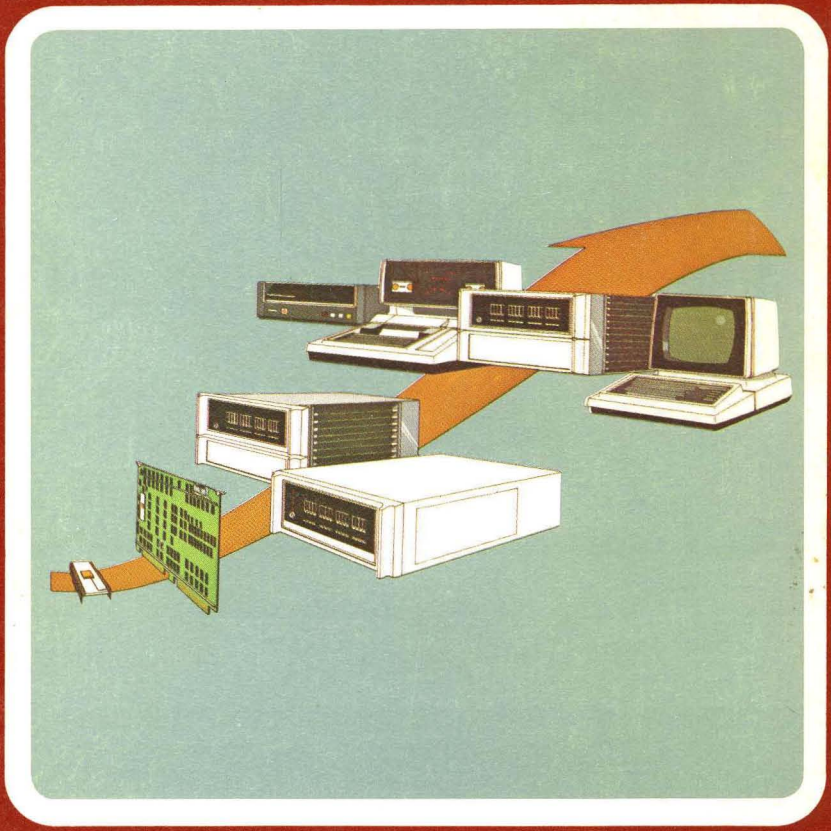
990

Computer Family
SYSTEMS HANDBOOK



TEXAS INSTRUMENTS
INCORPORATED

990

Computer Family
SYSTEMS HANDBOOK

MANUAL NO. 945250-9701

TEXAS INSTRUMENTS
INCORPORATED

3rd Edition, May 1976

Texas Instruments Incorporated
Digital Systems Division
P.O. Box 2909
Austin, Texas 78767

PREFACE

This is your copy of the Texas Instruments 990 Computer Family System Handbook. This handbook is designed to introduce the 990 to new users and also to function as a handy reference for 990 system designers and programmers. The 990 Computer Family represents the latest addition to the Texas Instruments 900 Series minicomputers. Ranging in power and cost from TMS 9900 microprocessor to the 990/10 minicomputer, the 990 Computer Family offers the user a software compatible selection with one or more well suited for his application.

Texas Instruments has been active for over a decade in all phases of digital computer design, manufacture, and use. These computers range in size from microprocessor to the large-scale Advanced Scientific Computer, and include products designed for business, scientific and industrial applications. In addition, TI produces a wide range of peripheral equipment.

TI is vertically integrated and serves its markets all the way from materials to products and services. This places TI in the unique position to be the price/performance leader among computer suppliers. The Texas Instruments Digital Systems Division (DSD) produces the 900 series minicomputers, computer systems, and in addition, peripherals such as terminals, tape transports, etc., scientific instruments, and seismic digital field systems.

The DSD computer products are manufactured in the TI Austin, Texas facility and we in Computer Systems look forward to providing you every assistance in solving your application requirements.

Frank Spitznogle

MANAGER, COMPUTER SYSTEMS

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

APPENDIXES

LIST OF ILLUSTRATIONS

LIST OF ILLUSTRATIONS (Continued)

LIST OF TABLES

SECTION I

INTRODUCTION

This Handbook was planned and written to provide you a valuable addition to your technical library.

Like the computer family it describes, the Handbook is designed for a wide variety of user requirements. The 990 Computer Family has the flexibility and power to perform the most sophisticated computational tasks. At the same time, these computers have been made as simple to program, operate, and maintain as possible.

Similarly, the material in this Handbook has been arranged so that the experienced programmer or systems engineer can refer directly to those sections that relate to his work, while the less experienced reader will find the book a valuable introduction to computer technology.

Since this Handbook is intended to function as a useful working tool, any comments or suggestions on the contents would be greatly appreciated. Please direct your remarks to the 990 System Handbook Editor, Texas Instruments, P.O. Box 2909 Mail Station 2107, Austin, Texas 78767.

1.1 SCOPE OF HANDBOOK

The purpose of the *990 Computer Family Systems Handbook* is to provide all the basic information necessary for system design, selection of standard 990 Computer hardware and software modules, and for system configuration. When used in conjunction with the *990 Computer Family Price List,* a budgetary estimate of system hardware cost can be determined.

This Handbook also provides detailed information on the 990 instructions and assembly language mnemonics as a handy reference for programmers. The Handbook can also serve as a training text for personnel who will be designing, programming, and operating systems incorporating the 990 Computers.

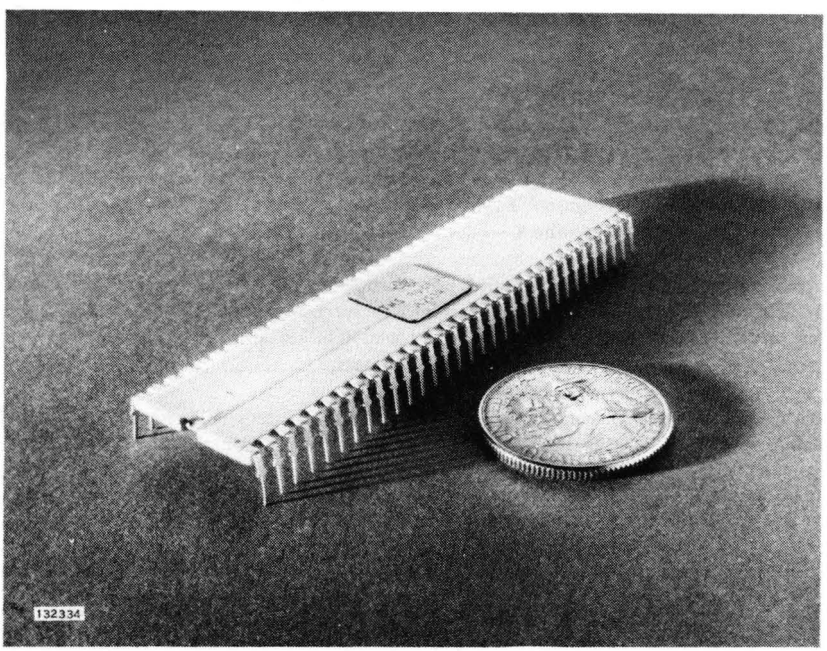
The final design, programming, and servicing of the system will require more detail than is presented in this Handbook. This information is contained in programming manuals, user manuals, and hardware maintenance manuals. The scope of these manuals is discussed in Section VII of this Handbook. Prices and manual numbers are listed in *990 Computer Family Price List.*

1.2 INTRODUCING THE 990 FAMILY

The Texas Instruments 990 Computer Family gives new meaning to the term, computer family. This new meaning is based upon a heritage of semiconductor leadership and is due to an important milestone in MOS technology – the TMS9900 single-chip, 16-bit microprocessor. This smallest member of the 990 family shares the same basic instruction set with its larger brothers, the 990/4 Microcomputer, and the 990/10 Minicomputer. This means that software developed for the TMS9900 will be compatible with the higher performance models. 990 users can expand their systems with a minimum of interface and software adaptation.

1.2.1 THE TMS9900 MICROPROCESSOR. The TMS9900 is a 16-bit, single-chip microprocessor using MOS N-channel silicon-gate technology. Its unique architecture permits data manipulation not easily achievable in earlier devices. With its repertoire of versatile instructions, hardware multiply/divide, and high-speed interrupt capability, the TMS9900 microprocessor provides computing power expected from a 16-bit TTL computer.

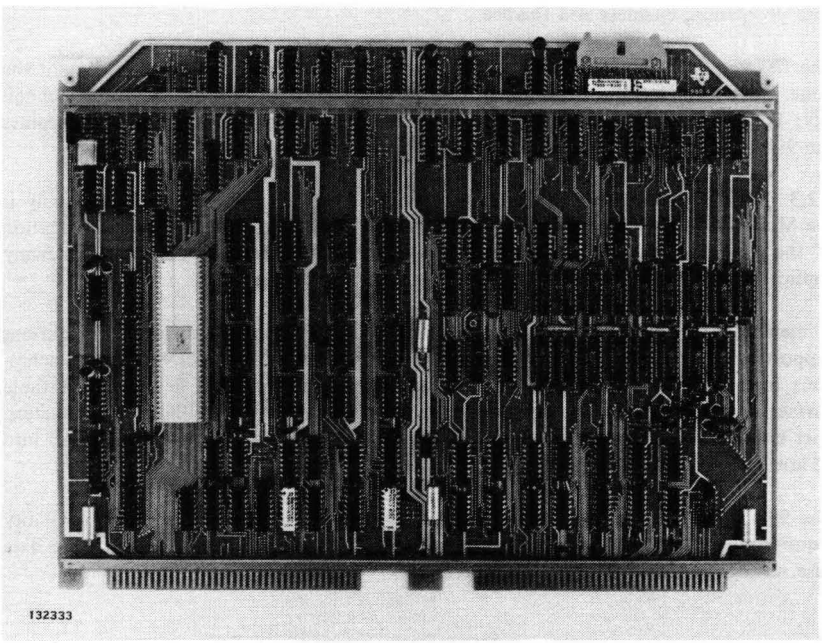
The TMS9900 microprocessor is well suited for a multitude of applications, and as replacements for minicomputers and hard-wired logic systems, or in new products which previously were not economically feasible. The primary thread in these applications will be where previous designs required 30 or more TTL integrated circuits,



TMS9900 Microprocessor

applications where the quantity to be produced numbers in the thousands, or in applications where miniaturization without loss of processing power is essential. The TMS9900 will be used primarily by original equipment manufacturers (OEM). The OEM systems include communication functions, controllers, terminals, point-of-sale terminals, and many new products. The TMS9900 opens up the world of industrial process control. Examples of typical applications include: machine-tool controllers, instrumentation, data-logging systems, and type setting. In these systems, the TMS9900 serves as a replacement for hard-wired logic as well as minicomputers. In process control the TMS9900 interfaces to a wide variety of sensors. It makes logic decisions, sends out control signals to motors, valves, switches, etc., and performs arithmetic functions such as linearization, scaling, and unit conversion.

The TMS9900 Microprocessor provides the basic building block for design into many different systems. To this building block must be added the circuitry for memory, control, system interfaces, and power supplies. The extent of added circuitry will be determined by the application. One application of the TMS9900 includes its mounting on a printed circuit board with an initial amount of memory, additional control and interface circuitry, and dc power input lines. This design becomes the 990/4 Microcomputer, the next member of the 990 Family.



990/4 Microcomputer

1.2.2 THE 990/4 MICROCOMPUTER. The 990/4 Microcomputer is a complete computer on a single printed circuit board using the TMS9900 as its central processor. In addition to the TMS9900 microprocessor, the 990/4 Microcomputer contains up to 4096 16-bit words of dynamic random access memory (RAM), 1024 16-bit words of either static RAM, read only memory (ROM), or programmable ROM (PROM), eight vectored interrupts, front panel interface, real-time clock input, two I/O buses for low- and high-speed devices, and dc power input lines.

The 990/4 Microcomputer board can function equally well in all the applications described for the TMS9900, provided miniaturization requirements can accommodate the full board size. In addition to satisfying those applications, the 990/4 Microcomputer can be mounted in a standard chassis, and amplified with additional memory, interface modules, and input/output peripherals. With these expanded features, the 990/4 becomes a flexible, powerful microcomputer system with the processing capability to tackle a wide range of minicomputer applications at a low price.

Texas Instruments complements this member of the family with an attractive selection of terminals, communication modules, floppy disc storage, and hard copy printers. These devices provide the computer system components that are well-suited for applications in manufacturing and process control, data communications, scientific data processing, business and finance.

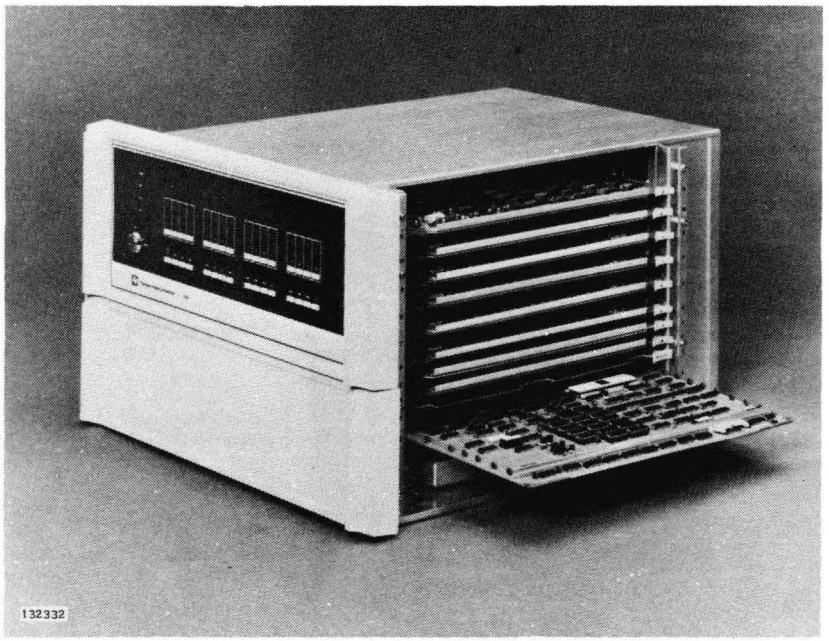
The next step up in micro/minicomputer processing performance is to implement the same instruction set and major architectural features in faster semiconductor technology. To this design add additional memory and high speed peripherals and it becomes the 990/10 Minicomputer, the third member of the 990 Family.

1.2.3 THE 990/10 MINICOMPUTER. The most powerful member of the family is the Model 990/10 general purpose minicomputer. The 990/10, a TTL implementation of the 990/4 architecture, provides the high-performance speeds demanded in many applications.

A memory mapping feature providing memory protection and privileged instructions supports memory expansion to one million 16-bit words. The TILINE*, an asynchronous high-speed I/O bus supports both high-speed and low-speed devices. With these devices comes a large online disc data base storage capability of millions of characters. Fast random access to a large data base extends the applicability of the 990/10 into all areas of minicomputer use.

The 990 Computer Family truly encompasses the full range of power and flexibility required for a multitude of applications from miniaturized controllers to large data base management and information systems.

*Trademark of Texas Instruments Incorporated



1.2.4 THE 990 FAMILY SOFTWARE. Software development is an essential element of implementing the 990 Computer into an application. With the pace setting lower prices of the 990 Micro/Minicomputers, this software development cost can easily reach an order of magnitude greater than the hardware cost. The quantity of systems to be implemented governs this cost since application software is primarily a one-time cost. The 990 Computer Family standard system software reduces overall software development. The flexible operating systems minimize the programmer/machine control interfacing and the high-level languages allow application software to be written in a format that is easy to understand and maintain.

The standard 990 software includes both memory-resident and disc-based operating systems, well-suited for real-time, multi-tasking environments. The programming languages encompass FORTRAN IV, COBOL, and Multi-user BASIC. Software development utilities are available to facilitate application program editing and testing.

Reservation Systems

1.3 990 SYSTEM APPLICATIONS

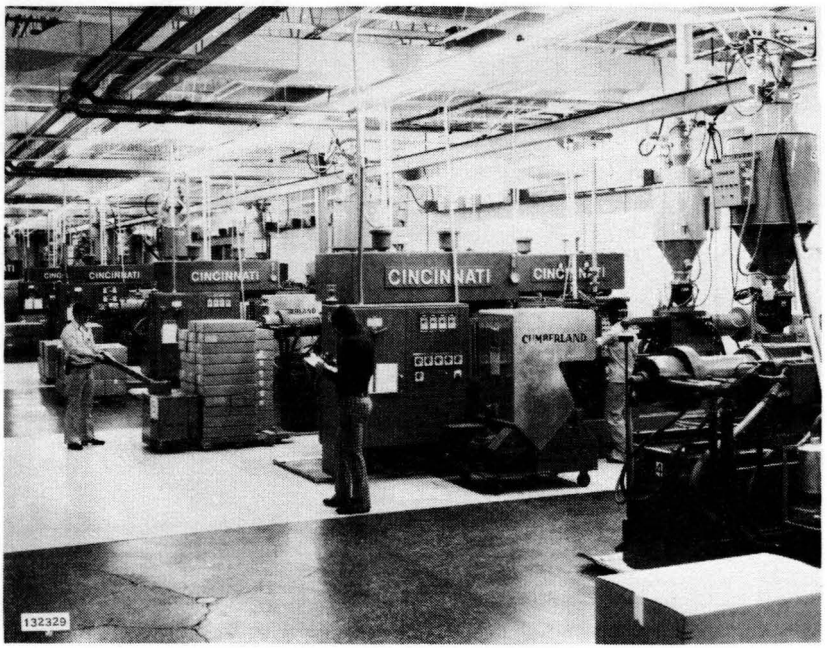
The first production of the Texas Instruments Model 990 Computer began in 1974 and over 1000 units have been shipped and over one million operating hours accumulated before public announcement. These early production units were comparable with the 990/10 member of the family. This extensive pre-announcement experience has proven the power and capability of the 990 Computer Family. These first 990 units have been utilized in nationwide reservation systems, as cluster terminal systems for a variety of applications, and as prototype units in real-time multi-processor environments.

In addition to the pre-announcement 990 experience, the accumulated years of experience in production and application of other 900 series minicomputers are reflected in the 990 architecture. The programmable input/output structure of the 990 family, the Communications Register Unit (CRU), has proven very effective in Texas Instruments Model 960 Computer applications in manufacturing and process control. Model 960 Computer users who desire to do so will find the transition to the 990 Computer Family comparatively easy. Texas Instruments has an experienced team of senior engineers and system analysts ready to assist the 990 user in system design and implementation.

Clustered Terminal Systems

1.3.1 SYSTEM DESIGN. There is a large body of literature on the subject of computer system design and obviously an in-depth analysis of this subject is beyond the scope of the handbook. The object of this handbook is not to train expert computer designers, but instead to present the 990 series in sufficient depth to permit analysis of the technical feasibility of a contemplated project and to outline a process for determining technical and economic feasibility.

1.3.2 SYSTEM COST ANALYSIS. Computer price information is volatile and cost analysis should be based on current information. There is a well-established trend toward lower prices in the CPU and memory and to a lesser extent in peripherals which tends to protect designers using current prices in these areas. This is counter to the general economic trend. In particular, increasing labor rates and high labor content tend to increase costs in maintenance, training, and software.

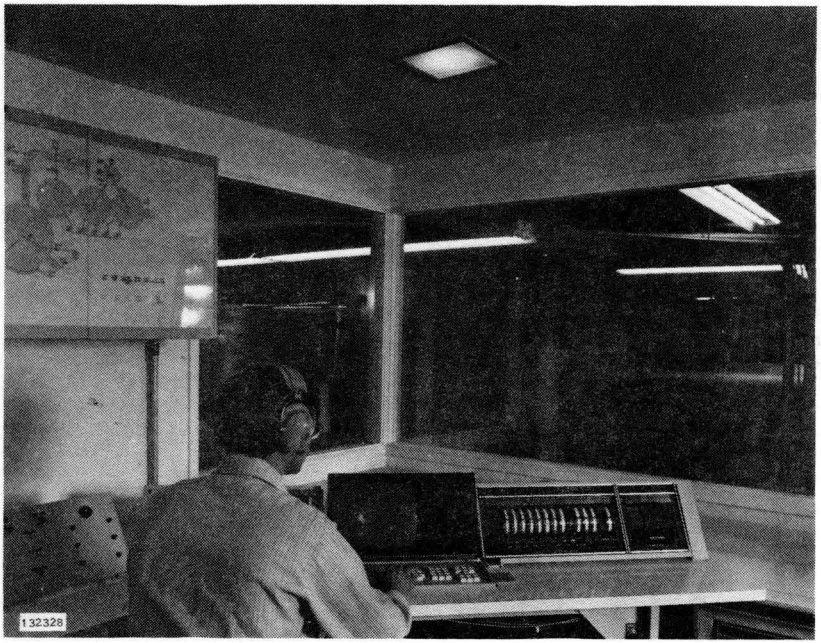


Manufacturing Automation

1.3.3 DESIGN PROCESS. Computer system design has the reputation of being complicated, time consuming, risky, and expensive. This reputation may be traced to projects which were undertaken with inadequate planning and usually an inadequate understanding of the design process and an incomplete analysis of the factors involved. There is a well-defined design process for 990 computer systems. Section VI describes the process and all of the factors that must be considered in implementing a 990 Computer System. With prudent, thorough planning, a project based on 990 computers has small risk to budget and schedule.

1.4 USING THE SYSTEM HANDBOOK

The system handbook is structured in seven sections; this first section serves as an introduction to the family. The remaining sections discuss 990 architecture, aspects of programming, system software, peripherals, system design guidelines, and Texas Instruments service and support. A wide variety of readers will find this handbook informative, useful, and a handy reference. The following reader guidelines may be used for the first cursory reading. Different users will predictably discover additional advantages and uses to those defined as follows.



Process Control

Management: This section, Sections IV, V, the first part of Section VI, and Section VII will provide you with a solid overview of the 990 computer family products, potential applications, service and support.

System Designers: Section II will provide a good overview of 990 features while Section VI will provide you a useful guide for system design and configuration.

System Analysts: Section IV will provide a working knowledge of the standard system software available to ease program development while Sections II and III will provide an insight into the flexibility and power of the 990.

Software Programmers: This section and Section II will provide an introduction to the 990, and Section III will serve as a detail definition of 990 machine and assembly language level programming features.

SECTION II

990 COMPUTER FAMILY HARDWARE

2.1 990 COMPUTER FAMILY CHARACTERISTICS

This section of the System Handbook will introduce you to the major features of the 990 computer architecture, and the unique features of the 990/4 Microcomputer and 990/10 Minicomputer. Additional specifications are included in appendices for the serious designer who requires added detail on execution speeds, interface signals, and other design data.

The following is a summary of characteristics for 990 Computers:

- 16-bit machines with bit, byte, and word capabilities
- Basic 16-bit instruction
  1. 69 instructions, 5 addressing modes
  2. Instruction adds 2nd and 3rd word as required for immediate operands and extended address
  3. Two address (memory to memory), single address and no-address instruction formats.
- Workspace organization
  1. Register file architecture with 16 registers
  2. Registers in memory, located by workspace pointer
  3. Both "Registers" and "Memory" are semiconductor devices
  4. Any number of workspaces, dedicated or shared.
- Status Register Organization

  Condition code, fault flags, mode control and interrupt mask are treated as one 16-bit word.
- Interrupts
  1. Interrupt driven design provides fast, automatic context switching
  2. Multi-level priority

3. Each level traps to fixed, separate location

4. Automatic masking to executing level (software can override).

- Extended Operations (XOPs)

1. 16 extensions, undefined at manufacture

2. Executes in hardware XOP processor if present, otherwise interpreted in software. No change in program for hardware or software execution.

3. XOP traps are separate and independent of interrupts

- Fast, Semiconductor Memory

1. PROM on CPU reached via CPU address space

2. 8K × 16 EPROM memory on one board

3. 20K × 16 RAM memory on one board

4. 32K × 16 RAM error correcting memory on two boards (990/10 only)

5. Parity, write protect, and error correcting options.

- Communication Register (CRU) I/O

1. Up to 4K input and 4K output lines

2. Instruction-driven.

- TILINE (not on 990/4)

1. Fast, multi-user, asynchronous bus

2. Approximately 50×10^6 bits/second bandwidth.

- Compatible Multi-Level Packaging

1. Chip, board, box and system level purchase option

2. Fully supported at every level.

- Three Chassis Options

  1. OEM (built-in) chassis

  2. 7-inch chassis

  3. 12-inch chassis (12¼-inch).

2.1.1 DATA FORMATS. 990 computers are 16-bit machines with memories that transfer 16-bit words in parallel. Machine instructions generally occupy one 16-bit word with certain instructions extending to two or three words. The instruction set provides for operations on words (16 bits), bytes (8 bits) and single bits. Byte and bit operations are performed internally in the CPU and all memory and memory bus operations transfer a 16-bit data word in parallel. 990 CPUs are very efficient at byte operations because:

- The CPU uses the LSB of addresses to reach bytes. A 16-bit address word thus describes 64K X 8 internal address space and 32K X 16 external (memory) address space.

- The CPU has an extensive repertoire of byte handling instructions.

2.1.1.1 Memory to Memory Instructions. Most 990 operations (Add etc.) use a two-address format, specifying both operand source and destination, which is notably efficient in conserving instruction space and consequently in reducing program storage requirements. For example, one two-address instruction, MOVE, replaces the two single-address instructions, LOAD and STORE, in the first place; one opcode, MOVE, replaces two opcodes, LOAD and STORE, saving space by using fewer instructions which usually speeds the process up as well as saving memory. The 990 is highly efficient in address fields too. This is accomplished by using minimum instruction space to define high-usage, near-reach operand storage. The workspace concept and the capability to dedicate workspaces makes for high usage of these efficient instructions. Options in operand development add a second or third word to the instruction to permit immediate operands (data imbedded in program) or full address reach. See figure 2-1.

2.1.1.2 Instruction Formats. Single-address program control instructions (BRANCH, etc.) are more efficient than two-address and therefore, most 990 program control instructions are single-address. See figure 2-2.

High program storage efficiency in a mixed program environment demands a number of alternatives in operand development, so in addition to the basic two-address, single-address, and no-address (IDLE, etc.) instruction formats, the 990 provides a number of alternatives with different "REACH" and development. There are a total of nine instruction formats, not counting the whole-word immediate operands and addresses. 990 code is terse; you can say what you want to say in a few words.

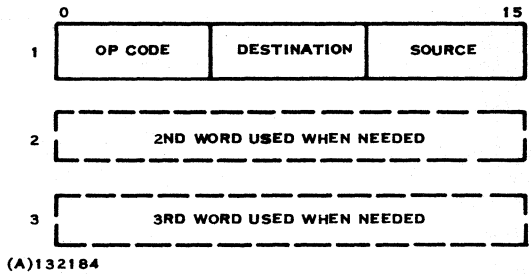


Figure 2-1. Two Address Instruction Format

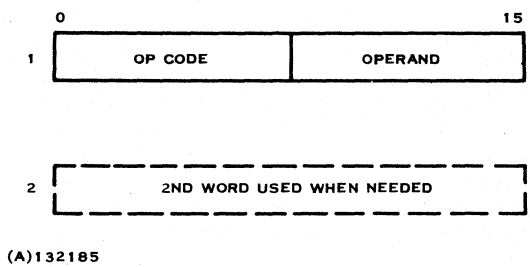


Figure 2-2. Single Address Instruction Format

There is a narrative description of the instruction formats in paragraph 3.1.3 and a detailed description of each instruction in paragraph 3.2 of this handbook.

2.1.2 STATUS REGISTER. Conditional branches reference a conditions code register, which holds relevant information on preceding operations. Program control and status operations treat the conditions code, fault flags, mode control bits, and the interrupt mask as a single 16-bit status register. Software can load and store the complete register and can also alter individual bits or fields with a single instruction.

2.1.3 REGISTER FILE. 990 computers are basically register file machines with 16 general purpose registers in the file. The large flexible file always offers the programmer another register which he can use for any purpose he chooses; consequently, the coding efficiency is high and program storage requirements are low. Certain registers are usually reserved for particular functions due to conventions in certain instructions. These conventions conserve space in the instruction word, thereby further reducing program storage requirements. These conventions use R11 as a link register (for Branch or Link) or the derived operand address (in extended operations-XOPs), R12 as a base register for CRU I/O addresses and R13, 14, and 15 as link registers for status exchange operations. There is a zero default in index register identification, which inhibits R0 from use as an index register, thus generating the immediate address mode.

2.1.4 WORKSPACE CONCEPT. The 990 register file is located in memory. This imposes very little speed penalty because the 990 computers use fast semiconductor memory. The classifications "hardware register" and "memory" are arbitrary where both are semiconductor devices. A "workspace pointer" register points to the file location in memory, the "workspace." The workspace may be located anywhere in memory on word (even numbered byte) boundaries. Workspaces may be dedicated to or shared by any task. All that is required to define or locate a workspace is to set the workspace pointer to the file start location. Any number of workspaces may be defined and it would be possible to completely fill the computer memory with workspaces (but not practical - there must be some program to use all those workspaces). Normal programming procedure for 990 computers is to assign a dedicated workspace for each task or subroutine. Note that this does not require 16 words of memory if the task uses fewer than 16 registers. Figure 2-3 shows the arrangement of workspace registers in memory.

2.1.5 INTERRUPTS. 990 CPUs provide either eight (990/4 and 990/10 in 7-inch chassis) or sixteen (990/10 in 12-inch chassis) interrupt levels. Levels are numbered from 0 with 0 having highest priority. There is a 4-bit interrupt mask which stores the level number of any interrupt presently executing and prevents interrupts from the same or lower levels from interrupting. Software can also address the mask and alter the automatically-set level. There is an acknowledge signal for each interrupt to provide synchronization, so that each interrupt signal is honored once and only once. Each interrupt level is uniquely associated with a two-word location in memory, the trap address, which directs the CPU to the start of the interrupting program and replaces the workspace pointer to provide the workspace associated with the interrupting program.

2.1.5.1 Internal Interrupts. Certain interrupts are reserved for internal functions or may be set to internal functions at the user's option. Refer to paragraph 2.2 for details on 990/4 and paragraph 2.3 for details on 990/10.

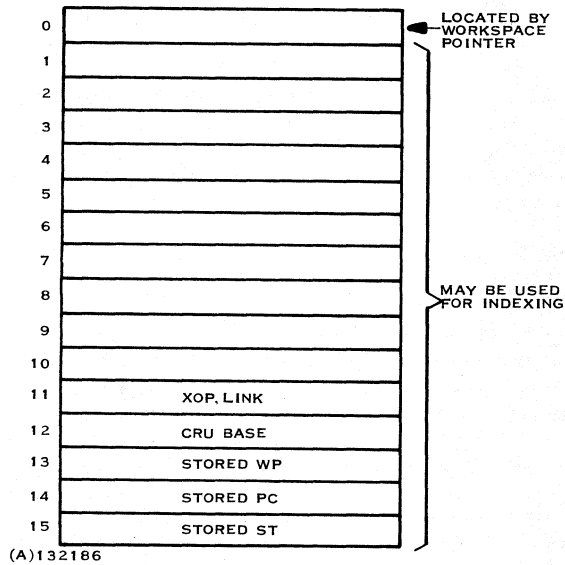


Figure 2-3. Workspace Registers

2.1.5.2 Interrupt Wiring. The 7-inch and 12-inch chassis for the 990 provide most backplane wiring in etch on the motherboard. The interrupts are an exception. Interrupts are made available on the backplane via pins near the CPU connector and are routed by wire-wrap or plug jumpers as desired, thereby providing for easy assignment and alteration.

2.1.5.3 Context Switching. The combination of fast memory, workspace concept and vectored interrupts results in a **fast automatic context switch.** Only 3 registers (program counter, workspace pointer, and status register) need to be stored and loaded to provide a complete level change including the 16 file registers, while preserving the capability to restore the interrupted level. This permits the 990 computers to handle extraordinary high interrupt rates, up to a theoretical limit of 100,000 interrupts per second in the 990/10 (60,000 in 990/4). The theoretical limit is the case where the processor is occupied full time in interrupt trap overhead. In practical terms, the 990/10 uses 10 percent of the processor time for overhead context switching to handle 10,000 interrupts per second (6,000 per second for 990/4).

This means that a 990 CPU can handle high speed I/O by economical CRU driven I/O, where most machines would require an expensive autonomous controller. The 990 has a tremendous potential for savings in cost at the systems level.

Designers should also consider the use of a multiprocessor organization using a 990 CPU instead of a special autonomous controller. This approach frequently results in both lower non-recurring and lower recurring cost. Recall also that the trend is toward lower cost and higher speed in successive generations of standard CPUs and toward higher cost in special controllers so that the most reasonable expectation is ever-increasing benefits through use of the 990 CPUs.

2.1.6 EXTENDED OPERATIONS (XOPs). Sooner or later, every computer is confronted with a special problem which demands a special instruction not implemented in the basic instruction set. Historically, some machines have provided for this situation by allowing microcode programming, a sort of second level program whereby the operations which make up the program level instruction are themselves programmable. The 990 does use microcode, but only as an implementation convenience, the code is not user alterable. When the instruction set of a machine is changed all previous software for that machine is disinherited; therefore, the 990 does not permit alteration of the microcode. The 990 provides for special instructions in the most direct, straightforward way—by providing for instructions which are not defined at implementation. The 990 instruction set provides 16 extended operation codes (XOPs), which may be defined by the user in hardware or software. When the CPU encounters one of these XOPs it tests (via a defined signal interface) to determine if a hardware XOP processor is present, and where one is present the CPU hands control to that XOP processor for execution. Where no XOP processor is present, the CPU traps through one of 16 locations in memory and hence to a software routine which executes the instruction. XOP traps work like interrupts, but are entirely independent and use separate trap locations.

The programmer need not concern himself with whether an XOP processor is present, the code is the same in either case. The same code can even be run on two different machines, one with an XOP processor and one interpreting XOPs in software, with identical results.

2.1.7 MEMORY DEVICES. 990 computers use semiconductor memories. The speed, economy, and reliability of semiconductor memories are superior to any available alternative and the rapid development in semiconductor memories promises to widen the gap. Four types of semiconductor memory devices are used: PROM, EPROM, static RAM, and dynamic RAM.

2.1.7.1 Programmable Read-Only Memory (PROM). The PROM devices used are fusible-link devices. The fuse link is blown, one bit at a time, to program the device. Once programmed, the content is permanent and may not be altered. Naturally, the content is not affected by power loss or power failure.

2.1.7.2 Erasable Programmable Read-Only Memory (EPROM). EPROM devices use a semiconductor with a hysteresis loop for a storage device. The EPROM is programmed one bit at a time in a special test set, similar to the PROM programmer. Once programmed, the content is non-volatile and is not affected by power loss or power failure. The program of the entire chip may be erased by ultraviolet radiation, in which event the EPROM may be reprogrammed from scratch. The EPROM must be removed from the computer for programming and program erasure.

2.1.7.3 Static Random Access Memory (RAM). Static RAM devices are read-write devices employing a bistable memory element. The memory content is volatile and is retained so long as power is present.

2.1.7.4 Dynamic Random Access Memory (RAM). Dynamic RAM devices are read–write devices employing a bistable memory device with a time-decay characteristic. The content is volatile and must be periodically refreshed in order to avoid data loss. The refresh cycle is built into 990 computers and invisible to the program. Refresh cycles steal approximately 2 percent of the memory cycles. A battery back-up for stand-by memory refresh is provided as an option.

Memory device size and shape factor (words X bits) varies according to the device used from 256 X 4 to 4096 X 1. Multiple devices are always operated in parallel to provide a 16-bit data word. (Byte and bit addressing are handled entirely in the CPU).

2.1.8 MEMORY ADDRESSING. 990 instructions build a 16-bit address word which may be viewed as describing a 64K X 8 CPU address space. The LSB is used inside the CPU to address bytes and the 15 MSBs are passed to the memory or mapping hardware. These 15 MSBs describe a 32K X 16 address space. Mapping hardware, when present, "maps" the 32K X 16 space into a much larger 1,024K X 16 address space.

All 990 memory devices; PROM, static RAM, or dynamic RAM are assigned some portion of the 64K X 8 (or 1024K X 16) address space. There is no "load cycle" where a ROM outside the address space is dumped into RAM inside the address space. The PROM devices are always available for instant response when addressed.

990 Computers reserve the lowest 32 words of memory for interrupt traps and the next 32 words for XOP traps. Refer to figure 2-4.

2.1.9 MEMORY PACKAGING. 990 CPUs allow some memory devices to be mounted on the CPU board (or boards). These devices are referred to in 990 literature as "on-board" memory. Other memory, mounted on boards dedicated to memory is referred to as "expansion memory."

2.1.10 COMMUNICATIONS REGISTER UNIT (CRU). The CRU is the 990 general-purpose, command-driven I/O. The CRU is the "bit picking" I/O originally designed for process control operations and introduced with the Model 960 Computers. The CRU has proven to be a very successful computer I/O system because of the flexibility which adapts to every command-driven application.

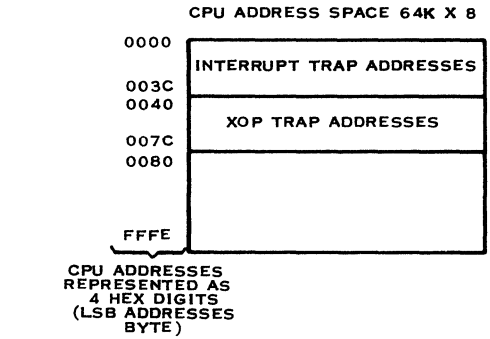


Figure 2-4. Trap Addresses

2.1.10.1 CRU Addressing. The CRU may be visualized as 4K consecutive bits, which are addressed as independent bits or in groups of up to 16 bits by CRU input/output instructions. The CRU is addressed only by these I/O instructions and the CRU bus is totally separate from the memory/TILINE bus. Furthermore, input and output bits may be separate and unrelated so that the CRU is best visualized as a 4096-bit input register and a 4096-bit output register.

Most CRU modules use 16 input and 16 output bits and most modules are half-sized cards. Thus, the 12¼-inch, 13 slot chassis will accommodate one 990/4 CPU and 24 half-size CRU modules (384 CRU bits). Normally, some of the slots in the CPU chassis will frequently be used for other purposes, thereby reducing the CRU space available. Systems, which require more CRU modules than the CPU chassis can accommodate, will use expansion chassis. The CRU is expanded via a CRU Expander Card in the CPU chassis and a CRU buffer card in the expansion chassis as shown in figure 2-5. One expander card can drive up to seven CRU expansion chassis, which is the maximum CRU expansion using standard expansion and buffer cards.

The expansion scheme based on the standard chassis does not use all 4096 bits. The decoding is simplified by using binary boundaries (in the interest of economy). Each half-sized card slot is assigned 16 consecutive CRU bits corresponding to the 4 LSBs of the CRU address. The next 5 bits of the CRU address are decoded to select the card slot (the module in that slot), but there are only 24 half-slots in the 12-inch chassis (10 in 7-inch chassis) and so only 24 of the 32 possible half card addresses are used. The remaining three bits of the CRU address, the three MSBs, are used to designate chassis number. See figure 2-6.

The upper 128 bits of the CRU are reserved for use by the CPU in controlling internal functions. These bits are not used in the standard CRU expansion scheme in any case.

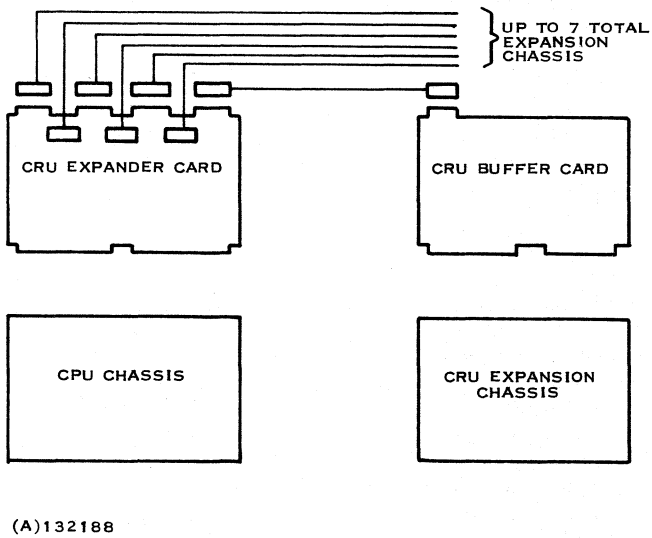


Figure 2-5. CRU Expansion Link

Each full-sized card slot in the standard chassis has two connectors, corresponding to the two half-card modules which may use the slot. Both connectors are furnished with the four LSBs of the CRU address and thus each may use 16 bits of the CRU. See figure 2-7. The Module Select signals are decoded from the eight MSBs of the CRU address and thus each Module Select signal addresses a 16-bit "register" in the CRU. P1 of the connector pairs in a slot receives one Module Select signal corresponding to one 16 bit register, but P2 receives two Module Selects and thus may use up to 32 bits of the CRU. P1 also receives the eight MSB of the CRU address and thus may be used for CRU expansion drivers or for modules that ignore Module Select and directly decode their own CRU address.

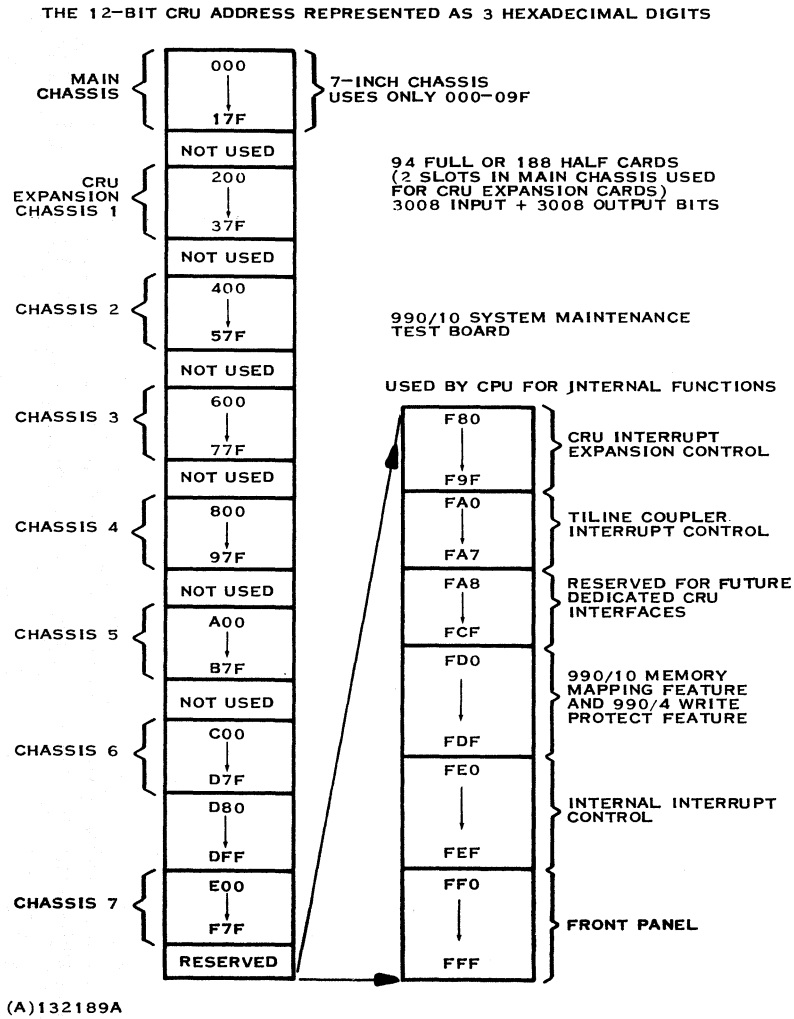


Figure 2-6. CRU Address Map for Standard Expansion

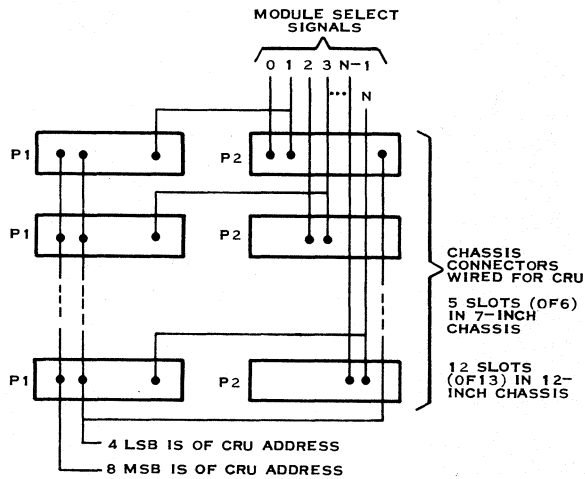


Figure 2-7. Standard Chassis CRU Wiring

2.1.10.2 CRU Expansion Chassis Interrupts. A hardware scanner for interrupts is located on the CRU buffer card. It scans up to 32 interrupt signals and transmits any interrupt plus a 5-bit interrupt ID signal to the CRU expansion card. The expander card attaches a 2-bit chassis ID and four power status bits. A 16-bit expansion card interrupt word is formed and is accessible to the CPU (via the CRU). Two such words are maintained, one for expansion chassis 1-4 the other for chassis 5-7. These words are formatted for establishing CRU interrupt trap zones commuted off two levels in the primary interrupts. The forward linkage is only two instructions longer than a direct interrupt (Read CRU and BLWP) and the return is only one instruction (Return) longer.

Any CRU module may be connected to any multiplexer interrupt by a wire wrap or plug jumper scheme. In addition to the scanner interrupt, two interrupts may be routed through the cards and cable to the CPU chassis for direct interrupts.

2.1.10.3 CRU Data. CRU data is distributed serially on one output line and one input line. The 12-bit CRU address selects a single bit and successive bits on the same CRU instruction are routed to the next higher CRU bits. Thus, a CRU output command, which addresses bit 010 and then transmits three bits, would send the first bit to 010, the second bit to 011 and the third bit to 012. A single CRU command is limited to 16 bits maximum and can cross module boundaries. Module boundaries are address 000_{16} , 010_{16} , 020_{16} and so forth.

The serial shift rate of the CRU data is 1.5 MHz for the 990/4. The 990/10 operates at a CRU data rate of 4 MHz, except for CRU input operations from external chassis for which the rate is 2 MHz.

2.1.11 TILINE. TILINE is the 990/10 high-speed, multi-user, asynchronous parallel bus. The TILINE is basically capable of over 3 million 16-bit transfers per second, a bandwidth of approximately 50 million bits per second. Multiple users, including 990/10 CPUs, contend for access to the TILINE in a positional priority scheme. Examples of other bus users are: (1) Autonomous controllers such as disc controllers and (2) other CPUs when the TILINE is used to construct multiprocessors. When access is granted to a user, then that user places an address on the bus. The address selects the respondent and in the case of a memory module also selects the desired word in the module. In 990 literature, users are called TILINE masters and respondents are called TILINE slaves. Certain devices, such as disc controllers, are both master and slave. The disc controller is a slave when receiving commands from the CPU and a master when contending for the bus to transfer data to the memory.

The TILINE data bus is bilateral and transfers 16-bit data in parallel from master to slave as during a memory write cycle, or from slave to memory as during a memory read cycle.

There are two factors in the TILINE bus scheme which profoundly affect the flexibility of the bus.

First, the TILINE is asynchronous. Slaves are required to signal when responding and masters are required to wait for the response signal. This permits fast devices and slow devices to be mixed on a single bus. This also permits TILINE buses to be transmitted over long distances. Naturally, the transfer rate slows down as the transfer distance increases because of increased signal propagation time.

Second, the bus user priority logic is distributed rather than centralized on some particular module such as the CPU. Because of distributed logic, there is no vital module, such as the CPU, which must be located in every chassis. Thus, a chassis may be used for memory expansion only, and filled with memory modules. Conversely, two CPU modules can be located on one TILINE (the higher priority CPU will steal most of the access when executing out of TILINE memory).

The TILINE itself consists of back plane bus wiring in a 990 chassis. Any full size card plugged into the chassis may engage the bus and participate in the TILINE transfers. The bus is divided between the two connectors in a full card slot.

2.1.11.1 TILINE Address Space. The TILINE uses a 20-bit address to define an address space of 1024K × 16. The upper 1K addresses are reserved for special functions. Of the upper 1K, the first 512 words constitute the TILINE Peripheral Control Space (TPCS), which is reserved for peripheral device controllers such as disc controllers.

2.1.11.2 TILINE Links. The modules interfaced to a TILINE may be physically located in two or more chassis. In this case, the TILINES (recall that a TILINE is back panel wiring) of the two or more chassis must be linked. The characteristics of this link are:

- The link consists of two TILINE couplers and a cable between the couplers. The link is always one chassis to one chassis with a coupler in each chassis. The coupler is a full-size card.

- One chassis may be directly linked to several other chassis by using multiple links, each connecting the first chassis to one other chassis. One chassis may also be indirectly linked to another chassis via a third chassis or a chain of chassis.

- The link is bilateral so that master units in either chassis may address slave units in either chassis. TILINE couplers have full-bilateral capability, but there are also provisions on the coupler boards to restrict this capability.

- The link is invisible when not in use so that there is a capability for a master unit in one chassis to address a slave unit in the same chassis while simultaneously at the other end of the link another master is addressing a slave unit.

- The TILINE coupler may be used to construct fixed-address systems in which every unit in the system has a unique address, which may be directly invoked from anywhere in the system. Fixed address systems are limited to 1024K total addresses and are usually restricted to one CPU in the system (because of interrupt trap address restrictions).

- The TILINE coupler may also be used to construct relative-address systems, in which the total addresses may greatly exceed 1024K and multiple CPUs may be used.

2.1.11.3 TILINE Coupler. Each TILINE coupler consists of a transmitter and a receiver. Figure 2-8 shows the convention used in naming these units. Simultaneous bilateral transmission through a cable is not possible. The "deadly embrace" in case of simultaneous requests is avoided by priority logic and a "preferred direction."

The transmitter on a TILINE coupler ignores all addresses when the receiver located on the same coupler card is active. When the receiver is passive, then the transmitter accepts consecutive addresses between a lower boundary and an upper boundary and also any address falling within the TILINE Peripheral Control Space (TPCS). The transmitter adds a bias value to any address between the lower and upper boundary and transmits the biased address or the unbiased TPCS address via the cable to the receiver on the far end of the link. The receiver, a master unit, contends for the TILINE access like any other master unit. The lower and upper boundary and the bias value for the transmitter are set by switches on the coupler card to any value from 0 to 1020K on 4K boundaries.

The boundaries and bias value may be converted to programmable operation by removing the switches and connecting 16 I/O data modules to the switch sockets.

It will normally be necessary to implement "attention" interrupts in multiprocessor systems. The TILINE coupler card provides a CRU bit on the transmitting end and routes this bit through the cable for interconnection to an interrupt on the far end. Any control words which must be passed from processor to processor are handled as normal TILINE data transfers.

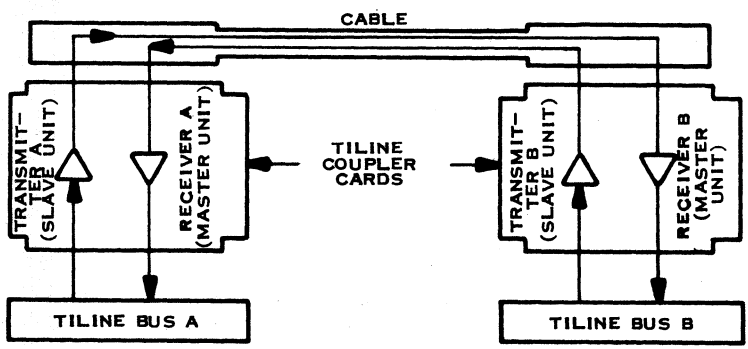
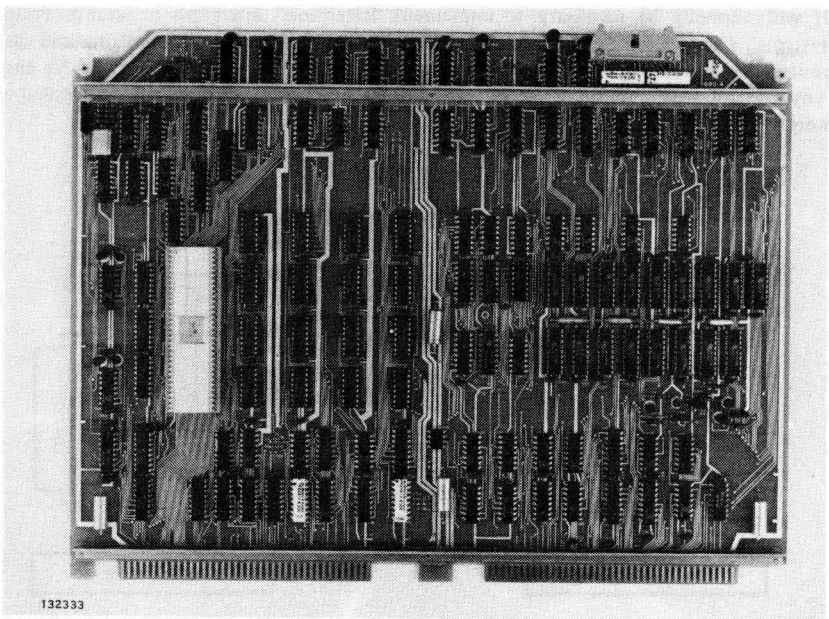


Figure 2-8. The TILINE Link

2.1.12 SELF TEST AND DIAGNOSIS. There is a general self-contained capability in 990 computers to detect and indicate faults in the computer, and to isolate such faults to a single replaceable module. The techniques used are different for the two computers covered by this handbook, the 990/4 and 990/10, and are covered in paragraphs 2.2 and 2.3, respectively. The capability is dependent on certain options, such as memory parity. TI strongly recommends that this capability should be implemented because:

- Repair time is reduced
- Maintenance cost over the system life is lower
- User satisfaction is increased.



132333

2.2 990/4 COMPUTER

The 990 Computer Family characteristics described in the preceding paragraph (2.1) fully apply to the 990/4 with the exception of the TILINE interface. There is no TILINE on the 990/4. 990/4 memory expansion is handled by a synchronous DMA bus.

This paragraph describes the special characteristics of the 990/4; it does not apply to other 990 Computers.

The 990/4 is a computer on one board. The board contains the CPU and up to 5K of on-board memory. The memory may be expanded via RAM and EPROM memory expander boards and the 990/4 interfaces the full range of CRU modules including CRU expansion options.

2.2.1 990/4 CHARACTERISTICS. 990/4 characteristics are as follows:

990/4 CPU

- Implementation:
  TMS9900 NMOS LSI Processor
  SN74S287 TTL 256 X 4 PROMs
  TMS 4043 NMOS 256 X 4 Static RAMs
  TMS 4051 NMOS 4096 X 1 Dynamic RAM
- Board size approximately 10.80 by 14.25 inches
- 1K words PROM or status RAM in 256 word increments
- 4K words dynamic RAM
- 8 interrupts, up to 7 external
- Real time clock input
- CRU interface for I/O
- DMA interface for extended memory can be used for autonomous I/O
- Console option via special connector
- Self-test option
- Parity option
- Fault indicator.

990/4 RAM Expansion Board

- Implemented in 4096 X 1 dynamic RAM
- 4K to 20K words in 4K increments
- Parity option
- Write protect option.

990 Series EPROM Expansion Board

- Implemented in 1024 X 8 EPROM
- 1K to 8K words in 1K increments
- Non-volatile, erasable, programmable.

2.2.2 990/4 MEMORY. 990/4 memory is subdivided into on-board memory, RAM expansion memory, and EPROM expansion memory.

2.2.2.1 On-Board Memory. The 990/4 CPU provides for three types of memory to be mounted on the CPU board: PROM, static RAM, and dynamic RAM.

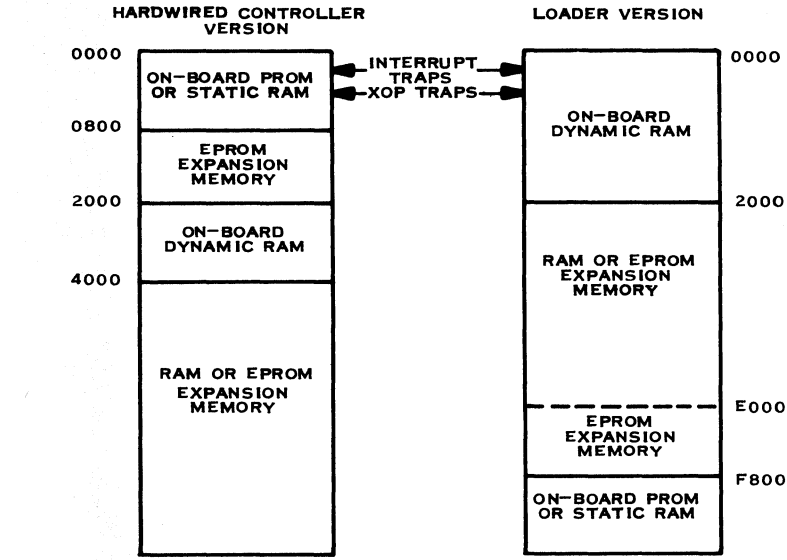
PROM Memory. The PROM device used is the SN74S287 256 X 4 TTL PROM. Four devices are used in parallel to provide the 16-bit data word. The cycle time is 667 ns. The SN74S287 devices are mounted on IC sockets for easy replacement. Refer to figure 2-9.

Each set of four devices provides one increment of 256 X 16 words. Up to four sets may be used to provide up to 1K (1024) X 16 PROM memory.

There is a jumper wire option on the card which sets the start address used by the PROM memory devices to 0000_{16} or to $F800_{16}$. Generally speaking, where the 990/4 is used as a hardwired controller with program in PROM and compatibility with standard 990 software is no object, then start address 0000_{16} should be used. Where the PROM is used to bootstrap load from peripheral devices or communication lines or where standard software compatibility is desired, then start address $F800_{16}$ should be used.

Static RAM Memory. The static RAM memory device is the TMS 4043 256 X 4 NMOS static RAM. Four devices are used in parallel to provide the 16-bit data word. The cycle time is 1.33 μs.

Each set of four devices provides one increment of 256 X 16 words. Each set of static RAM devices plugs into the same sockets used for PROM devices. The four set capacity of the board may be used for all PROM, all static RAM, or any mixture on 256 word boundaries. Do not mix PROM and static RAM inside a set.



NOTE: FOR COMPATIBILITY WITH STANDARD 990 SOFTWARE AND OTHER 990 COMPUTERS, THE LOADER VERSION MUST BE USED WITH PROM CONFINED TO FC00 UP (512 WORDS) AND ADDRESSES F800-FC00 (TILINE PERIPHERAL CONTROL SPACE) NOT USED IN 990/4. ADDRESSES 0000-004F MUST BE RESERVED FOR INTERRUPT TRAPS, XOP TRAPS, AND STANDARD WORKSPACE.

(A)132192

Figure 2-9. 990/4 Address Options

The jumper wire start address option described for PROM devices operates on both PROM and static RAM devices simultaneously.

Dynamic RAM Memory. The storage device used is the TMS 4051 4K × 1 dynamic NMOS random access memory (RAM). Sixteen devices are operated in parallel (17 with parity) to provide the 16-bit data word. Cycle time is 667 ns. The TMS 4051 devices are mounted on IC sockets for easy repair.

The dynamic RAM memory start address is set by a jumper wire option to 0000_{16} or 2000_{16}. This option overlaps the PROM/static RAM start address and both memories must not simultaneously be set to 0000_{16}.

On-Board Parity Option. Parity is optional on the dynamic RAM, but is strongly recommended by TI. When equipped with parity, the 990/4 CPU board indicates by the LED on the board when a memory error has occurred. The stimulus for an interrupt is also generated and may be wired to an external interrupt in order to cause a trap when parity errors occur.

2.2.2.2 990/4 RAM Memory Expansion. 990/4 RAM memory expansion module consists of a single full size plug in card which interfaces to the 990/4 data bus. Module capacity is from 4K to 20K words in 4K increments. Module capacity is set at manufacture and is not alterable in the field. Only populated IC sockets are furnished on the card. The storage device used is the TMS 4050 4K X 1 dynamic RAM. Cycle time is 667 ns.

The module interfaces to the 990/4 data bus and occupies a block of consecutive addresses. The start address may be set to address zero or any 4K word boundary in the TILINE address space. The start address is set by switches on the memory card. The upper limit is set by the module size, thus a 4K module responds to 4K addresses (only), an 8K module responds to 8K addresses and so forth.

