. TRANSMIT CHARACTER(S) BY POLLING (SHEET 2 OF 2)



NOTE: Flowchart symbols with circled numbers are user program inputs; others are internal to the TMS 9902.

FIGURE 5-6. LOADING AND EXECUTING THE INTERVAL TIMER USING INTERRUPTS

If the timer is not disabled at the TMS 9902, its interval register is reloade and the countdown starts over again. If it counts down to zero again and TIMELP has not been reset, another interrupt will be issued; however, TIMERR will also be active. TIMERR means an interrupt occurred while TIMELP was also active (was not reset after becoming active from a previous countdown).

5.3.5 Example Interrupt Operation (Figure 5-7)

Figure 5-7 uses interrupts to receive and store characters sent to the TMS 9902. Although the TMS 9902 can be polled for character receipt, interrupt operation allows the processor to do other chores in between receipts of characters. Several factors must be considered in interrupt operations:

- Interrupt level must be enabled at microcomputer board's TMS 9901 and in the TMS 9900's interrupt mask. In this example (shown in Figure 5-7), interrupt level 4 is enabled at both places.

- Vectors must be present in lower memory for the interrupts used. Since the example in Figure 5-7 uses interrupt level 4 (this is the TMS 9902 interrupt, tied to the INT4- pin of the TMS 9901), WP and PC vectors must be already loaded in memory addresses (hex) 0010 and 0012 in order to branch to the interrupt service routine.

- Four events can cause interrupts to be issued by the TMS 9902 as specified in the TMS 9902 data manual. These are:

  1) Receipt of a character (Receive Buffer Full).
  2) Character is ready to be transmitted (character shifted from Transmit Buffer Register to Transmit Shift Register).
  3) Interval Timer counts down to zero.
  4) DSR- or CTS- inputs change (data set change).

  In the example in Figure 5-7, event number (1) is used to cause an interrupt upon receipt of a character at RIN of the TMS 9902. In this case, the receiver-interrupt enable bit must be set at the TMS 9902 (bit 18).

- The interrupt service routine should disable the interrupt at the TMS 9901 to prevent it from being continuously requested.

- Before the interrupt service routine is exited, it should restore the interrupt conditions desired. For example, if it is desired that the next character received should cause an interrupt:

  - at the TMS 9901, re-enable the interrupt (INT4-),
  - at the TMS 9900, return the interrupt mask at the processor to a level four (it had been set to three when the interrupt request was answered) by executing an LIMI 4,
  - at the TMS 9902, turn off RBRL (Receive Buffer Register loaded) and re-enable the receiver interrupt; both can be done by writing a one to bit 18 (REINB).

As described in the TMS 9902 data manual and shown in Figure 5-7, the receiver interrupt is enabled by writing a one to bit 18 (RIENB) on the TMS 9902. When enabled, an interrupt will be issued from the TMS 9902 (INT- low) to the INT4- input on the TMS 9901. Before the interrupt can be serviced, INT4- must have been enabled at both the TMS 9901 and TMS 9900.

```
*SET UP FOR INTERRUPT IN MAIN BODY OF PROGRAM
**CHARACTER STORAGE AREA
COUNTR DATA 0              CHARACTER STORAGE COUNT ACCUMULATOR
STORE  BSS  200            RESERVE 200 BYTES
       .
       .
       .

**ROUTINE TO ENABLE INTERRUPT 4 AT TMS 9901
       LI   R12,>0100      ADDRESS TMS 9901 ON MICROCOMPUTER BOARD
       SBZ  0              ENTER INTERRUPT MODE
       SBO  4              ENABLE INTERRUPT 4 AT TMS 9901
       LIMI 4              ENABLE INTERRUPT 4 AT TMS 9900
**ROUTINE TO ENABLE THE ISSUING OF AN INTERRUPT FROM TMS 9902
**WHEN A CHARACTER IS RECEIVED AT THE TMS 9902
       LI   R12,>0080      ADDRESS TMS 9902
       SBO  18             ENABLE RECEIVER INTERRUPT ENABLE BIT (BIT 18)
**ROUTINE TO PLACE STORAGE-AREA ADDRESS IN R1 OF INTERRUPT SERVICE ROUTINE
       MOV  @>0010,R1      PLACE ADDRESS OF INTERRUPT WP START IN R1
       LI   R2,STORE       PLACE ADDRESS OF STORAGE START IN MAIN PROGM R2
       MOV  R2,*R1         MOVE STORAGE START TO R0 OF INTERRUPT SERV. RTN.
       CLR  @COUNTR        CLEAR CHARACTER COUNTER
```

(1) Set Up Main Program for Interrupt

FIGURE 5-7. EXAMPLE PROGRAM USING INTERRUPTS TO RECEIVE CHARACTERS (Sheet 1 of 2, see next page)

Figure 5-7 contains example code to use interrupts to receive characters at the EIA port. This example shows the major considerations of using interrupts and such routines can be expanded as needed by the user. This example is not intended as a complete example of an interrupt service routine (e.g., not shown are full error checking routines, a string termination character check, etc.).

Sheet 1 of the figure contains code to enable interrupt level 4 at the TMS 9901 and TMS 9900 on the microcomputer board. A counter used by the interrupt service routine is also set to zero.

```
*INTERRUPT SERVICE ROUTINE
**TURN OFF INTERRUPT AT TMS 9901
       LI   R12,>0100      ADDRESS BOARD TMS 9901
       SBZ  0              SET CONTROL BIT TO ZERO FOR INTERRUPT MODE
       SBZ  4              DISABLE INTERRUPT 4 (BIT 4)
**TURN OFF RECEIVER INTERRUPT AT TMS 9902
       LI   R12,>0080      ADDRESS BOARD TMS 9902
       SBZ  18             SET BIT 18 TO ZERO TO DISABLE RCVER INTERRUPT
**CHECK FOR ERRORS, IF NONE, RECEIVE AND STORE CHARACTER
       TB   21             IS CHARACTER IN RECEIVE BUFFER OF 9902?
       JNE  ERROR1         NO, CHARACTER NOT RECEIVED, CHECK FOR ERROR
       TB   9              YES; DID ERROR OCCUR IN CHARACTER RECEIPT?
       JEQ  ERROR2         YES, GO TO ERROR ROUTINE
       STCR R1,8           NO, CHARACTER RECEIVED OK: MOVE TO 8 MSB OF R1
       JMP  STOR           GO STORE CHARACTER RECEIVED IN PORT BUFFER AREA
**ERROR ROUTINES FOR CHARACTER RECEIPT
ERROR1
       .
       .
       .
ERROR2
       .
       .
       .

**STORE CHARACTER, KEEP TRACK OF CHARACTER COUNT
STOR   MOVB R1,*R0+        MOVE CHARACTER TO STORAGE AREA
       INC  @COUNTR        INCREMENT CHARACTER QTY COUNTER
**SET UP TO EXIT INTERRUPT SERVICE ROUTINE
***AT TMS 9901
       LI   R12,>0100      TM 9901 CRU BASE ADDRESS
       SBZ  0              ENTER INTERRUPT MODE
       SBO  4              RE-ENABLE INTERRUPT 4 (TMS 9902 INTERRUPT)
***AT TMS 9902
       LI   R12,>0080      TM 9902 CRU BASE ADDRESS
       SBO  18             BIT 18 TO ONE TO RE-ENABLE RECEIVER INTERRUPT
***AT TMS 9900
       LIMI 4              RE-ENABLE INTERRUPT 4
       RTWP
```

(2) Major Parts of Interrupt Service Routine

FIGURE 5-7. EXAMPLE PROGRAM USING INTERRUPTS TO RECEIVE CHARACTERS
(Sheet 2 of 2)

Sheet 2 of Figure 5-7 shows the interrupt service routine. Not shown are error checking routines. After receipt, the character is stored.

SECTION 6

THEORY OF OPERATION

6.1 GENERAL

This section presents the theory of operation of the TM 990/102 microcomputer. Information in the following manuals can be used to supplement material in this section:

- TMS 9900 Microprocessor Data Manual
- TMS 9902 Asynchronous Communications Controller Data Manual
- TTL Data Book, Second Edition (with 1981 Supplement)
- The Interface Circuits Data Book
- The MOS Memory Data Book.
- TMS 4500 Dynamic RAM Controller Data Sheet

Figure 6-1 shows a block diagram of the TM 990/102, highlighting the four major buses:

- Control bus
- Address bus
- Data bus
- Communications Register Unit (CRU) bus.

The major features of the TM 990/102 microcomputer board include the clock driver, the microprocessor, memory mapper, dynamic RAM (DRAM) controller, the TMS 9901 parallel I/O interface (interrupt input), the TMS 9902 asynchronous communication interface, the bidirectional and unidirectional backplane buffers, the EPROM, the DRAM, and the miscellaneous signals. These features are discussed in the following paragraphs of this section.

In normal operation the TMS 9900 microprocessor commands most of the control bus; the data bus and address buses are bidirectional, driven by both the microprocessor and offboard DMA devices. The two data-carrier lines on the CRU bus are not bidirectional: the serial output line is microprocessor driven and the serial input line is driven by the CRU device.

6.2 POWER SPECIFICATIONS

Approximate power values required by the TM 990/102-3 are listed in section 1.5 of this manual.

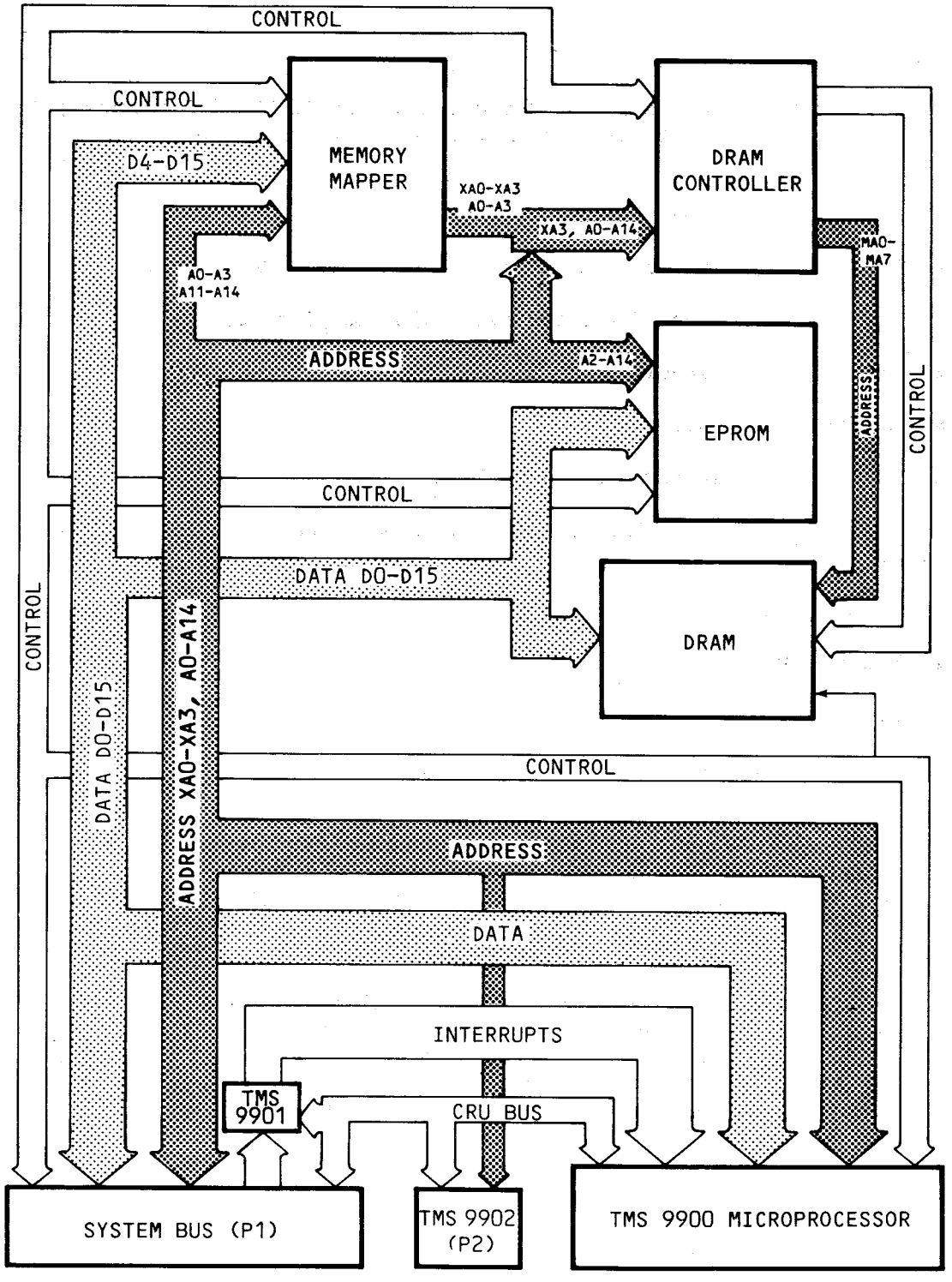


FIGURE 6-1. TM 990/102 BLOCK DIAGRAM

The -5 V supply is derived on the board by the MC79L05 regulator from the -12 V supplied from offboard. The -5 V supply is used primarily by the TMS 9900 microprocessor. The -12 V supply is used for the 75188 EIA line drivers as well as for supplying the voltage to the MC79L05 -5 V regulator.

The +12 V supply is used by the TMS 9900 microprocessor, the 75188 EIA line drivers, and the TMS 9904 clock generator.

All integrated circuits on the board, except the EIA line drivers, use the +5 V supply, and because of the heavy load, this voltage is not derived by an on-board regulator but must be supplied from off the board. The MOS parts use this supply for TTL compatibility, and, in fact, the TMS 9901, TMS 9902, and DRAM use only this voltage for supply since they contain internal charge pumps, eliminating the need for -5 or +12 V in their operation.

6.3 SYSTEM STRUCTURE

The block diagram in Figure 6-1 shows the system structure of the TM 990/102 microcomputer board. The microcomputer design centers around five buses: power, control, address, data, and CRU. The major blocks of the system are the processor, the miscellaneous control signals, address decoding, on-board memory, the TMS 9901 parallel I/O controller, TMS 9902 serial communication port, miscellaneous CRU devices, interrupt processing, and memory processing which includes the memory mapper (allows extended addressing) and dynamic RAM controller (monitors refresh cycles, row and column addressing, etc.).

Functionally, these major blocks represent the processing, memory and I/O portions of the microcomputer.

Throughout the remainder of this section, each block's function is discussed, grouping the explanations into three categories: processing, memory, and I/O. The first subject is the buses since the buses tie all the blocks together.

The power bus is explained in section 6.2; the following paragraph deals with the remaining buses.

6.4 SYSTEM BUSES, GENERAL

The four major buses are subdivided by function in Table 6-1. By referring to the schematics in Appendix A, each random logic line as well as the bus lines can all be traced. All bus signals appear on connector P1. Bus buffering is covered in section 6.12.

6.4.1 Address Bus

NOTE

The TMS 9900 addresses 64 K bytes; however, it uses a 15-line address bus (not 16 lines) consisting of lines A0 through A14. Each 0000_{16} to $FFFF_{16}$ memory access uses a 16-bit even-numbered address word; because the word's address is always even numbered, line A15, the byte-address bit, is not brought out of the TMS 9900. Byte operations are handled by fetching an entire 16-bit word into the processor, then modifying the addressed byte, then rewriting the 16-bit word back to memory if necessary. For this reason, the TM 990/102 board needs only 15 address lines to address up to 64K bytes and 19 address lines to address up to 1 M bytes.

TABLE 6-1. BUS SIGNALS

Signal	Functional Device Connections
Address Bus	
A0 to A14	Processor's address bus
XA0 to XA3	Extended addressing lines
Data Bus	
D0 to D7	Most significant byte
D8 to D15	Least significant byte
CRU Bus	
CRUIN	CRU input line, TMS 9901, TMS 9902
CRUOUT	CRU output line, TMS 9901, TMS 9902
CRUCLK	CRU clock, TMS 9901, TMS 9902, 74S138 external address decode
Control Bus	
MEMEN-	Memory control: board is in a memory cycle
DBIN	Memory control: memory map register read/write, data buffer direction
WE-	Memory control: memory map register strobe in read/write mode, DRAM read/write
MEMCYC-	Memory control: enable/disable data buffers, DRAM controller
READY	Memory control: control TMS 9900 during refresh
WAIT	Memory control: disable MEMCYC- when memory not ready
Auxiliary Control	
ø1-, ø3-	Clock: TMS 9901, TMS 9902, RESET logic, memory timing, DRAM controller,
INT1.B- to INT15.B-	Interrupt inputs: external, via TMS 9901 to TMS 9900
HOLD-, HOLDA, HOLDAI	CPU and buffer control for DMA: address, data, memory
LOAD-	Initialize CPU using vectors in upper memory
IAQ	Instruction being accesed from memory
RESET	Initialize CPU using vectors in lower memory

Maximum onboard dynamic RAM (DRAM) memory is 128 K bytes. To access above 64 K bytes, the memory mapper chip uses the four address lines A0 to A3 to select the most significant eight address bits (of 19) from one of the mapper's sixteen registers. These eight address bits are applied to memory address lines XA0 to A3 to evolve the extended address as shown in Figure 6-2 (a). Only XA3 is used along with A0 to A14 to select onboard memory; thus, onboard DRAM can be located with a maximum address of 128 K, ($1FFFE_{16}$). The memory mapper must first be loaded with desired register values; a general loading of the mapper is depicted in Figure 6-2 (b).

The other major address-line controller is the dynmamic RAM controller chip which controls address lines to the DRAMs as well as controls the refresh cycles for this memory. Maximum 128 K byte on board memory is provided on 16 TMS 4164 DRAMs, each organized on a 64 K x 1 format. The TMS 4500 DRAM controller chip receives a 16-address line (XA3 to A14) input which it decodes as an eightbit row address (RA7-RA0) and eight-bit column (CA7-CA0) address. The controller also provides for periodic refresh cycles during which memory access is not occurring.

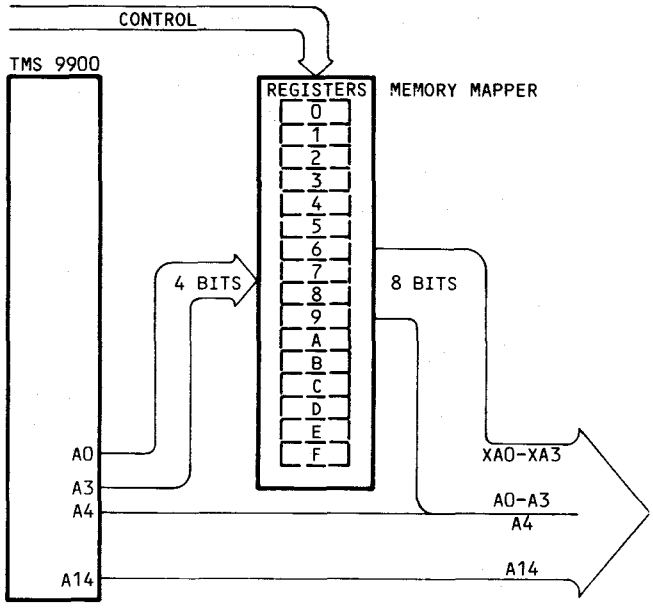
During a memory access, onboard-microprocessor address lines A0 to A3 are routed to the memory mapper chip, which can be programmed for a memory mapper-on mode (19 extended-address lines used) or a mapper-off mode (15 address lines used). In the mapper-on mode, mapper-chip registers containing the eight-bit extended addresses are selected by microprocessor address lines A0 to A3 resulting in an eight-bit value placed upon the eight memory address lines XA0 to A3. In the mapper-off mode, only A0 to A14 are enabled; XA0 to XA3 are all zeroes.

Address lines XA0 to XA3 go to comparator circuitry which decodes whether the addressed dynamic-RAM location is populated onboard in the lower 64 K block (0000_6 to $FFFF_{16}$) or the upper block of memory (10000_{16} to $1FFFF_{16}$). This applies to both DMA by the TM 990/102 as well as onboard memory access. The DRAM controller sets up the eight-bit row and column addresses for each enabled memory access.

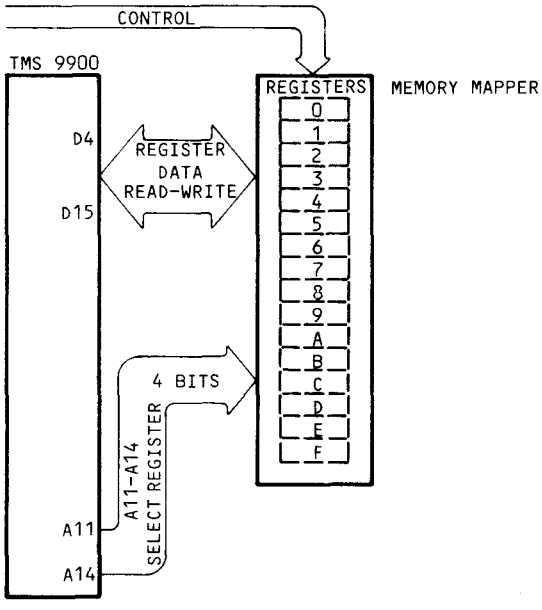
After determining that the DRAM is populated on board, address lines XA3 and A0 to A14 are decoded by the DRAM controller to set up the row and column address for each of the 16 DRAM chips. Each TMS 4164 chip is organized 1 by 64 K, providing 64 K words or 128 K bytes when populated; whereas TMS 4532 chips provide 32 K words or 64 K bytes when populated.

Access to EPROM is determined by a 8-bit comparator which monitors A0 to A3 and associates the lines used to the EPROM type selected by jumpers (TMS 2516s, TMS 2532s, or TMS 2564s can be used). EPROM always begins at address 0000_{16}; thus, logic selects the memory area jumpered as EPROM begining at 0000_{16}. If DRAM is also in this space, the EPROM will be laid over the DRAM with DRAM beginning immediately after the EPROM area. DRAM/EPROM selection is further discussed in section 6.9.5.

Lines A0, A1, and A2 are also routed to an external instruction decoder where, upon a CRUCLK pulse, the state of the address lines is decoded as to whether a CRU operation (A0, A1, A2 = all zeroes) or an external instruction (CKON, CKOF, RSET, IDLE, LREX) is occurring (A0, A1, A2 = greater than zero). External instruction CKON, CKOF, and RSET are used to control the memory mapper functions. External instructions are further covered in section 6.8



(a) Mapper-On Mode, Extended Address Operation



(b) Mapper-Register Access

FIGURE 6-2. BLOCK DIAGRAM USING MEMORY MAPPER

6.4.2 Data Bus

The data bus (D0 to D15) connects the TMS 9900 microprocessor data lines to onboard memory (DRAM and EPROM), to the memory mapper for reading/writing mapper register contents, and to 16 bidirectional data lines which are routed from the processor to bidirectional buffers for off-board use. D0 is the most significant bit, and D15 is the least significant bit. Logic provides selecting buffer direction for both onboard use as well as during offboard control (DMA).

6.4.3 CRU Bus

The two onboard CRU (Communications Register Unit) devices are the:

- TMS 9901 parallel interface used for interrupt monitoring and for controlling the user LED (DS2).
- TMS 9902 asynchronous serial data transmission controller used to monitor EIA communications protocol.

The TMS 9901 monitors incoming interrupts INT1.B- to INT.15B-. If the interrupt level is enabled at the TMS 9901 via the CRU, and no higher priority interrupt is active, the interrupt code is presented to the TMS 9900 IC0 to IC3 lines and the processor's INTREQ- line is brought low. This informs the processor of an incoming interrupt request.

The TMS 9902 asynchronous serial data transmission controller reads and writes to EIA devices connected to connector P2. Programming the TMS 9902 is covered in Section 5.

The three lines comprising the CRU bus are CRUIN, CRUOUT, and CRUCLK. Serial data is presented to the CRUIN line (read at the CPU) and written out at CRUOUT (sent by CPU) on a serial basis. CRUCLK is used for timing of CRUOUT operations and is not active in CRUIN (read) operations. Whenever address lines A0 to A2 are all zeroes and CRUCLK is active, a CRU operation is to be assumed; whereas a non-zero value on A0 to A2 during an active CRUCLK indicates external instruction execution. External instructions are further explained in Section 6.8.

Note that even if some CRU device writes to the CRUIN line while the bus changes value or the line is in some way invalid, no harm is done because the data presented to CRUIN by the addressed device will be ignored by the processor. Since the processor will poll CRUIN only when desired, CRU address decoding is simplified.

The CRU address is contained on address lines A3 to A14 with A0 to A2 used in external addressing and A15 not used on the TMS 9900 (byte access is handled by manipulating the left or right byte of a retrieved 16-bit word). The value to be placed on the address bus is first programmed into register 12 of the workspace registers. Only 12 bits of this register -- bits 3 to 14 -- are used to form the CRU address. This requires using the two terms:

- Software CRU base Address: Entire contents of register 12 as loaded in the program.
- Hardware CRU base Address: Contents on address lines A3 to A14 when a CRU instruction is being executed.

These terms are depicted in Figure 6-3. Programming the CRU devices is further explained in Sections 4 and 5.

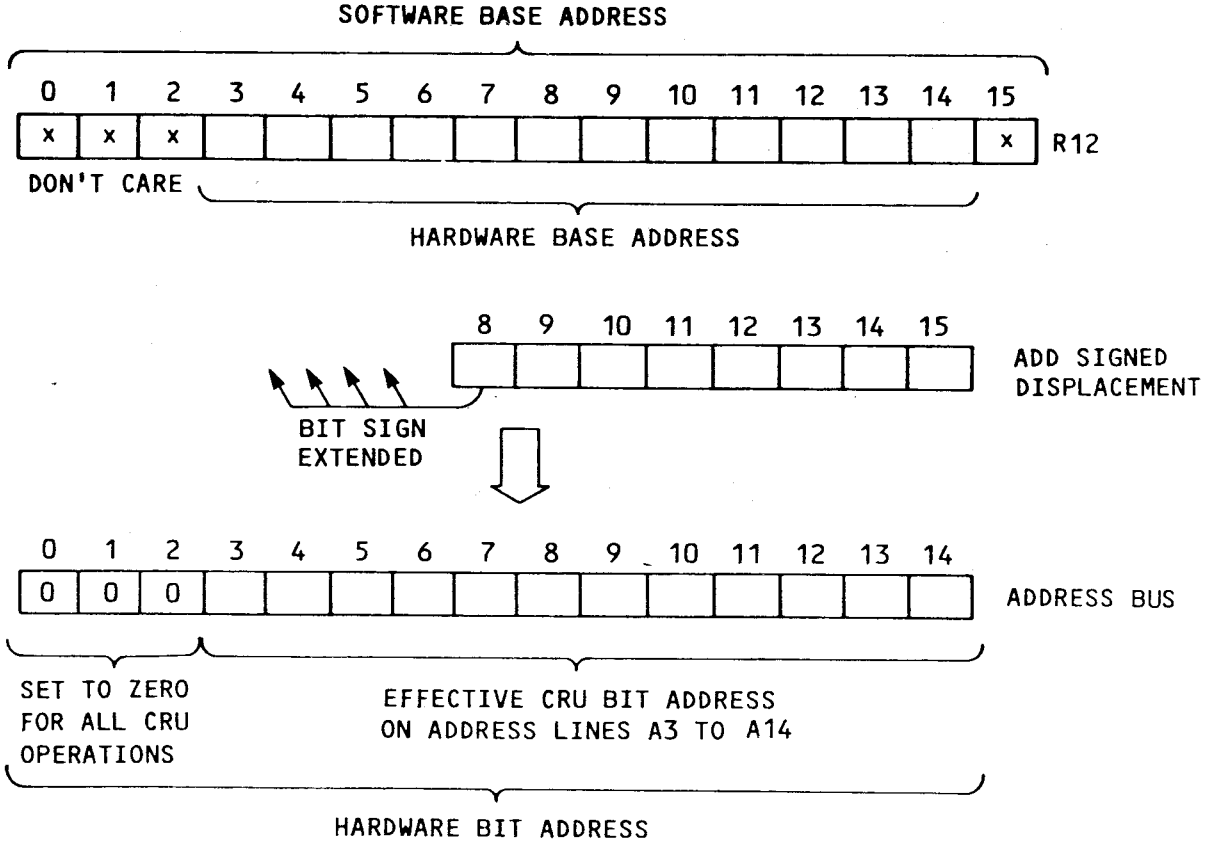


FIGURE 6-3. CRU ADDRESS EVOLUTION

6.4.4 Control Bus

This bus is not as homogenous as the other buses; therefore it is divided into groups as has been shown in Table 6-1. Table 6-2 gives a brief explanation of each signal in the control bus.

6.5 SYSTEM CLOCK

The system clock is generated by a crystal and tank circuit tuned to 4 times the desired system frequency. This network is attached to the TIM 9904A clock driver, which counts down the input signal from the tank and crystal into four non-overlapping clock phases at MOS signal levels for the TMS 9900. The inverse of these phases is output to TTL levels for the remainder of the system.

Also on the TIM 9904A, the RESET function is latched and synchronously presented to the TMS 9900; this ensures synchronization with the correct phase. The crystal is a third overtone series/parallel-resonant crystal (see Figure 6-4). The TTL clocks (ø1-, ø3-) are routed to the RESET-/LOAD- and MEMCYC- logic, as well as to the P1 connector, the TMS 9901, the TMS 9902, and memory control.

CAUTION

If pins 11 and 12 of the TMS 9904 (ø1 and ø2) are shorted, the device will overheat and go into thermal runaway almost instantly.

TABLE 6-2. CONTROL BUS SIGNALS (1 of 3)

Signal	Active State	Group	Purpose
MEMEN- (memory enable)	Low	Memory	Enables memory devices; address on address bus is for memory. This tri-state signal is driven low by the current bus master to indicate that a memory (or memory-mapped I/O) access is taking place. This signal may remain low for several consecutive memory cycles. It is inactive (high) during non-memory cycles.
DBIN (data bus input)	High	Memory	Shows state of processor's data bus: high is input to processor, low is output. This tri-state signal is driven active high when the current bus master is doing a read memory access and is active low when the current bus master is doing a write memory access. DBIN is used to control data buffer direction on memory modules.
WE- (write enable)	Low	Memory	This tri-state signal, generated by the current bus master, is active low to indicate that data is present on the data bus being written into memory. The rising edge of WE- is used to clock write data into memory.
MEMCYC- (memory cycle)	Low	Memory	This tri-state signal is driven low by the current bus master to indicate that a memory cycle is in progress. The rising edge of MEMCYC- indicates that this memory cycle is complete. MEMCYC- may be used to control output data buffer enable on memory modules.

TABLE 6-2. CONTROL BUS SIGNALS (2 of 3)

Signal	Active State	Group	Purpose
READY	High	Memory	This normally high open-collector signal can be pulled low by the addressed memory (or memory mapped I/O) module to allow that module additional time to process the other bus signals onboard. The current bus master enters a wait state until the addressed module releases READY.
WAIT	High	Memory	Signal from microprocessor when it enters a wait state and suspends operation. Occurs when READY line is not active (low) during a memory access. Acknowledges that the processor is waiting for the memory to raise the READY line.
HOLD-	Low	Processor Activity	Requests processor to give up control of address, data buses, MEMEN-, WE-, and DBIN (all go into high impedance, used for DMA).
HOLDA	High	Processor Activity	Processor acknowledges that it has given up control of lines specified above for HOLD, and has suspended activity.
ø1- (BUSCLK.B-)	Low	Clock	ø1TTL output of TIM 9904
ø3- (REFCLK-)	Low	Clock	ø3TTL output of TIM 9904
EXTCLK.B-	Low	Clock	External TTL clock input to TIM 9904A
CLK- (CLK.B-)	Low	Clock	Output of internal oscillator (OSCOUT) of TIM 9904A

TABLE 6-2. CONTROL BUS SIGNALS (3 of 3)

Signal	Active State	Group	Purpose
PRES.B- (power-on reset)	Low	Reset/Load	Causes reset interrupt; TMS 9900 retrieves vectors from M.A. 0000 and 0002 (hex); drives IORST-. Normally generated by system power supply during powerup. This open-collector signal, when active low, resets the designated primary processor which issues IORST- to the system. PRES.B- must remain active for 100 ns after all power voltages are stable. This asynchronous signal is intended for system initialization or full system reset if activated after initialization.
IORST- (IORST.B-)	Low		I/O resets mapper chip and TMS 9901. Does not cause interrupt reset. When active low, this signal indicates a system reset and that all I/O modules should be reset to their initial or idle condition. This signal is active for at least two cycles of REFCLK-.
RESTART.B-	Low	Reset/Load	Resets the TMS 9900, then causes IORST.B- to be active, then traps to vectors located at $FFFC_{16}$ and $FFFE_{16}$ (logical addresses).
INT1.B- to INT15.B-	Low	Interrupt	External request for interrupt to TMS 9900, monitored by TMS 9901.
IAQ	High	Misc.	Signifies this memory cycle to be an instruction fetch.
INTREQ- and IC0-IC3	Low	Interrupt	INTREQ- active signifies the TMS 9901 has decoded a valid interrupt. The level of the interrupt is read by the TMS 9900 on lines IC0 to IC3.

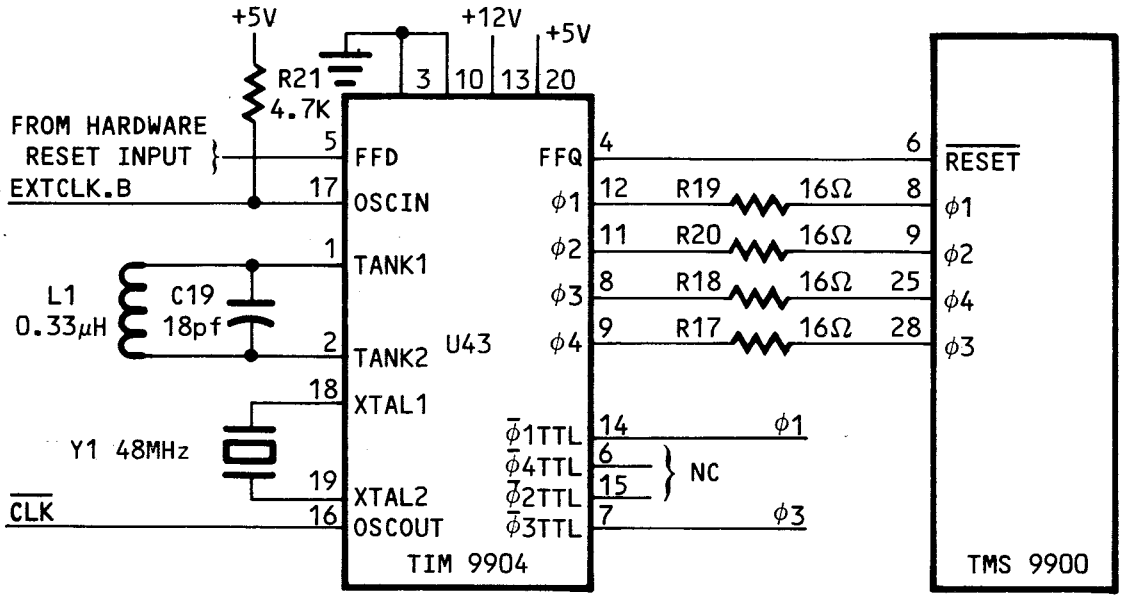


FIGURE 6-4. SYSTEM CLOCK CIRCUITRY

6.6 CENTRAL PROCESSING UNIT

The TMS 9900 microprocessor is the central processing unit (CPU) for the TM 990/102. The responsibilities of the CPU include:

- memory, CRU, and general bus control,
- instruction acquisition and interpretation,
- timing of most control signals and data,
- general system initialization.

Figure 6-5 groups the TMS 9900 pins by function. The address bus addresses memory as well as the TMS 9901 and TMS 9902 CRU devices, and provides the codes for the external instructions. The data bus carries all memory data, including instruction code as well as program data and addresses. External interrupt requests are encoded by the TMS 9901 as a binary number and presented to the TMS 9900 microprocessor via IC0 to IC3.

Memory operations are initiated by placing an address on the address lines along with MEMEN-, MEMCYC-, DBIN, and eventually WE-. If the memory cycle is an instruction fetch, IAQ goes active also. READY is sampled and the memory cycle is ended one clock cycle after READY is active. Memory timing is covered in detail in section 6.10.

CRU operations are initiated by placing an address on lines A3 to A14 of the address bus (A0 to A2 are zeroes during CRU operations). CRUIN is sampled for an input operation; otherwise it is ignored. For an output operation the datum is placed on CRUOUT and strobed with CRUCLK. The only onboard CRU operations are those using the TMS 9901 or TMS 9902; other CRU operations are offboard.

Figures 6-6 and 6-7 show the data flow and operational flowchart of the microprocessor. For more information, refer to the TMS 9900 Microprocessor Data Manual.

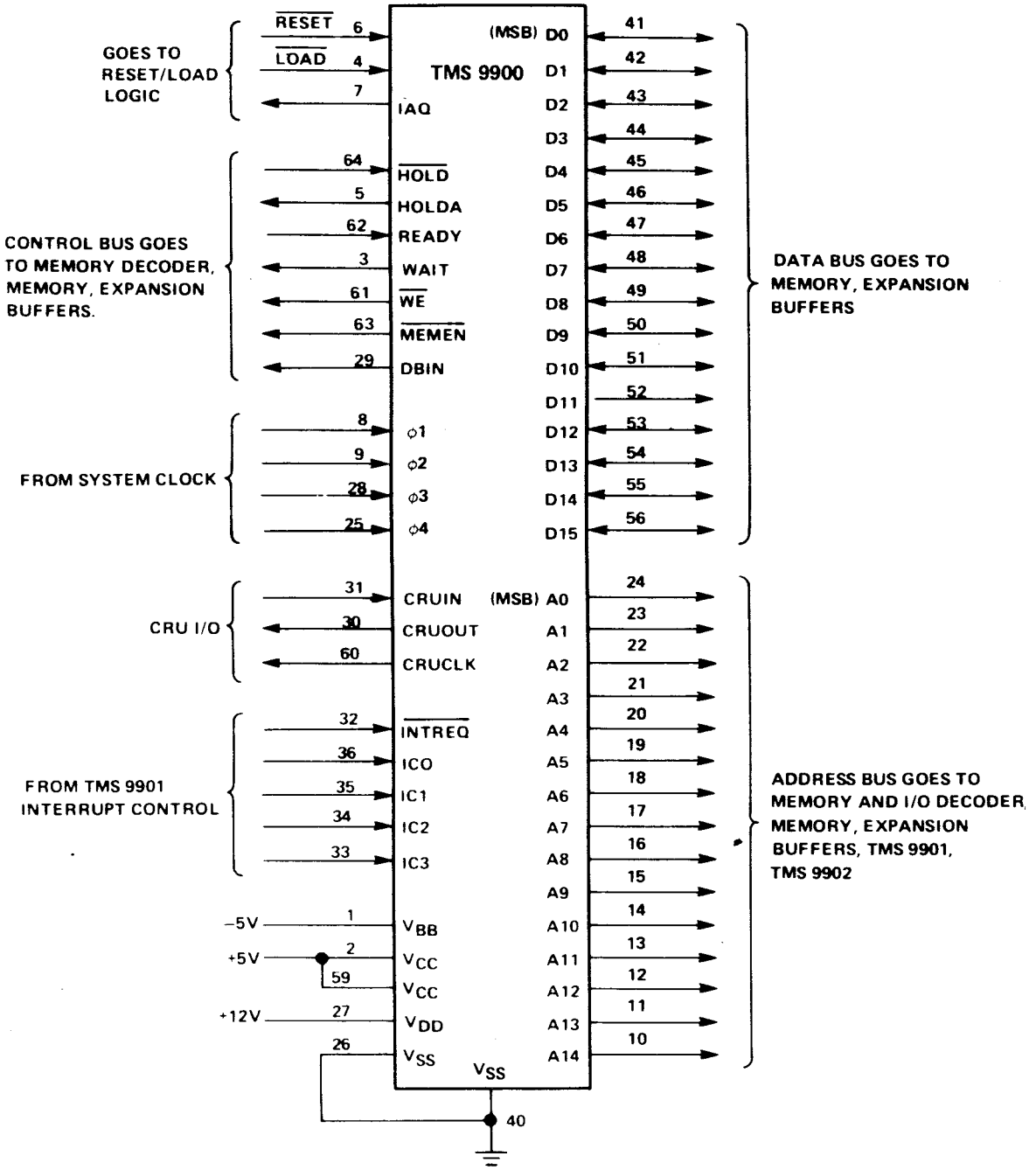


FIGURE 6-5. TMS 9900 PIN FUNCTIONS

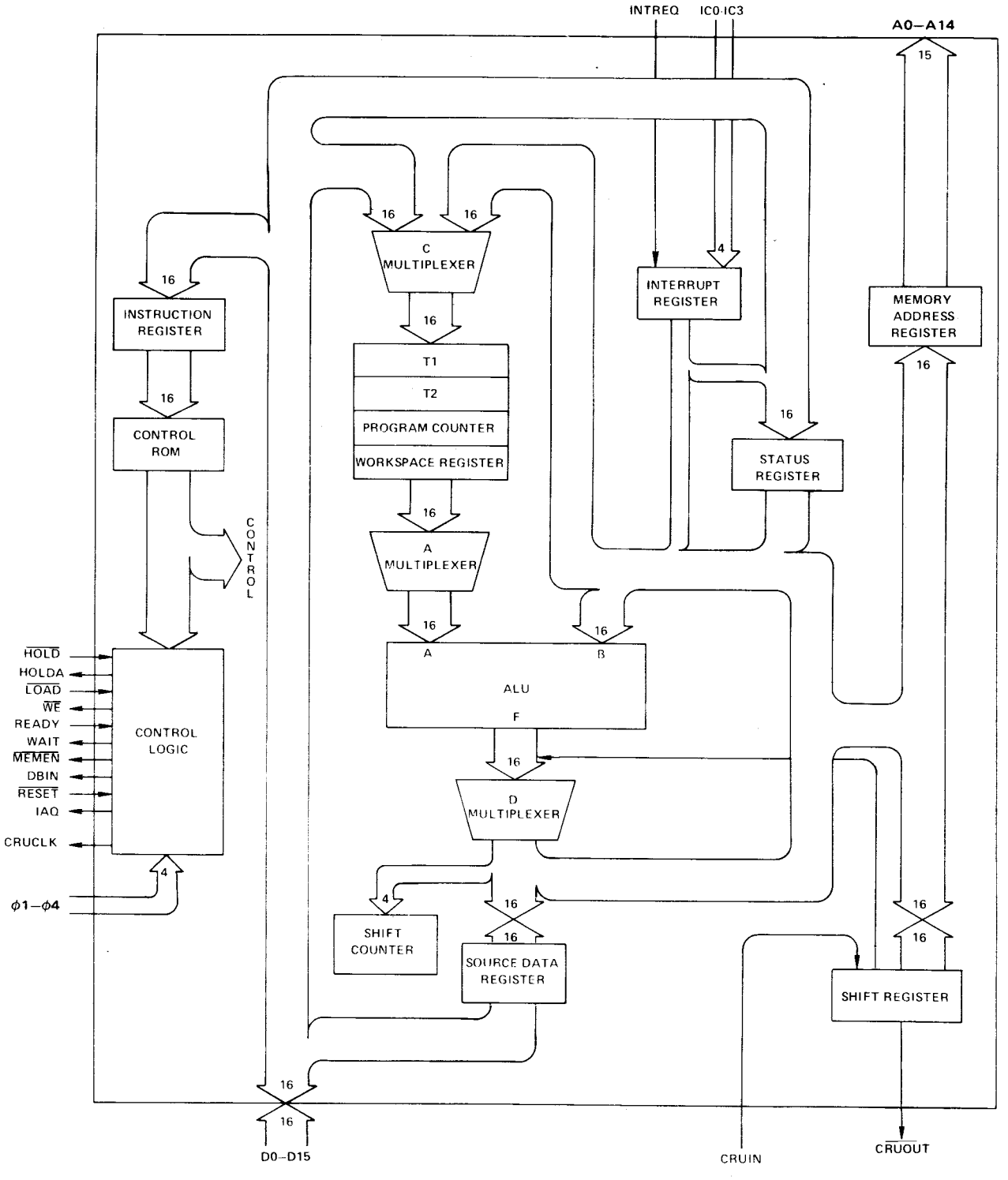


FIGURE 6-6. TMS 9900 DATA AND ADDRESS FLOW

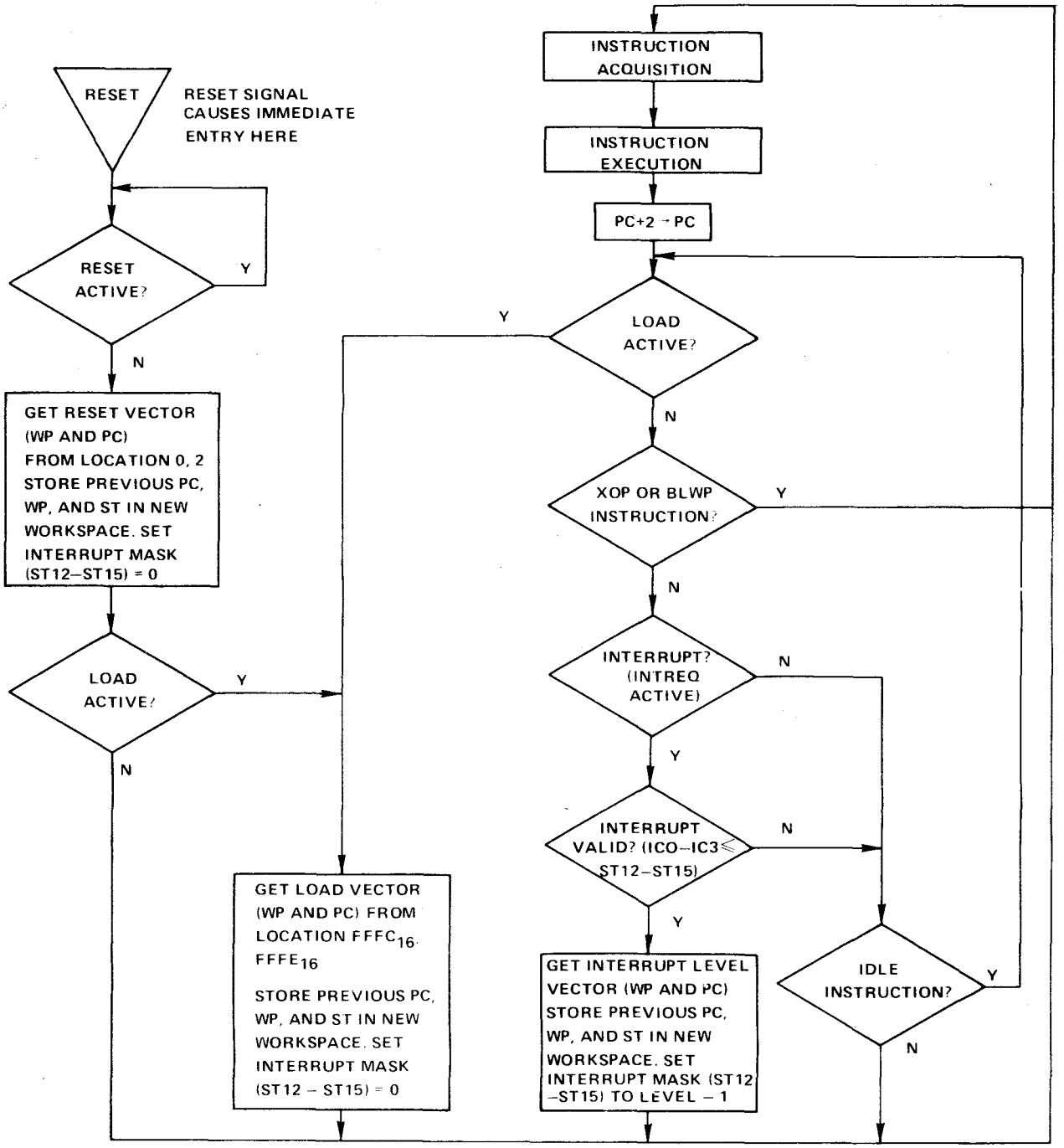


FIGURE 6-7. TMS 9900 CPU FLOW CHART

6.7 RESET/LOAD LOGIC

After the clock and the CRU, the next circuitry most closely associated with microcomputer operation is the logic dealing with RESET- and LOAD-. These functions provide the means for initializing the system with workspace (WP) and program counter (PC) vectors from two different sources:

- RESET vectors from 0000_{16} (WP) and 0002_{16} (PC),
- LOAD vectors from $FFFC_{16}$ (WP) and $FFFE_{16}$ (PC)

In the TMS 990/404 monitor, a normal RESET function is to initialize the board at powerup or for a hardware reset; while LOAD is used as a return vector.

6.7.1 RESET Function

A RESET switch can be installed on the terminal strip of a TM 990 card chassis as explained in section 2.4.3. The PRES.B- signal at connector P1 (P1-94) is the RESET switch signal: it goes to a Schmitt trigger gate to lessen the effect of multiple reset pulses due to noise or bounce on the microcomputer. A PRST- signal is then issued to reset the DRAM controller at U23 (shown in Figure 6-8). After being inverted again, the reset signal is routed to the FFD input of the TIM 9904A which then synchronizes it with clock phase ø3 before it is presented to the microprocessor's RESET- input.

The reset input also goes to two flip-flops which generate the IORST- signal, and keep it active for two ø3 clock pulses. IORST- sets the memory mapper (74LS612) to the mapper-off mode, resets the TMS 9901 interrupt controller, and resets any other devices attached to it offboard. This IORST- signal is also generated by the external instruction RSET, but it is important to realize that the RSET instruction in a program generates only IORST- and <u>not</u> a full (hardware) RESET interrupt.

A hardware reset causes the following to occur:

- clears I/O devices on IORST- line:
  - resets TMS 9901 (all interrupt masks to 0, clears IC0-IC3 to all zeroes, INTREQ- high, disables clock)
  - sets memory mapper to mapper-off mode,
- resets the DRAM controller (via signal PRST-),
- inhibits memory write and CRU operations,
- sets TMS 9900 status register interrupt mask to 0000_{16},
- causes processor to trap to WP and PC vectors at 0000_{16} and 0002_{16}.

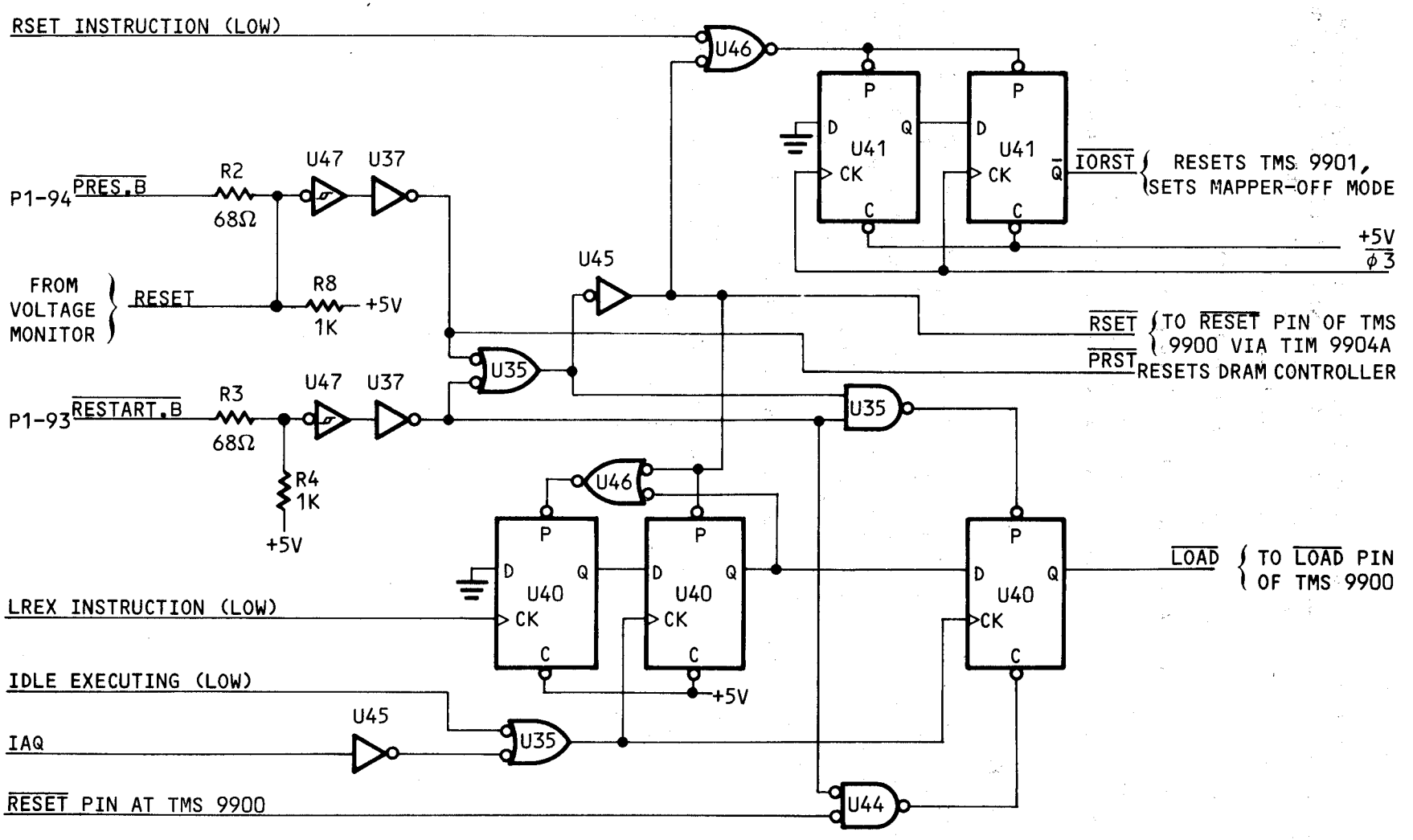


FIGURE 6-8. RESET AND LOAD LOGIC

A hardware reset is caused by bringing backplane pin P1-94 to a low state (tie momentarily to system ground).

A software reset causes the following to occur:

- clears I/O devices on IORST- line (same as first entry for hardware reset),
- puts mapping logic into mapper-off (transparent) mode.

A software reset is caused by executing the RSET instruction.

A TL 7705 voltage monitor is tied to the incoming reset line. This monitors Vcc. When a powerfail condition is sensed, the voltage monitor issues PRES.B-. This signal pulls low the pin-9 input to the NAND gate at U35 (shown in Figure 6-8), enabling the RESET function. This circuit is shown on page 2 of the schematics in Appendix A.

6.7.2 LOAD Function

The LOAD function causes the following to occur:

- RESET- occurs momentarily which issues IORST- which resets the TMS 9901 and places the memory mapper in the mapper-off mode (this does not apply when caused by LREX instruction),
- processor traps to WP and PC vectors at M.A. $FFFC_{16}$ and $FFFE_{16}$.

A LOAD is caused by one of the following if microprocessor output RESET- is inactive:

- by executing the software instruction LREX, or
- by setting RESTART.B- to logic zero state on connector P1 (P1-93).

The LOAD function is triggered by either activating RESTART.B- using an external switch connecting P1-93 to the ground plane or by executing the external instruction LREX (attaching an external switch is covered in Section 2). RESET overrides LOAD because a RESET- signal presets the LOAD- input high. This is important when both requests occur simultaneously. However, during a LOAD enabled by the hardware RESTART.B- input, a preset is first momentarily executed to preset circuitry, then the delayed LOAD function occurs.

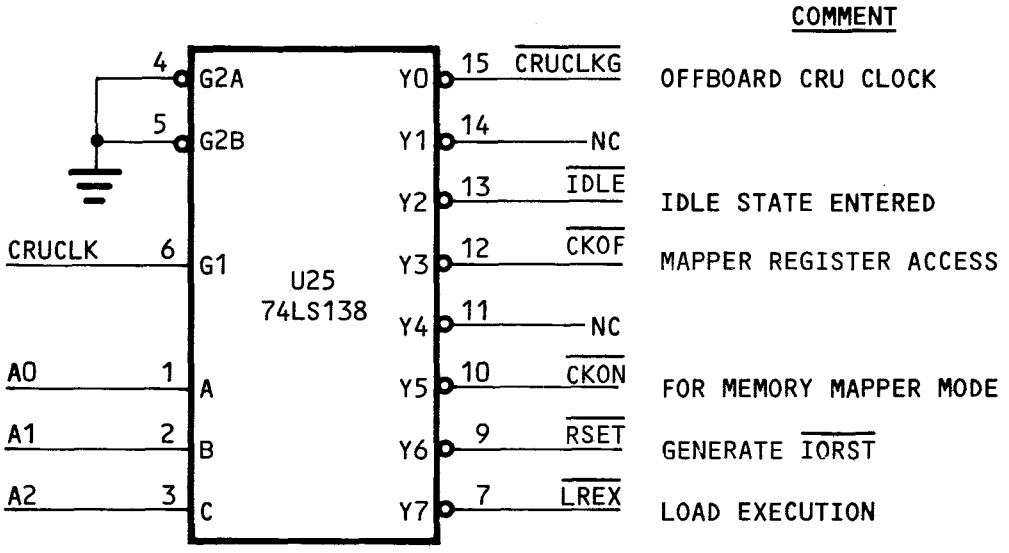
In using LREX, the LOAD function is delayed for two instruction cycles (two IAQ pulses) or idle pulses (IDLE mode active), then the LOAD is triggered. The single-step monitor function uses this delay. In the monitor routine, after LREX is executed, the next instruction is an RTWP to the user program; the second instruction is the single user instruction in single-step mode. Then the delayed LOAD brings control back to the monitor again.

6.8 EXTERNAL INSTRUCTIONS

External instructions are those which, when executed by the processor, cause address lines A0, A1, and A2 to be set to a known state, and CRUCLK to become active. By monitoring those address lines, execution can be defined external to the processor. The instructions and descriptions are listed in Table 6-3. Circuitry is shown in Figure 6-9.

TABLE 6-3. EXTERNAL INSTRUCTIONS

Instruction	Opcode	A0	A1	A2	Description
IDLE	0340	0	1	0	Suspends processor until RESET (PRES.B- active at P1-94), LOAD (RESTART.B- at P1-93), or enabled interrupt occur.
RSET	0360	0	1	1	Generates IORST- which takes the memory mapper chip out of the mapper mode and disables mapper-register access; it also resets the TMS 9901 and any offboard functions connected to IORST.B-. It does not pull RESET- low at the TMS 9900.
CKON	03A0	1	0	1	Sets the memory mapper chip to the mapper-on mode (extended addressing or 19-address lines). Does <u>not</u> take mapper out of mapper-register access.
CKOF	03C0	1	1	0	Sets the memory mapper chip to the mapper register mode to set the register contents. Takes mapper out of mapper-on mode.
LREX	03E0	1	1	1	Causes LOAD, delayed by two IAQ or IDLE pulses. A LOAD causes the TMS 9900 to trap to WP & PC vectors to be obtained while M.A. $FFFC_{16}$ and $FFFE_{16}$ respectively are on the TMS 9900 address lines A0-A14 (i.e., this address can be interpreted by memory mapper). Does <u>not</u> enable IORST-.



EXTERNAL INSTRUCTION	VALUE ON A0	A1	A2	OUTPUT ENABLED	COMMENT
None	0	0	0	Y0	No external instruction executed
IDLE	0	1	0	Y2	Microprocessor in IDLE state until LOAD, RESET, or interrupt
RSET	0	1	1	Y3	Causes IORST- to be active; mapper to mapper-off mode
CKON	1	0	1	Y5	Mapper to mapper-on mode (does not turn off mapper-register mode)
CKOF	1	1	0	Y6	Mapper to mapper-register mode (turns off mapper-on mode)
LREX	1	1	1	Y7	Causes LOAD following two instruction accesses (does not enable IORST-)

FIGURE 6-9. EXTERNAL INSTRUCTION CIRCUITRY

In all external instruction cases, note that CRUCLK is active (high); however, address lines A0, A1, and A2 are nonzero values so that these instructions are differentiated from a CRU output operation (in which these lines would be all zeroes). CRUCLK is pulsed high once for each instruction except IDLE. When in the idle state, the three-bit code and CRUCLK pulses occur repeatedly until the idle state is terminated.

6.9 MEMORY ADDRESS GENERATION AND DECODING

This section explains address encoding. Section 6.10 covers memory signal timing. The memory address map configurations are shown in Figure 6-10. Which address lines are decoded depends upon which function is taking place (i.e., memory access with or without the memory mapper, accessing memory mapper registers, etc.). Table 6-4 outlines different addressing modes.

Memory addresses originate at the onboard TMS 9900 microprocessor or by an offboard processor executing a DMA operation. For onboard memory operations, a memory mapper chip (74LS612) is used to expand the 15-bit (64 K bytes maximum) address bus of the TMS 9900 to a 19-bit (1 M bytes maximum) address. If the memory mapper is not used, memory is accessed via the processor's address bus A0-A15, addressing M.A. 0000_{16} to $FFFF_{16}$ (0 to 64 K bytes). By using the memory mapper, onboard addressing ranges from 0000_{16} to $1FFFF_{16}$ (0 to 128 K bytes) depending upon mapper register contents. DMA by an external CPU cannot use the memory mapper chip; it accesses memory directly. Offboard addressing by the TM 990/102 can be a maximum of $FFFFF_{16}$ (1 M bytes).

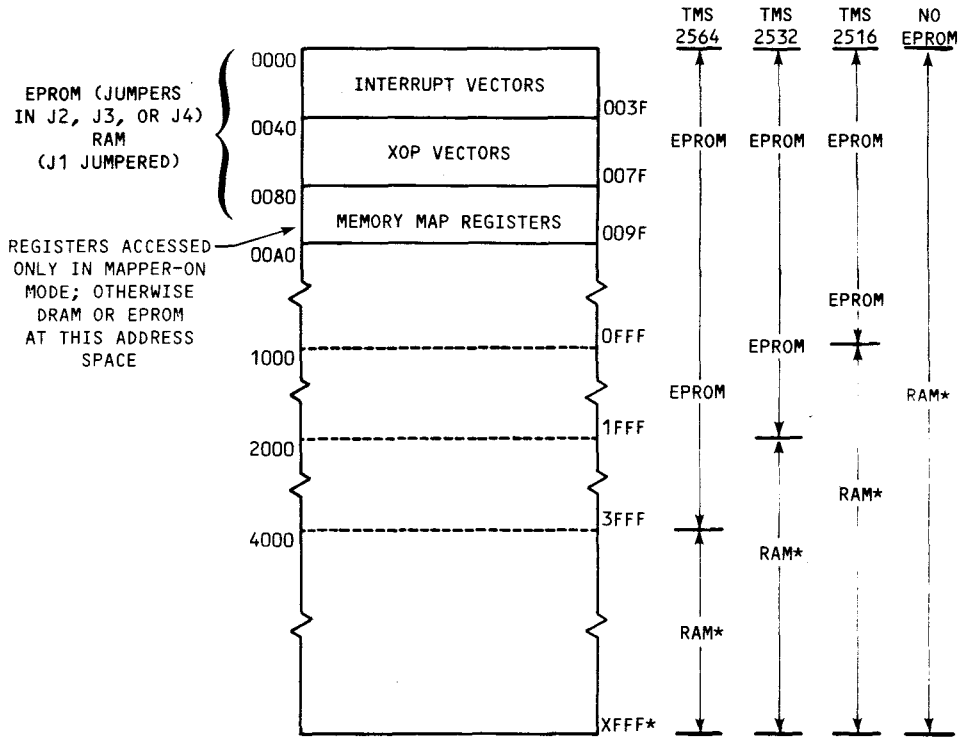
6.9.1 DMA Operations Onto the TM 990/102

An external processor can perform DMA to the TM 990/102 by bringing HOLD.B- active (pin P1-92). This pin is held high by a pullup resistor as shown on page 2 of the schematics. HOLD- active causes the microprocessor to enter a hold state and place its WE-, MEMEN-, DBIN, address, and data buses in the high impedance state. It then issues HOLDA (hold acknowledge) which switches the buffers for the address lines to the onboard direction along with offboard control signals MEMEN.B- (meaning address bus contains memory address), WE.B- (latches a DRAM write), and DBIN.B (used to state DMA processor data buffer direction). HOLDA places the mapper output lines in a high impedance state. DMA mode continues as long as HOLD.B- remains active. Because an external CPU cannot manipulate the external instruction circuitry (i.e., CKON, CKOF), it cannot use the memory mapper. However, onboard address lines XA0.B- (MSB) to XA3.B- can be used by DMA to access the upper 64 K of onboard memory (10000_{16} to $1FFFF_{16}$) with lines XA0.B- to XA3.B- having a value of 0001_2. The same is true for accessing the lower 64 K of onboard memory (0000_{16} to $FFFF_{16}$) with XA0.B- to XA3.B- equal to 0000_2.

6.9.2 Memory Mapping

Memory mapping on the TM 990/102 utilizes a 19-bit address bus for DRAM (not EPROM) addressing. Because the TMS 9900 microprocessor has a 15-bit bus, able to address 32 K words (64 K bytes), the 74LS612 memory mapper chip provides additional addressing capability. An important term used in this section:

IOReset: Signal IORST- enabled; caused by one of the following: (1) RSET instruction, (2) a hardware RESET caused by PRES.B- active at P1-94, (3) by a powerup reset; or (4) a hardware restart (RESTART.B-



* RAM CONFIGURATION DEPENDS UPON MEMORY MAPPING USED.

FIGURE 6-10. TM 990/102 MEMORY ADDRESS MAP

active at P1-93). Note that only no. (1) will cause IORST- to be active without causing a context switch to vectors in lower memory (i.e.,the RSET instruction merely enables IORST- for resetting the mapper chip as well as resetting the TMS 9901 and devices on the TM 990 bus.

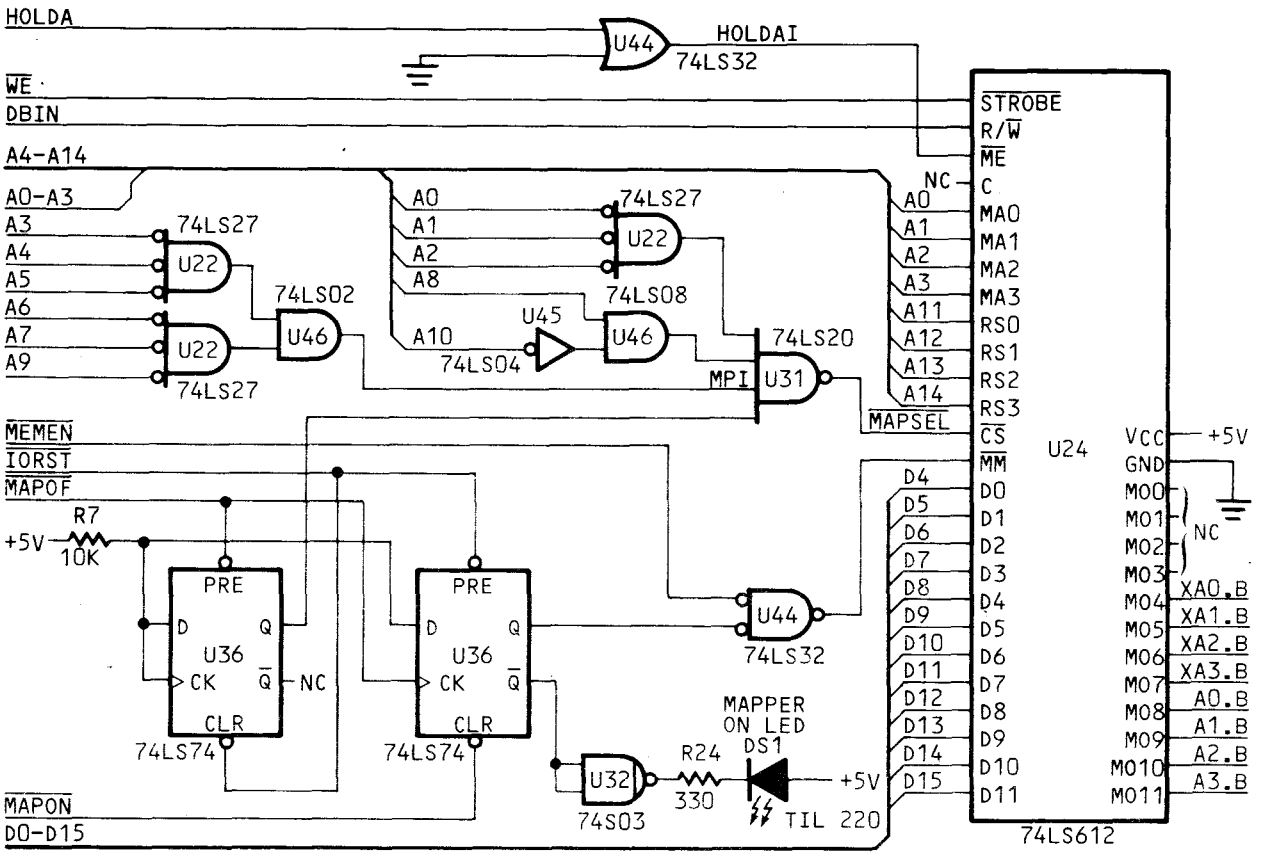
The memory mapper chip contains 16 twelve-bit registers, of which only the eight LSBs are used. When accessed during a memory read or write operation, the contents of one register are applied to address lines XA0 to A3 (eight MSBs of 19-line address). Although the memory mapper registers are 12 bits wide, only eight bits are used and the four MSBs (left-most bits) are not used. The memory mapper has two distinct modes:

- Mapper-On mode where the register contents are used to build extended addresses
- Mapper-Off mode where the extended address is not used (resulting in use of only address lines A0 to A14).

Mapper chip enabling logic (also see Figure 6-11) for different modes is as follows:

	CS-	STROBE-	R/W-	MM-	ME-
Mapper Register Write Access	0	0	0	X	X
Mapper Register Read Access	0	X	1	X	X
Mapper-On Mode	1	X	X	0	0
Mapper-Off Mode	1	X	X	1	0

X=Don't care



NOTE:

CKOF INSTRUCTION CAUSES MOMENTARY $\overline{\text{MAPOF}}$
CKON INSTRUCTION CAUSES MOMENTARY $\overline{\text{MAPON}}$
RSET INSTRUCTION CAUSES MOMENTARY $\overline{\text{IORST}}$

FIGURE 6-11. MEMORY MAPPER AND ENABLING CIRUITRY

6.9.2.1 Mapper Register Access

Mapper register access allows changing or reading register contents. Entering this requires two programming steps:

- Execute CKOF instruction to enable MAPOF- to go low, then high.
- Write or read to an address from 0080_{16} to $009E_{16}$ (address space of the mapper's 16 registers).

This mode requires the mapper CS- to be low and the mapper R/W- line high for a read and low for a write. STROBE- is uesd to enter data during a write. CS-

TABLE 6-4. ADDRESS LINES AND ADDRESSING MODES (1 of 2)

MODE/FUNCTION	ADDRESS LINES AFFECTED	COMMENTS
Mapper Register Access	A0-A14	In this mode, one of the sixteen memory mapper chip (74LS612) registers are being written to or read from. Address lines A0 to A10 must contain an address within 0080_{16} to $009E_{16}$, the memory mapped area for the registers. Address lines A11-A14 select the desired register to be read or written to. The contents of the registers become the values on address lines XA0 to XA3 and A0 to A3 in the mapper-on mode. Mapper register access is gained by executing the CKOF external instruction. It is exited by an IOReset (defined in section 6.9.2) but is not exited by entering mapper-on mode with a CKON.
Mapper-On	XA0-XA3, A0-A14	In this mode, a register on the memory memory mapper is addressed by A0-A3, and its eight-bit contents are placed on the address lines as XA0-XA3 and A0-A3, allowing a 19-bit address (addresses a maximum of 500 K words or 1 M bytes; this is also termed extended addressing) This mode is entered by executing the

enabling logic requires MAPSEL- to be low; this is the low-level output of a four-input NAND gate at U31 (shown in Figure 6-11). The enabling four high inputs to this gate are as follows:

- Address lines A0 to A2 low
- Address line A8 a one and A10 low
- Address lines A3-A7 and A9 low
- MAPOF- active which presets flip-flop U36 (executing CKOF instruction activates MAPOF-).

Logic to drive these gates is shown in the center of Figure 6-11. Following this logic, the memory mapper registers are memory mapped at address 0080_{16} (lines A3 to A10 low except for A8) with each register located on an even address (0080_{16} to $009E_{16}$).

MAPOF- is activated by executing the CKOF external instruction (external instructions shown in Figure 6-9). The 74LS138 3-to-8 decoder at U25 interprets address lines A0 to A2. The external instruction CKOF (110_2 value for A0-A2) causes signal MAPOF- to go low. MAPOF- presets flip-flop U36 with its latched Q output (high) as one of the enabling inputs to NAND gate U31. It also takes the mapper chip out of the mapper-on mode (section 6.9.2.2) by clocking the other flip-flop at U36 which unlatches the chip from the memory mapper mode. The mapper-register mode is exited by executing the RSET instruction or another cause of IOReset as defined in section 6.9.2.