Parity Option. Parity is optional but is strongly recommended by TI. Memory modules, equipped with parity, indicate by an LED on the board when a parity error has occurred. The stimulus for an interrupt is generated and sent to the memory error signal on the CPU.

Memory Protect Option. The memory protect option adds the capability to define a write protected zone in memory. The boundaries of the protected zone are programmable in 256 word increments. The protected zone is defined by two 7-bit registers, the upper bound and lower bound, which are addressed via the CRU. The protect function may be disabled under program control without altering the boundary registers.

2.2.2.3 990 Series EPROM Memory Expansion. 990 Series EPROM memory expansion module consists of a single full-size card which interfaces to the 990/4 data bus or to the 990/10 TILINE. Module capacity is from 1K to 16K words in 1K increments.

The storage device used is a 1024 X 8 EPROM mounted on IC sockets on the board. The board is fully populated with sockets. Module capacity is easily altered by adding (or removing) EPROM devices. Program content is easily altered by replacing devices or by erasing and reprogramming devices. The EPROM devices must be removed from the board for erasure and reprogramming.

The module occupies a block of consecutive addresses. The start address may be set to zero or any 1K boundary in the data bus or TILINE address space. The upper limit is also set by switches to correspond to the module size, thus a 3K module responds to 3K addresses.

2.2.2.4 Interrupts. The 990/4 CPU provides eight interrupt levels. Levels are numbered 0 through 7 with level 0 having highest priority. The 4-bit interrupt mask stores the level number of any interrupt presently executing and prevents interrupts from the same or lower levels from interrupting. There is an acknowledge signal for each interrupt to provide synchronization, so that each interrupt signal is honored once and only once. Each interrupt level is uniquely associated with a two-word location in

memory, the trap address, which directs the CPU to the start of the interrupting program and replaces the workspace pointer to provide the workspace associated with the interrupting program.

Level 0 is always reserved for internal use as the power-up trap. Levels 1 through 7 are available for external use. Three interrupt conditions are generated on the board and may be connected to certain levels, if desired. The three conditions are: power failure, memory error, and real time clock.

Where compatibility with standard software is desired, power failure will be connected to level 1, memory error will be connected to level 2, and real-time clock should be connected to level 5.

2.2.2.5 990/4 Special Options. Special options for the 990/4 are CPU self-test option and Programmer's Panel option. These options consist of TI-programmed PROM sets which plug in to the on-board PROM memory sockets. Where compatibility with standard software is desired, these options are installed at address FC00 and FE00; reserving addresses F800 to FC00 for user functions.

CPU Self-Test Option. TI strongly recommends the self-test PROM set (256 X 16). This option PROM set contains a diagnostic program which tests the 990/4 CPU. When this set is installed, the programmer panel "LOAD" button calls the diagnostic program before the loader and a CPU test is performed. A failure on the test is indicated by an LED mounted on the board.

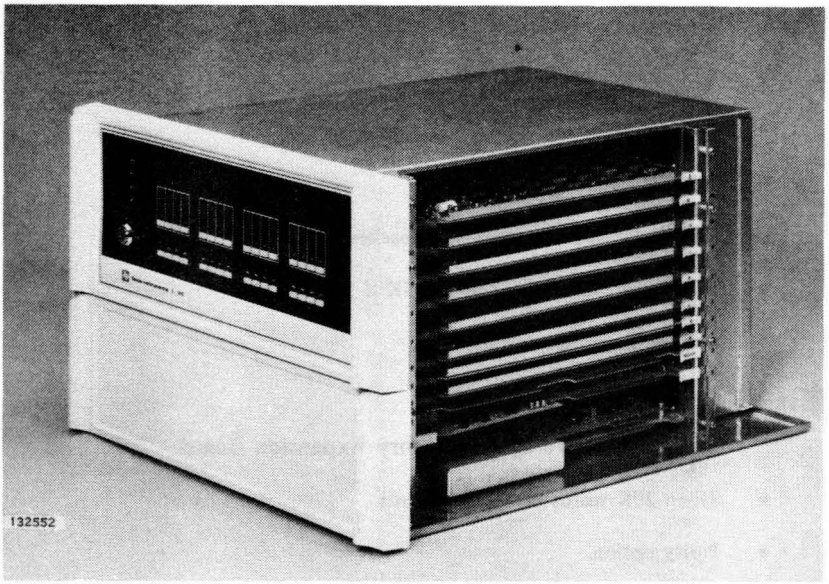
Programmer Panel Option. The programmer panel is driven via the CRU in 990/4 computers. Where the programmer's panel is used, either this optional TI-programmed PROM set must be included or else some other provision must be made for the software which interfaces this panel.

2.2.2.6 990/4 Power Requirements. The RAM memory devices used in the 990/4 are volatile, and where it is desired to retain the content of these devices during primary power interruptions it is necessary to provide some standby power source, such as a battery. The voltages identified as "MEM in table 2-1 are vital to RAM data content. In cases where data loss during power interruptions is acceptable, the "MAIN" and "MEM" voltages may be common.

Table 2-1. 990/4 Power Requirements

	+5V		+12V			−5V
	MAIN	MEM	MAIN	MEM		MEM
				OPER	STBY	
990/4 AU	2.5 A	–	40ma	–	–	–
256 X 16 PROM	0.45A	–	–	–	–	–
256 X 16 STATIC RAM	0.25A	–	–	–	–	–
4K X 17 RAM	–	0.32A		0.60A	0.10A	1.2ma
EXT MEM 4K	3.2 A	0.50A	–	0.29A	0.04A	3.0ma
8K	3.2 A	0.58A	–	0.33A	0.08A	3.0ma
12K	3.2 A	0.66A	–	0.37A	0.12A	3.0ma
16K	3.2 A	0.74A	–	0.41A	0.16A	3.0ma
20K	3.2 A	0.82A	–	0.45A	0.20A	3.0ma

1. All voltages ±3% except −5V is ±6%
2. Voltages labeled "MEM" are vital to dynamic RAM memory content



2.3 990/10 COMPUTER

The 990 series characteristics described in paragraph 2.1 fully apply to the 990/10. This paragraph describes the special characteristics of the 990/10; it **does not** apply to other 990 computers.

The 990/10 is a CPU on two full boards. The boards may be functionally identified as an arithmetic unit and a memory controller. Two versions of the 990/10 are offered, one without mapping and capable of addressing 32K total memory and the other version with mapping and capable of addressing 1024K of memory. The two versions differ in the memory controller board. The mapping feature cannot be added to a non-mapping controller.

2.3.1 990/10 CHARACTERISTICS. 990/10 characteristics are as follows:

990/10 CPU

- **Implementation: TTL MSI on two printed circuit boards**
- Board size is 10.8 by 14.25 inches
- 256 or 512 word on board PROM for Programmer Panel, loader and self-test options.

- 16 interrupts (in 12¼-inch chassis), 13 external
- Real time clock
- CRU interface for I/O
- TILINE fast multi-user bus
- Fixed address (no map/version reaches 32K X 16)
- Map option version reaches 1024K X 16
- Built-in diagnostic option
- On-board fault indicator.

990/10 RAM Memory Expansion Board

- 8K to 20K words in 4K increments
- Parity option.

990 Series EPROM Expansion Board

- Implementation in 1024 X 8 EPROM
- 1K to 16K words in 1K increments
- Non-volatile, erasable, programmable.

990/10 Error Correcting Memory Modules

- 8K plus controller on one PCB
- 8K, 16K, or 24K on second PCB
- Single bit error correction, 2-bit detection on each memory word
- Fault indicators on controller.

2.3.2 990/10 MEMORY MODULES. There are three basic alternatives in 990/10 memory modules:

- RAM module, 4K word increments from 8K to 20K on one card with parity option.

- EPROM module. 1K word increments from 1K to 8K words on one card.

- Error correcting memory module, 4K or 8K plus EC logic on first card, 8K increments to 24K on second card, 32K words total capacity on two cards.

2.3.2.1 990/10 RAM Memory Modules. The 990/10 basic memory modules consist of a single, full-size plug-in card which interfaces to the TILINE. Module capacity is from 8K to 20K words in 4K increments. Module capacity is set at manufacture and is not alterable in the field.

The storage device used is the TMS 4050 4K × 1 dynamic MOS random-access memory (RAM). Sixteen devices are operated in parallel (17 with parity) to provide the 16-bit data word. The access time is 500 ns typical and the cycle time is 725 ns. TMS 4050 devices are mounted on IC sockets for easy repair. Only populated IC sockets are furnished on the card.

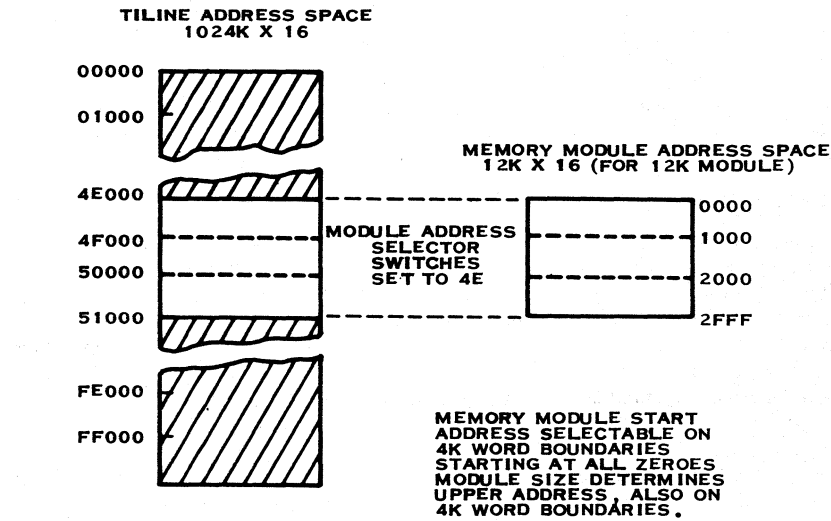
Memory modules interface to the TILINE and occupy a block of consecutive TILINE addresses. The start address of a module may by set to address zero or any 4K word boundary in the TILINE address space. The start address is set by switches on the memory card. The upper limit is set by the module size, thus a 4K module responds to 4K addresses (only), an 8K module responds to 8K addresses and so forth. Refer to figure 2-10 for an example of memory module addressing.

Parity is optional, but is strongly recommended by TI. Memory modules, equipped with parity, indicate by an LED on the board when a parity error has occurred.

2.3.2.2 990 Series EPROM Memory Expansion. 990 Series EPROM memory expansion module consists of a single, full-size card which interfaces to the 990/4 data bus or to the 990/10 TILINE. Module capacity is from 1K to 8K words in 1K increments.

The storage device used is a 1024 × 8 EPROM mounted on IC sockets on the board. The board is fully populated with sockets. Module capacity is easily altered by adding (or removing) EPROM devices. Program content is easily altered by replacing devices or by erasing and reprogramming devices. The EPROM devices must be removed from the board for erasure and reprogramming. The access time is 500 ns and the cycle time is 775 ns.

The module occupies a block of consecutive addresses. The start address may be set to zero or any 1K boundary in the data bus or TILINE address space. The upper limit is also set by switches to correspond to the module size, thus a 3K module responds to 3K addresses.



(A)132193

Figure 2-10. Memory Module Addressing Example

2.3.2.3 990/10 Error-Correcting Memory Modules. 990/10 error correcting memory modules consist of one full-size, plug-in card for a 4K or 8K module and two full-size cards for larger modules up to 32K. Multiple modules are used to construct memories larger than 32K. Single card modules and the first card of a two-card module are identical and contain the TILINE interface, memory control circuits, error correcting circuits, 4K or 8K memory, and LED indicators. The second card of two card modules contains 8K, 16K, or 24K of memory, and interfaces to the first card (only) via top edge connectors and cable. Memory content, 4K or 8K for the controller card and 8K, 16K, or 24K for the second card is set at manufacturing and is not field alterable, but the second card is easily added to an existing one card module. TMS 4060 RAM devices are used for memory storage elements, mounted on IC sockets for easy repair. (Only populated IC sockets are furnished on the card.) Access time is 600 ns typical and cycle time is 825 ns.

The error-correcting feature is implemented by attaching a 6-bit Hamming code to every 16-bit data word. The characteristics of this code permit identification and correction of all single-bit errors and the detection of all 2-bit errors. System operations are unaffected by a single bit error and proceed as though no error had occurred. The basic reliability of the TMS 4060 is extremely high so that the probability of two failures in a single 22-bit word is insignificant and the normal

operating mode is to ignore failures until periodic maintenance, at which time failed devices are noted and replaced. LEDs on the controller card signal when an error has occurred and identify whether the error was correctable or not.

An error correcting module occupies a block of consecutive addresses and the start address is set to any 4K boundary by switches on the card.

TI strongly recommends the use of error correcting memory modules where total memory size exceeds 20K words. The TMS 4060 RAM is the most reliable memory ever used in a computer, but there is a failure rate and when these devices are used in large quantities, then the number of failures becomes appreciable. Minicomputers were historically used with small memory systems and demonstrated high reliability. Reliability expectations are still high for "minicomputers" and the 990 can deliver high reliability together with performance equal to much higher priced machines. The error correcting memory module is a vital part of the high reliability where large memories are used.

2.3.2.4 990/10 CPU Address Space. The 990/10 CPU uses a 16-bit address word inside the processor to define a 64K X 8 bits address space. The standard software reserves certain spaces for particular functions as shown in figure 2-11.

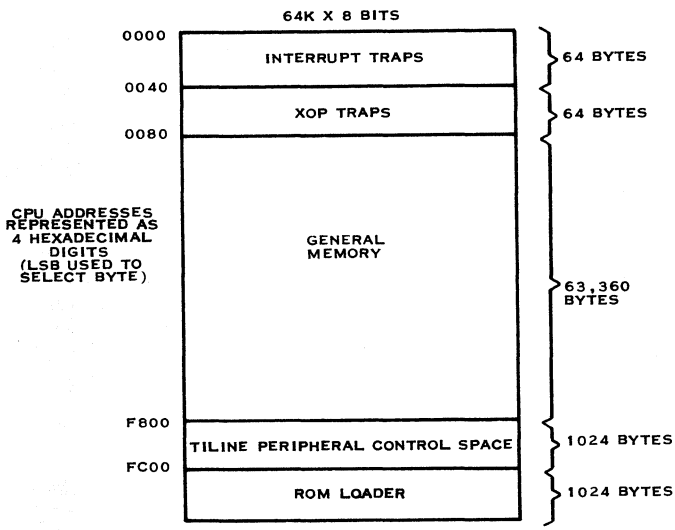


Figure 2-11. CPU Address Space

2.3.2.5 990/10 Without Map Option. When the 990/10 is provided without the mapping option, then CPU addresses are mapped into TILINE addresses according to a fixed scheme. The CPU reaches only the lowest 31K plus the 512 words of the TILINE address space. Refer to figure 2-12 for a pictorial of the 990/10 CPU–TILINE address space without map option.

2.3.2.6 990/10 With Map Option. The 990/10 map option provides three sets of mapping registers, each of which defines up to 32K X 16 bit address space which may be located anywhere in the 1024K X 16 TILINE address space on 16-word boundaries.

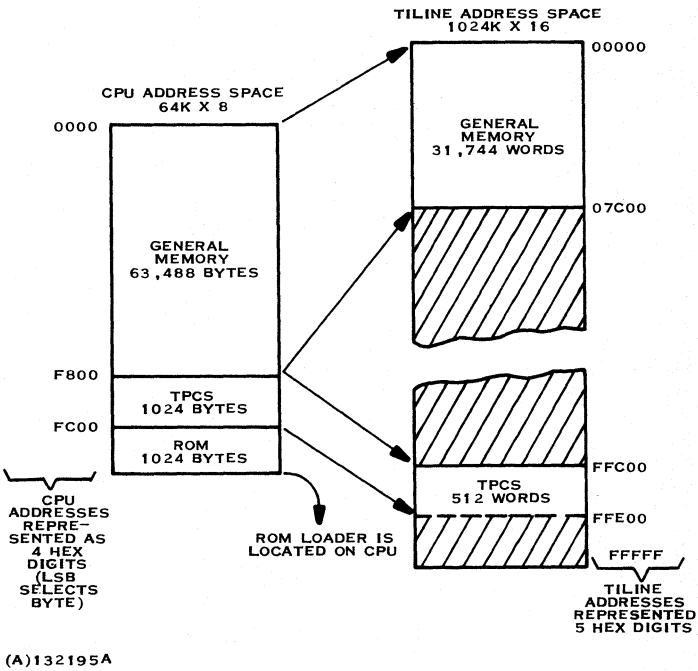


Figure 2-12. 990/10 CPU-TILINE Address Space without Map Option

Each set of map registers provides three 16-bit bias registers: B1, B2, and B3; and three 11-bit limit registers: L1, L2, and L3. The limit registers define three zones in the CPU address space on 16 word boundaries:

Zone 1: CPU address $\leq$ L1
Zone 2: L1 < CPU address $\leq$ L2
Zone 3: L2 < CPU address $\leq$ L3

CPU addresses that do not satisfy any of the above tests cause address violation traps. When more than one limit test is satisfied, the bias register selected is always related to the first successful limit test. That is, if both L1 and L2 are greater than the CPU address, B1 is selected.

Refer to figure 2-13 for a pictorial of the 990/10 CPU - TILINE address space with map option. CPU addresses in Zone 1 are remapped in consecutive order to TILINE addresses beginning with Bias 1 (B1) and extending to B1 + L1/2 + 15. Zone 2 uses Bias 2 and Zone 3 uses Bias 3. Note that Zone 2 addresses on the TILINE do not start with

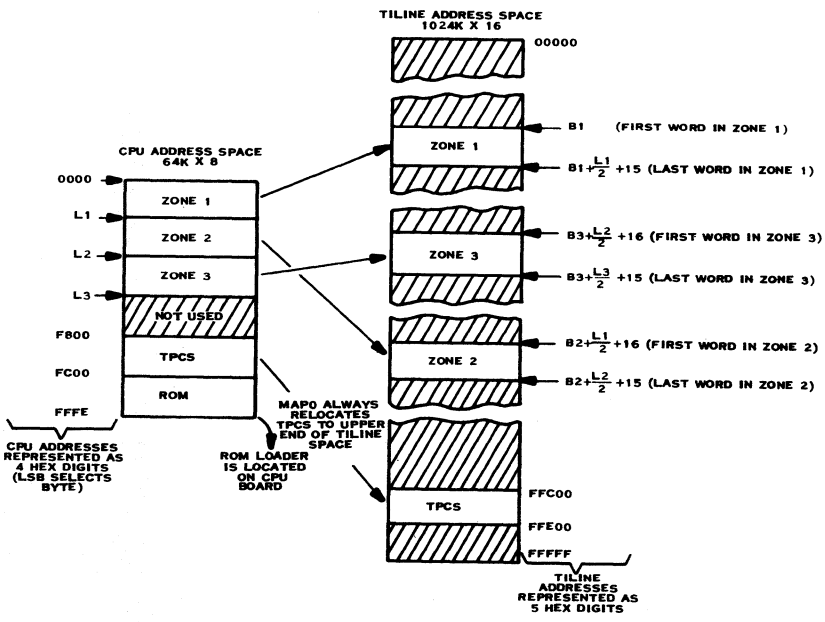


Figure 2-13. 990/10 CPU-TILINE Address Space with Map Option

Bias 2 (B2) but instead start at B2 + L1/2 + 16. Similarly Zone 3 addresses start at B3 + L2/2 + 16. This is simply an implementation convenience and does not restrict the programmer in locating zones anywhere he chooses in TILINE space. (The address logic is closed, thus the next address beyond FFFFF is 00000.) The zones are ordered in CPU address space, but may be relocated in any order in TILINE space by the bias registers.

The three map sets (each having three limit and three bias registers) are normally dedicated in software to software functions. MAP0 is used by the monitor and has special hardware features to facilitate monitor operations: Interrupts and XOPs always trap via MAP0, and MAP0 always reaches the TILINE Peripheral Control Space and ROM program. MAP1 is the normal user task map and MAP2 is used by special "long distance" instructions to provide monitor controlled user access outside the normal zones allowed for that user.

In addition to the map sets, mapping may be disabled on command, in which case mapping reverts to the scheme outlined under 990/10 without map option.

Figures 2-14 and 2-15 depict address modification and map registers, respectively.

2.3.2.7 Interrupts. The 990/10 CPU provides 16 interrupt levels. Levels are numbered 0 through 15 with level 0 having highest priority. The 4-bit interrupt mask stores the level number of any interrupt presently executing and prevents interrupts from the same or lower levels from interrupting. There is an acknowledge signal for each interrupt to provide synchronization, so that each interrupt signal is honored once and only once. Each interrupt level is uniquely associated with a two-word location in memory, the trap address, which directs the CPU to the start of the interrupting program and replaces the workspace pointer to provide the workspace associated with the interrupting program.

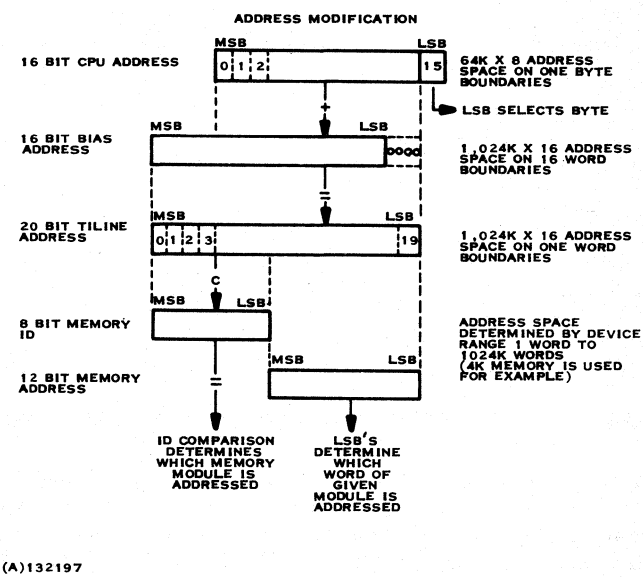


Figure 2-14. Address Modification

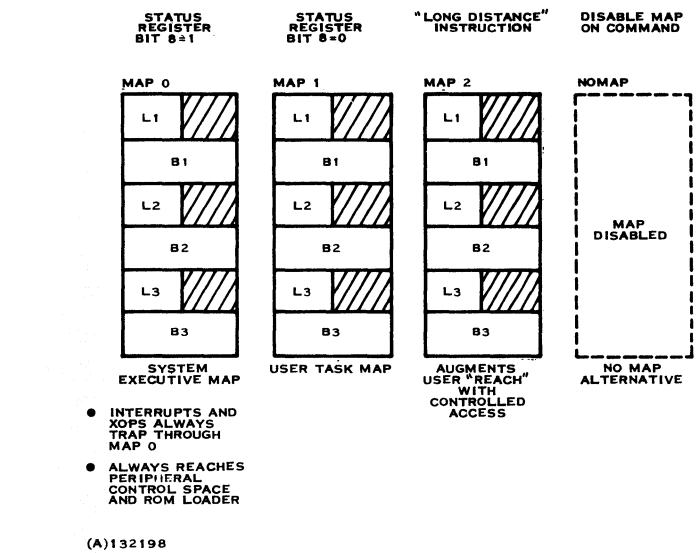


Figure 2-15. Map Registers

Three of the 16 interrupt levels are reserved for internal operations - power up and so forth. Levels 3 through 15 are available to the user. Level 5 or 15 will be used by the real time clock.

2.3.2.8 CPU Self Test. The 990/10 CPU self-test option contains a test routine in ROM which is executed before the LOAD function. A failure in self test is indicated by the fault indicator on the front panel.

Further testing and fault isolation is accomplished by Performance Assurance Tests (Pats) which may be loaded via magnetic tape cassettes.

2.4 990 CHASSIS

There are three basic alternatives in chassis, the OEM chassis, the 6-slot chassis, and the 13-slot chassis.

2.4.1 OEM CHASSIS. The OEM chassis is a skeleton chassis intended for breadboarding or for built-in mounting in custom OEM equipment. The OEM chassis provides connectors and mounting space for three full cards. Two half cards may be substituted for one full card in any slot. Interconnect wiring is in the backplane etch. Interrupt levels are assigned by jumper-plug options. The user must supply power and cooling.

2.4.2 SIX-SLOT CHASSIS. The six-slot chassis provides full support, including backplane wiring, power, and cooling air, for six full cards. Two half cards may be substituted for one full card in any slot. The chassis supports either 990/4 or 990/10 CPU and distributes CRU and TILINE (for 990/10) on backplane wiring as explained in paragraphs 2.1.10 and 2.1.11. Chassis dimensions and power supply capacity are provided in Section VI. (Figure 2-16 and 2-17)

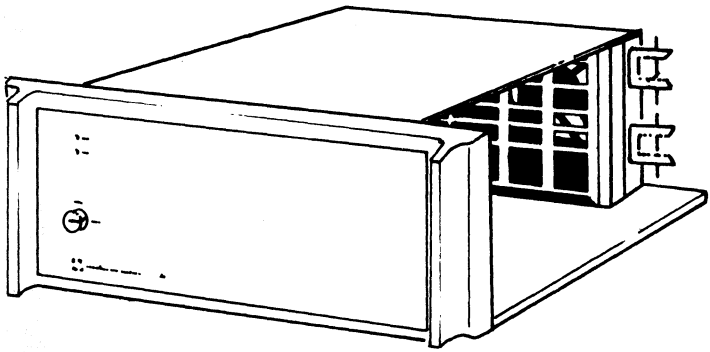
2.4.3 THIRTEEN-SLOT CHASSIS. The thirteen-slot chassis provides full support including backplane wiring, power and cooling air for thirteen full cards. Two half cards may be substituted for one full card in any slot. The chassis supports either 990/4 or 990/10 CPU and distributes CRU and TILINE (for 990/10) on backplane wiring as explained in paragraphs 2.1.10 and 2.1.11. The thirteen-slot chassis can be used for CPU or I/O expansion. Chassis dimension and power capacity are provided in Section VI. (Figure 2-18 and 2-19)

2.4.4 RACK MOUNTING OPTION. The rack mounting option provides slides and hardware for mounting either chassis in standard 19-inch RETMA racks.

2.4.5 TABLETOP OPTION. The tabletop option provides an attractive dust proof enclosure for the six-slot chassis only for desk or tabletop operation.

2.4.6 OPERATOR PANEL. This panel is the basic front panel for either chassis (figure 2-16). These chassis can be used for CPU or I/O expansion. The operator panel itself is basically a blank panel with indicators and switch as shown in figure 2-16.

2.4.7 PROGRAMMER PANEL. This panel is intended for applications requiring software troubleshooting. Panel controls (figure 2-17) may be locked out by the key switch to prevent unauthorized tampering. Note that this panel requires firmware DSR for installation (see paragraph 2.2.2.5).



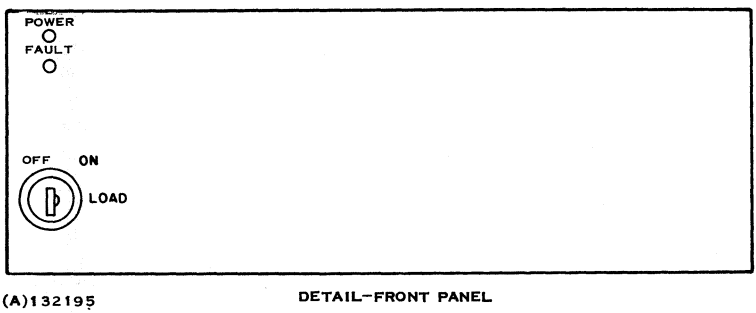
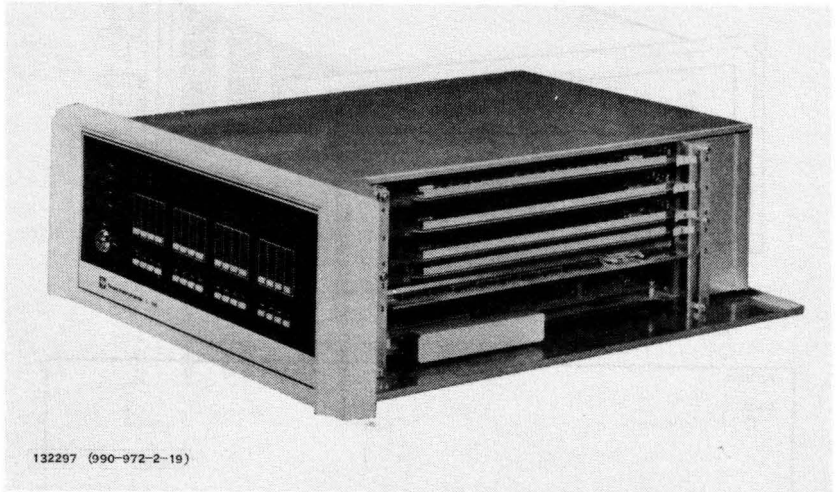
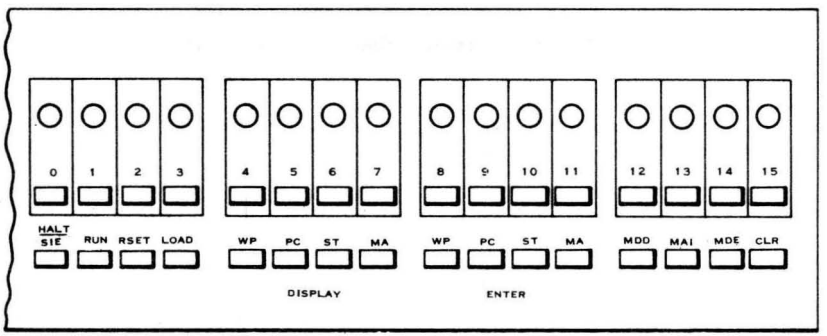


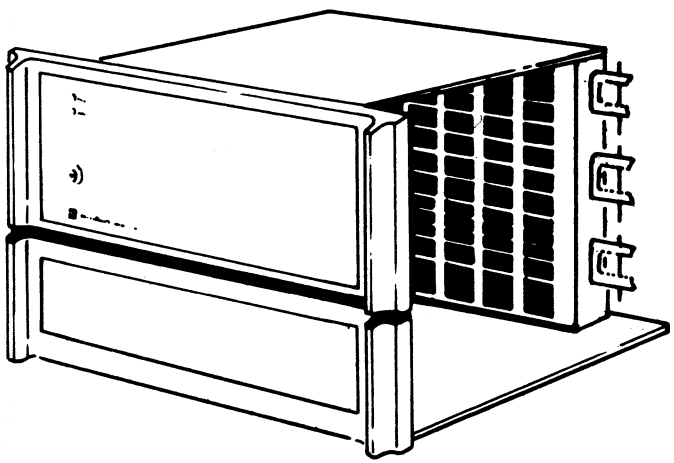
Figure 2-16. Operator Panel – 6 Slot Chassis

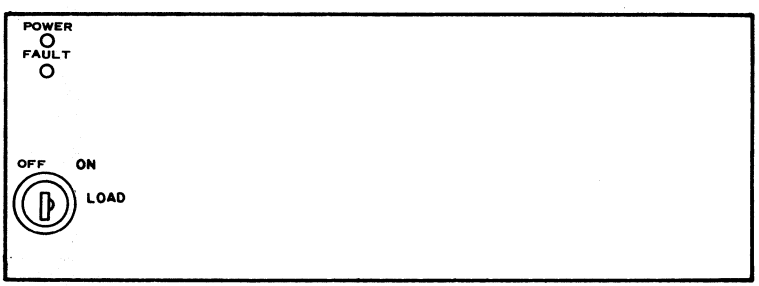




DETAIL–FRONT PANEL

Figure 2-17. Programmer Panel – 6 Slot Chassis

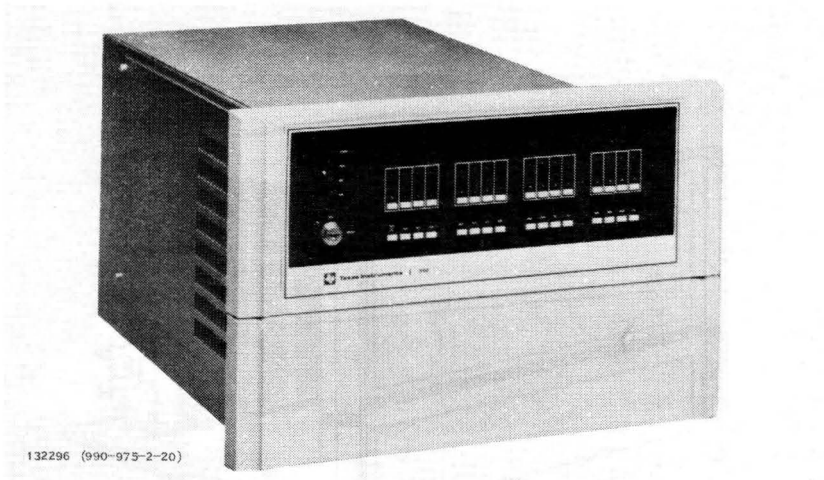


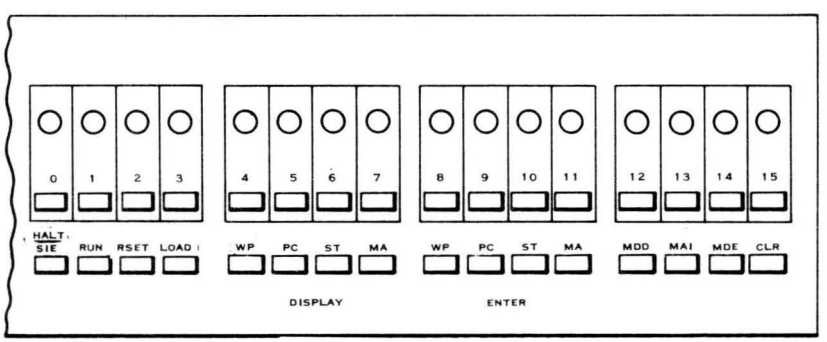


DETAIL–FRONT PANEL

(A)132197

Figure 2-18. Operator Panel – 13 Slot Chassis





DETAIL–FRONT PANEL

Figure 2-19. Programmer Panel – 13 Slot Chassis

SECTION III

990 INSTRUCTIONS AND ASSEMBLY LANGUAGE

3.1 INTRODUCTION

The instruction set of the Model 990 Computer family readily lends itself to simple and effective programming that results in efficient processing. This section describes the instruction set and the assembler directives that are used with the instruction set to form source programs for the assemblers supported by Texas Instruments. This section also includes guidelines for writing source programs, a description of memory addressing techniques, and an example program. For full details of the assembly language, refer to the *Model 990 Computer Assembly Language Programmer's Guide.*

3.1.1 VERSATILITY OF INSTRUCTIONS. Of the 69 instructions in the Model 990 Computer instruction set, 34 instructions allow a choice of one of five addressing modes for one or both of the operands. The addressing modes are:

- Workspace register addressing
- Workspace register indirect addressing
- Symbolic memory addressing
- Indexed memory addressing
- Workspace register indirect autoincrement addressing.

The versatility provided by these addressing modes allows these instructions to operate on data in workspace registers or data in memory locations. Data in memory locations may be addressed directly or indirectly. Direct addresses may be indexed, and indirect addresses may be automatically incremented. The autoincrement indirect address allows processing every byte or word in an array or table without the necessity of including an add instruction to increment the address.

Any instruction that operates on the contents of a memory location can also perform input to or output from any TILINE device, because the registers of the device controller are accessed in the same manner as memory addresses. Effectively, each of these memory instructions becomes a TILINE Input/Output instruction. Therefore, the instruction set does not include a separate TILINE I/O instruction.

The instructions that transfer groups of bits to or from devices connected to the Communications Register Unit (CRU) also provide versatility. Device controllers connected to the CRU may require one or more fields or groups of bits, and these fields may consist of one or more bits each. Typically, a controller requires a data field, control lines, and status lines. An input instruction transfers a field of one to

sixteen bits from a CRU device controller to memory, and another instruction transfers a field of from one to sixteen bits from memory to a CRU device controller. When the number of bits to be transferred is eight bits or less, both instructions operate as byte instructions. When the number of bits transferred is greater than eight, the instructions operate as word instructions. The instruction set also includes instructions that transfer a single bit to or from the device controller.

3.1.2 EFFICIENCY. A key concept implemented in the Model 990 Computer is the workspace concept. The workspace is a 16-word area of memory that is addressable as workspace registers. Workspace registers may be used to contain operands of instructions, and as accumulators, to contain the results of an operation. Workspace registers may also be used to contain data addresses, or index values to be used in address development.

As shown in figure 3-1, the Workspace Pointer (WP) register contains the address of the first word of the workspace (workspace register 0). Workspace register 0 is optionally used by the shift instructions to contain the shift count. Workspace register 11 is used by the branch and link instruction to store Program Counter (PC) contents as a link to the branching program. Workspace register 12 is used by the CRU instructions as the base address register for CRU I/O. Workspace register 13 contains WP register contents, workspace register 14 contains PC contents, and workspace register 15 contains Status (ST) register contents during context switch operations. Use of the other ten workspace registers is not preempted in any way. The workspace register concept promotes efficiency because an entire new set of workspace register contents becomes applicable simply by changing the contents of the WP register.

Typically, a change of workspace is accomplished as part of a context switch. A context switch is a transfer of control that uses a 2-word transfer vector to define the new environment. A context switch consists of the following operations:

- The first word of a transfer vector is placed in the WP register
- The second word of the same transfer vector is placed in the PC
- The previous contents of the WP register is stored in WR 13 of the new workspace
- The previous contents of the PC is stored in WR 14 of the new workspace
- The contents of the ST register is stored in WR 15 of the new workspace.

A context switch transfers control to a program or subroutine, and activates a workspace associated with the program or subroutine as control passes to the new PC contents. The context switch stores the program environment, that is, the workspace address, the address of the next instruction in sequence, and the program status. A

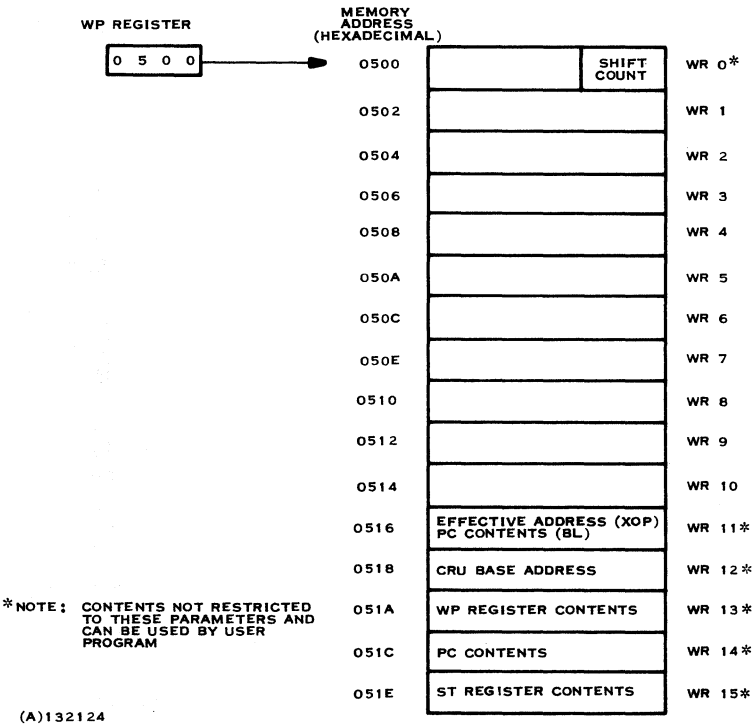


Figure 3-1. Typical Model 990 Computer Workspace

return instruction restores the environment by returning the stored data to the WP register, the PC, and the ST register. Context switches transfer control to subroutines when an interrupt occurs, when an extended operation instruction is executed, or when a branch and load workspace pointer instruction is executed.

3.1.2.1 Interrupt Context Switches. The Model 990 Computer provides vectored interrupts. Vectors for the interrupts are transfer vectors for context switches, and contain a workspace address and an entry point address. Vectors for the available interrupt levels are stored in interrupt level sequence starting at address zero. The level of interrupt is the priority of the interrupt. Level 0 is the highest priority and level 15 is the lowest priority. Interrupts are enabled by means of an interrupt mask, the four least significant bits of the ST register. The value in the mask specifies the lowest enabled level; all higher levels are enabled also. Table 3-1 lists the interrupt levels available in the Model 990/10 Computer. The table also applies to the Model 990/4 Computer, except that only levels 0 through 7 are available.

Table 3-1. Model 990/10 Computer Interrupt Level Data

Interrupt Level	Vector Location (Trap Address)	Assignment	Enabling Mask Values (Hexadecimal)
0	0000	Power On	0 through F
1	0004	Power Failing	1 through F
2	0008	Error	2 through F
3	000C	External Device	3 through F
4	0010	External Device	4 through F
5	0014	Line Frequency Clock (Optional)	5 through F
6	0018	External Device	6 through F
7	001C	External Device	7 through F
8	0020	External Device	8 through F
9	0024	External Device	9 through F
10	0028	External Device	A through F
11	002C	External Device	B through F
12	0030	External Device	C through F
13	0034	External Device	D through F
14	0038	External Device	E and F
15	003C	Line Frequency Clock (Optional)	F only

The vectored interrupt scheme of the Model 990 Computer, together with the transfer to the interrupt routine by means of a context switch, results in highly efficient processing of interrupts. When a pending interrupt condition has a priority higher than or equal to the interrupt mask contents, a context switch occurs following execution of the currently-executing instruction. The interrupt mask is then set to enable the level having the next higher priority, disabling the level of the current interrupt and all lower priority interrupts. The transfer of control, activation of a workspace, and storing of the context for processing of an interrupt are shown in figure 3-2.

In the Model 990/10 Computer, the Privileged Mode bit of the ST register is set to 0 when an interrupt occurs. This enables Privileged Mode instruction execution. When the map option is included, the Map File bit is also set to zero.

The vectored interrupt scheme is faster than other schemes because a single memory access obtains the address of the interrupt routine, rather than an indirect address or an instruction that branches to the routine. The context switch makes workspace registers available for processing the interrupt without having to store previous contents of a register file in the Arithmetic Logic Unit (ALU) and load these registers with values required to process the interrupt.

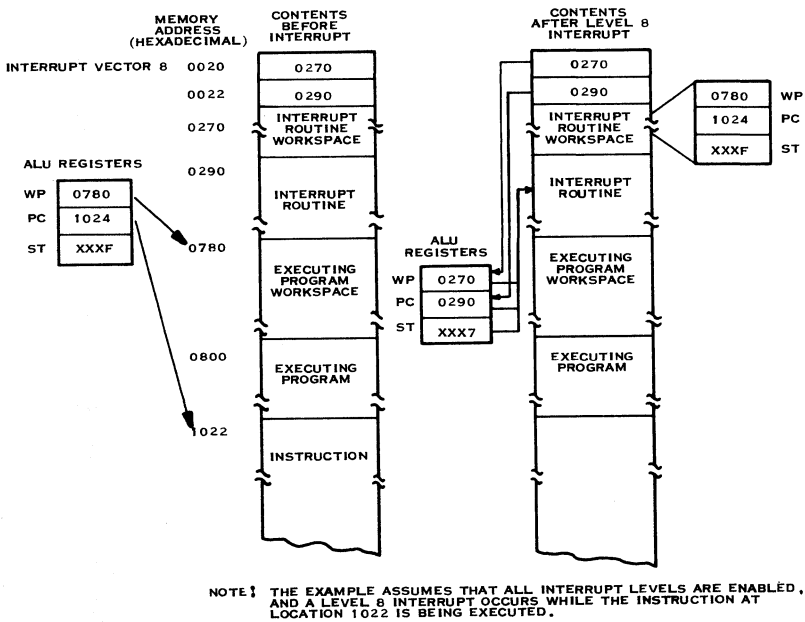


Figure 3-2. Interrupt Processing

Typical timing to accomplish transfer of control to an interrupt routine in the Model 990 Computer is 10 microseconds. This is significantly faster than contemporary computers using other interrupt schemes that require considerably more time to transfer control to the interrupt routine. Figure 3-3 shows the maximum number of interrupts that can be processed per second (as limited by transfer overhead) for three contemporary computers using various less efficient interrupt schemes, and the Model 990 Computer using vectored interrupts and context switching.

3.1.2.2 XOP Context Switches. The Model 990 Computer provides up to 16 extended operations to perform user-defined operations using software routines or hardware modules. The XOP instruction performs a context switch to a specified extended operation routine using a transfer vector in low-order memory. The context switch for an extended operation is similar to that previously described for an interrupt, except that the specified operation number selects a transfer vector in the memory area from 0040_{16} through $007F_{16}$. The computer develops the address in

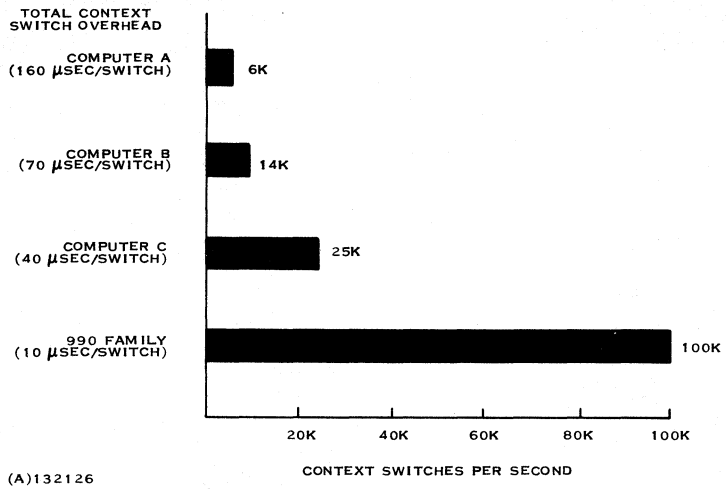


Figure 3-3. Comparison of Context Switching Efficiency

the XOP instruction and places the resulting effective address in workspace register 11 of the new workspace. An example of the use of an extended operation is the supervisor call from a user program to the disc executive, which is implemented as extended operation 15.

For the Model 990/10 Computer, the user may implement one or more (up to 16) extended operations by designing hardware modules to perform the special operations, such a hardware floating point processor. When a hardware module is connected for a given extended operation, the hardware is activated by an XOP instruction that specifies the operation, and no context switch occurs.

3.1.2.3 LREX Context Switch. The Model 990 Computer instruction set includes the LREX (Load or Restart Execution) instruction that performs a context switch using the transfer vector in address $FFFC_{16}$. Typically, the transfer vector and the routine that receives control are in Read Only Memory (ROM). The programmer panel routine is an example of a routine accessed by the LREX instruction.

3.1.2.4 BLWP Context Switch. The user may write subroutines to be accessed by a context switch. The BLWP (Branch and Load Workspace Pointer) instruction is used to access these subroutines. The transfer vector for a BLWP instruction may be placed in any available memory location, (at an even address) and is usually placed in the user area of memory.

3.1.3 INSTRUCTION SUMMARY. The 72 instructions of the Model 990 Computer are summarized in the paragraphs that follow under nine functional categories. They form a versatile and powerful instruction set.

3.1.3.1 Arithmetic Instructions. The following are the arithmetic instructions of the Model 990 Computer instruction set:

Add Words	Increment
Add Bytes	Increment by Two
Add Immediate	Decrement
Subtract Words	Decrement by Two
Subtract Bytes	Absolute Value
Multiply	Negate.
Divide	

The Add Words and Add Bytes instructions add word and byte operands, respectively, replacing the destination operand with the sum. The Add Immediate instruction adds an immediate operand to a workspace register operand, placing the sum in the workspace register. The Subtract Words and Subtract Bytes instructions subtract a word or byte operand from another word or byte operand, and replace the minuend with the remainder. The Multiply instruction multiplies two 16-bit operands as unsigned integers, and places the product in two consecutive workspace registers. The Divide instruction divides a 32-bit dividend in a pair of workspace registers by a 16-bit divisor. Division is performed assuming the operands to be unsigned integers, and the quotient and remainder replace the dividend in the pair of workspace registers. The Increment and Increment by Two instructions add one and two, respectively, to the operand, and replace the operand with the sum. Similarly, the Decrement and Decrement by Two instructions subtract one and two from the operand, and replace the operand with the remainder. The Absolute Value instruction replaces a negative operand with its two's complement. The Negate instruction replaces the operand with its two's complement. The thirteen instructions in the arithmetic category provide a powerful group of arithmetic operations.

3.1.3.2 Branch Instructions. The branch instruction category includes 18 instructions that transfer control to a location other than the next location in sequence, conditionally or unconditionally. The instructions are:

Branch	Branch and Load Workspace Pointer
Branch and Link	Return with Workspace Pointer
Unconditional Jump	Jump if Equal
Jump if Logical High	Jump if Not Equal
Jump if Logical Low	Jump On Carry
Jump if High or Equal	Jump if No Carry
Jump if Low or Equal	Jump if No Overflow
Jump if Greater Than	Jump If Odd Parity
Jump if Less Than	Execute

The Branch instruction branches unconditionally to the specified memory location. The Branch and Link instruction also branches unconditionally to the specified memory location, and stores the PC contents in workspace register 11 as a link to the branching program. The Branch and Load Workspace Pointer instruction performs an unconditional context switch using the transfer vector at the specified location. The

Return with Workspace Pointer instruction restores the calling or interrupted program environment using the values stored by the most recent context switch. The operand of any of the jump instructions is a displacement (in words) in the range of −128 through +127 that is added algebraically to the PC contents. That is, transfers of control by jump instructions are program counter relative. The thirteen jump instructions include an Unconditional Jump, eight jumps related to the results of a comparison, and four jumps on other ST register bits. The Execute instruction specifies an instruction to be executed. The program counter is not altered unless the replacing instruction normally alters the program counter. The branch instruction category provides a great deal of flexibility for transfer of program control.

3.1.3.3 Compare Instructions. The compare instruction category includes five instructions that set and reset the condition code bits of the ST register to indicate the relationship between operands. The instructions are:

Compare Words	Compare Ones Corresponding
Compare Bytes	Compare Zeros Corresponding
Compare Immediate	

The Compare Words and Compare Bytes instructions compare words and bytes, respectively. The Compare Immediate instruction compares an immediate operand with the contents of a workspace register. These instructions compare the operands as two's complement values (arithmetic comparison) and as unsigned values (logical comparison) simultaneously. The other two compare instructions provide a means of comparing selected bits of the contents of a workspace register to ones or zeros. The source operand of each of these instructions is a pattern or mask in which one bits specify the bits to be compared. The Compare Ones Corresponding compares the bits of the destination operand specified by the mask to logical ones. The Compare Zeros Corresponding compares the bits of the destination operand to logical zeros.

3.1.3.4 Control and CRU Instructions. The control and CRU instruction category includes five control instructions and five CRU I/O instructions. The instructions are:

Reset	Set Bit to Logic Zero
Idle	See Bit to Logic One
Clock Off	Test Bit
Clock On	Load CRU
Load or Restart Execution	Store CRU

In the Model 990 Computers, the Reset instruction clears the interrupt mask, disabling all but level 0 interrupts, and resets directly connected I/O devices. The instruction resets CRU devices that provide for reset in the interface with the CRU,

resets pending interrupts, and turns the line frequency clock off. The Idle instruction places the computer in the idle mode. The Clock Off instruction turns the line frequency clock off, and the Clock On instruction turns the line frequency clock on. The Load or Restart Execution instruction performs a context switch using the transfer vector at location $FFFC_{16}$.

In the TMS9900 microprocessor, the control instructions function differently. The Idle instruction places the microprocessor in the idle mode, and generates an external signal. The Reset, Clock Off, Clock On, and Load or Restart Execution instructions perform no function in the TMS9900, but these instructions do generate external signals. In implementing a micro computer using the TMS9900, the user may provide logic to perform any desired function when the signal generated by each of these control instructions is detected.

The Set Bit to Logic One and Set Bit to Logic Zero instructions each transmit a single bit on a CRU line to an I/O device. The Set Bit to Logic One instruction sets the addressed bit to the value of one, and the Set Bit to Logic Zero sets the bit to the value of zero. The Test Bit instruction reads a single bit on a CRU line, and sets the equal bit in the ST register to the value of the addressed line. The single bit CRU I/O instructions specify the desired line as a displacement from the CRU base address. The Load CRU instruction transfers a group or field of 1 to 16 bits from memory to a group or field of contiguous CRU lines. The Store CRU instruction transfers a similar field from CRU lines to memory. For these instructions the CRU base address must be set to the address of the lowest numbered CRU line in the group.

3.1.3.5 Load and Move Instructions. The load and move instruction category includes four instructions that load workspace registers, ALU registers, and mapping registers. The category also includes three move instructions, and two instructions that store ALU register contents. The instructions are:

Load Immediate	Move Byte
Load Interrupt Mask Immediate	Swap Bytes
Load Workspace Pointer Immediate	Store Status
Load Memory Map File	Store Workspace Pointer.
Move Word	

The Load Immediate instruction places the immediate operand into the specified workspace register. The Load Interrupt Mask Immediate instruction places the four least significant bits of the immediate operand into the interrupt mask, the four least significant bits of the ST register. The Load Workspace Pointer Immediate instruction places the immediate operand into the WP register.

The Load Memory Map File instruction applies only to the Model 990/10 Computer with the mapping option. The mapping option allows the 64K addresses available within the instruction format to be mapped to memory addresses in a 20-bit addressing format. The mapping option provides three map files consisting of six registers each. Three of the registers contain limits in one's complement form, and the other three registers contain bias addresses. (Refer to figure 3-4.) ALU addresses are compared to the one's complement of the limit registers. ALU addresses between zero and the contents of Limit 1 (L1) are added to Bias 1 (B1). ALU addresses greater than the contents of L1 and less than or equal to the contents of Limit 2 (L2) are added to Bias 2 (B2). ALU addresses greater than the contents of L2 and less than or equal to the contents of Limit 3 (L3) are added to Bias 3 (B3). ALU addresses greater than the contents of L3 are not accessible. The addition of ALU addresses to B1, B2, and B3 results in 20-bit addresses. The contents of the bias register is shifted to the left five bit positions, and added to the ALU address. The five least significant bits of the ALU address are retained, and the least significant bit is ignored.

The Load Memory Map File instruction places the contents of a six-word area in memory at the address contained in the specified workspace register into a map file. Map files 0 or 1 can be loaded by this instruction. Use of map file 2 is described in a subsequent paragraph.

The Move Word and Move Byte instructions copy the data in the source operand into the destination operand. The Swap Bytes instruction exchanges the bytes in a workspace register or memory word.

The Store Status instruction stores the contents of the ST register in a workspace register, and the Store Workspace Pointer instruction stores the contents of the WP register in a workspace register.

3.1.3.6 Logical Instructions. The logical instruction category includes 10 instructions that perform logical functions on words or bytes. The instructions are:

AND Immediate	Set to Ones
OR Immediate	Set Ones Corresponding
Exclusive OR	Set Ones Corresponding, Byte
Invert	Set Zeros Corresponding
Clear	Set Zeros Corresponding, Byte.

The AND Immediate instruction performs an AND operation between the immediate operand and the contents of a workspace register, and places the result in a workspace register. The OR Immediate instruction performs an OR operation between the immediate operand and the contents of a workspace register, and places the result in a workspace register. The Exclusive OR instruction performs an exclusive OR operation between an operand in memory or a workspace register and another operand in a

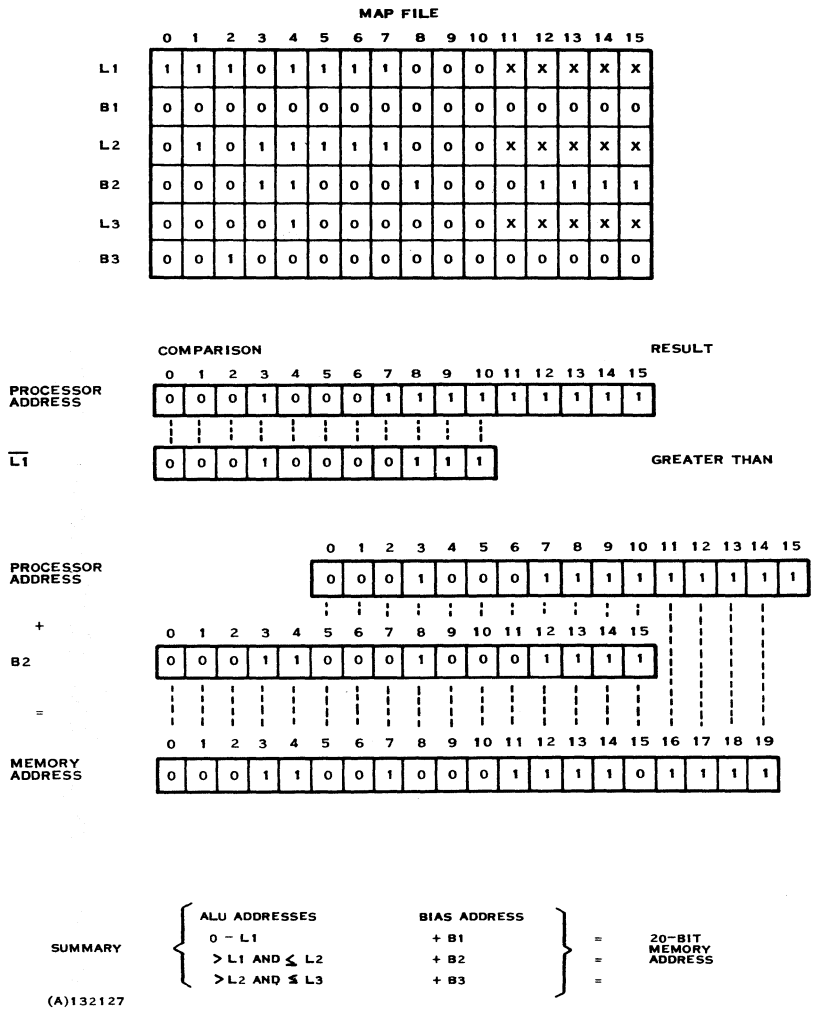


Figure 3-4. Development of Memory Address

workspace register, and replaces the operand in the workspace register. The Invert instruction replaces the operand with its one's complement, and the Clear instruction sets the bits of the operand to zeros. The Set to One instruction sets the bits of the operand to ones. The remaining logical instructions all use the source operand as a pattern or mask to select bits of the destination operand to be set. Ones in the source operand correspond to bits to be set, and zeros in the source operand correspond to bits to remain unaltered. The Set Zeros Corresponding sets bits of the destination word to zeros, corresponding to the mask. The Set Zeros Corresponding, Byte sets bits of the destination byte to zeros. The Set Zeros Corresponding instructions are effectively AND instructions that combine the one's complement of the mask with the destination operand. The Set Ones Corresponding instruction sets bits of the destination word to ones, corresponding to the mask. The Set Ones Corresponding, Byte instruction sets bits of the destination byte to ones. The Set Ones Corresponding instructions are effectively OR instructions that combine the two operands.

3.1.3.7 Shift Instructions. The shift instructions shift the bits in a workspace register according to the shift count. The instructions are:

Shift Right Arithmetic	Shift Right Logical
Shift Left Arithmetic	Shift Right Circular.

Shift instructions take the shift count from an operand, or from the four least significant bits of workspace register 0 when the shift count operand contains zero. When both the shift count operand and the four least significant bits of workspace register 0 contain zero, the register is shifted 16 bit positions. The carry bit of the ST register contains the last bit shifted out of the register.

The Shift Right Arithmetic instruction shifts the contents of a specified workspace register to the right, filling the vacated bit positions with the sign bit. The Shift Left Arithmetic instruction shifts the contents of the specified workspace register to the left, filling the vacated bit positions with zeros. The Shift Right Logical instruction shifts the contents of the specified workspace register to the right, filling the vacated bit positions with zeros. The Shift Right Circular instruction shifts the contents of the specified workspace register to the right, filling the vacated bit positions with the bits shifted off the right end of the workspace register.

3.1.3.8 Extended Operation Instruction. The Extended Operation instruction is a means of adding up to 16 additional computer operations, as required, to the Model 990 Computer instruction set. The user may supply a subroutine to perform the operation and place a transfer vector consisting of the workspace address and the entry address in the appropriate address (0040_{16} through $007C_{16}$). The Extended Operation instruction performs a context switch using the transfer vector corresponding to the extended operation number. In the Model 990/10 Computer, the user may supply a hardware module to perform the operation. When a hardware module is connected for the specified extended operation, the instruction activates the hardware instead of performing the context switch.