TABLE 6-4. ADDRESS LINES AND ADDRESSING MODES (2 of 2)

MODE/FUNCTION	ADDRESS LINES AFFECTED	COMMENTS
Mapper-On (continued)		CKON external instruction, and is exited by an IOReset (see 6.9.2) or a CKOF external instruction. LED DS1 illuminates only when in this mode.
Mapper-Off	A0-A14	In this case, the value on A0-A3 does not select a mapper register but is applied to address lines A0-A3; thus, addressing the least significant 32 K words (64 K bytes). This mode is entered by an IOReset (see 6.9.2) of CKOF. It is exited bya CKON external instuction.
EPROM Select	A0-A3	EPROM always begins at M.A. 0000_{16} if populated. Several EPROM types can be used: TMS 2516, TMS 2532, or TMS 2564. EPROM can be unpopulated; if populated along with DRAM in the same space, the EPROM overlays the DRAM. If unpopulated and J1 jumpered, DRAM begins at M.A. 0000_{16}. If populated, jumpers J2-J4 have to reflect EPROM type.
DRAM Select	XA0-XA3, A0-A3	Onboard DRAM can be located throughout the entire 128 K byte area by populating TMS 4164 DRAMs or in a 64 K byte block by populating 4532 DRAMs. Logic selects RAM for memory addresses above designated EPROM type. DRAM must be within a 0000_{16} to $1FFFF_{16}$ range, designated by 64 K/128 K jumpers (E7 to E10).

Thus, to write to or read a mapper register, first execute CKOF, then write to or read from one of the registers beginning at M.A. 0080_{16} (register 0). Each register is on an even address (0080, 0082, etc.). For example:

```
CKOF                    ENTER MAPPER-REGISTER ACCESS MODE
MOV   @>0082,R2         READ MAPPER REGISTER 1 CONTENTS INTO R2
MOV   R3,@>0080         WRITE R3 CONTENTS TO MAPPER REGISTER 0
```

6.9.2.2 Mapper-On Mode

In the mapper-on mode, an extended address is built using eight bits from one of the 16 mapper registers. These eight bits come from the eight LSBs of the mapper register and are placed on address lines XA0 (MSB) to XA3 and A0 to A3. Thus, the four MSBs of the TMS 9900's address bus drive the the mapper chip to derive the eight MSBs of the 19-bit address bus. LED DS1 illuminates only when the mapper-on mode is active.

Mapper chip criteria to place it in the mapper-on mode include:

- Pin ME- enabled, meaning the mapper chip should drive outputs
- Pin CS- disabled
- Pin MM- enabled, which calls up the memory mapper mode

To enter the mapper-on mode, execute the CKON external instruction. Its resulting A0 to A2 address-line value is interpreted by the external address decoder at U25 whose Y5 output (MAPON-) clears flip-flop U36. The resulting low Q output (PRE high) is one of two low inputs ANDed at the positive OR gate at U44; the other low input is MEMEN- (memory address on the address lines). The low output of this positive OR gate enables the MM- (memory map) input to the mapper chip. If both CKON and CKOF are executed without an intervening reset between, the mapper-on mode and the mapper-register can be entered alternately depending upon changes in the address lines. See CAUTION in section 6.9.2.1.

This mode is exited by (1) entering the mapper-register mode (6.9.2.1) or (2) an IOReset (defined in section 6.9.2) which enables IORST- to the pin 10 PREset input of the U36 flip-flop. Signal IORST- is enabled by either a RSET instruction, an external reset (RESTART.B- active at P1-93), or a powerup reset (PRES.B- active at P1-93). In exiting this mode by IORST-, the mapper-off mode is entered. Exiting this mode also turns off LED DS1.

6.9.2.3 Mapper-Off mode

This mode is entered when the board comes up from an IOReset as defined in 6.9.2.

In the mapper-off mode, the mapper is deleted from the address circuitry and memory addresses are, in essence, the same as A0 to A14 coming from the TMS 9900 microprocessor. In the mapper-off mode, address lines A0 to A3 (MA0 to MA3 inputs) will pass through the memory mapper and exit unchanged at outputs MO8 to MO11. These outputs are also the mapper-on mode outputs for these same address lines (A0.B to A3.B). The other outputs, MO0 to MO7, will be low in this mode.

Mapper chip criteria to place it in the mapper-off mode include:

- Pin ME- enabled, meaning the board in not in a hold mode
- Pin CS- disabled
- Pin MM- disabled

6.9.3 EPROM Organization

Two 28-pin sockets are provided for selecting one of three EPROM types:

- TMS 2516: 2 K x 8 for 2 K words
- TMS 2532: 4 K x 8 for 4 K words
- TMS 2564: 8 K x 8 for 8 K words

When populated, EPROM is always from M.A. 0000_{16} to the last byte populated depending upon memory type selected. If DRAM is mapped at the same locations, the EPROM will overlay the DRAM, with DRAM starting immediately after EPROM.

6.9.4 Dynamic RAM (DRAM) Organization

Onboard DRAM is organized as 16 chips, each a 64 K x 1 or 32 K x 1 organization which allows populating either 64 K words (128 K bytes) or 32 K words (64 K bytes) respectively. A TMS 4500 dynamic RAM controller provides the timing and logic for the eight-bit column and row address selection to access individual 16-bit words. It also monitors elapsed time to begin refresh cycles and strobe the appropriate pins during refresh.

DRAM can be one of two types:

- TMS 4164: 64 K x 1 each for 128 K words
- TMS 4532: 32 K x 1 each for 64 K words

6.9.5 DRAM/EPROM Selection

The 74LS682 decoder at U27 compares binary inputs to the P side of the chip with the values on the Q side with the P>Q- output active (high) to select EPROM. The MSBs, P7 to P4, are absolute values while P3 to P0 compare the variables on address lines A0.B to A3.B. The P>Q- output is inverted to make EPROMSEL- (EPROM selected). Jumpers J1 to J3 connect Q side to ground or no connection (thus Vcc. Logic to enable P7 to P4 (P not greater than Q) is:

	P-Side	Q-Side	Comment
P7	Vcc	MAPSEL- high	MAPSEL- disabled means not in mapper mode
P6	MEMEN-	Gnd	MEMEN- enabled
P5	MEMLOW-	Gnd	MEMLOW- indicates map 0 selected (0-64 K)
P4	Vcc	Unconnected	Jumper J1 selectable; unconnected = Vcc; if ground selected, all EPROM deselected

The eight address lines XA0 to A3 are monitored by logic as shown in Figure 6-12 for selection of DRAM or EPROM. The 74LS682 decoder at U27 monitors address lines A0.B to A3.B at inputs P3 to P0 as follows:

	P3	P2	P1	P0		Set Jumpers			
XA0	A0	A1	A2	A3	EPROM Used	J1	J2	J3	Comment
0	0	0	0	0	TMS 2516	Hi	Gnd	Gnd	0000 to 0FFF EPROM addressed
0	0	0	0	1	TMS 2532	Hi	Hi	Gnd	0000 to 1FFF EPROM addressed
0	0	0	1	X	TMS 2564	Hi	Hi	Hi	0000 to 3FFF EPROM addressed
X	X	X	X	X		Gnd			Cannot access EPROM
0	1	X	X	X					DRAM addressed (not EPROM)
1	X	X	X	X					DRAM addressed (not EPROM)

X = don't care

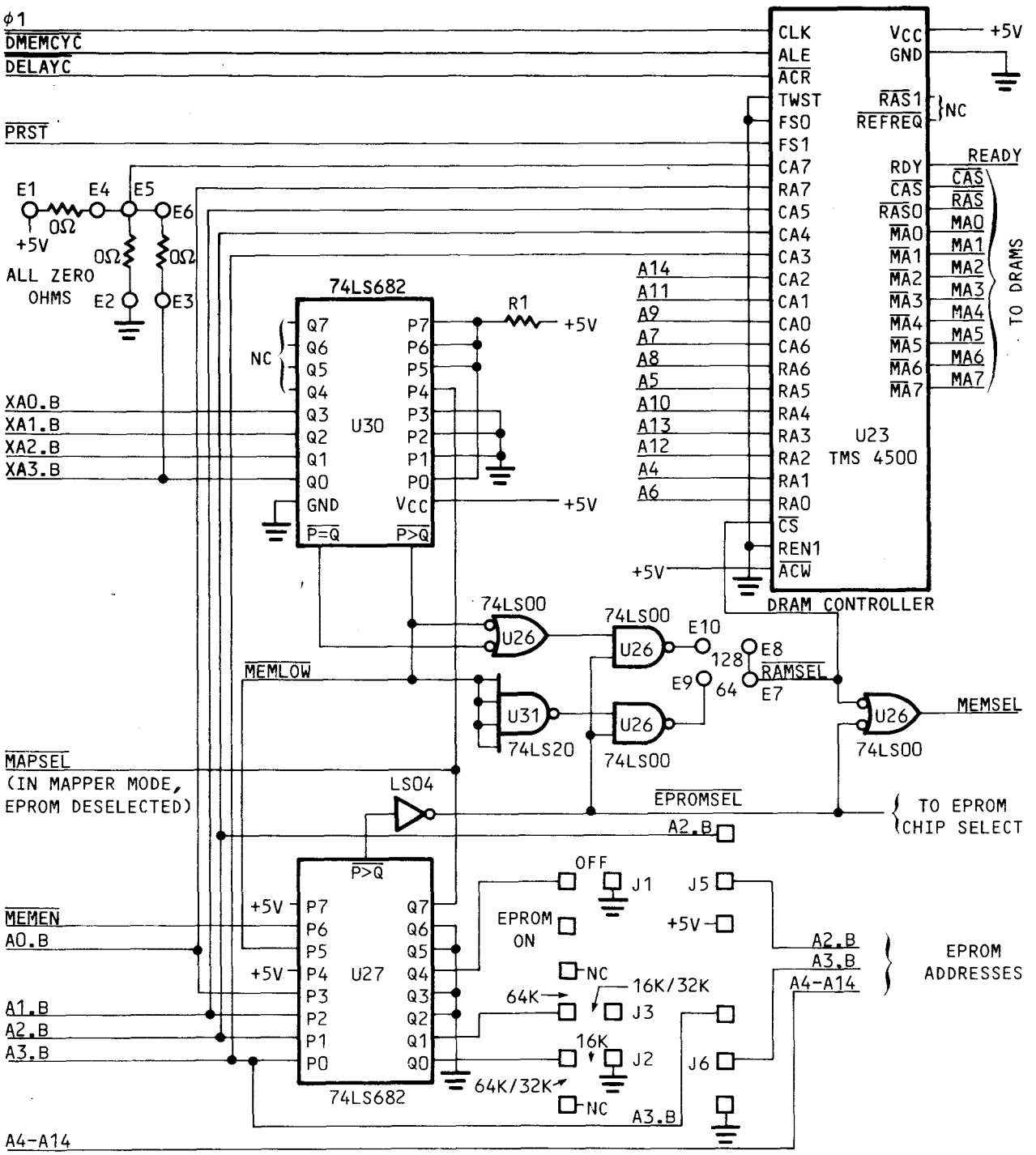


FIGURE 6-12. DRAM/EPROM SELECTION

When EPROMSEL- is active (indicating address to the EPROM area is valid), the dynamic RAM controller chip (TMS 4500 at U23) is held inactive by EPROMSEL- disabling both of the AND gates at U26 (inputs 10 and 13 shown in Figure 6-12) that control the dynamic RAM controller-chip select pin. When address lines XA0-XA3 are less than than 001_2, and EPROMSEL- is inactive, and a memory cycle is in progress, onboard DRAM is assumed as the memory desired. A 74LS682 comparator at U30 and the logic checking the amount of DRAM populated verify that extended address lines XA0.B to XA3.B contain the address (page 0 or page 1 which are the first and second 64 K) corresponding to the amount of DRAM populated (128 or 64K). The resulting signal, RAMSEL-, enables the chip select of the DRAM controller; this, in turn, triggers a DRAM memory access cycle. When onboard memory is selected, the data buffers to the bus are tri-stated to avoid conflicts on the data lines.

6.9.6 Dynamic RAM Controller (TMS 4500)

Figure 6-13 shows the TMS 4500 Dynamic RAM (DRAM) Controller. Controller functions include:

- control refresh cycle of DRAM chips,
- interpret row and column addressing for DRAM and provide timing for memory column and row address latch.

Address inputs to this controller are the regular address inputs A0 to A14 as well as the LSB of the extended addressing lines, XA3. These 16 address lines allow addressing up to 64K words of memory (128 bytes). Because only these lines are used, onboard memory is restricted to the space between 0000_{16} and $1FFFF_{16}$ even though memory mapping may be used (i.e., memory mapped addresses above $1FFFF_{16}$ will not be enabled onboard). Memory size depends upon the DRAM memory chip used:

TMS 4532	32 K x 1	16 each	32 K words (64 K bytes)
TMS 4164	64 K x 1	16 each	64 K words (128 K bytes)

Controller column and row address inputs and corresponding address line inputs are as follows:

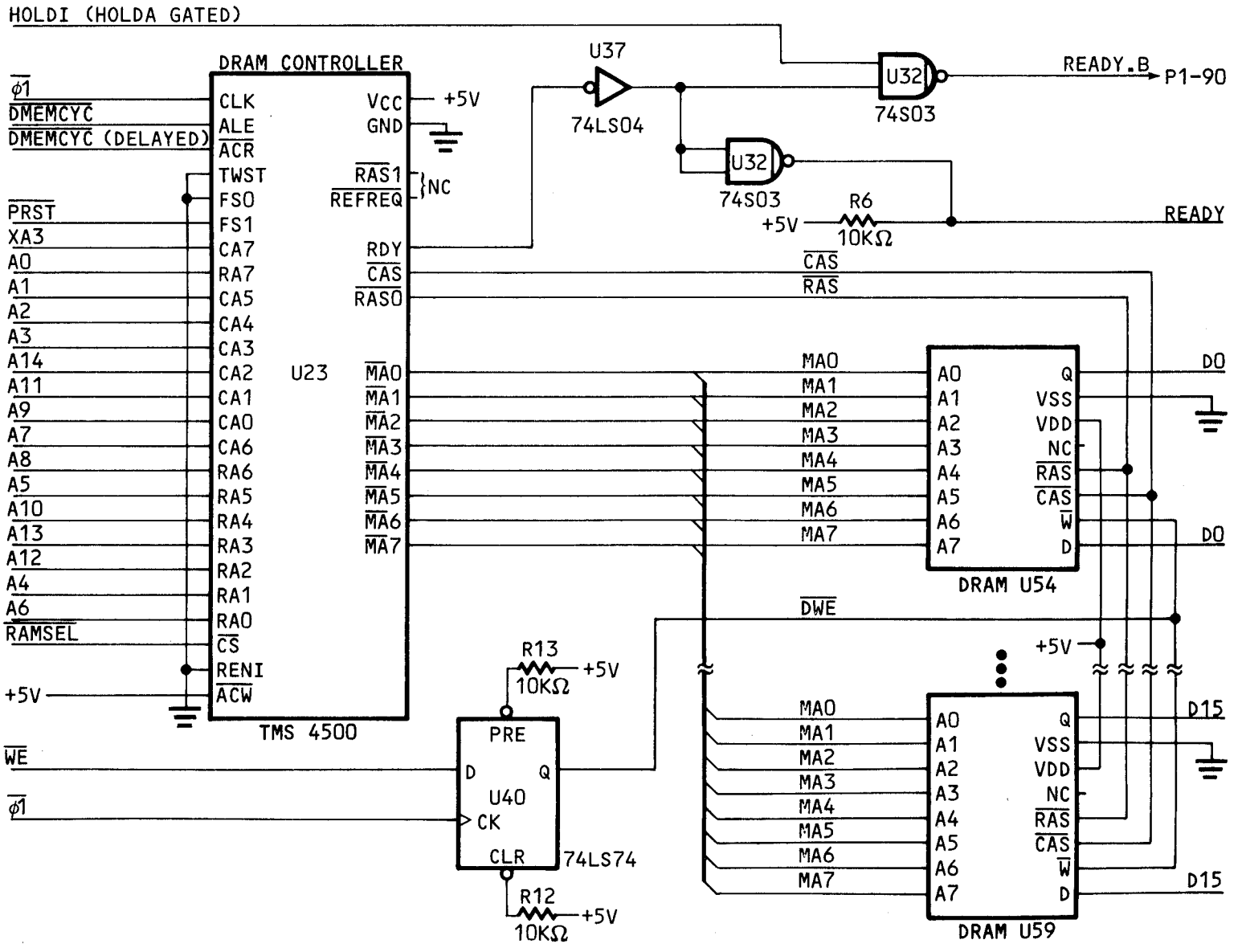
CA0	A9	CA4	A2	RA0	A6	RA4	A10
CA1	A11	CA5	A1	RA1	A4	RA5	A5
CA2	A14	CA6	A7	RA2	A12	RA6	A8
CA3	A3	*CA7	*XA3	RA3	A13	RA7	A0

*NOTE

For the TM 990/102-2, controller pin CA7 (connected to XA3), is terminated high (Vcc) or low depending upon which dash number of TMS 4532 DRAM is used. This setting is made at the factory by placing (zero-ohm) jumpers at sockets E1 to E6; these are shown in the upper left of Figure 6-12. On the same dash-2 board, all TMS 4532s must be the same dash number because the same quadrant in the memory chip must be used throughout the memory matrix.

Controller outputs to memory address inputs are:

MA0	A0	MA3	A3	MA6	A6
MA1	A1	MA4	A4	MA7	A7
MA2	A2	MA5	A5		



NOTE: DRAM PINOUTS ARE FOR TMS 4164s; SUBSTITUTE "AR" FOR "A7" FOR TMS 4532s.

FIGURE 6-13. DRAM CONTROL

In memory access, the TMS 9900's MEMEN- line is enabled (meaning a memory address is on the address bus), producing DMEMCYC- at the DRAM controller. With CS- enabled by RAMSEL- active (meaning address is not EPROM; thus DRAM) and DMEMCYC- active low to make ALE (address latch enable) low, the row address is latched in and presented on output lines MA0-MA7 with RAS- enabled. ALE- (active low) latches CS- and the address. After ACR- goes active low, the column address is latched in and presented on output lines MA0-MA7 with CAS- is enabled. When ACR- or ALE- go high, the memory cycle is terminated.

Memory wait states are strapped to a zero by DRAM controller pins TWST and FS0 grounded while FS1 is normally high since it is tied to PRST- (power up reset). These also set the refresh frequency to 64 kHz using a 3 MHz clock. At power up reset (PRES.B- pulled low at P1-94), all three lines are low causing a reset to initialize counters and reset timer circuitry.

6.10 MEMORY TIMING SIGNALS

The three memory timing signals are READY, WAIT, and MEMCYC-. These are arbitrarily grouped together for a discussion of their theory of operation.

6.10.1 READY

The READY signal is an input to the TMS 9900 microprocessor which indicates that during a memory cycle, the memory devices addressed will be ready at the next ø1 clock phase for a successful disposition of data.

READY is generated by the TMS 4500 DRAM controller at its RDY output or it originates offboard at pin P1-90 (READY.B). The offboard signal is buffered at U7 by a 74LS244 line driver. The DRAM controller uses READY to place the microprocessor in a wait state during a memory refresh.

The READY signal is sampled by the processor during ø1, after MEMEN- has gone low. If READY is high when sampled, the 9900 CPU will continue the memory operation in progress as shown by the READ cycle part of Figure 6-14. During a read cycle if READY is sampled and found to be high, the processor will read data from the selected memory device(s) on the leading edge of the next ø1. During a write cycle, if READY is sampled on the leading edge of ø1 and found high, the CPU will assume that data has successfully been stored in the selected memory device(s) by the time the next leading edge of ø1 occurs. If the selected memory device(s) cannot meet this timing constraint, the READY signal can be pulled low, which puts the TMS 9900 CPU into a wait state. The WAIT signal will go high to signify that the processor is in a wait state, and CPU operations will be suspended until READY is sampled high. When READY goes high again, WAIT will drop and the CPU will continue execution from the point where it stopped. (Refer to the write cycle portion of Figure 6-14.)

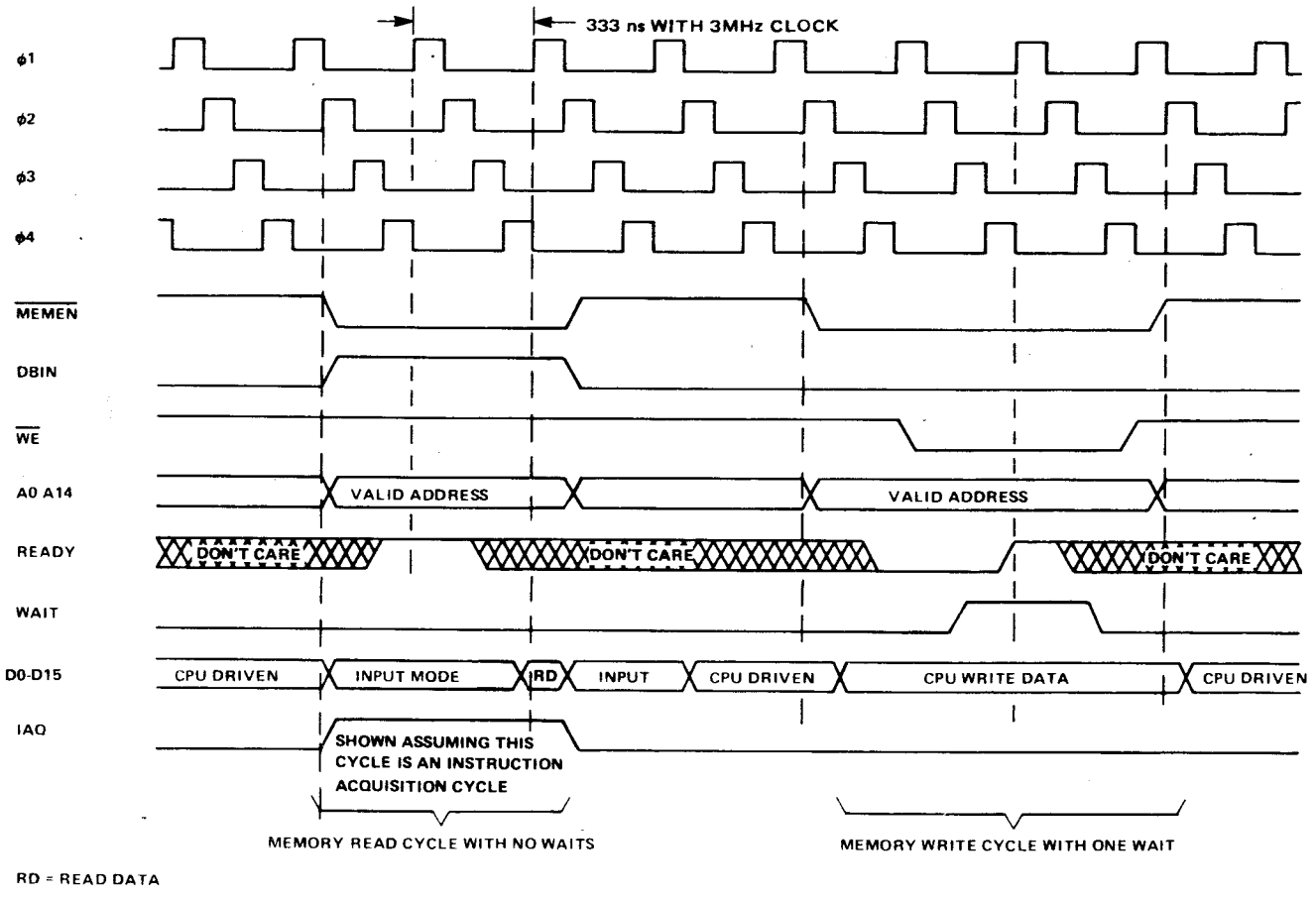


FIGURE 6-14. TMS 9900 MEMORY BUS TIMING

6.10.2 WAIT

The WAIT signal is output by the processor to acknowledge that addressed memory devices are not ready and that the processor is in a wait state. When the processor begins a memory cycle, it brings MEMEN- low and samples READY on the next ø1 clock cycle. If READY is not indicated (low), the processor enters a wait state and enables WAIT (high). The processor continues sampling READY at clock ø1. When a READY is indicated, the processor brings WAIT low and completes the memory cycle.

6.10.3 MEMCYC-

It is possible for the TMS 9900 microprocessor to activate MEMEN- and accomplish many fetches from memory by shifting the address bus, all while MEMEN- is still active. The MEMCYC- signal is synchronized to the ø3 clock edge after the beginning of the memory cycle, and goes inactive just before the instant the address bus could change. This signal thus delimits one complete memory cycle and differentiates between separate memory cycles.

The MEMCYC- signal is ORed with its offboard counterpart, MEMCYC.B-, to produce DMEMCYC-. This is to allow the earlier of the two signals to latch the TMS 4500. DMEMCYC- is used by the DRAM controller to synchronize the beginning of a memory access cycle.

6.11 CRU DEVICES

There are two CRU devices on the TM 990/102:

- TMS 9901 interrupt handler; LED DS2 controller
- TMS 9902 EIA asynchronous controller

Addresses of CRU devices on the TM 990/102 are shown in Table 6-5. CRU software and hardware base addresses are explained in section 7.7. The CRU software base address area of 0000_{16} to $013F_{16}$ is reserved for onboard use. Above this is considered offboard CRU addressing. The CRU address for the TMS 9901 and TMS 9902 are the same as for other TM 990 CPU boards.

CAUTION

Do not write to CRU hardware base addresses 0097_{16} to $009F_{16}$. These are unused I/O ports on the TMS 9901 that share external pins with interrupts INT7- to INT15- respectively. When written to via the CRU, these shared ports assume the output mode. If written to externally while in the output mode (such as if an external interrupt goes high), damage can result to the TMS 9901.

6.11.1 TMS 9901 as Interrupt Controller and LED Control

All maskable interrupts are monitored by the TMS 9901. In addition, the device controls LED DS2. Detailed data is provided in the TMS 9901 Programmable Systems Interface Data Manual, and examples of programming are shown in the TMS 9901 Programming section (Section 4) of this manual. A schematic of the TMS 9901 is shown in Figure 6-15. Dedicated interrupts are:

- RESET-: this is level zero interrupt, unmaskable (not on TMS 9901)
- INT3-: clock countdown from TMS 9901
- INT4-: interrupt from TMS 9902 (this is also called EIAINT-)
- INT1-, INT2-, INT5- to INT15-: user defined (INT1- usually power fail)

All interrupts except RESET (interrupt level zero) and LOAD are processed by the TMS 9901 Programmable Systems Interface device.

As shown in Table 6-5, the TMS 9901 occupies a 32-bit area of the CRU space (hardware base address 080_{16} to $09F_{16}$ or software base addresses 0100_{16} to $013F_{16}$). The first 16 bits mask and unmask the interrupt bits as well as set and trigger the interval timer. The 17th bit controls LED DS2. The remaining bits are not used. Do <u>not</u> write to hardware base addresses 091_{16} to $09F_{16}$ as the TMS 9901 can be damaged; see CAUTION above. A 74LS682 binary comparator compares address lines A3 to A9 for a CRU software base address of between 0100_{16} and $013F_{16}$ and MEMEN- disabled (<u>not</u> a memory address). When true, the TMS 9901s CE- is enabled. When the software base address is 0140_{16} or higher, external CRU is supposed (EXTCRU- is used to enable offboard CRU signals).

Write a one to TMS 9901 bit 17 to illuminate LED DS2. This one is inverted to provide a low to complete the LED circuitry. Bit 17 a zero turns off the LED.

The TMS 9901 internal bits are addressed via pins S0 (MSB) to S4. Data can be serially written to or read from the TMS 9901 on pins CRUOUT (out from CPU) and CRUIN respectively. Interrupts are received at the corresponding external pins (INT1- to INT15-) and the resulting binary value (1 to 15) is sent to the microprocessor via pins IC0 to IC3 (where it is received on similarly named pins). An INTREQ- signal to the microprocessor tells it that an interrupt is being requested.

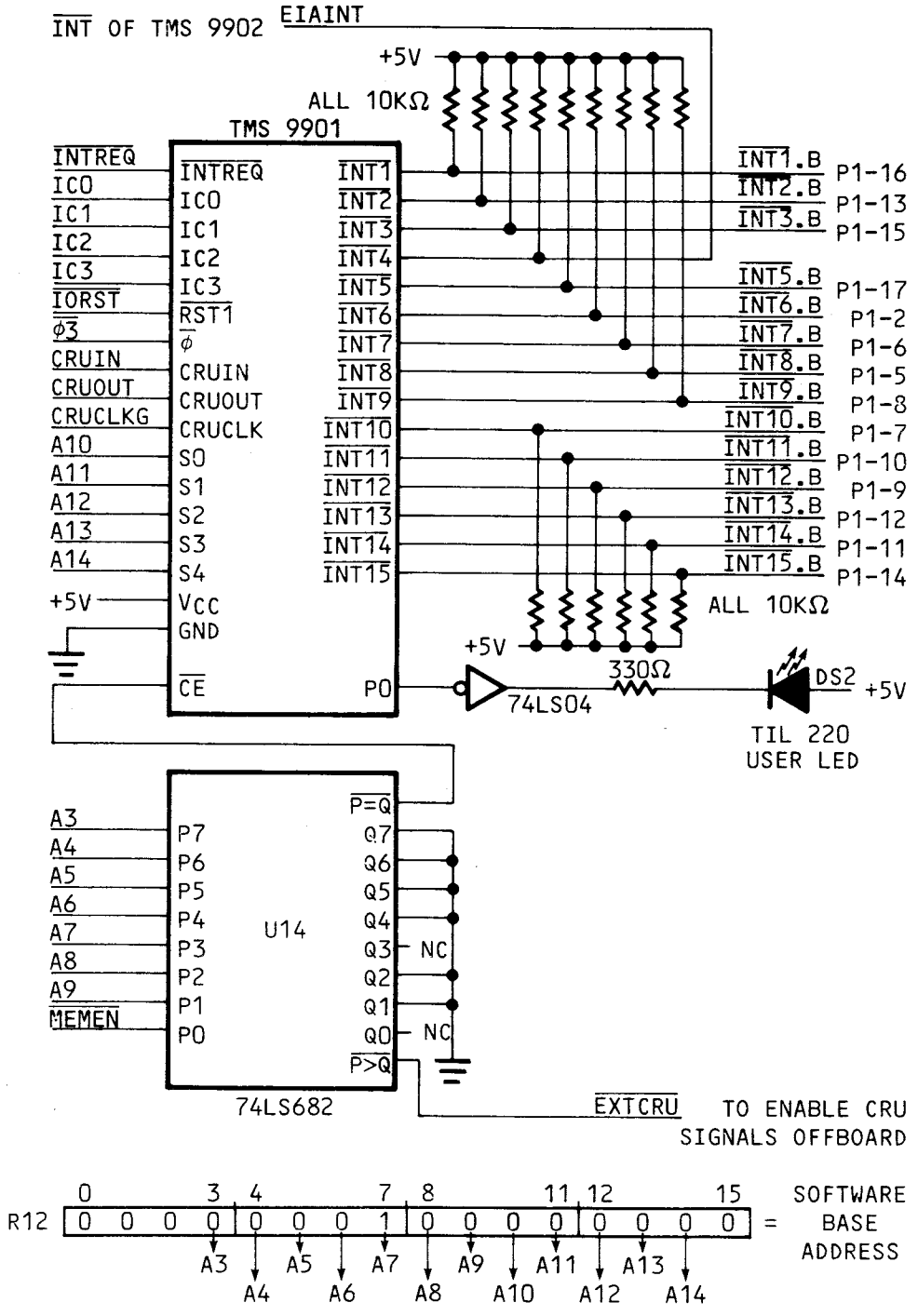


FIGURE 6-15. TMS 9901

TABLE 6-5. TM 990/102 CRU MAP

DEVICE/FUNCTION	SOFTWARE BASE ADDRESS (HEX)	HARDWARE BASE ADDRESS (HEX)	
Reserved	0000	000	
.	.	.	
.	.	.	
.	.	.	
Reserved	007E	03F	
TMS 9902	0080	040	
.	.	.	
.	.	.	
.	.	.	
TMS 9902	00BE	05F	
Reserved	00C0	060	
.	.	.	
.	.	.	
.	.	.	
Reserved	00FE	07F	
TMS 9901			
Control Bit	0100	080	
INT1-/CLK1	0102	081	
INT2-/CLK2	0104	082	
INT3-/CLK3	0106	083	
INT4-/CLK4	0108	084	
INT5-/CLK5	010A	085	
INT6-/CLK6	010C	086	
INT7-/CLK7	010E	087	
INT8-/CLK8	0110	088	
INT9-/CLK9	0112	089	
INT10-/CLK10	0114	08A	
INT11-/CLK11	0116	08B	
INT12-/CLK12	0118	08C	
INT13-/CLK13	011A	08D	
INT14-/CLK14	011C	08E	
INT15-/INTREQ	011E	08F	
P0 I/O, LED	0120	090	
P1 I/O	0122	091	Do not write to this address space
.	.	.	
.	.	.	
.	.	.	
P15 I/O	013E	09F	
Offboard CRU	0140	0A0	

As an interrupt controller, the TMS 9901 can be programmed to enable or disable any of the 15 maskable interrupts (INT1- to INT15-). The interrupt is enabled by first entering the interrupt mode by setting bit zero to a zero; then setting the desired interrupt mask bit to a one. This operation is done through the CRU. An enabled interrupt becomes active when:

- a low level is received externally, and
- no other interrupt of a higher priority is active (INT15- is the lowest priority and INT1- is the highest at the TMS 9901)

Note that if bit zero is a one, the clock mode is entered. The clock mode is covered in section 6.11.2.

The interrupt mask bits that contain zero will not honor interrupt requests. Note that the condition of the processor's Status Register priority mask is irrelevant if the TMS 9901's Interrupt Mask Register is a zero for a particular interrupt: the request will not even be presented to the processor. In summary, when an interrupt is enabled (unmasked or a one) at the TMS 9901 and becomes active (is the highest priority with its external input at a low level), a four-bit binary value of the interrupt level is sent to the TMS 9900s interrupt input lines IC0 to IC3. At the same time the TMS 9901 sends a low level signal via the INTREQ- input of the TMS 9900 (from the similarly named pin at the TMS 9901).

If the interrupt level is also unmasked at the microprocessor, the processor will obtain new WP and PC vectors from lower memory and transfer control to these values. (The processor's Status Register's four LSBs are used to unmask or mask off its interrupts; these bits are programmed with the LIMI instruction.) Part of the interrupt service routine would be to disable the answered interrupt. When the processor answers an interrupt request, it automatically sets its interrupt mask to one less than the current answered interrupt level and does not answer another interrupt request for one instruction cycle. Note that level 0 is the highest priority, and cannot be masked out since it is a number that is always equal to or lower than any number which can be in the mask register of the processor. The lowest priority is 15.

When one or more interrupt requests are presented to the TMS 9901, only those whose corresponding mask bits are one are considered. Interrupt requests are not latched for more than a clock cycle; instead, interrupt lines are sampled at each falling edge of clock ø3. At the falling edge of the next clock, the highest-priority request enabled (mask a one) present is encoded onto lines IC0 through IC3, and INTREQ- becomes active (low). This interrupt-level output to the processor will remain active until one of the following occurs at the TMS 9901:

- the corresponding external request is removed, or
- the corresponding interrupt is masked off (mask = 0), or
- a higher-priority interrupt request, properly enabled (mask = 1) becomes active.

When the highest priority interrupt becomes disabled at the TMS 9901 (this is usually by the interrupt service routine), the next-highest priority enabled interrupt becomes active. When all interrupt requests become inactive, then the interrupt logic lines to the processor become high (INTREQ-, IC0 to IC3). Bringing bit 15 low in the clock mode (bit 0 = 1 and bit 15 = 0), causes an RST2- which also resets these lines high.

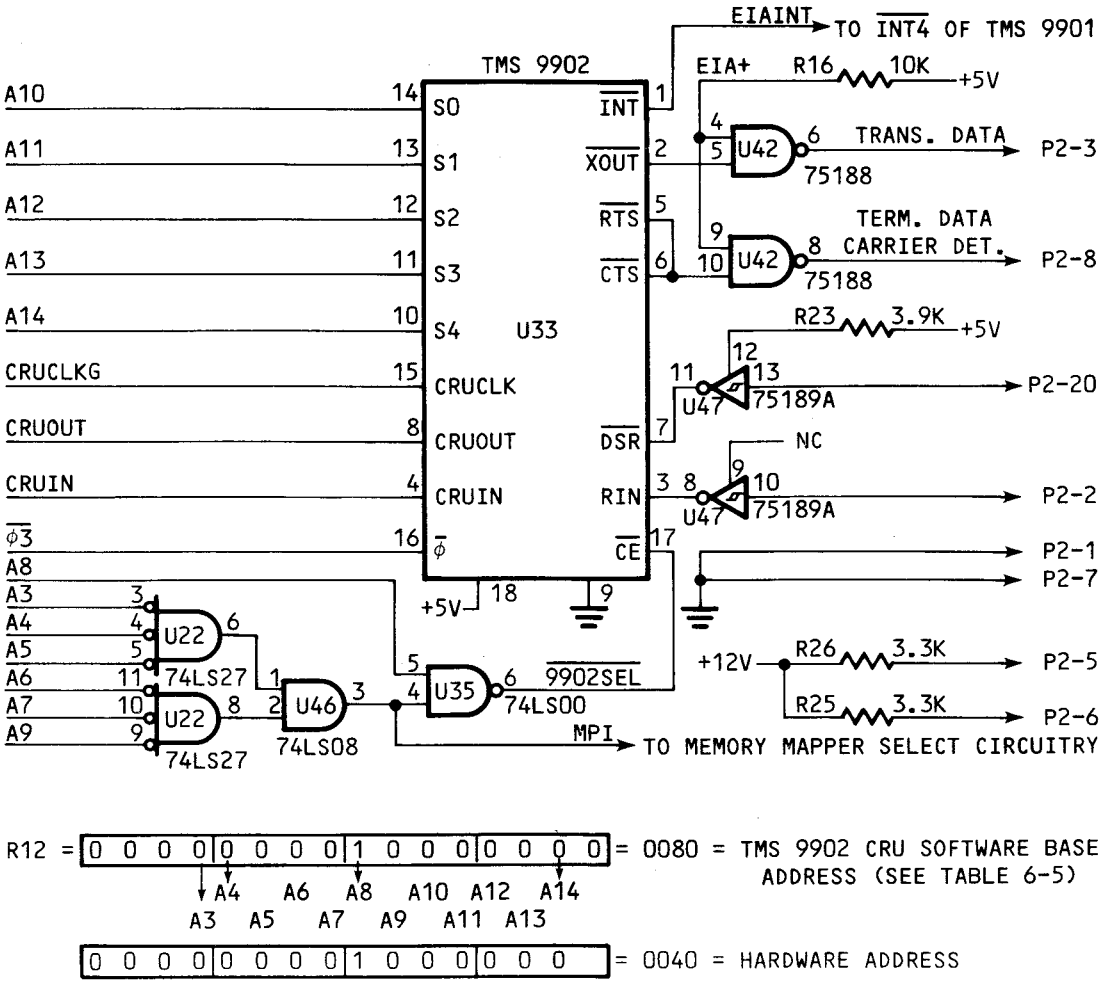


FIGURE 6-16. TMS 9902 AND SELECT CIRCUITRY

6.11.2 TMS 9901 as System Timer

The TMS 9901 has an internal real time clock which may be used as an interval timer by the user. It is a decrementer which generates an interrupt (level 3) when it decrements to 0. The counter starts counting when loaded with a non-zero quantity. Section 4 covers programming the TMS 9901.

6.11.3 TMS 9902 Communications Controller

The I/O port (P2) uses the TMS 9902 Asynchronous Communications Controller and is intended for operation with an EIA device. TTY devices are not supported by the TM 990/102. For detailed operation instructions for the TMS 9902, refer to Section 5 of this manual and to the TMS 9902 data manual. Note that the TMS 9902 INT- pin is connected to the INT4- pin of the TMS 9901 (signal EIAINT-). The TMS 9902 will generate an interrupt on four separate conditions; any of these will appear as INT4-. All four interrupt conditions can be enabled through programming via the CRU as explained in Section 5. Figure 6-16 shows TMS 9902 circuitry.

The EIA interface consists of 75188 line drivers and 75189A line receivers. The receive-data line goes to P2-2 and the transmit-data line to P2-3. This configuration forms a port suitable for connection to an RS-232-C compatible terminal. A data-terminal-ready (DTR) signal is supplied as an input for handshaking use with a device requiring it. Request-to-send (RTS) and clear-to-send (CTS) signals are tied together and brought out to P2-8, which functions as the data-carrier-detect (DCD) signal to the terminal.

The TMS 9902 32 bits are addressed by CRU instructions over A10 to A14 as shown in Figure 6-16. This figure also shows the chip select circuitry. A3 to A9 must be all zeroes except for A8 in order to effect the 0080_{16} software base address as also shown in Table 6-5. This is the value loaded into register 12. When applied to the address bus during a CRU operation, the chip select is enabled.

6.12 BUFFER CONTROL

Connector P1 is the system bus edge connector. It contains, in approximate order by pins: the system power, interrupt, data, address, and control signals. Appendix F lists pins and their functions. Power lines are discussed in section 6.2, and interrupts are detailed in section 6.11.1. This discussion covers the address bus buffers, the data bus buffers, control buffers, and a short discussion of HOLD, HOLDA, and direct memory access (DMA).

6.12.1 Address and Data Buffers

The address buffers consist of three 74LS245 octal bus transceivers. The address lines normally flow offboard. During DMA, the TM 990/102 processor is placed on hold via the offboard HOLD.B- signal; then its HOLDA (HOLD acknowledge) becomes active. Upon a HOLDA signal, the buffer directions reverse, allowing a DMA controller to input an address onto the board. Note that HOLDA disables the memory mapper, so the DMA device uses the absolute address instead of the logical address defined by the mapper chip.

The 74LS245s are used as data bus buffers. Direction data flow, however, is governed by the states of DBIN and HOLDA. When HOLDA is enabled (offboard DMA taking place), DBIN and HOLDA- to EXCLUSIVE-OR gate at U34 causes CDBIN high to turn the data buffers inward and DBIN low turns the data buffers outward. With the TM 990/102 accessing memory offboard, the opposite is true.

The data buffers (U4 and U15 on schematic page 3) are enabled by the following logic:

- MEMCYC- active low and offboard memory selected,
  or
- on a DMA access via the bus, MEMCYC- active low and onboard memory selected.

6.12.2 HOLD-, HOLDA, and DMA

Note that HOLD.B cannot be enabled by the TM 990/102; thus, DMA will be by an external DMA processor only.

When an offboard direct memory access controller (DMAC) wishes to initiate operation, it asserts a low state onto the HOLD.B- line. After finishing the current memory cycle, the microprocessor responds by placing into high

impedance its address, data, MEMEN-, DBIN, and WE- lines, and then forces HOLDA (HOLD acknowledge) high. In turn, HOLDA causes the address buffers to be directed onboard (inward) and the data buffers to be inward when offboard DBIN.B is low (DMA processor write) and outward when DBIN.B is high (DMA processor read). Control signals from the DMA processor are similarly controlled by HOLDA. HOLD.B and READY.B are dampened by terminating resistors to both Vcc and ground.

The DMAC is now free to use the system buses to transfer data directly in and out of memory as it wishes. Control of the data and address buses by the DMAC is explained in Sections 6.12.1 and 6.12.3.

6.12.3 Control Buffers

Control buffers are two 74LS245s at U3 and U7 as well as half of the 74LS245 at U18 that also buffers the three LSB address lines. These buffers control the following:

		Direction From
• U18:	MEMEN.B-	As required
	WE.B-	As required
	DBIN.B	As required
	A14.B	As required
	A13.B	As required
	A12.B	As required
• U7 :	HOLD.B-	Offboard
	HOLDA.B	Onboard
	IORST.B-	Onboard
	MEMCYC.B-	Onboard
	READY.B	Offboard
	CRUCLK.B	Onboard
• U3 :	CRUOUT.B	Offboard
	CRUIN.B	Onboard
	CLK.B	Onboard
	03.B-	Onboard
	01.B-	Onboard
	IAQ.B	Onboard

Buffer U18 (its lines are listed above) is enabled as long as HOLD- is not requested AND a HOLDA (hold acknowledge) is not active at the same time. When enabled, direction is controlled by HOLDA. When HOLDA is active (high), DMA is underway and buffer direction is to the board. When HOLDA is inactive, control is onboard and the direction for these signals is towards offboard.

At the 74LS244 buffer at U3, CRUOUT.B and CRUIN.B are enabled only when signal EXTCRU- is active (meaning operation is not onboard). At U7, READY.B and CRUCLK.B are enabled only when signal HOLDA is low (not in hold state).

SECTION 7

TM 990/102 INSTRUCTION SET

7.1 GENERAL

This section provides a general discussion of the instruction set used with the TM 990/102, including both assembly language and machine language. This instruction set is compatible with other members of the 990 family.

Other topics include:

- Hardware and software registers (sections 7.3 and 7.4)
- CRU addressing (section 7.7)

The TM 990/102 microcomputer is designed for use by a variety of users with varying technical backgrounds. Because a TM 990/102 user has the capability of writing his programs in machine language and entering them into memory using the TM 990/404 monitor, emphasis is on binary/ hexadecimal representations of assembly language statements. The assembly language described herein can be assembled on a 990 family assembler. If an assembler is used, this section assumes that the user will be aware of all prerequisites for using the particular assembler.

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).

Further information on the 990 assembly language is provided in the Model 990 Computer/TMS 9900 Microprocessor Assembly Language Programmer's Guide (P/N 943441-9701).

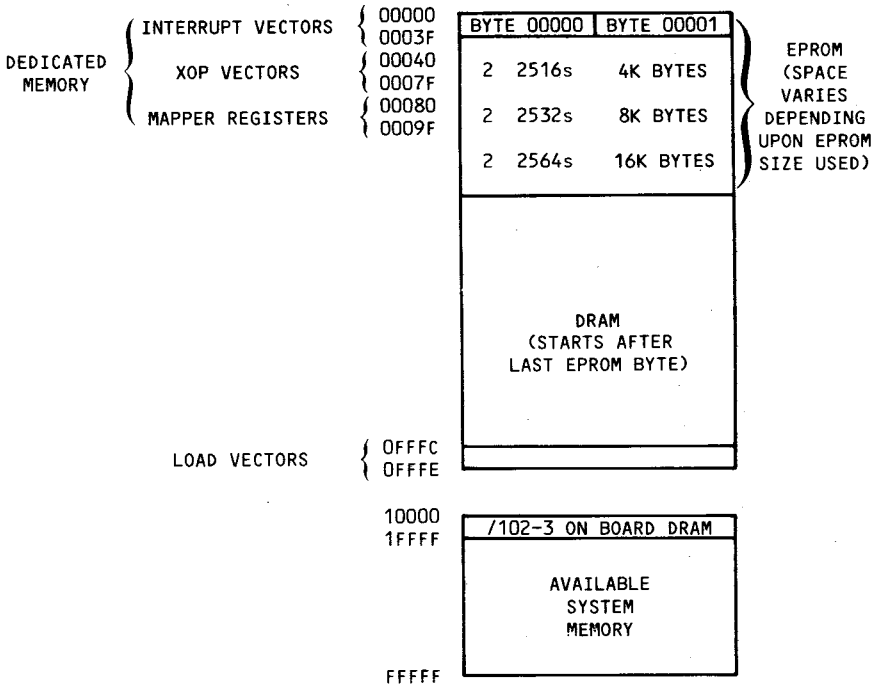
7.2 USER MEMORY

Figure 7-1 shows the user RAM space in memory available onboard for execution of user programs. Note that the memory address value is the number of bytes beginning at 0000; thus, all word addresses are even values from 0000_{16} to $FFFE_{16}$ for a -2 model and 0000_{16} to $1FFFE_{16}$ for a -3 model.

Programs in EPROMs can be read by the processor and executed; however, EPROM memory cannot be modified (written to). Therefore workspace register areas are in RAM where their values can be modified. RESTART vectors utilize the two words of DRAM memory at $FFFC_{16}$ and $FFFE_{16}$ as shown in Figure 7-1.

7.3 HARDWARE REGISTERS

The TM 990/102 uses three major hardware registers in executing the instruction set: Program Counter, Workspace Pointer, and Status Register.



DEDICATED MEMORY

ADDRESS (HEX)	PURPOSE
00000 - 0003F	INTERRUPT VECTORS
00040 - 0007F	XOP VECTORS
00080 - 0009F	MAPPER REGISTERS*
0FFFC - 0FFFE	RESTART (LOAD) VECTORS

* TRUE ONLY IN MAPPER-REGISTER MODE ONLY. IN OTHER MAPPER MODES, THIS IS SYSTEM MEMORY.

BOARD MEMORY MAP BY DASH NUMBER

EPROM TYPE	TM 990/102-1		TM 990/102-2		TM 990/102-3	
	EPROM	RAM	EPROM	RAM	EPROM	RAM
NO EPROM	-	-	-	00000-0FFFF	-	00000-1FFFF
2 2516s	00000-00FFF	-	00000-0FFFF	01000-0FFFF	00000-00FFF	01000-1FFFF
2 2532s	00000-01FFF	-	00000-01FFF	02000-0FFFF	00000-01FFF	02000-1FFFF
2 2564s	00000-03FFF	-	00000-03FFF	04000-0FFFF	00000-03FFF	04000-1FFFF

FIGURE 7-1. MEMORY MAP

7.3.1 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 (or more depending upon instruction format) so that it "points" to the next sequential instruction.

7.3.2 Workspace Pointer (WP)

This register contains the beginning memory address 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. Section 7.4 explains a workspace in detail.

7.3.3 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 7-2.

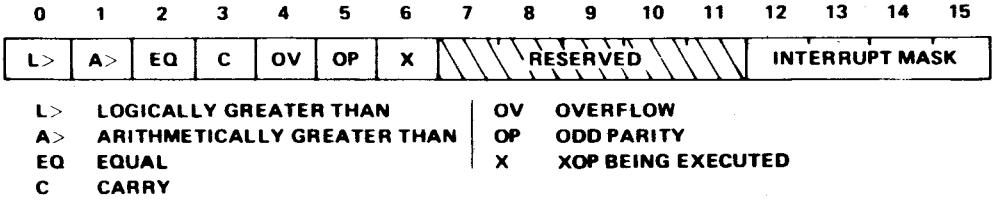


FIGURE 7-2. STATUS REGISTER

7.3.3.1 Logical Greater Than

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

7.3.3.2 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.

7.3.3.3 Equal

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

7.3.3.4 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 operations to store the value of the last bit shifted out of the workspace register being shifted.

7.3.3.5 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 MSBs 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 MSBs 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}.

7.3.3.6 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.

7.3.3.7 Extended Operation

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

7.3.3.8 Status Bit Summary

Table 7-1 lists the instruction set and the status bits affected by each instruction.

7.4 SOFTWARE REGISTERS

Registers used by programs are contained in memory. This speeds up context-switch time because only the contents of the three hardware registers (WP, PC, and ST registers) needs to be saved.

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 7-3 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 7-1. 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.

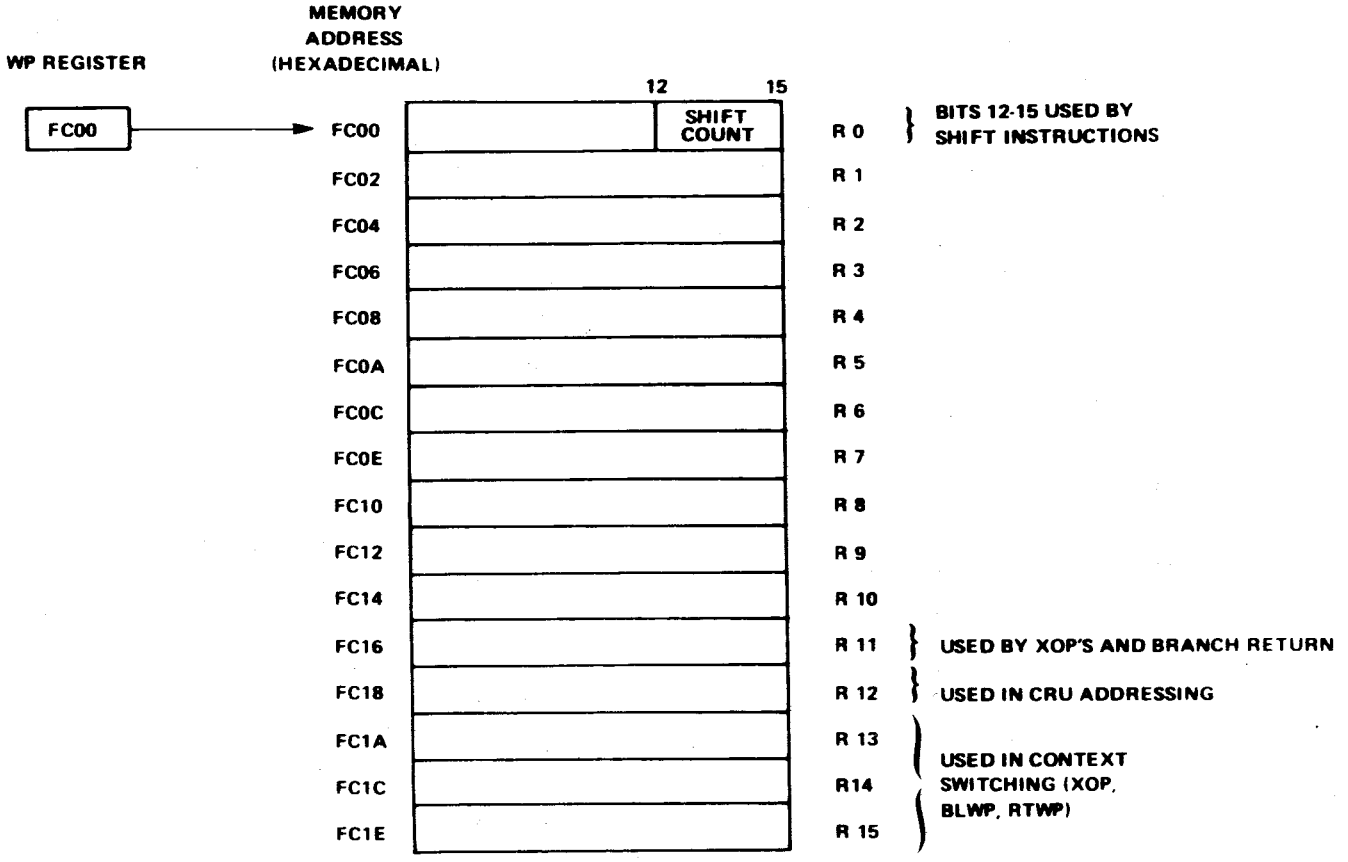


FIGURE 7-3. WORKSPACE EXAMPLE

7.5 INSTRUCTION FORMATS AND ADDRESSING MODES

The instructions used by the TM 990/102 are contained in 16-bit memory words and require one, two, or three words for full definition. The first word (or the single word) of an instruction will describe the purpose of the instruction while the succeeding one or two words will be numbers that are referenced by the initial instruction word. A word describing an instruction is interpreted by the Central Processing Unit (CPU) by decoding the various fields within the 16 bits. These fields are shown in Figure 7-4 for the 9900 instruction set which is also categorized into nine instruction formats as shown in the figure.

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.

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

OP CODE	OPERATION CODE
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	PROGRAM COUNTER RELATIVE, NOT INDEXED (SR OR DR = 0) PROGRAM COUNTER RELATIVE + INDEX REGISTER (SR OR DR>0)
11	INDIRECT REGISTER, AUTOINCREMENT REGISTER

FIGURE 7-4. TM 990/102 INSTRUCTION FORMATS

The fields within an instruction word contain the following information (see Figure 7-4):

- Op code which identifies the desired operation to be accomplished when this instruction is executed.

- B code which identifies whether the instruction will affect a full 16-bit word in memory or an 8-bit byte. A one indicates a byte will be affected, while a zero indicates a word will be affected.

- T fields identified by TD for the destination T field and TS for the source T field. The T field is a two-bit code which identifies which of five different addressing modes will be used (direct register, indirect register, memory address, memory address indexed, and indirect register autoincremented). These modes are described in detail in sections 7.5.1 through 7.5.5. The source T field is the code for the source address and the destination T field is the code for the destination address. As shown in Figure 7-4, only five instruction formats use a T field.

- Source and destination register fields which contain the number of the register affected (0 through 15).

- Displacement fields that contain a bias to be added to the program counter in program counter relative addressing. This form of addressing is further described in section 7.5.7.

- Fields that contain counts for indicating the number of bits that will be shifted in a shift instruction or the number of Communication Register Unit (CRU) bits that will be addressed in a CRU instruction.

7.5.1 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 7-5, 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 (a byte instruction addressing a register would address the most significant 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 T field and register number of the destination register first.

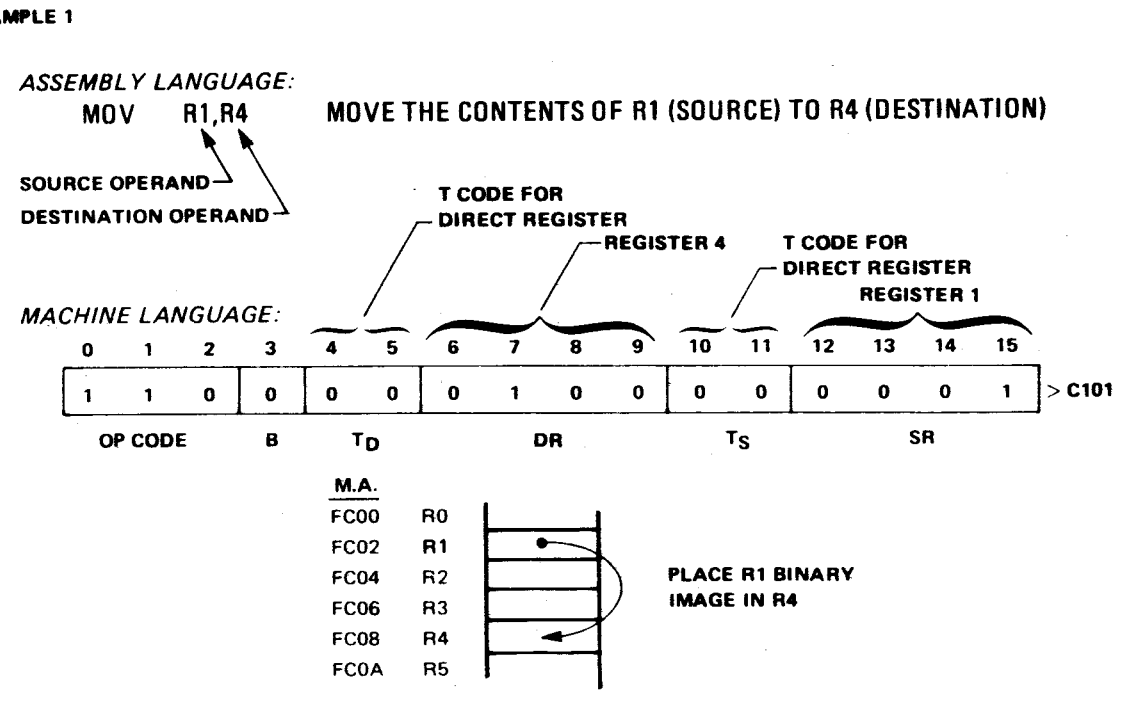
7.5.2 Indirect Register Addressing ($T=01_2$)

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 7-6 specifies to move the contents of register 1 to the address which is contained in register 4 (indirect register 4). Instead of moving the value in

register 1 to register 4 as was the case in Figure 7-5, 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 $FD00_{16}$. This instruction stores the value in register 1 into memory address (M.A.) $FD00_{16}$.

Indirect register addressing is specified in assembly language source code by preceding the register number with an asterisk (*). For example, A *R1,*R2 means to add the contents of the memory address in register 1 to the contents of the memory address in register 2, leaving the sum in the memory address contained in register 2.

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



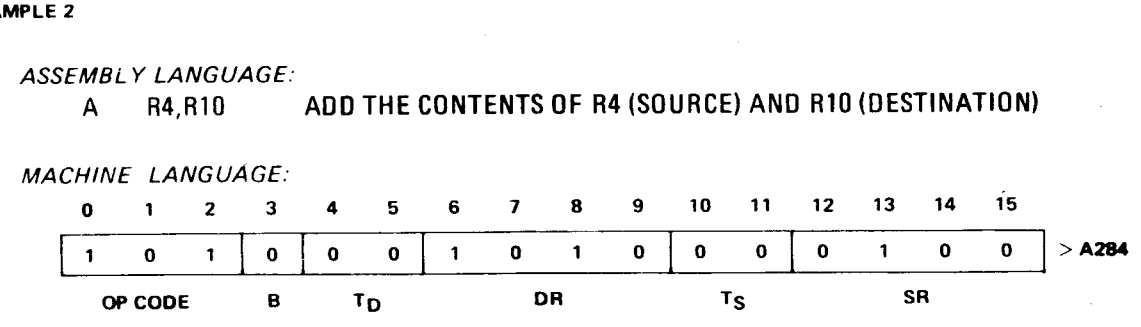


FIGURE 7-5. DIRECT REGISTER ADDRESSING EXAMPLES

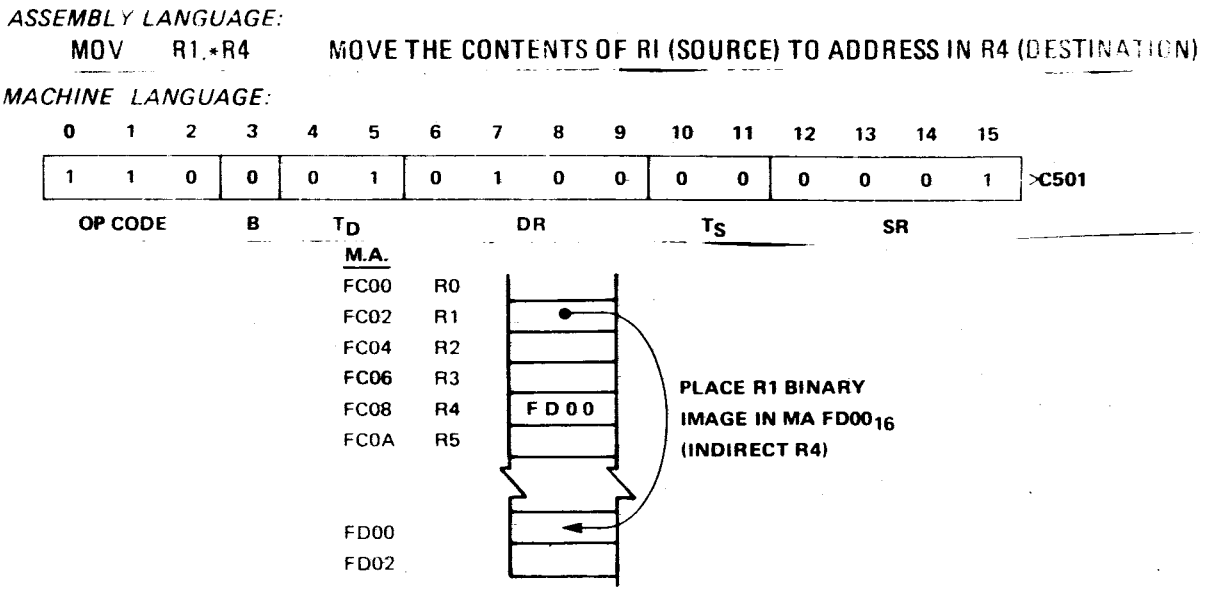


FIGURE 7-6. INDIRECT REGISTER ADDRESSING EXAMPLE

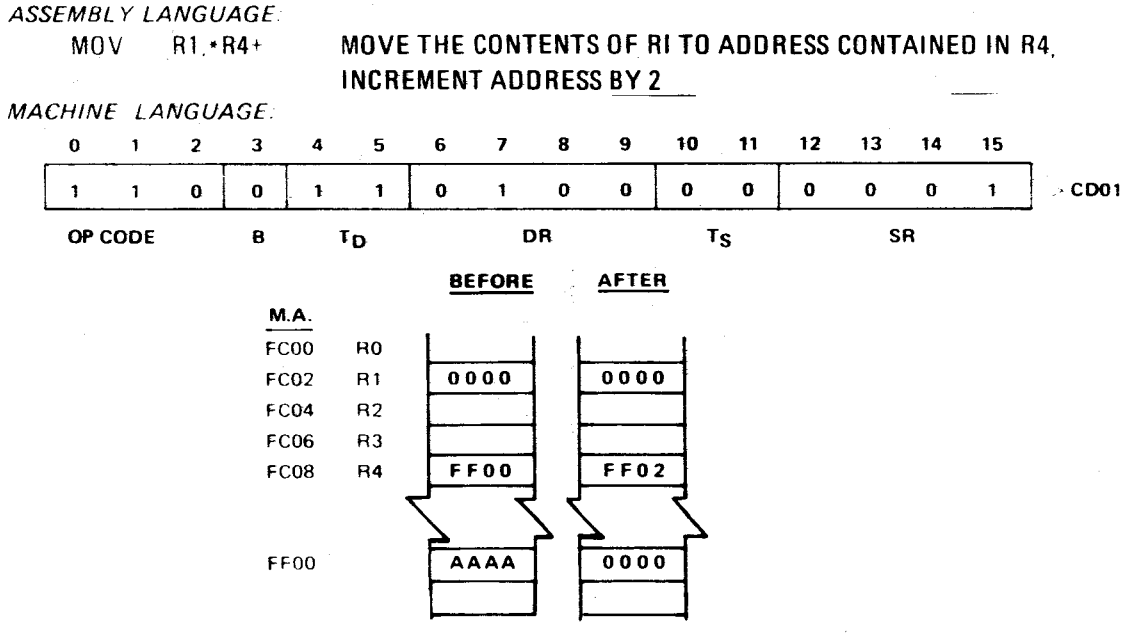


FIGURE 7-7. INDIRECT REGISTER AUTOINCREMENT ADDRESSING EXAMPLE

execute an 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.

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

Indirect register autoincrement addressing is the same as indirect register addressing (section 7.5.2) 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 7-7. 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.

7.5.4 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 7-8. When the T field contains 10_2, the CPU retrieves the contents of the next 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. Numerical values preceded by an @ sign will be assembled as an absolute address.

In the second example in Figure 7-8, 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.

7.5.5 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 7-9, 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 $FF02_{16}$ value in the source memory address

word plus the value in the index register (0004_{16}). The effective address in this case is $FF06_{16}$ as shown by the addition in the left part of the figure.

Note that only symbolic addressing can be indexed.

EXAMPLE 1

ASSEMBLY LANGUAGE:

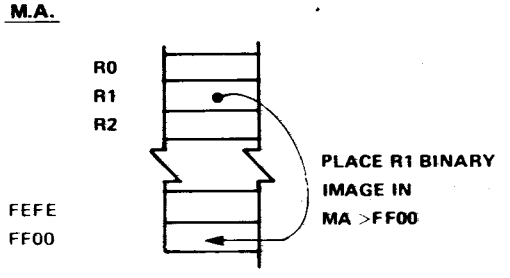
```
MOV    R1,@>FF00        MOVE THE CONTENTS OF R1 TO ADDRESS >FF00
```

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	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	>FF00



EXAMPLE 2

ASSEMBLY LANGUAGE:

```
MOV    @>FF0A,@>FF08        MOVE THE CONTENTS OF >FF0A TO >FF08
```

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	1	1	1	1	1	1	1	1	0	0	0	0	1	0	1	0	>FF0A (SOURCE)
3rd WORD	1	1	1	1	1	1	1	1	0	0	0	0	1	0	0	0	>FF08 (DESTINATION

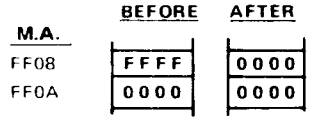


FIGURE 7-8. DIRECT MEMORY ADDRESSING EXAMPLES

ASSEMBLY LANGUAGE:

```
MOV    @>FF00,@>FF02(R1)      MOVE THE CONTENTS OF >FF00 TO >FF02 + 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
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	>FF00 (SOURCE)
1	1	1	1	1	1	1	1	0	0	0	0	0	0	1	0	>FF02 (DESTINATION

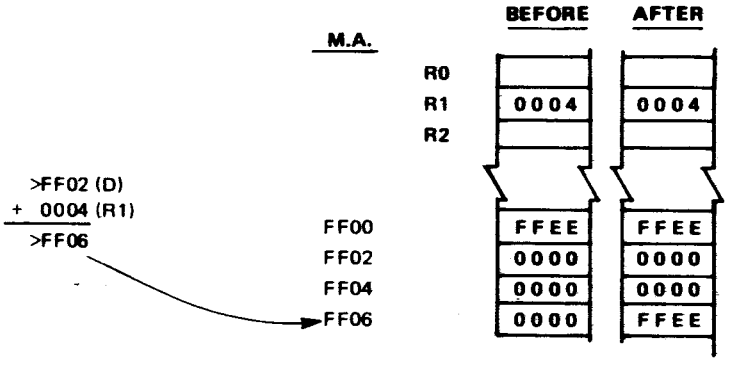


FIGURE 7-9. DIRECT MEMORY ADDRESSING, INDEXED EXAMPLE

7.5.6 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
```