3.1.3.9 Long Distance Addressing Instructions. The long distance addressing instructions are available in the Model 990/10 Computer with the map option. These instructions enable accesses outside of the current memory map for a single address. The instructions are:

Long Distance Source Long Distance Destination.

Each instruction places the contents of a 6-word area of memory in map file 2. The first word of the 6-word area is at the source operand address. The Long Distance Source instruction loads map file 2 and assigns map file 2 to the source address of the next instruction. The Long Distance Destination address loads map file 2 and assigns map file 2 to the destination address of the next instruction.

3.2 MACHINE INSTRUCTIONS

The machine instructions of the Model 990 Computer instructions are listed in table 3-2. The machine formats are shown on the following pages, along with the operation code, execution results, and list of status bits affected. The operation codes (Op Codes) are shown as four digit hexadecimal numbers in which operand fields contain zero. The machine instruction formats use the following abbreviations:

- T_S Mode bits for source address, defined in table 3-2.
- S Workspace register field for source address, specifies the workspace register that contains the operand, the address of the operand, or the index value for the address of the operand.
- T_d Mode bits for destination address, defined in table 3-2.
- D Workspace register field for destination address, specifies the workspace register that contains the operand, the address of the operand, or the index value for the address of the operand.
- W Workspace register field for a workspace operand.
- C Count field, containing either a bit count or a shift count.
- M Map file field

In the execution results, the following conventions and abbreviations apply:

- ga_s General address, source. An address in one of the modes defined in table 3-2.
- ga_d General address, destination. An address in one of the modes defined in table 3-2.
- wa Workspace register address. Specifies a workspace register that contains an operand.

- iop Immediate operand
- ( ) Specifies the contents of an address or register.
- | | Specifies an absolute value
- → Specifies "replaces"

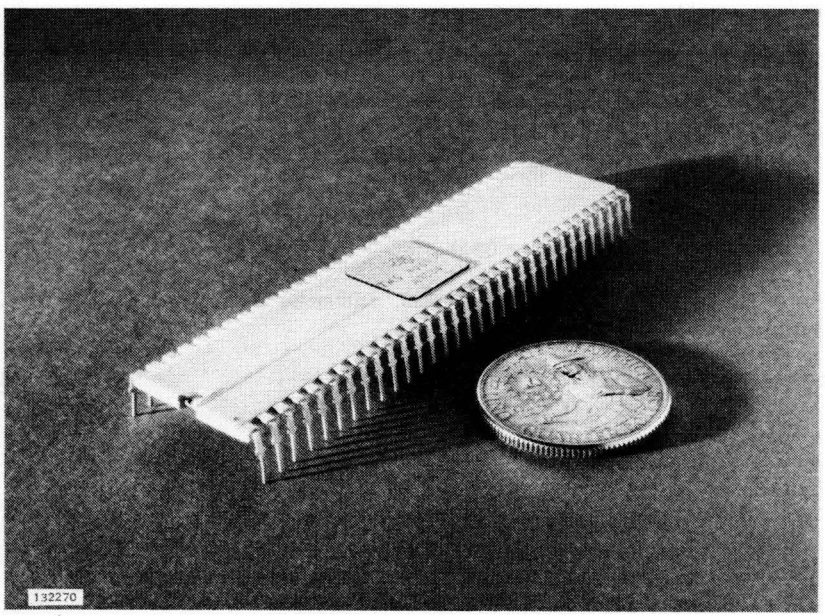
Table 3-2. Machine Instruction Addressing Modes

Mode Bits	Addressing Mode
00_2	Workspace Register Addressing
01_2	Workspace Register Indirect Addressing
10_2	Symbolic Memory Addressing, when corresponding workspace register field contains 0
10_2	Indexed Memory Addressing, when corresponding workspace register field contains a value greater than 0
11_2	Workspace Register Indirect Autoincrement Addressing

The status bits affected by the instruction are bits in the ST register, defined as follows:

- Logical greater than - result of a comparison of operands as unsigned numbers.
- Arithmetic greater than - result of a comparison of operands as two's complement numbers.
- Equal - result of a comparison.
- Carry - carry out of most significant bit of an operand.
- Overflow - result too large or too small to be correctly represented in a byte or word (as applicable) in two's complement form.
- Odd parity - result of operation is a byte that contains an odd number of one bits.
- Extended operation - set as a software-implemented extended operation is initiated.

The comparison that sets or resets the logical greater than, arithmetic greater than, and equal bits is a comparison of the result to zero, except for compare instructions. Compare instructions set or reset these bits according to the result of comparing two operands.

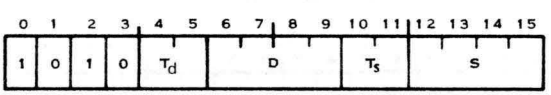


The instructions apply to the TMS9900 microprocessor and the Model 990 Computer except as noted.

3.2.1 ADD WORDS A

Op Code: A000

Format:



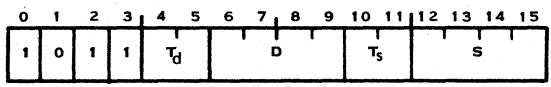
Execution results: $(ga_s) + (ga_d) \rightarrow (ga_d)$

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, and overflow.

3.2.2 ADD BYTES AB

Op Code: B000

Format:



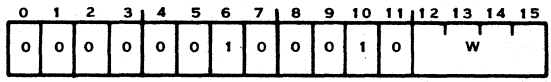
Execution results: $(ga_s) + (ga_d) \rightarrow (ga_d)$

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, overflow and odd parity.

3.2.3 ADD IMMEDIATE AI

Op Code: 0200

Format:



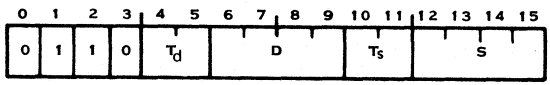
Execution results: (wa) + iop → (wa)

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, and overflow.

3.2.4 SUBTRACT WORDS S

Op Code: 6000

Format:



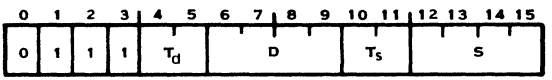
Execution results: $(ga_d) - (ga_s) \rightarrow (ga_d)$

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, and overflow.

3.2.5 SUBTRACT BYTES SB

Op Code: 7000

Format:



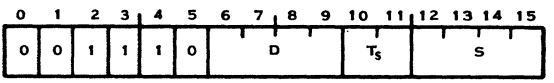
Execution results: $(ga_d) - (ga_s) \rightarrow (ga_d)$

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, overflow, and odd parity.

3.2.6 MULTIPLY MPY

Op Code: 3800

Format:



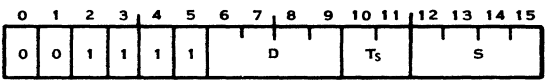
Execution results: $(ga_s) \cdot (wa_d)$. The product (32-bit magnitude) is placed in wa_d and $wa_d + 1$, with the most significant half in wa_d.

Status bits affected: None

3.2.7 DIVIDE DIV

Op Code: 3C00

Format:



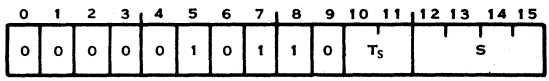
Execution results: The contents of wa_d and $wa_d + 1$ (32-bit magnitude) are divided by the contents of ga_s and the quotient is placed in wa_d. The remainder is placed in $wa_d + 1$.

Status bits affected: Overflow.

3.2.8 INCREMENT INC

Op Code: 0580

Format:



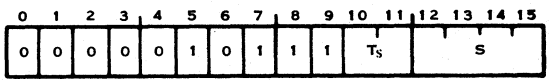
Execution results: $(ga_s) + 1 \rightarrow (ga_s)$

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, and overflow.

3.2.9 INCREMENT BY TWO INCT

Op Code: 05C0

Format:



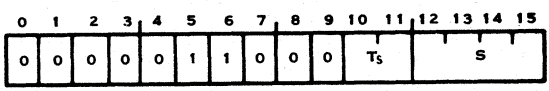
Execution results: $(ga_s) + 2 \rightarrow (ga_s)$

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, and overflow.

3.2.10 DECREMENT DEC

Op Code: 0600

Format:



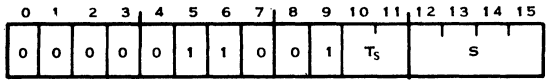
Execution results: $(ga_S) - 1 \rightarrow (ga_S)$

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, and overflow.

3.2.11 DECREMENT BY TWO DECT

Op Code: 0640

Format:



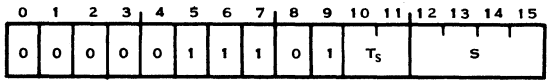
Execution results: $(ga_S) - 2 \rightarrow (ga_S)$

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, and overflow.

3.2.12 ABSOLUTE VALUE ABS

Op Code: 0740

Format:



Execution results: $|(ga_S)| \rightarrow (ga_S)$

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, and overflow.

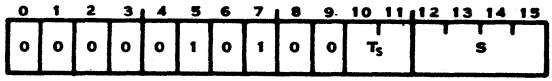
NOTE

These status bits are affected by contents of the source operand before instruction execution.

3.2.13 NEGATE NEG

Op Code: 0500

Format:



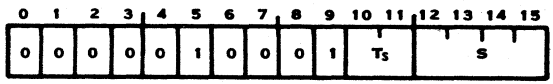
Execution results: $-(ga_s) \rightarrow (ga_s)$

Status bits affected: Logical greater than, arithmetic greater than, equal, and overflow.

3.2.14 BRANCH B

Op Code: 0440

Format:



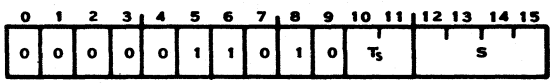
Execution results: $ga_s \rightarrow (PC)$

Status bits affected: None.

3.2.15 BRANCH AND LINK BL

Op Code: 0680

Format:



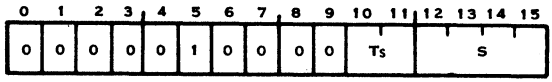
Execution results: (PC) → (Workspace register 11)
$ga_s \rightarrow (PC)$:

Status bits affected: None.

3.2.16 BRANCH AND LOAD WORKSPACE POINTER BLWP

Op Code: 0400

Format:



Execution results: (WP)→ (Workspace register 13)
(PC) → (Workspace register 14)
(ST) → (Workspace register 15)
(ga_s) → (WP)
(ga_s + 2) → (PC)

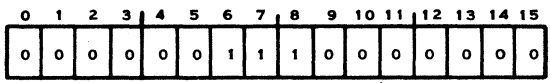
} "New" Workspace

Status bits affected: None.

3.2.17 RETURN WITH WORKSPACE POINTER RTWP

Op Code: 0380

Format:



Execution results: (Workspace register 13) → (WP)
(Workspace register 14) → (PC)
(Workspace register 15) → (ST)

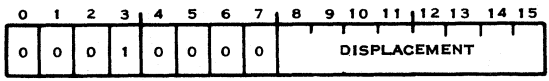
Status bits affected: Restores all status bits to the value contained in workspace register 15.

Model 990/10 Computer: In the Model 990/10 Computer with the Privileged Mode bit (bit 7) of the ST register set to 1, only bits 0 through 6 of workspace register 15 are placed in bits 0 through 6 of the ST register. When bit 7 of the ST register is set to 0, the instruction places all 16 bits of workspace register 15 in the ST register.

3.2.18 UNCONDITIONAL JUMP JMP

Op Code: 1000

Format:



Execution results: (PC) + Displacement → (PC)

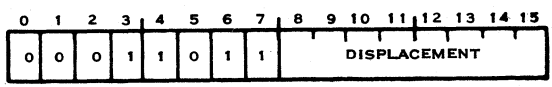
The PC is incremented to the address of the next instruction prior to execution of an instruction. The execution results of jump instructions refer to the PC contents after the contents have incremented to address the next instruction in sequence. The displacement (in words) is shifted to the left one bit position to orient the word displacement to the word address, and added to the PC contents.

Status bits affected: None.

3.2.19 JUMP IF LOGICAL HIGH JH

Op Code: 1B00

Format:



Execution results: If logical greater than bit is equal to 1 and equal bit is equal to 0: (PC) + Displacement → (PC)

If logical greater than bit is equal to 0 or equal bit is equal to 1: (PC) → (PC)

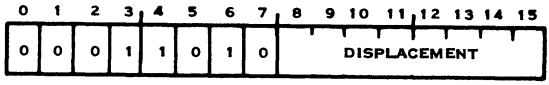
Refer to explanation of execution in paragraph 3.2.18

Status bits affected: None.

3.2.20 JUMP IF LOGICAL LOW JL

Op Code: 1A00

Format:



Execution results: If logical greater than bit and equal bit are equal to 0: (PC) + Displacement → (PC)

If logical greater than bit is equal to 1 or equal bit is equal to 1: (PC → (PC)

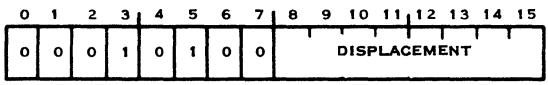
Refer to explanation of execution in paragraph 3.2.18

Status bits affected: None

3.2.21 JUMP IF HIGH OR EQUAL JHE

Op Code: 1400

Format:



Execution results: If logical greater than bit is equal to 1 or equal bit is equal to 1: (PC) + Displacement → (PC)

If logical greater than bit and equal bit are equal to 0: (PC) → (PC)

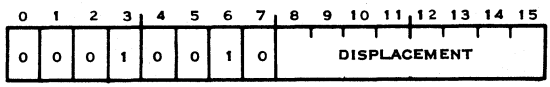
Refer to explanation of execution in paragraph 3.2.18

Status bits affected: None

3.2.22 JUMP IF LOW OR EQUAL JLE

Op Code: 1200

Format:



Execution results: If logical greater than bit is equal to 0 or equal bit is equal to 1: (PC) + Displacement → (PC)

If logical greater than bit is equal to 1 and equal bit is equal to 0: (PC) → (PC)

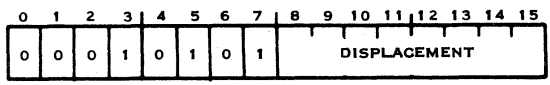
Refer to explanation of execution in paragraph 3.2.18.

Status bits affected: None.

3.2.23 JUMP IF GREATER THAN JGT

Op Code: 1500

Format:



Execution results: If arithmetic greater than bit is equal to 1: (PC) + Displacement → (PC)

If arithmetic greater than bit is equal to 0: (PC) → (PC)

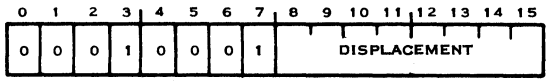
Refer to explanation of execution in paragraph 3.2.18.

Status bits affected: None.

3.2.24 JUMP IF LESS THAN JLT

Op Code: 1100

Format:



Execution results: If arithmetic greater than bit and equal bit are equal to 0: (PC) + Displacement → (PC)

If arithmetic greater than bit is equal to 1 or equal bit is equal to 1: (PC) → (PC)

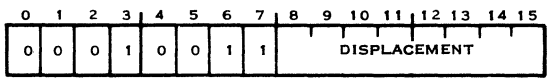
Refer to explanation of execution in paragraph 3.2.18.

Status bits affected: None

3.2.25 JUMP IF EQUAL JEQ

Op Code: 1300

Format:



Execution results: If equal bit is equal to 1: (PC) + Displacement → (PC)

If equal bit is equal to 0: (PC) → (PC)

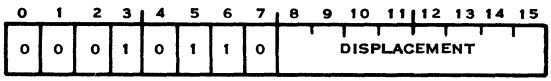
Refer to explanation of execution in paragraph 3.2.18.

Status bits affected: None.

3.2.26 JUMP IF NOT EQUAL JNE

Op Code: 1600

Format:



Execution results: If equal bit is equal to 0: (PC) + Displacement → (PC)

If equal bit is equal to 1: (PC) → (PC)

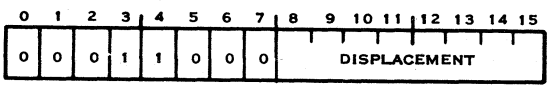
Refer to explanation of execution in paragraph 3.2.18.

Status bits affected: None.

3.2.27 JUMP ON CARRY JOC

Op Code: 1800

Format:



Execution results: If carry bit is equal to 1: (PC) + Displacement → (PC)

If carry bit is equal to 0: (PC) → (PC)

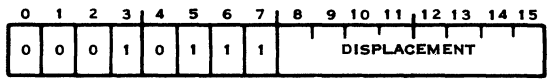
Refer to explanation of execution in paragraph 3.2.18

Status bits affected: None

3.3.28 JUMP IF NO CARRY — JNC

Op Code: 1700

Format:



Execution results: If carry bit is equal to 0: (PC) + Displacement → (PC)

If carry bit is equal to 1: (PC) → (PC)

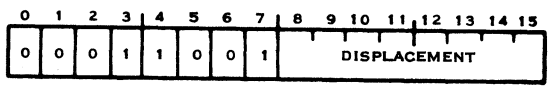
Refer to explanation of execution in paragraph 3.2.18.

Status bits affected: None.

3.2.29 JUMP IF NO OVERFLOW — JNO

Op Code: 1900

Format:



Execution results: If overflow bit is equal to 0: (PC) + Displacement → (PC)

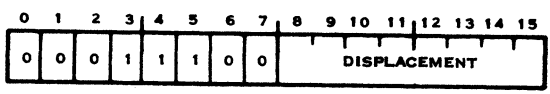
If overflow bit is equal to 1: (PC) → (PC)

Refer to explanation of execution in paragraph 3.2.18.

3.2.30 JUMP IF ODD PARITY — JOP

Op Code: 1C00

Format:



Execution results: If odd parity bit is equal to 1: (PC) + Displacement → (PC)

If odd parity bit is equal to 0: (PC) + 2 → (PC)

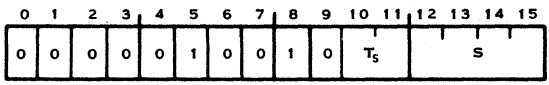
Refer to explanation of execution in paragraph 3.2.18.

Status bits affected: None.

3.2.31 EXECUTE X

Op Code: 0480

Format:

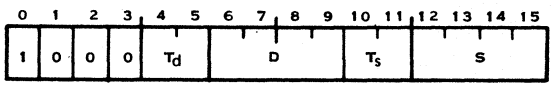


Execution results: An instruction at ga_s is executed. Status bits affected: None, but substituted instruction affects status bits normally.

3.2.32 COMPARE WORDS C

Op Code: 8000

Format:



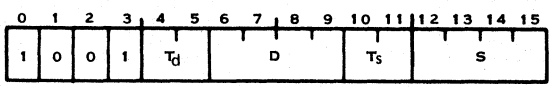
Execution results: (ga_s) : (ga_d)

Status bits affected: Logical greater than, arithmetic greater than, and equal.

3.3.33 COMPARE BYTES CB

Op Code: 9000

Format:



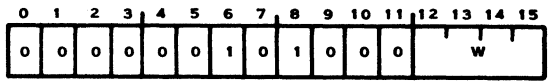
Execution results: $(ga_s):(ga_d)$

Status bits affected: Logical greater than, arithmetic greater than, equal, and odd parity.

3.2.34 COMPARE IMMEDIATE CI

Op Code: 0280

Format:



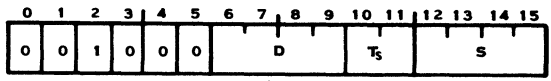
Execution results: (wa): iop

Status bits affected: Logical greater than, arithmetic greater than, and equal.

3.2.35 COMPARE ONES CORRESPONDING COC

Op Code: 2000

Format:



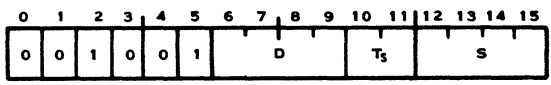
Execution results: Equal bit set if all bits of (wa) that correspond to the bits of (ga_s) that are equal to 1 are also equal to 1.

Status bit affected: Equal.

3.2.36 COMPARE ZEROS CORRESPONDING CZC

Op Code: 2400

Format:



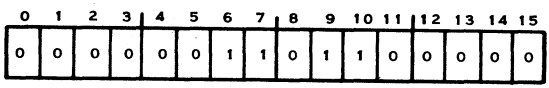
Execution results: Equal bit set if all bits of (wa) that correspond to the bits of (ga_S) that are equal to 1 are equal to 0.

Status bit affected: Equal

3.2.37 RESET RSET

Op Code: 0360

Format:



Execution results: Clears the interrupt mask, resets directly connected I/O devices, resets the CRU devices that provide for reset in the interface with the CRU, resets pending interrupts, and turns the clock off.

Status bits affected: Interrupt Mask

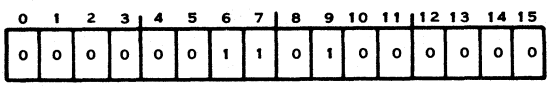
TMS9900 Microprocessor: Provides a signal that a RSET instruction is identified, but performs no processing. User may implement hardware to perform desired processing when the signal is present.

Model 990/10 Computer: When Privileged Mode (bit 7 of ST register) is set to 0, instruction executes normally. When Privileged Mode bit is set to 1, an error interrupt occurs when execution of an RSET instruction is attempted.

3.2.38 IDLE IDLE

Op Code: 0340

Format:



Execution results: Places the computer in the idle mode, suspending program execution until an interrupt occurs.

Status bits affected: None.

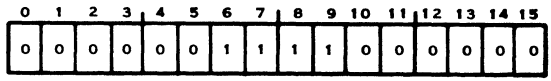
TMS9900 Microprocessor: Provides a signal that an IDLE instruction is being executed, and places the microprocessor in the idle mode. User may implement hardware to perform additional processing when the signal is present.

Model 990/10 Computer: When Privileged Mode bit (bit 7 of ST register) is set to 0, instruction executes normally. When Privileged Mode bit is set to 1, an error interrupt occurs when execution of a IDLE instruction is attempted.

3.2.39 CLOCK OFF — CKOF

Op Code: 03C0

Format:



Execution results: Line frequency clock disabled.

Status bits affected: None.

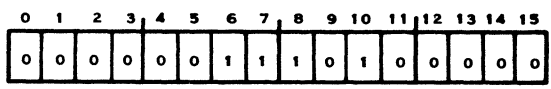
TMS9900 Microprocessor: Provides a signal that a CKOF instruction is identified, but performs no processing. User may implement hardware to perform desired processing when signal is present.

Model 990/10 Computer: When Privileged Mode bit (bit 7 of ST register) is set to 0, instruction executes normally. When Privileged Mode bit is set to 1, an error interrupt occurs when execution of a CKOF instruction is attempted.

3.2.40 CLOCK ON — CKON

Op Code: 03A0

Format:



Execution results: Line frequency clock enabled.

Status bits affected: None.

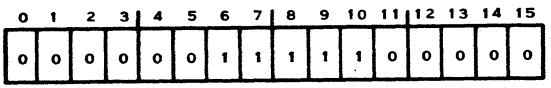
TMS9900 Microprocessor: Provides a signal that a CKON instruction is identified, but performs no processing. User may implement hardware to perform desired processing when signal is present.

Model 990/10 Computer: When Privileged Mode bit (bit 7 of ST register) is set to 0, instruction executes normally. When Privileged Mode bit is set to 1, an error interrupt occurs when execution of a CKON instruction is attempted.

3.2.41 LOAD OR RESTART EXECUTION LREX

Op Code: 03E0

Format:



Execution Results: Performs a context switch using the transfer vector at location $FFFC_{16}$, and sets the interrupt mask to 0. The transfer vector and the subroutine to which control is transferred are often in Read Only Memory (ROM).

Status bits affected: Interrupt Mask

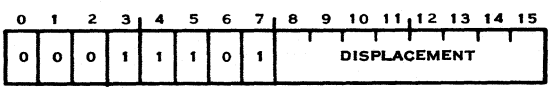
TMS9900 Microprocessor: Provides a signal that an LREX instruction is identified, but performs no processing. User may implement hardware to perform desired processing when signal is present.

Model 990/10 Computer: When the Privileged Mode bit (Bit 7) is set to 0 prior to execution of an LREX instruction, the instruction executes normally. When the Privileged Mode bit is set to 1 and execution of an LREX instruction is attempted, an error interrupt occurs instead. When the map option is included, the LREX instruction also sets the Map File bit (bit 8) to 0.

3.2.42 SET BIT TO LOGIC ONE SBO

Op Code: 1D00

Format:



Execution results: CRU bit addressed by the sum of the contents of workspace register 12 + displacement is set to 1.

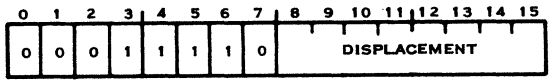
Status bits affected: None.

Model 990/10 Computer: When the Privileged Mode bit (bit 7) is set to 0, the SBO instruction executes normally. When the Privileged Mode bit is set to 1, and the effective CRU address is equal to or greater than $E00_{16}$, an error interrupt occurs.

3.2.43 SET BIT TO LOGIC ZERO SBZ

Op Code: 1E00

Format



Execution results: CRU bit addressed by the sum of the contents of workspace register 12 + displacement is set to 0.

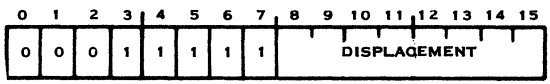
Status bits affected: None.

Model 990/10 Computer: When the Privileged Mode bit (bit 7) of the ST register is set to 0, the SBZ instruction executes normally. When bit 7 is set to 1 and the effective CRU address is equal to or greater than $E00_{16}$, an error interrupt occurs.

3.2.44 TEST BIT TB

Op Code: 1F00

Format:



Execution results: Equal bit is set to the value of the CRU bit addressed by the sum of the contents of workspace register 12 + displacement.

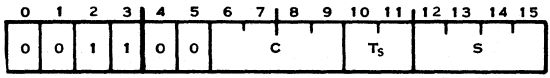
Status bit affected: Equal.

Model 990/10 Computer: When the Privileged Mode bit (bit 7) of the ST register is set to 0, the TB instruction executes normally. When bit 7 is set to 1 and the effective CRU address is equal to or greater than $E00_{16}$, an error interrupt occurs.

3.2.45 LOAD CRU LDCR

Op Code: 3000

Format:



Execution results: Number of bits specified by C are transferred from memory at address ga_S to consecutive CRU lines beginning at the address in workspace register 12.

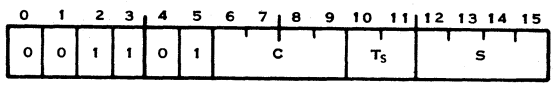
Status bits affected: Logical greater than, arithmetic greater than, and equal. When C is less than 9, odd parity is also set or reset.

Model 990/10 Computer: When the Privileged Mode bit (bit 7) of the ST register is set to 0, the LDCR instruction executes normally. When bit 7 is set to 1 and the effective CRU address is equal to or greater than $E00_{16}$, an error interrupt occurs.

3.2.46 STORE CRU STCR

Op Code: 3400

Format:



Execution results: Number of bits specified by C are transferred from consecutive CRU lines beginning at the address in workspace register 12 to memory at address ga_S.

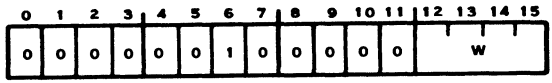
Status bits affected: Logical greater than, arithmetic greater than, and equal. When C is less than 9, odd parity is also set or reset.

Model 990/10 Computer: When the Privileged Mode bit (bit 7) of the ST register is set to 0, the STCR instruction executes normally. When bit 7 is set to 1 and the effective CRU address is equal to or greater than $E00_{16}$, an error interrupt occurs.

3.2.47 LOAD IMMEDIATE LI

Op Code: 0200

Format:



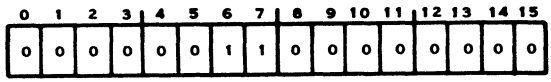
Execution results: iop → (wa)

Status bits affected: Logical greater than, arithmetic greater than, and equal.

3.2.48 LOAD INTERRUPT MASK IMMEDIATE LIMI

Op Code: 0300

Format:



Execution results: Places the four least significant bits of iop into the interrupt mask, the four least significant bits of the ST register.

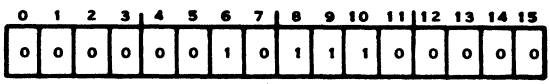
Status bits affected: Interrupt Mask

Model 990/10 Computer: When Privileged Mode bit (bit 7 of ST register) is set to 0, instruction executes normally. When Privileged Mode bit is set to 1, an error interrupt occurs when execution of an LIMI instruction is attempted.

3.2.49 LOAD WORKSPACE POINTER IMMEDIATE LWPI

Op Code: 02E0

Format:



Execution results: iop → (WP)

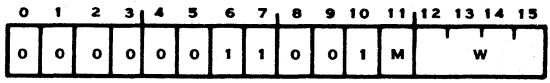
Status bits affected: None.

3.2.50 LOAD MEMORY MAP FILE LMF

This instruction is only available on the Model 990/10 Computer with map option.

Op Code: 0320

Format:



Execution results: When Privileged Mode bit (bit 7 of ST register) is set to 0: The contents of a six-word area at address wa are placed in map file M.

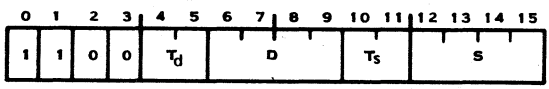
When Privileged Mode bit is set to 1: Error interrupt.

Status bits affected: None.

3.2.51 MOVE WORD MOV

Op Code: C000

Format:



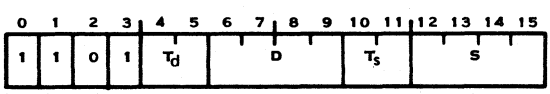
Execution results: $(ga_s) \rightarrow (ga_d)$

Status bits affected: Logical greater than, arithmetic greater than, and equal.

3.2.52 MOVE BYTE MOVB

Op Code: D000

Format



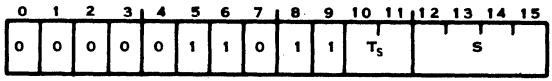
Execution results: $(ga_s) \rightarrow (ga_d)$

Status bits affected: Logical greater than, arithmetic greater than, equal, and odd parity.

3.2.53 SWAP BYTES — SWPB

Op Code: 06C0

Format:



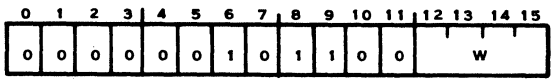
Execution results: Exchanges left and right bytes of word (ga_S).

Status bits affected: None.

3.2.54 STORE STATUS — STST

Op Code: 02C0

Format:



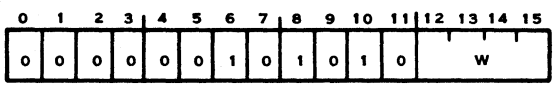
Execution results: (ST) → (wa)

Status bits affected: None.

3.2.55 STORE WORKSPACE POINTER — STWP

Op Code: 02A0

Format:



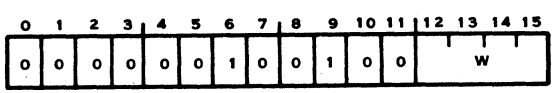
Execution results: (WP) → (wa)

Status bits affected: None.

3.2.56 AND IMMEDIATE — ANDI

Op Code: 0240

Format:



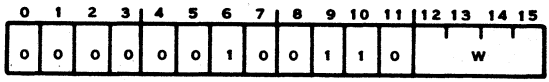
Execution results: (wa) AND iop → (wa)

Status bits affected: Logical greater than, arithmetic greater than, and equal.

3.2.57 OR IMMEDIATE ORI

Op Code: 0260

Format:



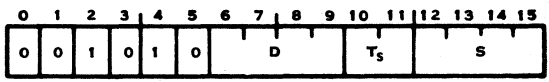
Execution results: (wa) OR iop → (wa)

Status bits affected: Logical greater than, arithmetic greater than, and equal.

3.2.58 EXCLUSIVE OR XOR

Op Code: 2800

Format:



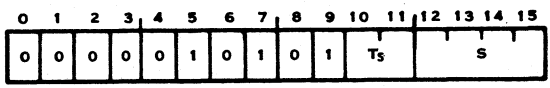
Execution results: (ga_S) XOR (wa_d) → (wa_d)

Status bits affected: Logical greater than, arithmetic greater than, and equal.

3.2.59 INVERT INV

Op Code: 0540

Format:



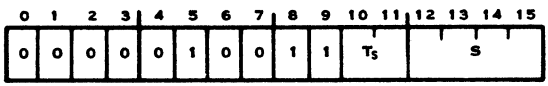
Execution results: The one's complement of (ga_S) is placed in (ga_S).

Status bits affected: Logical greater than, arithmetic greater than, and equal.

3.2.60 CLEAR CLR

Op Code: 04C0

Format:



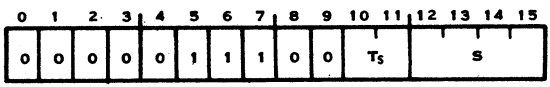
Execution results: $0 \rightarrow (ga_S)$

Status bits affected: None.

3.2.61 SET TO ONE SETO

Op Code: 0700

Format:



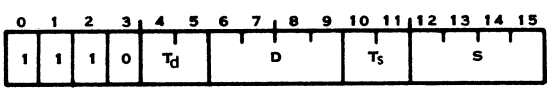
Execution results: $FFFF_{16} \rightarrow (ga_S)$

Status bits affected: None.

3.2.62 SET ONES CORRESPONDING SOC

Op Code: E000

Format:



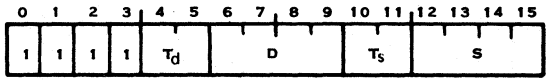
Execution results: Bits of (ga_d) corresponding to bits of (ga_S) equal to 1 are set to 1.

Status bits affected: Logical greater than, arithmetic greater than, and equal.

3.2.63 SET ONES CORRESPONDING, BYTE SOCB

Op Code: F000

Format:



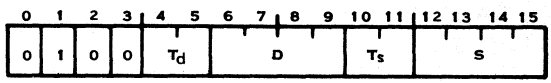
Execution results: Bits of (ga_d) corresponding to bits of (ga_s) equal to 1 are set to 1.

Status bits affected: Logical greater than, arithmetic greater than, equal, and odd parity.

3.2.64 SET ZEROS CORRESPONDING SZC

Op Code: 4000

Format:



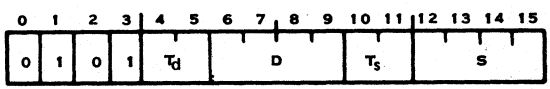
Execution results: Bits of (ga_d) corresponding to bits of (ga_s) equal to 1 are set to 0.

Status bits affected: Logical greater than, arithmetic greater than, and equal.

3.2.65 SET ZEROS CORRESPONDING, BYTE SZCB

Op Code: 5000

Format:



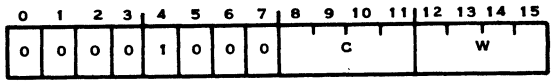
Execution results: Bits of (ga_d) corresponding to bits of (ga_s) equal to 1 are set to 0.

Status bits affected: Logical greater than, arithmetic greater than, equal, and odd parity.

3.2.66 SHIFT RIGHT ARITHMETIC SRA

Op Code: 0800

Format:



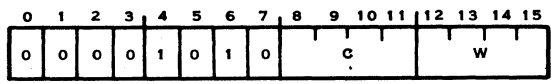
Execution results: Shift the bits of (wa) to the right, extending the sign bit to the right to fill vacated bit positions. When the contents of C is greater than 0, shift the number of bit positions contained in C. Otherwise shift the number of bit positions contained in the four least significant bits of workspace register 0. When C and the four least significant bits of workspace register 0 contain 0, shift 16 bit positions.

Status bits affected: Logical greater than, arithmetic greater than, equal, and carry.

3.2.67 SHIFT LEFT ARITHMETIC SLA

Op Code: 0A00

Format:



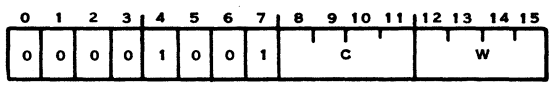
Execution results: Shift the bits of (wa) to the left, filling the vacated bit positions with zeros. The number of bit positions is determined by the value of C or the contents of workspace register 0 as described in paragraph 3.2.66.

Status bits affected: Logical greater than, arithmetic greater than, equal, carry, and overflow.

3.2.68 SHIFT RIGHT LOGICAL SRL

Op Code: 0900

Format:



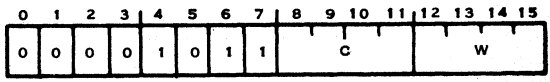
Execution results: Shift the bits of (wa) to the right, filling the vacated bit positions with zeros. The number of bit positions is determined by the value of C or the contents of workspace register 0 as described in paragraph 3.2.66.

Status bits affected: Logical greater than, arithmetic greater than, equal, and carry.

3.2.69 SHIFT RIGHT CIRCULAR SRC

Op Code: 0B00

Format:



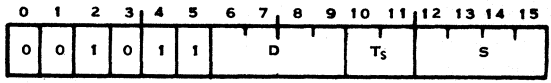
Execution results: Shift the bits of (wa) to the right, filling the vacated bit positions with the bits shifted out at the right. The number of bit positions shifted is determined by the value of C or the contents of workspace register 0 as described in paragraph 3.2.66.

Status bits affected: Logical greater than, arithmetic greater than, equal, and carry.

3.2.70 EXTENDED OPERATION XOP

Op Code: 2C00

Format:



Execution results: $ga_S \rightarrow$ (workspace register 11)
$(0040_{16} + (D *4)) \rightarrow (WP)$
$(0042_{16} + (D *4)) \rightarrow (PC)$

(WP) → (workspace register 13)
(PC) → (workspace register 14)
(ST) → (workspace register 15)

Status bits affected: Extended operation.

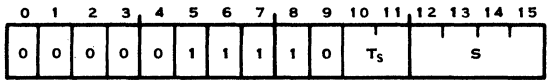
Model 990/10 Computer: An extended operation may be alternatively implemented by user-supplied hardware. When hardware is connected for the specified operation no context switch occurs, and the hardware performs the operation. When a Model 990/10 Computer performs a software-implemented extended operation, the Privileged Mode bit is set to 0. When the map option is included, the Map File bit is also set to 0.

3.2.71 LONG DISTANCE SOURCE LDS

This instruction is only available on the Model 990/10 Computer with map option.

Op Code: 0780

Format:



Execution results: When Privileged Mode bit (bit 7 of ST register) is set to 0: The contents of a six-word area at address ga_S are placed in map file 2, and the source address of the following instruction is mapped with map file 2. (If T_S of following instruction is equal to 0, or if following instruction is a B, BL, or BLWP instruction, new map is not used).

When Privileged Mode bit is set to 1: Error interrupt

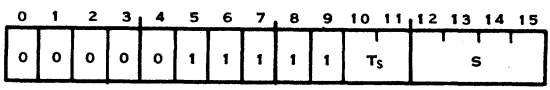
Status bits affected: None.

3.2.72 LONG DISTANCE DESTINATION LDD

This instruction is only available on the Model 990/10 Computer with map option.

Op Code: 07C0

Format:



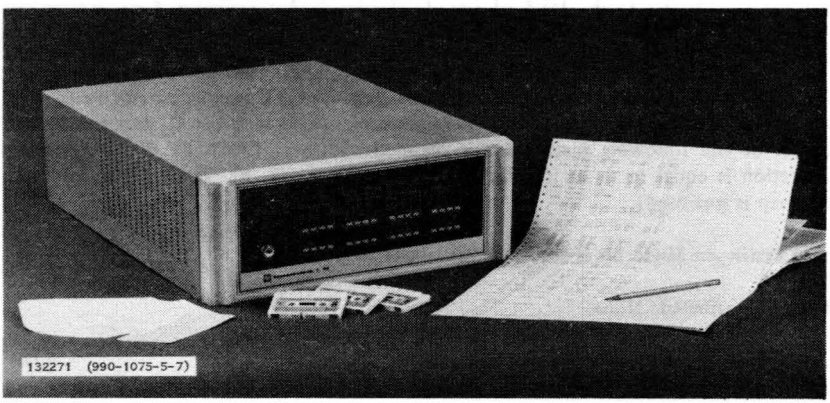
Execution results: When Privileged Mode bit (bit 7 of ST register) is set to 0: The contents of a six-word area at address ga_S are placed in map file 2, and the destination address of the following instruction is mapped with map file 2. (If T_d of the following instruction is equal to 0, or if the destination address is a workspace register address, the new map is not used.)

When Privileged Mode bit is set to 1: Error interrupt.

Status bits affected: None.

3.3 ASSEMBLY LANGUAGE CODING

The assembly language provides a means of writing a program without having to be concerned with actual memory addresses or machine instruction formats. The following paragraphs are an introduction to assembly language coding. Refer to the *Model 990 Computer Assembly Language Programmer's Guide*, for complete details of the language.



3.3.1 SYMBOLIC ADDRESSES. One of the big difficulties of programming in machine language is keeping track of memory addresses. The assembly language solves this problem by using symbolic addresses. The programmer assigns symbols to the locations and instructions as labels and refers to these locations by symbol rather than by address.

The assembler maintains a location counter to provide addresses for the symbols in the label field (labels) as they are read in the source program. As the assembler reads source program statements, it advances the location counter by one for each byte of object code resulting from each source statement. When there is a symbol in the label field, the assembler places the symbol and the corresponding location counter value in a symbol table. When the complete program has been read, the assembler has assigned

a machine address to each symbolic address (label) of the program. The assembler assembles object code using the address corresponding to the label in the entry in the symbol table.

A one-pass assembler reads the source statements of the program one time only, and builds the symbol table as it reads the statements. For this reason, the one-pass assembler requires that many of the labels used as operands be defined prior to their use as operands. Two-pass and multi-pass assemblers read the source statements two or more times, and build the complete symbol table before beginning to substitute machine addresses in a subsequent pass. These assemblers require fewer of the labels to be defined prior to their use as operands. The Model 990 Computer Assembly Language is used with a one-pass assembler (PX9ASM), a two-pass assembler (Cross Assembler), and a multi-pass assembler (SDSMAC).

3.3.2 SYMBOLIC OPERATION CODES. The assembly language includes a mnemonic operation code for each machine instruction of the instruction set. During the appropriate pass of the assembler, it substitutes the machine code for the mnemonic operation code as it assembles the instruction. Refer to table 3-3 for a summary of mnemonics.

The assembly language allows a label to be assigned to a machine instruction, and operands to be specified, according to machine instruction syntax, as follows:

[<label>]b...opcodeb...[<operand>][,<operand>]b...[<comment>]

This syntax definition means that a machine instruction may have a label, which is defined by the user. One or more blanks separate the label from the opcode. When the label is omitted, at least one blank must precede the opcode. The opcode, a generic name, may be any one of the set of mnemonic operation codes defined for the machine instructions. The opcode may also be an operation code defined in an assembler directive described in a subsequent paragraph. One or more blanks separate the opcode from an operand, if an operand is required. When a second operand is required, it is separated from the first operand by a comma. One or more blanks separate the operand or operands from the comments. Comments are optional, and have no effect on the assembly processing. They are printed in the listing to document the program.

Table 3-3. List of 990 Machine Instructions

MNEMONIC	OP CODE	INSTRUCTION
A	A000	ADD (WORD)
AB	B000	ADD (BYTE)
ABS	0740	ABSOLUTE VALUE
AI	0220	ADD IMMEDIATE
ANDI	0240	AND IMMEDIATE
B	0440	BRANCH
BL	0680	BRANCH AND LINK (W11)
BLWP	0400	BRANCH; LOAD WORKSPACE POINTER
C	8000	COMPARE (WORD)
CB	9000	COMPARE (BYTE)
CI	0280	COMPARE IMMEDIATE
CKOF	03C0	CLOCK OFF (NOTE 1,3)
CKON	03A0	CLOCK ON (NOTE 1,3)
CLR	04C0	CLEAR OPERAND
COC	2000	COMPARE ONES CORRESPONDING
CZC	2400	COMPARE ZEROES CORRESPONDING
DEC	0600	DECREMENT (BY ONE)
DECT	0640	DECREMENT (BY TWO)
DIV	3C00	DIVIDE
IDLE	0340	COMPUTER IDLE (NOTE 1,3)
INC	0580	INCREMENT (BY ONE)
INCT	05C0	INCREMENT (BY TWO)
INV	0540	INVERT
JEQ	1300	JUMP EQUAL (ST2=1)
JGT	1500	JUMP GREATER THAN (ST1=1)
JH	1B00	JUMP HIGH (ST0=1 AND ST2=0)
JHE	1400	JUMP HIGH OR EQUAL (ST0 OR ST2=1)
JL	1A00	JUMP LOW (ST0 AND ST2=0)
JLE	1200	JUMP LOW OR EQUAL (ST0 = 0 OR ST2 = 1)
JLT	1100	JUMP LESS THAN (ST1 AND ST2=0)
JMP	1000	JUMP UNCONDITIONAL
JNC	1700	JUMP NO CARRY (ST3 = 0)
JNE	1600	JUMP NOT EQUAL (ST2 = 0)
JNO	1900	JUMP NO OVERFLOW (ST4 = 0)
JOC	1800	JUMP ON CARRY (ST3 = 1)
JOP	1C00	JUMP ODD PARITY (ST5 = 1)
LDCR	3000	LOAD CRU
LDD	07C0	LONG DISTANCE DESTINATION (NOTE 1,2)
LDS	0780	LONG DISTANCE SOURCE (NOTE 1,2)
LI	0200	LOAD IMMEDIATE
LIMI	0300	LOAD INTERRUPT MASK IMMEDIATE (NOTE 1)
LMF	0320	LOAD MAP FILE (NOTE 1,2)
LREX	03E0	LOAD OR RESTART EXECUTION (NOTE 1,3)
LWPI	02E0	LOAD IMMEDIATE TO WORKSPACE POINTER
MOV	C000	MOVE (WORD)
MOVB	D000	MOVE (BYTE)
MPY	3800	MULTIPLY
NEG	0500	NEGATE (TWO'S COMPLEMENT)
ORI	0260	OR IMMEDIATE
RSET	0360	RESET AU (NOTE 1,3)
RTWP	0380	RETURN FROM INT. SUBR. (NOTE 1)
S	6000	SUBTRACT (WORD)
SB	7000	SUBTRACT (BYTE)
SBO	1D00	SET CRU BIT TO ONE
SBZ	1E00	SET CRU BIT TO ZERO
SETO	0700	SET ONES
SLA	0A00	SHIFT LEFT ARITHMETIC
SOC	E000	SET ONES CORRESPONDING (WORD)
SOCB	F000	SET ONES CORRESPONDING (BYTE)
SRA	0800	SHIFT RIGHT (MSB EXTENDED)
SRC	0B00	SHIFT RIGHT CIRCULAR
SRL	0900	SHIFT RIGHT LOGICAL
STCR	3400	STORE FROM CRU
STST	02C0	STORE STATUS REGISTER
STWP	02A0	STORE WORKSPACE POINTER
SWPB	06C0	SWAP BYTES
SZC	4000	SET ZEROES CORRESPONDING (WORD)
SZCB	5000	SET ZEROES CORRESPONDING (BYTE)
TB	1F00	TEST CRU BIT
X	0480	EXECUTE
XOP	2C00	EXTENDED OPERATION
XOR	2800	EXCLUSIVE OR

NOTES

1. PRIVILEGED INSTRUCTIONS – MODEL 990/10
2. MODEL 990/10 WITH MAP ONLY
3. NOT IMPLEMENTED IN TMS9900

The syntax definition just described illustrates the following conventions used in the syntax definitions for assembler directives in succeeding paragraphs:

- Items in capital letters, and special characters, must be entered as shown.
- Items within angle brackets (< >) are defined by the user.
- Items in lower case letters are classes (generic names) of items.
- Items within brackets ([]) are optional.
- Items within braces ({}) are alternative items; one must be entered.
- An ellipsis (...) indicates that the preceding item may be repeated.
- The symbol b represents a blank or space.

3.4 ASSEMBLER DIRECTIVES

The assembly language includes assembler directives that supplement the machine instructions to form a source program. The assembler directives control the assembler, and define data for in the program. Subsequent paragraphs describe the five categories of directives, and additional paragraphs describe individual directives and the syntax for the directives.

3.4.1 INITIALIZING OR MODIFYING LOCATION COUNTER CONTENTS. The directives in this category are:

Absolute Origin	Block Starting With Symbol
Relocatable Origin	Block Ending With Symbol
Dummy Origin	Work Boundary.

The assembly language includes three directives that initialize the location counter, the Absolute Origin directive, the Relocatable Origin directive, and the Dummy Origin directive. If neither of the three directives is included before the first statement that results in object code, the assembler initializes the location counter to zero and assembles object code with relocatable addresses. When the assembler reads an Absolute Origin directive, it initializes the location counter with the value of the operand and assembles object code with absolute addresses. When the assembler reads a Relocatable Origin directive, the assembler initializes the location counter and assembles object code with absolute addresses. When the assembler reads a Dummy Origin directive, it initializes the location counter to the value of the operand, and processes source statements without producing object code.

The other three directives in this category modify the contents of the location counter. The Block Starting With Symbol statement reserves a specified area of memory and optionally assigns a label to the first location of the area. The Block Ending With Symbol statement also reserves an area of memory and optionally assigns a label to the location that follows the last location of the area. The Even directive forces word alignment by incrementing the location counter by one if it contains an odd address. It should be used wherever word alignment is required.

3.4.2 DEFINING THE ASSEMBLER OUTPUT. The directives in this category are:

Program Identifier	List Source
Page Title	No Source List
Output Options	Page Eject.

The Program Identifier directive assigns an identifier to the program and places the identifier in the object code. The Page Title directive specifies a title to be printed at the top of each page of the listing. The Output Options directive specifies output options for the assembly. Different options are available with each assembler. Refer to the *Model 990 Computer/TMS9900 Microprocessor Assembly Language Programmer's Guide* for the options of each assembler. The List Source directive relates to the No Source List directive. These directives allow the user to specify areas of his program to be omitted in the listing. The No Source List directive disables printing of the listing, and the List Source directive restores printing of the listing. The Page Eject directive allows the user to force a new page in the listing, which aids documentation of the program.

3.4.3 INITIALIZING CONSTANTS. The directives that initialize constants for the program are:

Initialize Byte	Initialize Text
Initialize Word	Define Assembly-Time Constant.

The Initialize Byte directive specifies values to be placed in one or more bytes of the object program. These may be decimal or hexadecimal values, or single characters. The assembler converts decimal and hexadecimal values to binary numbers, and characters to ASCII representation. The Initialize Word directive specifies values to be placed in one or more words of the object program, performing the same conversions. When characters are specified with the Initialize Word directive, the user must specify a pair of characters to be placed in a word. The Initialize Text directive specifies a string of characters to be placed in successive bytes of the object program. The characters are converted to ASCII representation. The Define Assembly-Time Constant directive allows the user to assign values to symbols to be used during assembly of the program.

3.4.4 DEFINING PROGRAM LINKAGE. The directives that define linkage between modules of a program are:

External Definition External Reference.

These directives allow a program to be assembled as a number of modules and linked to form an integrated program. Linking requires that any symbols in one module that are referenced in one or more other modules be identified. Linking also requires that any symbols referenced in a module that appear in another module be identified. The External Definition directive defines one or more symbols that appear as labels in the program module and are referenced in other modules of the program. The External Reference directive defines one or more symbols that are used as operands in the program module but appear as labels in other modules of the program.

3.4.5 ADDITIONAL DIRECTIVES. The following two miscellaneous directives are available in all assemblers:

Define Extended Operation Program End.

The Define Extended Operation directive allows the user to assign a symbolic operator code to an extended operation. Following entry of this directive, the symbol defined by the directive may be used as an opcode in a source statement. The Program End directive is required in every source program to define the end of the program.

The Macro Assembler of the Software Development System (SDSMAC) supports the following additional directives:

Define Operation Copy Source File

Workspace Pointer.

The Define Operation directive allows the user to define a synonym for an opcode. The synonym may then be used as an opcode in a source statement. The Workspace Pointer directive defines the current workspace to the assembler, allowing SDSMAC to recognize workspace register addresses expressed as symbolic memory addresses. The Copy Source File directive copies source statements from source files, incorporating the copied statements into the current source program.

3.4.6 ABSOLUTE ORIGIN. The absolute origin directive defines location counter contents as absolute, and initializes the location counter with the specified value. The syntax of the AORG directive is defined as follows:

[<label>]b...AORGb...<wd exp>b...[<comment>]

The operand (wd exp) must be a well-defined expression, which is an expression that has an absolute value, any symbol of which is previously defined. The following is an example of an AORG directive:

AORG	>40	Initializes the location counter to absolute address 0040_{16}.

3.4.7 RELOCATABLE ORIGIN. The relocatable origin directive defines location counter contents as relocatable, and initializes the location counter with the specified value. When the operand is omitted, the directive initializes the location counter to the value following the previous relocatable code of the program, or to zero. The syntax of the RORG directive is defined as follows:

[<label>]b... RORGb ... [<exp>]b ... [<comment>]

The following are examples of RORG directives:

START	RORG	256	Initializes the location counter to relocatable address 0100_{16} and assigns that value to label START.
	RORG		If there is no previous RORG directive in the program, initializes the location counter to relocatable address zero. If a previous RORG directive had resulted in a block of relocatable code ending at location $00FF_{16}$, initializes the location counter to relocatable address 0100_{16}.

3.4.8 DUMMY ORIGIN. The dummy origin directive defines location counter contents as a dummy value, and initializes the location counter with the specified value. The syntax of the DORG directive is defined as follows:

[<label>]b...DORGb...<exp>b...[<comment>]

The following is an example of a DORG directive:

TDATA	DORG	0	Initializes the location counter to provide a dummy section in which addresses are relative to zero.

3.4.9 BLOCK STARTING WITH SYMBOL. The block starting with symbol directive advances the location counter a specified number of bytes, and optionally assigns a label to the first location of the block. The syntax of the BSS directive is defined as follows:

[<label>]b...BSSb...<wd exp>b...[<comment>]

The following is an example of a BSS directive:

BUFF	BSS	80	Reserves an 80-byte area at location BUFF.

3.4.10 BLOCK ENDING WITH SYMBOL. The block ending with symbol directive advances the location counter a specified number of bytes, and optionally assigns a symbol to the location following the block. The syntax of the BES directive is defined as follows:

[<label>]b...BESb...<wd exp>b...[<comment>]

The following is an example of a BES directive:

	BES	24	Reserves a 24-byte area of memory.

3.4.11 WORD BOUNDARY. The word boundary directive advances the location counter to the next word boundary (even) address when the location counter contains an odd address. The syntax of the EVEN directive is as follows:

[<label>]b...EVENb...[<comment>]

The following is an example of an EVEN directive:

MSG	EVEN	If the location counter contains an even address, leaves the location counter unaltered. Otherwise, adds one to the location counter value, and places the sum in the location counter. Assigns the previous value to label MSG.

3.4.12 PROGRAM IDENTIFIER. The program identifier directive places the specified identifier in the object output. The syntax of the IDT directive is defined as follows:

[<label>]b...IDTb...<string>b...[<comment>]

The operand is a string of up to eight characters enclosed in single quotes (' '). The following is an example of an IDT directive:

```
IDT     'PROG1'     Assigns identifier PROG1 to the
                    program, and places the identifier
                    in the object program.
```

3.4.13 PAGE TITLE. The page title directive specifies a title to be placed in the heading of each page of the listing. The directive must be the first source statement if it is to be printed on the first page. The syntax of the TITL directive is defined as follows:

[<label>]b...TITLb...<string>b...[<comment>]

The operand consists of a string of up to 50 characters. The following is an example of a TITL directive:

```
TITL    'MAIN PROGRAM'    Specifies the page title
                          MAIN PROGRAM to be printed on
                          the pages of the listing.
```

3.4.14 OUTPUT OPTIONS. The output options directive specifies options for the assembly. This directive does not apply to the one-pass assembler, PX9ASM. The Cross Assembler and the Macro Assembler, SDSMAC, include different options. Refer to the *Model 990 Computer Assembly Language Programmer's Guide* for the keywords and options applicable to the assembler being used. The syntax of the OPTION directive is defined as follows:

b...OPTIONb...<keyword>[,<keyword>]...b...[<comment>]

The following is an example of an OPTION directive:

```
OPTION    XREF          Specifies the printing of a cross
                        reference listing of the assembly.
```

3.4.15 LIST SOURCE. The list source directive restores printing of the source listing. The syntax of the LIST directive is defined as follows:

[<label>]b...LISTb...[<comment>]

The following is an example of a LIST directive:

```
LIST          Causes listing of source statements
              to resume.
```

3.4.16 NO SOURCE LIST. The no source list directive inhibits printing of the source listing. The syntax of the UNL directive is defined as follows:

[<label>]b...UNLb...[<comment>]

The following is an example of an UNL directive:

UNL	Causes listing of source statements to be suspended.

3.4.17 PAGE EJECT. The page eject directive causes the assembler to continue printing the source listing at the top of the next page. The syntax of the PAGE directive is defined as follows:

[<label>]b...PAGEb...[<comment>]

The following is an example of a PAGE directive:

PAGE	Causes the printer on which the source listing is being printed to skip to the top of the next page.

3.4.18 INITIALIZE BYTE. The initialize byte directive places one or more values in one or more successive bytes of memory. The values may be expressed as decimal or hexadecimal numbers, labels, or as single ASCII characters. The syntax of the BYTE directive is defined as follows:

[<label>]b...BYTEb...<exp>[,<exp>]...b...[<comment>]

The following is an example of a BYTE directive:

KON	BYTE	12,>F,'A'	Initializes a three-byte area at location KON. The bytes contain $0C_{16}$, $0F_{16}$, and 41_{16}, respectively.

3.4.19 INITIALIZE WORD. The initialize word directive places one or more values in one or more successive words of memory. The values may be expressed as decimal or hexadecimal numbers, labels, or as pairs of ASCII characters. The syntax of the DATA directive is defined as follows:

[<label>]b...DATAb...<exp>[,<exp>]...b...[<comment>]

The following is an example of a DATA directive:

D1	DATA	BUFF,>08FE,'AB'	Initializes a three-word area at location D1. The words contain the value of label BUFF, $08FE_{16}$, and 4142_{16}, respectively.

3.4.20 INITIALIZE TEXT. The initialize text directive places the ASCII representations of a string of characters (up to 52) in successive bytes of memory. When a minus sign (–) is placed before the string, the ASCII character representation of the last character is negated (two's complement). The syntax of the TEXT directive is defined as follows:

[<label>]b...TEXTb...[–]<string>b...[<comment>]

The following is an example of a TEXT directive:

MSG1	TEXT	'ENTER NAME'	Places ten ASCII characters in ten successive bytes, as follows: 45_{16}, $4E_{16}$, 54_{16}, 45_{16}, 52_{16}, 20_{16}, $4E_{16}$, $4A_{16}$, $4D_{16}$, 45_{16}.

3.4.21 DEFINE ASSEMBLY-TIME CONSTANT. The define assembly-time constant directive defines a constant to use during assembly. The syntax of the EQU directive is defined as follows:

<label>b...EQUb...<exp>b...[<comment>]

The following is an example of an EQU directive:

TWO	EQU	2	Assigns the value 2 to the label TWO.

3.4.22 EXTERNAL DEFINITION. The external definition directive makes one or more symbols in a program module available for use in other modules. The syntax for the DEF directive is defined as follows:

[<label>]b...DEFb...<symbol>[,<symbol>]...b...[<comment>]

The following is an example of a DEF directive:

	DEF	BUFF,MSG1	Causes the assembler to make the values of labels BUFF and MSG1 available for linking in other modules.

3.4.23 EXTERNAL REFERENCE. The external reference directive accesses one or more symbols in other program modules. The syntax for the REF directive is defined as follows:

[<label>]b...REFb...<symbol>[,<symbol>]...b...[<comment>]

The following is an example of an REF directive:

REF	DAT1,BUFF2	Causes the assembler to make labels DAT1 and BUFF2 available for linking to other modules.

3.4.24 WORKSPACE POINTER. The workspace pointer directive is available with the SDSMAC assembler only. The workspace pointer directive defines the current workspace to the assembler. The syntax for the WPNT directive is defined as follows:

[<label>]b...WPNTb...<label>b...[<comment>]

The following is an example of a WPNT directive:

WPNT	WS1	Defines the workspace at location WS1 to the assembler as the current workspace.