7.5.7 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 (PC + 4)
JMP   >3E26        JUMP TO M.A. >3E26
```

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

7.6 INSTRUCTIONS

Table 7-2 lists terms used in describing the instructions of the TM 990/102. Table 7-3 is an alphabetical list of instructions. Table 7-4 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 7-2. INSTRUCTION DESCRIPTION TERMS

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

TABLE 7-3. INSTRUCTION SET, ALPHABETICAL INDEX

ASSEMBLY LANGUAGE MNEMONIC	MACHINE LANGUAGE OP CODE	FORMAT	STATUS REG. BITS AFFECTED	RESULT COMPARED TO ZERO	INSTRUCTION	PARAGRAPH
A	A000	1	0-4	X	Add (word)	7.6.1
AB	B000	1	0-5	X	Add (byte)	7.6.1
ABS	0740	6	0-2	X	Absolute Value	7.6.6
AI	0220	8	0-4	X	Add Immediate	7.6.8
ANDI	0240	8	0-2	X	AND Immediate	7.6.8
B	0440	6	–		Branch	7.6.6
BL	0680	6	–		Branch and Link (R11)	7.6.6
BLWP	0400	6	–		Branch; New Workspace Pointer	7.6.6
C	8000	1	0-2		Compare (word)	7.6.1
CB	9000	1	0-2,5		Compare (byte)	7.6.1
CI	0280	8	0-2		Compare Immediate	7.6.8
CKOF	03C0	7	–		User Defined	7.6.7
CKON	03A0	7	–		User Defined	7.6.7
CLR	04C0	6	–		Clear Operand	7.6.6
COC	2000	3	2		Compare Ones Corresponding	7.6.3
CZC	2400	3	2		Compare Zeroes Corresponding	7.6.3
DEC	0600	6	0-4	X	Decrement (by one)	7.6.6
DECT	0640	6	0-4	X	Decrement (by two)	7.6.6
DIV	3C00	9	4		Divide	7.6.3
IDLE	0340	7	–		Computer Idle	7.6.7
INC	0580	6	0-4	X	Increment (by one)	7.6.6
INCT	05C0	6	0-4	X	Increment (by two)	7.6.6
INV	0540	6	0-2	X	Invert (One's Complement)	7.6.6
JEQ	1300	2	–		Jump Equal (ST2=1)	7.6.2
JGT	1500	2	–		Jump Greater Than (ST1=1), Arithmetic	7.6.2
JH	1B00	2	–		Jump High (ST0=1 and ST2=0), Logical	7.6.2
JHE	1400	2	–		Jump High or Equal (ST0 or ST2=1), Logical	7.6.2
JL	1A00	2	–		Jump Low (ST0 and ST2=0), Logical	7.6.2
JLE	1200	2	–		Jump Low or Equal (ST0=0 or ST2=1), Logical	7.6.2
JLT	1100	2	–		Jump Less Than (ST1 and ST2=0), Arithmetic	7.6.2
JMP	1000	2	–		Jump Unconditional	7.6.2
JNC	1700	2	–		Jump No Carry (ST3=0)	7.6.2
JNE	1600	2	–		Jump Not Equal (ST2=0)	7.6.2
JNO	1900	2	–		Jump No Overflow (ST4=0)	7.6.2
JOC	1800	2	–		Jump On Carry (ST3=1)	7.6.2

TABLE 7-3. INSTRUCTION SET, ALPHABETICAL INDEX (CONCLUDED)

ASSEMBLY LANGUAGE MNEMONIC	MACHINE LANGUAGE OP CODE	FORMAT	STATUS REG. BITS AFFECTED	RESULT COMPARED TO ZERO	INSTRUCTION	PARAGRAPH
JOP	1C00	2	–		Jump Odd Parity (ST5=1)	7.6.2
LDCR	3000	4	0-2,5	X	Load CRU	7.6.4
LI	0200	8	–	X	Load Immediate	7.6.8
LIMI	0300	8	12-15		Load Interrupt Mask Immediate	7.6.8
LREX	03E0	7	12-15		Load and Execute	7.6.7
LWPI	02E0	8	–		Load Immediate to Workspace Pointer	7.6.8
MOV	C000	1	0-2	X	Move (word)	7.6.1
MOVB	D000	1	0-2,5	X	Move (byte)	7.6.1
MPY	3800	9	–		Multiply	7.6.3
NEG	0500	6	0-2	X	Negate (Two's Complement)	7.6.6
ORI	0260	8	0-2	X	OR Immediate	7.6.8
RSET	0360	7	12-15		Reset AU	7.6.7
RTWP	0380	7	0-15		Return from Context Switch	7.6.7
S	6000	1	0-4	X	Subtract (word)	7.6.1
SB	7000	1	0-5	X	Subtract (byte)	7.6.1
SBO	1D00	2	–		Set CRU Bit to One	7.6.2
SBZ	1E00	2	–		Set CRU Bit to Zero	7.6.2
SETO	0700	6	–		Set Ones	7.6.6
SLA	0A00	5	0-4	X	Shift Left Arithmetic	7.6.5
SOC	E000	1	0-2	X	Set Ones Corresponding (word)	7.6.1
SOCB	F000	1	0-2,5	X	Set Ones Corresponding (byte)	7.6.1
SRA	0800	5	0-3	X	Shift Right (sign extended)	7.6.5
SRC	0B00	5	0-3	X	Shift Right Circular	7.6.5
SRL	0900	5	0-3	X	Shift Right Logical	7.6.5
STCR	3400	4	0-2,5	X	Store From CRU	7.6.4
STST	02C0	8	–		Store Status Register	7.6.8
STWP	02A0	8	–		Store Workspace Pointer	7.6.8
SWPB	06C0	6	–		Swap Bytes	7.6.6
SZC	4000	1	0-2	X	Set Zeroes Corresponding (word)	7.6.1
SZCB	5000	1	0-2,5	X	Set Zeroes Corresponding (byte)	7.6.1
TB	1F00	2	2		Test CRU Bit	7.6.2
X	0480	6	–		Execute	7.6.6
XOP	2C00	9	6		Extended Operation	7.6.9
XOR	2800	3	0-2	X	Exclusive OR	7.6.3

TABLE 7-4. 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

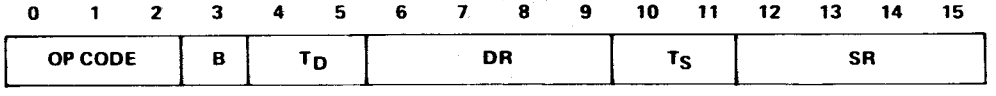
TABLE 7-4. INSTRUCTION SET, NUMERICAL INDEX (Concluded)

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	6
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

7.6.1 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

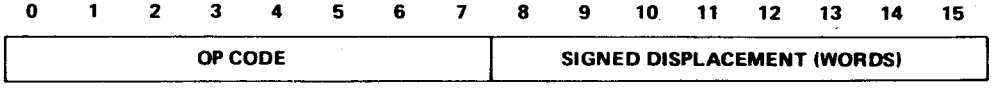
In byte instruction designating a register, the left byte is used. In the above example, the left byte (8 MSB's) of R1 is compared to the left byte of R2, and the ST set to the results.

7.6.2 Format 2 Instructions

7.6.2.1 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

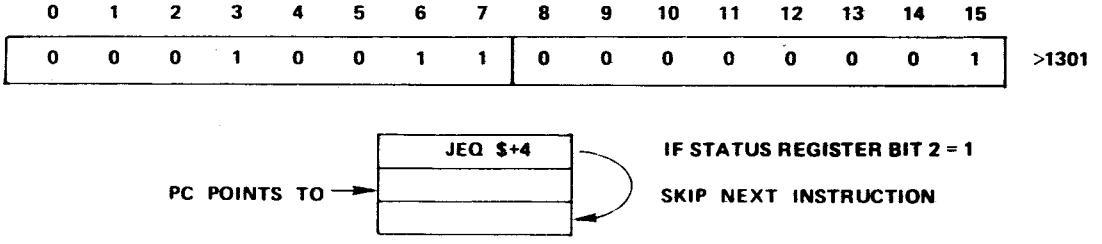
In assembly language, $ in the operand indicates "at this instruction". Essentially JMP $ causes an unconditional loop to the same instruction location, and JMP $+2 is essentially a no-op ($+2 means "here plus two bytes"). Note that the number following the $ is a byte count while displacement in machine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

EXAMPLES:

(1) ASSEMBLY LANGUAGE:

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

MACHINE LANGUAGE:

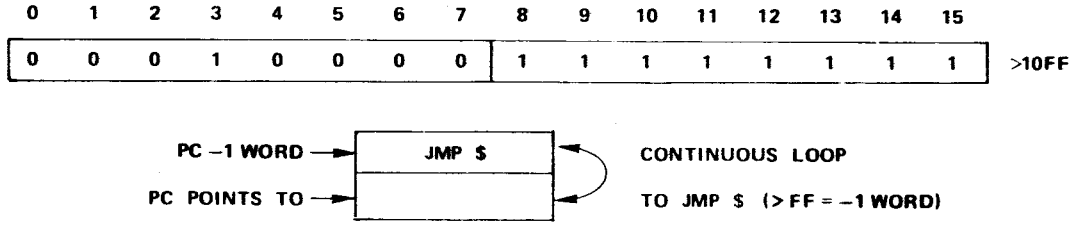


The above instruction continues execution 4 bytes (2 words) from the instruction location or, in other words, two bytes (one word) from the Program Counter value (incremented by 2 and now pointing to next instruction while JEQ executes). Thus, the signed displacement of 1 word (2 bytes) is the value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

7.6.2.2 CRU Single-Bit Instructions

These instructions test or set values at the CRU. The CRU bit is selected by the CRU address in bits 3 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. CRU addressing is discussed in detail in section 5.2. CRU multibit instructions are defined in section 7.6.4.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
General Format:	OP CODE								SIGNED DISPLACEMENT							

MNEMONIC	OP CODE 0 1 2 3 4 5 6 7	MEANING	STATUS BITS AFFECTED	DESCRIPTION
SBO	0 0 0 1 1 1 1 0	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 3 TO 14 = >100

ASSEMBLY LANGUAGE:

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

MACHINE LANGUAGE:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	0	1	1	1	0	1	0	0	0	0	0	1	0	0	>1D04

7.6.3 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 section 7.6.9.

	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 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 (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⊕0 = 1; 0⊕0 = 0; 1⊕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    @>FE00,R5    DIVIDE CONTENTS OF R5 AND R6 BY VALUE AT M.A. > FE00
```

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
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	>FE00

	BEFORE	AFTER	
M.A. > FE00	0005	0005	
R5	0000	0003	
R6	0011	0002	← REMAINDER

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.

7.6.4 Format 4 (CRU Multibit) Instructions

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
General Format:	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 3 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 WR 12 are not affected. Ts and SR 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.

NOTE

CRU addressing is discussed in detail in section 5.2. CRU single bit instructions are defined in section 7.6.2.2

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 $1 \leq C \leq 8$.

EXAMPLE

ASSEMBLY LANGUAGE:

```
LDCR   @>FE00,8      LOAD 8 BITS ON CRU FROM M.A. >FE00
```

MACHINE LANGUAGE:

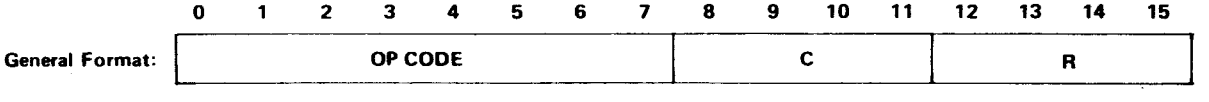
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0	0	1	1	0	0	1	0	0	0	1	0	0	0	0	0	>3220
1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	>FE00

NOTE

CRU addressing is discussed in detail in section 5.2.

7.6.5 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.

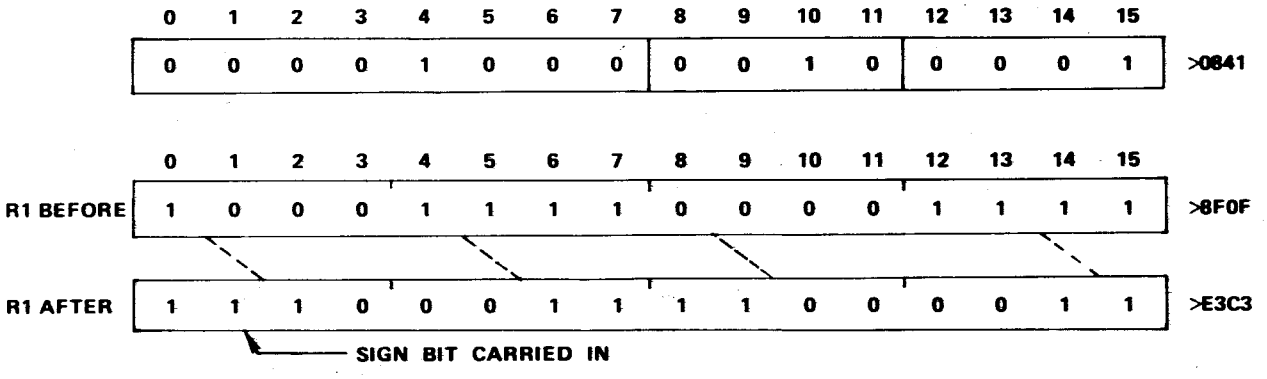
MNEMONIC	OP CODE 0 1 2 3 4 5 6 7	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
SLA	0 0 0 0 1 0 1 0	Shift left arithmetic	Yes	0-4	Shift (R) left. Fill vacated bit positions with 0.
SRA	0 0 0 0 1 0 0 0	Shift right arithmetic	Yes	0-3	Shift (R) right. Fill vacated bit positions with original 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

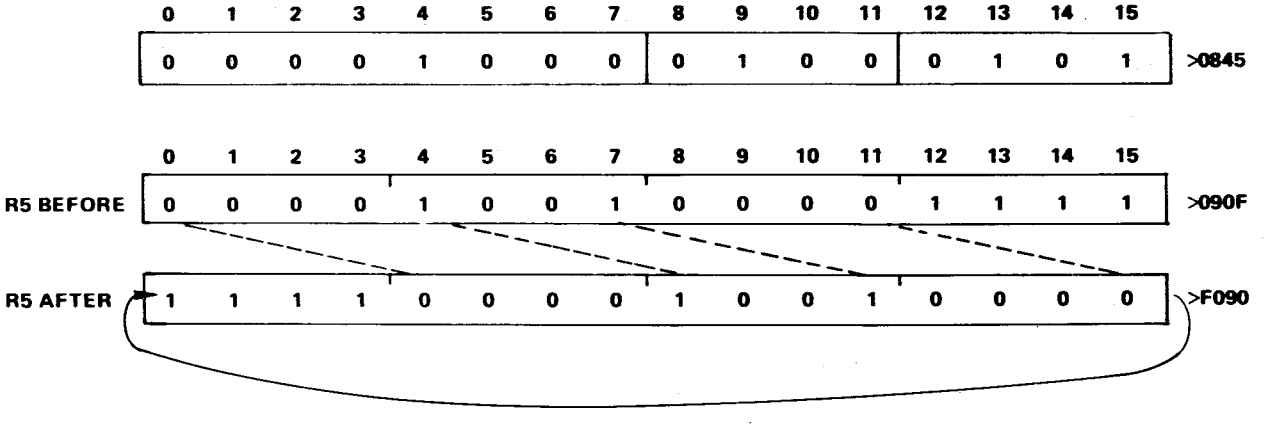
MACHINE LANGUAGE:



(2) ASSEMBLY LANGUAGE:

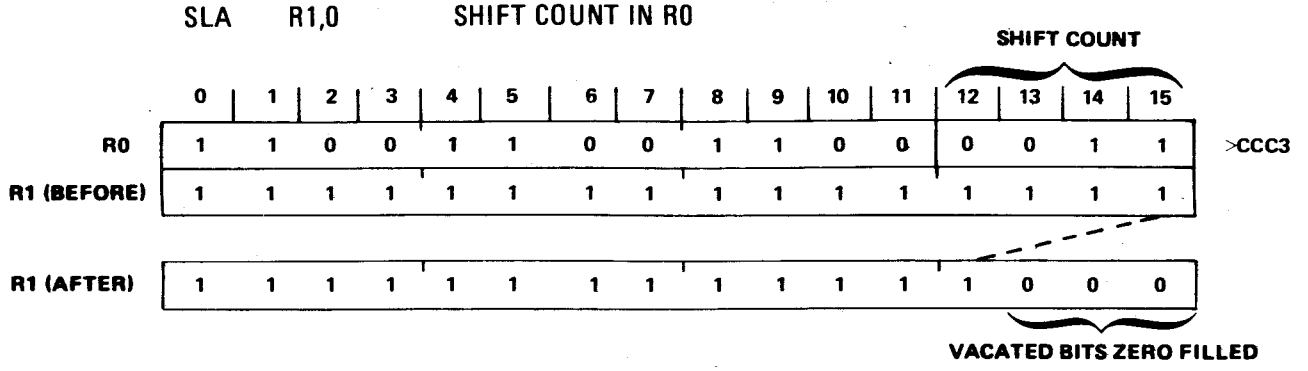
SRC R5,4 CIRCULAR SHIFT R5 4 POSITIONS

MACHINE LANGUAGE:



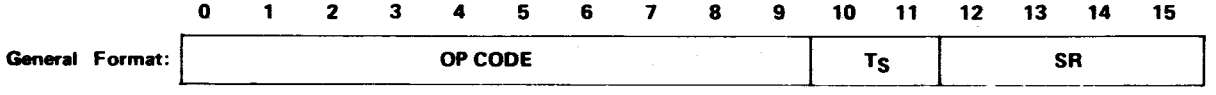
(3) ASSEMBLY LANGUAGE:

SLA R1,0 SHIFT COUNT IN R0



7.6.6 Format 6 Instructions

These are single operand instructions.



The Ts 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.

NOTE

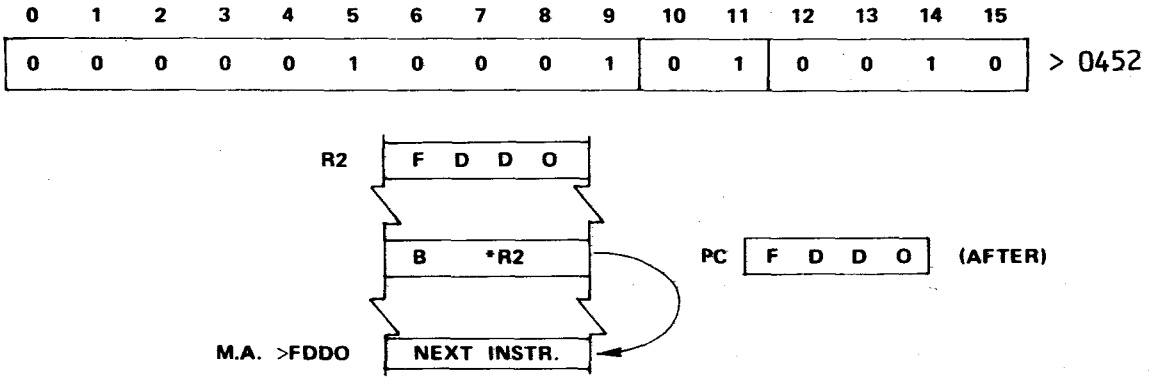
Jumps, branches, and XOP's are compared in Table 7-5.

EXAMPLES

(1) ASSEMBLY LANGUAGE:

B *R2 BRANCH TO M.A. IN R2

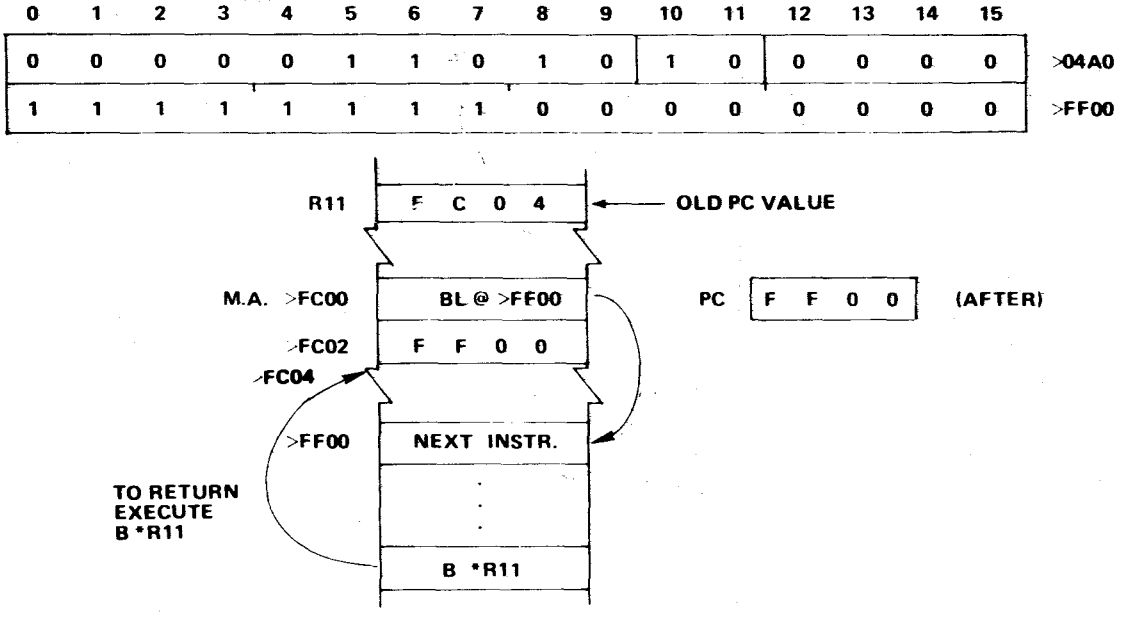
MACHINE LANGUAGE:



(2) ASSEMBLY LANGUAGE:

BL @>FF00 BRANCH TO M.A. >FF00, SAVE OLD PC VALUE (AFTER EXECUTION) IN R11

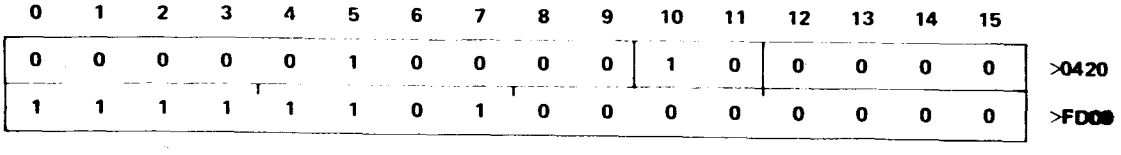
MACHINE LANGUAGE:



(3) ASSEMBLY LANGUAGE:

BLWP @>FD00 BRANCH, GET NEW WORKSPACE AREA

MACHINE LANGUAGE:



This context switch provides a new workspace register file and stores return values in the new workspace. See Figure 7-10. The operand (>FD00 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.

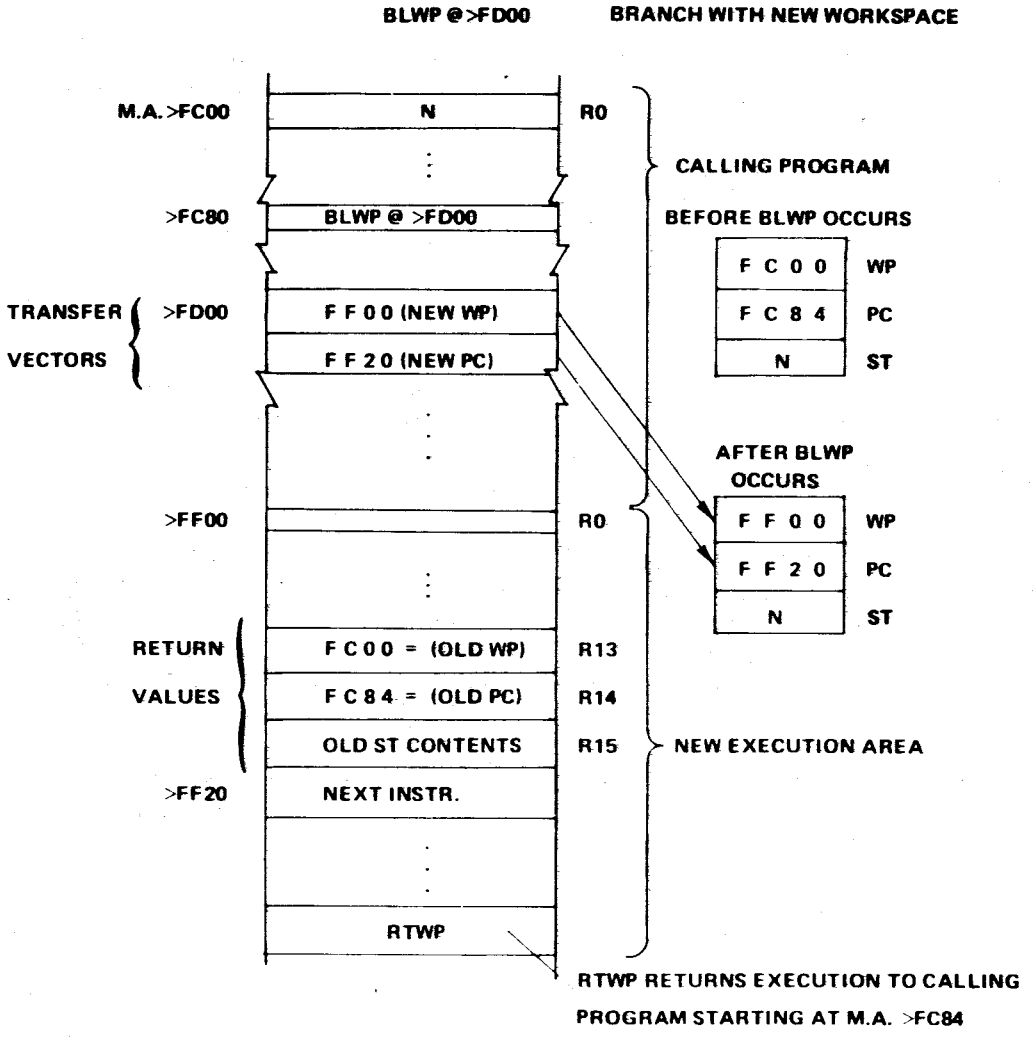


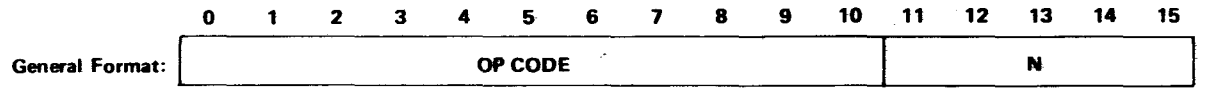
FIGURE 7-10. BLWP EXAMPLE

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 7-10).

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

MNEMONIC	PARAGRAPH	DEFINITION SUMMARY
JMP	7.6.2	One-word instruction, destination restricted to +127, −128 words from Program Counter value.
B	7.6.6	Two-word instruction, branch to any memory location.
BL	7.6.6	Same as B with PC return address in R11.
BLWP	7.6.7	Same as B with new workspace; old WP, PC and ST contents (return vectors) are in new R13, R14, R15.
XOP	7.6.9	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.

7.6.7 Format 7 (RTWP, CONTROL) Instructions



External instructions cause the three most-significant address lines (A0 through A2) 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 A0, A1, and A2. 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. Note, that although included here because of its format, the RTWP instruction is not classified as an external instruction because it does not affect the address lines or CRUCLK.

CKOF- is used to gain access to the mapper registers. CKON- is used to enable the mapper chip to drive extended address. For more information, see Section 3.

MNEMONIC	OP CODE 0 1 2 3 4 5 6 7 8 9 10	MEANING	STATUS BITS AFFECTED	DESCRIPTION	ADDRESS BUS* A0 A1 A2
IDLE	0 0 0 0 0 0 1 1 0 1 0	Idle	–	Suspend TMS 9900 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	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 *TIBUG*	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)	

These outputs from the TMS 9900 go to a SN74LS138 as shown in Figure 7-14.

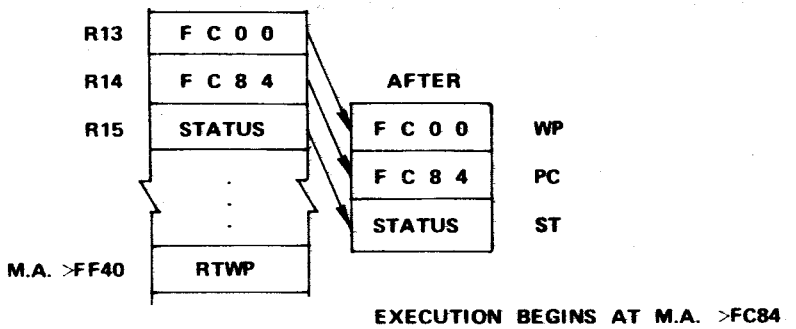
ASSEMBLY LANGUAGE:

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 WP (R13), PC (R14), ST (R15) VALUES



EXECUTION BEGINS AT M.A. >FC84
WITH R0 AT M.A. >FC00.

7.6.8 Format 8 (IMMEDIATE, INTERNAL REGISTER LOAD/STORE) Instructions

7.6.8.1 Immediate Register Instructions

General format

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE											N	R			
IOP															

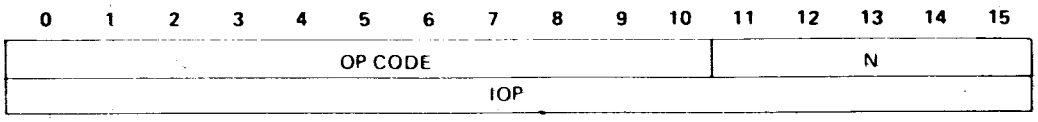
MNEMONIC	OP CODE 0 1 2 3 4 5 6 7 8 9 10	MEANING	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
AI	0 0 0 0 0 0 1 0 0 0 1	Add immediate	Yes	0-4	(R) + IOP → (R)
ANDI	0 0 0 0 0 0 1 0 0 1 0	AND immediate	Yes	0 2	(R) AND IOP → (R)
CI	0 0 0 0 0 0 1 0 1 0 0	Compare immediate	Yes	0 2	Compare (R) to IOP 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
1 + 1 = 1
0 + 0 = 0

7.6.8.2 Internal Register Load Immediate Instructions

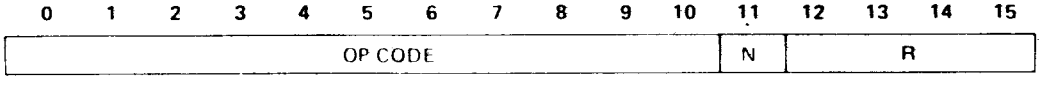
General format:



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

7.6.8.3 Internal Register Store Instructions

General format:



NO ST BITS ARE AFFECTED.

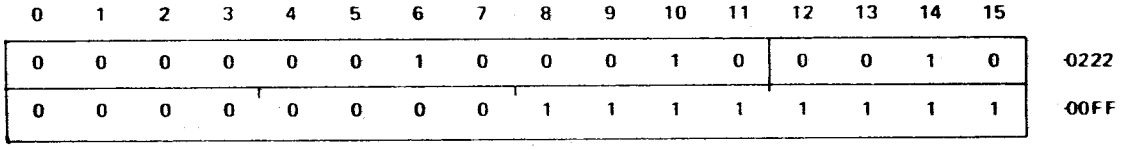
MNEMONIC	OP CODE (0 1 2 3 4 5 6 7 8 9 10)	MEANING	DESCRIPTION
STST	0 0 0 0 0 0 1 0 1 1 0	Store status register	(ST) → (R)
STWP	0 0 0 0 0 0 1 0 1 0 1	Store workspace pointer	(WP) → (R)

EXAMPLES

(1) ASSEMBLY LANGUAGE:

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

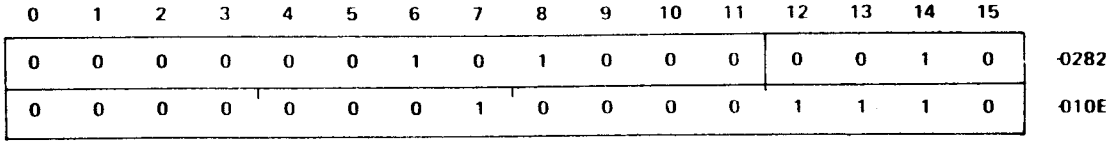
MACHINE LANGUAGE:



(2) ASSEMBLY LANGUAGE:

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

MACHINE LANGUAGE:



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

(3) ASSEMBLY LANGUAGE:

```
LWPI    >FC00       WP SET AT >FC00 (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
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	>FC00

(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.

7.6.9 Format 9 (XOP) Instructions

Other format 9 instructions (MPY, DIV) are explained in section 7.6.3 (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	D (XOP NUMBER)				T_S		SR			

The TS 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 9900 does not test interrupt request (INTREQ-) upon completion of the XOP instruction.

An XOP is a means of calling one of 16 subtasks available for use by any executing task. The EPROM memory area between M.A. 40_{16} and $7E_{16}$ is reserved for the transfer vectors of XOP's 0 to 15 (see Figure 7-1). 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 7-11 depicts calling an XOP to process a table of data; the data begins at M.A. $FF00_{16}$. This XOP example uses XOP vectors that point directly to the XOP service routine WP and PC.

ASSEMBLY LANGUAGE:

```
XOP     @>FF00,4
```

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
1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	>FF00

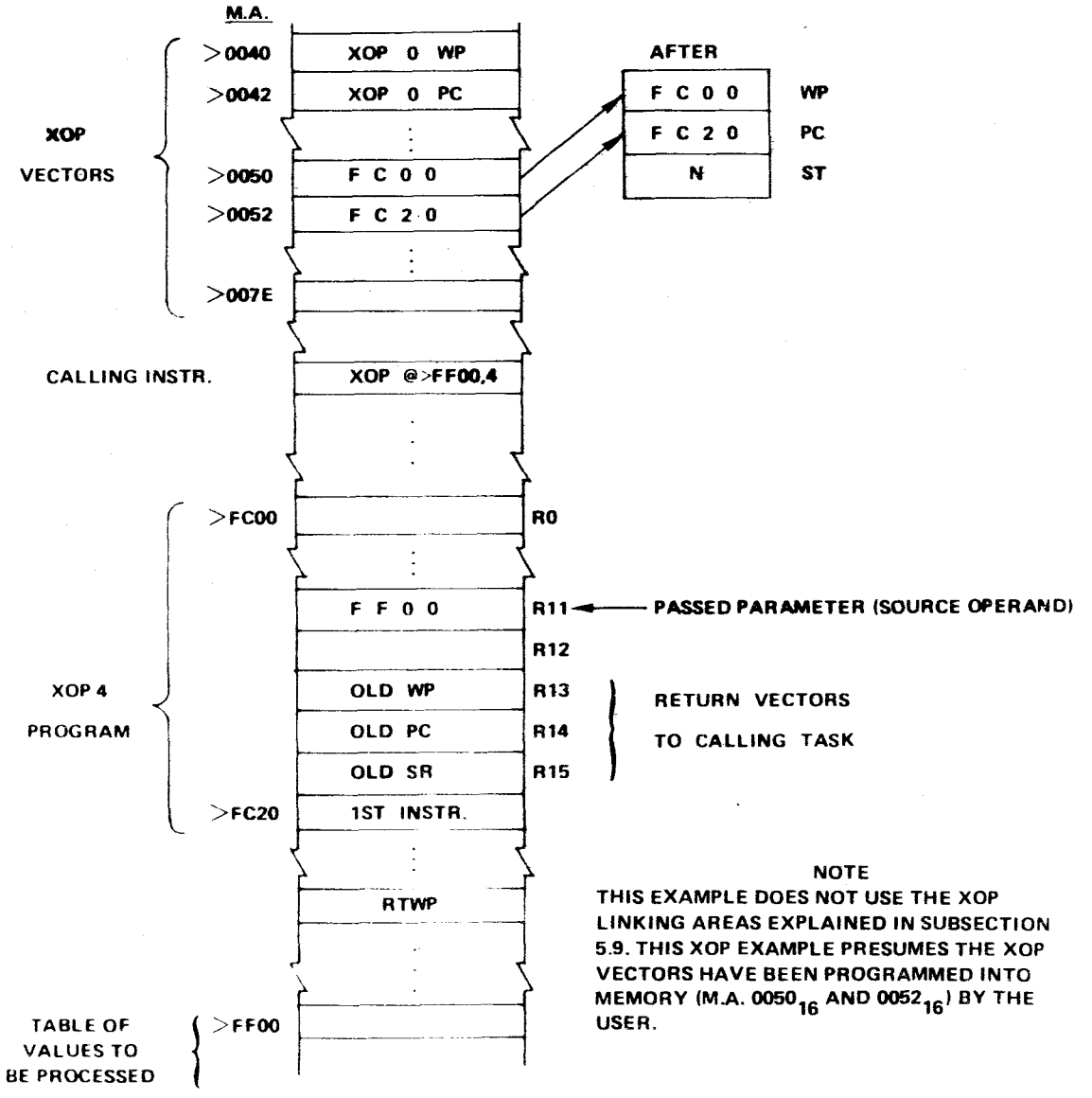


FIGURE 7-11. XOP EXAMPLE

7.7 COMMUNICATIONS REGISTER UNIT (CRU)

Input and output is mainly done on the TM 990/102 using the Communications Register Unit or CRU. This is a separate hardware structure with its own data and control lines. Thus the TMS 9900 microprocessor has one address bus, but two sets of control and data buses. One set, the memory set, has a 16-bit parallel bidirectional data bus and three control lines, MEMEN-, DBIN, and WE-.

The other set, the CRU I/O set, uses two lines, one line for input (CRUIN), and one for output (CRUOUT). There is one control line, CRUCLK, used to strobe a bit being output on CRUOUT. A bit being input on CRUIN has no strobe and is simply sampled by the microprocessor at its discretion.

CRU devices are run on one phase of the system clocks, and therefore, the rate of data transfer on the CRUIN line is a function of the system clock. Since the CPU also uses this system clock, it will sample the CRUIN line at a rate that is a function of the system clock when doing a CRU read operation (executing a CRU read instruction - STCR or TB).

Thus, the CRU data group consists of three lines - CRUIN, CRUOUT, and CRUCLK. The address bus supplies CRU addresses as well as memory addresses; which operation being performed is determined by the presence of the proper control signals. Memory operations use address bits 0 through 14 externally, bit 15 is used inside the processor for byte operations. CRU operations, however, use only bits 3 through 14; bits 0, 1, and 2 are set to zero, and bit 15 of an address is totally ignored.

When CRU instructions are executed, data is written or read through the CRUOUT or CRUIN pins respectively, of the TMS 9900 to or from designated devices addressed via the address bus of the microprocessor.

The CRU software base address is maintained in register 12 (bits 0 to 15) of the workspace register area. Only bits 3 through 14 of the register are interpreted by the CPU for the desired CRU address, and this 12-bit value is called the CRU hardware base address. When the displacement is added to the hardware base address, the result is the CRU bit address further explained in section 7.7.1.

TM 990/102 devices driven off the CRU interface include the TMS 9901 programmable systems interface and the TMS 9902 serial interface which are accessed through the CRU addresses noted in Table 7-6. This table also lists the functions of the other CRU addresses which can be used for on-board or off-board I/O use. Addressing the TMS 9901 and TMS 9902 for use as interval timers is explained, along with programming examples, in Sections 4 and 5. Further detailed information on these two devices can be obtained from their respective data manuals.

7.7.1 CRU Addressing

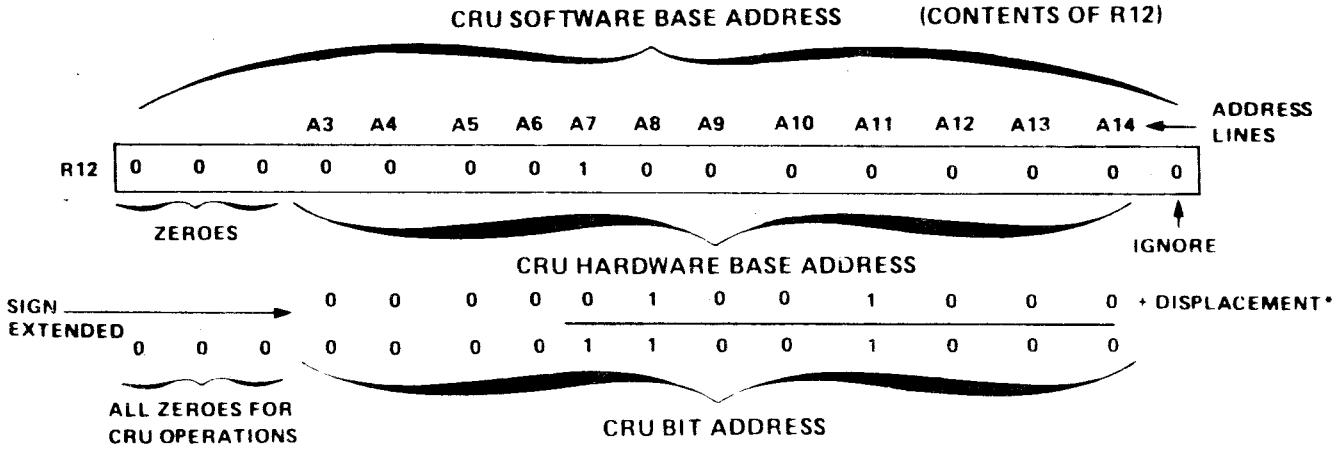
The CRU software base address is contained in the 16 bits of register 12. From the CRU software base address, the processor is able to determine the CRU hardware base address and the resulting CRU bit address. These concepts are illustrated in Figure 7-12.

TABLE 7-6. TM 990/102 PREDEFINED CRU ADDRESSES

Function	CRU Hardware Base Address (R12, bits 3-14)	CRU Software Base Address (R12, bits 0-15)
TMS 9902, Main I/O (Lower Half)	040	0080
TMS 9902, Main I/O (Upper Half)	050	00A0
TMS 9901 Interrupt Mask, System Timer	080	0100
Status LED	090	0120
Offboard CRU	0A0	0140

NOTE

The CRU software base address is equal to 2X the hardware base address, or the hardware base address is 1/2 the software base address.



*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 CRU instructions (TB, SBO, and SBZ).

FIGURE 7-12. CRU BASE AND BIT ADDRESSES

7.7.1.1 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 I/O and clock lines. The CRU bit address is the sum of the displacement value of the CRU instruction (displacement value is the same as operand value of the single-bit instructions TB, SBO, and SBZ) and the CRU hardware base address in bits 3 to 14 of R12. Note that the sign bit of the eight-bit displacement is extended to the left and added as part of the address. The resulting CRU hardware bit address is then placed on address lines A3 to A14; address lines A0 to A3 will always be zeroes in CRU instruction execution.

7.7.1.2 CRU Hardware Base Address

The CRU hardware base address is the value in bits 3 to 14 of R12. For instructions that do not specify a displacement (LDCR and STCR do not), the CRU hardware base address is the same as the first CRU bit address (see 7.7.1.1). 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.

7.7.1.3 CRU Software Base Address

The CRU software base address is the entire 16-bit contents of R12. This is the value loaded into register 12 when setting up to program the CRU device. Bits 0, 1, 2, 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. Hardware logic to a CRU-addressable device can reveal its CRU bit address and hardware base address by tracing the device-enabling bit pattern that will be on address lines A3 to A14. This hardware base address fits conveniently in bits 3 to 14 of register 12. However, there is a bit 15 in the register which must be filled; thus, a zero can be added in this rightmost bit. In essence, this shifts the hardware base address one bit to the left, essentially multiplying it by two. Therefore, when approaching the CRU base address from a hardware standpoint, consideration must be given to the difference between the CRU bit address and the value to be loaded in register 12 (CRU software base address).

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}, must be added or subtracted from R12. For example, with R12 containing 0000_{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 addresses, etc.

7.7.2 CRU Timing

CRU timing is shown in Figure 7-13. Timing phases (ø1 to ø4) are shown at the top of the figure. The CRU address is valid on the address bus beginning at the start of ø2, and stays valid for eight timing phases (two clock cycles). At the start of the next ø2 phase, CRUCLK at the TMS 9900 goes high for two phases to provide timing for CRUOUT sampling. Note that for LDCR and STCR instructions, the address bus is incremented for each data bit to be output or input. For input operations, the address is placed on the address bus at the beginning of phase ø2, and the input is sampled between phases ø4 and ø1.

7.7.3 CRU Instructions

The five instructions that program the CRU interface are:

- LDCR Place the CRU hardware base address on address lines A3 to A14. Load from memory a pattern of 1 to 16 bits and serially transmit this pattern through the CRUOUT pin of the TMS 9900. Increment

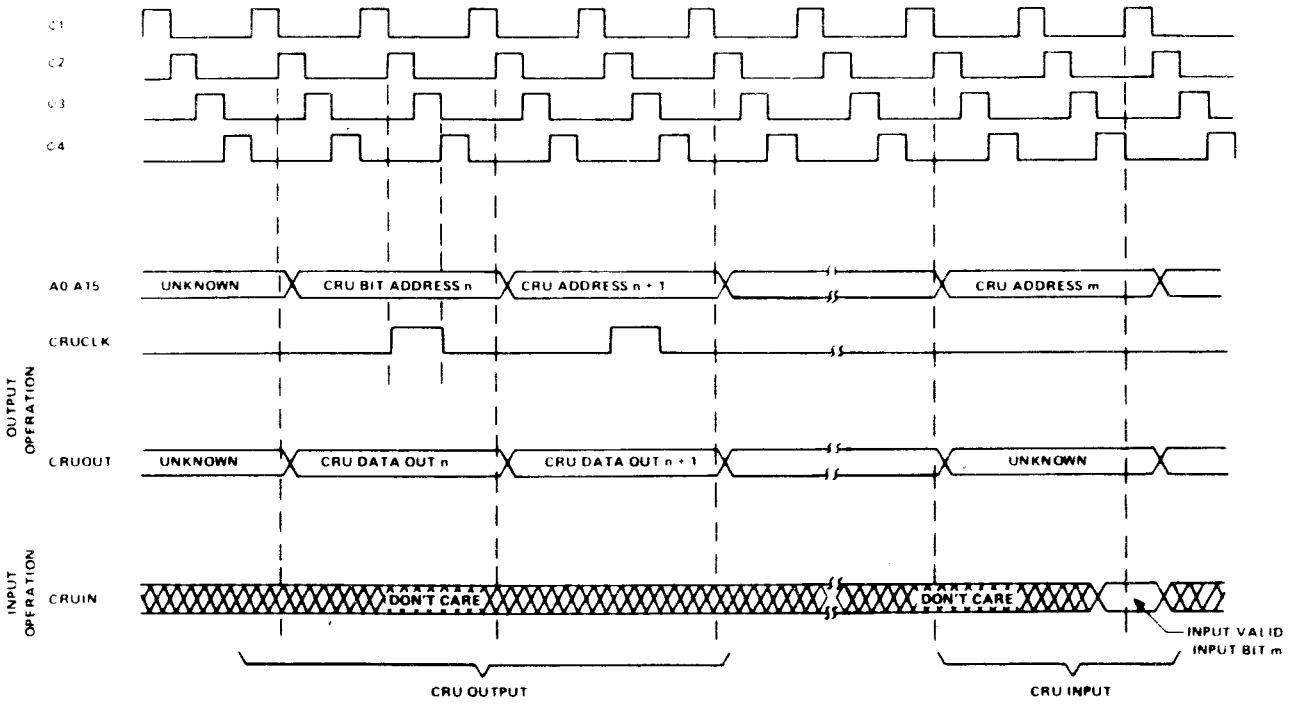


FIGURE 7-13. TMS 9900 CRU INTERFACE TIMING

the address on A3 to A14 after each CRUOUT transmission.

- STCR Place the CRU hardware base address on lines A3 to A14. Store into memory a pattern of 1 to 16 bits obtained serially at the CRUIN pin of the TMS 9900. Increment the address on A3 to A14 after each CRUIN sampling.

- SBO Place the CRU hardware base address plus the instruction's signed displacement on address lines A3 to A14. Send a logical one through the CRUOUT pin of the TMS 9900.

- SBZ Place the CRU hardware base address plus the instruction's signed displacement on address lines A3 to A14. Send a logical zero through the CRUOUT pin of the TMS 9900.

- TB Place the CRU hardware base address plus the instruction's signed displacement on address lines A3 to A14. Sample the CRUIN pin of the TMS 9900 and place the bit read into ST2, the Equal Bit of the Status register.

7.7.3.1 CRU Multibit Instruction

The two multibit instructions, LDCR and STCR, address the CRU devices by placing bits 3 through 14 (hardware base address) of R12 on address lines A3 through A14. A0, A1, and A2 are set to zero for all CRU operations. The first operand of the instruction is the source field address and the second operand is the number of bits in the operation.

If the number of bits is from 1 through 8, only the left byte of the source or receiving field takes part in the operation, and bits are shifted in or out from the least significant bit(s) of that left byte. For example, LDCR R2,1

outputs the eighth bit of R2 to the CRU at the address derived from register 12. An STCR R5,2 would receive two bits of data serially and insert them into bit 7 and then bit 6 of register 5 (register bit numbers go from 0 to 7 for the left byte in this example). The CRU address lines are automatically incremented to address each new CRU bit, until the required number of bits are transferred. In an STCR instruction, unused bits of the byte or word are zeroed. In this last example, bits 0-5 are zeroed, the right byte (bits 8-15) is unaffected.

An LDCR loads the CRU device serially from memory over CRUOUT timed by CRUCLK. An STCR stores data into memory obtained serially through CRUIN from the addressed CRU device. Figures 7-14 and 7-15 show this operation graphically. The TMS 9901 is used in the example as the CRU device because it most simply shows the bit transfers involved.

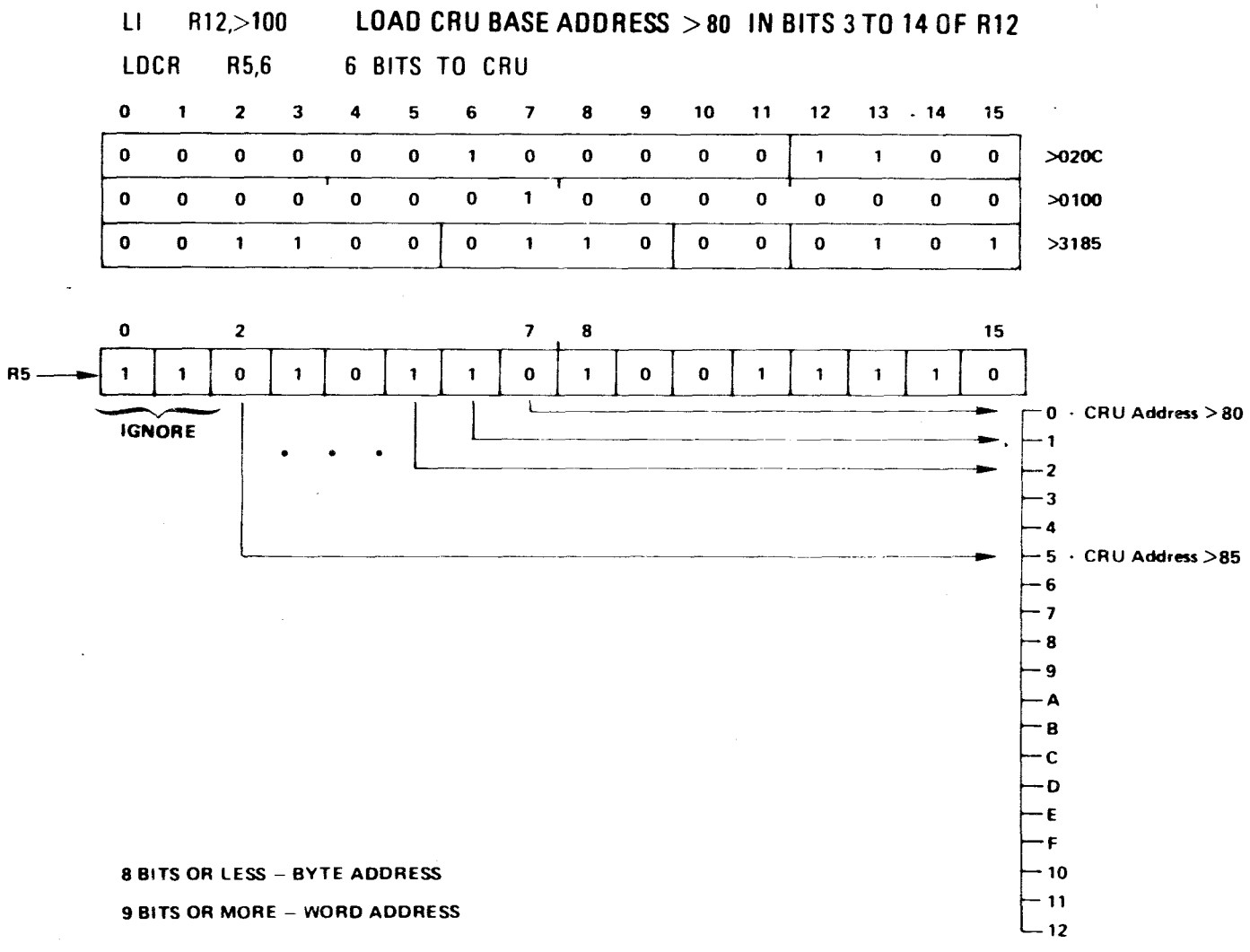


FIGURE 7-14. LDCR INSTRUCTION

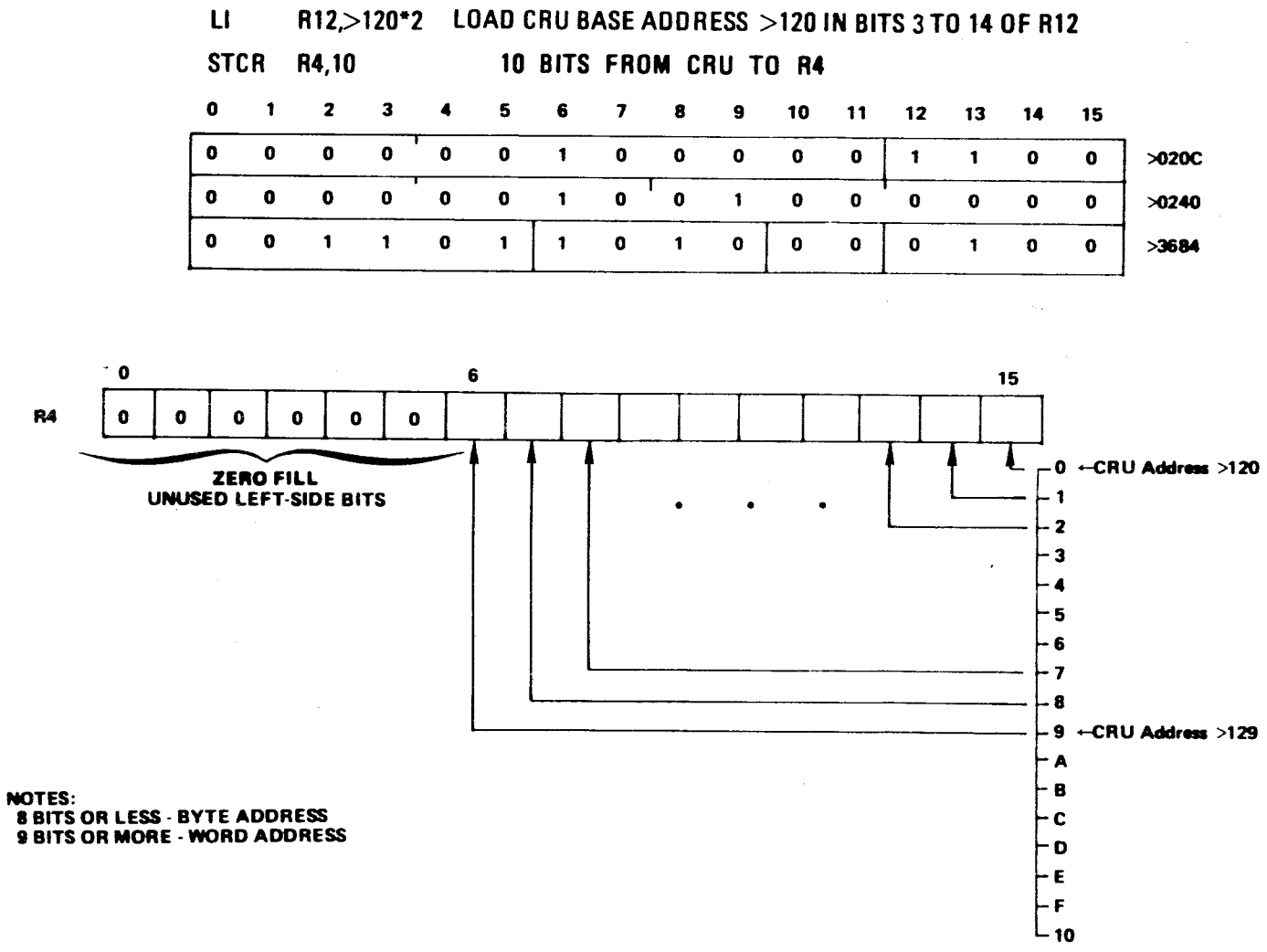


FIGURE 7-15. STCR INSTRUCTION

7.7.3.2 CRU Single-Bit Instructions

The three single-bit instructions are set bit to zero (SBZ), set bit to one (SBO), and test bit (TB). The first two are output instructions, and the last one is an input instruction. All three instructions have only one operand, which is assembled into an eight-bit signed displacement to be added to the CRU hardware base address to provide the CRU bit address. The SBZ instruction sets the addressed bit to zero (zero on CRUOUT), and the SBO instruction sets the addressed bit to one (one on CRUOUT). The TB instruction reads the logical value on the CRUIN line and places this value in bit 2 (EQ) of the Status Register; the test can be proven by using the JEQ or JNE instructions.

The operand value is treated as a signed, eight-bit number, and thus has a range of values of -128 to +127. This number is added to the CRU hardware base address derived from bits 3 to 14 of register 12, and the result is placed on the address lines. This process is illustrated in Figure 7-16.

Notice that after execution of a TB instruction, a JEQ instruction will cause a jump if the logic value on CRUIN was a one, and the JNE will cause a jump if the logic value was a zero.

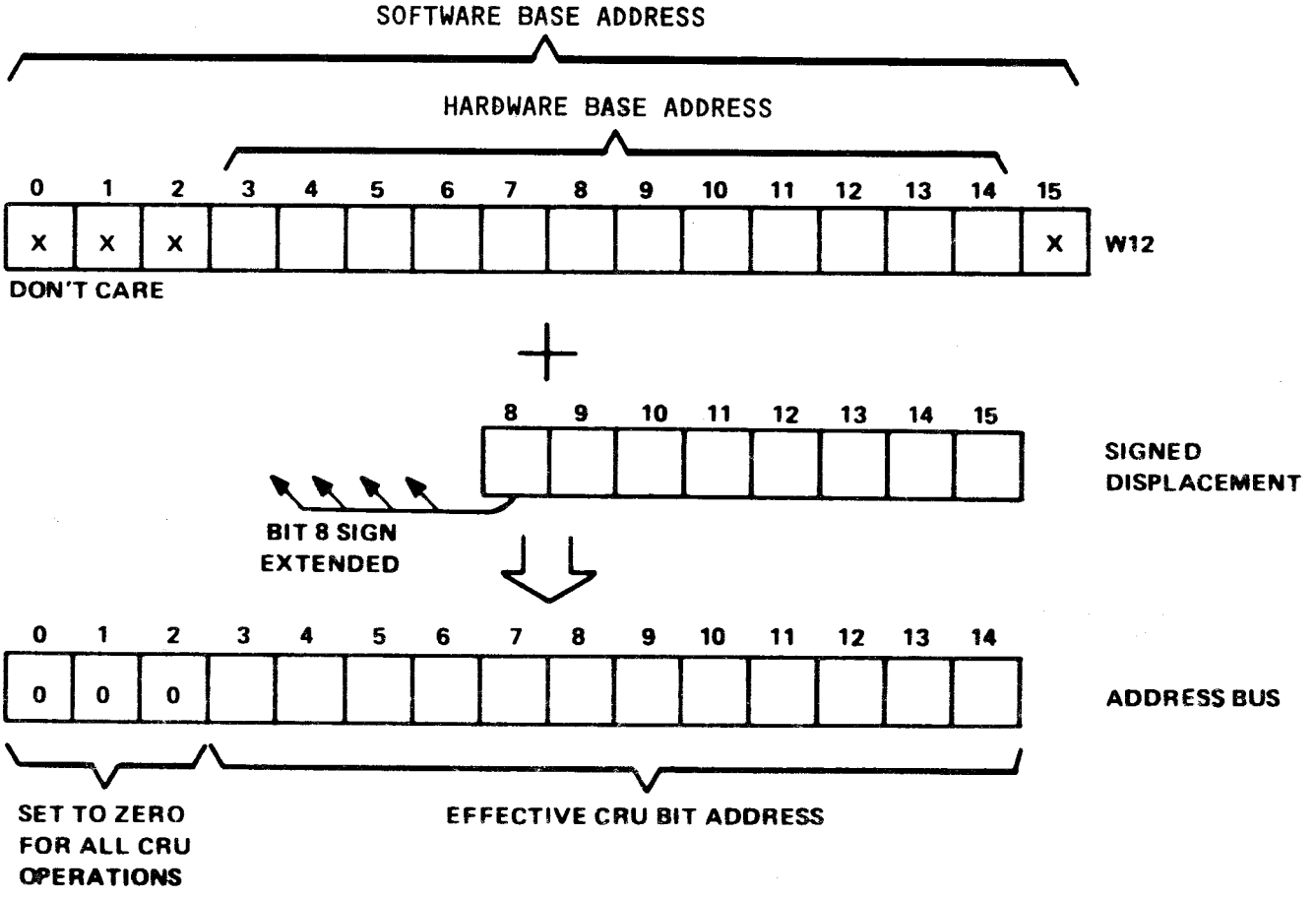


FIGURE 7-16. ADDITION OF DISPLACEMENT AND R12 CONTENTS TO DRIVE CRU BIT ADDRESS

APPENDIX A

SCHEMATICS

1600284 SH 1

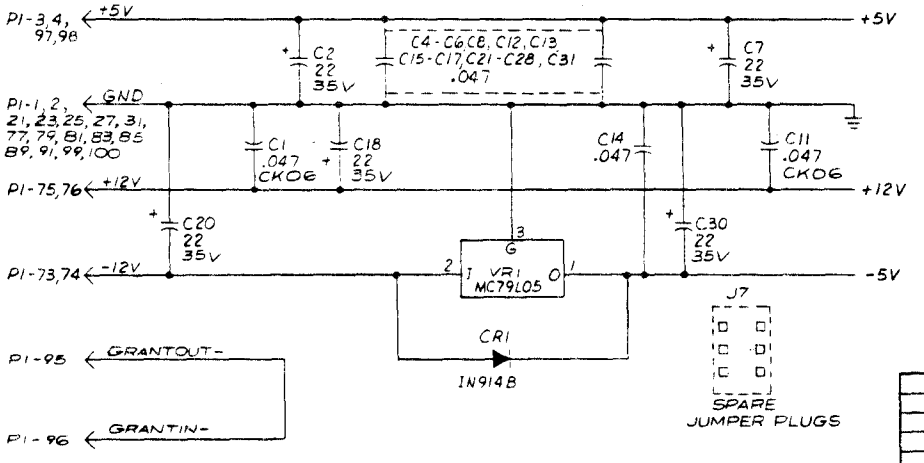
NOTES: UNLESS OTHERWISE SPECIFIED

1. ALL RESISTORS ARE .25W, 5%
2. ALL RESISTANCE VALUES ARE IN OHMS
3. ALL CAPACITANCE VALUES ARE IN MICROFARADS
4. TWENTY EIGHT (28) PIN SOCKETS PROVIDED AT U10 AND U12. USER MAY POPULATE WITH TMS2516, TMS2532 (24 PINS) OR TMS2564 (28 PIN) DEVICE.
5. 102-2 ASSY USES TMS4532-20NL-1 OR TMS4532-20NL-2 POPULATED AT U49 THRU U64 (PARTS CANNOT BE MIXED)
   102-3 ASSY USES TMS4164-20NL POPULATED AT U49 THRU U64
   102-1 ASSY-NONE
6. FOR 102-2 ASSY USING TMS4532-20NL-1 RAMS, E2 TO E5 IS CONNECTED
7. FOR 102-2 ASSY USING TMS4532-20NL-2 RAMS, E1 TO E4 IS CONNECTED
8. FOR 102-3 ASSY USING TMS4164-20NL RAMS E3 TO E6 IS CONNECTED
9. FOR 102-2 ASSY, E7 TO E9 IS CONNECTED
   FOR 102-3 ASSY, E8 TO E10 IS CONNECTED
   FOR 102-1 ASSY, E11 TO E12 IS CONNECTED

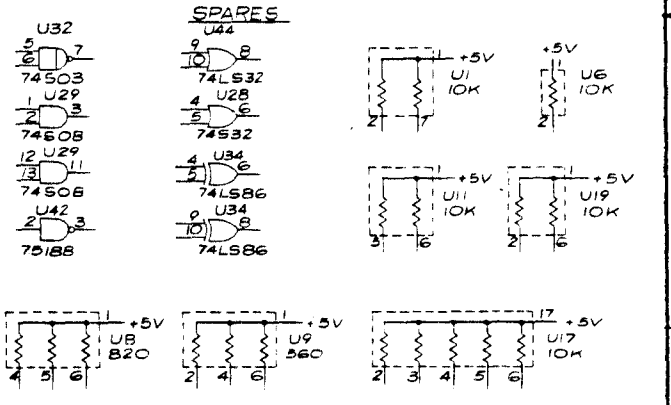
DEVICE	GND	+5V	+12V	-12V	-5V
74LS00 (U26, U35)	7	14			
74S03 (U32)	7	14			
74LS04 (U37, U45)	7	14			
74LS08 (U46)	7	14			
74S08 (U29)	7	14			
74LS20 (U31)	7	14			
74LS27 (U22)	7	14			
74LS32 (U44)	7	14			
74S32 (U28)	7	14			
74LS74 (U36, U20, U39-U41)	7	14			
74S74 (U38)	7	14			
74LS86 (U34)	7	14			
74LS138 (U25)	8	16			
74LS244 (U3, U7)	10	20			
74LS245 (U4, U5, U15, U16, U18)	10	20			
7518B (U42)	7		14	1	
75189AN (U47)	7	14			
74L6682 (U14, U27, U40)	10	20			
74LS612 (U24)	20	40			
TL7705 (U48)	4	8			
TMS 4500 (U23)	20	40			
TMS 9900 (U21)	26,40	2,59	27		1
TMS 9901 (U2)	16	40			
TMS 9902 (U33)	9	18			
TIM 9904 (U43)	3,10	20	13		
TMS 4164 (U49-U64)	16	8			
TMS 4532 (U49-U64)	16	8			
TMS 2516/2532 (U10, U12)	12	24			
TMS 2564 (U10, U12)	14	1,26			

REFERENCE DESIGNATORS	
USED	NOT USED
C1-C31	C3, C29
CR1	
DS1, DS2	
E1-E12	
J1-J7	
L1	
P1, P2	
VR1	
R1-R27	R5, R9, R10 R13, R14, R22
U1-U64	
Y1	

REVISIONS			
REV	DESCRIPTION	DATE	APPROVED
A	INFORMAL ENG/DFTG CHANGE	10/5/81	Hansen
B	CN 471635 w/ Cuts	10/29/81	Hansen



SPARES



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

PARTS LIST

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

.013 - .004 DIA	.128 - .005 DIA	.196 - .005 DIA	.358 - .005 DIA
.129 - .001 DIA	.360 - .001 DIA	.800 - .001 DIA	
.501 - .005 DIA	.751 - .010 DIA	1.801 - .013 DIA	
.750 - .001 DIA	.750 - .010 DIA	1.750 - .013 DIA	
.750 - .001 DIA	.750 - .010 DIA	2.000 DIA	

1600286	8117
NEXT ASSY	USED ON
APPLICATION	

SEQ NO	IDENT	P SPEC	NC	ADDITIONAL	NOTES
	PROCESS		CLASSIFICATION		
PROCESSES — FOR CORRELATION TO GOVT AND SPECIFICATIONS, SEE TI DRAWING 7247					

REV STATUS OF SHEETS	REV	A	A	A	B	A	A
	SH	1	2	3	4	5	6

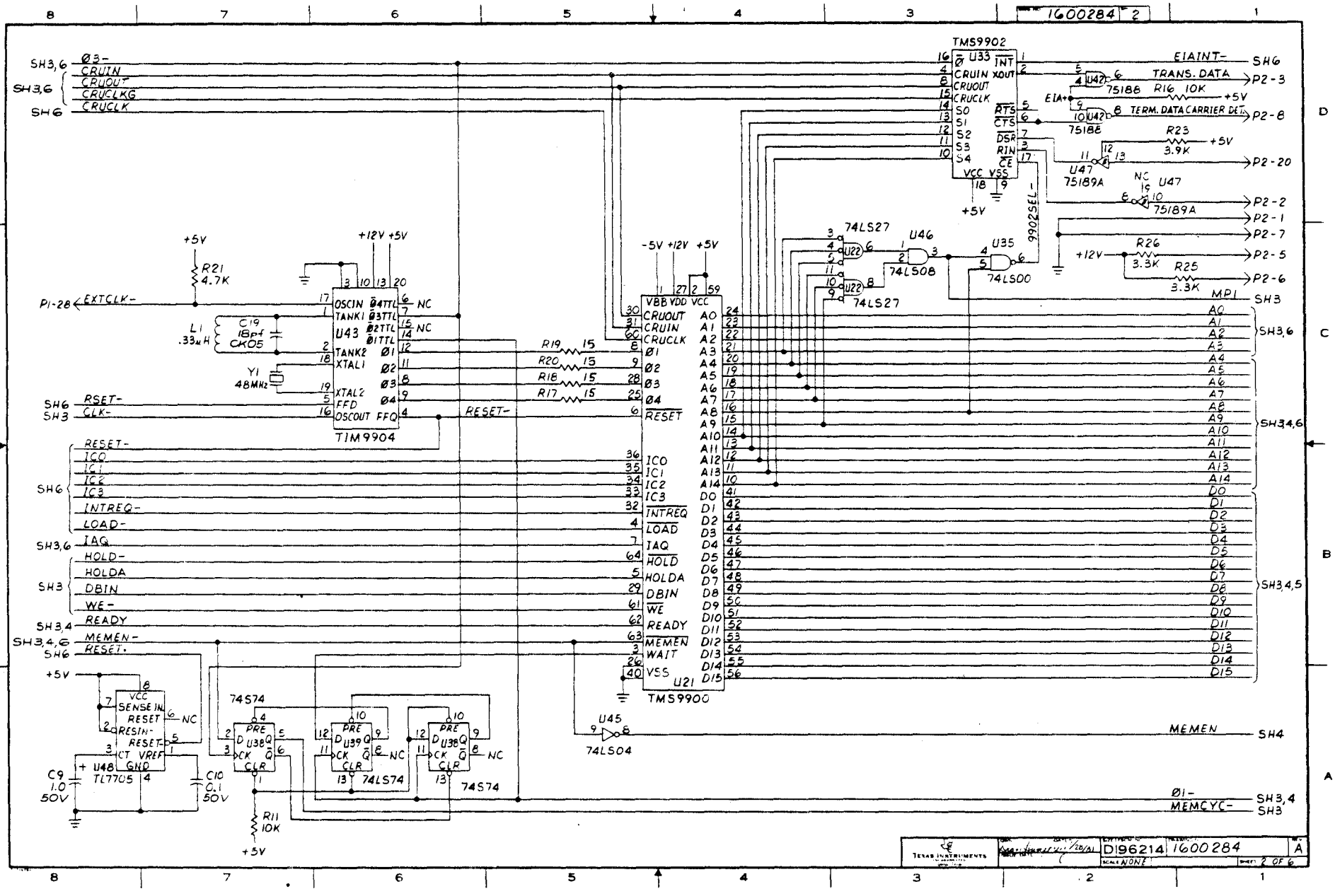
DRAWN: Thompson 6/1/81
CHK: Hansen 10/29/81
ENGR: [illegible] IV. 2
MFG:
QA:
CONFIG:
MGR:

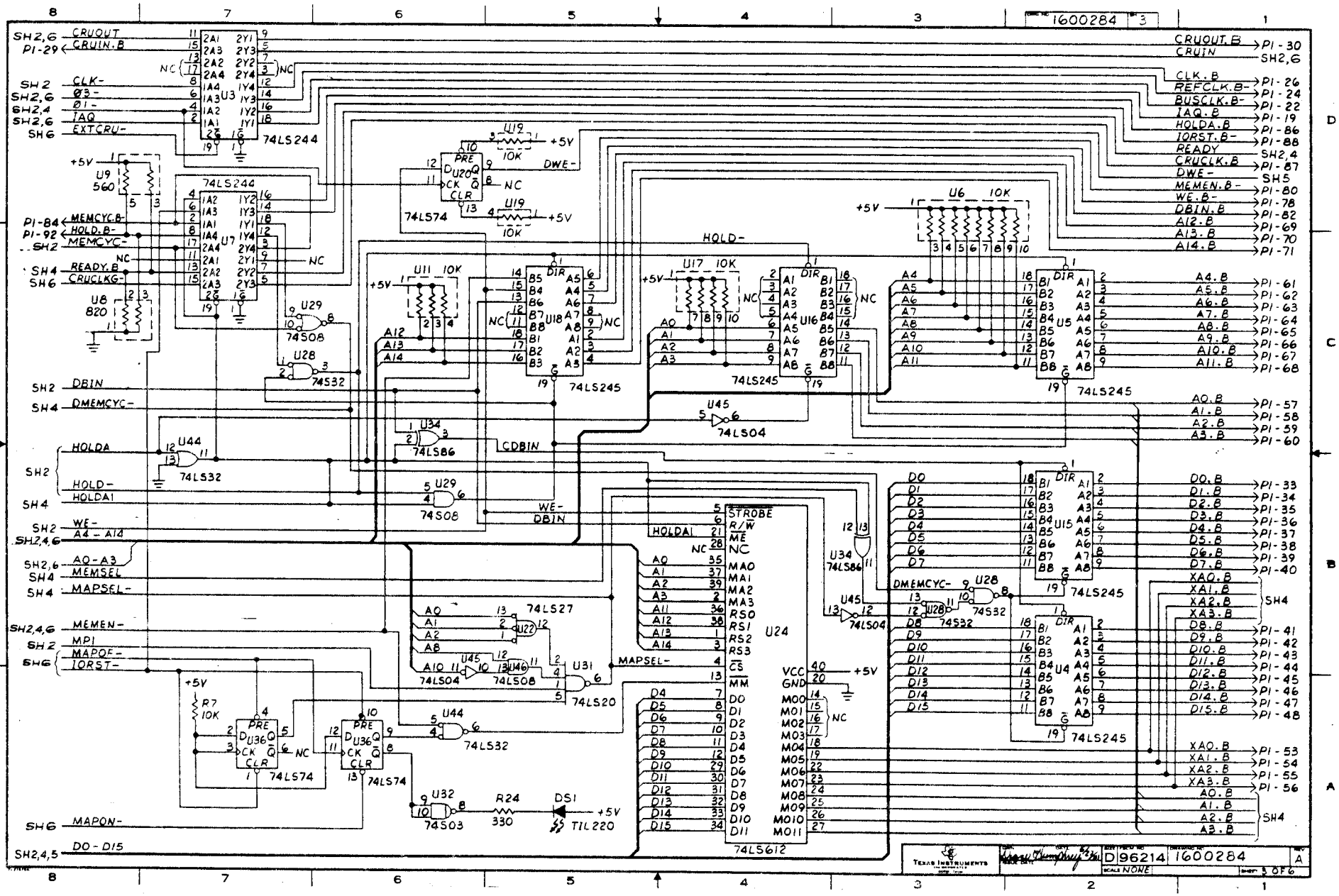
TEXAS INSTRUMENTS INCORPORATED Dallas, Texas

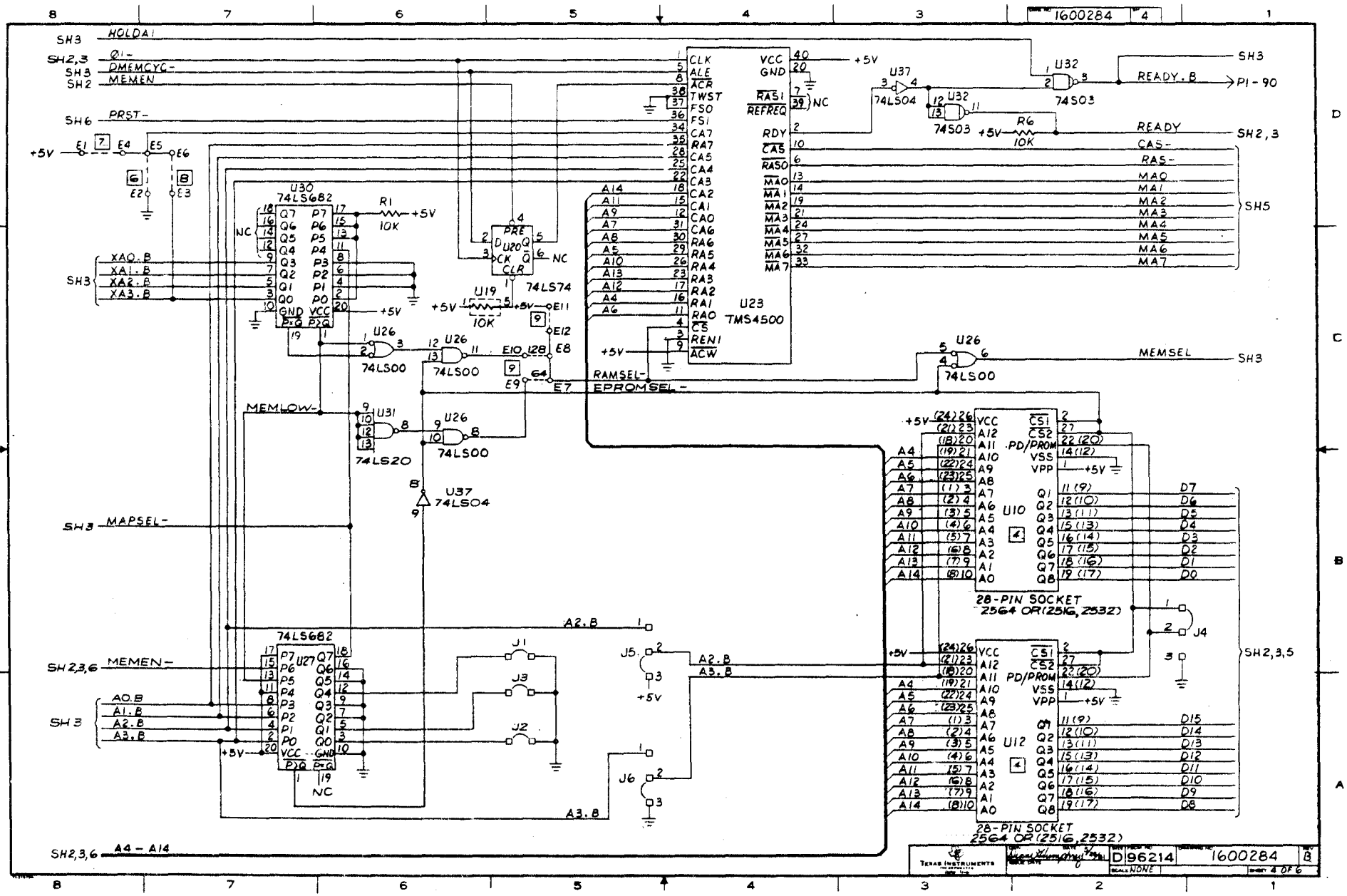
DIAGRAM, LOGIC, TM990/102

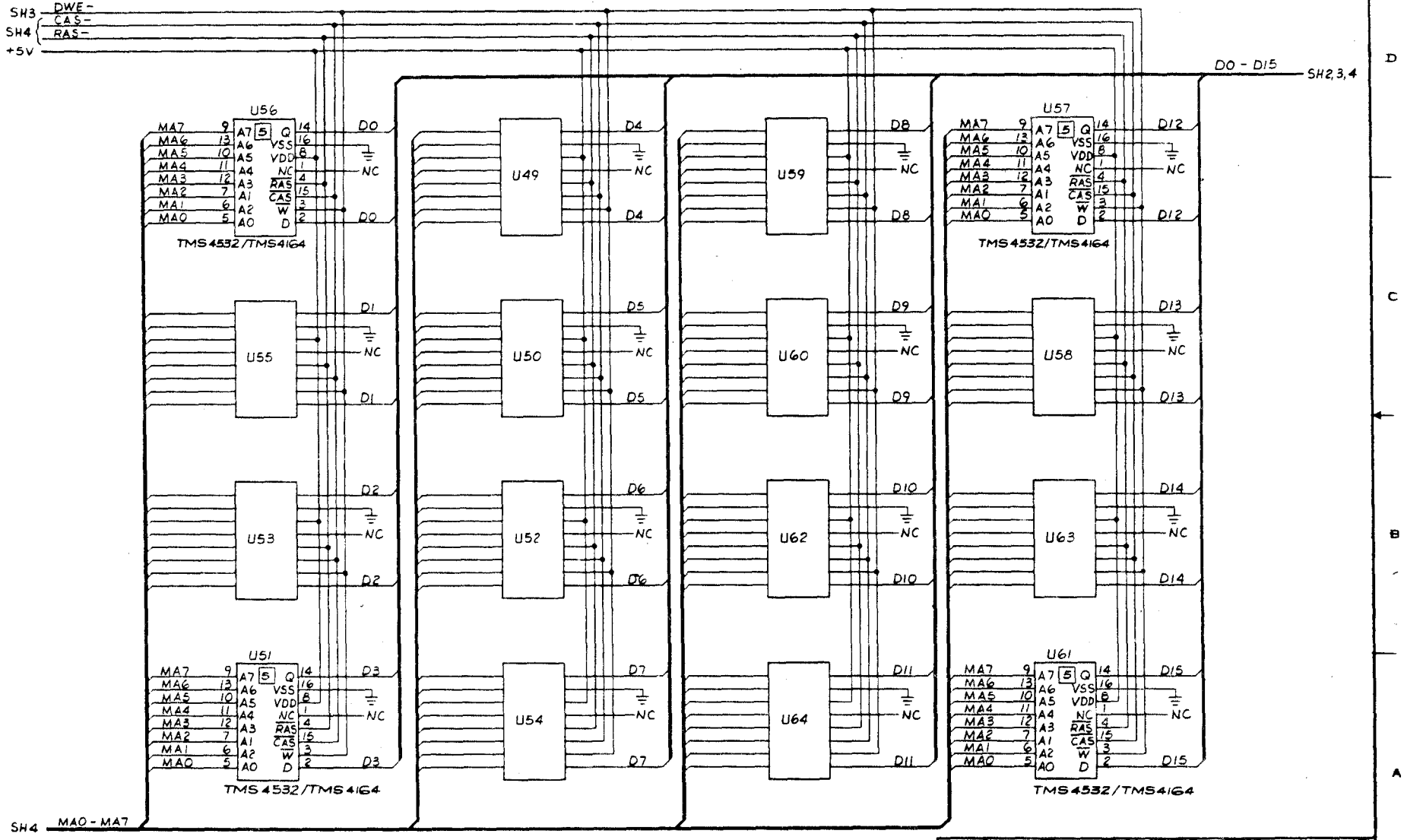
SIZE D | DWG NO 1960214 | REV 1600284

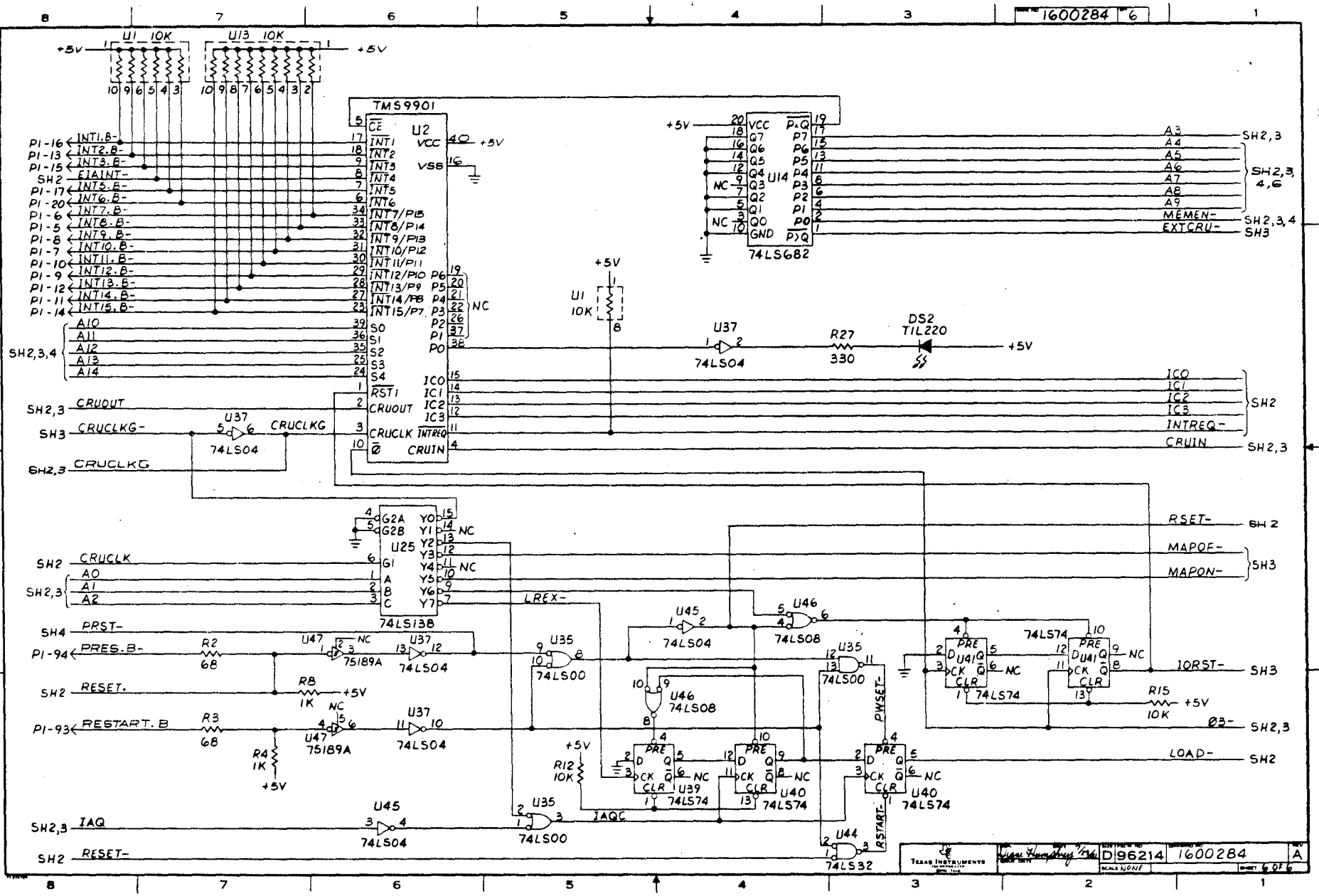
SCALE NONE | SHEET 1 OF 6


1600284 2
TMS9902
U33
INT
XOUT
CRUIN
CRUOUT
CRUCLK
S0
S1
S2
S3
S4
RTS
CTS
DSR
RIN
CE
VCC VSS
9902SEL-
EIAINT- SH6
TRANS. DATA P2-3
75188
R16 10K
+5V
EIA+
TERM. DATA CARRIER DET. P2-8
R23
3.9K
P2-20
U47
75189A
P2-2
P2-1
P2-7
P2-5
P2-6
R26
3.3K
R25
3.3K
74LS27
U46
74LS08
U35
74LS00
MPI SH3
SH3,6
SH3,4,6
SH3,4,5
TIM9904
OSCIN
TANK1
U43
TANK2
XTAL1
XTAL2
FFD
OSCOUT FFQ
EXTCLK-
P1-28
L1
.33μH
C19
18pf
CKO5
Y1
48MHz
R21
4.7K
RSET-
CLK-
R19 15
R20 15
R18 15
R17 15
RESET-
TMS9900
U21
VBB VDD VCC
VSS
ICO
IC1
IC2
IC3
INTREQ-
LOAD-
IAQ
HOLD-
HOLDA
DBIN
WE-
READY
MEMEN-
WAIT
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 A10 A11 A12 A13 A14
D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12 D13 D14 D15
74S74
U38
U39
PRE
CLR
74LS74
U45
74LS04
MEMEN SH4
Ø1- SH3,4
MEMCYC- SH3
U48
TL7705
SENSE IN
RESET
RESIN-
RESET-
CT VREF
GND
C9
1.0
50V
C10
0.1
50V
R11
10K
D96214
1600284
A
SCALE NONE
SHEET 2 OF 6


1600284 3
SH2,6 CRUOUT
PI-29 CRUIN.B
CRUOUT.B PI-30
CRUIN SH2,6
SH2 CLK-
SH2,6 Ø3-
SH2,4 Ø1-
SH2,6 IAQ
SH6 EXTCRU-
74LS244
U3
CLK.B
REFCLK.B- PI-26
BUSCLK.B- PI-24
IAQ.B PI-22
HOLDA.B PI-19
IORST.B- PI-86
READY PI-88
CRUCLK.B SH2,4
DWE- PI-87
MEMEN.B- SH5
WE.B- PI-80
DBIN.B PI-78
AI2.B PI-82
AI3.B PI-69
AI4.B PI-70
PI-84 MEMCYC.B-
PI-92 HOLD.B-
SH2 MEMCYC-
SH4 READY.B
SH6 CRUCLKG-
U7
U8 820
U9 560
U10 10K
U11 10K
U17 10K
U19 10K
U29
74S08
U28
74S32
U34
74LS86
U44
74LS32
U45
74LS04
74LS574
74LS245
U16
U18
U5
U4
U15
HOLD-
HOLDA
HOLD-
HOLDA1
SH2 DBIN
SH4 DMEMCYC-
SH2 WE-
A4 - A14
SH2,6 AQ-A3
SH4 MEMSEL
SH4 MAPSEL-
SH2,4,6 MEMEN-
SH2 MPI
SH6 MAPOF-
IORST-
MAPON-
SH2,4,5 DO-D15
CDBIN
WE-
DBIN
U24
74LS612
74LS27
U22
74LS20
U31
74LS503
R7 10K
R24 330
DS1
TIL220
A4.B PI-61
A5.B PI-62
A6.B PI-63
A7.B PI-64
A8.B PI-65
A9.B PI-66
AI0.B PI-67
AI1.B PI-68
AQ.B PI-57
A1.B PI-58
A2.B PI-59
A3.B PI-60
DO.B PI-33
D1.B PI-34
D2.B PI-35
D3.B PI-36
D4.B PI-37
D5.B PI-38
D6.B PI-39
D7.B PI-40
XAQ.B
XA1.B
XA2.B
XA3.B
SH4
D8.B PI-41
D9.B PI-42
D10.B PI-43
D11.B PI-44
D12.B PI-45
D13.B PI-46
D14.B PI-47
D15.B PI-48
XAQ.B PI-53
XA1.B PI-54
XA2.B PI-55
XA3.B PI-56
AQ.B
A1.B
A2.B
A3.B
SH4
TEXAS INSTRUMENTS
D 96214 1600284 A
SCALE NONE
SHEET 5 OF 6


1600284
SH3 HOLDA1
SH2,3 Ø1-
SH3 DMEMCYC-
SH2 MEMEN
SH6 PRST-
+5V E1 E4 E5 E6 E2 E3
U30 74LS682
R1 10K
U20 74LS74
U19 10K
U23 TMS4500
CLK ALE ACR TWST FSO FSI CA7 RA7 CA5 CA4 CA3 CA2 CA1 CA0 CA6 RA6 RA5 RA4 RA3 RA2 RA1 RA0 CS REN1 ACW
VCC GND RAS1 REFREQ RDY CAS RAS0 MA0 MA1 MA2 MA3 MA4 MA5 MA6 MA7
U37 74LS04
U32 74S03
SH3 READY.B P1-90
R6 10K
READY SH2,3
CAS- RAS- MA0 MA1 MA2 MA3 MA4 MA5 MA6 MA7 SH5
XA0.B XA1.B XA2.B XA3.B SH3
U26 74LS00
RAMSEL- EPROMSEL-
MEMSEL SH3
MEMLOW-
U31 74LS20
U37 74LS04
SH3 MAPSEL-
U10 U12 28-PIN SOCKET 2564 OR(25L6, 2532)
PD/PROM
D7 D6 D5 D4 D3 D2 D1 D0
D15 D14 D13 D12 D11 D10 D9 D8
J4 SH2,3,5
74LS682 U27
SH2,3,6 MEMEN-
A0.B A1.B A2.B A3.B SH3
J1 J2 J3 J5 J6
A2.B A3.B
SH2,3,6 A4 - A14
TEXAS INSTRUMENTS
D96214
1600284
SCALE NONE
SHEET 4 OF 6
A-5


1600284 5
SH3 DWE-
SH4 CAS-
RAS-
+5V
D0 - D15
SH2,3,4
U56
U57
U49
U59
U55
U50
U60
U58
U53
U52
U62
U63
U51
U54
U64
U61
TMS4532/TMS4164
MA0 - MA7
D96214
1600284
A
SCALE NONE
5 OF 6



TEXAS INSTRUMENTS D 96214 1600284 A SCALE NONE SHEET 6 OF 6

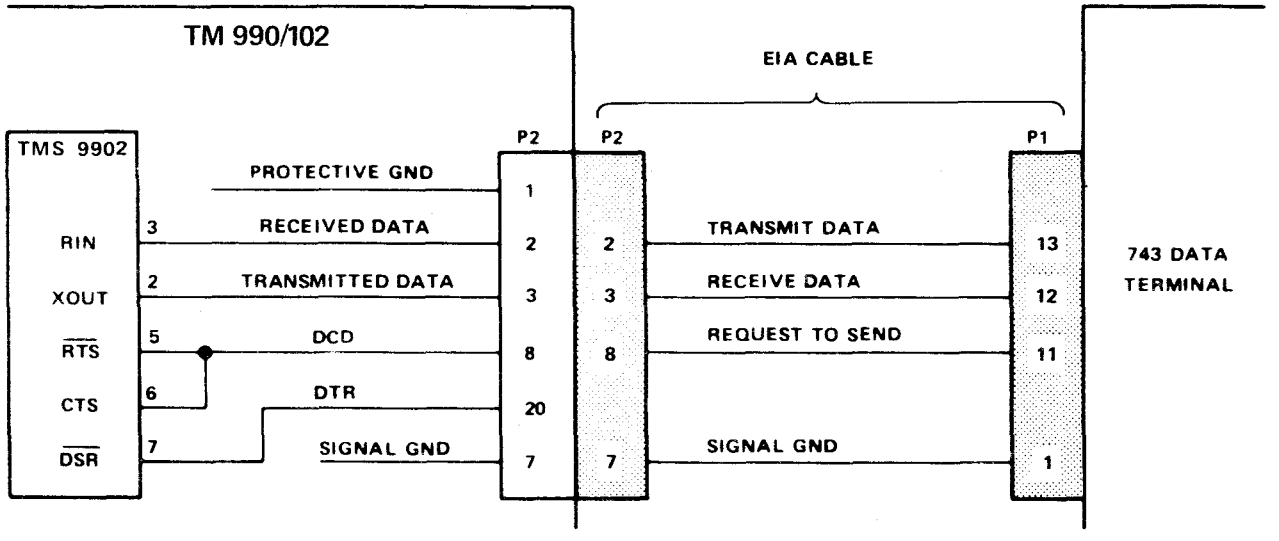
APPENDIX B

EIA RS-232-C CABLING

Figure B-1 shows the wiring for the 743 KSR cable attached between connector P2 on the TM 990/102 and a 743 KSR data terminal. Also shown is the relationship between cable wires and signals to the serial interface, the TMS 9902. Figure B-2 shows the cable configuration for the 733 data terminal. Figure B-3 shows the cable configuration for the 765 data terminal.

NOTE

If you want to make your own cable, be aware that the connector plugs of various vendors, including TI, do not necessarily use the numbering schemes on the board connector. ALWAYS refer to the board nomenclature when wiring a connector.



NOTE: Suggested EIA cable connectors (ITT Cannon or TRW Cinch):
P2: DB 25P
P1: DE-15S

FIGURE B-1. EIA RS-232-C CABLING FOR 743 DATA TERMINAL

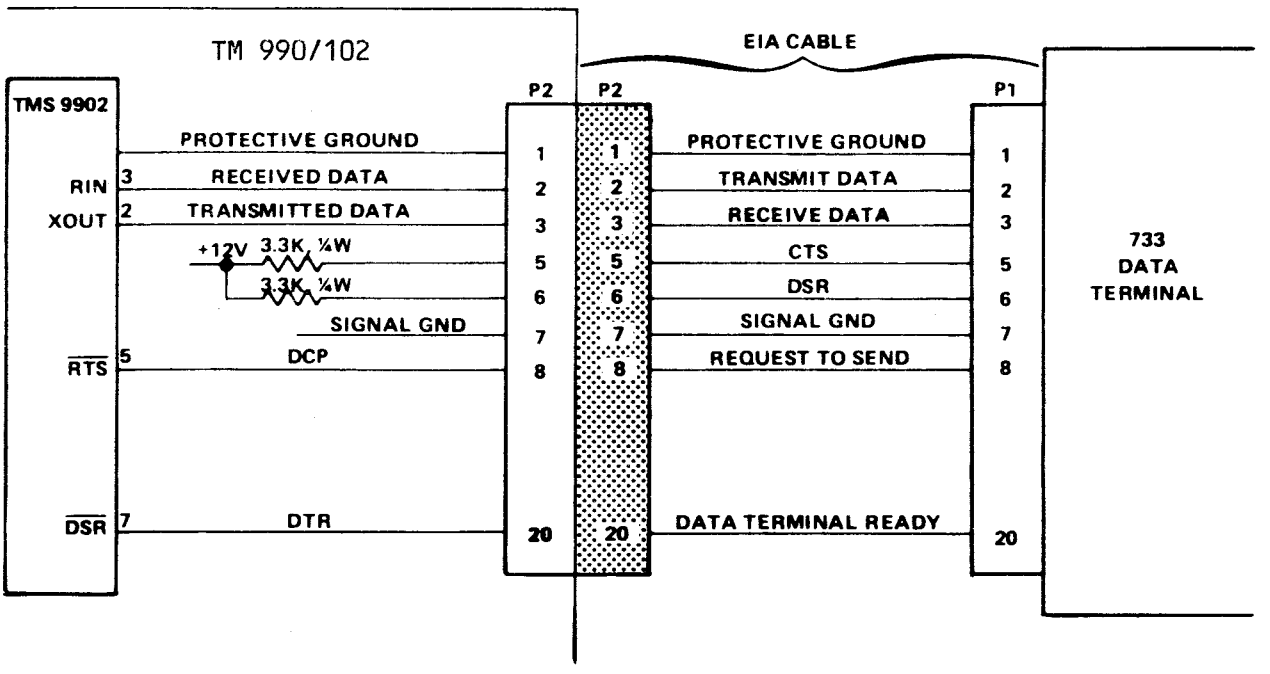


FIGURE B-2. EIA RS-232-C CABLING FOR 733 DATA TERMINAL

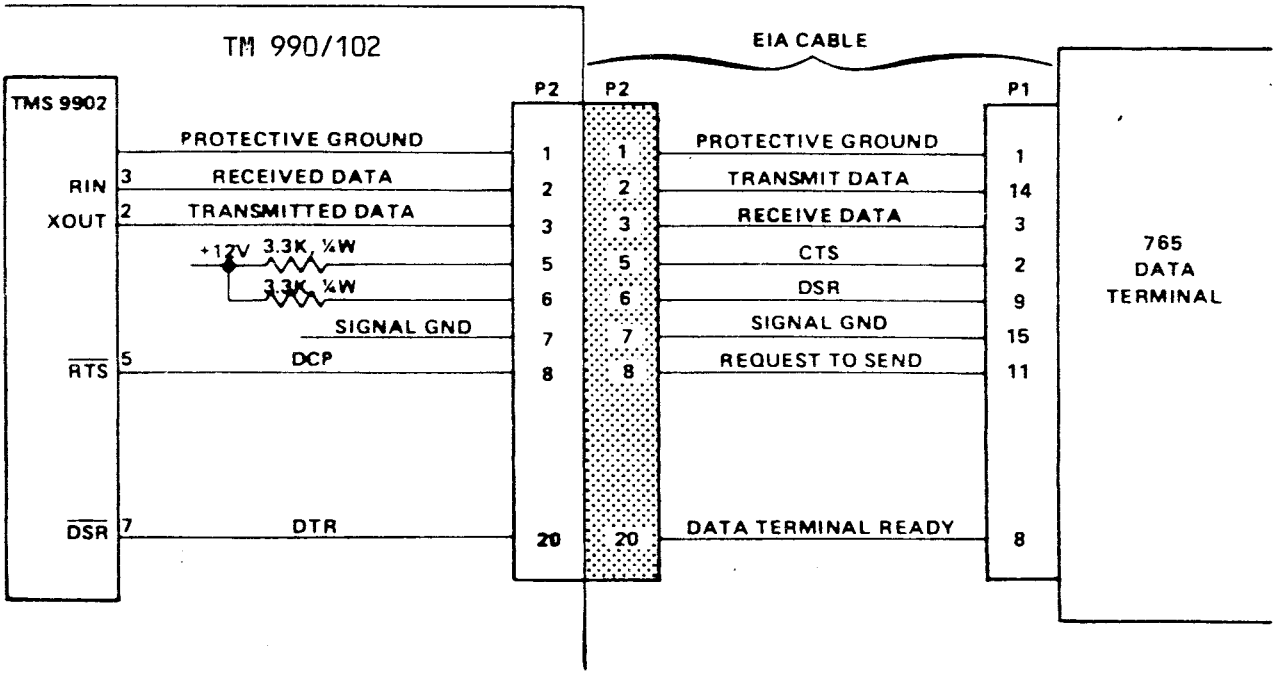


FIGURE B-3. EIA RS-232-C CABLING FOR 765 DATA TERMINAL

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

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, X X X, X X 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

$4 \times 10^0 = 4 \times 1 = 4$

$6 \times 10^1 = 6 \times 10 = 60$

$2 \times 10^2 = 2 \times 100 = 200$

$5 \times 10^3 = 5 \times 1000 = 5000$

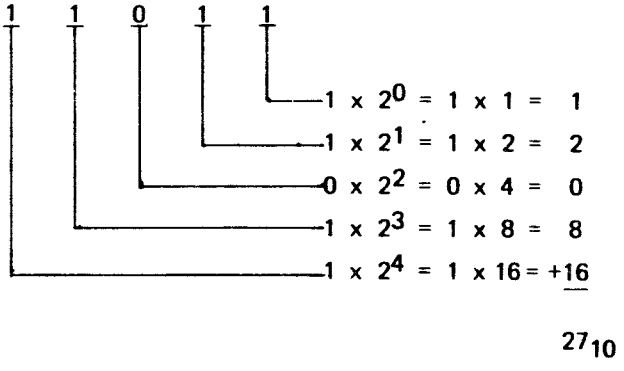
$7 \times 10^4 = 7 \times 10{,}000 = +70000$

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:

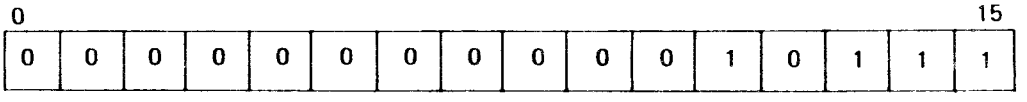
2^{15}		2^6	2^5	2^4	2^3	2^2	2^1	2^0
(32,768)	• • •	(64)	(32)	(16)	(8)	(4)	(2)	(1)
X	• • •	X	X	X	X	X	X	X

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:

7	B	A	5	
			$5 \times 16^0 = 5 \times 1$	= 5
			$10 \times 16^1 = 10 \times 16$	= 160
			$11 \times 16^2 = 11 \times 256$	= 2 816
			$7 \times 16^3 = 7 \times 4096$	= 28 672
				$31\,653_{10}$

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

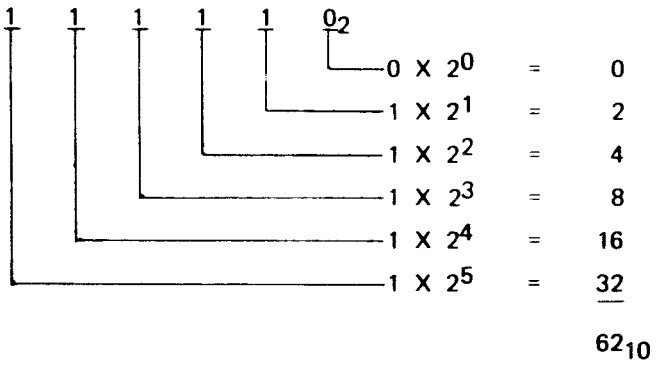
$003E_{16}$ or >003E (> indicates hexadecimal)

is used instead of

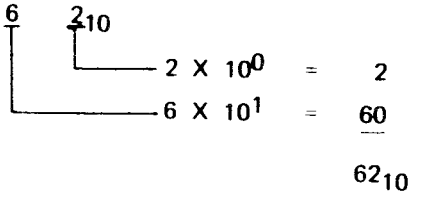
$0000\ 0000\ 0011\ 1110_2$

to represent 62_{10} as computed below:

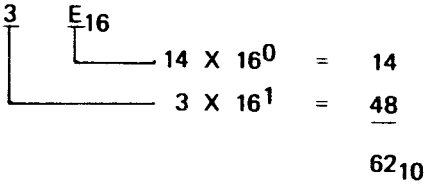
BASE 2



BASE 10



BASE 16



Note that separating the 16 binary bits into four-bit parts facilitates recognition and translation into hexadecimal.

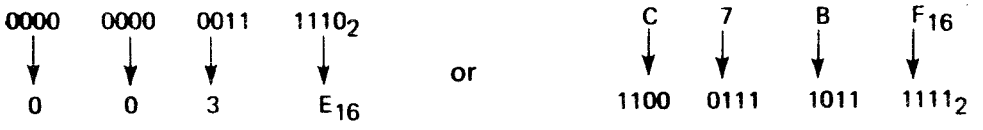


Table D-1 is a conversion chart for converting decimal to hexadecimal and vice versa. Table D-2 shows binary, decimal and hexadecimal equivalents for numbers 0 to 15. Note that Table D-1 is divided into four parts, each part representing four of the 16-bits of a memory cell or word (bits 0 to 15 with 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
+ 1    + 1
---    ---
  1     10   (the carry, 1, is carried to the left)
```

```
  10
 - 1
 ---
  01   (1 is borrowed from
        top left)
```

```
   1 }
     }  = 0 + carry 1
   1 }
 + 1    = 0 (from above) + 1 = 1
 ---
  11
  └── carry
```

```
   11
    1
  + 1
  ---
  101
carry 1 + 1 = 10 ──┘
```

```
  1 }
    }  = 0 + 1 carry
  1 }
  1 }
    }  = 0 + 1 carry
+ 1 }
---
100
│└── 0 + 0 = 0
└── carry 1 + carry 1
```

```
                        1
1000 }               { 0110
 - 1 }  Borrow the 1 {  - 1
 ---                 {  ---
0111 }               { 0111
```