3.4.25 COPY SOURCE FILE. The copy source file directive is available with the SDSMAC assembler only. The copy source file directive causes the assembler to obtain source statements form the specified file. The syntax for the COPY directive is defined as follows:

[<label>]b...COPYb...<file name>b...[<comment>]

The following is an example of a COPY directive:

COPY	FILE1	Includes the contents of FILE1 in the current source program at this point.

3.4.26 DEFINE OPERATION. The define operation directive is available with the SDSMAC assembler only. The directive defines a synonym for an operation. The syntax for the DFOP directive is defined as follows:

[<label>]b...DFOPb...<symbol>,<operation>b...[<comment>]

The following is an example of a DFOP directive:

ADD	DFOP	ADD,A	Defines ADD as a synonym for operation code A.

3.4.27 DEFINE EXTENDED OPERATION. The define extended operation directive assigns a symbol for an extended operation. The syntax for the DXOP directive is defined as follows:

[<label>]b...DXOPb...<symbol>,<term>b...[<comment>]

The term is a number, 0 through 15, that specifies an extended operation. The symbol is the symbol to be used as an operation code for the extended operation. The following is an example of the DXOP directive:

DXOP	FADD,7	Assigns the symbol FADD as an opcode for extended operation 7.

3.4.28 PROGRAM END. The program end directive terminates a program module or program. The syntax for the END directive is as follows:

[<label>]b...ENDb...[<symbol>]b...[<comment>]

Including the optional symbol operand causes the assembler to place the value of the symbol in the object code as the entry point. The following is an example of the END directive:

END	START	Terminates the program module and defines START as the entry point.

3.5 PSEUDO-INSTRUCTIONS

The assemblers for the Model 990 Computers support two pseudo-instructions, each of which assembles an instruction with a specific operand. In addition, the SDSMAC assembler supports an additional pseudo-instruction that assembles three assembler directives.

The no-operation pseudo-instruction is used to provide an instruction that has no effect on the execution of the program. It has the effect of reserving a word location in the executable portion of the program. The return pseudo-instruction is used to return from a subroutine when a link to the calling program is stored in workspace register 11. The BL instruction stores a link in workspace register 11 when it transfers control to a subroutine. The transfer vector pseudo-instruction (SDSMAC only) provides a transfer vector for a subroutine to which control is passed by a context switch with a BLWP instruction. The pseudo-instruction also defines the new workspace to the assembler.

3.5.1 NO OPERATION. The no operation pseudo-instruction has no effect on the execution of the program. The assembler substitutes a jump instruction to the next instruction in sequence. The syntax for the NOP pseudo-instruction is defined as follows:

[<label>]b...NOPb...[<comment>]

3.5.2 RETURN. The return pseudo-instruction returns control from a common workspace subroutine entered by a BL instruction. The assembler substitutes a branch instruction to the address stored in workspace register 11. The syntax for the RT pseudo-instruction is defined as follows:

[<label>]b...RTb...[<comment>]

3.5.3 TRANSFER VECTOR. The transfer vector pseudo-instruction provides a transfer vector consisting of an address to be placed in the WP register and another address to be placed in the PC. The assembler substitutes two DATA directives to build the vector, and a WPNT directive to define the workspace to the assembler. The syntax for the XVEC pseudo instruction is defined as follows:

<label>b...XVECb...<wp address>[,<subr address>]b...[<comment>]

The first operand is the workspace address, to be placed in the WP register. The optional second operand is the subroutine address, to be placed in the PC. When the second operand is omitted, the assembler assumes that the entry point of the subroutine is the instruction that follows the XVEC directive. The following is an example of an XVEC directive:

```
CONV    XVEC    WS2,SUB2        Provides a transfer vector at
                                location CONV consisting of
                                address WS2 and address SBU2. Also
                                defines WS2 to the assembler as the
                                current workspace. Equivalent to:

                                CONV    DATA    WS2
                                        DATA    SUB2
                                        WPNT    WS2
```

3.6 MEMORY ADDRESSING

The following paragraphs provide some guidelines for addressing the memory of the Model 990 Computer. The following five modes are addressing options for one or both operands of many instructions:

- Workspace register addressing
- Indirect workspace register addressing

- Symbolic memory addressing
- Indexed memory addressing
- Indirect autoincrement workspace register addressing.

Other instructions use one of the following specific types of addresses:

- Jump addresses
- CRU addresses
- Immediate addresses.

3.6.1 CODING OF WORKSPACE REGISTER ADDRESSES. The workspace register address is one of five addressing options for many instructions, and is the specific address type for other instructions.

The workspace concept is an important feature of the Model 990 Computer. A workspace is a set of 16 consecutive memory words that may be addressed as workspace registers 0 through 15. Beginning with the initial context switch that occurs as power is applied, a value is placed in the WP register during each context switch. This value is the address of the first word of the workspace (workspace register 0), and may be any address in memory. It should be greater than 80_{16}, and less than or equal to the highest address in memory minus 31. Execution of a BLWP, LREX, XOP or RTWP instruction or processing of an interrupt causes a context switch. Execution of an LWPI instruction activates a new workspace without performing a context switch.

A workspace register address may be an integer from 0 through 15, or a symbol having a value in that range. The addressed workspace register contains the operand. The assemblers have preassigned symbols for workspace register addresses. These symbols are R0 through R15, for workspace registers 0 through 15, respectively. Figure 3-5 shows an instruction having two workspace register addresses, in memory location 300_{16}. The assembly language form of the instruction is shown, but actually memory would contain the machine language assembled for the instruction. The WP register contents, 100_{16}, is the address of workspace register 0. The PC contains 302_{16}, the address following the current instruction. The ST register contents are unimportant to the execution of the instruction. The current instruction is an Add instruction, that adds the contents of workspace register 5 to the contents of workspace register 6, and places the sum in workspace register 6. The contents of workspace registers 5 and 6 are shown prior to execution. After executing the instruction, workspace register 5 still contains 3, and workspace register 6 contains 10.

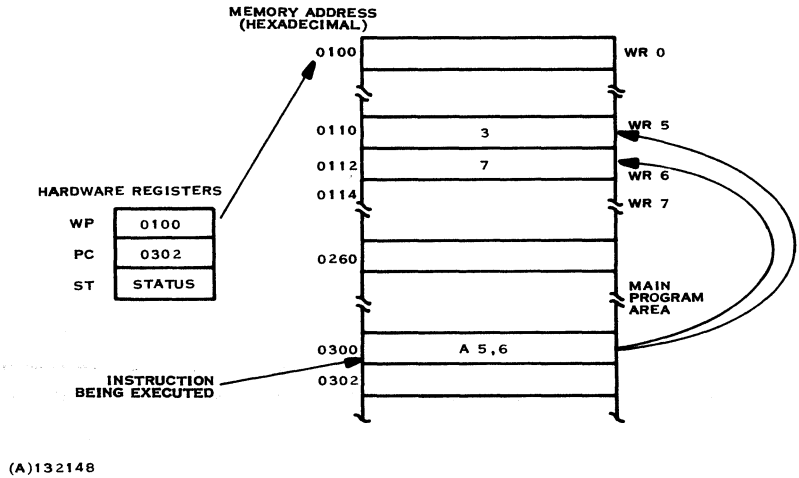


Figure 3-5. Workspace Register Addressing

The following are additional examples of coding of workspace register addresses:

	DECT	6	Decrement the contents of workspace register 6 by two.
	SB	R1,R5	Subtract the contents of the left byte of workspace register 1 from the contents of the left byte of workspace register 5, and place the remainder in the left byte of workspace register 5.
SUM	EQU	7	Assigns the value 7 to label SUM.
	CLR	SUM	Sets the contents of workspace register 7 to zero.

3.6.2 · CODING OF INDIRECT ADDRESSES. Another of the five addressing options is the indirect workspace register address. The workspace register is specified in the same manner as in a workspace register address, but the number or symbol is preceded by an asterisk (*). The workspace register contains the address of the operand.

Figure 3-6 shows an example of indirect addressing. The instruction in address 0304_{16} has a workspace register address for the source operand, and an indirect workspace

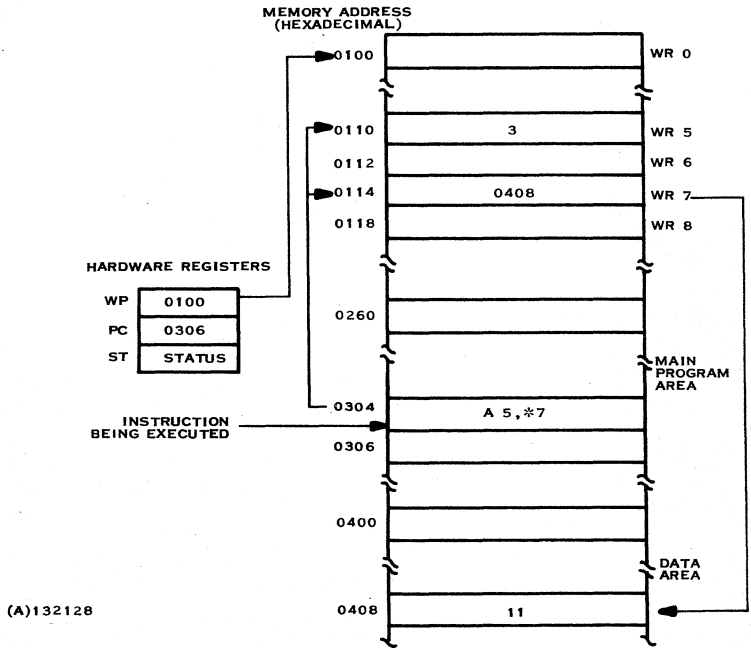


Figure 3-6. Indirect Workspace Register Addressing

register address for the destination operand. The WP register contents defines the same workspace as in the previous example, and the PC contains 0306_{16}, the address of the following instruction. The ST register contents are unimportant to the execution of the instruction. The current instruction is an Add instruction that adds the contents of workspace register 5 to the operand addressed by workspace register 7. Since workspace register 7 contains 0408_{16}, the operand is the contents of address 0408_{16}. The instruction places the sum in the destination operand, the contents of address 0408_{16}. Figure 3-6 shows the operands prior to execution of the instruction. After execution, workspace register 5 still contains 3, workspace register 7 still contains 0408_{16}, and address 0408_{16} now contains the sum, 14.

Indirect workspace register addressing may be used to access data at an address that has been placed in a workspace register. For example, the BL instruction places the address of the word following the instruction in workspace register 11. This word may contain data related to the BL instruction. The following source code could be used in a subroutine to access data in the word following the BL instruction that transfers control to the subroutine:

```
        BL      @SUB
        DATA    10
          .
          .
          .
SUB     MOV     *11,R2      Copy data in the address in workspace
                            register 11 into workspace register 2.
```

After the instructions that perform the processing of the subroutine, the subroutine may include the following instructions:

```
        INCT    11          Add two to the contents of work-
                            space register 11, to return to
                            the instruction following the data
                            word.

        B       *11         Return to the address in work-
                            space register 11.
```

Other examples of coding indirect workspace register addresses are as follows:

```
        A       *1,*5       Add the contents of the word at
                            the location in workspace register
                            1 to the contents of the word
                            at the location in workspace register
                            5, and place the sum in the location
                            in workspace register 5.

        SB      *R1,*R5     Subtract the byte at the address in
                            workspace register 1 from the byte
                            at the address in workspace
                            register 5, and place the remainder
                            in the byte at the address in
                            workspace register 5.
```

3.6.3 CODING OF SYMBOLIC ADDRESSES. The third of the five addressing options is the symbolic memory address. When this mode is used, the assembler supplies the memory address corresponding to the symbol in a word following the instruction. The address is coded as an at sign (@) followed by a symbol or expression that represents the address. Integer addresses may be used when the addresses are known; for example, when they are absolute addresses.

Figure 3-7 illustrates an instruction with a symbolic address. The instruction at address $030C_{16}$ has a workspace register address for the source operand and a symbolic memory address for the destination operand. The WP register defines 0100_{16} as the address of the workspace, and the PC contains 0310_{16}, the address of the next instruction. The ST register contents are unimportant to the execution of the instruction. The current instruction is an Add instruction that adds the contents of workspace register 5 to the contents of location PRIME, address 0400_{16}. Figure 3-7 shows the contents of the operands prior to execution of the instruction. After execution, workspace register 5 still contains 3, but address 0400_{16} contains the sum, 5.

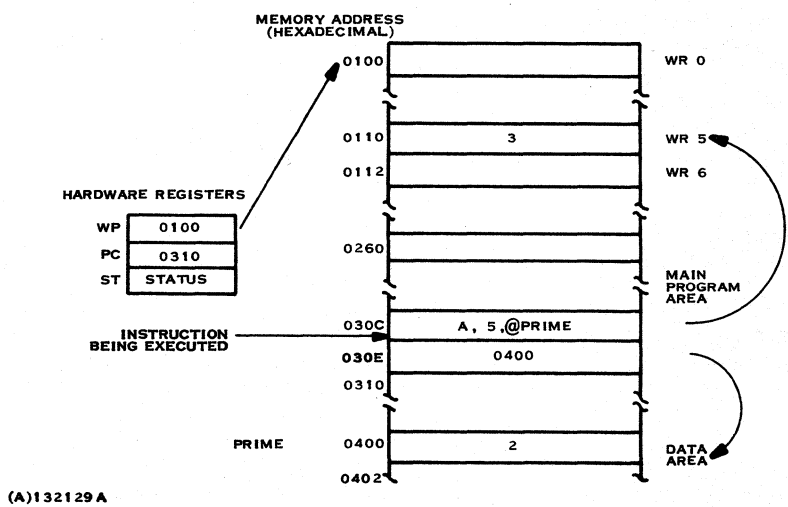


Figure 3-7. Symbolic Memory Addressing

The following are additional examples of symbolic memory address coding:

MOV	@IN,@OUT+2	Copy the contents of location IN into the word location following location OUT.
BL	@>42	Branch to absolute location 0042_{16}, and store a link in workspace register 11.
C	@>100,@WP1	Compare the contents of address 0100_{16} to the contents of location WP1.

Symbolic memory addresses for the SDSMAC Assembler may be coded without the at sign. SDSMAC maintains a record of the current workspace address, and translates a symbolic memory address into a workspace register address when the value of the symbol is neither less than the workspace address nor greater than the sum of the workspace address plus 30. Processing of symbolic addresses in this manner makes it unnecessary to require the use of the at sign(@). The following source statements would both be accepted by SDSMAC, and would result in identical object code:

ABS	@TOTAL	Places the absolute value of the contents of location TOTAL in location TOTAL.
ABS	TOTAL	Places the absolute value of the contents of location TOTAL in location TOTAL.

3.6.4 CODING OF INDEXED ADDRESSES. The fourth of the five addressing options is the indexed memory address. When this mode is used, the address includes a workspace register designated as an index register. The computer adds the contents of this register to the address at execution time to form the effective address for the instruction. An indexed memory address is coded as an at sign followed by a symbol, expression, or integer, and an index register enclosed in parentheses. The index register is specified as an integer in the range of 1 through 15, or as a symbol having a value in that range. In other words, an indexed memory address is a symbolic memory address followed by an index register specification.

Figure 3-8 shows an example of indexed memory addressing. The instruction at address 0310_{16} has a workspace register address for the source operand, and an indexed memory address for the destination operand. The WP register defines 0100_{16} as the current workspace, and the PC contains 0314_{16}, the address of the next instruction. The ST register contents are unimportant to the execution of the

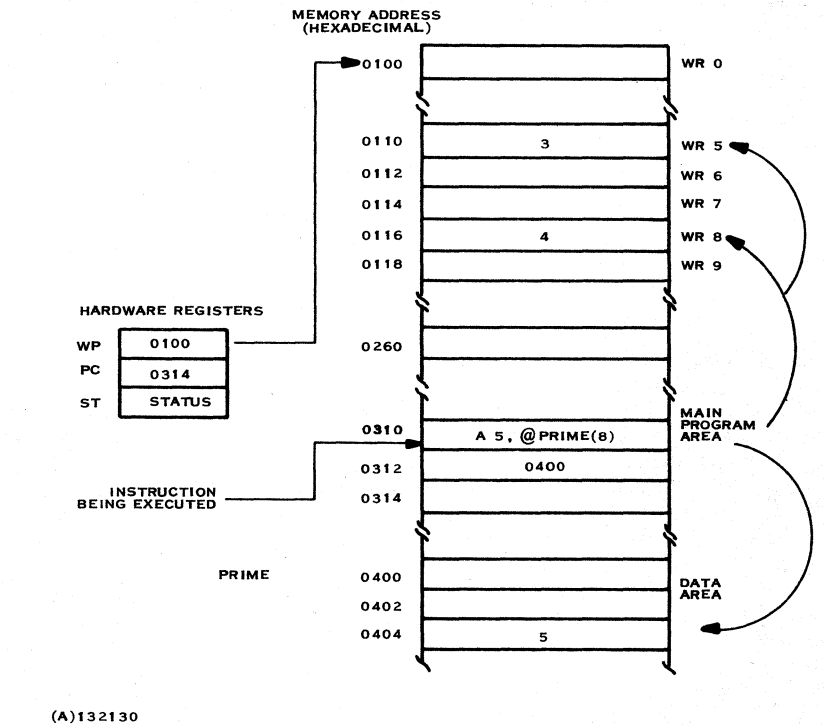


Figure 3-8. Indexed Memory Addressing

instruction. The current instruction is an Add instruction that adds the contents of workspace register 5 to the contents of an address that is the sum of the contents of workspace register 8 and the value of symbol PRIME. The value of symbol PRIME is 0400_{16}, and the contents of workspace register 8 is 4, so the destination operand is the contents of location 0404_{16}. The instruction places the sum in this address. Figure 3-8 shows the contents of the operands prior to the execution of the instruction. After execution, workspace register 5 still contains 3, workspace register 8 still contains 4, but address 0404_{16} now contains the sum, 8.

When the address of a buffer, table, or array is placed in a workspace register other than workspace register 0, that register may be used as an index register to access the

data. For example, if the address of a buffer were in workspace register 6, the following code would access specific bytes and words of the buffer:

MOV	@6(6),@VAL	Copy the contents of the fourth word of the buffer (buffer address + 6) into location VAL.
CB	@1(6),@KEY	Compare the contents of the second byte of the buffer (buffer address +1) with the contents of the byte at location KEY.
SZC	@MASK,@8(6)	Mask off the fifth word of the buffer (buffer address + 8) according to the one bits of location MASK.

3.6.5 CODING OF AUTOINCREMENT ADDRESSES. The fifth addressing option is the indirect workspace register autoincrement address. In this mode, the contents of the specified workspace register is the address of the operand, and are incremented after the operand is accessed. The increment is two for word instructions or one for byte instructions. An indirect workspace register autoincrement address is coded as an asterisk followed by an integer from 0 through 15, or a symbol or expression having a value in that range, and a plus sign (+). In other words, an indirect workspace register autoincrement address is an indirect workspace register address followed by a plus sign.

Figure 3-9 shows an example of autoincrement addressing. The instruction at address 0308_{16} has a workspace address for the source operand and an indirect autoincrement address for the destination operand. The WP register defines the current workspace at address 0100_{16}, and the PC contains the address of the following instruction, $030A_{16}$. The ST register contents are unimportant to the execution of the instruction. The current instruction is an Add instruction that adds the contents of workspace register 5 to the contents of the address in workspace register 7, places the sum in that address, and increments workspace register 7 by two. Figure 3-9 shows the contents of the operands prior to the execution of the instruction. After the instruction is executed, workspace register 5 still contains 3, workspace register 7 contains $040A_{16}$, and address 0408_{16} contains the sum, 14.

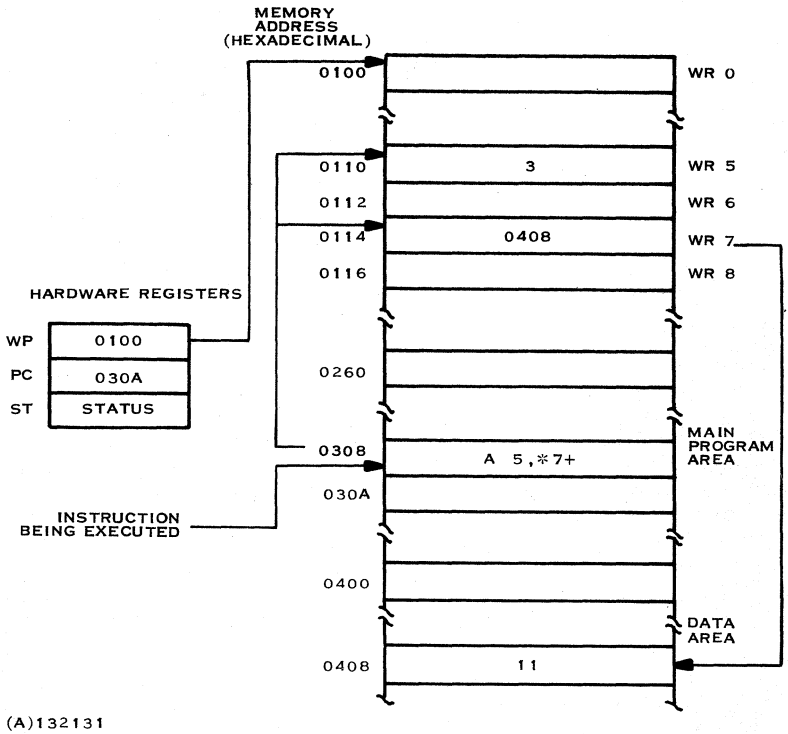


Figure 3-9. Indirect Workspace Register Autoincrement Addressing

This addressing mode is very useful for processing an array or table. For example, if an array of 12 numbers at location IN is to be added to the numbers in an array at location SUM, and the sums placed in the second array, the following instructions could be used:

LI	3,12	Place 12 in workspace register 3
LI	4,IN	Place the value assigned to label IN in workspace register 4.
LI	5,SUM	Place the value assigned to label SUM in workspace register 5.

```
LOOP    A       *4+,*5+     Add an element of array IN
                            to the corresponding element
                            of array SUM, and place the sum
                            in the element of array
                            SUM. Increment each address by
                            two.

        DEC     3           Subtract one from the
                            contents of workspace
                            register 3.

        JNE     LOOP        Branch to location LOOP
                            until the remainder in
                            workspace register 3 is
                            zero.
```

The Add instruction is executed a total of 12 times, once for each element of the arrays. Workspace register 4 contains the address of the word following array IN, and workspace register 5 contains the address of the word following array SUM, following execution of the code shown.

The following are additional examples of coding of indirect workspace register autoincrement addressing:

```
        MOV     *R11+,@ARG1 Copy the contents of
                            the address in workspace
                            register 11 into
                            location ARG1 and increment
                            the contents of workspace
                            register 11 by two.

        SB      3,*7+       Subtract the contents of
                            the most significant byte of
                            workspace register 3 from
                            the contents of the address
                            in workspace register 7,
                            and increment the contents
                            of workspace register 7
                            by one.

        CLR     *R3+        Clear the contents of the
                            address in workspace
                            register 3 to zero, and
                            increment the address in
                            workspace register 3 by
                            two.
```

3.6.6 CODING OF JUMP ADDRESSES. The jump instructions require symbolic addresses or expressions that are not preceded by at signs. The assembler computes the displacement required in the machine instruction by evaluating the address, subtracting the location counter value from the address value, and dividing the difference by two. This displacement must be in the range of −128 through +127. A symbolic address that results in a displacement outside of this range is an invalid address.

The following are examples of coding of jump addresses:

JMP	START	Transfer control to the instruction at location START.
JNE	LOOP	Transfer control to the instruction at location LOOP if the equal status bit is equal to zero, or to the following instruction if the equal status bit is equal to one.
JOP	ODD	Transfer control to the instruction at location ODD if the odd parity status bit is equal to one, or to the following instruction if the odd parity status bit is equal to zero.
JOC	CARRY	Transfer control to the instruction at location CARRY if the carry status bit is equal to one, or to the following instruction if the carry status bit is equal to zero.

3.6.7 CODING OF CRU ADDRESSES. The Communication Register Unit is addressed using base addressing. The CRU instructions that transfer or test a single bit specify that bit as a displacement in the range of −128 through +127 from the base address in workspace register 12. The CRU instructions that transfer a group of consecutive bits require that the base address in workspace register 12 be the lowest bit address in the group of bits. In either case, the base address in workspace register 12 is shifted one bit position to the left, placing it in bits 3 through 14 of the register. This means that the base address is effectively twice the bit number.

The displacement of an SB0, SBZ, or TB instruction is an integer in the range of −128 through +127, or an expression having a value within that range. If an expression is used, any symbol in the expression must have been previously defined. The following are examples of coding CRU instructions:

LI	R12,>40	Place the value 40_{16} into CRU base register. This value corresponds to CRU line 32.
SB0	3	Set CRU line 35 to one.
SBZ	A	Set CRU line 42 to zero.
TB	4	Place the value on CRU line 36 into the equal bit of the status register.

The first operand for the LDCR and STCR instructions is an address in either of the five addressing modes that specifies the memory locations for the transfer of data. The second operand is the number of bits to be transferred, expressed as an integer 0 through 15, or an expression or symbol having a value in that range. The value of zero transfers 16 bits. The following are additional examples of coding CRU instructions:

LI	R12,>20	Place the value 20_{16} in the CRU base register. This value corresponds to CRU line 16.
STCR	*R5,0	Place the value on CRU lines 16 through 31 into a word of memory at address in workspace register 5. CRU line 16 goes into bit 15, CRU line 31 into bit 0.
LI	R12,>28	Place the value 28_{16} in the CRU base register. This value corresponds to CRU line 20.
LDCR	@CHAR,8	Place the eight bits of a byte of memory at location CHAR on CRU lines 20 through 27. Bit 7 goes to CRU line 20, and bit 0 to CRU line 27.

3.6.8 CODING OF IMMEDIATE ADDRESSES. The Model 990 Computer instruction set includes seven instructions that use immediate addresses. The assembler places the immediate address in the word following the instruction word. The immediate address is used by the instruction as an operand. An immediate address (operand) may be coded as an integer, a symbol, or an expression. The following examples show coding of immediate operands:

LI	R4,>AD45	Place immediate value $AD45_{16}$ into workspace register 4.
LWPI	WS2	Place immediate value WS2 into WP register.
LIMI	15	Place immediate value 15 into interrupt mask, enabling all interrupt levels.
AI	6,-7	Add immediate value, -7, to the contents of workspace register 6, and place the result in workspace register 6.
CI	5,BUFF	Compare the contents of workspace register 5 to an immediate value, the value of label BUFF.
ANDI	R7,>0FF0	Perform an AND operation on the contents of workspace register 7 and an immediate value, $0FF0_{16}$, and place the result in workspace register 7.

3.7 EXAMPLE PROGRAM

A simple program has been prepared to show the coding and assembling of a program on the Model 990 Computer. The program is organized as a data division and a procedure assembled as a single module. A ten-digit account number is placed in a buffer called NUMBER as the program is assembled. The program computes a check digit for the account number exclusive of the least significant digit and replaces the least significant digit with the computed check digit. The check digit is the least significant digit of the sum of the second, fourth, sixth, and eighth digits plus the product of the sum of the remaining digits times three. This is one of several check digit algorithms used in business applications to promote accuracy.

The example does not include input and output programming because this is ordinarily done by the executive under which the program is executed. Refer to the user's guide for a specific executive for I/O programming information.

3.7.1 CODING THE SOURCE PROGRAM. Before beginning to code a program, the programmer should thoroughly define the problem. Then he should draw a flowchart of the desired solution to the problem. After checking the flowchart for accuracy, he is ready to begin writing the code. The source statements may be written on coding sheets as shown in figure 3-10. Include an adequate number of comment statements and use the comment fields of other source statements. Good comments make it easy to debug the program and to make any modifications required at a later time.

After the coding is complete, the data of the source statements should be transcribed onto the appropriate medium for assembly. The data may be keypunched on punched cards, or written onto a cassette using the 733 ASR Data Terminal. The data may also be entered directly from the keyboard of a 733 ASR Data Terminal or a CRT Display Terminal.

3.7.2 ASSEMBLING THE SOURCE CODE. Refer to the user's guide for a specific executive for information on loading and executing the assembler under that executive, or to the *Model 990 Computer Cross Support System User's Guide* for information on executing the Cross Assembler. Place the source code in the input device and execute the assembler. The assembler should print a listing similar to that shown in figure 3-11.

The data division consists of a three-word block at location TST, workspace WS, and a buffer at location NUMBER. Six workspace registers are initialized with constants required in the program. The block at location TST is typical of the interface required by the executives for the Model 990 Computer. Refer to the specific executive user's guide for information. The first BSS directive reserves 8 bytes at location WS for workspace registers 0 through 3. The next statement is a DATA directive that places the address of buffer NUMBER in the word that becomes workspace register 4. Similarly, the next DATA statement places a mask constant, $F0F0_{16}$ in the succeeding word for workspace register 5. Next, another DATA statement places constant 3030_{16} in the next word, for workspace register 6. The next statement is a DATA directive that places the address corresponding to label CORR in a word, for workspace register 7. Two more DATA directives place constants $0A00_{16}$ and 0600_{16} in the next two words, for workspace registers 8 and 9, respectively. The DATA directives are followed by a BSS directive that reserves 12 bytes for workspace registers 10 through 15, completing the workspace. The last statement in the Data Division is a DATA statement that initializes buffer NUMBER with the account number to be processed. The digits of the account number are represented as ASCII characters, two per word. The even-numbered bytes of buffer NUMBER contain the first, third, fifth, seventh, and ninth digits of the account number; i.e., the odd-numbered digits, as they are numbered in the algorithm for the check digit.

TI 990 ASSEMBLY CODING FORM

LABEL (1–6)	OPER (8–11)	OPERAND (13–25)	COMMENTS (26–60)
	TITL	'SAMPLE PROGRAM'	
	IDT	'SPLE1'	
*			
* DATA DIVISION			
*			
TST	DATA	WS,PC,>F	EXECUTIVE INTERFACE
WS	BSS	8	R0 - R3
	DATA	NUMBER	R4
	DATA	>FOFO	R5
	DATA	>3030	R6
	DATA	CORR	R7
	DATA	>0A00	R8
	DATA	>0600	R9
WS1	BSS	12	R10 - R15
NUMBER	DATA	'92','47','62','81','95'	
*			
* PROCEDURE - COMPUTES CHECK DIGIT FOR ACCOUNT NUMBER			
*			
PC	MOV	R4,R1	INITIALIZE ADDRESS
	LI	R2,5	INITIALIZE COUNT
L1	SZC	R5,*R1+	CLEAR MS BITS OF NUMBERS
	DEC	R2	DECREMENT COUNT
	JNE	L1	REPEAT FOR EACH WORD
	CLR	R3	CLEAR SUM
	MOV	R4,R1	INITIALIZE ADDRESS
	LI	R2,5	INITIALIZE COUNT

PROGRAM | PROGRAMMED BY | CHARGE | PAGE | OF

(A)132132 (1/3)

Figure 3-10. Example Program (Sheet 1 of 3)

945250-9701

TI 990 ASSEMBLY CODING FORM

LABEL	OPER	OPERAND	COMMENTS
L2	AB	*R1,R3	ADD ODD DIGITS
	BL	*R7	BCD CORRECTION
	INCT	R1	INCREMENT ADDRESS
	DEC	R2	DECREMENT COUNT
	JNE	L2	REPEAT FOR ODD DIGITS
	MOV	R3,R10	MOVE SUN TO R10
	SLA	R3,1	MULTIPLY SUM BY 2
	BL	*R7	BCD CORRECTION
	A	R10,R3	ADD SUM (SUM X 3)
	BL	*R7	BCD CORRECTION
	MOV	R4,R1	INITIALIZE ADDRESS
	INC	R1	FOR ODD BYTE
	LI	R2,4	INITIALIZE COUNT
L3	AB	*R1,R3	ADD EVEN DIGITS
	BL	*R7	BCD CORRECTION
	INCT	R1	INCREMENT ADDRESS
	DEC	R2	DECREMENT COUNT
	JNE	L3	REPEAT FOR EVEN DIGITS
	S2CB	R5,R3	STRIP OFF LEFT BITS
	MOVB	R3,@NUMBER+9	INSERT CHECK DIGIT
	MOV	R4,R1	INITIALIZE ADDRESS
	LI	R2,5	INITIALIZE COUNT
L4	SOC	R6,*R1+	INSERT ZONE BITS
	DEC	R2	DECREMENT COUNT
	JNE	L4	REPEAT FOR ALL WORDS
	XOP	@TERM,15	EXECUTIVE INTERFACE FOR EXIT

PROGRAM | PROGRAMMED BY | CHARGE | PAGE | OF

(A)132132 (2/3)

Figure 3-10. Example Program (Sheet 2 of 3)

TI 990 ASSEMBLY CODING FORM

LABEL (1–6)	OPER (8–11)	OPERAND (13–25)	COMMENTS (26–80)
CORR	C	R3,R8	VALID BCD?
	JLT	OK	JUMP IF SO
	A	R ,R3	CORRECT INVALID BCD
	SZCB	R5,R3	STRIP OFF MS BITS
OK	RT		
	END		

PROGRAM	PROGRAMMED BY	CHARGE	PAGE	OF

(A)132132 (3/3)

Figure 3-10. Example Program (Sheet 3 of 3)

```
SAMPLE PROGRAM                                              PAGE 0001

                        TOT   'SPLF1'
                *
                *  DATA DIVISION
                *
0000  0000  0006' TST   DATA  WS,PC,>F    EXECUTIVE INTERFACE
      0002  0032'
      0004  000F
0006  0006       WS    RSS   8           R8 - R15
0008  000F  0026' NUMBER DATA  >F0F0      R4
0009  0010  F0F0        DATA  >F0F0       R5
0010  0012  3030        DATA  >3030       R6
0011  0014  0002'       DATA  CORR        R7
0012  0016  0A00        DATA  >0A00       R8
0013  0018  0000        DATA  >0000       R9
0014  001A        WS1   RSS   12          R10 - R15
0015  002C  3032 NUMBER DATA  '92','47','62','81','90'
      002E  3437
      0030  3632
      0032  3631
      0034  3035
0034            *
0035            *  PROCEDURE - COMPUTES CHECK DIGIT FOR ACCOUNT NUMBER
0036            *
0037  0038  C044  PC    MOV   R4,R1       INITIALIZE ADDRESS
0038  003A  0202        LI    R2,5        INITIALIZE COUNT
      003C  0005
0039  003E  4C45  L1    SZC   R5,*R1+     CLEAR MS BITS OF NUMBERS
003A  0040  0602        DEC   R2          DECREMENT COUNT
0041  0042  16FD        JNE   L1          REPEAT FOR EACH WORD
0042  0044  04C3        CLR   R3          CLEAR SUM
0043  0046  C044        MOV   R4,R1       INITIALIZE ADDRESS
0044  0048  0202        LI    R2,5        INITIALIZE COUNT
      004A  0005
0045  004C  80D1  L2    AB    *R1,R3      ADD ODD DIGITS
0046  004E  0697        BL    *R7         BCD CORRECTION
0047  0050  05C1        INCT  R1          INCREMENT ADDRESS
0048  0052  0602        DEC   R2          DECREMENT COUNT
0049  0054  16F8        JNE   L2          REPEAT FOR ODD DIGITS
005A  0056  C283        MOV   R3,R10      MOVE SUM TO R10
0051  0058  0A13        SLA   R3,1        MULTIPLY SUM BY 2
0052  005A  0697        BL    *R7         BCD CORRECTION
0053  005C  A0CA        A     R10,R3      ADD SUM (SUM X 3)
0054  005E  0697        BL    *R7         BCD CORRECTION
0055  0060  C044        MOV   R4,R1       INITIALIZE ADDRESS
0056  0062  0581        INC   R1            FOR ODD BYTE
0057  0064  0202        LI    R2,4        INITIALIZE COUNT
      0066  0004
0058  0068  80D1  L3    AB    *R1,R3      ADD EVEN DIGITS
0059  006A  0697        BL    *R7         BCD CORRECTION
006A  006C  05C1        INCT  R1          INCREMENT ADDRESS
0061  006E  0602        DEC   R2          DECREMENT COUNT
0062  0070  16F8        JNE   L3          REPEAT FOR EVEN DIGITS
0063  0072  50C5        SZCB  R5,R3       STRIP OFF LEFT BITS
0064  0074  D803        MOVB  R3,@NUMBER+9  INSERT CHECK DIGIT
      0076  002F'
0065  0078  C044        MOV   R4,R1       INITIALIZE ADDRESS
0066  007A  0202        LI    R2,5        INITIALIZE COUNT
      007C  0005
0067  007E  EC40  L4    SOC   R6,*R1+      INSERT ZONE BITS
0068  0080  0602        DEC   R2          DECREMENT COUNT
0069  0082  16FD        JNE   L4          REPEAT FOR ALL WORDS
007A  0084  2FE8        XOP   @TERM,15    EXECUTIVE INTERFACE FOR EXIT
      0086  0030'
0071  0088  8203  CORR  C     R3,R8       VALID BCD?
0072  008A  1102        JLT   OK          JUMP IF SO
0073  008C  A0C9        A     R9,R3       CORRECT INVALID BCD
0074  008E  50C5        SZCB  R5,R3       STRIP OFF MS BITS
0075  0090  0456  OK    RT
0076            END
0000 ERS

(B)132133
```

Figure 3-11. Example Program Listing

The procedure begins with a MOV instruction that copies the addresss of buffer NUMBER in workspace register 4 into workspace register 1, where it is used to address the words of the buffer. The next instruction, a LI instruction places 5 in workspace register 2 as the word count. The next instruction, at location L1, is an SZC instruction that masks off the bits in the word at the address in workspace register 1 according to the mask in workspace register 5, and increments workspace register 1 by two. Since workspace register 1 contains the address of the first word in buffer NUMBER, and workspace register 5 contains $F0F0_{16}$, the effect of this instruction is to clear the four most significant bits of the characters in the buffer to zero, leaving the Binary Coded Decimal (BCD) values of the digits in the buffer. The next instruction, a DEC instruction, decrements the count in workspace register 2 and compares the remainder to zero. This instruction is followed by a JNE instruction that causes the instruction at location LI to be repeated for each of the five words of the buffer.

The next three instructions initialize variables for the summation of the odd-numbered digits of the account number. The CLR instruction zeros workspace register 3 in which the sum is to be accumulated. The MOV instruction re-initializes workspace register 1 with the buffer address, and the LI instruction re-initializes workspace register 2 with the word count of 5. The instruction at location L2, an AB instruction, adds the first character BCD value (at the address in workspace register 1) to the contents of workspace register 3, and places the sum in workspace register 3. The next instruction, a BL instruction, branches to the address in workspace register 7, the address of location CORR. This instruction transfers control to a subroutine that tests the sum in workspace register 3 to determine if it is a valid BCD value, and corrects the sum to the BCD equivalent when it is not a valid BCD value. The subroutine is described in a subsequent paragraph. It returns control to the instruction following the BL instruction with a valid BCD value in workspace register 3. The next instruction, an INCT instruction, increments the address in workspace register 1 to the next word in the buffer. Next, a DEC instruction reduces the word count in workspace register 2, and tests for a zero count. The last instruction in the loop, a JNE instruction, jumps to the instruction at location L2 until all five words of the buffer have been processed.

The next five instructions multiply the sum in workspace register 3 by 3, maintaining a valid BCD value. The first instruction, a MOV instruction, copies the sum from workspace register 3 into workspace register 10. The next instruction, a SLA instruction, shifts the contents of workspace register 3 to the left one bit position. This in effect multiplies the sum by two. The SLA instruction is followed by a branch to the BCD correction subroutine, and an A instruction. The A instruction adds the contents of workspace register 10 to the contents of workspace register 3, and places the sum in workspace register 3. This instruction is also followed by a branch to location CORR. When control returns following this branch, the valid BCD product of the sum multiplied by 3 is in workspace register 3.

The next three instructions initialize variables for adding the sum of the remaining digits to the value in workspace register 3. The MOV instruction re-initializes the address in workspace register 1, and the INC instruction adds one to the address. The effect of the INC instruction is to address the odd (least significant) bytes of the words in the buffer, which contain the even-numbered digits of the account number. The third instruction, an LI instruction, initializes the word count to four. Only four of the even-numbered digits of the account number are involved in computing the check digit. The instruction at location L3, an AB instruction, adds the contents of the byte at the address in workspace register 1 to the sum in workspace register 3, and places the new sum in register 3. The next instruction, a BL instruction, branches to the BCD correction subroutine, which performs correction, if needed, to provide a valid BCD value in workspace register 3. The BL instruction is followed by an INCT instruction that increments the address in workspace register 1 to the next word of the buffer. The next instruction is a DEC instruction, that decrements the word count in workspace register 2, and compares the result to zero. The next instruction, a JNE instruction, jumps to location L3 until all four digits have been added to the sum.

The next two instructions place the computed check digit in the buffer. The first, an SZCB instruction, sets the bits of the check digit in workspace register 3 that correspond to the one bits in the most significant byte of workspace register 5. Workspace register 5 contains $F0F0_{16}$, so the result of the SZCB instruction is to strip off the most significant four bits of the character that contains the check digit. The MOVB instruction copies the check digit in workspace 3 into the high order byte of buffer NUMBER.

The next two instructions initialize variables for restoring the BCD values in buffer NUMBER to ASCII characters. The first of these instructions, a MOV instruction, re-initializes the address in workspace register 1, and the second, an LI instruction, initializes the word count to five. The instruction at location L4, a SOC instruction, sets ones in the word at the address in workspace register 1 corresponding to the ones in workspace register 6, and increments the address in workspace register 1. Workspace register 6 contains 3030_{16}, so the result of this instruction is to place a hexadecimal 3 ahead of the BCD value in each byte of the word of the buffer. This converts the BCD values to the equivalent ASCII characters. The SOC instruction is followed by a DEC instruction that decrements the word count and compares the result to zero. It is followed by a JNE instruction that jumps to the instruction at location L4 until all words in the buffer have been converted to ASCII.

The next instruction, an XOP instruction, performs extended operation 15 using a value at location TERM. This instruction is typical of the interface with the executives for the Model 990 Computers to return control to the executive. Refer to the user's guide for a specific executive for more information.

The subroutine at location CORR tests the contents of workspace register 3 to determine whether it contains a valid BCD value, and to correct the contents to the equivalent BCD value. A valid BCD number is in the range of 0 through 9. When computation results in a value greater than 9, the least significant digit may be

corrected by adding 6 to the value, and clearing the four most significant bits to zero. The first instruction of the subroutine, a C instruction, compares the contents of workspace register 3 to the contents of workspace register 8. Workspace register 8 contains $0A00_{16}$. The next instruction, a JLT instruction, jumps to the instruction at location OK when the result of the comparison is less than. If the contents of workspace register 3 are not less than $0A00_{16}$, correction is required. The A instruction adds the contents of workspace register 9 to the contents of workspace register 3, and places the result in workspace register 3. The next instruction is a SZCB instruction that sets bits of the contents of the most significant byte of workspace register 3 to one that correspond to the one bits in the most significant byte of workspace register 5. Workspace register 5 contains $F0F0_{16}$, so the instruction clears the four most significant bits of the contents of workspace register 3. The instruction at label OK is an RT pseudo-instruction that returns control to the calling program at the address stored in workspace register 11. The last statement is an END directive that terminates the source program.

3.7.3 EXECUTING THE PROGRAM. After assembling the program, load the object code into memory and debug the program. Refer to the user's guide for the specific executive under which the program is to be executed for loading instructions. Different debug programs are available with different executives, so which of these is used also depends upon the applicable executive. When a program is assembled on the Cross Assembler, it may be debugged using the TMS9900 Simulator as described in the **Model 990 Computer Cross Support System User's Guide.** The correct contents of buffer NUMBER following execution of the program consists of the following ASCII characters:

9247628190

SECTION IV

990 FAMILY SOFTWARE

4.1 990 STANDARD SOFTWARE

The standard software developed by Texas Instruments complements the versatile and powerful features of the 990 Computer Family. This software provides the range and depth of support to significantly reduce the user's program development effort. In addition to both memory resident and disc resident operating systems, a comprehensive selection of higher level languages and program development aids are available. This section describes the functional characteristics of these features and the physical construction of the configured software packages available for 990 users. A general description of the compatibility features, guidelines for minimum hardware requirements, plus lists of hardware supported and recommended software prerequisites, are included to acquaint the new 990 user with the standard 990 Family software.

The 990 system software features two operating systems. Both are multi-tasking real time systems:

- TX990 Memory Resident System
- DX10 Disc Executive System.

Program development tools include assembly language, FORTRAN, COBOL and BASIC*. Assembly language is implemented at three functional levels by three different assembler programs. All assemblers provide absolute or relocatable object programs and are upward compatible. The minimum assembler operates in one pass on the TI Model 733 ASR. An intermediate level assembly language is implemented by a two-pass Cross Assembler for IBM System/3X0 operation. A very comprehensive 990 assembly language is implemented by the macro assembler furnished with DX10 development systems. Software facilities for source program preparation, object program linking and debugging are packaged along with the appropriate assembler to provide integrated program development systems.

Higher level language support is provided for the 990/10 and includes FORTRAN IV, COBOL, and Multi-user BASIC. These languages are fully supported by the DX10 operating system. FORTRAN programs can be developed using the DX10 compiler and then linked to the memory resident operating system (TX990) to form dedicated application systems for use in either the 990/4 microcomputer or 990/10 minicomputer.

4.2 990 FAMILY COMPATIBILITY

The 990 Family is centered around a standard instruction set and common architectural characteristics. Primary family compatibility is achieved by conformity of the TMS9900, 990/4 and 990/10 to the standard family instruction set. Similarly, most architectural characteristics are shared by all members of the 990 family. These common points enhance program transportability between members of the 990 family.

4.2.1 EQUIPMENT CHARACTERISTICS. The primary concept of compatibility for interchangeability of software between the 990/4 and 990/10 requires compensation for unique equipment characteristics.

The unique features of the 990/4 are:

- Optional memory write protect on expansion memory
- PROM/ROM/Static RAM memory options.

*Trademark registered by the Trustees of Dartmouth College, Hanover, N.H.

- 9900 Memory Bus
- Eight interrupts standard.

Unique features of the 990/10 are:

- Optional Memory mapping providing memory addressing to one million words, privileged instructions, memory protection and long distance addressing
- TILINE memory bus
- Sixteen interrupts standard
- Extended operation (XOP) hardware expansion capability
- Expanded CRU addressability
- Execution speed twice that of the 990/4
- Illegal operation decoding
- Level 2 interrupt for memory parity error, illegal operation, privileged instruction fault, TILINE timeout, or memory map fault
- Augmented Status Register with enable/disable bits for map selection and privilege instructions.

4.2.2 990 STANDARD SOFTWARE COMPATIBILITY. Software compatibility within the 990 family is fundamentally derived from utilization of a standard instruction set and common architectural features. The assembly languages implemented by the one-pass assembler, the two-pass IBM System/3X0 Cross Assembler, and the DX10 development system are upward compatible. The set of supervisor calls implemented in TX990, the memory resident operating system, and in DX10, the disc based operating system, are also upward compatible. In general, 990 system software is planned so that smaller, less complicated systems are a subset of larger systems. However, software that takes full advantage of unique hardware features cannot be easily transferred to a system that does not have that feature.

For example, the prototyping system software utilizes the optional write protection feature of the 990/4 expansion memory. This feature is not provided in 990/10 memory systems.

4.3 990 OPERATING SYSTEM SOFTWARE

Two operating systems are offered for the 990 computer family. They provide a flexible framework for implementation of application software. These two systems, a

memory resident system and a disc based system, are structured to ensure application software compatibility where desired. The Memory Resident Operating System (TX990) provides the 990/4 and 990/10 users with a memory resident operating system suitable for low cost computer controller applications using a minimum of supporting peripheral devices. DX10, the general purpose multi-tasking operating system, provides support and flexibility for larger 990/10 systems requiring high mass storage support and rapid access to programs and data files.

4.3.1 MEMORY RESIDENT OPERATING SYSTEM (TX990). The TX990 memory resident operating system software is provided as modular, linkable components to allow 990 users to select only required functions. These selected functions are then combined with user developed application tasks into a compact software system.

4.3.1.1 Features. The Memory Resident TX990 operates in a real-time environment and is capable of controlling concurrent execution of multiple tasks. Its main functions include task scheduling, interrupt handling, I/O servicing, and processing standard system functions.

Task scheduling is performed on a priority basis. Four levels of priority are provided. Any number of tasks may be included at each level. First-In-First-Out (FIFO) scheduling is performed for tasks at the same priority level. In addition to the priority structure, time slicing helps share CPU time among tasks.

The input/output supervisor of TX990 processes I/O requests from the tasks. Its functions include accepting I/O requests from application tasks, initiating I/O operations, synchronizing task execution with I/O operations, and managing the interface between application tasks and the device service routines that control operation of the system I/O devices. System service routines perform commonly used functions. These routines are accessable by the tasks via supervisor calls. TX990 supervisor calls, input/output functions and task interface are upward compatible with the DX10 operating system.

4.3.1.2 System Design Considerations. When designing a 990 system in an application requiring a memory resident system, four factors should be considered:

1. Method of loading system software,
2. Operator communications during normal operation,
3. Program and data storage requirements,
4. Hardware components to be supported.

The following information provides the guidelines for establishing preliminary estimates for a TX990 functional system. Consult your local Texas Instruments field sales office for additional specifications.

A TX990 system is generated by linking the applications tasks with TX990 system modules. Once the TX990 system software is linked with application tasks, the resultant software system can be transferred to the target 990 system via magnetic tape cassette for the 733 ASR Data Terminal. The availability of PROM devices for the 990 allows user creation and use of special methods for downloading software. As an example, a user generated loader, resident in PROM, could be designed for downloading system software over an asynchronous communications line from another 990 system. Other methods are also possible with user-generated PROM/ROM loaders.

TX990 operator communications interface primarily through the 733 ASR Data Terminal. If the user designs the loading methods, the TX990 operator communications can also interface through a 733 KSR Data Terminal or a 913 Video Display Terminal. The operator communication module is optional in TX990 systems.

Program and data storage requirements depend upon the particular application; however, the following list provides preliminary design guidelines for users regarding memory requirements for the TX990 operating system:

Operating System Nucleus:	1500 (16 bit) words
Card Reader:	300 words
Line Printer:	250 words
733 ASR:	500 words
Floppy Disc:	500 words (storage only)
913A Video Display:	300 to 4000 words
Asynchronous Communications:	2500 words
Synchronous Communications:	2500 words

The 913A Video Display Terminal storage requirements depend upon the degree of utilization of available functions, and upon the number of included user-developed functions.

4.3.1.3 System Hardware. The minimum hardware required to support a TX990 system after generation and linkage of application software depends on system design requirements covered above. When using standard 990 ROM loaders and TX990 system modules, the following equipment is necessary to meet system requirements:

- 990 central processor unit with 4,096 words of memory
- 733 ASR ROM loader

- 733 ASR Data Terminal
- Necessary chassis and power supply to support above hardware.

The additional hardware supported by standard TX990 modules includes:

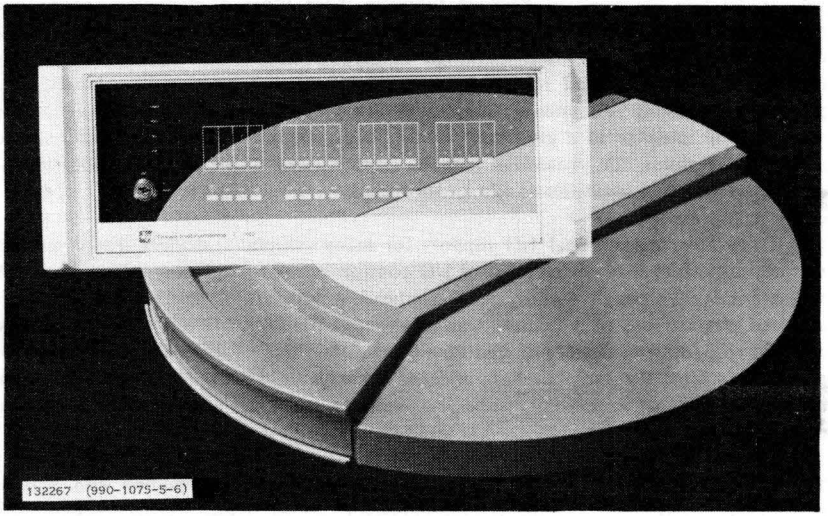
- Model 804 Card Reader
- Model 306 Line Printer
- Model 588 Line Printer
- 733 KSR Data Terminal(s)
- 913 Video Display Terminal(s)
- Additional 733 ASR Data Terminals
- Floppy disc (for storage medium)
- Memory expansion to 32K (16 bit) words.

4.3.1.4 System Generation Requirements. A DX10 based, 990 software development system is required for generating application programs and building TX990 target systems. Other facilities available are described in succeeding paragraphs.

4.3.2 GENERAL PURPOSE DISC BASED OPERATING SYSTEM (DX10). The Disc Executive (DX10) is a disc based operating system for the 990/10 Computer. It is a multi-tasking system which supports from 24K words to one megaword of memory. The minimum resident memory requirement for DX10 is 10K words. DX10 utilizes the 990 vectored interrupt structure and fast context-switch capability to provide excellent real-time support.

4.3.2.1 Features. Task structure under DX10 provides a maximum of flexibility to the user's programs. A full 32K address space is available to each task; this address space may be divided in up to three segments comprised of any combination of the following entities:

- A self-contained task
- A separately attached re-entrant procedure
- A re-entrant subroutine library



DX10 - Disc-Based Power for the 990/10 Minicomputer

- A system common area for inter-task communication
- Dynamically allocated (via supervisor call) memory acquired at execution time.

Overlay support allows the size of user programs to be virtually unlimited.

Task scheduling is accomplished on a priority basis using a time-slicing algorithm. There are four priority levels, scheduled on a FIFO basis within each level. To prevent total lockout of low priority tasks, the scheduler periodically skips the higher priority levels for one time slice.

DX10 provides support for both memory resident and disc resident tasks. Installation of tasks is an Operator Communications Package (OCP) function. Disc resident tasks are relocated at installation time, resulting in faster response upon execution of the task.

Tasks are isolated within their address space through the use of hardware, memory mapping registers. This feature ensures the integrity of both other tasks and the executive. The memory mapping technique also facilitates a roll-in/roll-out scheme which is implemented in the scheduler when available memory is exhausted. This swapping is transparent to the user; however, as in any swapping scheme, response time may be affected as system loading increases.

The DX10 File Management Package supports three types of disc files: sequential, relative record, and indexed. File organization on disc may be specified to be either contiguous (fixed length) or non-contiguous (dynamically allocated). Multiple logical records for sequential and relative record file I/O may be optionally blocked into physical records to minimize access time. Two levels of access are available to maintain file integrity in a multi-user environment: exclusive write access and shared access. In addition, file protection may be implemented through the specification of an 8-bit verification identifier at file creation time.

DX10 provides conventional I/O support for many peripheral devices (listed later in this section). A special character-level I/O routine interfaces the computer with 913A Video Display Terminals with minimal overhead. A measure of device independence is achieved through use of a Logical Unit Number (LUNO) in I/O calls. LUNOs may be assigned to peripheral devices or disc files using either an OCP function or a supervisor call from within the task. LUNOs assigned through OCP are global (default) assignments, while those assigned by supervisor call are task-local and override any global definitions.

The Operator Communications Package is a dual mode (interactive and batch), free-format interface, providing a comprehensive set of debug, status, tasking, and system functions.

4.3.2.2 System Generation. Custom system generation is facilitated by an interactive system utility. This utility accepts the user's conversational description of his system and constructs the necessary software to fit his specifications. The utility can also test the newly generated system prior to installing it as the permanent system. This utility may also be used at any time the user wishes to upgrade his system. The utility is designed to relieve the user of the need to be concerned about system details in order to perform a system generation.

4.3.2.3 System Hardware. The Disc Executive is designed to fully utilize the high performance features of the 990/10 minicomputer. To support this capability, the following minimum hardware is required:

- 990/10 with memory mapping
- 24K words of memory (operating system requires 10K words of memory)
- ROM loader
- 733 ASR Data Terminal
- Model DS31 disc (removable disc)
- Necessary chassis and power supply to support above hardware.

The additional hardware supported includes following:

- Memory expansion to 1 million words
- Model 804 Card Reader
- Model 306 Line Printer(s)
- Model 588 Line Printer(s)
- 913 Video Display Terminal(s)
- 733 ASR/KSR Data Terminal(s)
- Floppy disc as storage medium
- Additional DS31 or DS32 disc (fixed disc) storage
- Asynchronous communications modules
- Synchronous communications modules.

Additional hardware interface modules are available to facilitate implementation of special custom devices. DX10 source on card media is available for modifying system software for these custom functions.

4.4 PROGRAMMING LANGUAGES

The 990 Family of Computers offers a choice of programming languages to meet the programming environment of a wide variety of users. For users developing software for tightly constrained hardware systems, an efficient assembly language optimizes memory usage while maintaining a syntax that is easy to understand. For programming on larger computer systems in the Family, high level programming languages are also available. FORTRAN IV for mathematical and scientific applications, COBOL for business environments, and Multiuser BASIC for simplified interactive programming offer the user of 990 Family Computers a choice of effective high level languages.

4.4.1 ASSEMBLY LANGUAGE. The assembly language for the 990 Family of Computers is a simple and easy to learn symbolic language that can produce highly efficient programs for use on any member of the computer family. Mnemonic codes are used to represent computer machine instructions and are much easier to use and remember than the binary bit patterns of their corresponding machine instructions. Mnemonic codes are also provided for directives that control the assembly process, reserve storage, initialize data in the program and assign symbols to values that are used in the program. An assembly language program, called a source program, must be processed by an assembler to obtain a machine language program that can be executed by the computer. The output of an assembler is called an object program or object program module.

All 990 assemblers provide absolute or relocatable object programs and are upward compatible.

The assembly language includes the instruction set, plus the assembler directives that control the assembly process, place data in the program, and provide linkage data for modules that have been assembled separately and must be subsequently linked together. All addressing modes may be coded in assembly language, and symbols for workspace register addresses are predefined. Operands of machine instructions and assembler directives may be expressions using arithmetic operators and symbolic values.

4.4.1.1 One-Pass Assembler. The one-pass assembler (PX9ASM) produces a program or a linkable object module with only one pass of the source program. It supports all machine instructions for the 990/4 microcomputer plus a full complement of assembler directives. The assembler runs equally well on either a 990/4 microcomputer or a 990/10 minicomputer under control of the 990 Debug Monitor (PX9MTR). PX9ASM provides the convenience of a one-pass assembler with very few of the restrictions that normally complicate one-pass assemblers.

4.4.1.2 Cross Assembler. The cross assembler is part of the Cross Support System that is available on timesharing networks. It allows the user to assemble 990 Assembly Language source code on an IBM System/3X0 computer system and receive programs or linkable object modules in 990 standard object format. The cross assembler supports the instruction and directive set of the one-pass assembler plus an additional directive for specifying output options. The cross assembler processes the source code input in two passes. An option for the cross assembler combines a symbol table listing with the object code output; additionally, a cross-reference listing and/or object listing may be printed.

4.4.1.3 Macro Assembler. The macro assembler implements a comprehensive assembly language for the 990 Family of Computers. The macro assembler (SDSMAC) provides additional assembler directives and includes the extended instruction set recognized by 990/10 computers with memory mapping feature.

SDSMAC extensions include:

- Support for logical and relational operators in expressions
- Optional listings of a symbol table with the object code output, or a cross-reference listing and/or object listing with the source listing

- Ability to define macro-instructions as sets of assembly language instructions and directives

- Tracking of the current workspace to identify memory addresses that are also workspace registers so that the more efficient workspace address can be used wherever possible.

The macro assembler is a powerful tool for development of programs for members of the 990 Computer Family.

4.4.2 **FORTRAN IV.** FORTRAN is an easy-to-use, high-level computer language that allows complex problems to be stated in common mathematical and English expressions for input to the computer. However, in order to translate this high level language into a form usable by the computer, the FORTRAN input must first be processed by a FORTRAN compiler program.

The FORTRAN compiler for the 990 Family of Computers translates FORTRAN language input code into machine code for running on a Model 990/10 computer system. The compiler operates under the DX10 operating system. Together with the operating system requirements, the computer system must have at least 32K words of memory plus disc storage files. The FORTRAN compiler conforms to the American National Standards Institute (ANSI) standard FORTRAN, or FORTRAN IV. In addition, the compiler incorporates the FORTRAN extension recommended by the Instrument Society of America (ISA) in their document ISA-S61.1, 1975.

Texas Instruments has also incorporated into the FORTRAN compiler several useful attributes that provide more effective coding and aid in program development. These additional features include:

- Direct access I/O

- Overlapped I/O

- Free format source input

- Literal character strings represented in quoted form

- Double-word (32-bit) integer data types

- An IMPLICIT statement to allow data type declaration for groups of data without separately declaring the type of each group member

- DATA statement array names

- Re-entrant subprograms (optional)

- Scaled binary data types

- COPY directive
- ACCEPT and DISPLAY directives for interfacing with a 913 VDT.

A FORTRAN function library is provided that includes all intrinsic functions and the basic external functions defined in the ANSI standard. Optionally, the compiler can generate a cross-reference listing during execution for each variable in the program and provide a debug module that references specific line numbers in the source program input. A component module of the compiler coordinates the interaction of the compiler with the operating system. In addition, a runtime utility package performs the following services for the compiler:

- Format editing
- Data conversion from ASCII code to binary and from binary to ASCII as required
- Subprogram linking to allow easy argument setup
- Floating point arithmetic routines
- FORTRAN tracing to aid in debugging the FORTRAN source code
- Other special FORTRAN functions such as PAUSEn.

4.4.3 COBOL. COBOL is a high level computer language that allows problems to be stated in words and syntax similar to the English language. It consists of a set of English words and symbols that the programmer uses to define a problem and create his program to solve that problem. Because of its similarity to the English language, programs written in COBOL are self-documenting and reduce the time required for a new programmer to understand the language.

The COBOL language compiler for the 990 Family of Computers operates under the DX10 operating system and requires a minimum memory size of 32K words. This compiler complies with the American National Standard Institute COBOL subset as defined in ANSI document X3.23-1974, and incorporates extensions to this subset to provide added capabilities. The compiler package employs the following COBOL modules at the level indicated:

Level 1:

- Nucleus
- Table Handling
- Sequential I/O

- Library
- Segmentation
- Indexed I/O.

Level 2:

- Relative I/O.

Nonstandard Features:

- Debug
- Communication.

In addition to these modules, 990 COBOL incorporates software modules that simplify the interface between the compiler and the operating system. These modules perform the following functions:

- Optionally, produce a cross reference listing for each variable in the program
- Aid in debugging a COBOL program at the source level
- Coordinate interaction between the compiler and the operating system
- Perform re-entrant runtime functions required for execution of a COBOL program; these functions include control, editing, decimal and binary arithmetic, logical I/O interface and automatic debug
- Interface the COBOL I/O structure to the operating system
- Perform COBOL implied functions not supported by the operating system.

4.4.4 MULTIUSER BASIC. BASIC is an easily understood programming language that is applicable to both business and scientific problem solving applications. It is an interactive language for simultaneous use by several different users, each working from his own terminal. The computer provides feedback to each command entered on the terminal and points out any programming errors by printing diagnostic messages following the erroneous command.

The BASIC interpreter operates under the DX10 Disc Executive. The language implemented by this system is equivalent to Dartmouth BASIC with certain extensions to enhance its use in the business world. The system package includes the following components:

- An executive for processing commands and translating BASIC statements
- A re-entrant multi-terminal interface module to coordinate the interaction of the executive, the interpreter and the operating system
- A re-entrant runtime interpreter that performs all arithmetic and logical functions during the execution of a BASIC program. These functions include: floating point arithmetic, string manipulation, I/O editing and matrix arithmetic.

The memory size required to operate the BASIC system varies with the amount of memory assigned to each user and to the number of users involved in the system. A minimum of 32K words of memory supplies space for the system package, user housekeeping areas, user programs and the operating system.

4.5 SYSTEM SOFTWARE PACKAGES

The following paragraphs define the configured system software packages available for 990 users. These package contents are listed with minimum hardware requirements, and general software features. Additional descriptions are provided for utility functions not defined in earlier paragraphs.

4.5.1 THE 990 PROTOTYPING SYSTEM SOFTWARE PACKAGE. The Prototyping System package is specifically designed to facilitate the generation and verification of software for the TMS9900 microprocessor based computers. This classification includes the 990/4 computer-on-a-board, and customer-designed dedicated purpose computers using TMS9900 microprocessor.

The hardware configuration required by the Prototyping System is as follows:

- 990/4 microcomputer with 4,096 words of dynamic RAM, 512 words of ROM, and 256 words of static RAM
- 990/4 memory expansion board with 4096 words of dynamic RAM with write protect
- 733 ASR Data Terminal
- 733 ASR bootstrap loader
- Chassis, power supply, and packaging

- Programmer panel
- PROM/ROM Programmer (optional).

The Prototyping System package supports up to 32K words of memory and consists of the following:

- Prototyping System Debug Monitor (PX9MTR)
- One-pass 990 Assembler (PX9ASM)
- 733 ASR Based ASCII Text Editor (PX9EDT)
- Programmer Panel and Bootstrap Firmware.

The Prototyping system is available in loadable form on magnetic tape cassette.

4.5.1.1 Programmer Panel and Bootstrap Firmware. This software executes from ROM situated in the last 512 words of the 32K address space, and serves as the system bootstrap. It also gives the operator an elementary level of control over executing programs via the firmware panel when the Debug Monitor (PX9MTR) is not resident.

PX9MTR programs are provided in a combination of the standard object format and a compressed absolute load format. Standard object format may contain either relocatable or absolute load data and is produced by all 990 assemblers. The standard format boot is used for functions that require the ability to change the load point at bootstrap time. The compressed absolute data format is used to speed up program loading from the 733 ASR cassettes.

4.5.1.2 Prototyping System Debug Monitor (PX9MTR). PX9MTR is a modular control program which resides in the memory space shown in figure 4-1.

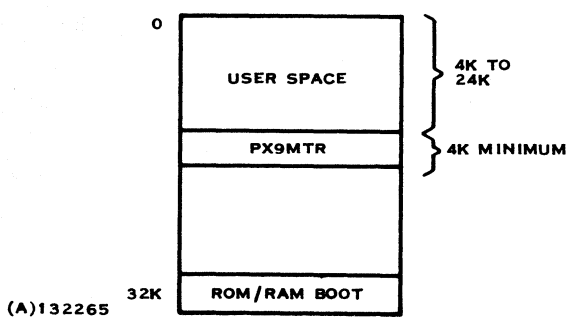


Figure 4-1. PX9MTR Memory Allocation

PX9MTR is the first member in a series of upward compatible, general purpose 990 standard control programs. Programs written to run under PX9MTR will also run under TX990 and DX10 operating systems. Further, programs targeted for PX9MTR may be written and initially verified under TX990 and DX10 if the program limits itself to only those program services provided by PX9MTR. Primarily, PX9MTR provides an efficient mechanism to the TMS9900 user for generating and testing standalone programs, and for transferring those programs into PROM or ROM devices.

PX9MTR provides four distinct purposes which are discussed below.

Operator Interface. PX9MTR interacts with the operator via the 733 ASR keyboard and printer to receive and decode commands, and to activate the various command processors. All commands consist of two characters and associated operands and are similar in style to their counterparts in operator interface software for other 990 systems. Examples of the capabilities offered are:

- LP - Load a Program
- AL - Assign a Logical Unit Number (LUNO) to specific devices
- GO - Execute a Program
- OV - Overlay the monitor transient area with a cassette resident command (having once been loaded, transient commands are operated exactly like resident commands).

Program Service Call Interface. PX9MTR intercepts requests for monitor service from user programs. I/O requests are passed to the I/O Executive. Non-I/O requests are processed immediately and consist of user program termination and the following data conversion services:

NOTE

Decimal ASCII and Hexadecimal ASCII refer to character strings that consist of ASCII characters representing the decimal or hexadecimal numbers, respectively.

- Decimal ASCII to binary
- Hexadecimal ASCII to binary
- Binary to Decimal ASCII
- Binary to Hexadecimal ASCII.

Upward compatibility with other members of the 990 software family is maintained in that these service calls are format compatible with the corresponding service calls defined by TX990 and DX10.

I/O Executive. The I/O executive decodes and processes 733 ASR I/O service requests from other PX9MTR modules and from user programs. Upward compatibility is maintained by the I/O service request being format compatible with TX990 and DX10. Upward compatibility is further ensured by the executive strictly adhering to standard 990 input/output rules for those I/O services defined in PX9MTR. Similar to other 990 system executives, PX9MTR provides file and record level I/O performed independently of the device type to which the I/O is directed. Input/output is status driven consistent with requirements of the program debug function.

Program Debug. Program debug consists of a collection of operator command processor modules that assist in the test and validation of user generated programs. The debug facilities are constrained to execute from program and data stored entirely within the 4096 words reserved for PX9MTR, leaving the interrupt and XOP vectors for the exclusive use of user programs. The commands are functionally categorized into the following groups:

- Set Commands - These commands allow the user to define up to four each of the following aids: program counter breakpoints, formatted snapshots, trace regions, and trace formats.

- Clear Commands - These commands allow the user to negate the effect of a previous set command.

- Inspect Commands - These commands allow the user to display the contents of AU registers, workspace registers, memory regions, and CRU lines. These commands are also used to force snapshots.

- Modify Commands - These commands allow the user to examine, and optionally modify, memory, workspace registers, AU registers, and CRU lines (inspect input, modify output).

- Miscellaneous Commands - These commands include functions such as word and byte level memory search, and hexadecimal arithmetic with automatic decimal conversion.

These debug tools provide both the ease of use required by the novice programmer, and the power demanded by the sophisticated programmer in the test and verification of user generated programs. The novice need only learn 4 types of operations (Set, Clear, Inspect, Modify) to be applied to any of several debug or machine resources (memory, breakpoints, etc.). The experienced user will quickly learn to associate snapshots and trace formats (by number) with specific breakpoints and trace regions

respectively. Trace formats and snapshots are predefined for the novice but may be modified if desired.

Transient Commands. To efficiently use the 4096 words reserved for PX9MTR, certain infrequently used command processors are stored on cassette tape and are loaded into a transient area within PX9MTR. The transient processor, once loaded, can contain only one function at a time, but commands in this category may be re-used as often as desired without being reloaded. These commands are listed and explained below:

- LINK AND LOAD - The transient area is initially loaded with this processor, and it is immediately usable after bootstrapping PX9MTR. The Link and Load (PX9LAL) command inputs one or more object modules (from a 990 Assembler) and/or linked object module (from SDSLED or the cross-support systems) from magnetic tape cassette, and performs a relocating, linking load of the modules into a single program in the user area of 990/4 memory (RAM area not occupied by PX9MTR). Both relocatable and absolute object code may be intermixed in the input stream to the loader. Once loaded, the operator can issue the GO (or RUN) command to execute the program in either a production or a controlled debug mode.

- PROGRAM PROM - Once a program has been verified, the code may be transferred to PROM devices by the PROM Programmer commands. These are an extremely flexible set of commands which are used to move data from the 990/4 memory to PROM devices in a manner consistent with the characteristics of the PROM device types being used. The commands can program PROMs targeted for any word width from 1 to 16 bits implemented with any number of PROM device types as long as two requirements are met:

  1. A personality card, which adapts the programmer hardware to the PROM, exists for the chosen device type.

  2. The data to be programmed is loaded into 990/4 memory in an orderly pattern. After specifying the data organization in memory and the data organization on the PROM device(s), the operator can then select the Program mode, Verify mode, or Program with Verification mode.

- ABSOLUTE DUMP - The ABS DUMP command saves a program/data space in memory on 733 ASR cassettes. The memory image is stored in compressed absolute load format, and can be reloaded using either the monitor load command or the ROM bootstrap. The command can also be used to save an entire program session complete with current debug parameter definitions. The session can then be rebooted, and continued as if it were never interrupted.

- INSTRUCTION TRACE - The trace feature permits the user to specify up to four different program regions for instruction level tracing. Associated with each region is a template number defining the trace point elements and conditions under which the elements will be printed. Four normal templates are predefined at trace load time to simplify the procedure for the novice user. The templates may be redefined via trace commands by the more experienced user to reduce, expand or alter the elements for printing as each instruction executes. Examples of traceable elements are: program counter, status register, workspace pointer, workspace registers, specific memory words, branch instructions, operand effective addresses, and instructions.

  A given template may be associated with several trace regions; however, only one template per region is permitted. The trace is an effective tutorial aid for the beginning programmer as well as a powerful debug tool for the experienced programmer.

- BNPF DUMP - This feature allows the user to move data from 990/4 memory to cassettes in card image, BNPF format. This format has become a generally accepted industry standard for mass-producing ROM and/or PROM devices.

- HIGH/LOW DUMP - High/Low format. Moves data to cassettes in card image.

4.5.1.3 Prototyping System Text Editor (PX9EDT). PX9EDT is an interactive text editor that runs as a user program under the Prototyping System Monitor (PX9MTR). PX9EDT edits existing source code or creates and saves new source code. It reads an existing program then from magnetic tape cassette to a memory buffer for editing and then outputs it to a second cassette. Programs which are too large to fit in memory at one time are scrolled from the input cassette into memory to satisfy user references. The text is then written to the output cassette when the user completes editing his current buffer.

The text editor processes 3 different classes of commands:

- Set-up commands control the edit operation by specifying print options, text margins (for printing), and column limits for use with search commands.

- Edit operation commands perform the actual text manipulation. Edit operations work on either a sequence of lines specified by the line number or on a sequence of lines relative to a pointer which may be positioned within the buffer. Edit operations may change, insert, move, remove, or print lines of text. Other operations can set the pointer position and cause additional data to be read from cassette, or search for specific character strings.

- Output commands cause the current buffer, and optionally, any remaining input, to be written to the output cassette.

Since the object module format for the 990 family consists of ASCII strings acceptable to the text editor, PX9EDT may also be used to edit object modules.

4.5.1.4 Prototyping System Assembler (PX9ASM). PX9ASM is a one-pass assembler that runs as a user program under the Prototyping System Monitor (PX9MTR). PX9ASM accepts source code input from cassette and produces an object program on cassette, a listing and an error summary. The object code produced may contain either relocatable or absolute origin code. It may also contain references to external symbols in other modules and define external symbols. A collection of object modules may then be linked and loaded into memory by the linking loader (PX9LAL).

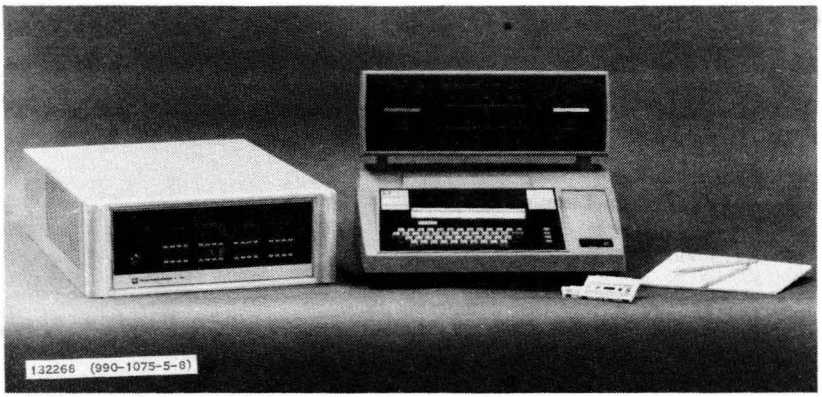
4.5.2 990-733 ASR SYSTEM SOFTWARE. The 990-733 ASR system software package is a magnetic tape cassette based system that supports application program development for all members of the 990 Computer Family. The package requires a minimum hardware configuration consisting of the following equipment:

- A 990/4 microcomputer with 4K words of RAM
- Expanded memory controller with 4K words of RAM
- 733 ASR ROM Loader
- 733 ASR Data Terminal
- Necessary chassis, power supply, and packaging.

In addition, the package will support up to a total of 32K words of memory. The software in this package includes the following program modules:

- System debug monitor (PX9MTR)
- One-pass assembler (PX9ASM)
- Linking loader (PX9LAL)
- Source editor (PX9EDT)
- Instruction level trace package (TRACE).

This package is available in object format on magnetic tape cassette for loading into the system via the 733 ASR Data Terminal. Additionally, a source format for the package is available on punched cards for use by customers wishing to add their own modifications.



990/4 - 733 ASR Software System

4.5.3 990/10 DISC SYSTEM SOFTWARE PACKAGE. The Disc System software package supports a software development facility centered around a Model 990/10 minicomputer with a Model DS31 moving head disc (removable disc). This system supplies a coherent set of capabilities for creating and debugging application programs to be used for all members of the 990 Computer Family.

The software included in the Disc System software package includes the following program modules:

- Disc based operating system (DX10) (see paragraph 4.3.2)
- Automatic system generation program (GEN990)
- Macro-assembler (SDSMAC) (see paragraph 4.4.1.3)
- 913 Video Display Terminal source editor (SDSTIE)
- Link editor (SDSLED)
- 913 Video Display Terminal operator's console and debug package (SDSDEB)
- Librarian (SDSLIB)
- Card reader loader program.

The 990/10 Disc System software requires the following minimum hardware for successful loading and execution:

- A 990/10 Minicomputer with memory mapping
- 733 ASR ROM loader
- 733 ASR Data Terminal
- 913 Video Display Terminal
- DS31 Moving Head Disc (removable disc)
- Necessary chassis and power supply.

In addition to this minimum equipment, the operating system supports the following hardware:

- Addressable memory to 1 million words (see Section VI of this manual for memory limitations with standard chassis equipment)
- Model 804 Card Reader
- Model 306 Line Printer
- Model 588 Line Printer
- Floppy disc as storage device
- Additional 913 Video Display Terminals
- Additional DS31 Moving Head Disc (removable disc)
- DS32 Moving Head Disc (nonremovable disc)
- Asynchronous communications
- Synchronous communications.

The disc system software package is provided in binary image form on a disc cartridge for the DS31/32 moving head disc. Backup for the disc software is available on both punched cards and magnetic tape cassette. Linkable object modules are included on the system disc for custom system generation. Source code for the system is also available on either disc or punched card media. When the disc media is used, two DS31 moving head disc units are required for system backup.

Software options available for this system are the higher level programming languages FORTRAN IV, COBOL, and Multi-user BASIC described earlier. Additional options include both asynchronous and synchronous communications software.

4.5.3.1 913 VDT Source Editor (SDSTIE). SDSTIE provides the capability to create and modify source file data from the 913 Video Display Terminal. The editor receives a file on disc as input, executes the edito commands received from the 913 keyboard and creates a new source file on disc. The user views the file under modification on the display terminal screen. This "window" may be positioned forward and backward within the file using function keys and commands. Other function keys and commands allow the user to:

- Add, delete or change characters or fields in statements (records)
- Add or delete statements (records)
- Move, copy or delete a set of consecutive statements
- Locate, delete or change one or more occurrences of a character string within the file.

A new file may be composed from the 913 keyboard if no file exists.

4.5.3.2 Link Editor (SDSLED). SDSLED accepts object code modules produced by the assemblers and compilers, and links the modules to form a single linked object module on a file. Linking the independently assembled object modules consists of resolving the external references in each module. The resulting object module is equivalent to the object module that would have been written had the modules been combined and assembled as a single module. This module may also be used in subsequent link edits to other program sequences.

SDSLED supports linking modules into an overlay structure, which allows shared use of memory. An overlay structure consists of a resident program and a set of overlays that are not resident in memory but are loaded into memory as they are required. For example, some applications require initial processing, followed by several iterations of intermediate processing, and then final processing. The code common to the three types of processing may be resident, and the unique code for each type of processing may be an overlay. The resident portion and the overlay for the initial processing is loaded first, and executed. When the initial processing is complete, code in the resident portion loads the overlay for the intermediate processing. When only the final processing remains, code in the resident portion loads the overlay that performs the final processing. Instead of requiring memory space for the entire program, the program only requires space for the resident portion and the longest overlay.

913 VDT Source Editor

SDSLED supports commands to control the linking and to define the structure. These commands specify options for the linking operation, define the overlays to be produced, and identify the program modules. Object modules may be obtained from a library by name, or by a search of the library.

4.5.3.3 Operator Console and Debug Program (SDSDEB). SDSDEB is a symbolic debug program that executes in a Model 990/10 Computer under the control of the disc based operating system, DX10. SDSDEB uses the 913 VDT screen to dynamically display the contents of: the AU registers, workspace registers, areas of memory, or the CRU. The keyboard of the 913 VDT is used to enter commands and to enter data to modify the contents of memory or registers. The labels assigned when the program under test was assembled appear in the display as appropriate. Symbolic operation codes also appear, when applicable.

SDSDEB executes the program under test an instruction at a time, performing any traces or breakpoints specified by the user, and observing the protection limits specified by the user. A trace may be requested when a branch is taken, a CRU address is accessed, a memory location is accessed, an extended operation is performed, or after execution of each instruction. A breakpoint may be requested when a memory location is altered or referenced, when a CRU address is accessed, when the

PC value is within a specified range, when the ST register reaches a specified value, or when an extended operation at a specified level is executed. When a breakpoint occurs, execution of the program halts and specified memory locations are displayed. The user may resume execution by entering the appropriate command at the keyboard. Limits may be set on memory locations that may be altered, memory locations that may be referenced, CRU addresses that may be accessed, PC contents, ST contents, or extended operations. When an attempt is made to execute an instruction that would violate the limits, execution halts and the specified memory locations are displayed.

SDSDEB supports commands for use in debugging, one of which performs address arithmetic, converting operands and results for displaying in both decimal and hexadecimal form. The speed of the 913 VDT combined with the capabilities of SDSDEB provide an effective tool in software development.

4.5.3.4 Librarian (SDSLIB). SDSLIB executes on a Model 990/10 Computer under control of the disc based operating system, DX10. This system librarian provides a means of storing and retrieving files within a structured catalog. The catalog structure for a disc consists of a Master File Directory, (MFD), one or more User File Directories, (UFD), and the cataloged files. The MFD lists the UFDs, which partition the disc among users. Each UFD lists the files of the user or users for which it is defined. A file is identified to the librarian by specifying the disc (MFD), the UFD, and the file. For example, a file may be designated as:

DSC1. PGMDEV. MACROS

The first portion of the designation, DSC1, identifies the file as one of the files on a disc, DSC1. The file is in user file directory, PGMDEV. The name of the file is MACROS. This structure allows files named MACROS to be cataloged in more than one UFD, and a UFD named PGMDEV on each of several discs, as appropriate.

Source and object modules are frequently too short to make efficient use of disc space when they and cataloged as files. To make more efficient use of disc space, these modules may be cataloged as members of a library file. A library file module may be designated as follows:

DSC2. PGMDEV. SOURCE (PROG1)

File SOURCE in partition PGMDEV of disc DSC2 is a library file. The specified module in the library file is PROG1. The object module corresponding to this source module could be designated as follows:

DSC2. PGMDEV.OBJECT (PROG1)

Files DSC2.PGMDEV.SOURCE and DSC2.PGMDEV.OBJECT are defined as library files.

SDSLIB supports commands for defining files and structures of discs, for manipulating files, and for deleting files, partitions, or entire structures. In addition, compressed source libraries (blanks suppressed) can be created and stored using the librarian.

4.5.4 TX990 MEMORY RESIDENT OPERATING SYSTEM. TX990 is a multitask operating system for use in dedicated non-disc systems. TX990 is furnished as a set of linkable object modules for invariant portions of the end system and source modules for variable portions. System generation is performed by assembly and linkage of the TX990 modules and the application program tasks. This allows TX990 to be easily customized for specific applications where a premium is placed on minimum memory requirements. Access to a DX10 software development system is recommended for this procedure. The resulting application system can be transported to the target computer via magnetic tape cassettes. Refer to discussion of Operating Systems earlier in this section for complete details about TX990.

4.5.5 COMMUNICATIONS SOFTWARE. The 990 Family of Computers offers communications software to control either synchronous or asynchronous data transmission. Either communications software package can operate in conjunction with the TX990 or the DX10 operating environment. The hardware modules supported are described in Section V of this manual.

4.5.5.1 Asynchronous Communications Package. This software package supports asynchronous data communications links at transmission rates up to 9600 baud in half-duplex mode. In addition, the software allows interface to auto-answer/auto-call devices and provides fundamental error checking. The asynchronous communications package requires approximately 2.5K words of memory in addition to the operating system requirements, and is available on either magnetic tape cassette or punched cards. The communications package requires the following minimum hardware:

- 990/4 or 990/10 computer system
- 990 Asynchronous communications kit, or asynchronous Bell data set interface kit plus Bell data set.

In addition, the package supports the following optional hardware:

- 990 auto-call kits (pulse or touch-tone).

4.5.5.2 Synchronous Communications Package. This software package supports synchronous data communications transfers between two members of the 990 Computer Family. The interface allows data transfer at rates up to 9600 bits per second, supports both auto-call and auto-answer devices, and provides fundamental error checking. The synchronous communications package requires 2.5K words of memory in addition to the operating system requirements, and is available on magnetic tape cassette or punched cards. The communications package requires the following minimum hardware:

- 990/4 or 990/10 computer system
- 990 synchronous communications kit, or synchronous Bell data set interface kit plus synchronous Bell data set

In addition, the package supports the following optional hardware:

- 990 auto-call kits (pulse or touch-tone).

4.6 IBM SYSTEM/3X0 CROSS SUPPORT SYSTEM

The Cross Support System is available on several worldwide timesharing services or may be installed on the user's IBM System/3X0. It consists of a Cross Assembler and a TMS9900 Simulator. The Cross Support System allows the user to assemble programs for any of the 990 Family of Computers, and to simulate the instructions of the TMS9900 instruction set in debugging those programs.

4.6.1 CROSS ASSEMBLER. The Cross Assembler is a two-pass assembler that is compatible with the TMS9900 prototyping system assembler (PX9ASM). All operations are available in the Cross Assembler. One additional directive for specifying output options is also provided. In addition to the object code output and the assembly listing the Cross Assembler optionally prints a symbol table. It may also provide a cross-reference listing and/or an object listing. Restrictions on forward references for the Cross Assembler are consistent with two-pass operation. Source code written for the Cross Assembler may be assembled on SDSMAC, and also on PX9ASM if forward reference restrictions are met. Source code written for PX9ASM may be assembled on the Cross Assembler.

4.6.2 TMS9900 SIMULATOR. The TMS9900 Simulator simulates execution of programs for the 990 Family of Computers that include only the instructions in the TMS9900 instruction set. Simulator control statements allow the user to debug the program as it is simulated.

The Simulator supports a set of loader commands that allow the user to specify load and entry points, and to specify object files to be loaded. Object modules may be loaded along with the loader commands.

Simulator control statements allow the user to:

- Specify memory size, memory timing, and microprocessor timing for the simulation
- Define CRU input data for the simulation, and whether characters in the input are to be supplied to the simulator as ASCII characters or as EBCDIC characters
- Define output lines for the simulation
- Set starting and stopping points for simulation
- Determine addresses at which breakpoints are to occur if memory is accessed or altered
- Define locations to be traced
- Request display of contents of memory locations, CPU registers, or CRU lines
- Select values to be placed in memory locations or CPU registers
- Specify simulation in batch or interactive mode.

The Simulator provides simulation of programs with a powerful debugging capability that operates as a simulation proceeds. Optionally, the simulator interfaces with the Manufacturing Output Utility Program to provide files to control manufacture of ROMs or programming of PROMs. The Simulator also interfaces using cassettes with the Prototyping System to debug the program further on a Model 990/4 microcomputer.

4.7 990 SYSTEM REQUIREMENTS FOR SOFTWARE

The software package descriptions in this section specify minimum hardware requirements and software system requirements that must be available before successfully using each of the packages. These requirements ensure that the user will be able to modify (if desired), link, load, initialize and execute programs with the system and still have storage space available for user programs. To aid in determining the requirements for a particular system, table 4-1 summarizes the software requirements for each of the software packages. In addition, table 4-2 summarizes the available and required hardware for use with each of the software packages.

Because the 990 Family of Computers adapts easily to many diverse applications, the user may want to incorporate special equipment in a 990 system to achieve a required purpose. To aid the customer in interfacing his equipment to the Computer Family,

Texas Instruments maintains a staff of experienced hardware design engineers and software programmers to implement efficient solutions to interfacing and system development problems. This service is available to 990 Computer customers on an individual quotation basis. Consult your local Texas Instruments sales office for information.

Table 4-1. 990 Software Packages - Software Requirements

Software Package	Software Prerequisite
Prototyping System	No restriction for object programs Requires 990/10 system to assemble/link source
990-733 ASR System Software	No restriction for object programs Requires 990/10 system to assemble/link source
990/10 Disc System	No restriction for object programs Requires 990/10 system to assemble/link source
TX990	990/10 development system recommended for creating applications programs and incorporating TX990 before installation in target computer
DX10 (Source)	Requires 990/10 Development System
FORTRAN IV	Requires 990/10 Disc System
COBOL	Requires 990/10 Disc System
BASIC	Requires 990/10 Disc System
Cross Support	Requires IBM System/370 or System/360
Communications	Requires 990/10 Development System for system generation. Requires TX990 or DX10 for operation.

Table 4-2. 990 Software Packages - Hardware Configurations

		Prototype System	990-733 ASR System	990/10 Disc System	TX990 GPOS	Communications Sync.	Communications Async.
Computer:	990/4	Yes	Yes	No	Yes	Yes	Yes
	990/10	No	Yes	Yes	Yes	Yes	Yes
Dynamic RAM:	Min Words	8K	8K	24K	4K	2.5K	2.5K
	Max Words	28K	32K	1M	32K	–	–
Write Protect		Req	Opt		No	–	–
Memory Mapping		No	No	Req	No	–	–
733 ASR Kit		Req	Req	Req	Req	–	–
733 ROM Loader		Req	Req	Req	Req	–	–
Programmer Panel		Req	Req	Req	Opt	–	–
PROM Programmer		Opt	Opt	No	No	–	–
733 KSR		No	No	Opt	Opt	–	–
913 VDT		No	No	Req	Opt	–	–
Card Reader		No	No	Opt	Opt	–	–
Line Printer		No	No	Opt	Opt	–	–
Floppy Disc		No	No	Opt	Opt	–	–
Moving Head Disc		No	No	Req	No	–	–
Async. Comm. Kit		No	No	No	No	No	Req
Sync. Comm. Kit		No	No	No	No	Req	No
990 Auto-call (Pulse)		No	No	No	No	Opt	Opt
990 Auto-call (Touch Tone)		No	No	No	No	Opt	Opt

Key: Yes = Available for this CPU
No = Not supported
– = Not applicable
Req = Required
Opt = Optional

SECTION V

PERIPHERALS AND INPUT/OUTPUT INTERFACES

5.1 GENERAL

Texas Instruments provides a variety of peripheral equipment for the 990 Computer Family to satisfy user application requirements for interactive communications, punched card input, hard copy output, remote communications, and mass storage. A selection of modules is available to assist the 990 user in implementing interfaces for other peripheral devices. This section provides primary characteristics and physical descriptions of these input/output options. These are discussed in the following functional categories:

- Terminals - interactive input/output units for operator/machine communications.
- Communications - modules and kits to support user applications requiring communications to remote terminal devices and distant computer systems.
- Mass storage - storage units with removable/non-removable media for online storage and rapid access of user data and program files, with the removable media serving as either transportable storage or offline storage.
- General purpose hardcopy peripherals - line printers and card reader for hardcopy output and punched card input.
- General purpose interfaces - modules with EIA and TTL interface levels.
- Read only memory implementation device - programming module to assist the user in implementing PROM and EROM memory for 990 computers.

The interactive data terminals include keyboard entry devices and either printed or displayed output. The available terminals are the Model 913 Video Display Terminal, the Model 733 ASR Twin Cassette Data Terminal, and the Model 733 KSR Data Terminal.

The communications modules allow for communication with either asynchronous or synchronous modem systems. Available modules include communication interface modules, 1200 and 2400 baud modems, and automatic calling units.

Mass storage functions are accomplished by disc systems, with a choice of systems to meet different performance and cost requirements. A moving head disc is available for high-speed, large-volume data storage and retrieval, and a floppy disc is available for

low-cost data storage and retrieval. The disc systems used with the 990 Computer are the Model 31 Moving Head Disc (removable cartridge) Unit, the Model 32 Moving Head Disc (non-removable cartridge) Unit, and the Floppy Disc System.

The medium-speed and low-speed peripheral devices for input/output batch processing operations are line printers and a card reader. The impact line printer in the 990 Computer system may be either medium-speed or low-speed. The devices available are the Model 306 Line Printer, the Model 588 Line Printer, and the Model 804 Card Reader.

This section will provide the 990 user information on physical size and description, operator controls and indicators, input/output formats, electrical characteristics, and 990 interface/controller description. The following section, "System Design and Configuration," will provide design guidelines for configuring these devices into a 990 system.

5.2 TERMINALS

The primary interface between the human operator and the computer is typically a terminal device with keyboard and either a video display screen or printer. Texas Instruments "Silent 700" ASR/KSR Data Terminals combine silent electronic printing with smooth, quiet magnetic tape cassettes to offer users a powerful alternative to conventional ASR paper-tape teletypewriters. The Model 913 Video Display Terminals provide a versatile, interactive, cluster terminal capability to the 990 family. The design of these terminals reflect strong consideration towards the human operator in keyboard layout, display and printed output readability, as well as an overall pleasing packaging well suited for the office environment.

The following paragraphs discuss the terminal models available for direct interfacing to the 990 Computer Family. These terminals, when configured with the other 990 Computer hardware and peripherals, will provide a cost-effective solution for processing information across a room or across a country. Texas Instruments also provides a wide selection of "Silent 700" Data Terminals to function as remote data entry and response devices over long distance dial-up or multi-drop communication lines. Contact your local Texas Instruments sales representative for more information on these terminals.

5.2.1 MODEL 913 VIDEO DISPLAY TERMINAL. Texas Instruments Model 913 Video Display Terminal (VDT), shown in figure 5-1, is an attractive table top terminal ideally suited for a variety of applications. This versatile, interactive, terminal may be utilized as a local computer system terminal or arranged in a cluster configuration centered around the 990/4 as a dedicated controller. Blending equally well into either the decor of the office environment or in the rugged no-nonsense assembly line area, the 913 VDT offers the 990 user an excellent solution to multiple information handling requirements.

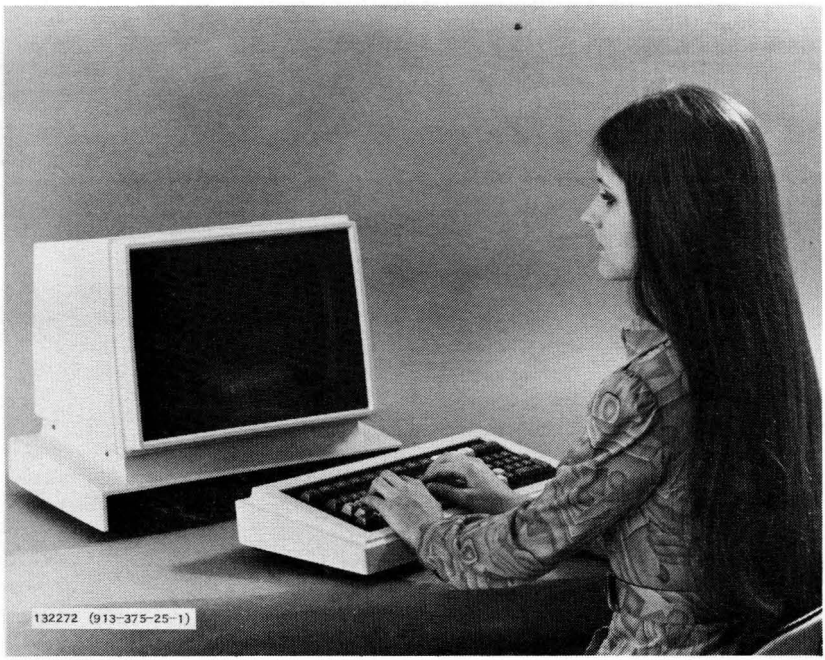


Figure 5-1. Model 913 Video Display Terminal

The bright, easily readable alphanumeric characters will provide operators with rapid access to critical information. The 913 VDT screen is shown in figure 5-2. The keyboard layout (figure 5-3) is designed to function equally well for single-handed data and command entry, and also for data base manipulation and updating. The 990 user will discover the 913 VDT can be used for in-house time sharing, data entry, inventory control, and production control and planning. When implemented in a remote cluster configuration the 990/913 system will serve many different inquiry response applications. These include reservations, inventory control for remotely located warehouses, data base inquiry and update for parts distributors, banking and other financial industries, and for transportation scheduling and control.

The 913 video display is housed in an eye-pleasing, soft-white console with a 12-inch diagonal cathode ray tube featuring a non-reflective, high-resolution screen. The display video driving circuitry consists of all solid-state components mounted on a single printed-circuit board next to the tube. Natural convection cooling, without a fan, ensures quiet operation well-suited for the office environment. Character generation, video generation, buffer memory, and other control circuitry are on the full size controller board which mounts in the 990 Computer chassis. The 913 display

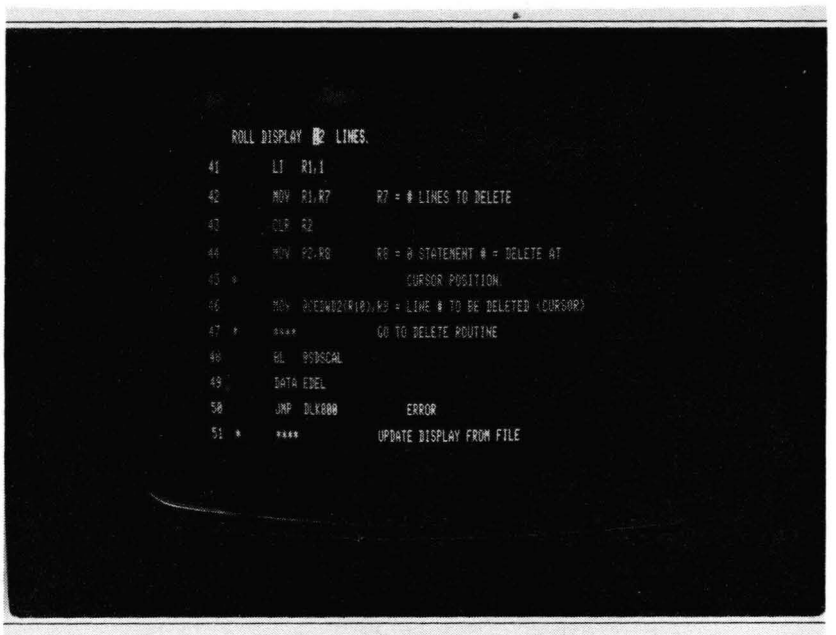


Figure 5-2. Model 913 Video Display

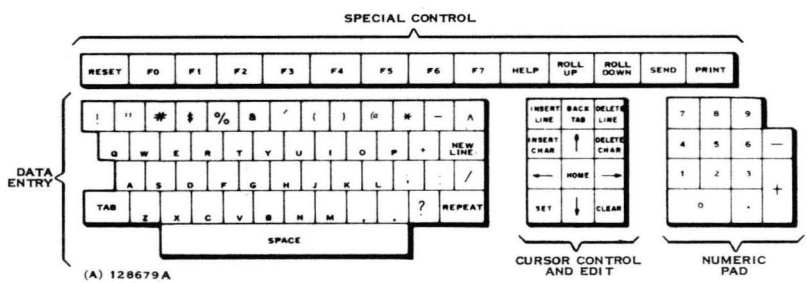


Figure 5-3. Model 913 Video Display Terminal Keyboard

can be remotely located up to 2000 feet from the controller and computer. The 913 keyboard connects to the 913 video display through a standard 6-foot cable. This feature provides versatility in arrangement with the keyboard locatable in various positions around the display or neatly snuggled in front of the console. The remote cabling feature ensures the 913 Video Display Terminal's suitability for scattered applications throughout a typical office building, factory, or warehouse. The keyboard has "double shot" molded, non-glare-finish plastic keys functionally arranged to ensure ease of operation. Five cursor control functions, three format keys, and fourteen general, user definable, function keys provide a programmable versatility for multiple applications. The numeric keys are arranged into the industry standard 10-key pad to accommodate rapid entry of accounting, financial, and other forms of numeric data.

The following paragraphs provide a summary of the specifications on the video display and keyboard.

5.2.1.1 Operating Controls, Display and Keyboard. The 913 VDT features several controls which enhance operation and increase operator convenience. Included are the following:

Display Controls. The display controls, which are located at the top left rear corner of the display package, include vertical hold, horizontal hold, and audio (beeper) volume.

Cursor. Cursor manipulation is provided by conveniently located keygrouping on the front panel. Cursor controls include UP, DOWN, LEFT, RIGHT, and HOME (top upper left of screen).

Format. Format keys include TAB, SPACE and REPEAT keys.

Control. The 913 Video Display Terminal also includes 14 additional keys intended for user definition.

Volume. The volume control adjusts the volume of the 913 alarm.

5.2.1.2 Display Features. The display characters are formed by a 5 X 7 dot matrix. The screen capacity is 12 horizontal lines of 80 characters per line. The characters available on the 913 VDT consist of 57 upper case display characters and 32 control characters. The character set is standard ASCII code and is shown in table 5-1.

Model 913 VDT is a flexible machine and has the following capabilities:

Instant Display. The screen on the 913 VDT can be filled in less than 20 milliseconds.

Programmable Cursor Positioning. The cursor can be positioned by program control.

Table 5-1. Character Set as Read From Refresh Memory and Displayed on the CRT Screen

ASCII Code	Keyboard Character	Refresh Memory Character Read	Display Character
02	HOME	42	B
08	←	48	H
09	TAB	49	I
0A	↓	4A	J
0D	NEW LINE	4D	M
1A	↑	5A	Z
1C	→	5C	\
20	space	20	space
23	:	23	#
24	+	24	$
26	/	26	&
2B	+	2B	+
2C	,	2C	,
2D	-	2D	-
2E	.	2E	.
2F	?	2F	/
30	0	30	0
31	1	31	1
32	2	32	2
33	3	33	3
34	4	34	4
35	5	35	5
36	6	36	6
37	7	37	7
38	8	38	8
39	9	39	9
3A	;	3A	:
3B	SET	3B	;
3C	CLEAR	3C	<
3D	INSERT CHAR	3D	=
3E	DELETE CHAR	3E	>
3F	^	3F	?
41	A	41	A

Table 5-1. Character Set as Read From Refresh Memory and Displayed on the CRT Screen (Continued)

ASCII Code	Keyboard Character	Refresh Memory Character Read	Display Character
42	B	42	B
43	C	43	C
44	D	44	D
45	E	45	E
46	F	46	F
47	G	47	G
48	H	48	H
49	I	49	I
4A	J	4A	J
4B	K	4B	K
4C	L	4C	L
4D	M	4D	M
4E	N	4E	N
4F	O	4F	O
50	P	50	P
51	Q	51	Q
52	R	52	R
53	S	53	S
54	T	54	T
55	U	55	U
56	V	56	V
57	W	57	W
58	X	58	X
59	Y	59	Y
5A	Z	5A	Z
5B	INSERT LINE	5B	[
5C	BACK TAB	5C	\
5D	DELETE LINE	5D	]
5E	REPEAT	5E	^
60	@	20	space
61	!	21	!
62	"	22	"
63	#	23	#

Table 5-1. Character Set as Read From Refresh Memory and Displayed on the CRT Screen (Continued)

ASCII Code	Keyboard Character	Refresh Memory Character Read	Display Character
64	$	24	$
65	%	25	%
66	&	26	&
67	'	27	'
68	(	28	(
69	)	29	)
6A	*	2A	*
6B	-	2B	+
70	RESET	30	0
71	F0	31	1
72	F1	32	2
73	F2	33	3
74	F3	34	4
75	F4	35	5
76	F5	36	6
77	F6	37	7
78	F7	38	8
79	HELP	39	9
7A	ROLL UP	3A	:
7B	ROLL DOWN	3B	;
7C	SEND	3C	<
7D	PRINT	3D	=

Processor Control Functions. The 913 VDT has a capability of responding to certain control functions from the 990 computer.

Screen Refresh. The screen on the 913 refreshes at 60 frames per second from memory located on the controller printed circuit board. Refresh for 50 Hz operation can be selected on the controller board.

Programmable Editing and Protective Display Features. The fully programmable 913 VDT allows user to define many added features as required by his application.

5.2.1.3 Power Requirements. The electrical power requirements for the 913 VDT are:

- Display - 115 Vac, 50-60 Hz, 1.13A
- Controller - +12 Vdc at 0.1A, -12 Vdc at 0.14A +5 Vdc at 2.7A (The controller power is supplied by the CPU chassis power supply.)

5.2.1.4 Space Requirements. Space requirements for the 913 VDT are:

- Display - 15.5 in. (H) X 12.8 in. (D) X 19.0 in. (W)
- Keyboard - 4.0 in. (H) X 9.0 in. (D) X 18.75 in. (W).

5.2.1.5 Documentation. Additional information about the 913 VDT is contained in the *Model 990 Computer Model 913 CRT Display Terminal Installation and Operation,* Manual No. 943457-9701.

5.2.2 MODEL 733 "SILENT 700" DATA TERMINALS. The Texas Instruments Model 733 "Silent 700" data terminals are one of the most popular terminals available in the data processing industry. Second in sales only to the teletype terminal, these terminals can be used as a system keyboard/printer and principle input/output device for the Model 990 Computer Family. Data can be input to the machine via the keyboard, printed output from the CPU appears on this terminal, and the magnetic tape cassettes can be used as input/output media for program and data storage.

The Model 733 terminals (figures 5-4 and 5-5) are modular-designed data terminals with keyboard, printer mechanism, transmit/receive electronics and associated controls, and a two-station cassette tape input/output unit (ASR terminal). These terminals use ASCII code for the keyboard and printer mechanisms and provide the following features:

- The standard keyboard permits manual typing operations and transmission of printable upper-case characters and operational codes in ASCII code.
- The printer mechanism features a solid-state printhead, paper handling mechanics, and printhead movement devices.
- The transmit/receive electronics control communications between the Model 990 Computer and the data terminal.
- (ASR terminal) - The record section controls recording of local (from the keyboard) or remote (from the computer) messages on magnetic-tape cassettes.
- (ASR terminal) - The playback section controls playback of messages that were previously recorded on magnetic-tape cassettes for local use (printed by the printer) and/or transmission to the computer.

Figure 5-4. Model 733 ASR Data Terminal

5.2.2.1 Controls, Indicators, and Keyboard Characters. The following controls and indicators are used to control the data terminal and the exchange of information between the data terminal and the Model 990 Computer. Texas Instruments makes available the *Terminal User's Guide,* Manual No. 945259-9701, for more detailed information.

Controls. The controls for the terminal are:

- Terminal - Power-On/Off, Speed-Low/Medium/High, Transmission Mode-Half/Full Duplex, Parity Select-Odd/Even/Mark, Terminal Status-On-Line/Off-Line, and Speed Status-Low/High (with 1200 baud option).
- Printer - Printer Contrast, Line Space-Single/Double, Control-Line/Off/Local.
- Keyboard - REPEAT, PAPER ADVANCE, LINE FEED, HERE IS, BREAK, CARRIAGE RETURN, TAPE, TAPE keys, Control-Line/Off/Local.
- ASR only - Cassette 1/Cassette 2 - Control-Record/Playback, Tape-Rewind/Stop and Load/Fast Forward/Stop.

- Record Control - Control-Line/Off/Local and On/Off, Tape-Erase/Print, Tape Format-Line/Continuous.

- Playback Control - Control-Line/Off/Local, Tape-Continuous/Start/Stop, Block Forward/Reverse, and Character Forward.

Indicators. The indicators listed below indicate the status of the terminal:

- Terminal - Power On light.

- ASR only - Cassette 1/Cassette 2 - READY, END, RECORD, and PLAYBACK lights.

- Record Control - ON light and 8-bit CHARACTER display lights.

- Playback Control - ON and ERROR lights.

Keyboard Character Codes. The keyboard character codes generated are shown in table 5-2. These codes conform to the standard ASCII character set shown in figure 5-6.

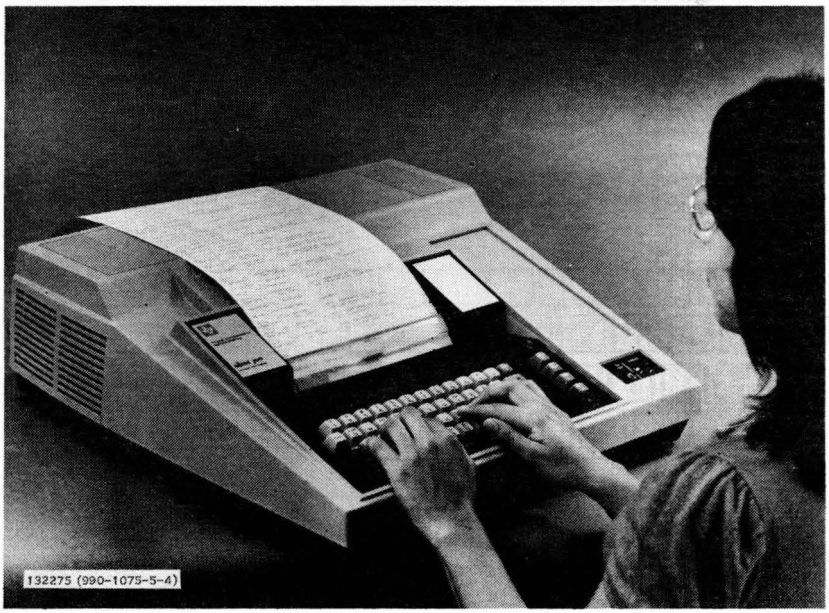


Figure 5-5. KSR Data Terminal

Table 5-2. Keyboard Character Codes

$b_4\ b_3\ b_2\ b_1$	$b_7 \rightarrow 0$, $b_6 \rightarrow 0$, $b_5 \rightarrow 0$	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
0 0 0 0		NUL	DLE	SPACE	0	@	P	`	p
0 0 0 1		SOH	DC1	!	1	A	Q	a	q
0 0 1 0		STX	DC2	"	2	B	R	b	r
0 0 1 1		ETX	DC3	#	3	C	S	c	s
0 1 0 0		EOT	DC4	$	4	D	T	d	t
0 1 0 1		ENQ	NAK	%	5	E	U	e	u
0 1 1 0		ACK	SYN	&	6	F	V	f	v
0 1 1 1		BEL	ETB	'	7	G	W	g	w
1 0 0 0		BS	CAN	(	8	H	X	h	x
1 0 0 1		HT	EM	)	9	I	Y	i	y
1 0 1 0		LF	SUB	*	:	J	Z	j	z
1 0 1 1		VT	ESC	+	;	K	[	k	{
1 1 0 0		FF	FS	,	<	L	\	l	¦
1 1 0 1		CR	GS	-	=	M	]	m	}
1 1 1 0		SO	RS	.	>	N	^	n	~
1 1 1 1		SI	US	/	?	O	_	o	DEL

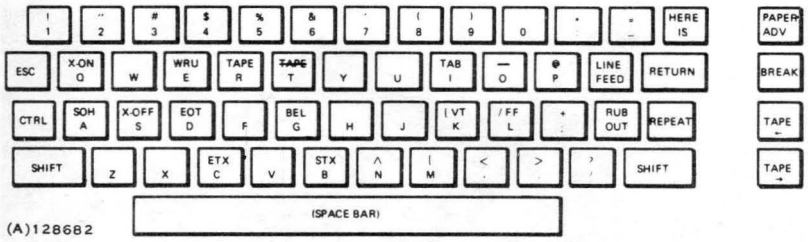
PRINTABLE CHARACTER

PRINTER CONTROL CHARACTER

AUXILIARY DEVICE CONTROL CHARACTER

CODES GENERATED BY KEYBOARD, BUT NO ACTION TAKEN

(A)128790



(A)128682

Figure 5-6. Standard Keyboard Layout

The optional full ASCII keyboard contains 95 printable characters and 33 control characters arranged as shown in figure 5-7.

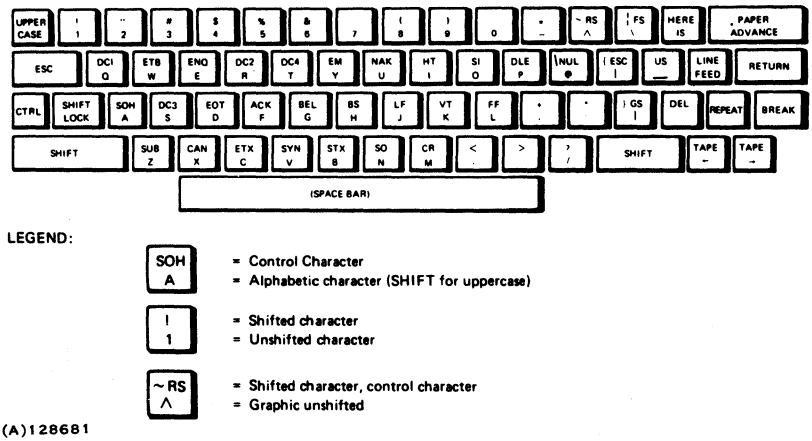


Figure 5-7. Optional Full Keyboard Layout

The terminals permit the computer operator or programmer to input/output system data including developmental software, application programming or control information for the 990 family of computers. This input/output data can be sent directly to the printer, from the keyboard, or with the ASR terminal, to/from magnetic tape cassettes.

5.2.2.2 Terminal Specifications. A summary of specifications for the Model 733 terminals is contained in the following paragraphs and referenced tables.

Data Format and Transmission. Data is routed within the terminal through a single-data bus. The data is sent serially as an 8-bit character. The 8 bits include a 7-bit ASCII character and for an ASR terminal, an end-of-block indicator. Transmission speed is 1200 bits per second.

Printer. The printer specifications are listed in table 5-3.

Communication Line Interface. The Communication Line Interface conforms to EIA standard RS232C. The terminal can receive, without error, signals with mark and space distortion of up to 45 percent. For error-free reception, at any speed, the minimum stop bit time is 0.6 of a normal bit time.

Tape Transport (ASR Terminal). The tape transport specifications are shown in table 5-4.

Table 5-3. Terminal Printer Specifications

Characteristic	Specification/Description
Printing method	5 X 7 dot matrix, electronically heated, on heat-sensitive paper
Line length	7.9 inches, 80 characters
Character spacing	0.1 inch, character center to center
Line spacing	Six or three lines per inch (single or double spaced)
Paper (TI part number 213714-0001 or 953167-0001)	Roll, 8.5 inches wide by 3.625 inches maximum diameter (300 feet), heat-sensitive
Platen	Friction feed
Carriage return time	195 milliseconds maximum
Line feed time	33 milliseconds maximum (single space), 66 milliseconds maximum (double space)
Audible alarm time	250 (±50) milliseconds on receipt of the BEL character
Printable characters	95
Carriage return and line feed (CR/LF)	Automatic at column 81, no code is transmitted
Visibility of printed lines	At least 50 previous lines of print (including line and character being printed) are visible and unobstructed
Print contrast	Operator adjustable

Power Requirements. The electrical power requirements for the data terminal are:

- Frequency - Normal operation with primary input frequencies in the range of 48 to 62 Hz.
- Voltage - 115 (+10%, -15%) Vac/rms.
- Power - Required primary input power at maximum rated voltage is 200 VA.

Table 5-4. Tape Transport Specifications

Characteristic	Specification/Description
Recording speed	8 inches per second
Recording method	Phase-encoding
Recording density	800 bits per inch (1600 flux changes per inch)
Rewind time	60 seconds maximum
Playback speed	120 characters per second (to communication line or printer) or 250 characters per second (duplication)
Tape drive	Capstan drive for recording or playback
Error rate	One in 10^6 maximum, using certified cassette tapes and proper head cleaning procedures; one in 10^7 typical
Interchangeability	Any tape recorded on any 733 ASR transport operating within specifications may be read on any other 733 ASR of the same model operating within specifications
Sensors	EOT, BOT, cassette in place, write tab, and transport door closed
Media	Improved Philips type cassette containing 275 to 300 feet of digital grade magnetic tape with approximately 20 inches of transparent tape joined to each end.

Space Requirements. The space requirements for the terminals are defined in table 5-5.

5.2.2.3 Documentation. Additional information about the terminals is contained in the *Model 990 Computer Model 733 ASR/KSR Terminal Installation and Operation Manual,* Manual Number 945259-9701.

5.3 DATA COMMUNICATIONS

The low cost, high performance capabilities of microcomputers and minicomputers enhance their application in large distributed networks for information processing. This distributed processing concept provides rapid access and update of information for numerous application areas such as reservations, transportation control and scheduling, financial transactions, inter-warehouse inventory control and order entry,

Table 5-5. Data Terminal Dimensions

Characteristic	Specification (inches)
Height	
733 ASR	14.62
733 KSR	6.85
Width	21.18
Depth	19.50

and management information systems. This movement of data across town or across the country requires data communications. The complexity and performance of communications equipment ranges from the relative slow teletypewriter at 10 characters per second up to satellite links at 50 megabits (6 million characters per second). The 990 computer family communication options are designed for the majority of applications which utilize the most available and least expensive approach, the telephone network.

The telephone network lines are used in two primary ways, the dialed telephone circuit and a dedicated leased line. The dialed telephone network is limited in speed by the worst expected case, typically 120 characters to 300 characters per second. The dedicated lease line can extend this capability up to 1200 characters per second with special "line conditioning" and use of multiple "lines."

5.3.1 990 COMMUNICATIONS. The 990 communication options provide the necessary communications hardware and software to support data communications up to 2400 baud (300 characters per second) over dialup lines and up to 9600 baud (1200 characters per second) over leased line. The standard 990 communication hardware will support user-developed software for speeds up to 9600 baud asynchronous and 48,000 bits per second synchronous. The context switch feature, for interrupt processing, of the 990 architecture ensures minimum processing overhead for these communication speeds.

5.3.1.1 Asynchronous Communications Asynchronous communications offer the 990 user the most economical means of communications. This method uses a start bit, an 8-bit character, and a stop bit. This approach allows the use of less complicated communication controllers and modems and is typically used for data communications with remote terminals. The selectable speeds include 75, 110, 150, 300, 1200, 2400, 4800, or 9600 baud (≈bits per second). The 990 asynchronous communications modem operates at 1200 baud.

5.3.1.2 Synchronous Communications. Synchronous communications are more commonly used between computers and offer the user a more efficient, and reliable means of communications at higher speeds. The 990 synchronous communications

modem operates at 2400 baud where synchronous (8-bit) characters are transmitted in a continuous stream. Each message block starts with a series of unique sync characters. When the receiving equipment acknowledges that it has detected the sync code, a continuous stream of data characters is transmitted. The receiving equipment must divide the stream into 8-bit segments to derive meaningful information.

5.3.2 990 COMMUNICATIONS MODULES. There are six different communications modules for the 990 computers. These modules are plug-in printed circuit boards which require one half-slot each in the 990 chassis. The following describes the features of each module.

5.3.2.1 Asynchronous Communications Interface Module. This module provides a standard RS232 EIA full duplex asynchronous interface featuring selectable baud rates including: 75, 110, 150, 300, 1200, 2400, 4800, and 9600 baud. Data pattern selectable features include: 5, 6, 7, or 8-bit characters with odd, even, or no parity, and a selection of one or two stop bits. A selectable loopback is provided to facilitate fault isolation to the module level.

This module serves as an interface to the 990 for the following:

- 1200 baud 990 asynchronous modem
- Terminal device having EIA standard RS232 interface
- Bell data sets 103A or F, or equivalent
- Bell data sets 202C or D or equivalent.

This module will support auto answer for both the 990 asynchronous modem and the above listed Bell data sets.

5.3.2.2 990 Asynchronous Modem. This modem provides 1200 baud asynchronous communications and is compatible with the Bell data set 202C. The modem is mounted on a standard half card and plugs into the 990 chassis. Top edge connectors are provided for three cables: one cable to the 990 communications module described in the previous paragraph, the second to an auto calling module, and the third to an external Direct Access Arrangement (DAA) device.

5.3.2.3 Synchronous Communications Interface Module. The synchronous communications interface module functions as an interface to the 990 computer for any of the following:

- 990 synchronous communications modem
- Internally clocked Bell 201 data set, or equivalent
- Internally clocked Bell 208 data set, or equivalent.

The data format is 8 bits per character. Module features programmable sync register, selectable parity, 500 millisecond timer, new sync pulse generation, and transparent mode operation. Selectable loopback allows fault isolation to the module level.

The module provides auto answer capability when used with the 990 Sync Modem and either Bell data set 201 or 208B.