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

PARTS LIST

Symbol	Description	-0001	-0002	-0003
C1, C11	Capacitor, 0.047 uF	x	x	x
C2, C7, C18, C20, C30	Capacitor, 22 uF	x	x	x
C4-C6, C8, C12-C17, C21-C28, C31	Capacitor, 0.047 uF	x	x	x
C9	Capacitor, 1.0 uF	x	x	x
C10	Capacitor, 0.1 uF	x	x	x
C19	Capacitor, 18 pF	x	x	x
CR1	Diode, IN914B	x	x	x
DS1, DS2	Diode, TIL220, LED	x	x	x
L1	Coil, RF, 0.33 uH	x	x	x
P2	Connector, 25-pin, EIA	x	x	x
R1, R6, R11, R12, R15, R16	Resistor, 10.0K ohm	x	x	x
R2, R3	Resistor, 68.0 ohm	x	x	x
R4, R8	Resistor, 1.0K ohm	x	x	x
R17-R20	Resistor, 16 ohm, $\frac{1}{4}$ W	x	x	x
R21	Resistor, 4.7K ohm	x	x	x
R23	Resistor, 3.9K ohm	x	x	x
R24, R27	Resistor, 330 ohm, $\frac{1}{4}$ W	x	x	x
R25, R26	Resistor, 3.3K ohm	x	x	x
U1, U6, U13, U17	Resistor, 10K SIP, 10 pins	x	x	x
U2	TMS 9901	x	x	x
U3, U7	IC, SN74LS244	x	x	x
U4, U5, U15, U16, U18	IC, SN74LS245	x	x	x
U8	Resistor, 820 SIP, 6 pins	x	x	x
U9	Resistor, 560 SIP, 6 pins	x	x	x
U11, U19	Resistor, 10K SIP, 6 pins	x	x	x

PARTS LIST, Cont.

Symbol	Description	-0001	-0002	-0003
U14, U30, U27	IC, SN74LS682	x	x	x
U20, U36, U39, U40, U41	IC, SN74LS74	x	x	x
U21	TMS 9900	x	x	x
U22	IC, SN74LS27	x	x	x
U23	TMS 4500 RAM Controller		x	x
U24	IC, SN74LS612	x	x	x
U25	IC, SN74LS138	x	x	x
U26	IC, SN74LS00	x	x	x
U28	IC, SN74S32	x	x	x
U29	IC, SN74S08	x	x	x
U31	IC, SN74LS20	x	x	x
U32	IC, SN74S03	x	x	x
U33	TMS 9902	x	x	x
U34	IC, SN74LS86	x	x	x
U35	IC, SN74LS00		x	x
U37, U45	IC, SN74LS04	x	x	x
U38	IC, SN74S74	x	x	x
U42	IC, 75188	x	x	x
U43	TIM 9904A	x	x	x
U44	IC, SN74LS32	x	x	x
U46	IC, SN74LS08	x	x	x
U47	IC, 75189AN	x	x	x
U48	IC, TL7705	x	x	x
U49-U64	TMS 4532-20		x	
U49-U64	TMS 4164-20			x
VR1	Regulator, MC79L05	x	x	x
XU2, XU23, XU24	Socket, 40 pin	x	x	x

PARTS LIST, Cont.

Symbol	Description	-0001	-0002	-0003
XU10, XU12	Socket, 28 pin	x	x	x
XU21	Socket, 64 pin	x	x	x
XU33	Socket, 18 pin	x	x	x
XU43	Socket, 20 pin	x	x	x
XU49-XU64	Socket, 16 pin		x	x
Y1	Crystal, 48 MHz, Quartz HC-18U	x	x	x

APPENDIX F

P1 AND P2 PIN ASSIGNMENTS

TABLE F.1 CHASSIS INTERFACE CONNECTOR (P1) SIGNAL ASSIGNMENTS

P1 PIN	SIGNAL	P1 PIN	SIGNAL	P1 PIN	SIGNAL
33	D0.B	71	A14.B	12	$\overline{\text{INT13}}$.B
34	D1.B	72	A15.B	11	$\overline{\text{INT14}}$.B
35	D2.B	22	$\overline{\text{Ø1}}$.B (BUSCLK-)	14	$\overline{\text{INT15}}$.B
36	D3.B	24	$\overline{\text{Ø3}}$.B (REFCLK-)	28	EXTCLK.B
37	D4.B	92	$\overline{\text{HOLD}}$.B	3	+5V
38	D5.B	86	HOLDA.B	4	+5V
39	D6.B	82	DBIN.B	97	+5V
40	D7.B	26	$\overline{\text{CLK}}$.B	98	+5V
41	D8.B	80	$\overline{\text{MEMEN}}$.B	75	+12V
42	D9.B	84	$\overline{\text{MEMCYC}}$.B	76	+12V
43	D10.B	78	$\overline{\text{WE}}$.B	73	−12V
44	D11.B	90	READY.B	74	−12V
45	D12.B	87	$\overline{\text{CRUCLK}}$.B	1	GND
46	D13.B	30	CRUOUT.B	2	GND
47	D14.B	29	CRUIN.B	21	GND
48	D15.B	19	IAQ.B	23	GND
57	A0.B	94	$\overline{\text{PRES}}$.B	25	GND
58	A1.B	88	$\overline{\text{IORST}}$.B	27	GND
59	A2.B	16	$\overline{\text{INT1}}$.B	31	GND
60	A3.B	13	$\overline{\text{INT2}}$.B	77	GND
61	A4.B	15	$\overline{\text{INT3}}$.B	79	GND
62	A5.B	18	$\overline{\text{INT4}}$.B	81	GND
63	A6.B	17	$\overline{\text{INT5}}$.B	83	GND
64	A7.B	20	$\overline{\text{INT6}}$.B	85	GND
65	A8.B	6	$\overline{\text{INT7}}$.B	89	GND
66	A9.B	5	$\overline{\text{INT8}}$.B	91	GND
67	A10.B	8	$\overline{\text{INT9}}$.B	99	GND
68	A11.B	7	$\overline{\text{INT10}}$.B	100	GND
69	A12.B	10	$\overline{\text{INT11}}$.B	93	$\overline{\text{RESTART}}$.B
70	A13.B	9	$\overline{\text{INT12}}$.B	95-96	CONNECTED TOGETHER (GRANTIN/GRANTOUT)

NOTE

If you want to make your own cable, be aware that the connector plugs of various vendors, including TI, do not necessarily use the numbering schemes on the board edge connector. ALWAYS refer to the board edge when wiring a connector.