5.3.2.4 990 Synchronous Modem. This modem is Bell 201 equivalent and provides the 2400 baud transmission rate for the 990 synchronous communications interface module. Mounted on the standard half-size card, this module plugs into in a half-slot in the 990 chassis with top edge connectors for an external DAA device, interface module, and auto calling module.

5.3.2.5 Pulse Auto Calling Module. This module provides auto calling for pulse type telephone switching circuitry. This plug-in module requires one half-slot in 990 chassis and can be used with either of the 990 asynchronous or synchronous communication modems described in 5.3.2.2 and 5.3.2.4.

5.3.2.6 "Touch Tone" Trademark Auto Calling Module. This module provides auto calling for touch tone type telephone switching circuitry.

5.3.3 990 COMMUNICATIONS SYSTEM CONFIGURATION. Standard cabling is furnished in appropriate kits for interfacing the above modules and either external DAA devices or Bell data sets. Cabling between the 990 modules are designed for intr-chassis location of modules as depicted in figure 5-8.

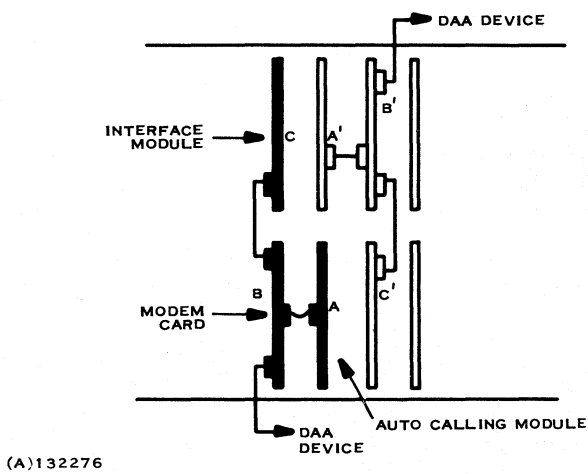


Figure 5-8. Intra-Chassis Location of Modules

Cards A', B', and C' allow compact placement of modules within the 990 chassis.

5.3.4 DOCUMENTATION. Additional information is available in either the *Model 990 Computer Synchronous Modem System Installation and Operation Manual,* Manual No. 945263-9701 or the *Model 990 Computer Asynchronous Modem System Installation and Operation Manual*, Manual No. 945400-9701.

5.4 MASS STORAGE

The implementation of minicomputer technology in management information processing, distributed processing networks, and interactive data base management systems requires on-line rapid random access and storage of millions of characters. The disc presents an economical solution to this requirement. Texas Instruments provides the 990 user two primary options to satisfy his mass storage needs. The Model DS31 and Model DS32 provide up to 12.4 million characters of storage with only the ease of removability of medium denoting the basic difference between these two models. The second primary option is the floppy disc, or "diskette." The floppy disc provides an effective response to the 990 user who requires a really low-cost random access storage device. The diskette provides the 990 user a transportable medium capable of storing up to 242 thousand characters on a flexible, low cost 8-inch Mylar disc.

The use of disc storage provides the 990 system designer the following major functional assets:

- Access within milliseconds to randomly selected data from multiple files
- Transfer of data to/from disc at thousands of characters per second
- Economical storage of large programs and data files
- Compact, low cost medium for offline storage and backup of data and programs
- Transportable, machine-to-machine compatible medium
- System resource capable of providing logical behavior independent of mechanical behavior.

The latter asset means simply that a disc unit can perform the logical functions of several devices, and that it can do this in one of several operational modes. The disc can be used in these modes as an intermediate storage of data and programs to be processed. This intermediate storage is arranged and processed by application programs in the same logical procedure as magnetic tape, card reader, printers, and other peripheral devices. The disc also acts as an extension of memory for program storage allowing the system software "functioning as supervisor" to allocate computer memory to application programs on time and task priorities. The DX10 Disc Executive provides these functions for the 990 user. Refer to Section IV for additional specifications on 990 system software.

The following paragraphs provide descriptions and specifications on the two primary storage devices available for the 990.

5.4.1 MODEL DS31 AND DS32 MOVING HEAD DISC UNITS. The Model DS31 and DS32 moving head disc units, shown in figure 5-9, provide a maximum of 1.55 million 16-bit word storage per disc. The primary disc controller will accommodate three additional drives to provide a total capacity of 6.2 million words storage. The Model DS31 and DS32 are functionally and electrically identical. Their essential difference is the ease of removing the Type 2315 single disc cartridge and the operator controls. The Model DS31 provides a hinged front door and load/run controls to facilitate quick removal and exchange of disc cartridge. The Model DS32 requires the use of a screw driver to remove front panel and typically is used where the cartridge seldom requires removal. In a multiple disc system, two DS31 removable disc units are recommended while the remaining units may be the lower cost DS32 units.

Figure 5-9. Moving-Head Disc System

Principal features of the Model DS31 and DS32 discs are as follows:

- Automatic track switching, over head and cylinder boundaries
- Automatic verification of track, sector and format on every disk access. All data is error checked with CRC polynomial
- Variable record formats from one sector to full track
- Four drives per controller with overlapped seek capability
- Individual unit interrupt masking
- Single card controller for high mean time between failures and quick mean-time-to-repair
- Efficient disc formatting for high percentage storage utilization
- Maximum storage capacity of 1.55 million words with full track records
- Controller diagnostic capability with dead fault and busy LED indicators
- System parameter list readable with command
- Each recording surface of the cartridge contains 203 tracks with 24 sectors per track, for a total of 9744 sectors on two surfaces. A track can be formatted for any number of physical data records from 1 to 24 per track; the controller can write across gaps to increase storage efficiency to 99%. At one sector per record, the storage capacity of one cartridge is 1,403,136 words, and at 24 sectors per record is 1,552,544 words
- The MHD controller occupies a single, full-width circuit card that interfaces the disc drive with the 990 TILINE data bus. The controller is microprocessor-based to achieve a high degree of integration. A single controller can control up to four disc drives with overlapped seeks.

The summary for DS31 and DS32 specifications is provided in table 5-6.

Table 5-6. Model DS31 and Model DS32 Moving Head Disc Unit Specifications

Characteristic	Specification/Description
Data Storage and Retrieval	
Storage Capacity	3 million bytes, unformatted
Storage Capacity per Track	60,000 bits, unformatted
Track Density	100 tracks per inch
Bit Density	2200 bits per inch, innermost track
Recording Surfaces per Disc	2
Cylinders per Disc Drive	203
Tracks per Cylinder	2
Data Transfer Rate	1.56 million bits per second
	95,700 words per second, one disc unit connected to controller
Disc Rotation	
Rotation Speed	1500 rpm ±1%
Rotational Latency	
Maximum	40 milliseconds ±1%
Average	20.0 milliseconds
Heads	
Heads per Disc Drive	2
Positioning Method	Servo linear motor
Maximum Head Positioning Time (Including Settling Time)	135 milliseconds
	70 milliseconds average, nonadjacent cylinders
	15 milliseconds, track-to-track, average
	15 milliseconds, adjacent cylinders
Data Access Time	70 milliseconds, average
Electrical Characteristics	
Disc Drive Input Line Power	105 to 240 Vac, 47 to 63 Hz, power supply, power supply for 2 disc drive units 4A at 115 Vac, 60 Hz.
Disc Controller Input Power	+5 Vdc at 4.8 A

Table 5-6. Model DS31 and Model DS32 Moving Head Disc Unit Specifications (Continued)

DISC COMMAND FORMAT

TILINE ADDRESS	FUNCTION
F,F800	Disc Status
F,F802	Command & Surface
F,F804	Format & Sector
F,F806	Cylinder
F,F808	Count
F,F80A	Address
F,F80C	Select & Add.
F,F80E	Controller Status

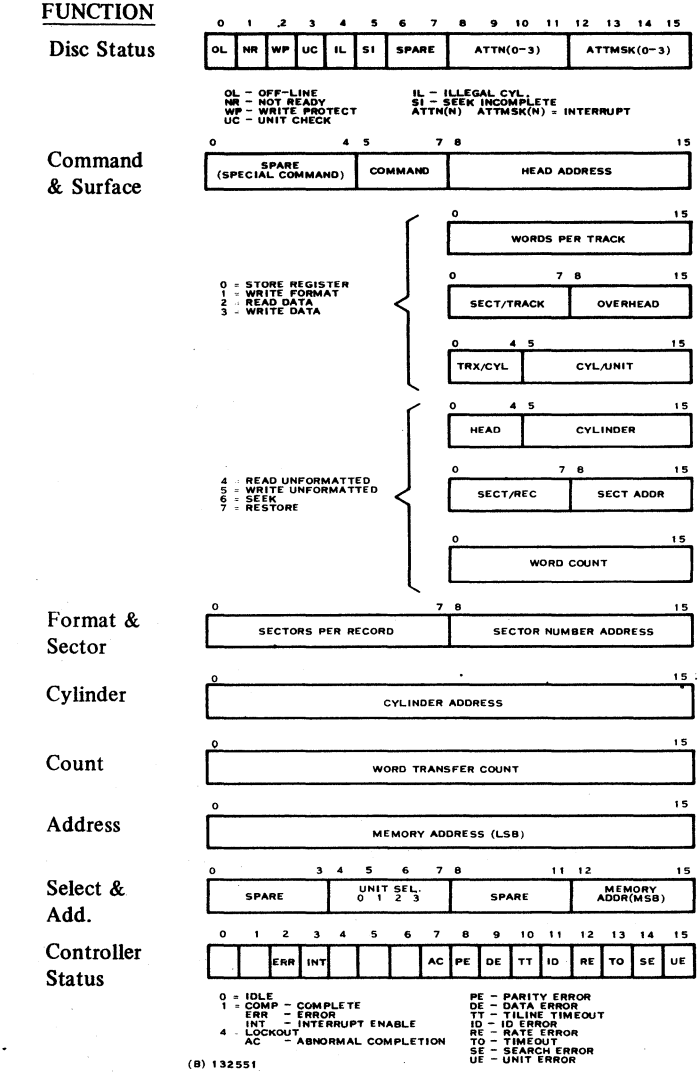


Table 5-6. Model DS31 and Model DS32 Moving Head Disc Unit Specifications (Continued)

Characteristic	Specification/Description
Dimensions	
Height	
Disc	7 inches
Power Supply	3.5 inches
Width	19.0 inches
Depth	24 inches
Space Requirements	
Disc Drive	
Height	7.0 inches
Depth	18.32 inches
Width	19 inches
Disc Unit Power Supply	
Height	3.4 inches
Depth	18.13 inches
Width	8.125 inches
Weight	
Disc	43 pounds
Power Supply	33 pounds
Mounting Rack	19 inches wide, with slides 7.0 inches of rack space per disc Power supply mounted in rear of cabinet

5.4.1.1 Operating Controls and Indicators. The operating controls and indicators for the Model DS31 and DS32 disc units are shown in figure 5-10 and described below:

Controls. The disc units have two controls:

- LOAD/RUN (Model DS31 only) - When in the LOAD position and the LOAD indicator is on, the disc cartridge can be installed or removed. When the LOAD indicator is off, the door is latched and cannot be opened. When this switch is in the RUN position, the front door is latched and the disc is rotating.
- PROTECT Indicator/Switch (Models DS31 and DS32) - A lighted push-button switch. When on, it indicates that the disc unit is in the write-protect mode. When off, the disc will accept both read and write commands.

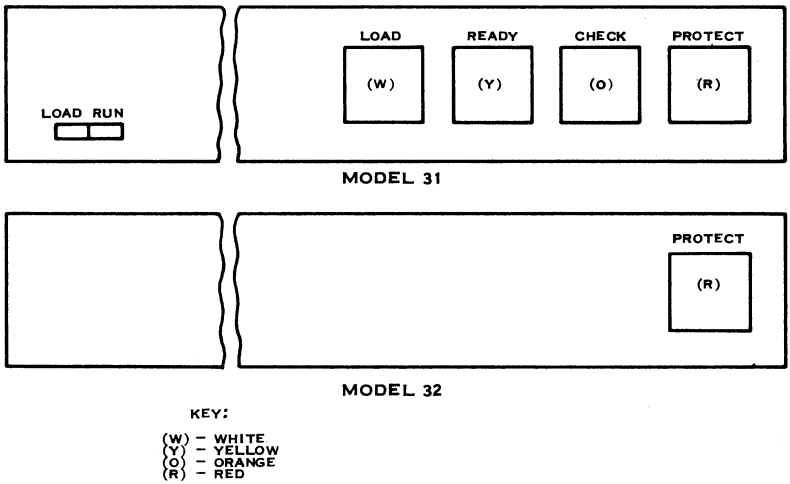


Figure 5-10. Disc Drive Unit Controls and Indicators

Indicators. There are three indicators on the Model DS31 disc unit:

- LOAD (White) - When on, indicates that the door can be opened and a cartridge either installed or removed. When off, indicates that the door is latched and the disc is rotating.
- READY (Yellow) - When on, indicates that the disc unit is loaded, up to speed and ready to accept seek, read, or write commands. This indicator stays on during the above operation.
- CHECK (Orange) - When on, indicates a faulty power condition exists and a status is presented to the controller.

Recording Method. The disc unit uses the Phase Encoded (PE) method for recording data.

5.4.1.2 Disc System Configuration. The expansion guidelines for the DS31 and DS32 allow daisy chaining up to a total of four drives from one controller. The following is a list of the available options from Texas Instruments:

- Master kit. The Model DS31 or DS32 master kits include a controller, 10-foot cable assembly, disc drive, power supply for two drives, and cartridge. This unit requires 7 inches of vertical height for rack mounting.

- Secondary kit. This kit provides the first add-on drive and utilizes the power supply mounted in the master kit. This kit includes the disc drive, cartridge, and the 4-foot daisy chain and power supply cables. This kit also functions as third add-on drive when used with the secondary kit with power supply.

- Secondary kit with power supply. This kit includes the disc drive, cartridge, cables, and power supply for two drives.

Figure 5-11 shows the Disc System configuration.

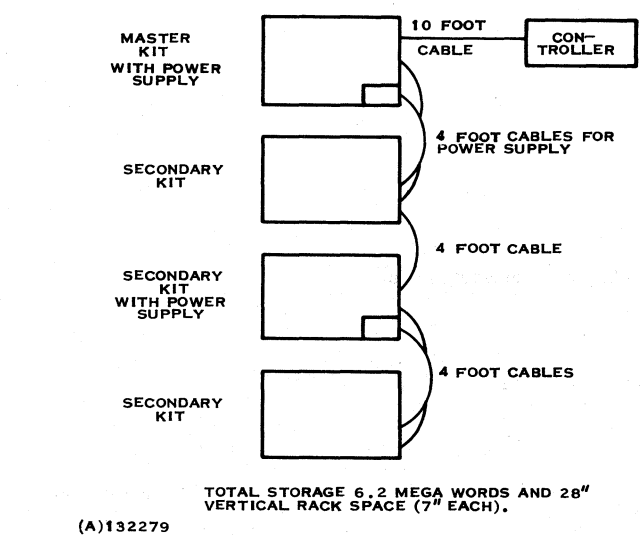


Figure 5-11. Disc System Configuration

5.4.1.3 Documentation. Additional information on the disc system is contained in the *Model 990 Computer Moving Head Disc System Installation and Operational Manual,* Manual No. 945260-9701.

5.4.2 FLOPPY DISC SYSTEM. The floppy disc system provides the 990 user a low-cost, random access, mass storage disc with storage capacity from 242K bytes to a total of 968K bytes. The diskette is an economical, flexible, oxide-coated, Mylar disc enclosed in a plastic protective jacket. This floppy system meets IBM format, recording, and medium specifications which enhances the diskette usage as a transportable medium. Technically the floppy disc is a combination of magnetic tape and disc technologies. The diskette recording medium has the appearance of a 45 RPM record; the circular diskette is packaged in an 8-inch square plastic case. The disc drive's head moves radially across the diskette as do large disc. Data may be read from or written to the diskette by specifying a track and sector address. The read/write head moves radially across the surface to the specified track and waits until the specified sector is brought to the read/write head through disc rotation.

The following are primary features of the floppy disc available for 990 users:

- Recording medium is removable diskette (oxide-coated, Mylar disc) enclosed in plastic case
- IBM-compatible
- Rack-mountable
- Four disc drives per controller with overlapped seek capability
- Power supply capable of handling two disc drives
- Additional individual disc drives available for expansion
- Head loading and unloading feature
- Internal on-board diagnostics
- Cyclic redundancy error checking of sectors
- Basic write protect feature
- Mechanical door interlock to ensure that diskette is completely inserted in drive
- Power failure detection to prevent data alteration during ac power loss.

5.4.2.1 Specifications. Table 5-7 lists in summary the floppy disc specifications.

Table 5-7. Floppy Disc Specifications

Characteristics	Specification
Data Capacity	
Tracks per Surface	77
Sectors per Track	26
Bytes per Sector	128
Bytes per Surface	256.2 K
Transfer Rate (bits per second)	250 K
Access Time	
Rotation (RPM)	360
Rotational Latency	167
Positioning Time	
• Track to Track	10 milliseconds
• Head Stabilization	20 milliseconds
• Head Load	40milliseconds

5.4.2.2 Floppy Disc System Configuration. The following are the standard floppy disc kit options available for the 990 user:

- Master kit. This kit includes one floppy drive, 7" X 19" X 15" chassis which can house two drives, power supply for two drives, floppy CRU controller, and a 12-foot cable assembly.
- Secondary kit. This kit includes disc drive, mounting hardware, and cable assembly. This unit mounts either in the master kit chassis or the floppy disc expansion kit.
- Floppy disc expansion kit. This kit includes a floppy disc expansion chassis and power supply for two drives. The expansion kit allows the 990 user to expand a master kit to a total of four drives.

The Floppy Disc System configuration is shown in Figure 5-12.

5.5 HARDCOPY INPUT/OUTPUT PERIPHERALS

Almost all computer system applications require a hard copy record capability. The impact line printer provides a solution through multicopy, on preprinted forms if required. The printed page serves as a low-cost means of distribution for system output. The preprinted forms enhance the computer application in reservations, order

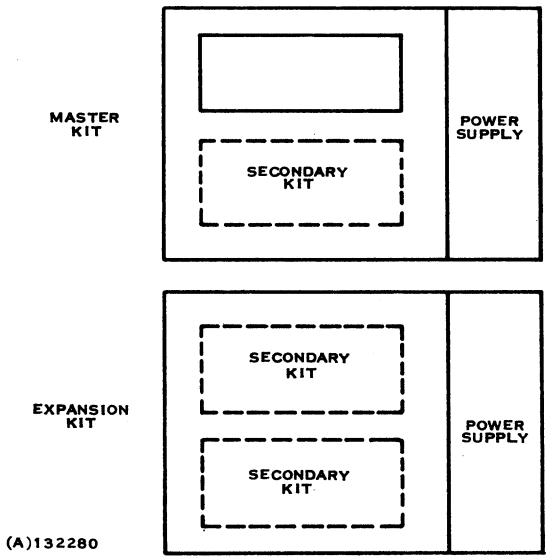


Figure 5-12. Floppy Disc System Configuration

entry, and financial transactions by providing the customer a confirmation hard copy record of the transaction. Automatic billings and other mailings also will utilize this form of printed output. Texas Instruments offers two standard models of line printers for the 990 family: the Model 306 and the Model 588.

The punched card continues to function as a widely used unique form of computer system input. The uniqueness of punched card lies in the discreteness of each card as a unit record and the convenience of manipulating it. This allows easy addition, deletion, and rearrangement of records, and provides a manual reference to individual items. When interpreted, the data punched into the card can be quickly read by humans. The Texas Instruments Model 804 card reader provides an attractive low-cost answer for these requirements.

5.5.1 MODEL 306 LINE PRINTER. The Model 306 line printer (figure 5-13) is a medium-speed, impact printer that uses a standard 5 × 7 dot matrix for character generation. This unit prints at a rate of 120 characters per second, which is approximately one full 80-character line per second (including carriage return time). Paper is sprocket-fed, and paper widths from 4 inches to 9½ inches can be accommodated. The printer can produce one original and four copies. Standard print format consists of 10 characters per inch horizontally and 6 lines vertically.

Figure 5-14 provides a cross index of the standard character set and their 8-bit patterns. Table 5-8 provides further explanation of the special control codes and their function.

5.5.1.1 Operating Controls and Indicators. The operating controls and indicators for the Model 306 line printer (figure 5-15) are as follows:

POWER Switch. The POWER switch is located on the top at the center of the control panel. This switch supplies ac power to the drive motor and power supply of the line printer when in the ON position.

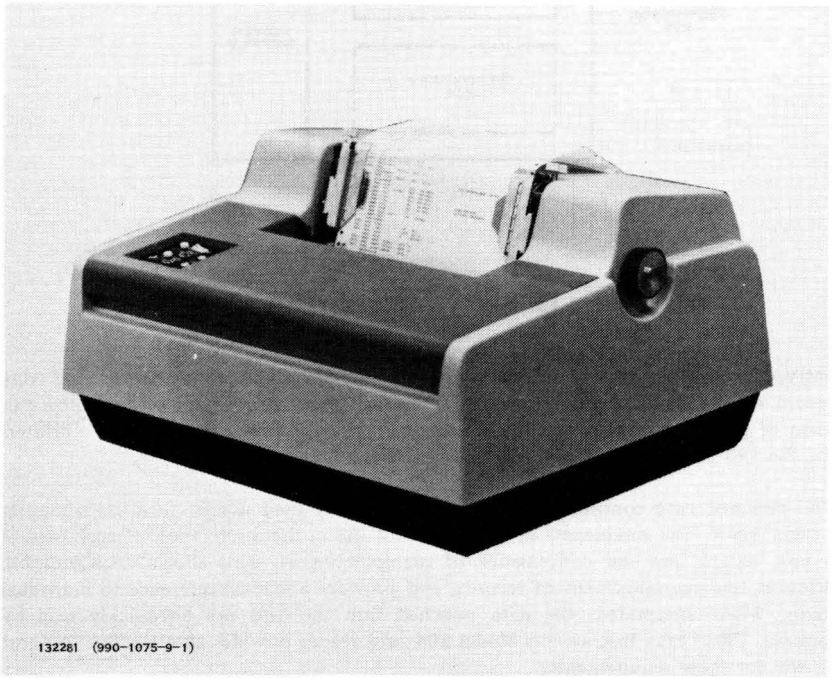


Figure 5-13. Model 306 Line Printer

b7 b6 b5 / Bits b4 b3 b2 b1 / COLUMN / ROW	000	001	010	011	100	101	110	111
	0	1	2	3	4	5	6	7
0 0 0 0 0	NUL	DLE	SP	0	@	P	`	p
0 0 0 1 1	SOH	DC1	!	1	A	Q	a	q
0 0 1 0 2	STX	DC2	"	2	B	R	b	r
0 0 1 1 3	ETX	DC3	#	3	C	S	c	s
0 1 0 0 4	EOT	DC4	$	4	D	T	d	t
0 1 0 1 5	ENQ	NAK	%	5	E	U	e	u
0 1 1 0 6	ACK	SYN	&	6	F	V	f	v
0 1 1 1 7	BEL	ETB	'	7	G	W	g	w
1 0 0 0 8	BS	CAN	(	8	H	X	h	x
1 0 0 1 9	HT	EM	)	9	I	Y	i	y
1 0 1 0 10	LF	SUB	*	:	J	Z	j	z
1 0 1 1 11	VT	ESC	+	;	K	[	k	{
1 1 0 0 12	FF	FS	,	<	L	/	l	¦
1 1 0 1 13	CR	GS	–	=	M	]	m	}
1 1 0 0 14	SO	RS	.	>	N	^	n	~
1 1 1 1 15	SI	US	/	?	O	_	o	DEL

CONTROL CODES — STANDARD — OPTIONAL

ASCII CODE

NOTES. 1. UNDERSCORE (OCTAL 137) IS REPLACED BY A BACKARROW IN THE STANDARD 9 X 7 MATRIX

2. ON THE MODEL 306 DC2 (HEX 12) IS NORMALLY USES AS A CHARACTER DENSITY CONTROL CODE. HOWEVER, AS AN OPTION, OTHER CODES MAY BE USED.

(A)132282

Figure 5-14. Standard Character Set

SELECT Switch. The SELECT switch is located immediately below the POWER switch. When depressed, the SELECT switch selects or deselects the line printer. (Puts in on-line or off-line.) When the SELECT switch has been activated, and the POWER switch is ON, the line printer is ready to receive information from the computer.

FORMS OVERRIDE Switch. The FORMS OVERRIDE switch is located immediately below the SELECT switch. Pressing the FORMS OVERRIDE switch overrides the internal paper out switch, allowing the last form to be printed before changing paper.

TOP OF FORM Switch. The TOP OF FORM switch is located immediately to the left of the FORMS OVERRIDE switch. Pressing this switch advances the paper to the next top of form.

SELECT Indicator. The SELECT indicator is illuminated when the line printer power is applied and the line printer has been selected by the SELECT Switch. This condition indicates that the line printer has been selected and is ready to receive data.

Table 5-8. Special Control Codes

Function	Hexadecimal Code	Description
Bell	07	If the printer contains optional speaker and alarm circuit, a bell code generates a 2-second audible tone to alert the operator.
Line Feed	0A	Advances the paper one line.
Vertical Tab	0B	Advances the paper until the next hole in Channel 5 of the VFU paper tape is reached. Requires optional VFU.
Form Feed	0C	Advances the paper until the next hole in Channel 7 of the VFU paper tape is reached. Requires optional VFU.
Carriage Return	0D	Causes the line of characters stored in the printer buffer to be printed. As a standard feature, an automatic line feed occurs after printing the line. No action if CR code is received before the first printable character in a line.
Elongated Character	0E	Prints entire line of characters in elongated format (double width).
Select	11	Selects the printer (i.e., makes it available to receive data).
De-Select	13	De-selects the printer (i.e., prevents the printer from receiving data).
Character Density	12	Causes the line of characters to be printed opposite to the density indicated by the N/C switch.
Delete	7F	Clears the buffer and initializes the printer electronics. As an option on new models, this function can be inhibited.

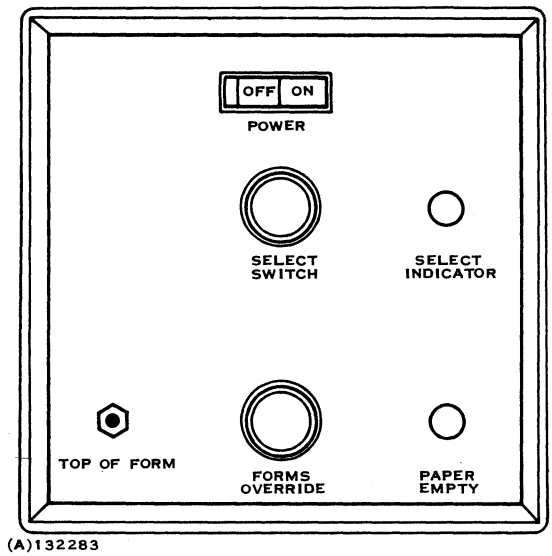


Figure 5-15. Model 306 Line Printer Operator Control Panel

PAPER EMPTY Indicator. The PAPER EMPTY indicator is illuminated when the line printer internal paper out switch is activated by the absence of paper (only one form left) in the paper hopper.

5.5.1.2 Specifications. Specifications for the Model 306 line printer are given in table 5-9.

5.5.1.3 Line Printer Configuration. The Model 306 line printer is furnished with the CRU interface controller and a 20-foot cable. Extension cables are available as options to allow remote locating the printer up to 1000 feet from the 990 computer.

5.5.1.4 Documentation. Additional information about the line printer is contained in the *Model 990 Computer Line Printer Installation and Operation Manual,* Manual No. 945261-9701.

Table 5-9. Model 306 Line Printer Specifications

Characteristic	Specification/Description
Printing Method	Impact, character-by-character, one line at a time
Printing Rate	120 characters per second 60 lines per minute (80 character line)
Transmission Rate	100 to 9600 baud
Character Structure	5 X 7 dot matrix, 10-point type
Code	ASCII, 64 characters printed
Character Buffer	80 characters (one line)
Paper Feed	Adjustable to 9½ inches
Paper	Standard sprocketed paper
Number of Copies	Original plus 4 copies
Dimensions	12¾ inches high 18¾ inches deep 23¼ inches wide
Weight	66 pounds
Electrical Power	115 Vac ±10%, 60 Hz
Operating Temperature	40°F to 100°F
Operating Humidity	5% to 90%

5.5.2 MODEL 588 LINE PRINTER. The Model 588 line printer is a heavy duty, self-contained, impact printer using a dot matrix technique similar to the Model 306 line printer for character generation. It is attractively packaged and suitable for tabletop operation. It also interfaces with the 990 computer CRU through the full duplex TTY/EIA terminal interface module.

The Model 588 line printer (figure 5-16) is a low-speed impact printer that prints at a rate of 40 characters per second over 132 columns or 150 characters per second over 20 to 30 columns, including carriage return time. Paper is sprocket-fed, and paper widths from 4 inches to 14-7/8 inches can be accommodated. The printer can produce one original and four copies.

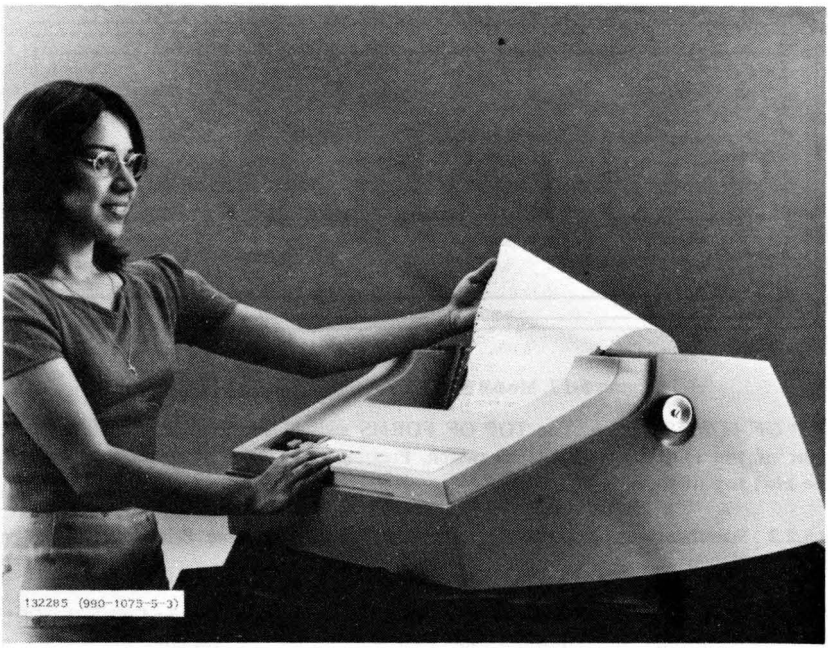


Figure 5-16. Model 588 Line Printer

5.5.2.1 Operating Controls and Indicators. The operating controls and indicators for the Model 588 line printer, shown in figure 5-17, are located on the operators panel on the lower front corner of the line printer. Each control and indicator is described in the following paragraphs:

POWER Switch. The POWER switch is located on the left side of the control panel. This switch applies ac power to the drive motor and power supply of the line printer when pressed to the on position.

SELECT Switch/Indicator. The SELECT switch/indicator is located immediately to the right of the POWER switch. When depressed, the SELECT switch/indicator selects or deselects the line printer (puts it on-line or off-line). When the SELECT switch/indicator has been activated, and the POWER switch is on, the line printer is ready to receive information from the computer and this switch/indicator is illuminated.

FORMS OVERRIDE Indicator/Switch. The FORMS OVERRIDE indicator/switch is located immediately to the right of the SELECT switch/indicator. Pressing the FORMS OVERRIDE indicator/switch overrides the internal paper out switch, allowing the last form to be printed before changing paper. This indicator/switch becomes illuminated when the internal paper out switch is activated by the absence of paper.

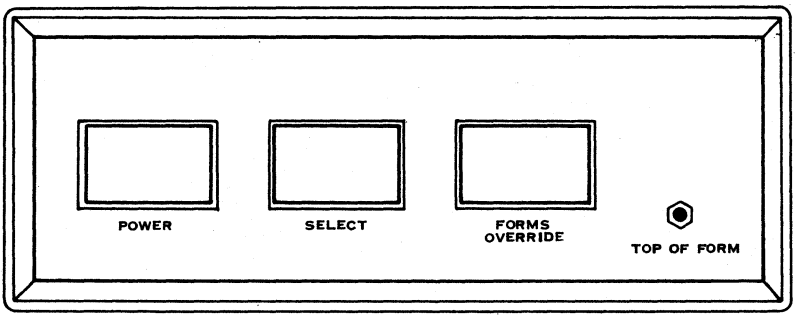


Figure 5-17. Model 588 Line Printer Control Panel

TOP OF FORMS Switch. The TOP OF FORMS switch is located immediately to the right of the FORMS OVERRIDE switch. Pressing this switch advances the paper to the next top of form.

5.5.2.2 Specifications. Specifications for the Model 588 Line Printer are listed in table 5-10.

5.5.2.3 Line Printer Configuration. The Model 588 printer is furnished with the CRU interface controller and 20-foot cable. Extension cables are available as options to allow remote locating of the printer up to 1000 feet from the 990 computer.

5.5.2.4 Documentation. Additional information on the Model 588 line printer is contained in the *Model 990 Computer Line Printer Installation and Operation Manual,* Manual No. 945261-9701.

5.5.3 MODEL 804 CARD READER. One of the most convenient methods for entering data and information which has been generated externally is by punched cards. The card medium allows the operator to generate and edit data externally to the computer and provides a convenient medium for storing program material for later use.

The Model 804 card reader (figure 5-18), is a column-oriented device intended to read punched hole data in 80 column cards at approximately 400 cards per minute. It reads cards by transporting them individually past a fiber optic read station. The data image is transferred through the fiber optics to an array of photosensors. The Model 804 card reader is housed in an attractive table-top package and is complete with input hopper, feed mechanism, read station, stacker mechanism, output stacker and timing mechanism with supporting drive belt and motor. In addition to the mechanical assemblies, the card reader contains control and error electronics, a power supply, operating controls and indicators, a cabinet, and a hopper card weight.

Table 5-10. Model 588 Line Printer Specifications

Characteristic	Specification/Description
Printing Method	Impact, character-by-character, one line at a time
Printing Rate	88 characters per second 25 lines per minute (132 character line)
Transmission Rate	100 to 9600 baud
Character Structure	5 X 7 dot matrix, 10-point type
Code	ASCII, 64 characters printed
Character Buffer	132 characters (one line)
Paper Feed	Adjustable to 14-7/8 inches
Paper	Standard sprocketed paper
Number of copies	Original plus 4 copies
Dimensions	13-3/4 inches high 21-3/4 inches deep 32-inches wide
Weight	98 pounds
Electrical Power	115 Vac ±10%, 60 Hz
Operating Temperature	40°F to 100°F
Operating Humidity	5% to 90%

5.5.3.1 Operating Controls and Indicators. The card reader controls and indicators shown in figure 5-19, function as described in the following paragraphs:

POWER switch. The POWER switch is located on panel at the back of the card reader. It applies ac power to the internal power supply.

POWER ON indicator. The POWER ON indicator indicates power is applied to the reader and the power switch is on.

RESET switch/indicator. The RESET switch/indicator is a momentary contact switch that resets the electronics and conditions the reader to a ready or not ready condition depending upon the previous condition.

HOPPER indicator. The HOPPER indicator indicates an empty input hopper or a feed failure.

STACK indicator. The STACK indicator indicates a full output stacker or a stack failure.

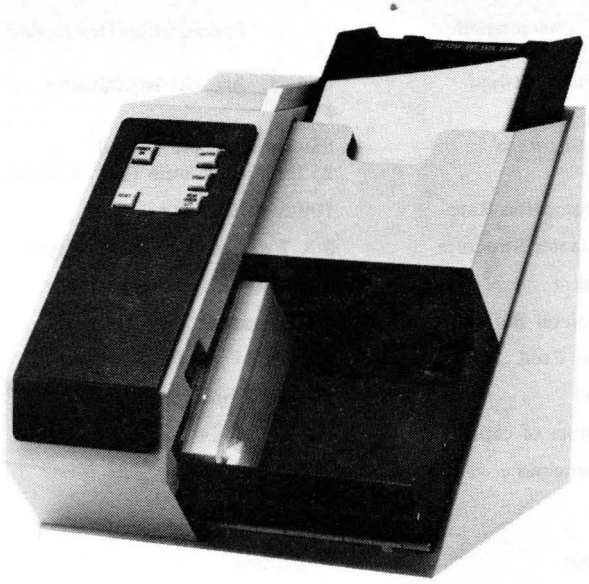


Figure 5-18. Model 804 Card Reader

READ CHECK L/T indicator/switch. The READ CHECK L/T indicator/switch indicates a read station failure or possible data error. When pressed, the switch tests all front panel indicators if the reader is not in the ready condition.

5.5.3.2 Documentation. Additional information about the card reader is contained in the *Model 990 Computer Card Reader Installation and Operation Manual,* Manual No. 945262-9701.

5.6 INTERFACE MODULES

Since computers often converse with equipment other than standard peripherals, Texas Instruments offers a selection of interface modules which provide communications capabilities between the computer and custom/or non-standard peripherals. To simplify the interface design, interface modules are available in full duplex TTY/EIA configuration, 16-bit I/O with TTL level outputs, and 16-bit I/O data module with EIA level outputs. Each of the modules available is described in the following paragraphs.

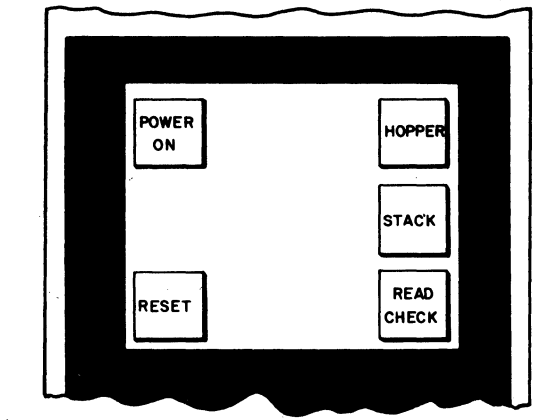


Figure 5-19. Card Reader Controls and Indicators

5.6.1 TTY/EIA TERMINAL INTERFACE. The full duplex asynchronous TTY/EIA terminal interface module/interfaces the 990 Computer to a variety of RS232C compatible equipment via the CRU and is operated under program control to send and receive character data and data terminal control signals. Full duplex communications is by means of a voltage interface compatible with EIA standard RS232C. Available as an option is 20 mA neutral signaling. Input from the data terminal is converted to CRU interface signals by the module and the signals are used by the 990 Computer to determine terminal status, activity requirements, and user input requirements of the data terminal. The module may be wired to generate interrupts to the 990 computer. Interrupts are generated for each character sent, each character received, and when the status of the attached device changes (RS232C only).

The TTY/EIA module permits the connection of TTY compatible or EIA compatible devices to the 990 Computer CRU for communication interaction between the attached devices and the computer.

Wireable options on the module permit the selection of the data transmission rate of the device and the selection of the type of interface from either TTY compatible or EIA compatible specifications. The data transmission rate may be selected from several rates that begin at 110 baud and extend through 9600 baud. An 11-bit code is used with the 110 baud rate while 10-bit codes are used with higher data transmission rates.

5.6.1.1 Specifications. Specifications for the full duplex TTY/EIA Interface Module are listed in table 5-11.

Table 5-11. TTY/EIA Interface Module Specifications

Characteristic	Specification/Description
Peripheral Input Lines	
EIA Data	-3 V to -25 V for logic 1 (marking) +3 V to +25 V for logic 0 (spacing)
EIA Control	+3 V to +25 V for logic 1 (on) -3 V to -25 V for logic 0 (off)
TTY Data and Control	Short circuit for logic 1 Open circuit for logic 0
Module Output Lines	
EIA Data	-5 V to -11 V for logic 1 +5 V to +11 V for logic 0
EIA Control	+5 V to +11 V for logic 1 -5 V to -11 V for logic 0
TTY Data and Control	20 mA current loop for logic 1 Open circuit for logic 0
Electrical Power (from Computer or Expansion Chassis)	+5 Vdc at 0.38 A +15 Vdc at 20 mA -15 Vdc at 20 mA
Transmission Rates	110, 150, 300, 600, 1200, 1760, 2400, 4800 and 9600 baud

5.6.1.2 Documentation. Additional information about the full duplex EIA communications interface module is contained in the *Model 990 Computer Hardware Reference Manual,* Manual No. 945251-9701.

5.6.2 16 I/O DATA MODULE (TTL) The 16 Input/16 Output (16 I/O) Data Module provides two-way communication between the computer and devices or transducers which are operated by, or generate digital control signals. The module may be used in any Communication Register Unit (CRU) port within the CPU chassis or in a CRU expansion chassis.

The 16 Input/16 Output Data Module provides 16 input lines and 16 output lines. Each line may be addressed as a single independent binary value or as a member of a group of 2 to 16 lines. An alternate version of the data module provides 15 normal inputs, 14 normal outputs, an interrupt input line, and an interrupt mask output.

Data module inputs and outputs are negative logic levels that switch between 0 volt and a positive voltage level. Each output is an open collector transistor that is capable of sinking up to 50 milliamperes at up to 30 volts. Pads are available on the module for installing pull-up resistors to the module +5 volt supply. Each input is connected to the base of an emitter-follower transistor that operates at TTL levels. Pads for input filter capacitors are available on the module.

If the interrupt option is selected, jumper selections are provided for the interrupt to be set by a logic ZERO or logic ONE level, or by a positive or negative transition.

The operation of the 16 I/O module depends entirely upon what peripheral is connected to the interface. Output lines may be connected to non-computer or computer peripherals for control applications and input lines may be used for data input from these peripherals.

Additional information about this module is contained in the *Model 990 Computer 16 Input/Output TTL Data Module Depot Maintenance* Manual, No. 945407-9701.

5.6.3 16 I/O EIA DATA MODULE. The 16 I/O EIA data module provides a general purpose 16-bit input and output interface between the 990 Computer and any external device(s) whose interface requires EIA voltage levels. This module provides 16 input lines and 16 output lines. Each line may be addressed as a single binary value or as a member of a group of 2 to 16 lines. The input and output specifications are:

- 16 inputs - logic one: +3 to +25 V
  logic zero: -3 to -25 V

- 16 outputs - logic one: +5 to +8 V
  logic zero: -5 to -8 V.

The 16 I/O EIA data module requires one half-slot in the standard 990 chassis and requires 340 mA of +5 V, 110 milliamps of +12 V, and 70 mA of -12 V as typical average dc power.

Additional information about the 16 I/O Data Module (EIA) is presented in Specification Drawing 964882.

5.7 READ-ONLY-MEMORY IMPLEMENTATION DEVICE

The microprocessor and microcomputer on a board provide a lower cost/higher performance replacement for electromechanical controls for use in major appliances, heavy equipment and process control equipment. In each of these applications the processor needs to function as a dedicated stand-alone controller preferably without supporting peripherals. This requires that once the unit is installed, it must continue to function through power-on/power-off cycles without removal except for corrective maintenance.

The read-only-memory (ROM) provides the system designer an effective solution to this requirement. The processor software coding can be implemented in read-only-memory as firmware, and the volatile dynamic or static Random Access Memory (RAM) can be utilized for temporary storage of data and as software working space.

The type of read-only-memory used for implementation of firmware depends on the quantity to be produced. Large quantities are implemented in standard ROM, while very low quantities are more economically implemented in Programmable Read-Only-Memory (PROM). During the prototyping and breadboard stages of development, the firmware for both cases are best developed in either PROM or the erasable PROM (EPROM).

The Texas Instruments Universal PROM Programming Module provides the means for implementing these devices for the 990 user. This unit is available as either an attractive tabletop model or with mounting hardware for installation in a standard 19-inch RETMA cabinet. Supporting software for this module is described in Section IV.

5.7.1 PHYSICAL FEATURES. The PROM Programming Module (tabletop model) is housed in an attractive, low profile enclosure with 12-foot cable and CRU interface module for connection to the 990/4 Microcomputer. Plug-in Personality cards for both the PROM and EROM devices are available. The module enclosure includes front panel controls, device sockets with personality card, and power supply.

5.7.2 OPERATING FEATURES. The module operates as a CRU device and software programs have direct control over the PROM address, data to be implemented, and limited control over the width of programming pulse. This control is exercised in the module by the contents of five 8-bit registers. These registers contain address, data, and pulse width information. Status information supplied back from the module includes module online, busy signal, timing error, and a hardware fault signal.

Program implementation of the PROM/EPROM device is performed individually for each of the memory locations and consists basically of applying a signal to each output terminal where a bit is to be programmed. The procedure is irreversible for the PROM devices, and once altered, the output for that bit is permanently programmed to provide a high level. Outputs never having been altered may later be programmed to supply a high-level output.

The EPROM devices can be erased using a high intensity ultraviolet lamp and then reprogrammed by the Universal PROM Programming Module.

5.7.3 DOCUMENTATION. Additional information about the PROM Programming Module is contained in the *Model 990 Computer PROM Programming Module Installation and Operation Manual,* Manual No. 945258-9701.

SECTION VI

990 SYSTEM DESIGN

6.1 SCOPE

This section is a designer's guide to the implementation of 990 computer systems. The objective of this section is to present the 990 series in sufficient depth to permit analysis of the technical feasibility of contemplated projects based on the 990 series. The material presented combined with current pricing information, to permit analysis of the economic feasibility, will provide an overall feasibility outlook. Computer price information is volatile and cost analyses should be based on current information.

The 990 Computer is priced lower than previously available computers of comparable power. A number of projects which were not previously economically feasible will now become attractive through use of the 990 series. In many cases, the potential designer may never have implemented a computer system; however, there are many examples of completely successful computer systems which were the designer's first effort. Granted a prudent, thorough, and methodical approach, there is every expectation of success for a first design that there is for a tenth. The material presented and the factors considered in this section are complete according to contemporary practice. The processes outlined may seem elaborate at first reading, but these processes were developed through experience and each factor has the potential to vitally affect project success. It is economically sound to deal with these factors in planning rather than later.

The 990 series offers a designer the best value in currently available computers, measuring value in performance/cost ratio. It is Texas Instruments firm intention to maintain this value leadership. The 990 series, therefore is the most reasonable benchmark for testing the soundness of an economic proposition based on computer systems.

The range of the series - from single chip microprocessors through packaged computer systems, all upward compatible in software and packaging - offers the designer an attractive range of choices in initial design and a series of backup positions should circumstances change after design commences. Texas Instruments support at every level offers a designer leverage and the opportunity to multiply his effectiveness by using the support where it suits his purposes and concentrating his efforts in critical areas.

Finally, the 990 series is Texas Instruments mainstream computer effort so that designers are assured that Texas Instruments resources are directed toward development of this series resulting in a continuous competitive position for the 990 series and for computer systems based on the 990 series.

6.2 DEFINITION OF DESIGN PHASES

Texas Instruments normally identifies six phases in the development of a project. These are:

- Planning
- Evaluation
- Development
- Testing
- Production
- Sustaining.

These phases are relevant to the design process in that the requirements of each phase are different and the designer must provide for the requirements of each phase during the initial phase: planning. The size of the project is not relevant. Small projects and large projects alike go through these six phases, adjusting the scale of the effort in each phase to the scale of the project. (Very small projects sometimes eliminate the production phase).

Serious consideration of a project requires completion of the first two phases. The technical and economic feasibility of a contemplated project should be regarded as questionable until evaluation is completed and demonstrates a favorable result.

This handbook is primarily aimed at supporting these initial two phases. Other manuals support later phases.

6.2.1 PLANNING PHASE. The planning phase develops a fairly detailed design together with cost and schedule information on a proposed project. The milestone signaling completion of this phase is a system specification.

6.2.2 EVALUATION PHASE. The evaluation phase consists of a consideration of the feasibility and desirability of the proposed project, generally by comparison of the material developed during planning to available resources and to market research information. This phase in particular varies considerably in time and level of effort according to the size of the project. Evaluation finally terminates with a GO/NO-GO decision.

6.2.3 DEVELOPMENT PHASE. The development phase completes the design and particularly the documentation and usually builds one or more prototypes.

6.2.4 TESTING PHASE. The basic purpose of the testing phase is to determine whether the prototype meets specifications and is suitable for release to customers. This phase varies considerably in time and level of effort with the size of the project.

This phase ends when all discrepancies are satisfied and the project is released to production.

6.2.5 PRODUCTION PHASE. The production phase commences with release for production, builds and ships the projected systems, and terminates when production is complete. During the planning phase designers should consider the effects of lead time, level of inventory, testing requirements, repair cycles, maintenance of adequate records, and methods for introducing modifications in the production phase. Each of these factors contribute to cost.

6.2.6 SUSTAINING PHASE. The level of effort during the sustaining phase varies over a considerable range according to the number of systems, the size and complexity of an individual system, and geographic dispersal. Support tasks commence with deliveries to customers, including training. It will usually be most convenient to the designer to consider that the sustaining phase commences with sustaining tasks and overlaps production. Normal business ethics require that a manufacturer either conform to industry standards in support or clearly state what support he provides. For computer systems, the minimum expectation is support throughout the amortization period.

During the planning phase designers should consider support response time, repair time, spares inventories, and training for programmers, operators, and maintenance technicians.

Texas Instruments provides support for the standard 990 series hardware and software and is generally willing to quote on customer designed systems based on 990 Computers. See Section VII for explanation of standard 990 support and a discussion of special support.

6.2.7 SUMMARY. This handbook, together with current price information, is intended to support designers during the planning and evaluation phases of a project. Later phases are addressed to the extent required by planning and evaluation and these later phases are themselves supported by more detailed technical manuals.

This handbook supports standard 990 series products. Special circuit designs are easily incorporated and such designs are supported by Texas Instruments technical manuals on circuits and design.

6.3 DEFINITION OF DESIGN PROCESS IN PLANNING PHASE

The design process is a process of definition, of bringing concepts into sharp, clear definition. It naturally follows that the early stages are less well-defined than the later stages. During the planning phase design must address a broad range of subjects and it is important to use a methodical approach. The definitions which follow are used internally at Texas Instruments. Like project definition, the same steps are always followed and only the scale of the effort is adjusted according to the magnitude of the design task.

Texas Instruments identifies four major stages in design:

- Concept
- Functional Description
- Implementation
- Cost Model.

These stages are subdivided into steps as described in the following paragraphs.

The design process is iterative in two senses. First, the process itself is iterative in that later stages may require altering earlier steps; consequently, iterating the process while closing on the final design. Second, the evaluation of alternatives will frequently require that each alternative be carried to final design in order that all factors may be properly considered.

6.3.1 CONCEPT. The concept stage of design is the idea stage. The level of effort varies over a wide range primarily with the level of support that the idea attracts, and may include market studies and consultation with experts. At this stage the designer should study the products and the design process he intends to use to the extent that only specific references on particular points are required during the later steps.

6.3.1.1 Early or Late Decisions. An important decision which the designer faces immediately is whether to design around particular equipment or to specify his system in a way that permits later decision. Generally, the design process will be shorter and easier with an early decision on specific equipment. The alternative, a deferred decision, is desirable only, (1) where the project intends to conduct competitive bidding at a later stage or, (2) where the availability of the product at a later stage is questionable and hence a capability to use alternative products is desired. Contemporary practice is nearly universal in an early decision on the computer with the exception being government agencies required by law to conduct competitive bidding and those projects with sufficient volume to significantly effect the computer manufacturer's output.

6.3.1.2 Build or Buy? The designer may discover that the 990 series offers an impressive number of alternatives: for example, should he buy the Texas Instruments chassis or build his own? Buy the 990/4 CPU assembled or buy the 9900 processor and assemble his own? And in the latter case, buy the Texas Instruments circuit boards or build his own? Note that six basic alternatives were outlined without considering the numbers of options in chassis and CPUs. The most satisfactory solution will nearly always be the alternative that offers the designer the greatest economic advantage. The greatest advantage usually lies with lowest cost, which in turn is dependent on the projected volume. Where the volume is very high so that non-recurring charges are insignificant compared to recurring charges, and the overhead of production, inspection, testing, training, etc., are low compared to material cost, then a "build" decision is indicated. Where volume is very low so that

production is no consideration and design costs are not charged to the project; then again, a "build" decision is indicated. With these exceptions, the best alternative is to buy the most complete product available and augment or modify, if necessary, to suit the project.

Designers who contemplate buying the TMS9900 chip and assembling their own computer should weigh the possible advantages of maintaining compatibility with Texas Instruments software and packaging. The TMS9900 is itself compatible, but software compatibility is a multi-level subject and full compatibility imposes requirements on the organization of software in memory, trap locations, and peripheral interfaces.

6.3.1.3 990/4 or 990/10? Designers who intend to buy at the board level or higher should use the 990/4 where possible to gain a cost advantage. The general exceptions will be systems requiring the TILINE (only present on 990/10) for memory size larger than 32K or high-speed peripherals (including other CPUs), or else requiring the speed of the 990/10 CPU.

Designers should also consider the possibility of using mixed systems, for example, some units using TMS9900, some using 990/4 CPUs and some using 990/10 CPUs. The usual historical objection to such systems has been the increased support primarily in software and maintenance required by multiple computer types. The 990 series compatibility and the broad range of possibilities in using Texas Instruments support tends to remove this objection.

6.3.2 FUNCTIONAL DESCRIPTION. When the preliminary considerations outlined above are completed, the designer is ready to begin the formal design process. Functional description is the first formal stage in definition. It commences with a functional block diagram and ends with a system specification.

6.3.2.1 Level of Detail. The boundary between functional description and implementation (the next stage of design) is flexible. It is desirable to partition the system into hardware and software during the description stage and to specify this boundary in the system specification. Unusual tasks or functions must be described in sufficient detail that the specification is exact and not dependent on supplementary explanations. Beyond this, too much detail on implementation simply delays the system specification without adding to the usefulness.

6.3.2.2 Functional Block Diagram. The functional description commences with a block diagram. The important point in the functional block diagram is to get the idea on paper with all of the major functions represented. Drawing conventions are unimportant and the diagram will probably evolve during design.

6.3.2.3 Supporting Descriptions. Supporting descriptions are explanations and elaborations of the functional block diagram. The description always includes a narrative and adds diagrams, schematics, tables, figures, etc., as required. Standard equipment should be identified and specified. Special developments should be addressed in detail. The description is usually addressed in three stages:

- Steady state
- Start-up
- Fault analysis

Steady State. First, the system is considered in normal steady state operation, usually a flow process for a computer system. All operational inputs, outputs, and processes should be identified and described. The description should include media, format, and rate information. Approximations and estimates should be clearly identified as such. Operator interactions should be specified, if required, and where considerable interaction is required then, the man-machine interface should be separately identified and analyzed in the description.

Start-up. The next step is to specify how the system is to be started. The designer should consider the following subjects:

- Initial Start Sequences (cold start)

  Power Up

  Program load, specify media

  Commence execution

  Operator interaction, such as date-time entry.

- Restart Sequence

  With program and/or data in memory

  With system in partial operation

- Software update

  Deployment methods

  Operation during conversion.

Start up sequences are greatly simplified through the use of ROM devices. A program contained in ROM is always present with a dependability which precludes any need for back-up loading and the program is tamper-proof. EPROM devices provide a controlled software update capability. Future cost considerations also favor ROM devices, which are decreasing rapidly in cost, over peripheral loaders and communication line downloading.

Fault Analysis. The final step in description is to specify system behavior in the presence of faults, fault detection and isolation, controlling the propagation of errors, and repair and recovery techniques. The following faults should be addressed:

- Power failure
- Operator error, including omissions
- Irreversible error and cancellation
- Hardware faults, including cables
- Software "bugs".

Designers should consider the effects of bit errors (corresponding to "noise" in analog systems), uptime requirements, maintenance response and repair time, the possibility of continued operation in a degraded mode, and the possibility of providing redundancy for backup.

At this point the designer should consider the use of 990/10 computers with error correcting memory. The error correcting memory does *not* reduce the device failure rate and systems with error correcting memory will still experience device failures in direct proportion to the number of devices used. The error correcting memory *does* practically eliminate bit errors in the memory system and on systems with memories larger than 20K words will reduce the *system* failure rate by tolerating failed devices. The improvement is proportional to memory size and for large memories error correction should be considered mandatory.

6.3.2.4 Preliminary Schedule. The material developed above constitutes a description of the system. The next step is to assign a preliminary schedule for the project and to develop a plan for the later phases of the project. The most critical aspect of the preliminary schedule is that it will determine what products in rapidly developing fields may be considered for use in implementing the system. In planning for later phases the detailed decisions may be deferred but there should be a statement of intent on the following points:

1. Evaluation Plan
    - Identify feasibility demonstrations if required.

2. Development Plan

   - Training requirements for programmers and designers
   - Number of prototypes to be developed
   - Method for producing software.

3. Testing Plan

   - Scope of testing and possible deployment of test systems.

4. Production Plan

   - Specify the number of systems to be produced and production rate versus schedule.
   - Allow build-up for high volume production.

5. Sustaining Plan

   - Identify customer training requirements.
   - Specify method for installations and maintenance.

6.3.2.5 System Specification. The material developed above should be incorporated in a system specification. Accuracy and completeness are important. This specification is the milestone that signals completion of the functional description.

6.3.3 IMPLEMENTATION. Implementation is the stage following the functional description in the planning phase. The level of detail desired is easily determined by looking ahead to the next step where accurate costs must be determined. When the level of detail permits accurate cost determination, then the objectives of this step have been realized and further definition simply delays the projects.

It is in the implementation stage that the advantage of an early decision to design with particular equipment pays off with a quicker, simpler design cycle. First, given that a particular computer is to be used, the design requirements of implementation can usually be satisfied by a single pass through the design process; whereas the consideration of alternatives will usually cause a separate pass for each alternative. Second, the implementation process itself is simplified by the early identification of equipment to be used; for example, given that the 990 series is to be used, the implementation requirement for a synchronous communication line interface is satisfied by simply selecting the kit or module which best suits the system requirement.

Familiarity with the current 990 series price list at the time implementation is undertaken will pay off in shorter design time. There are generally a number of alternatives offered in each basic module and a menu approach to implementation could easily overlook the most suitable combination. Texas Instruments sales engineers will be glad to assist designers by either explaining the price list items or in selecting the most suitable combination.

The implementation of a computer system is usually subdivided into hardware and software implementation. This is entirely due to the fact that separate specialists in hardware and software usually perform the design. The hardware and software implementation tasks interact and therefore should be performed simultaneously, although they are described separately in this handbook.

6.3.3.1 Hardware Implementation. Hardware implementation is subdivided into the following tasks, which are performed in sequence:

- Detailed block diagram
- Electrical analysis
- Packaging analysis.

Detailed Hardware Block Diagram. The objective of the detailed block diagram is to identify every input, output, and peripheral device in the system and to identify where it will interface. There is both an electrical and a physical interpretation of the diagram and it may be desirable to generate separate diagrams. Furthermore, in complex systems it is usually desirable to diagram in levels; thus, for example, the top level diagram might show chassis and peripherals, while a lower level shows content of each chassis.

Electrical Analysis. This analysis explains and enlarges the detailed block diagram. The electrical analysis is basically a narrative description supported by diagrams, schematics, tables, etc., as required, to define the block diagram. The subjects addressed are throughput, capacity and compatibility; and, the objectives are continuity and completeness. The level of detail varies as required, for example, one interface might be described as "Bell 103A compatible," while a special interface might require several pages of tables, timing diagrams, and schematics. Designers must provide the level of detail to define and specify each feature.

Packaging Analysis. The packaging analysis completes the hardware design by specifying the packaging, cabling, and space and power requirements. Packaging is usually addressed in three levels:

- Circuit board layout and wiring (for custom designs)
- Chassis
- Cabinets or equipment racks.

Packaging analysis must include cabling between chassis, cabinets, and peripherals. The packaging summary is easily handled with either an indented list format or drawings showing circuit board/chassis and chassis/rack arrangements. Table 6-1 presents the space requirements for 990 series components.

The power summary is best accomplished by generating primary and secondary power tables. The primary power table simply lists the equipment using primary power and the current requirements for each. The secondary power summary should contain a separate table for each power supply except for those power supplies specifically designed for a particular load and driving only that load (for example a 913A VDT driving only the 913 display and keyboard). Each secondary power table should list the voltages generated, the current available, and the current load of each module on that supply. Table 6-1 presents the power requirements for 990 series components.

Standby Power. The RAM devices used in 990 series memories are dependent on uninterrupted power to maintain the data content. This factor is of no interest in many applications, but where the system requires that data content be preserved with power off, then the designer must provide some alternative power source, such as a battery, to preserve data during the primary power interruption.

Texas Instruments offers a standby power option consisting of a battery, charger, and the circuitry required to maintain memory content during power interruptions. This kit mounts in either the 6-slot or 13-slot chassis. The primary power drain is increased by 60 watts, but the chassis dimensions and card capacity are not affected. The charger will fully charge the battery in 24 hours. A fully charged battery will sustain memory content for:

4K memory, 8.0 hours at 25°C

32K memory, 1.0 hours at 25°C.

Standby Power Supply Capacity

Voltage Furnished	+5 V MEM	+12 V MEM	-5 V MEM
Current Available	1.4 A	1.2 A	0.1 A

Should the designer prefer to furnish standby power from another source he should refer to tables 6-2 and 6-3 which present the standby power requirements for 990 series RAM memory modules. Plus 5 volt power is split on the memory modules and 990 series chassis into +5 V MAIN and +5 V MEM. The +5 V MAIN is not required during standby operation. All voltages designated "MEM", +5 V MEM, +12 V MEM, and -5 V MEM, are vital to memory content and must be maintained without interruptions or transients. All currents on the tables are in amperes. Operating currents are the current drawn during normal operation. Standby currents are the power drain during standby.

Table 6-1. 990 Power and Space Requirements

Price List Item No.	Description*	Prerequisite Item Number	Part Number	DC Power +5V	DC Power +12V	DC Power -12V	Space Requirements
	990/4 Microcomputer						
	The following items are standard 990/4 CPU options. These items include only the 990/4 CPU board. Additional options for memory expansion, chassis, power supplies, interfaces, etc. are described in subsequent tables.						
200	990/4 CPU w/256 Words Static RAM	–	944910-0001	-1.25	-0.04	–	Full Slot
201	990/4 CPU (4K Dynamic RAM)	–	944910-0002	-1.12	-0.64	–	Full Slot
	990/4 Microcomputer On-board Options						
	The following options are available for mounting on the 990/4 CPU board. The CPU has sockets for up to a total of 1024 words of ROM/PROM/Static RAM (Items 204 through 207)						
202	Dynamic RAM Parity Option	201	945120-0006	-0.02	-0.02	–	–
203	733 ASR/Card Reader ROM Loader	201	945121-0001	-0.45	–	–	–
204	733 ASR/Card Reader ROM Loader w/Self-Test	201	945121-0002	-0.9	–	–	–
205	990/4 Floppy Disc Loader	201	945121-0003	-0.45	–	–	–
206	990/4 Floppy Disc Loader w/self-test	201	945121-0004	-0.9	–	–	–

*Memory storage is expressed in **Number of 16-bit Words.**

Table 6-1. 990 Power and Space Requirements (Continued)

Price List Item No.	Description*	Prerequisite Item Number	Part Number	DC Power +5V	+12V	-12V	Space Requirements
207	733 ASR ROM Loader (Prototyping)	209	945121-0005	-0.9	–	–	–
208	PROM Device Kit	200-201	945123-0001	-0.45	–	–	–
209	Static RAM Device Kit	200-201	945122-0001	-0.25	–	–	–

990/4 Microcomputer Memory Expansion Options

The following are the memory expansion options available for the 990/4. This memory is on a single full-sized board. Memory parity options are mounted on the associated memory expansion module. EPROM memory module (item 235) is supplied with 1024 words. The EPROM device kit includes an additional 1024 words and the EPROM memory module has sufficient sockets to accommodate up to 16K words.

220	4K Memory Expansion Module	200-201	944935-0001	-0.7	-0.02	–	Full Slot
221	8K Memory Expansion Module	200-201	944935-0002	-0.77	-0.03	–	Full Slot
222	12K Memory Expansion Module	200-201	944935-0003	-0.84	-0.04	–	Full Slot
223	16K Memory Expansion Module	200-201	944935-0004	-0.91	-0.05	–	Full Slot
224	20K Memory Expansion Module	200-201	944935-0005	-0.98	-0.06	–	Full Slot
225	4K Memory Expansion Module w/write protect	200-201	944935-0006	-0.75	-0.02	–	Full Slot

Table 6-1. 990 Power and Space Requirements (Continued)

Price List Item No.	Description*	Prerequisite Item Number	Part Number	DC Power +5V	DC Power +12V	DC Power -12V	Space Requirements
226	8K Memory Expansion Module w/write protect	200-201	944935-0007	-0.82	-0.03	–	Full Slot
227	12K Memory Expansion Module w/write protect	200-201	944935-0008	-0.89	-0.04	–	Full Slot
228	16K Memory Expansion Module w/write protect	200-201	944935-0009	-0.96	-0.05	–	Full Slot
229	20K Memory Expansion Module w/write protect	200-201	944935-0010	-1.03	-0.06	–	Full Slot
230	4K Memory Parity Option	220,225	945120-0001	–		–	–
231	8K Memory Parity Option	221,226	945120-0002	–		–	–
232	12K Memory Parity Option	222,227	945120-0003	–		–	–
233	16K Memory Parity Option	223,228	945120-0004	–		–	–
234	20K Memory Parity Option	224,229	945120-0005	–		–	–
235	EPROM Memory Module	200-	945170-0001	-0.4	-0.1	-0.06	Full Slot
236	EPROM Device Kit	235	945123-0004	-0.01	-0.1	-0.06	–

990/10 Minicomputer

The following are standard 990/10 CPU options. These listed items include three full size boards. The first two are the arithmetic units and the third board includes memory controller with indicated memory as listed. The error correcting memory is abreviated as ECC.

300	990/10 CPU w/8K	–	944920-0002	-6.95	-0.7	–	3 Full Slots
301	990/10 CPU w/12K	–	944920-0003	-7.02	-0.71	–	3 Full Slots

Table 6-1. 990 Power and Space Requirements (Continued)

Price List Item No.	Description*	Prerequisite Item Number	Part Number	DC Power +5V	+12V	-12V	Space Requirements
302	990/10 CPU w/16K	–	944920-0004	-7.09	-0.72	–	3 Full Slots
303	990/10 CPU w/20K	–	944920-0005	-7.16	-0.73	–	3 Full Slots
304	990/10 CPU w/mapping and 8K	–	944920-0007	-8.55	-0.7	–	3 Full Slots
305	990/10 CPU w/ mapping and 12K	–	944920-0008	-8.62	-0.71	–	3 Full Slots
306	990/10 CPU w/mapping and 16K	–	944920-0009	-8.69	-0.72	–	3 Full Slots
307	990/10 CPU w/mapping and 20K	–	944920-0010	-8.76	-0.73	–	3 Full Slots
308	990/10 CPU w/8K ECC	–	944920-0001	-9.4	-0.9	–	3 Full Slots
309	990/10 CPU w/mapping and 8K ECC	–	944920-0006	-11.0	-0.9	–	3 Full Slots

990/10 Minicomputer On-board Options

Only one of the following options can be mounted on the CPU board. These options supply the necessary 990 instructions in ROM/PROM to allow operation of programmer panel, boot loading, CPU self diagnostics, and/or custom user generated loaders using PROM kits items 312 or 313.

310	733 ASR/Card Reader ROM Loader	300-309	945134-0001	-0.21	–	–	
311	733 ASR/Card Reader ROM Loader w/Self-Test	300-309	945134-0002	-0.42	–	–	
312	PROM Device Kit (256 words)	300-309	945123-0002	-0.21	–	–	

Table 6-1. 990 Power and Space Requirements (Continued)

Price List Item No.	Description*	Prerequisite Item Number	Part Number	DC Power +5V	DC Power +12V	DC Power -12V	Space Requirements
	990/10 Minicomputer Memory Expansion Options						
	The following items are standard 990/10 memory expansion options. These are on one full size board. Memory parity feature (Items 324-327) mount on the associated memory module. Error correcting memory is annotated w/ECC.						
313	990/10 Floppy Disc ROM Loader	300-309	945134-0006	–	–	–	–
314	990/10 Floppy Disc Loader w/self-test	300-309	945134-0007	–	–	–	–
315	Mapping Conversion Kit	300-303, 308	944976-0001	-25	–	–	–
320	8K Memory Expansion Module	300-309	944945-0002	-0.75	-0.02	–	Full Slot
321	12K Memory Expansion Module	300-309	944945-0003	-0.82	-0.03	–	Full Slot
322	16K Memory Expansion Module	300-309	944945-0004	-0.89	-0.04	–	Full Slot
323	20K Memory Expansion Module	300-309	944945-0005	-0.96	-0.05	–	Full Slot
324	8K Memory Parity Feature	300,304,320	945120-0002	–		–	–
325	12K Memory Parity Feature	301,305,321	945120-0003	–		–	–
326	16K Memory Parity Feature	302,306,322	945120-0004	–		–	–
327	20K Memory Parity Feature	303,307,323	945120-0005	–		–	–
328	8K Memory Expansion Module w/ECC	300-309	946655-0002	-3.2	-0.04	–	Full Slot

Table 6-1. 990 Power and Space Requirements (Continued)

Price List Item No.	Description*	Prerequisite Item Number	Part Number	DC Power +5V	DC Power +12V	DC Power -12V	Space Requirements
329	8K Add-On Memory Module w/ECC	308,309,328	945093-0002	-0.34	-0.02	–	Full Slot
330	16K Add-On Memory Module w/ECC	308,309,328	945093-0004	-0.49	-0.03	–	Full Slot
331	24K Add-On Memory Module w/ECC	308,309,328	945093-0006	-0.64	-0.05	–	Full Slot
332	EPROM Memory Module	300-309	945170-0001	-0.4	-0.1	-0.06	Full Slot
333	EPROM Device Kit	332	945123-0004	-0.01	-0.1	-0.06	–
990 Terminal Options							
600	Model 733 KSR Data Terminal Kit	See note	945162-0001	-0.38	-0.02	-0.02	1/2 Slot
601	Model 733 ASR Data Terminal Kit	See note	945161-0001	-0.38	-0.02	-0.02	1/2 Slot
603	TTY/EIA Terminal Interface Module	See note	945075-0013	-0.38	-0.02	-0.02	1/2 Slot
620	Model 913A Video Display Terminal Kit	See note	975070-0013	-1.4	-0.1	-0.14	Full Slot
990 Communications Options							
631	990 Comm Interface Module	See note	945114-0002	-1.0	-0.1	-0.05	1/2 Slot

Table 6-1. 990 Power and Space Requirements (Continued)

Price List Item No.	Description	Prerequisite Item Number	Part Number	DC Power +5V	DC Power +12V	DC Power -12V	Space Requirements
632	990 Asynchronous Modem Kit	631	945104-0003	-0.75	-0.2	-0.2	1/2 Slot
633	Async Bell Data Set Interface Kit	See note	945104-0004	-1.0	-0.1	-0.05	1/2 Slot
634	Pulse Auto Calling Kit	630,638	945164-0001	-1.0	-0.15	-0.05	1/2 Slot
635	Touch Tone Auto Calling Kit	630,638	945163-0001	-1.0	-0.15	-0.15	1/2 slot
638	990 Synchronous Modem Kit	631	945094-0003	-0.75	-0.2	-0.2	1/2 Slot
990 Mass Storage							
650	990 Floppy Disc Kit	See note	945082-0001	-3.0	-0.2	-0.01	12-1/4 X 15, Full Slot
651	Floppy Disc Interface Kit	See note	945082-0002	-3.0	-0.2	-0.01	Full Slot
652	Secondary Floppy Disc Unit	650,653	945082-0003	–	–	–	–
653	990 Floppy Disc Expansion Kit	650	945082-0004	–	–	–	12-1/4 X 15

Table 6-1. 990 Power and Space Requirements (Continued)

Price List Item No.	Description	Prerequisite Item Number	Part Number	DC Power +6V	DC Power +12V	DC Power -12V	Space Requirements
655	Model DS31 Disc Master Kit, M.H. Removable	300-309	945242-0001	-4.5	–	–	7 X 24-1/8, Full Slot
656	Model DS32 Disc Master Kit, M.H. Nonremovable	300-309	945242-0005	-4.5	–	–	7 X 24-1/8, Full Slot
657	Model DS31/32 Disc Interface Only Kit	300-309	945242-0002	-4.5	–	–	Full Slot
658	Model DS31 Disc Secondary Kit	655,656	945242-0003	–	–	–	7 X 24-1/8
659	Model DS31 Secondary Unit With Power Supply	658,660	945242-0004	–	–	–	7 X 24-1/8
660	Model DS32 Disc Secondary Kit	655,656	945242-0006	–	–	–	7 X 24-1/8
661	DS32 Second Secondary With Power Supply	658,660	945242-0007	–	–	–	7 X 24-1/8
990 Peripherals							
670	Model 306 Impact Line Printer Kit	See note	945113-0001	-0.38	-0.02	-0.02	1/2 Slot
671	Model 306 Printer Interface Kit	See note	945113-0002	-0.38	-0.02	-0.02	1/2 Slot
677	Model 588 Line Printer Kit - 5 X 7	See note	945112-0001	-0.38	-0.02	-0.02	1/2 Slot

Table 6-1. 990 Power and Space Requirements (Continued)

Price List Item No.	Description*	Prerequisite Item Number	Part Number	DC Power +5V	DC Power +12V	DC Power -12V	Space Requirements
678	Model 588 Line Printer Interface Kit	See note	945112-0002	-0.38	-0.02	-0.02	1/2 Slot
679	Model 588 Line Printer Kit - 9 × 7	See note	945112-0004	-0.38	-0.02	-0.02	1/2 Slot
686	804 Card Reader Kit	See note	945083-0001	-0.6	–	–	1/2 Slot
687	Card Reader Interface Kit	See note	945083-0002	-0.6	–	–	1/2 Slot
690	PROM Programming Kit (Tabletop)	See note	944924-0001	-0.5	–	–	1/2 Slot
691	PROM Programming Kit (Rack Mount)	See note	944924-0002	-0.5	–	–	5-1/4 × 15, Slot
692	PROM Programming Adapter	690,691	945135-0001	–	–	–	–
693	EPROM Programming Adapter	690,691	945165-0001	–	–	–	–
695	16 I/O EIA Data Module	See note	945140-0001	-0.34	-0.08	-0.04	1/2 Slot
696	16 I/O TTL Data Module w/Int Option	See note	945145-0001	-0.53	–	–	1/2 Slot

*Memory storage is expressed in **Number of 16-bit Words.**

NOTE:

The various CPU, memory I/O expansion, and peripherals require necessary chassis, power supply, and packaging options. Consult Texas Instruments for system configuration guidelines to ensure that all prerequisite hardware items are included in system design.

Table 6-2. Memory Standby Power Requirements

				+12 MEM		
		+5 MAIN	+5 MEM	Operating	Standby	-5MEM
990/4 Expansion Memory	4K	0.5	0.20	0.5	0.008	0.002
	8K	0.5	0.27	0.51	0.016	0.002
	12K	0.5	0.34	0.52	0.024	0.002
	16K	0.5	0.41	0.53	0.032	0.002
	20K	0.5	0.48	0.54	0.040	0.002
990/4 Write Protect Option Adds		0.05	–	–	–	–
990/10 Expansion Memory	8K	0.5	0.25	0.7	0.015	0.002
	12K	0.5	0.32	0.71	0.024	0.002
	16K	0.5	0.39	0.72	0.032	0.002
	20K	0.5	0.46	0.73	0.040	0.002
990/4 with onboard 4K dynamic RAM with parity		0.8	0.32	0.6	0.1	0.001

Table 6-3. Error-Correcting Memory Standby Power

		+5 MEM		+12 MEM		
	+5 MAIN	Operating	Standby	Operating	Standby	-5 MEM
8K Module (1 Card)	2.5	0.7	0.014	0.9	0.04	0.002
16K Module } 2 Cards	3.7	0.92	0.018	0.915	0.055	0.004
24K Module } 2 Cards	4.2	1.02	0.020	0.930	0.070	0.004
32K Module } 2 Cards	4.7	1.12	0.022	0.945	0.085	0.004

6.3.3.2 Software Implementation. Software implementation is presented in the following paragraphs:

Software Definitions. Software has become a general term that includes all the programs required for the operation of a computer system. Sometimes a qualifier is used such as "operating system software" or "application system software" to clarify the meaning, but when the software for a computer system is examined it will generally be found to consist of eight different types of programs.

- Operating System
- Application Programs
- System Initial Loaders
- Performance Assurance Tests
- Peripheral Demonstration Tests
- Function Demonstration Tests
- General Utility Programs
- Development Programs.

Close examination reveals that each category may contain several programs and that each program can be sub-divided into many modules and that each module is made up of many individual computer instructions. The composition of instructions into modules and modules into programs is known as programming and those who practice this discipline are called programmers. The selection of programs required to complete a system or to perform a given task is often done by a system analyst who also may be a programmer.

These categories are described as follows:

1. *Operating System.* A collection of programs that schedules operation of other programs in the computer, performs input/output operations and provides common service functions used by application program.

   TX990 and DX10 are operating systems offered by Texas Instruments to reduce development cost of new computer applications. Additional information on TX990 and DX10 can be found in Section IV.

2. *Application Programs.* Programs that perform specific application tasks. In general, these are the programs that must be written to obtain a functional computer system. Communication software like the synchronous and asynchronous communications subsystems described in Section IV are largely

operating system programs. However, applications software is also required in these systems for message handling and distribution.

3. *System Initial Loaders.* Programs that reside in ROM and provide a means for transferring programs from an external device into the memory of the computer. ROM is unaffected by power outages or software malfunctions and is an ideal way to preserve programs like the initial loader that must be in place and ready to operate when the system power is turned on.

4. *Performance Assurance Tests.* Programs that test the computer and computer memory to prove correct operation of the computer.

5. *Peripheral Demonstration Tests.* Programs that prove correct operation of peripheral equipment attached to the computer.

6. *Function Demonstration Tests.* Programs that prove correct operation of system programs. A method of testing each on-line program in the system should be planned. Functional demonstration tests are also used to prove operation of a program after modifications or improvements have been made.

7. *General Utility Programs.* Programs that perform helpful functions not necessarily related to on-line operations of the computer. A program that copies data records from one peripheral device to another is a good example of a general utility program.

8. *Development Programs.* Programs that are used in the process of software development assemblers, editors, compilers and debug programs are examples of development programs.

Programming Languages. Programming languages are used for writing computer programs. These languages include:

- Machine Language
- Assembly Language
- High Level Languages

    1. FORTRAN
    2. COBOL
    3. BASIC.

Use of machine language is the most fundamental form of computer programming and requires detailed knowledge of the computer. The computer instructions are encoded step by step in binary form. Programming in this fashion is both tedious and prone to error. Use of Machine Language is generally not cost effective for programs longer than a hundred instructions or so.

Assembly language allows the computer instructions to be expressed in symbolic terms. Programs written in assembly language are processed by a computer program called an assembler to produce machine instructions. Assembly language programs are also encoded on an instruction by instruction basis and require the programmer to have detailed knowledge of the computer. Assembly language programs are potentially the most concise form of a given program. Assembly language also provides practical features for documentation and for segmenting programs into program modules. Well documented program modules can often be used again in other applications.

High level languages offer the most expedient method for developing programs rapidly for computer applications. FORTRAN IV is a well-established language for scientific applications. COBOL is widely used in commercial applications. BASIC is widely used in interactive systems for a variety of purposes.

A program written in FORTRAN is processed by a computer program called a FORTRAN compiler to produce a machine level program. Fewer programmer coded statements are required to write a program in FORTRAN than are required to write the same in assembly language, and the programmer is much less concerned about the details of individual computer instructions. FORTRAN programs at the machine level will tend to be somewhat less concise than the same program written in assembly language, but the FORTRAN program will cost less to develop.

Part of the FORTRAN language program cannot be processed directly by the FORTRAN compiler and is deferred until the program is run on the computer. A special set of programs called a FORTRAN run time package is used to perform the processing deferred by the compiler. In DX10 the runtime package can be shared by all the FORTRAN programs used in the system. A library of FORTRAN library functions and subroutines is also provided.

Programs written in COBOL are processed by a COBOL compiler. The COBOL program is converted to an encoded form. The encoded COBOL program is run on the computer by an interpreter that decodes each program step and performs the required functions. In DX10 the interpreter is shared by all COBOL programs in the system. Use of COBOL in commercial applications will significantly reduce programming costs.

Programs written in BASIC are stored in the computer in compressed source form and are interpreted directly by another program called a BASIC Interpreter. The original source program can be recovered from storage at any time for inspection or modifications. This can be a significant advantage in systems that are changed frequently. As the name implies, BASIC is an easy language to learn, and BASIC programs are easy to write. BASIC is an interactive, terminal-oriented language.

More information on FORTRAN, COBOL and BASIC is presented in Section IV.

Software Planning. All practical computer systems require software for their operation. Software plans are usually made in parallel with equipment plans and definition of the functions to be performed by the system. Planning should include development programs and equipment to support the analysts and programmers who put the system together. Texas Instruments offers support systems for programming development. Choice of a support system will be determined largely by the scope of the programming project. Test software should also be planned early in the project.

Software maintenance is a task that is easily overlooked in the development stage of a project. Maintenance is simplified when good documentation test programs and test procedures are developed along with the system software. The development software and equipment required for implementing program changes should be accounted for in project planning.

The actual writing of a program is less than half of the complete programming job. The new program must also be corrected (debugged), tested, and documented. Debugging is a tedious and time consuming task, which is simplified by a special debug program used to assist the programmer. The debug programs offered with Texas Instruments development systems and the 990 prototyping system are described in Section IV.

Testing assures the accuracy and proves the performance of the program. Documentation provides the information required to use the program in the present system and perhaps in other systems as well.

Support Systems. Development Support systems described in detail in Section IV include:

1. IBM System/3X0 Cross Support
2. 990 - 733 ASR System Software
3. Prototyping System
4. DX10 Disc System Software.

Cross Support may be operated in batch mode and in a time sharing mode. Batch operation is generally useful when the cross support package is installed in an in-house system. The timesharing mode is useful when each programmer can have convenient access to a timesharing terminal. Timesharing operation with the 733 ASR (operating at 300 bps) also provides convenient cassette medium for transporting programs to the "target" system. (A computer system under development is often called a "target" system.)

Batch operation of the Cross Support package will likely result in punched cards as the program transfer medium.

Turnaround time, the time from programmer's input to system output, is a potential problem in batch operation of the cross support programs and should be considered.

The Cross Support simulator is designed specifically for the TMS9900 microprocessor and the 990/4 Microcomputer. Almost all operations are equivalent to 990/10 Minicomputer operations, but none of the 990/10 unique characteristics are provided in the simulator.

The 990 - 733 ASR System Software and the 990 prototyping system provide essentially the same support software. The 990 - 733 ASR System Software is packaged for operation on the 990/4 and utilizes unique features of the 990/4 to specifically support implementation of TMS9900 microprocessors and 990/4 microcomputer systems. Additional information on these can be found in Section IV.

The 733 ASR magnetic tape cassette units operate at a maximum rate of 120 characters per second and the 733 ASR printer operates at 30 characters per second. Utilization of the system for program debugging should also be taken into account when estimating the utilization of one of these systems during program composition. For a rule of thumb about two programmers should be able to conveniently share a system.

The DX10 software development system is packaged for the 990/10 with the memory mapping option used offers both improved performance and support programs designed expressly for operating with the high-speed Model 913 VDT. This is the principal system used by Texas Instruments for development and maintenance of the software for the 990 Computer family. The DX10 operating system with memory protection mapping allows the software development program like the macro assembler and debug program to be coresident with an application system under development.

The development programs utilize the 913 VDT as the principal input and display device. Listings and printed results are available on the 733 ASR printer at 30 characters per second or one the optional line printer at 80 characters per second. Printing the entire listing can often be avoided altogether during early stages of program composition by saving the listing data on a disc file and using the 913A VDT to display the appropriate portions for inspection. As a practical matter, listings and results will eventually be used during debugging and for documentation.

The DX10 development system also allows utilization of the high level language compilers for program development. FORTRAN, COBOL and BASIC are recommended for development of appropriate applications programs for DX10 system.

A DX10 development system is often the best solution for developing programs for smaller target systems. Programs can be transported to the target system on 733 ASR magnetic tape cassettes or where communications equipment is part of the target

system, customer system loaders can be implemented and used to download programs from the DX10 system to the target system. Transferring a program from a host computer to a remote computer in a dispersed network via a communications circuit is often referred to as downloading, especially if the program is the entire system program of the target computer. This is especially effective when program downloading is required for operation of the system under development.

Software for the Target System. Texas Instruments offers operating system software and communications subsystem software for use in target systems.

The DX10 and TX990 operating systems and the communications subsystems are modular in nature and can be customized to fit the purposes of a variety of applications. DX10 is provided with an interactive program to further simplify manipulation of standard options like the number and type of peripherals used in the system. Practical TX990 systems consisting of only the task scheduler and interrupt decoder can be generated for small systems. Large systems can include DX10 compatible input/output and floppy disc file management. The communications subsystems can be used with either TX990 or DX10. Additional information on the operating systems and communication subsystems can be found in Section IV.

Small, dedicated target systems may not require an operating system at all. The workspace concept of the 990 combined with priority interrupts and line frequency clock provide a simple real time environment for operating a small number of application tasks directly without the aid of software schedules. Simple, elegant systems of this type are very practical in the 990/4 microcomputer.

Figure 6-1 shows the evolution of a typical TMS9900 target system from initial evaluation to maintenance and offers guidelines for selecting development software.

Program Loading Operations. When a program is in operation it must reside in the main memory of the computer. The act of placing the program into main memory is called loading the program. Disc systems like DX10 can change programs in memory several times a second. Programs in other systems may be memory resident at all times and seldom change except when revisions are made. Some programs designed for special purposes and encoded in ROM are never changed.

RAM is used for program storage where the program is changed regularly. PROM and EPROM is suitable where infrequent changes are required. The contents of PROM and EPROM are unaffected by loss of power. Use of masked ROM is applicable where the program is very seldom, if ever, changed.

Computer systems using RAM memory for program storage will generally be loaded from some external device such as a disc, 733 ASR magnetic tape cassette unit card reader, or from another computer via a communications link. Programs in RAM memory can be erased in a variety of ways and on initial system loader or control program must always be included in the system. The initial system loader or bootstrap will typically reside in some type of ROM.

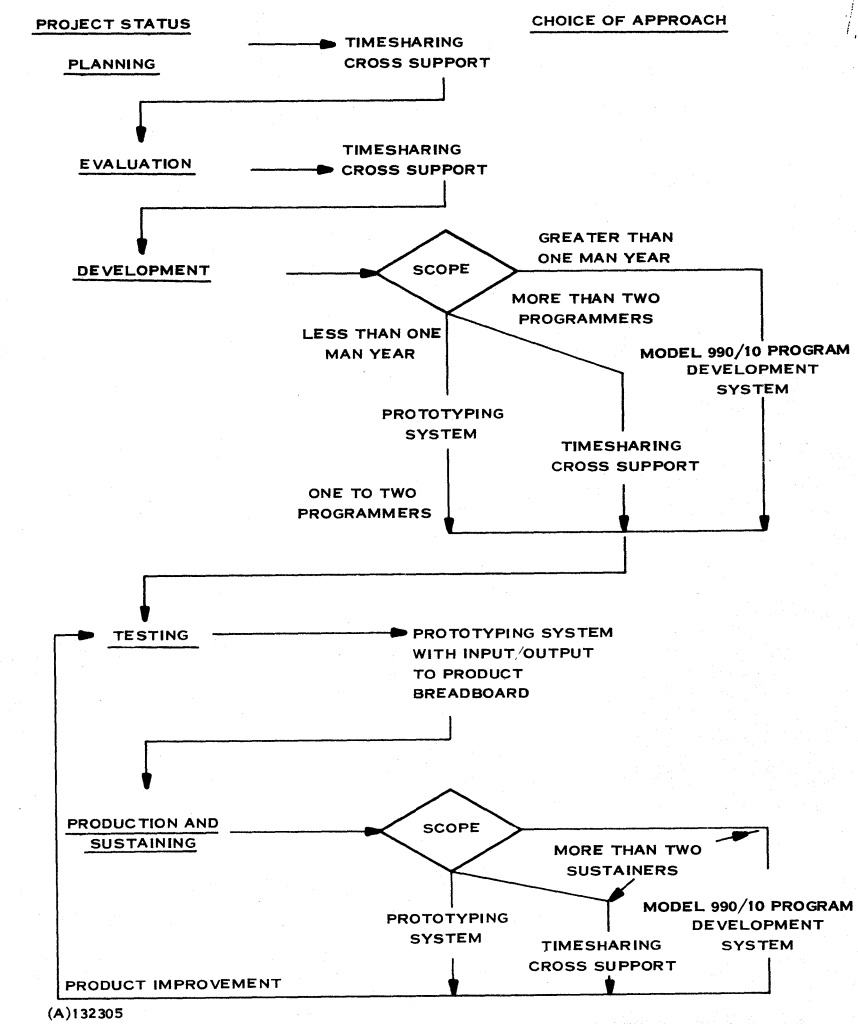


Figure 6-1. TMS9900 Microprocessor Software Development

The standard bootstrap loader of the 990/4 and 990/10 are ROM or PROM devices. Other loaders or functions can replace the bootstrap loader. For example, when the entire program resides in some type of ROM there is no need for loader as such and the load function can be used to initiate another system oriented function, for example, to start the system in operation.

ROM, PROM and EROM cannot be changed in place. In addition a RAM of some type must always be provided in 990 systems for workspaces and data storage.

The 990 Prototyping Systems optionally provides the capability for transferring programs into PROM and EROM devices. The PROM programming option is required. The personality card for the specific device used is also required.

The program to be transferred to the PROM or EROM device is first loaded into the prototyping system RAM memory and there copied into the PROM or EROM device. The PROM or EROM device is then removed from the PROM programmer and installed in the memory of the system where it is used.

6.3.3.3 Software Implementation Summary. It is fairly common for a designer to implement a projected computer system entirely in existing hardware modules. It is extremely rare to implement the software entirely from existing software modules. Usually the software will consist of a mixture of existing modules and custom modules generated specifically for the application. Support packages, such as the 990 series, cross support and software development systems can normally be used without modification. Application programs will normally be custom modules and hence the designer's output in the planning phase will include an estimate of the manpower required to develop these modules. The size of the projected modules can be estimated by identifying existing software modules of comparable complexity. Comparisons should not be extended upward or downward in complexity because program size and development time increase nonlinearly with increasing complexity. The size can be directly applied to memory size requirements and where development time of the existing module is known it can be directly applied to the manpower estimate. Where development time is unknown it can be estimated by applying an industry rule of thumb of 10 lines of code per man day, where documentation and debug time are included in the estimate. Using assembly language, one line of code corresponds to one instruction.

The total output from software implementation thus consists of:

- Manpower estimate to generate special software
- Software support system requirements at every phase of the project
- Specification of existing software modules required
- Memory requirements size and type (ROM/RAM).

6.3.4 COST MODEL. The completion of the implementation step demonstrates the technical feasibility of the project. The final step in the planning phase is to develop a cost model with the objective of determining economic feasibility.

The first step is to review the preliminary schedule and to revise it if indicated by later developments. The cost model should be developed against this schedule, totaling the costs for each phase separately and in case of an extended phase such as a long production run the costs should be further segmented in time. The result is a cost versus time curve which, when combined with projected income versus time defines the level of expenditure or net income at every phase of the project. In turn, the comparison of expenditures to resources demonstrates the economic feasibility; comparison of expenditures to returns demonstrates the desirability.

6.4 SAMPLE IMPLEMENTATION

Preceding paragraphs have presented a complete description of the considerations a designer faces each time he implements a computer system, but because it is thorough it is also quite lengthy and conveys an impression that computer system design is very complicated. In practice, computer design with standard 990 series modules is very straightforward and an experienced designer, with materials readily available, can design a simple system (planning phase) in one day.

The following two examples illustrate the key steps in implementation of (1) a simple hardwired controller, and (2) an intelligent terminal system of medium complexity.

Example 1: Hardwired Controller Implementation

Assume that a designer's task is to design a hardwired controller to be built into a larger machine. The controller is to interface through two parallel 16-bit input words and one parallel 16-bit output word. All signal levels are TTL logic. The control function is to read both input words five times per second and to output the sine of the product. (Any other control function could be implemented and the only effect would be to increase memory size for complicated functions). Further assume that +5, +12 and -5 volts are available in the machine. The projected volume is 20 units per year and a 990 prototyping system is available to the project.

Note that the configuration can be implemented with standard 990 modules - a 990/4 CPU and two 16 I/O data modules. The volume is so low that the non-recurring from a custom circuit board layout would never be recovered and custom layout will therefore not be considered.

The block diagram for this system is shown in figure 6-2. In this example the functional block diagram and detailed hardware block diagram are identical.

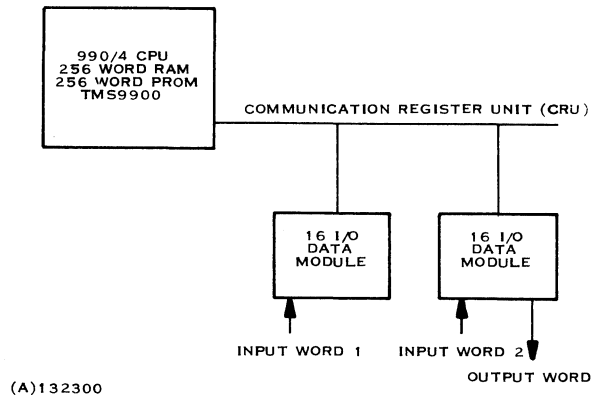


Figure 6-2. Hardwired Controller Block Diagram

The OEM chassis will hold the 990/4 CPU and two data modules which totals two full slots, leaving one spare slot. There are terminals on the chassis for system power connection and power up sensing circuit (RESET) which will cause the 990 to trap through memory location 0000 (or FFFC) when the +5 V power is applied. At this point it is necessary to develop the software implementation to determine memory size and type.

The 990/4 has a multiply instruction and the sine function subroutine is approximately 50 words so that it is clear that the software will fit in the minimum memory size of 256 words. The software will be contained in a 256 words of PROM memory which can easily be developed and debugged on the 990 prototyping system. The application program is "loaded" by plugging the programmed PROMS into the CPU. Since PROMS are non-volatile, no standby power is required. There must be 256 words of static RAM for writeable workspace provided. Figure 6-3 shows the OEM chassis backpanel. For this example +5 MAIN and +5 MEM will be jumpered together.

Figure 6-4 is a map of the memory, showing start address locations for the PROM and RAM memory modules.

Table 6-4 lists material identifying the parts and quantity of each. Part numbers are obtained from the 990 price list.

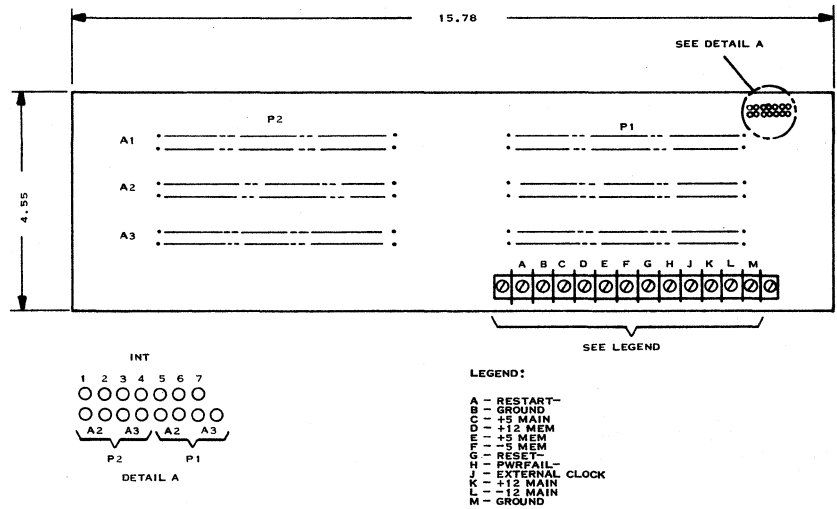


Figure 6-3. OEM Chassis Backpanel

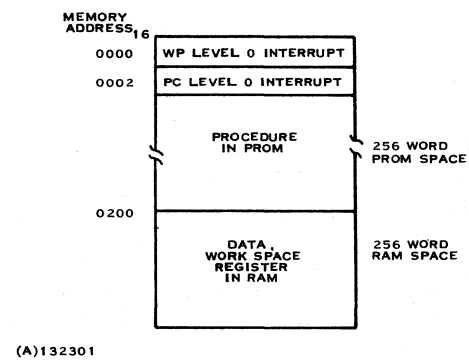


Figure 6-4. Memory Map - Hardwired Controller Application

Table 6-4. List of Materials

Item	Description	Part Number	Qty
200	990/4 CPU with 256 words static RAM	944910-0001	1
208	PROM Device Kit	945123-0001	1
400	3-Slot OEM Chassis	945040-0001	1
696	16 I/O TTL Data Module	945145-0001	2

Table 6-5 presents the power requirements for this system.

Table 6-5. Power Requirements

Item No.	Description	+5 V	+12 V	-12 V	-5 V
200	990/4 CPU with 256 words static RAM	-1.25	-0.04	–	-0.001
208	256 Word PROM	-0.45A	–	–	–
696	16 I/O TTL Data Module	-0.53A	–	–	–
696	16 I/O TTL Data Module	-0.53A	–	–	–
		-2.76	-0.04		-0.001

Figure 6-5 shows the layout of the 990/4 CPU with the memory devices and jumper wires installed.

Example 2: Intelligent Terminal System

This second example presents the key figures and tables in the implementation of an intelligent terminal system containing two video displays with keyboard, a line printer, and an asynchronous communication line with built-in modem and autocall capability.

Figure 6-6 shows the typical layout for an intelligent terminal system.

A functional block diagram for an intelligent terminal system is presented in figure 6-7. A detailed hardware block diagram is shown in figure 6-8.

Tables 6-6, 6-7 and 6-8 present the list of materials, primary power requirements, and secondary power requirements, respectively.

Figures 6-9 and 6-10 show chassis layout and memory map for an intelligent terminal, respectively.

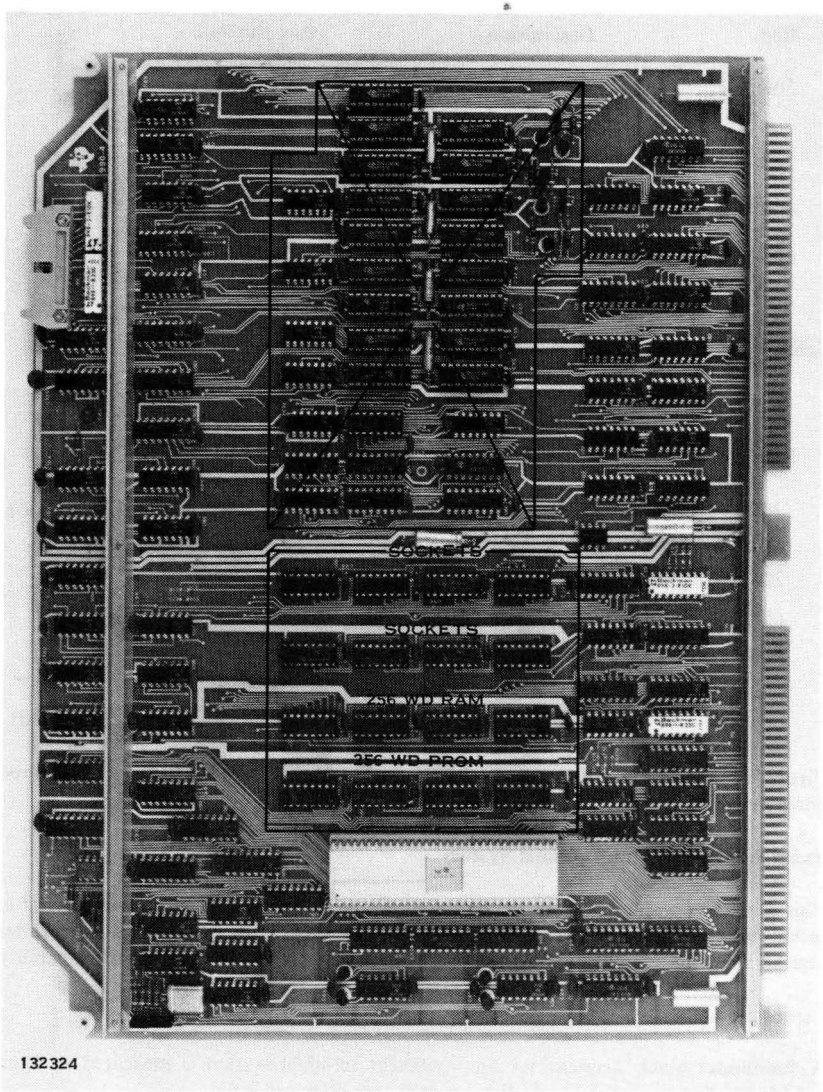


Figure 6-5. 990/4 Circuit Board Options

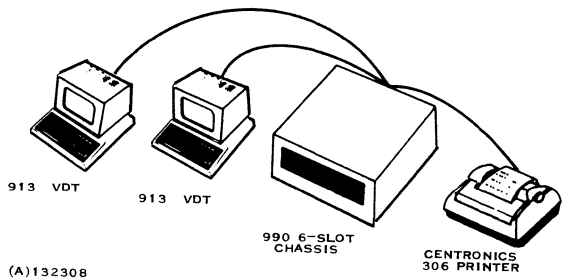


Figure 6-6. Intelligent Terminal System

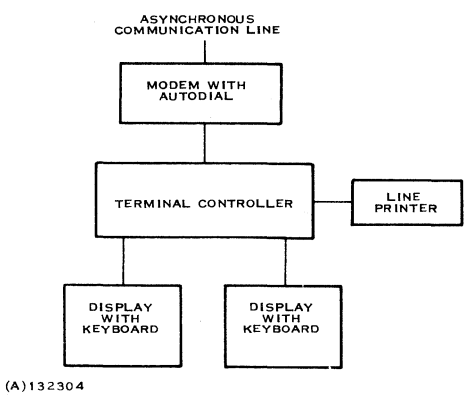


Figure 6-7. Functional Block Diagram for Intelligent Terminal

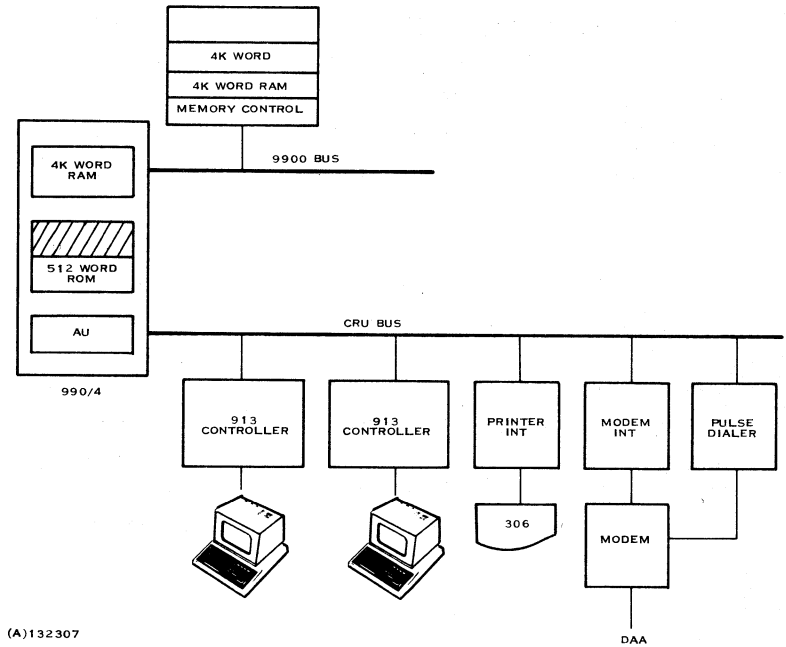


Figure 6-8. Detailed Hardware Block Diagram for Intelligent Terminal

Table 6-6. List of Materials for an Intelligent Terminal

Item	Description	Part Number	Qty
401	6-Slot Chassis with Operator Panel	944960-0002	1
407	Tabletop Chassis Option	945126-0001	1
410	Standby Power Supply	945128-0001	1
201	990/4 CPU (4K Dynamic RAM)	944910-0002	1
202	Dynamic RAM Parity Option	945120-0006	1
208	PROM Device Kit	945123-0001	2
221	8K Memory Expansion Module	944935-0002	1
231	8K Memory Parity Option	945120-0002	1
620	Model 913A Video Display Terminal Kit	975070-0013	2
670	Model 306 Impact Line Printer Kit	945113-0001	1
631	990 Asynchronous Communication Interface	945104-0001	1
632	990 Asynchronous Modem Kit	945114-0003	1
634	Pulse Auto Calling Kit	945164-0001	1

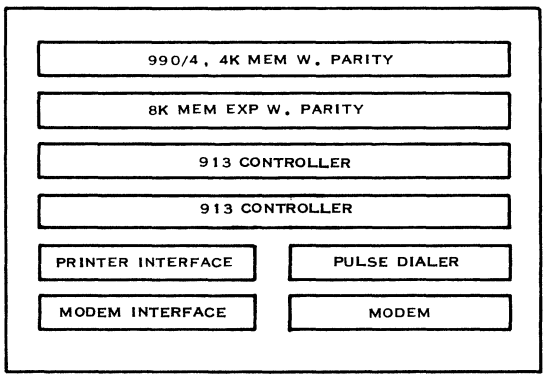
Table 6-7. Primary Power Table for an Intelligent Terminal

Unit	Power
6-Slot Chassis Power Supply	300 VA
Standby Power Supply	60 VA
913 VDT	100 VA
913 VDT	100 VA
Line Printer	400 VA
Total Primary Power 115 V, 48 - 62 Hz	960 VA

Table 6-8. Secondary Power Tables for an Intelligent Terminal

Chassis Power Supply	+5 MAIN	+12 MAIN	-12 MAIN
6-Slot Chassis, Operator Panel	+20.0	+2.0	+1.0
990/4 CPU w/4K RAM	-0.8	-0.04	–
512 Word PROM (Downloader and Self-Test)	-0.90	–	–
			–
8K Memory Expansion	-0.50	–	–
913 VDT Interface	-1.40	-0.10	-0.14
913 VDT Interface	-1.40	-0.10	-0.14
Line Printer Interface	-0.38	-0.02	-0.02
Asynchronous Communication Interface	-1.25	-0.07	-0.03
Built-in Modem	-1.00	-0.03	-0.02
Pulse Autodial	-1.00	-0.15	-0.05
Balance (+ Indicates Spare Capacity)	+11.37	+1.49	+0.6

Standby Power Supply	+5 MEM	+12 MEM	-5 MEM
Standby Power Supply	+1.40	+1.20	+0.10
Onboard 4K RAM	-0.32	-0.60	-0.001
8K Memory Expansion	-0.27	-0.51	-0.002
Balance (+ Indicates Spare Capacity)	+0.81	+0.09	+0.097



(A)132303

Figure 6-9. Chassis Layout for Intelligent Terminal

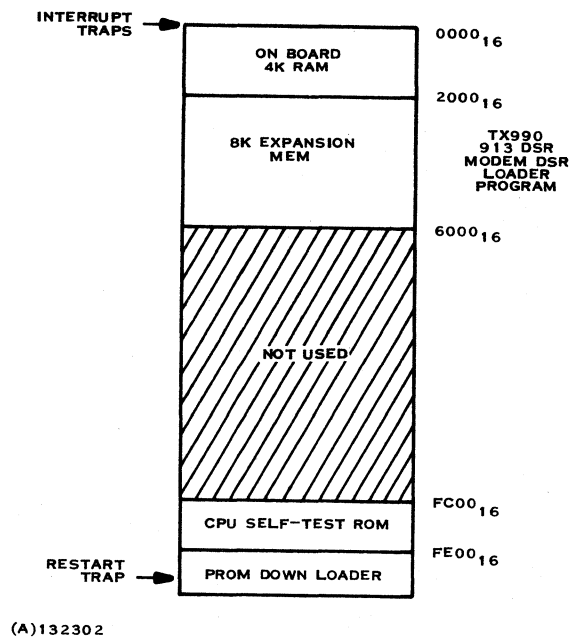


Figure 6-10. Memory Map for Intelligent Terminal

SECTION VII

CUSTOMER SERVICE AND SUPPORT

7.1 INTRODUCTION

The Texas Instruments 990 Family of Computers incorporates ease of operation with dependability of performance. The powerful instruction set and host of peripheral devices make the Computer easily adaptable to customer requirements. Texas Instruments years of semiconductor and computer expertise ensure a highly reliable product. However, should problems occur with applications, interfacing or hardware, Texas Instruments provides the type of support that customers expect from a leader in the industry.

Recognizing that equipment availability is the important factor in any computer installation, Texas Instruments has designed 990 Computer systems for rapid field repair. The field Customer Engineer can utilize simplified troubleshooting procedures and built-in diagnostic firmware to quickly isolate machine failures to an easily replaceable assembly. Replacement of the faulty assembly then allows him to return the system to productive operation in the shortest possible time. Once the system is operating properly, the faulty assembly can be returned to Texas Instruments manufacturing facility for detailed analysis and repair.

This designed-in repairability demonstrates Texas Instruments concern for providing superior products and service, but that concern does not stop with the design and sale of the computer system. High quality customer service; up-to-date, end-user oriented documentation; readily available customer support and training; plus a user information exchange organization combine to ensure optimum system performance and complete customer satisfaction.

7.2 CUSTOMER SERVICE

Texas Instruments maintains a responsive, nationwide service organization dedicated to customer support. Trained personnel, facilities and resources provide the 990 Family of Computers with the same exacting attention after their sale as was given them during manufacture. The service available to 990 Computer users ranges from on-the-spot repairs made by an experienced Customer Service Engineer to the new and innovative TI-CARE* program.

7.2.1 TI-CARE. Texas Instruments Computer Aided Repair Effort (TI-CARE) is a unique answer to national account service requirements. With a single location to call, real-time computer dispatching and online computer diagnostic assistance, TI Customer Service Engineers provide the service needed to maximize customer uptime. TI-CARE is available on a contract basis to all Texas Instruments systems users.

*Service Mark of Texas Instruments Incorporated.

When the customer calls the TI-CARE center in Houston, a dispatcher receives the call and enters the pertinent system data into the computer via a video display terminal (see figure 7-1). The call is then transferred to a TI-CARE technician who discusses the problem with the customer in an attempt to clear the problem or determine if remote diagnostics are appropriate. If required, the technician instructs the TI-CARE computer to dispatch a Customer Service Engineer from the district service office and to advise the Customer Service Engineer of the probable cause of the malfunction. The district service office, in turn, notifies the TI-CARE center of the estimated time of arrival at the customer site and also informs the center when the service call has been successfully completed. Should the Customer Service Engineer require additional assistance, the TI-CARE center immediately sets up a conference call with skilled engineers at the manufacturing facilities in Austin, Houston or Dallas.

Figure 7-1. TI-CARE Dispatcher Enters System Data into the Computer

Depending upon the terms of individual service contracts, the TI-CARE system also offers complete management reporting of critical repair and response parameters such as mean-time-between failure (MTBF), mean-time-to-repair (MTTR), dispatch and arrival times, etc. These reports enable Texas Instruments Service management and individual customers to monitor the performance of the maintenance system. If contract commitments are not being met, the system generates flag reports to alert Service Managers to take immediate corrective action.

7.2.2 CUSTOMER SERVICE ENGINEER. The Texas Instruments Customer Service Engineer is a skilled professional. He has a complete background of technical education, formal training on Texas Instruments equipment, and experience in solving customer problems in the field. To supplement this background, Texas Instruments provides him with a continuing educational program to keep his technological abilities up-to-date. The Customer Service Engineer knows TI equipment thoroughly and how to keep it operating properly. Should he encounter a problem that he cannot speedily solve, TI has a staff of regional and national Technical Specialists as well as engineering support groups at the manufacturing facilities that provide him with the professional counsel required to solve detailed technical problems.

7.2.3 CUSTOMER SERVICE VANS. At many of its local service offices, TI operates station wagons or vans that are completely equipped with test equipment and replacement parts. These vehicles permit Customer Service Engineers to quickly make at-the-site tests and repairs to maximize equipment uptime (see figure 7-2).

7.2.4 FIELD SPARE PARTS INVENTORY. Texas Instruments maintains extensive spare parts inventories both at local service offices and at the national parts support depot. This policy allows most spare parts requirements to be filled directly and quickly from a local inventory. An automated system tracks parts usage throughout the nation and keeps local inventories at the proper levels. However, if a spare parts emergency arises, a computer-assisted search can locate any required part at the nearest service center in order to expedite repairs.

7.2.5 MAINTENANCE AGREEMENT. In many cases, the most cost-effective approach to maintenance is through a Texas Instruments Maintenance Agreement. The basic agreement provides service during any consecutive 9-hour period between 7 a.m. and 6 p.m. (local time), Monday through Friday excluding TI holidays. Optional coverage up to 24 hours per day, 7 days per week is available. Resident Engineers are also available for critical requirements.

7.3 CUSTOMER BULLETIN

The **Customer Bulletin** is a periodical publication of Texas Instruments that keeps its computer customers informed of the factory support and services available to meet the customers' data processing requirements. This publication includes descriptions of hardware and software improvements, and how to obtain these advancements for use on current systems. It also describes availability of new manuals, revisions to existing

Figure 7-2. Another Service Call Complete, the Customer Service Engineer Returns to His Mobile Service Van

manuals, and changes to the available peripheral documentation kits. Texas Instruments provides this unique service without charge to every computer customer that requests it. An order card included with each computer shipment invites the customer to take advantage of this opportunity.

7.4 CUSTOMER SUPPORT LINE

The Customer Support Line is a telephone number that provides computer customers with a single point of contact for technical assistance. This support service consists of a selected group of senior engineers and programmers with extensive experience in all computer system products. This capable staff answers complex technical questions and makes available the expertise of the entire manufacturing facility to solve data processing problems. On-site diagnostics and repair by TI Customer Service Engineers is the most efficient and expedient means of maintenance; however, for those rare

situations requiring the factory engineer's assistance, the Texas Instruments Customer Support Line provides the answer.

7.5 CUSTOMER TRAINING

Texas Instruments maintains a fully staffed and supported training facility at its plant site in Houston, Texas. Classes are scheduled regularly in both software programming and hardware maintenance for all TI computer systems. The courses consist of both classroom lectures that employ the latest educational techniques (see figure 7-3) and laboratory instruction that gives the student first-hand experience with the equipment. Included in the student fees are a complete set of appropriate manuals and other instructional material. Students at the training center must provide for their transportation, lodging and meals.

Before attending any of the programming courses at the training center, the student should be familiar with the fundamentals of programming in either a machine or assembly language. Similarly, students in hardware maintenance courses should know the basic principles of digital logic and computer electronics. With the appropriate background the student can reap maximum benefits from the course material.

Courses offered for the 990 Family of Computers cover topics such as an overview of the Family and its architecture, machine and assembly language programming, programming under a specific operating system, and hardware maintenance courses for

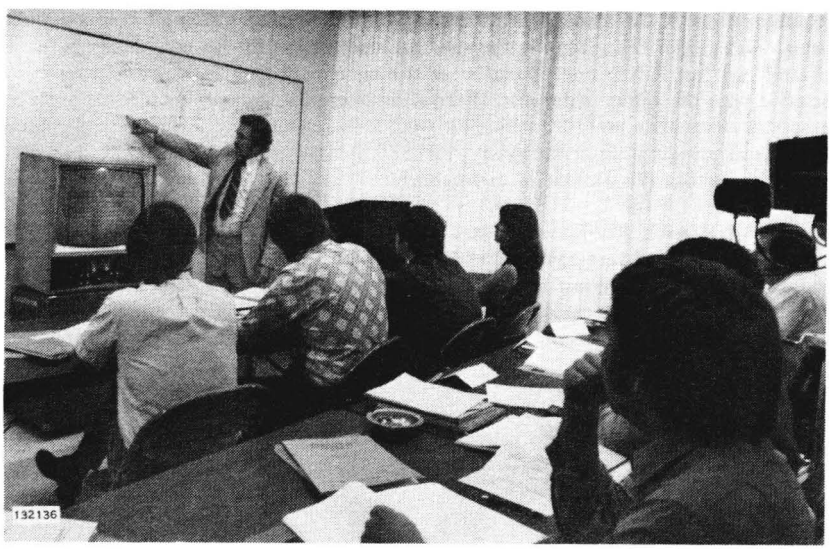


Figure 7-3. Classroom Training Sessions Combine Both Video Tape and Instructor Presentations

the computers and their peripheral interfaces. Consult your TI Sales Representative for exact information regarding each course, when it is offered, and how to register for a course.

7.6 TEXAS INSTRUMENTS MINICOMPUTER INFORMATION EXCHANGE (TIMIX)

TIMIX is an organization of users of Texas Instruments Incorporated minicomputer systems. It was established in 1973 to provide a vehicle for exchange of ideas and information pertaining to applications for and development of Texas Instruments computer systems. To accomplish this goal, TIMIX publishes a bi-monthly magazine *(TI-MIXER)* that contains letters, papers and articles from members describing recent activities in the field. TIMIX also maintains a software library for distribution of member-donated programs to other members at a nominal reproduction cost. A complete catalog describes the programs available through the library. To encourage closer contact among the members, TIMIX sponsors a semiannual symposium so that members can discuss problems and new developments with other Texas Instruments minicomputer users.

Membership in this unique organization is available to owners or users of Texas Instruments minicomputers. A card included with each minicomputer shipment invites the user to participate. No membership fees or dues are required.

7.7 PUBLICATIONS

Texas Instruments supports the 990 Family of Computers with a comprehensive array of publications. These manuals provide complete and verified information in a concise format that guides the reader in the use and maintenance of his equipment. Manuals required for installation and operation of the equipment and associated software are included with the system shipment. Detailed maintenance manuals are also available to those customers that perform their own equipment maintenance. Table 7-1 illustrates the manual structure that documents the 990 Family of Computers. The following paragraphs outline the content of each manual.

7.7.1 HARDWARE REFERENCE MANUALS. Each of these manuals, describes the operation of the hardware components in the applicable processing unit. The manual includes interface descriptions, technical descriptions, installation instructions, electrical characteristics and other essential data concerning the processor and associated chassis that is installed in your system. This manual accompanies the shipment of the applicable system.