TABLE F-2. SERIAL I/O INTERFACE (P2) PIN ASSIGMENTS

P2	SIGNAL	DESCRIPTION
1	GND	
7	GND	
3	RS232 XMT	RS232 Serial Data Out
2	RS232 RCV	RS232 Serial Data In
5	CTS	Clear to Send (3.3 KΩ pull-up to +12 V)
6	DSR	Data Set Ready (3.3 KΩ pull-up to +12 V)
8	DCD	Carrier Detect
20	DTR	Data Terminal Ready

APPENDIX G

990 OBJECT CODE FORMAT

G.1 GENERAL

In order to correctly load a program into memory using a loader, the program in hexadecimal machine code must be in a particular format called object format.This object format has a tag character for each 16-bit word of coding which flags the loader to perform one of several operations. These operations include:

- Load the code at a user-specified absolute address and resolve relative addresses. (Most assemblers assemble a program as if it was loaded at memory address 0000_{16}; thus, relative addresses have to be resolved.)
- Load entire program at a specific address.
- Set the program counter to the entry address after loading.
- Check for checksum errors that would indicate a data error in an object record.

G.2 STANDARD 990 OBJECT CODE

Standard 990 object code consists of a string of hexadecimal digits, each representing four bits, as shown in Figure G-1.

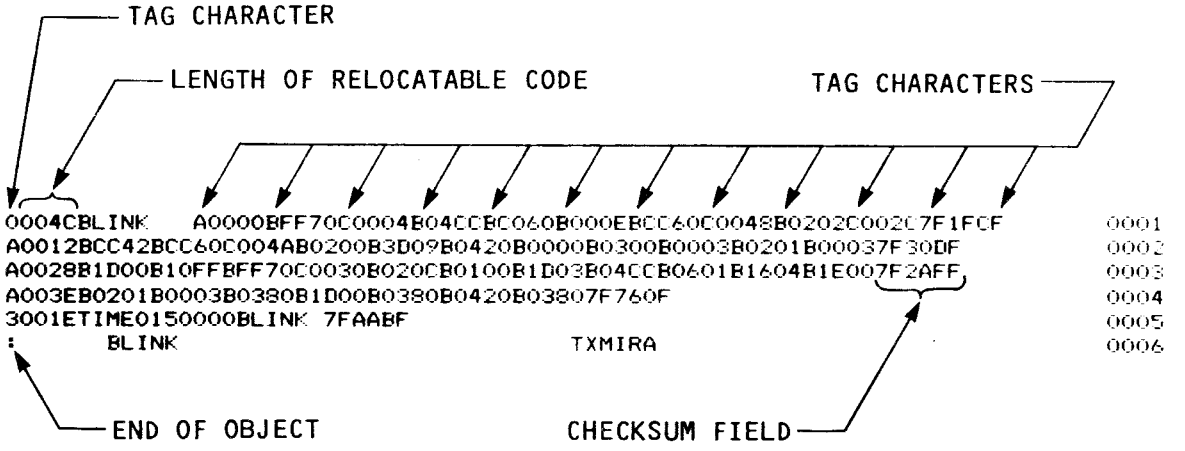


FIGURE G-1. OBJECT CODE EXAMPLE

The object record consists of a number of tag characters, each followed by one or two fields as defined in Table G-1. The first character of a record is the first tag character, which tells the loader which field or pair of fields follows the tag. The next tag character follows the end of the field or pair of fields associated with the preceding tag character. When the assembler has no more data for the record, the assembler writes the tag character 7 followed by the checksum field, and the tag character F, which requires no fields. The assembler then fills the rest of the record with blanks, and begins a new record with the appropriate tag character.

Tag character 0 is followed by two fields. The first field contains the number of bytes of relocatable code, and the second field contains the program identifier assigned to the program by an IDT assembler directive. When no IDT directive is entered, the field contains blanks. The loader uses the program identifier to identify the program, and the number of bytes of relocatable code to determine the load bias for the next module or program. The PX9ASM assembler is unable to determine the value for the first field until the entire module has been assembled, so PX9ASM places a tag character 0 followed by a zero field and the program identifier at the beginning of the object code file. At the end of the file, PX9ASM places another tag character zero followed by the number of bytes of relocatable code and eight blanks.

Tag characters 1 and 2 are used with entry addresses. Tag character 1 is used when the entry address is absolute. Tag character 2 is used when the entry address is relocatable. The hexadecimal field contains the entry address. One of these tags may appear at the end of the object code file. The associated field is used by the loader to determine the entry point at which execution starts when the loading is complete.

Tag characters 3 and 4 are used for external references. Tag character 3 is used when the last appearance of the symbol in the second field is in relocatable code. Tag character 4 is used when the last appearance of the symbol is absolute code. The hexadecimal field contains the location of the last appearance. The symbol in the second field is the external reference. Both fields are used by the linking loader to provide the desired linking to the external reference.

For each external reference in a program, there is a tag character in the object code, with a location, or an absolute zero, and the symbol that is referenced. When the object code field contains absolute zero, no location in the program requires the address that corresponds to the reference (an IDT character string, for example). Otherwise, the address corresponding to the reference will be placed in the location specified in the object code by the linking loader. The location specified in the object code similarly contains absolute zero or another location. When it contains absolute zero, no further linking is required. When it contains a location, the address corresponding to the reference will be placed in that address by the linking loader. The location of each appearance of a reference in a program contains either an absolute zero or another location into which the linking loader will place the referenced address.

TABLE G-1. OBJECT OUTPUT TAGS SUPPLIED BY ASSEMBLERS

TAG CHARACTER	HEXADECIMAL FIELD (FOUR CHARACTERS)	SECOND FIELD	MEANING
0	Length of all relocatable code	8-character program identifier	Program start
1	Entry address	None	Absolute entry address
2	Entry address	None	Relocatable entry address
3	Location of last appearance of symbol	6-character symbol	External reference last used in relocatable code
4	Location of last appearance of symbol	6-character symbol	External reference last used in absolute code
5	Location	6-character symbol	Relocatable external definition
6	Location	6-character symbol	Absolute external definition
7	Checksum for current record	None	Checksum
8	Ignore checksum	None	Do not checksum for error
9	Load address	None	Absolute load address
A	Load address	None	Relocatable load address
B	Data	None	Absolute data
C	Data	None	Relocatable data
D	Load bias value*	None	Load point specifier
F	None	None	End-of-record
G	Location	6-character symbol	Relocatable symbol definition
H	Location	6-character symbol	Absolute symbol definition

*Not supplied by assembler

Tag characters 5 and 6 are used for external definitions. Tag character 5 is used when the location is relocatable. Tag character 6 is used when the location is absolute. Both fields are used by the linking loader to provide the desired linking to the external definition. The second field contains the symbol of the external definition.

Tag character 7 precedes the checksum, which is an error detection word. The checksum is formed as the record is being written. It is the 2's complement of the sum of the 8-bit ASCII values of the characters of the record from the first tag of the record through the checksum tag 7. If the tag character 7 is replaced by an 8, the checksum will be ignored. The 8 tag can be used when object code is changed in editing and it is desired to ignore checksum.

Tag characters 9 and A are used with load addresses for data that follows. Tag character 9 is used when the load address is absolute. Tag character A is used when the load address is relocatable. The hexadecimal field contains the address at which the following data word is to be loaded. A load address is required for a data word that is to be placed in memory at some address other than the next address. The load address is used by the loader.

Tag characters B and C are used with data words. Tag character B is used when the data is absolute; an instruction word or a word that contains text characters or absolute constants, for example. Tag character C is used for a word that contains a relocatable address. The hexadecimal field contains the data word. The loader places the word in the memory location specified in the preceding load address field, or in the memory location that follows the preceding data word.

To have object code loaded at a specific memory address, precede the object program with the D tag followed by the desired memory address (e.g., DFD00).

Tag character F indicates the end of record. It may be followed by blanks.

Tag characters G and H are used when the symbol table option is specified with other 990 assemblers. Tag character G is used when the location or value of the symbol is relocatable, and tag character H is used when the location or value of the symbol is absolute. The first field contains the location or value of the symbol, and the second field contains the symbol to which the location is assigned.

The last record of an object code file has a colon (:) in the first character position of the record, followed by blanks. This record is referred to as an end-of-module separator record.

Figure G-2 is an example of an assembler source listing and corresponding object code. A comparison of the object tag characters and fields with the machine code in the source listing will show how object code is constructed for use by the loader.

SOURCE STATEMENT NO.

LOCATION COUNTER (ADDRESS RELATIVE TO FIRST OBJECT BYTE)

MACHINE CODE

```
SAMPLE          SDSMAC 945278 **
                                                                  PAGE 0001

0001                      IDT   'SAMPLE'
0002 0000 0006'           DATA  WSPACE
0003 0002 008A'           DATA  START
0004 0004 0000            DATA  0
0005 0006         WSPACE  BSS   32
0006 0026         TABLE   BSS   100
0007 008A         START
0008 008A 04CC            CLR   12
0009 008C 04C0            CLR   0
0010 008E 0202            LI    2,TABLE
     0090 0026'
0011 0092 C800            MOV   0,@TABLE+2
     0094 0028'
0012 0096 1001            JMP   $+4
0013 0098         LOOP
0014 0098 0204            LI    4,>1234
     009A 1234
0015 009C 0244            ANDI  4,>FEED
     009E FEED
0016 00A0 DC84            MOVB  4,*2+
0017 00A2 0205            LI    5,>5555
     00A4 5555
0018 00A6 C805            MOV   5,@TABLE
     00A8 0026'
0019                      END
NO ERRORS


000AASAMPLE   A0000C0006C008AB0000A008AB04CCB04C0B0202C0026BC8007F200F         000
C0028B1001B0204B1234B0244BFEEDBDC84B0205B5555BC805C00267F3C1F                  000
:        SAMPLE      00/00/00   08:14:23                   SDSMAC 945278 **
```

FIGURE G-2. SOURCE CODE AND CORRESPONDING OBJECT CODE

APPENDIX H

MEMORY MAPPER DATA SHEET,

TMS 74LS610 THROUGH 74LS613

(The 74LS612 is the memory mapper used on the TM 990/102)

TYPES SN54LS610 THRU SN54LS613, SN74LS610 THRU SN74LS613 MEMORY MAPPERS

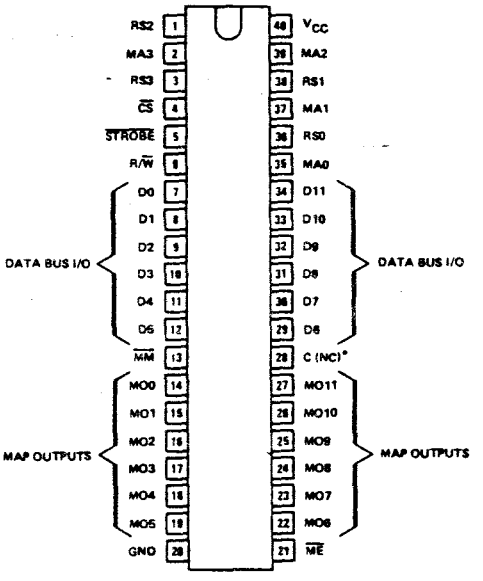
D2549, JANUARY 1981

(TIM99610 THRU TIM99613)

- Expands 4 Address Lines to 12 Address Lines
- Designed for Paged Memory Mapping
- Output Latches Provided on 'LS610 and 'LS611
- Choice of 3-State or Open-Collector Map Outputs
- Compatible with TMS 9900 and Other Microprocessors

DEVICE	OUTPUTS LATCHED	MAP OUTPUT TYPE
'LS610	Yes	3-State
'LS611	Yes	Open-Collector
'LS612	No	3-State
'LS613	No	Open-Collector

SN54LS' . . . J PACKAGE
SN74LS' . . . J OR N PACKAGE
(TOP VIEW)



*NOTE: Pin 28 has no internal connection on 'LS612 and 'LS613

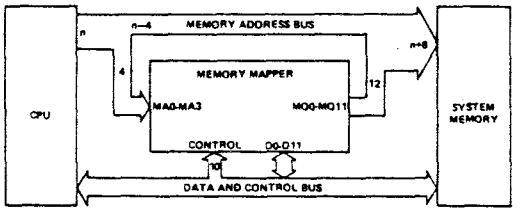
description

These memory-mapper integrated circuits contain a 4-line to 16-line decoder, a 16-word by 12-bit RAM, 16 channels of 2-line to 1-line multiplexers, and other miscellaneous circuitry on a monolithic chip. The 'LS610 and 'LS611 also contain 12 latches with an enable control.

The memory mappers are designed to expand a microprocessor's memory address capability by eight bits. Four bits of the memory address bus (see the figure below) can be used to select one of 16 map registers that contain 12 bits each. These 12 bits are presented to the system memory address bus through the map output buffers along with the unused memory address bits from the CPU. However, addressable memory space without reloading the map registers is the same as would be available with the memory mapper left out. The addressable memory space is increased only by periodically reloading the map registers from the data bus.

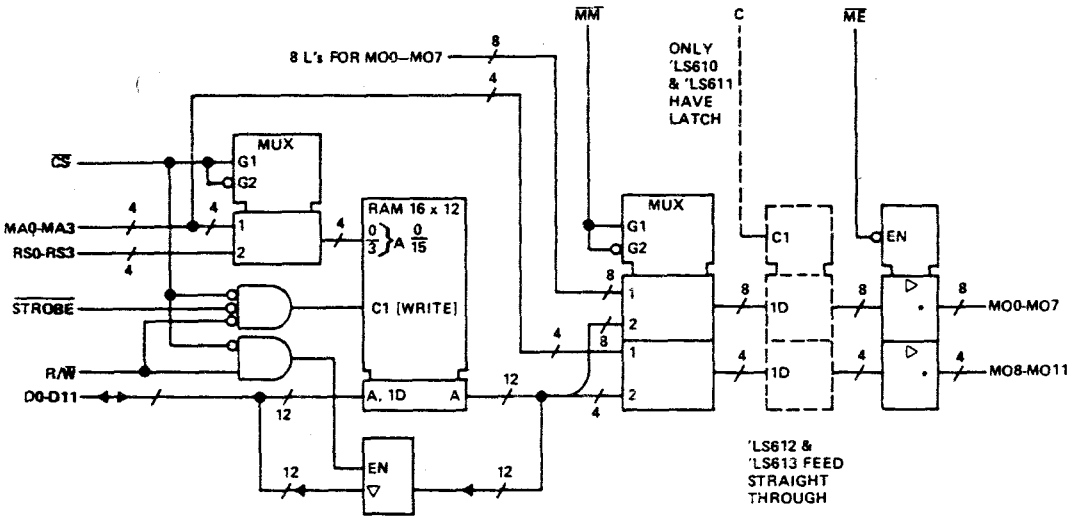
This configuration lends itself to memory utilization of 16 pages of $2^{(n-4)}$ registers each without reloading (n = number of address bits available from CPU).

These devices have four modes of operation (read, write, map, and pass). Data may be read from or loaded into the map register selected by the register select inputs (RS0 thru RS3) under control of R/$\overline{W}$ whenever chip select ($\overline{CS}$) is low. The data I/O takes place on the data bus D0 thru D7. The map operation will output the contents of the map register selected by the map address inputs (MA0 thru MA3) when $\overline{CS}$ is high and $\overline{MM}$ (map mode control) is low. The 'LS612 and 'LS613 output stages are transparent in this mode, while the 'LS610 and 'LS611 outputs may be transparent or latched. When $\overline{CS}$ and $\overline{MM}$ are both high (pass mode), the address bits on MA0 thru MA3 appear at MO8–MO11, respectively, (assuming appropriate latch control) with low levels in the other bit positions of the map outputs.



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

functional block diagram (positive logic)



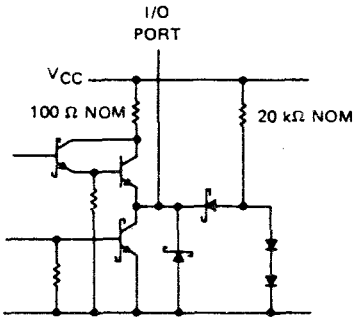
*'LS610 and 'LS612 have 3-state (▽) map outputs.
'LS611 and 'LS613 have open-collector (Ω) map outputs.

PIN FUNCTION TABLE

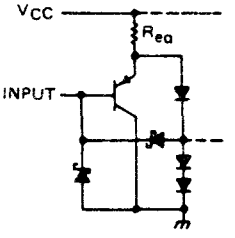
PIN	PIN NAME	FUNCTIONAL DESCRIPTION
7–12, 29–34	D0 thru D11	I/O connections to data and control bus used for reading from and writing to the map register selected by RS0–RS3 when $\overline{CS}$ is low. Mode controlled by R/$\overline{W}$.
36, 38, 1, 3	RS0 thru RS3	Register select inputs for I/O operations.
6	R/$\overline{W}$	Read or write control used in I/O operations to select the condition of the data bus. When high, the data bus outputs are active for reading the map register. When low, the data bus is used to write into the register.
5	$\overline{STROBE}$	Strobe input used to enter data into the selected map register during I/O operations.
4	$\overline{CS}$	Chip select input. A low input level selects the memory mapper (assuming more than one used) for an I/O operation.
35, 37, 39, 2	MA0 thru MA3	Map address inputs to select one of 16 map registers when in map mode ($\overline{MM}$ low and $\overline{CS}$ high).
14–19, 22–27	MO0 thru MO11	Map outputs. Present the map register contents to the system memory address bus in the map mode. In the pass mode, these outputs provide the map address data on MO8–MO11 and low levels on MO0–MO7.
13	$\overline{MM}$	Map mode input. When low, 12 bits of data are transferred from the selected map register to the map outputs. When high (pass mode), the 4 bits present on the map address inputs MA0-MA3 are passed to the map outputs MO8-MO11, respectively, while MO0-MO7 are set low.
21	$\overline{ME}$	Map enable for the map outputs. A low level allows the outputs to be active while a high input level puts the outputs at high impedance.
28	C	Latch enable input for the 'LS610 and 'LS611 (no internal connection for 'LS612 and 'LS613). A high level will transparently pass data to the map outputs. A low level will latch the outputs.
40, 20	V_{CC}, GND	5-V power supply and network ground (substrate) pins.

schematics of inputs and outputs

INPUT/OUTPUT PORTS, D0-D11

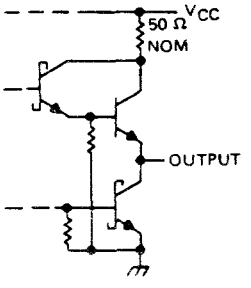


EQUIVALENT OF OTHER INPUTS

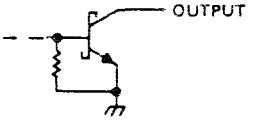


$\overline{MM}$: R_{eq} = 7 kΩ NOM
RS, $\overline{STROBE}$: R_{eq} = 9 kΩ NOM
$\overline{CS}$, R/$\overline{W}$, MA: R_{eq} = 6 kΩ nom

TYPICAL OF MAP OUTPUTS ('LS610, 'LS612)



TYPICAL OF MAP OUTPUTS ('LS611, 'LS613)



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

Supply voltage, V_{CC} (see Note 1)		7 V
Input voltage:	Data Bus I/O	5.5 V
	All other inputs	7 V
Operating free-air temperature range:	SN54LS610 through SN54LS613	−55°C to 125°C
	SN74LS610 through SN74LS613	0°C to 70°C
Storage temperature range		−65°C to 150°C

NOTE 1: Voltage values are with respect to network ground terminal.

recommended operating conditions

		SN54LS610 SN54LS612			SN74LS610 SN74LS612			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}		4.5	5	5.5	4.75	5	5.25	V
High-level output current, I_{OH}	MO			−12			−15	mA
	D			−1			−2.6	
Low-level output current, I_{OL}	MO			12			24	mA
	D			4			8	
Width of strobe input pulse, t_{SLSH}	See Figure 1	75			75			ns
$\overline{CS}$ setup time ($\overline{CS}$ low to strobe low), t_{CSLSL}		20			20			ns
R/$\overline{W}$ setup time (R/$\overline{W}$ low to strobe low), t_{WLSL}		20			20			ns
RS setup time (RS valid to strobe low), t_{RVSL}		20			20			ns
Data setup time (D0-D11 valid to strobe high), t_{DVSH}		75			75			ns
$\overline{CS}$ hold time (Strobe high to $\overline{CS}$ high), t_{SHCSH}		20			20			ns
R/$\overline{W}$ hold time (Strobe high to R/$\overline{W}$ high), t_{SHWH}		20			20			ns
RS hold time (Strobe high to RS invalid), t_{SHRX}		20			20			ns
Data hold time (Strobe high to D0-D11 invalid), t_{SHDX}		20			20			ns
Operating free-air temperature, T_A		−55		125	0		70	°C

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

PARAMETER			TEST CONDITIONS†		SN54LS610 SN54LS612			SN74LS610 SN74LS612			UNIT
					MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage				2			2			V
V_{IL}	Low-level input voltage						0.7			0.8	V
V_{IK}	Input clamp voltage		V_{CC} = MIN, I_I = −18 mA				−1.5			−1.5	V
V_{OH}	High-level output voltage	MO	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max	I_{OH} = −3 mA	2.4			2.4			V
				I_{OH} = MAX	2			2			
		D		I_{OH} = MAX	2.4			2.4			
V_{OL}	Low-level output voltage	MO	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max	I_{OL} = 12 mA		0.25	0.4		0.25	0.4	V
				I_{OL} = 24 mA					0.35	0.5	
		D		I_{OL} = 4 mA		0.25	0.4		0.25	0.4	
				I_{OL} = 8 mA					0.35	0.5	
I_{OZH}	Off-state output current, high-level voltage applied		V_{CC} = MAX, V_{IH} = 2 V, V_{IL} = V_{IL} max, V_O = 2.7 V				20			20	μA
I_{OZL}	Off-state output current, low-level voltage applied	MO	V_{CC} = MAX, V_{IH} = 2 V, V_{IL} = V_{IL} max, V_O = 0.4 V				−20			−20	μA
		D					−400			−400	
I_I	Input current at maximum input voltage	D	V_{CC} = MAX	V_I = 5.5 V			100			100	μA
		All others		V_I = 7 V			100			100	
I_{IH}	High-level input current		V_{CC} = MAX, V_I = 2.7 V				20			20	μA
I_{IL}	Low-level input current		V_{CC} = MAX, V_I = 0.4 V				−0.4			−0.4	mA
I_{OS}	Short-circuit output current§	MO	V_{CC} = MAX		−40		−225	−40		−225	mA
		D			−30		−130	−30		−130	
I_{CC}	Supply current		V_{CC} = MAX	Outputs high		112	180		112	180	mA
				Outputs low		112	180		112	180	
				Outputs at high impedance		150	230		180	230	

†For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡All typical values are at V_{CC} = 5 V, T_A = 25°C.
§Note more than one output should be shorted at a time, and duration of the short-circuit should not exceed one second.

switching characteristics, V_{CC} = 5 V, T_A = 25°C, C_L = 45 pF to GND

PARAMETER		FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	'LS610			'LS612			UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
t_{CSLDV}	Access (enable) time	$\overline{CS}$↓	D 0-11			28	50		26	50	ns
t_{WHDV}	Access (enable) time	R/$\overline{W}$↑	D 0-11	R_L = 2 kΩ		20	35		20	35	ns
t_{RVDV}	Access time	RS	D 0-11	See Figure 1,		49	75		39	75	ns
t_{WLDZ}	Disable time	R/$\overline{W}$↓	D 0-11	See Note 2		32	50		30	50	ns
t_{CSHDZ}	Disable time	$\overline{CS}$↑	D 0-11			42	65		38	65	ns
t_{ELQV}	Access (enable) time	$\overline{ME}$↓	MO 0-11			19	30		17	30	ns
t_{CSHQV}	Access time	$\overline{CS}$↑	MO 0-11			56	85		48	85	ns
t_{MLQV}	Access time	$\overline{MM}$↓	MO 0-11	R_L = 667 Ω,		25	40		22	40	ns
t_{CHQV}	Access time	C↑	MO 0-11	See Figure 2,		24	40				ns
t_{AVQV1}	Access time ($\overline{MM}$ low)	MA	MO 0-11	See Note 2		46	70		39	70	ns
t_{MHQV}	Access time	$\overline{MM}$↑	MO 0-11			24	40		22	40	ns
t_{AVQV2}	Propagation time ($\overline{MM}$ high)	MA	MO 8-11			19	30		13	30	ns
t_{EHQZ}	Disable time	$\overline{ME}$↑	MO 0-11			14	25		14	25	ns

NOTE 2 For load circuits and measurement points, see page 3-11 of *The TTL Data Book for Design Engineers*, second edition, LCC 4112. Access times are tested as t_{PLH} and t_{PHL} or t_{PZH} or t_{PZL}. Disable times are tested as t_{PHZ} and t_{PLZ}.

explanation of letter symbols

This data sheet uses a new type of letter symbol to describe time intervals. The format is:

$$t_{AB\text{-}CD}$$

where: subscripts A and C indicate the names of the signals for which changes of state or level or establishment of state or level constitute signal events assumed to occur first and last, respectively, that is, at the beginning and end of the time interval.

Subscripts B and D indicate the direction of the transitions and/or the final states or levels of the signals represented by A and C, respectively. One or two of the following is used:

H = high or transition to high
L = low or transition to low
V = a valid steady-state level
X = unknown, changing, or "don't care" level
Z = high-impedance (off) state.

The hyphen between the B and C subscripts is omitted when no confusion is likely to occur. For these letter symbols on this data sheet, the signal names are further abbreviated as follows:

SIGNAL NAME	B or D SUBSCRIPT
C	C
$\overline{CS}$	CS
D0–11	D
MA0–MA3	A
MO0–MO11	Q
$\overline{ME}$	E
$\overline{MM}$	M
R/$\overline{W}$	W
RS0–RS3	R
$\overline{STROBE}$	S

recommended operating conditions

		SN54LS611 SN54LS613			SN74LS611 SN74LS613			UNIT
		MIN	NOM	MAX	MIN	NOM	MAX	
Supply voltage, V_{CC}		4.5	5	5.5	4.75	5	5.25	V
High-level output voltage, V_{OH}	MO			5.5			5.5	V
High-level output current, I_{OH}	D			−1			−2.6	mA
Low-level output current, I_{OL}	MO			12			24	mA
	D			4			8	
Width of strobe input pulse, t_{SLSH}	See Figure 1	75			75			ns
$\overline{CS}$ setup time ($\overline{CS}$ low to strobe low), t_{CSLSL}		20			20			ns
R/$\overline{W}$ setup time (R/$\overline{W}$ low to strobe low), t_{WLSL}		20			20			ns
RS setup time (RS valid to strobe low), t_{RVSL}		20			20			ns
Data setup time (D0-D11 valid to strobe high), t_{DVSH}		75			75			ns
$\overline{CS}$ hold time (Strobe high to $\overline{CS}$ high), t_{SHCSH}		20			20			ns
R/$\overline{W}$ hold time (Strobe high to R/$\overline{W}$ high), t_{SHWH}		20			20			ns
RS hold time (Strobe high to RS invalid), t_{SHRX}		20			20			ns
Data hold time (Strobe high to D0-D11 invalid), t_{SHDX}		20			20			ns
Operating free-air temperature, T_A		−55		125	0		70	°C

NOTE 2: Voltage values are with respect to network ground terminal.

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

PARAMETER			TEST CONDITIONS†		SN54LS611 SN54LS613			SN74LS611 SN74LS613			UNIT
					MIN	TYP‡	MAX	MIN	TYP‡	MAX	
V_{IH}	High-level input voltage				2			2			V
V_{IL}	Low-level input voltage						0.7			0.8	V
V_{IK}	Input clamp voltage		V_{CC} = MIN, I_I = −18 mA				−1.5			−1.5	V
V_{OH}	High-level output voltage	D	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max, I_{OH} = MAX		2.4			2.4			
I_{OH}	High-level output current	MO	V_{CC} = MIN, V_{IH} = 2 V, V_{OH} = 5.5 V				100			100	µA
V_{OL}	Low-level output voltage	MO	V_{CC} = MIN, V_{IH} = 2 V, V_{IL} = V_{IL} max	I_{OL} = 12 mA		0.25	0.4		0.25	0.4	V
				I_{OL} = 24 mA					0.35	0.5	
		D		I_{OL} = 4 mA		0.25	0.4		0.25	0.4	
				I_{OL} = 8 mA					0.35	0.5	
I_{OZH}	Off-state output current, high-level voltage applied	D	V_{CC} = MAX, V_{IH} = 2 V, V_{IL} = V_{IL} max, V_O = 2.7 V				20			20	µA
I_{OZL}	Off-state output current, low-level voltage applied	D	V_{CC} = MAX, V_{IH} = 2 V, V_O = 0.4 V				−400			−400	µA
I_I	Input current at maximum input voltage	D	V_{CC} = MAX	V_I = 5.5 V			100			100	µA
		All others		V_I = 7 V			100			100	
I_{IH}	High-level input current		V_{CC} = MAX, V_I = 2.7 V				20			20	µA
I_{IL}	Low-level input current		V_{CC} = MAX, V_I = 0.4 V				−0.4			−0.4	mA
I_{OS}	Short-circuit output current§	D	V_{CC} = MAX		−30		−130	−30		−130	mA
I_{CC}	Supply current		V_{CC} = MAX	Outputs high		100	170		100	170	mA
				Outputs low		100	170		100	170	
				Outputs at high impedance		110	200		110	200	

†For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡All typical values are at V_{CC} = 5 V, T_A = 25°C.
§Not more than one output should be shorted at a time, and duration of the short-circuit should not exceed one second.

switching characteristics, V_{CC} = 5 V, T_A = 25°C, C_L = 45 pF to GND

PARAMETER		FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	'LS611			'LS613			UNIT
					MIN	TYP	MAX	MIN	TYP	MAX	
t_{CSLDV}	Access (enable) time	$\overline{CS}$↓	D 0-11			31	50		28	50	ns
t_{WHDV}	Access (enable) time	R/$\overline{W}$↑	D 0-11	R_L = 2 kΩ,		23	35		21	35	ns
t_{RVDV}	Access time	RS	D 0-11	See Figure 1,		51	75		47	75	ns
t_{WLDZ}	Disable time	R/$\overline{W}$↓	D 0-11	See Note 2		32	50		31	50	ns
t_{CSHDZ}	Disable time	$\overline{CS}$↑	D 0-11			41	65		40	65	ns
t_{ELQV}	Access (enable) time	$\overline{ME}$↓	MO 0-11			21	30		19	30	ns
t_{CSHQV}	Access time	$\overline{CS}$↑	MO 0-11			57	90		53	90	ns
t_{MLQV}	Access time	MM↓	MO 0-11	R_L = 667 Ω,		25	40		25	40	ns
t_{CHQV}	Access time	C↑	MO 0-11	See Figure 2,		30	45				ns
t_{AVQV1}	Access time ($\overline{MM}$ low)	MA	MO 0-11	See Note 2		47	70		44	70	ns
t_{MHQV}	Access time	MM↑	MO 0-11			31	50		31	50	ns
t_{AVQV2}	Propagation time ($\overline{MM}$ high)	MA	MO 8-11			21	30		20	30	ns
t_{EHQZ}	Disable time	$\overline{ME}$↑	MO 0-11			15	25		15	25	ns

NOTE 2 For load circuits and measurement points, see page 3-11 of *The TTL Data Book for Design Engineers*, second edition, LCC 4112. Access times are tested as t_{PLH} and t_{PHL} or t_{PZH} or t_{PZL}. Disable times are tested as t_{PHZ} and t_{PLZ}.

TIMING DIAGRAMS

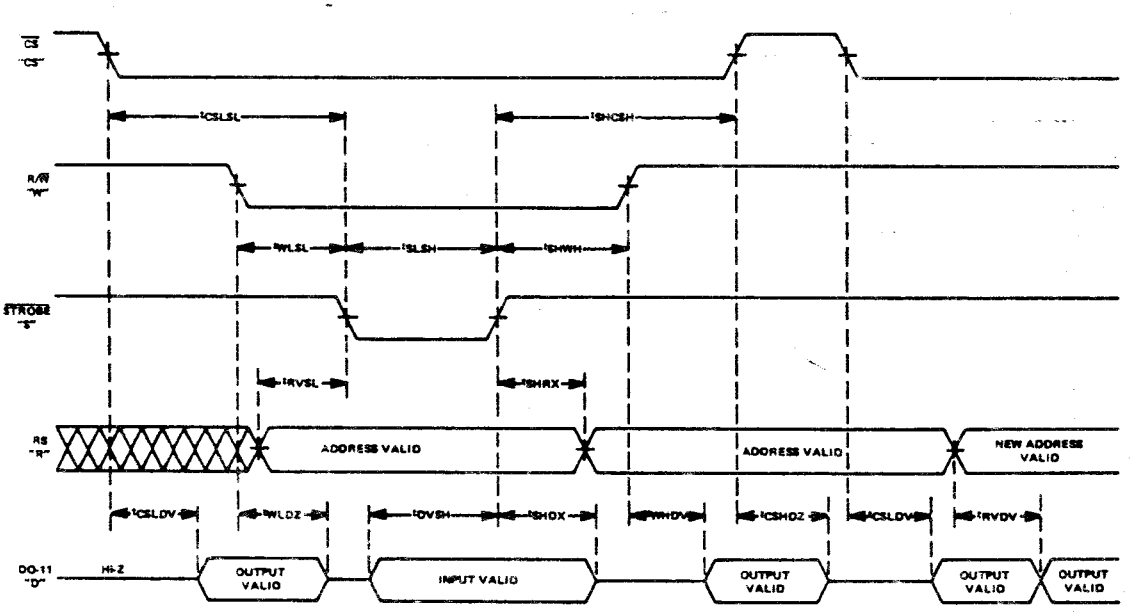


FIGURE 1–WRITE AND READ MODES

TIMING DIAGRAMS

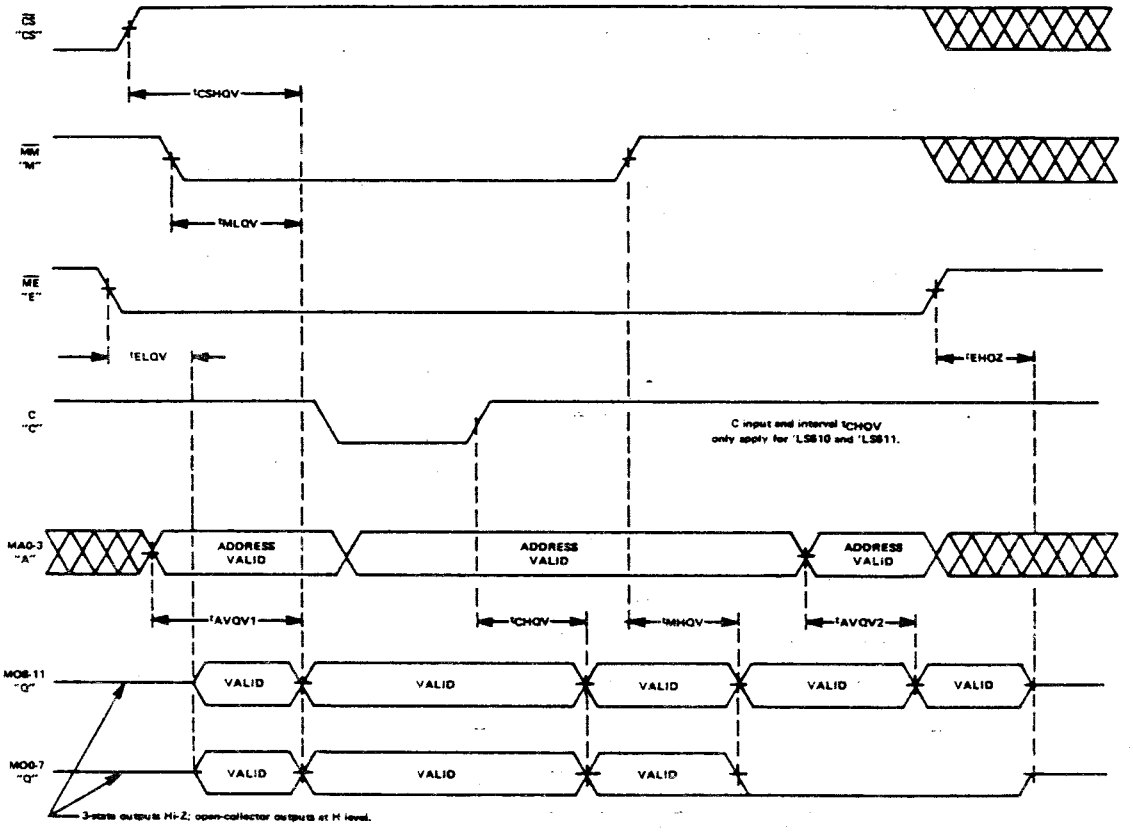


FIGURE 2–MAP AND PASS MODES

INDEX

Except for numbers preceded by a T or F, this index shows page numbers of different subjects. Subject matter is covered by a table if a T precedes the page number, or is covered by a figure if an F precedes the page number.