Table 7-1. 990 Computer Family Manual Set

Manual Title	Part Number
990 Computer Family Systems Handbook	945250-9701
Software Manuals	
Model 990 Computer Diagnostics Handbook	945400-9701
Model 990 Computer TMS9900 Microprocessor Assembly Language Programmer's Guide	943441-9701
Model 990 Computer Programming Card	943440-9701
Model 990 Computer DX10 Operating System Programmer's Guide	945257-9701
Model 990 Computer TX990 Operating System Programmer's Guide	945416-9701
Model 990 Computer FORTRAN Programmer's Guide	945411-9701
Model 990 Computer COBOL Programmer's Guide	945412-9701
Model 990 Computer BASIC Programmer's Guide	945413-9701
Model 990 Computer Cross Support System User's Guide	945252-9701
Model 990 Computer TMS9900 Microprocessor Assembler and Simulator User's Guide	945420-9701
990-733 ASR System Software User's Guide	945254-9701
990 Prototyping System Operation Guide	945255-9701
Model 990/10 Program Development System Operation Guide	945256-9701
Mainframe Hardware Manuals	
Model 990/4 Computer System Hardware Reference Manual	945251-9701
Model 990/4 Computer System Field Maintenance Manual	945401-9701
Model 990/4 Computer System Depot Maintenance Manual	945403-9701

Table 7-1. 990 Computer Family Manual Set (Continued)

Manual Title	Part Number
Model 990/10 Computer System Hardware Reference Manual	945417-9701
Model 990/10 Computer System Field Maintenance Manual	945402-9701
Model 990/10 Computer System Depot Maintenance Manual	945404-9701
Model 990 Computer Family Maintenance Drawings	945421-9701
Peripheral Equipment User Manuals	
Model 990 Computer 733 ASR/KSR Terminal Installation and Operation	945259-9701
Silent 700® Electronic Data Terminals Model 733 ASR/KSR Operating Instructions (USASCII Code)	959227-9701
Model 990 Computer Model 913 CRT Display Terminal Installation and Operation	943457-9701
Model 990 Computer Line Printer Installation and Operation	945261-9701
Model 990 Computer Card Reader Installation and Operation	945262-9701
Model 990 Computer Synchronous Modem System Installation and Operation	945263-9701
Modem 990 Computer Asynchronous Modem System Installation and Operation	945400-9701
Modem 990 Computer Floppy Disc Installation and Operation Guide	945253-9701
Model 990 Computer PROM Programming Module Installation and Operation	945258-9701
Model 990 Computer Moving Head Disc System Installation and Operation	945260-9701

Table 7-1. 990 Computer Family Manual Set (Continued)

Manual Title	Part Number
Peripheral Equipment Maintenance Manuals	
Model 990 Computer Peripheral Equipment Field Maintenance Manual	945419-9701
Silent 700® Electronic Data Terminals Models 732/733 ASR/KSR Maintenance Manual	960129-9701
Model 990 Computer Model 913 CRT Display Terminal Depot Maintenance Manual	945406-9701
Technical Manual, Model 306 Printer (plus Model 306C Addendum)	974993-9701
Technical Manual, Model 500 Printer (plus Model 588 Addendum)	974998-9701
Model 990 Computer Synchronous Modem System Depot Maintenance Manual	945410-9701
Model 990 Computer Asynchronous Modem System Depot Maintenance Manual	945409-9701
Model 990 Computer Floppy Disc Depot Maintenance Manual	945418-9701
Model 990 Computer PROM Programming Module Depot Maintenance Manual	945405-9701
Model 990 Computer Models DS31, DS32 Disc troller Depot Maintenance Manual	945414-9701
Maintenance Manual for Series 30 Disc Drive	961684-9701
Product Description for Model 029 Power Supply (Disc Drive Power Supply)	961684-9702
Model 990 Computer 16 Input/Output EIA Data Module Depot Maintenance Manual	945415-9701
Model 990 Computer 16 Input/Output TTL Data Module Depot Maintenance Manual	945407-9701
Model 990 Computer TTY/EIA Interface Module Depot Maintenance Manual	945408-9701

7.7.2 COMPUTER SYSTEM FIELD MAINTENANCE MANUALS. One field maintenance manual for each member of the processor family provides the information required to identify the nature of a problem and isolate the cause of the problem to a replaceable assembly within the computer chassis, or indicate a problem with a peripheral device. Each field maintenance manual includes the following topics as applicable:

- System and subsystem block diagram
- Special test equipment list and description
- Troubleshooting procedures to a replaceable assembly
- Diagnostic tests and the procedure for running the tests on the equipment.
- Replacement instructions for assemblies.

7.7.3 COMPUTER DEPOT MAINTENANCE MANUALS. One depot maintenance manual set for each member of the processor family provides the information required to identify and isolate problems to the component level within replaceable assemblies of the processor family. Each depot maintenance manual set includes the following topics as applicable:

- Detailed circuit descriptions
- Fault isolation procedures including test points, waveforms and test set-ups as required
- Assembly repair procedures
- Wire lists, schematic diagrams, and computer printout documentation of circuit implementation
- Component identification (nomenclature and part number).

7.7.4 MAINTENANCE DRAWINGS. This manual supplies all electrical schematics and/or logic diagrams for circuit boards used within the 990 Family. Although intended for use at a depot level, these drawings may also supplement the field service engineer's information.

7.7.5 INSTALLATION AND OPERATION MANUALS. Installation and Operation manuals supply complete information to enable the user to install and operate the specified peripheral device in conjunction with the 990 Computer Family. This information includes detailed installation procedures, description of operator controls, and programming information to interface the device with the computer family. Installation and Operation manuals for each type of peripheral device in a system are included with the system shipment.

7.7.6 PERIPHERAL DEVICE FIELD MAINTENANCE MANUAL. This manual provides information to identify faulty components in any of the standard peripheral products. It includes standalone test procedures for checking each machine offline, as well as a preventive maintenance schedule for all standard peripheral devices.

7.7.7 PERIPHERAL DEVICE DEPOT MAINTENANCE MANUALS. A depot maintenance manual for each peripheral device provides instructions for maintaining the unit to the component level within each of the major assemblies. The depot maintenance manual includes the same type of information as listed for the computer depot maintenance manuals.

7.7.8 ASSEMBLY LANGUAGE PROGRAMMER'S GUIDE. This manual provides detailed descriptions of the instructions in the computer repertoire, a description of the form and use of assembly language, plus a discussion of conventions to use when programming the computer with assembly language.

7.7.9 COMPUTER PROGRAMMING CARD. This handy pocket-size card condenses all essential information required to program the computer. It lists the instruction set, describes the different instruction formats and addressing modes, and summarizes many other helpful concepts.

7.7.10 SYSTEM OPERATION GUIDES. A system operation guide is provided with each of the packaged systems for the 990 Computer Family. This manual links together the hardware components of the system with the software that accompanies it, and describes the concepts required to effectively use the system. This information includes installation instructions, procedures for verifying that the system is operating effectively, plus operating instructions for using each of the software packages in conjunction with the hardware included with the system.

7.7.11 USER GUIDES. User Guides provide information about individual software packages that are not ordinarily used in conjunction with a specific system. Each guide contains a description of the functions and capabilities of the package as well as detailed instructions for effectively using the package.

7.7.12 PROGRAMMER'S GUIDES. Programmer's Guides provide complete, detailed coverage concerning an operating system or programming language. It provides all the information an experienced programmer requires to interface with the 990 Family of Computers through the subject medium.

APPENDIX A

TMS9900/990 INSTRUCTION SET SUMMARY

APPENDIX A

TMS9900/990 INSTRUCTION SET SUMMARY

A.1 DEFINITION

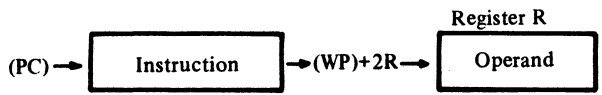
Each 990 instruction performs one of the following operations:

- Arithmetic, logical, comparison, or manipulation operations on data
- Loading or storage of internal registers (program counter, workspace pointer, or status)
- Data transfer between memory and external devices via the CRU
- Control functions.

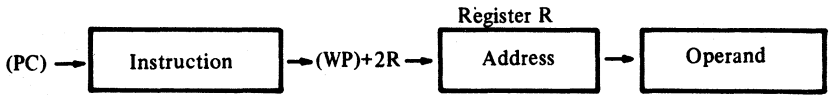
A.2 ADDRESSING MODES

990 instructions contain a variety of available modes for addressing random-memory data (e.g., program parameters and flags), or formatted memory data (character strings, data lists, etc.). The following figures graphically describe the derivation of the effective address for each addressing mode. The applicability of addressing modes to particular instructions is described in Section III along with the description of the operations performed by the instruction. The symbols following the names of the addressing modes [R, *R, *R+, @ LABEL, or @ TABLE (R)] are the general forms used by 990 assemblers to select the addressing mode.

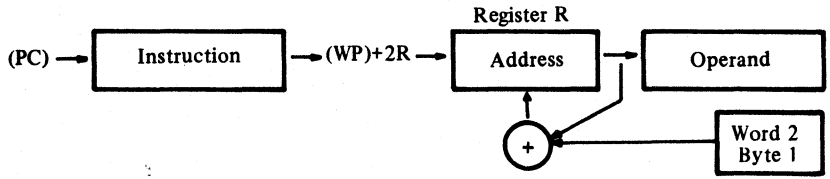
A.2.1 WORKSPACE REGISTER ADDRESSING R. Workspace Register R contains the operand.



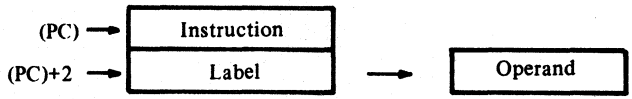
A.2.2 WORKSPACE REGISTER INDIRECT ADDRESSING *R. Workspace Register R contains the address of the operand.



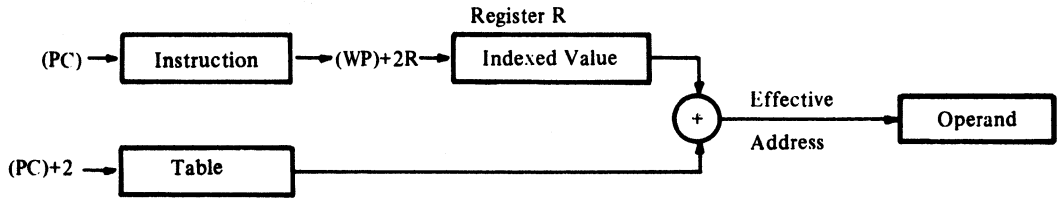
A.2.3 WORKSPACE REGISTER INDIRECT AUTO INCREMENT ADDRESSING *R+. Workspace Register R contains the address of the operand. Upon completion of the operation, the contents of workspace register R are incremented.



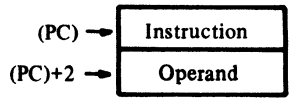
A.2.4 SYMBOLIC (DIRECT) ADDRESSING @ LABEL. The word following the instruction contains the address of the operand.



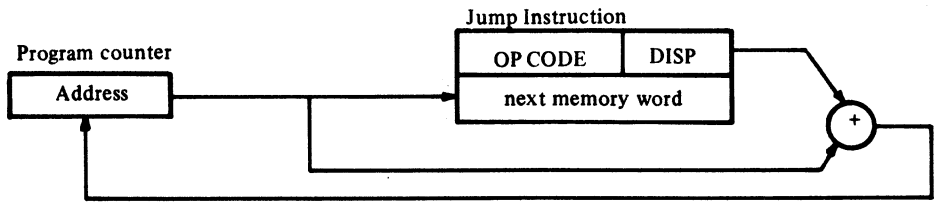
A.2.5 INDEXED ADDRESSING @ TABLE (R). The word following the instruction contains the base address. Workspace register R contains the indexed value. The sum of the base address and the indexed value results in the effective address of the operand.



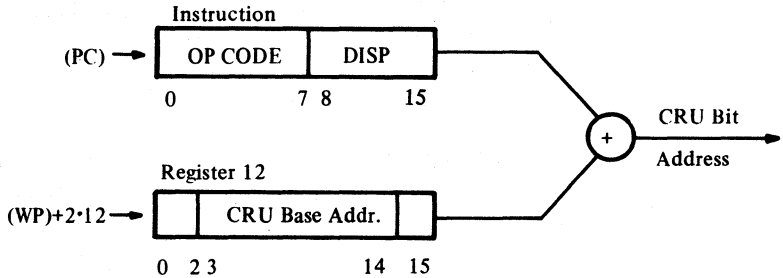
A.2.6 IMMEDIATE ADDRESSING. The word following the instruction contains the operand.



A.2.7 PROGRAM COUNTER RELATIVE ADDRESSING. The 8-bit signed displacement in the right byte (bits 8 through 15) of the instruction is added to the updated contents of the program counter. The result is placed in the PC.



A.2.8 CRU RELATIVE ADDRESSING. The 8-bit signed displacement in the right byte of the instruction is added to the CRU base address (bits 3 through 14 of the workspace register 12). The result is the CRU address of the selected CRU bit.



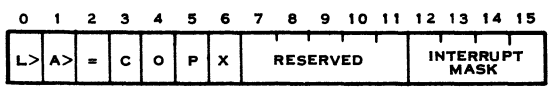
A.3 TERMS AND DEFINITIONS. The terms used in describing the instructions of the 990 Family are defined in the following table.

TERM	DEFINITION
B	Byte indicator (1=byte, 0=word)
C	Bit count
D	Destination address register
DA	Destination address
IOP	Immediate operand
LSB(n)	Least significant (right most) bit of (n)
MSB(n)	Most significant (left most) bit of (n)
N	Not Used

TERM	DEFINITION
PC	Program counter
Result	Result of operation performed by instruction
S	Source address register
SA	Source address
ST	Status register
STn	Bit n of status register
T_D	Destination address modifier
T_S	Source address modifier
W	Workspace register
WRn	Workspace register n
(n)	Contents of n
a→b	a is transferred to b
|n|	Absolute value of n
+	Arithmetic addition
–	Arithmetic subtraction
AND	Logical AND
OR	Logical OR
⊕	Logical exclusive OR
$\overline{n}$	Logical complement of n

A.4 STATUS REGISTER

The status register contains the interrupt mask level and information pertaining to the instruction operation. The following table defines those status bits used by all family members.

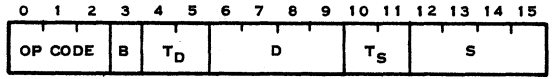


BIT	NAME	INSTRUCTION	CONDITION TO SET BIT TO 1
ST0	LOGICAL GREATER THAN	C,CB	If MSB(SA)=1 and MSB(DA)=0, or if MSB(SA)=MSB(DA) and MSB of (DA)−(SA)=1
		CI	If MSB(W)=1 and MSB of IOP=0, or if MSB(W)=MSB of IOP and MSB of IOP−(W)=1
		ABS	If (SA) ≠ 0
		All Others	If result ≠ 0
ST1	ARITHMETIC GREATER THAN	C,CB	If MSB(SA)=0 and MSB(DA)=1, or if MSB(SA)=MSB(DA) and MSB(DA)−(SA)=1
		CI	If MSB(W)=0 and MSB of IOP=1, or if MSB(W)=MSB of IOP and MSB of IOP−(W)=1
		ABS	If MSB(SA)=0 and (SA)≠0
		All Others	If MSB of result=0 and result≠0
ST2	EQUAL	C,CB	If (SA)=(DA)
		CI	If (W)=IOP
		COC	If (SA) AND $(\overline{DA})$=0
		CZC	If (SA) AND (DA)=0
		TB	If CRUIN=1
		ABS	If (SA)=0
		All Others	If result=0

BIT	NAME	INSTRUCTION	CONDITION TO SET BIT TO 1
ST3	CARRY	A,AB,ABS,AI,DEC,DECT INC,INCT,NEG,S,SB	If CARRYOUT=1
		SLA,SRA,SRC,SRL	If last bit shifted out=1
ST4	OVERFLOW	A,AB	If MSB(SA)=MSB(DA) and MSB of result≠MSB(DA)
		AI	If MSB(W)=MSB of IOP and MSB of result≠MSB(W)
		S,SB	If MSB(SA)≠MSB(DA) and MSB of result≠MSB(DA)
		DEC,DECT,INC,INCT	If MSB(SA)≠MSB of result
		SLA	If MSB changes during shift
		DIV	If MSB(SA)=0 and MSB(DA)=1, or if MSB(SA)=MSB(DA) and MSB of (DA)–(SA)=0
		ABS,NEG	If (SA)=8000_{16}
ST5	PARITY	CB,MOVB	If (SA) has odd number of 1's
		LDCR,STCR	If $1 \leqslant C \leqslant 8$ and (SA) has odd number of 1's
		AB,SB,SOCB,SZCB	If result has odd number of 1's
ST6	XOP	XOP	If XOP instruction is executed
ST12 ↓ ST15	Interrupt Mask	LIMI	If corresponding bit of IOP is 1
		RTWP	If corresponding bit of WR15 is 1

A.5 DUAL OPERAND INSTRUCTIONS WITH MULTIPLE ADDRESSING MODES FOR SOURCE AND OPERAND

General format:



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

The addressing mode for each operand is determined by the T field of that operand.

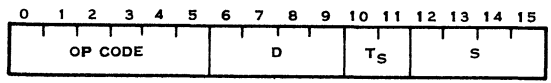
T_S OR T_D	S OR D	ADDRESSING MODE	NOTES
00	0,1, . . .15	Workspace register	1
01	0,1, . . .15	Workspace register indirect	
10	0	Symbolic	4
10	1,2, . . .15	Indexed	2,4
11	0,1, . . .15	Workspace register indirect auto-increment	3

NOTES:
1. When a workspace register is the operand of a byte instruction (bit 3=1), the left byte (bits 0 through 7) is the operand and the right byte (bits 8 through 15) is unchanged.
2. Workspace register 0 may not be used for indexing.
3. The workspace register is incremented by 1 for byte instructions (bit 3=1) and is incremented by 2 for word instructions (bit 3=0).
4. When $T_S = T_D = 10$, two words are required in addition to the instruction word. The first word is the source operand base address and the second word is the destination operand base address.

Mnemonic	Op Code B 0 1 2 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	Yes	0-2	Compare (SA) to (DA) and set appropriate status bits
CB	1 0 0 1	Compare bytes	Yes	0-2,5	Compare (SA) to (DA) and set appropriate status bits
S	0 1 1 0	Subtract	Yes	0-4	(DA)-(SA)→(DA)
SB	0 1 1 1	Subtract bytes	Yes	0-5	(DA)-(SA)→(DA)
SOC	1 1 1 0	Set ones corresponding	Yes	0-2	(DA) OR (SA)→(DA)
SOCB	1 1 1 1	Set ones corresponding bytes	Yes	0-2,5	(DA) OR (SA)→(DA)
SZC	0 1 0 0	Set zeroes corresponding	Yes	0-2	(DA) AND ($\overline{SA}$)→(DA)
SZCB	0 1 0 1	Set zeroes corresponding bytes	Yes	0-2,5	(DA) AND ($\overline{SA}$)→(DA)
MOV	1 1 0 0	Move	Yes	0-2	(SA)→(DA)
MOVB	1 1 0 1	Move bytes	Yes	0-2,5	(SA)→(DA)

A.6 DUAL OPERAND INSTRUCTIONS WITH MULTIPLE ADDRESSING MODES FOR THE SOURCE OPERAND AND WORKSPACE REGISTER ADDRESSING FOR THE DESTINATION

General format:



The addressing mode for the source operand is determined by the T_S field.

T_S	S	Addressing Mode	Notes
00	0,1, . . .15	Workspace register	
01	0,1, . . .15	Workspace register indirect	
10	0	Symbolic	
10	1,2, . . .15	Indexed	1
11	0,1, . . .15	Workspace register indirect auto increment	2

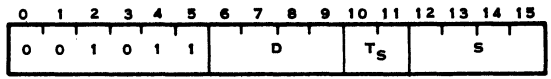
NOTES: 1. Workspace register 0 may not be used for indexing.
2. The workspace register is incremented by 2.

Mnemonic	Op Code 0	1	2	3	4	5	Meaning	Result Compared to 0	Status Bits Affected	Description
COC	0	0	1	0	0	0	Compare ones corresponding	No	2	Test (D) to determine if 1's are in each bit position where 1's are in (SA). If so, set ST2.
CZC	0	0	1	0	0	1	Compare zeroes corresponding	No	2	Test (D) to determine if 0's are in each bit position where 1's are in (SA). If so, set ST2.
XOR	0	0	1	0	1	0	Exclusive OR	Yes	0-2	(D) $\oplus$ (SA)→(D)
MPY	0	0	1	1	1	0	Multiply	No	-	Multiply unsigned (D) by unsigned (SA) and place unsigned 32-bit product in D (most significant) and D+1 (least significant). If WR15 is D, the next word in memory after WR15 will be used for the least significant half of the product.

Mnemonic	Op Code 0 1 2 3 4 5	Meaning	Result Compared to 0	Status Bits Affected	Description
DIV	0 0 1 1 1 1	Divide	No	4	If unsigned (SA) is less than unsigned (D), perform no operation and set ST4. Otherwise, divide unsigned (D) and (D+1) by unsigned (SA). Quotient → (D), remainder → (D+1). If D = 15, the next word in memory after WR15 will be used for the remainder.

A.6 EXTENDED OPERATION (XOP) INSTRUCTION

General format:



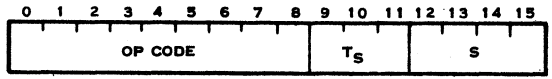
The T_S and S fields provide multiple mode addressing capability for the source operand. When the XOP is executed, ST6 is set and the following transfers occur:

$(40_{16} + 4D) \rightarrow$ (WP)
$(41_{16} + 4D) \rightarrow$ (PC)
(SA) → (new WR11)
(old WP) → (new WR13)
(old PC) → (new WR14)
(old ST) → (new WR15)

990's do not test interrupt requests upon completion of the XOP instruction.

A.7 SINGLE OPERAND INSTRUCTIONS

General format:



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

Mnemonic	Op Code 0	1	2	3	4	5	6	7	8	9	Meaning	Result Compared to 0	Status Bits Affected	Description
B	0	0	0	0	0	1	0	0	0	1	Branch	No	–	(SA) → (PC)
BL	0	0	0	0	0	1	1	0	1	0	Branch and link	No	–	(PC) → (WR11);(SA) → (PC)
BLWP	0	0	0	0	0	1	0	0	0	0	Branch and load workspace pointer	No	–	(SA) → (WP); (SA+1) → (PC); (old WP) → (new WR13); (old PC) → (new WR14); (old ST) → (new WR15); the interrupt input is not tested upon completion of the BLWP instruction.

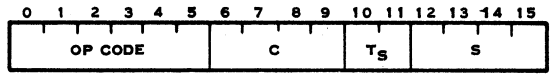
Mnemonic	Op Code 0	1	2	3	4	5	6	7	8	9	Meaning	Result Compared to 0	Status Bits Affected	Description
CLR	0	0	0	0	0	1	0	0	1	1	Clear operand	No	–	0 → (SA)
SETO	0	0	0	0	0	1	1	1	0	0	Set to ones	No	–	$FFFF_{16}$ → (SA)
INV	0	0	0	0	0	1	0	1	0	1	Invert	Yes	0-2	$(\overline{SA})$ → (SA)
NEG	0	0	0	0	0	1	0	1	0	0	Negate	Yes	0-4	–(SA) → (SA)
ABS	0	0	0	0	0	1	1	1	0	1	Absolute value	No*	0-4	|(SA)| → (SA)
SWPB	0	0	0	0	0	1	1	0	1	1	Swap bytes	No	–	(SA), bits 0 through 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.**

* Unmodified source operand is compared to zero.

**If additional memory words are required to define the operands of the instruction at SA, those words are accessed from PC and the PC will be incremented accordingly.

A.8 CRU MULTIPLE-BIT INSTRUCTIONS

General format:



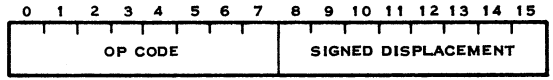
The C field specifies the number of bits to be transferred. If C = 0, 16 bits will be transferred. The CRU base register (WR12, bits 3 through 14) defines the starting CRU bit address. The bits are transferred serially and the CRU address is incremented with each bit transfer, although the contents of WR12 are not affected. T_S and S provide multiple mode addressing capability for the source operand. If 8 or fewer bits are transferred (C = 1 through 8), the source address is a byte address. If 9 or more bits are transferred (C = 0, 9 through 15), the source address is a word address. If the source is addressed in the workspace register indirect auto increment mode, the workspace register is incremented by 1 if C = 1 through 8, and is incremented by 2 otherwise.

Mnemonic	Op Code 0	1	2	3	4	5	Meaning	Result Compared to 0	Status Bits Affected	Description
LDCR	0	0	1	1	0	0	Load communication 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 communication 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 $1 \leq C \leq 8$.

A.9 CRU SINGLE-BIT INSTRUCTIONS

General format:

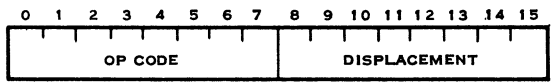


CRU relative addressing is used to address the selected CRU bit.

Mnemonic	Op Code 0	1	2	3	4	5	6	7	Meaning	Status Bits Affected	Description
SBO	0	0	0	1	1	1	0	1	Set bit to one	–	Set the selected CRU output bit to 1.
SBZ	0	0	0	1	1	1	1	0	Set bit to zero	–	Set the selected CRU output bit to 0.
TB	0	0	0	1	1	1	1	1	Test bit	2	If the selected CRU input bit = 1, set ST2.

A.10 JUMP INSTRUCTIONS

General format:

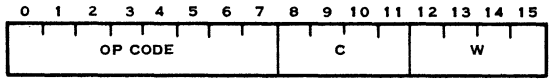


Jump instructions cause the PC to be loaded with the value selected by PC relative addressing if the bits of ST are at specified values. Otherwise, no operation occurs and the next instruction is executed since PC points to the next instruction. The displacement field is a word count to be added to PC. Thus, the jump instruction has a range of −128 to 127 words from memory-word address following the jump instruction. No ST bits are affected by jump instruction.

Mnemonic	Op Code 0	1	2	3	4	5	6	7	Meaning	ST Condition to Load PC
JEQ	0	0	0	1	0	0	1	1	Jump equal	ST2 = 1
JGT	0	0	0	1	0	1	0	1	Jump greater than	ST1 = 1
JH	0	0	0	1	1	0	1	1	Jump high	ST0 = 1 and ST2 = 0
JHE	0	0	0	1	0	1	0	0	Jump high or equal	ST0 = 1 or ST2 = 1
JL	0	0	0	1	1	0	1	0	Jump low	ST0 = 0 and ST2 = 0
JLE	0	0	0	1	0	0	1	0	Jump low or equal	ST0 = 0 or ST2 = 1
JLT	0	0	0	1	0	0	0	1	Jump less than	ST1 = 0 and ST2 = 0
JMP	0	0	0	1	0	0	0	0	Jump unconditional	unconditional
JNC	0	0	0	1	0	1	1	1	Jump no carry	ST3 = 0
JNE	0	0	0	1	0	1	1	0	Jump not equal	ST2 = 0
JNO	0	0	0	1	1	0	0	1	Jump no overflow	ST4 = 0
JOC	0	0	0	1	1	0	0	0	Jump on carry	ST3 = 1
JOP	0	0	0	1	1	1	0	0	Jump odd parity	ST5 = 1

A.11 SHIFT INSTRUCTIONS

General format:

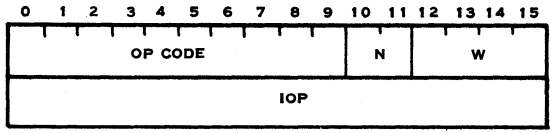


If C = 0, bits 12 through 15 of WR0 contain the shift count. If C = 0 and bits 12 through 15 of WR0 = 0, the shift count is 16.

Mnemonic	Op Code 0	1	2	3	4	5	6	7	Meaning	Result Compared to 0	Status Bits Affected	Description
SLA	0	0	0	0	1	0	1	0	Shift left arithmetic	Yes	0-4	Shift (W) left. Fill vacated bit positions with 0.
SRA	0	0	0	0	1	0	0	0	Shift right arithmetic	Yes	0-3	Shift (W) right. Fill vacated bit positions with original MSB of (W).
SRC	0	0	0	0	1	0	1	1	Shift right circular	Yes	0-3	Shift (W) right. Shift previous LSB into MSB.
SRL	0	0	0	0	1	0	0	1	Shift right logical	Yes	0-3	Shift (W) right. Fill vacated bit positions with 0.

A.12 IMMEDIATE REGISTER INSTRUCTIONS

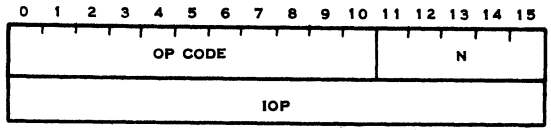
General format:



Mnemonic	Op Code 0	1	2	3	4	5	6	7	8	9	10	Meaning	Result Compared to 0	Status Bits Affected	Description
AI	0	0	0	0	0	0	1	0	0	0	1	Add immediate	Yes	0-4	(W) + IOP → (W)
ANDI	0	0	0	0	0	0	1	0	0	1	0	AND immediate	Yes	0-2	(W) AND IOP → (W)
CI	0	0	0	0	0	0	1	0	1	0	0	Compare immediate	Yes	0-2	Compare (W) to IOP and set appropriate status bits
LI	0	0	0	0	0	0	1	0	0	0	0	Load immediate	Yes	0-2	IOP → (W)
ORI	0	0	0	0	0	0	1	0	0	1	1	OR immediate	Yes	0-2	(W) OR IOP → (W)

A.13 INTERNAL REGISTER LOAD IMMEDIATE INSTRUCTIONS

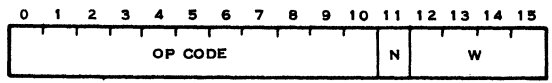
General format:



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

A.14 INTERNAL REGISTER STORE INSTRUCTIONS

General format:

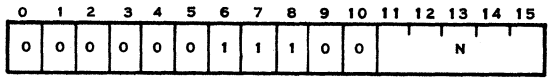


No ST bits are affected.

Mnemonic	Op Code 0	1	2	3	4	5	6	7	8	9	10	Meaning	Description
STST	0	0	0	0	0	0	1	0	1	1	0	Store status register	(ST) → (W)
STWP	0	0	0	0	0	0	1	0	1	0	1	Store workspace pointer	(WP) → (W)

A.15 RETURN WORKSPACE POINTER (RTWP) INSTRUCTION

General format:



The RTWP instruction causes the following transfers to occur:

(WR15) → (ST)
(WR14) → (PC)
(WR13) → (WP)

APPENDIX B

990/4 MICROCOMPUTER

APPENDIX B

990/4 MICROCOMPUTER

B.1 GENERAL

The 990/4 Microcomputer is a circuit board that is 10.8 inches by 14.25 inches. Combined on this circuit board are the following functional components:

- Arithmetic unit, implemented with the TMS9900 microprocessor chip
- Up to 5K words of memory
- Direct command driven serial input/output bus (Communication Register Unit)
- Synchronous, parallel 9900 bus for direct memory access and memory expansion
- Eight vectored interrupts
- Real time clock interrupt logic
- Power fail and restart interrupt logic
- Programmers panel or maintenance panel interface.

The 990/4 microcomputer is intended for applications where the I/O activity is handled mainly via the Communication Register Unit (CRU); the 9900 bus is used for memory expansion or customer supplied DMA interfaces. Texas Instruments supplied DMA interfaces are implemented on the TILINE and are not compatible with the 990/4 microcomputer.

B.2 BOARD LAYOUT

Figure B-1 shows the functional layout of a fully implemented 990/4 microcomputer. Figure B-2 illustrates the outline of the circuit board.

B.3 MEMORY

Two memory sections are implemented on the 990/4; a 4K X 17 bit dynamic RAM and 1K X 16 PROM/Static RAM section.

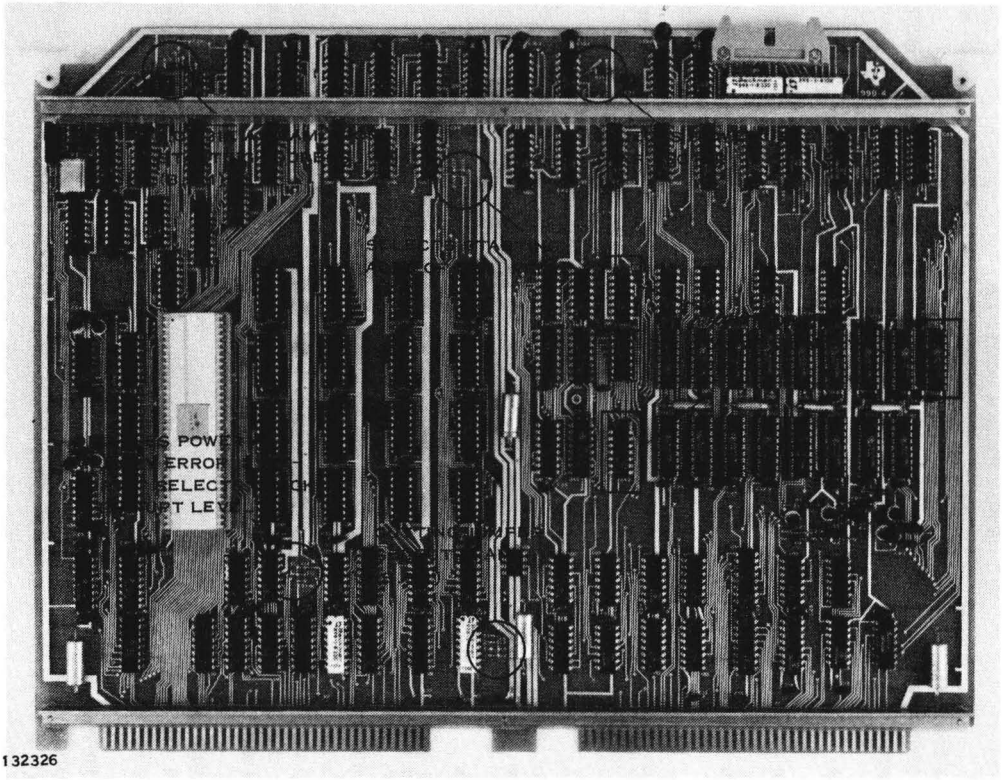


Figure B-1. 990/4 Board Layout

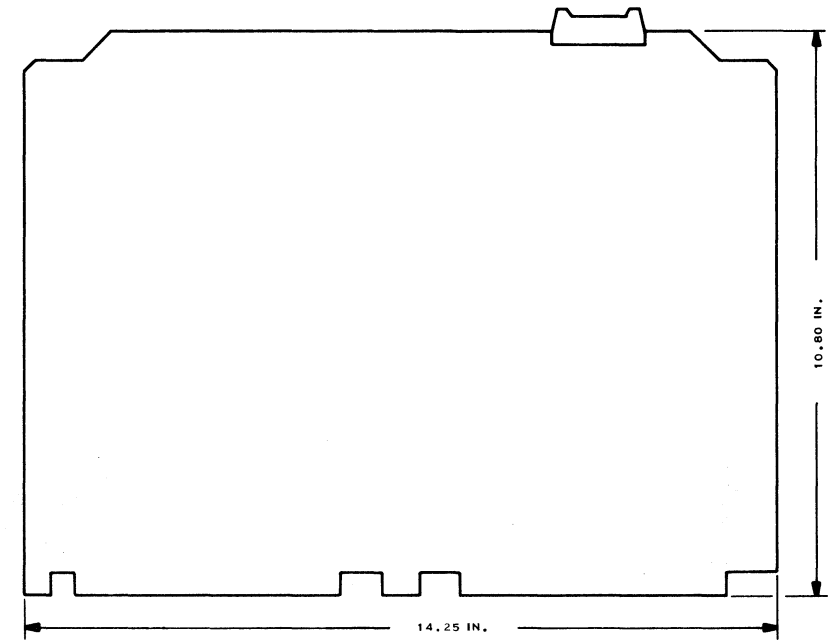


Figure B-2. 990/4 Board Outline

B.3.1 DYNAMIC RAM. The dynamic RAM is implemented using TMS4051-1 dynamic MOS RAMs. This memory may be addressed starting at address 0 or at address 4K (word address). Zero or 4K is selected when the jumper is installed as follows:

E17 → E16	0000_{16} Dynamic RAM Start Address
E17 → E18	2000_{16} Dynamic RAM Start Address

Odd parity is implemented as an option by installing a 17th TMS4051 plus two 74180 circuits.

All memory cycles to the dynamic RAM require 667 nanoseconds unless memory refresh occurs. Memory refresh contention may cause one memory cycle every 30 microseconds to be 1000 or 1333 nanoseconds.

The memory controller is powered by a separate +5 V MEM bus. If the standby power supply option is elected, +5 V MEM, as well as +12 V MEM, and -5 V MEM, is maintained from a battery when ac power is off. The 4K RAM is maintained for 8 hours on the battery normally supplied with the standby power supply option.

B.3.2 PROM/RAM. A 1024 word memory is implemented using 16-256 x 4 devices. These may be SN74S287 TTL PROMs or TMS4043 Static MOS RAMs. The memory has four 256 word banks, each bank consisting of four devices. Each bank must be all PROM or all RAM. There is a jumper for each 256 words to differentiate between PROM and RAM. Cutting a jumper indicates RAM.

E7 → E8	Address 0 - 255 is ROM
E5 → E6	Address 256 - 511 is ROM
E3 → E4	Address 512 - 767 is ROM
E2 → E1	Address 768 - 1023 is ROM

This memory may be addressed starting at address 0 or at address 31K (word address). Two jumper wires select 0 or 31K.

E19 → E20 E24 → E25	0000_{16} Start Address
E24 → E26	$F800_{16}$ Start Address

Cycle time for these memory devices is:

PROM Read = 667 nanoseconds
RAM Read or Write = 1333 nanoseconds

No standby power is provided for RAM in this memory.

B.4 INTERRUPTS

Inputs for seven external interrupts are provided (Levels 1 through 7) (Level 0 is the power-up trap). Interrupt inputs are synchronized with the CPU clock on the 990/4 board, and are encoded and presented to the TMS9900 microprocessor along with an interrupt request. Three interrupt conditions are generated on-board and may be wired to these external interrupt inputs: power failure, memory error and real time clock.

B.4.1 POWER FAILURE. Logic to generate an interrupt at Level 1 in event of imminent power failure is implemented on the 990/4 board. This interrupt may be disconnected by cutting a jumper wire. This interrupt is cleared with RSET.

E11 → E12	Power fail is connected to Level 1 interrupt

B.4.2 MEMORY ERROR. Logic to generate an interrupt at Level 2 in event of a parity error in the local 4K RAM or in externally implemented memory is implemented as an option on the 990/4 board. Memory error interrupts are cleared by a CRU output to bit 12 of the console interface base address or by RSET. The interrupt may be disconnected by cutting a jumper wire.

E13 → E14 MEMERR is connected to Level 2 interrupt

B.4.3 REAL TIME CLOCK. The 120 Hz Real Time Clock Interrupt logic is implemented on the 990/4 board. The interrupt is enabled via the CKON instruction and disabled with either CKOF or RSET. The interrupt is cleared with the sequence CKOF, CKON. The interrupt may be connected with jumper wires either to Level 5 or to Level 7 or disconnected. The normal configuration is with the Level 5 jumper wire installed.

E10 → E9	Clock connected to Level 5 interrupt
E10 → E23	Clock connected to Level 7 interrupt

B.5 CRU DECODING

24 CRU module select lines are decoded from CRU Base address 0000 through $02E0_{16}$. They are routed to the 990/4 board bottom edge connectors for connection to the CRU I/O slots in the backplane. CRU address bits 12-15 are buffered for use by all CRU Modules. Address bits 4-11 are buffered for use only by CRU expander boards.

B.6 PROGRAMMER PANEL CONTROL CRU BIT ASSIGNMENTS

CRU base $1FE0_{16}$ is decoded for use by the programmer panel. These bit assignments are shown in table B-1. Panel functions implemented directly on the 990/4 board are:

- Power Light - Power Reset is inverted and supplied to the panel connector through a 180 ohm resistor.
- Idle Light - The Idle signal is buffered and supplied to the panel connector for the Idle LED through a 390 ohm resistor.
- Set Run Control (Panel Output Bit 10) - An SBO or SBZ instruction will set the Run Indicator flip-flop on the 990/4 board. The Run signal is buffered and supplied to the panel connector. Run is automatically cleared by Restart.

Table B-1. Programmer Panel CRU Bit Assignments

	Output Bits		Input Bits
0	Light 7	0	Switch Column 7
.	.	.	.
.	.	.	.
.	.	.	.
.	.	.	.
7	Light 0	7	Switch Column 0
8	Increment Scan	8	Scan Count Bit 1
9	Clear Scan	9	Scan Count Bit 0
10	Run Light	10	Timer Active
11	Fault Light	11	Front Panel Not Present
12	Memory Error Interrupt Clear	12	Spare
13	Start Panel Timer	13	.
14	Set SIE Function	14	.
15	Flag	15	Flag

- Fault Light (Panel Output Bit 11) - An SBO instruction to Bit 11 of CRU Base $IFE0_{16}$ will cause the Fault Light on both the 990/4 board and the panel to light. The Fault Indicator flip-flop is implemented on the 990/4 board and is buffered to drive both Fault Lights. The Fault signal for the Panel Fault Light is supplied to the panel connector through a 180 ohm resistor. I/O reset or an SBZ instruction to panel output Bit 11 will turn off the Fault Lights. Power reset will turn on the Fault Lights.

- Memory Error Interrupt Clear (Panel Output Bit 12) - An SBO or SBZ instruction to bit 12 of CRU Base $1FE0_{16}$ or power reset will cause the memory error interrupt to be cleared.

- SIE (Single Instruction Execute) Function (Panel Output Bit 14) - An SBO or SBZ instruction to bit 14 of CRU Base $1FE0_{16}$ will cause the Restart Trap to occur after two additional instructions are executed.

B.7 RESTART

Restart is an unmaskable interrupt input which causes a trap through a trap vector at $FFFC_{16}$. RESTART- comes into the 990/4 board from the backplane and from the console connector. Restart is caused also by the LREX instruction. RESTART- is debounced on the 990/4 board. One edge is used to set the Load input to the TMS9900 in such a way as to cause a trap to the console service or loader software. Restart is inhibited until RUN (Console CRU output 10) is set again. RESTART- is terminated on the 990/4 board with a 4.7K ohm pull-up resistor.

B.8 POWER-UP INITIALIZATION

A jumper wire is implemented on the 990/4 CPU board to control whether the power-up interrupt traps through address 0000 or through address FFFC. This will allow more flexible memory allocation for dedicated systems not having the operator console. Cutting the jumper causes the trap through 0000.

E21 → E22	Power up traps to FFFC
E22 → E27	Power up traps to 0000

B.9 SPECIAL CAUTIONS FOR 990/4

Observe the following unique characteristics when planning for the 990/4 microcomputer:

- TILINE Peripheral Control Space - The 990/4 does not implement the TILINE.
- Interrupts - Only Interrupt levels 0-7 are implemented on the 990/4. Level 2 has only one interrupt (Memory Error); consequently no error-interrupt status register is implemented. The real-time clock interrupt may be optionally wired to Level 5 or 7 (not 5 or 15).
- Memory Mapping - No memory mapping is implemented on the 990/4.
- XOP Hardware Interface - No provision is made to execute XOP instructions in hardware. XOP instructions must be interpreted in software.
- Instruction Set - LDS, LDD, and LMF instructions are not implemented on the 990/4. No Illegal Operation Codes are detected. Unimplemented or Illegal operation codes are treated as NO-OPs.

B.10 INTERFACE DESCRIPTION

The system interface is made through the bottom edge of the 990/4 board. The signals are interfaced in two groups of 80 gold contacts for mating with two backplane connectors, P1 and P2. Table B-2 lists the signals that are presented to the system interface connectors to interface add-on memory and peripheral devices.

Table B-2. 990/4 System Interface

Note: Refer to CRU and data bus specifications for explanation of functions

Pin	Function	Pin	Function
P1-1,2	GND	P1-38	CRUBIT14
3,4	+5 MAIN	39,40	+12 MAIN
5,6	+12 MEMORY	41,42	-12 MAIN
7,8	+5 MEMORY	43	MODSEL3-
9,10	-5 MEMORY	44	MODSEL4-
11	DBIN	45	MODSEL5-
12	GND	46	MODSEL6-
13	TLPRES-	47	MODSEL7-
14	TLIORES-	48	MODSEL8-
15	GND	49	MODSEL9-
16	TLPFWP	50	CRUBIT7
17	GND	51	MODSEL10-
18	CRUBITOUT	52	CRUBIT6
19	GND	53	MODSEL11-
20	READY	54	CRUBIT5
21	GND	55	TLMER-
22	STORECLK-	56	CRUBIT4
23	MODSEL0-	57	GND
24	GND	58	WE-
25	MEMEN-	59	GND
26	GND	60	CRUBITIN
27	DAT12-	61	MODSEL12-
28	DAT13-	62	CRUBIT8
29	120HZ	63	WAIT-
30	DAT14-	64	CRUBIT9
31	DAT15-	65	INT1-
32	CRUBIT13	66	INT2-
33	IAQ-	67	MODSEL13-
34	CRUBIT15	68	CRUBIT10
35	MODSEL1-	69	MODSEL14-
36	CRUBIT12	70	CRUBIT11
37	MODSEL2-	71	OPEN

Table B-2. 990/4 System Interface (Continued)

Pin	Function	Pin	Function
		P2-26	HOLD-
P1-72	GND	27	ADR12-
73	CLK1-	28	RESTART-
74	GND	29	ADR11-
75	CLK3-	30	SPARE
76	MODSEL15-	31	ADR14-
77,78	+5 MAIN	32	MODSEL19-
79,80	GND	33	DAT09-
		34	MODSEL18-
		35	DAT02-
P2-1,2	GND	36	MODSEL17-
3,4	+5 MAIN	37	DAT03-
5	OPEN	38	MODSEL16-
6	HOLDA-(OUT)	39,40	+12 MAIN
7	GND	41,42	-12 MAIN
8	ADR09-	43	DAT06-
9	ADR10-	44	Open
10	ADR05-	45	DAT07-
11	ADR07-	46	INT4-
12	ADR06-	47	ADR01-
13	MODSEL23-	48	INT5-
14	Open	49	ADR02-
15	ADR08-	50	INT6-
16	MODSEL22-	51	Open
17	ADR03-	52	INT7-
18	MODSEL21-	53	Open
19	ADR04-	54	Open
20	DAT11-	55	Open
21	DAT08-	56	Open
22	MODSEL20-	57	Open
23	DAT10-	58	Open
24	INT3-	59	ADR00-
25	ADR13-	60	Open

Table B-2. 990/4 System Interface (Continued)

Pin	Function	Pin	Function
		P2-68	Open
P2-61	DAT04-	69	DAT01-
62	Open	70	Spare
63	DAT05-	71,72	-5 MEMORY
64	Open	73,74	+5 MEMORY
65	Open	75,76	+12 MEMORY
66	Open	77,78	+5 MAIN
67	DAT00-	79,80	GND

B.11 ENVIRONMENTAL REQUIREMENTS

- Operating:
  Temperature 0 to 65°C
  Humidity 0 to 95% non-condensing

- Storage:
  Temperature -40 to 70°C
  Humidity 0 to 95% non-condensing

B.12 990/4 INSTRUCTION EXECUTION TIMES

Instruction execution times for the 990/4 are a function of:

1) Clock cycle time [t_c = .333 μs]

2) Addressing mode used where operands have multiple addressing mode

3) Memory speed. (Add 2 clock cycles for every access to static RAM.)

Table B-3 lists the number of clock cycles required to execute each 990/4 instruction. For instructions with multiple addressing modes for either or both operands, the table lists the number of clock cycles with all operands addressed in the workspace register mode. To determine the additional number of clock cycles required for modified addressing, add the appropriate values from the referenced tables (A or B). The total instruction execution time for an instruction is:

$$T = (t_c)\ (C) + 2(t_c)\ (S)$$

where:

T = total instruction execution time;
t_c = clock cycle time;
C = number of clock cycles
S = number of memory cycles accessing static RAM

Table B-3. Instruction Execution Times

Instruction	Clock Cycles C	Address Modification† Source	Dest
A	14	A	A
AB	14	B	B
ABS (MSB = 0)	12	A	–
(MSB = 1)	14	A	–
AI	14	–	–
ANDI	14	–	–
B	8	A	–
BL	12	A	–
BLWP	26	A	–
C	14	A	A
CB	14	B	B
CI	14	–	–
CKOF	12	–	–
CKON	12	–	–
CLR	10	A	–
COC	14	A	–
CZC	14	A	–
DEC	10	A	–
DECT	10	A	–
DIV (ST4 is set)	16	A	–
DIV (ST4 is reset)	92-124	A	–
IDLE	12	–	–
INC	10	A	–
INCT	10	A	–
INV	10	A	–
Jump (PC is changed)	10	–	–
(PC is not changed)	8	–	–
LDCR (C = 0)	52	A	–
(1 ≤C ≤ 8)	20+2C	B	–
(9 ≤ C ≤ 15)	20+2C	A	–
LI	12	–	–

Table B-3. Instruction Execution Times (Continued)

Instruction	Clock Cycles C	Address Modification† Source	Dest
LIMI	16	–	–
LREX	12	–	–
RESET function	28	–	–
LOAD function	24	–	–
Interrupt context switch	24	–	–
LWPI	10	–	–
MOV	14	A	A
MOVB	14	B	B
MPY	52	A	–
NEG	12	A	–
ORI	14	–	–
RSET	12	–	–
RTWP	14	–	–
S	14	A	A
SB	14	B	B
SBO	12	–	–
SBZ	12	–	–
SETO	10	A	–
Shift (C + 0)	12+2C	–	–
(C=0,Bits 12-15 of WRO=0)	52	–	–
(C=0, Bits 12-15 of WRP-N≠0)	20+2N	–	–
SOC	14	A	A
SOCB	14	B	B
STCR (C=0)	60	A	–
(1 ⩽ C ⩽ 7)	42	B	–
(C=8)	44	B	–
(9 ⩽ C ⩽ 15)	58	A	–
STST	8	–	–
STWP	8	–	–
SWPB	10	A	–
SZC	14	A	A

Table B-3. Instruction Execution Times (Continued)

Instruction	Clock Cycles C	Address Modification† Source	Dest
SZCB	14	B	B
TB	12	–	–
X**	8	A	–
XOP	44	A	–
XOR	14	A	–
Undefined op codes 0000 01FF, 0320-033F, 0C00-0FFF	6		–
0780-07FF	8		–

*Execution time is dependent upon the partial quotient after each clock cycle during execution.

**Execution time is added to the execution time of the instruction located at the source address.

†The letters A and B refer to the respective tables that follow.

Table A Addressing Mode	Clock Cycles C	Table B Addressing Mode	Clock Cycles C
WR (T_S or T_D = 00)	0	WR (T_S or T_D = 001)	0
WR indirect (T_S or T_D = 01)	4	WR indirect (T_S or T_D = 01)	4
WR indirect auto	8	WR indirect auto	6
Increment (T_S or T_D = 11)		Increment (T_S or T_D = 11)	
Symbolic (T_S or T_D = 10, S or D = 0)	8	Symbolic (T_S or T_D = 10, S or D = 0)	8
Indexed (T_S or T_D = 10, S or D = 0)	8	Indexed (T_S or T_D = 10, S or D = 0)	8

As an example, the instruction MOVB is used in a system with $t_c(0) = 0.333$ μsec. Both operands are addressed in the workspace register mode:

$T = t_c(0)\ (C) = 0.333\ (14)\ \mu s = 4.662\ \mu s.$

If the source operand was addressed in the symbolic mode:

$T = t_c(0)\ (C)$

$C = 14 + 8 = 22$

$T = 0.333\ (22)\ \mu s = 7.326\ \mu s.$

APPENDIX C

990/10 MINICOMPUTER

APPENDIX C

990/10 MINICOMPUTER

C.1 HARDWARE DESCRIPTION

A complete 990/10 Minicomputer consists of 3 printed circuit boards: an Arithmetic Unit board (AU1), a system interface/memory mapping board (AU2), and either a non-error correcting (parity option) or error correcting (and detecting) board.

C.1.1 ARITHMETIC UNIT. The Arithmetic Unit (AU1) is a single 2-sided printed circuit board which contains all electronics to process instructions from the memory and to sequence CRU input/output operations.

The AU has several special registers which have defined uses. Additional locations called workspace registers exist in memory as defined by one of the special registers. Table C-1 lists the special registers.

The PC and WP always contain an even address (Bit 15 = 0). The workspace pointer contains the beginning address of the sixteen consecutive memory locations that are actively being used as workspace registers.

The status register contains information about the state of the CPU as defined in Appendix A. In addition, Bit 7 designates the privileged mode and Bit 8 selects one of the mapping registers for the memory map option.

C.1.2 TILINE PERIPHERAL CONTROL SPACE (TPCS). CPU addresses in the range F800-FBFF are mapped to the TILINE Peripheral Control Space (FFC00-FFDFF). Five address bits are appended to the left (most significant side) of the address in order to form a 20-bit TILINE word address. The last 512 words (FC00-FFFE) are preempted to address the TTL PROM.

Table C-1. 990/10 Special Registers

Name	Size	Abbreviation
Program Counter	16 bits	PC
Workspace Pointer	16 bits	WP
Status Register	16 bits	ST

C.1.3 VECTORED INTERRUPTS. Sixteen priority vectored interrupt levels are implemented in the Model 990/10 Computer. The interrupts generated by extended operation codes (XOPs) are not a part of the priority structure. When an interrupt is recognized by the hardware (interrupt pending at level not masked by status register mask), a Branch and Load Workspace Pointer (BLWP) instruction is forced using the words stored at the trap address for the Workspace Pointer and Program Counter.

A Return (RTWP) instruction is used to exit from an interrupt subroutine. The prior operating environment is completely restored by the RTWP.

C.1.4 INTERNAL INTERRUPTS. The highest priority level is reserved for the Power Restored interrupt. Seven internal CPU functions are wired to other interrupts. These are: Power Failure Imminent, Memory Data Error, Memory Mapping Error, 120 Hz Clock, Illegal Operation, Privileged Instruction Violation, and Bus Timeout.

C.1.4.1 Power Restored. Any time AC power is restored to the Central Processing Unit chassis, program execution will begin using the Program Counter and Workspace Pointer which were previously stored at the level zero trap locations. The Mask Field of the Status Register will be set to zero. This disables all interrupts except level zero itself. Level zero cannot be disabled.

C.1.4.2 Power Failure Imminent. When the CPU power supply senses a loss of AC power, an interrupt will be generated and 7.0 ms of program time can elapse before the system will be reset for the power loss state. This function is wired to level one.

C.1.4.3 Error Interrupt Register. A register is implemented at CRU Base $IFC0_{16}$ which is used as a status indicator for interrupt level 2. It is a 16-bit register of which 5 bits are implemented. These bits are set automatically by hardware upon encountering the error condition and they are logically OR'ed together to cause a TRAP at interrupt level 2. These bits are cleared either by the level 2 interrupt service routine, or automatically by the power-up trap, level 0. The implemented bits are as follows:

Bit 11*	MAPERR - Memory Mapping Error
Bit 12	MER - Uncorrectable Memory Data Error
Bit 13	ILLOP - Illegal Instruction Code
Bit 14	PRIVOP - Privileged Instruction Violation
Bit 15	TIMEOUT - TILINE Timeout-Addressing unimplemented memory.

*Implemented only in 990/10 with mapping feature.

C.1.5 EXTERNAL INTERRUPTS. Levels three through fifteen are reserved for externally requested equipment interrupts.

C.1.5.1 Real Time Clock. A line frequency synchronized oscillator on the power supply is input to the 990/10 CPU. On every cycle of the oscillator the real time clock interrupt function is generated. This function may be connected with jumper wires to level 5 or level 15 or may be disconnected.

C.1.5.2 Restart. Restart is an unmaskable interrupt that traps through a trap vector at $FFFC_{16}$. RESTART comes into the 990/10 from the backplane and from the console connector. Restart can be initiated by executing an LREX instruction and by the console SIE function. RESTART is debounced on the 990/10 (AU2). One edge is used to cause a trap to the console service or loader firmware. Restart is inhibited until console CPU RUN (Programmer Panel CRU bit 10) is set again.

C.1.6 FEATURES UNIQUE TO 990/10. The following features are not found in the 990/4 microcomputer.

C.1.6.1 Privileged Instruction Feature. Bit 7 is set by software to indicate privileged instruction mode. When set to a "1", instructions designated as privileged are processed as illegal instructions, the level 2 interrupt trap is taken, and bit 14 of the Error Interrupt Register is set to a "1". Status Bit 7 is automatically cleared by any Trap. The privileged instructions are listed below:

LIMI
LMF
LDS
LDD
LREX
RSET
CKON
CKOF
IDLE
SBO
SBZ
LDCR
STCR
RTWP*

} For CRU bit address greater than or equal to $E00_{16}$ (CRU base $1C00_{16}$).

*RTWP does not trap, but is executed differently in the Privileged Mode. RTWP operates in the normal way except that during the transfer, (W15) → Status Register, the modification of bits 7, 8, and 12-15 in the Status Register is inhibited.

C.1.6.2 Illegal Instruction Codes. The following instruction codes cause Bit 13 of the Error Interrupt Register to be set :

INSTRUCTIONS	
0000 – 01FF	*0320 – 033F
0210 – 021F	0341 – 035F
0230 – 023F	0361 – 037F
0250 – 025F	0381 – 039F
0290 – 029F	03A1 – 03BF
02B0 – 02BF	03C1 – 03DF
02D0 – 02DF	03E1 – 03FF
02E1 – 02FF	*0780 – 07FF
0301 – 031F	0C00 – 0FFF

*NOT ILLEGAL WITH MAP OPTION.
MODEL 990/10 ONLY

C.1.6.3 Memory Mapping Feature. This is an optional feature for the 990/10. This feature expands 990/10 memory addressability to 1024K words.

Status Register Bit 8 is set or cleared by software to indicate whether mapping is to be performed using Map Register File 0 or Map Register File 1. (ST8 = 0 indicates File 0) ST8 is automatically cleared by the power-up trap. If memory is addressed outside of the limits imposed by the mapping limit registers of the active Map File, the memory cycle is aborted and bit 11 of the Error Interrupt Register is set. A trap through the error interrupt (level 2) will then occur. This error occurs if all three limit registers in the active Map File are set to a value less than the address generated by a program.

For diagnostic purposes, a control register is implemented at CRU Base $1FA0_{16}$. Using this register, diagnostic software can disable mapping and read the registers of the active Map File or a saved mapped address.

C.1.6.4 XOP Interface. The 990/10 includes a feature which allows selected XOP instructions to be executed in a user supplied hardware module instead of software trap routines. The following signals are presented at the top edge of AU2 or AU2B for connection of a ribbon cable to a hardware XOP module:

XOPIAQCK–: A low active signal from the instruction processor that indicates to all hardware XOP functions installed in the chassis, that there is an instruction on the TILINE data lines (TLDAT00-15-).

XOPTHERE–: A low active signal from the XOP hardware that indicates that XOP hardware has decoded the TILINE data concurrent with XOPIAQCK. This data represents an XOP function designated to that XOP module. If the instruction processor does not see XOPTHERE, then it executes the XOP using software subroutines.

XOPSTB–: A low active signal from the instruction processor that indicates that the effective address of the XOP instruction is on the TILINE address lines (TLADR00-19-). XOPSTB- remains active and the address lines remain valid until the XOP module generates TLTM- indicating that its bus stored the effective address and is ready to proceed. Failure to generate a TLTM- will cause a timeout interrupt. If the XOP hardware wants to fetch the operand at the effective address it may assert TLGO-. A memory module at the effective address will then generate TLTM- and the XOP hardware will read the operand from the TILINE data lines.

ENXOPQ–: A low active signal from the instruction processor that indicates that the instruction processor is ready to turn over the control to the XOP hardware. The signal (ENXOPQ-) remains low (active) until the XOP module completes or terminates the operations.

XOPCOMP– and XOPABORT–: These low active signals from the XOP module tell the instruction processor that status information is on the TILINE data lines (TLDAT00-15-). XOPCOMP- indicates a completed operation; XOPABORT– indicates that the XOP module has sensed an interrupt and is designed such that it will not proceed any further in the operation.

PINTQ–: A low active signal from the instruction processor that indicates the presence of an interrupt.

C.2 990/10 CPU CHASSIS INSTALLATION

A 990/10 CPU consists of two full-sized circuit boards. One board contains the TILINE bus interface, interrupt logic, CRU interface, and (optionally) the memory mapping feature. This board plugs into slot 1 in either the 6-slot or 13-slot 990 chassis. This board is called AU2 or (with mapping) AU2B. AU1 is the companion board for AU2 or AU2B. It plugs into slot 2 in the chassis and contains the instruction processor portion of the CPU. It interfaces with AU2 via ribbon cable jumpers across the top edges of both boards. A smaller ribbon cable to AU2 interfaces an operator panel or programmer panel as illustrated in figure C-1.

C.3 INTERFACE DESCRIPTION

The system interface is made through the bottom edge of the 990/10 AU2(B) board. The signals are interfaced through two groups of 80 gold contacts for mating with two backplane connectors, P1 and P2. Table C-2 lists the signals that are presented to the system interface for interrupts, TILINE and CRU.

The 990/10 also provides an interface to the same Programmer Panel used on the 990/4. Refer to paragraph B.6 for a description of the panel interface.

Table C-2. 990/10 System Interface (AU 2) Pin Out

Pin	Function	Pin	Function
P1-			
1,2	GND	P1-55	TLMER-
3,4	+5 MAIN	56	CRUBIT4
5,6	+12 MEMORY	57	GND
7,8	+5 MEMORY	58	TLAV
9,10	-5 MEMORY	59	GND
11	TLREAD	60	CRUBITIN
12	GND	61	MODSEL12-
13	TLPRES-	62	CRUBIT8
14	TLIORES-	63	TLWAIT-
15	GND	64	CRUBIT9
16	TLPFWP	65	OPEN
17	GND	66	OPEN
18	CRUBITOUT	67	MODSEL13-
19	GND	68	CRUBIT10
20	TLTM-	69	MODSEL14-
21	GND	70	CRUBIT11
22	STORECLK-	71	TLAK-
23	OPEN	72	GND
24	GND	73	OPEN
25	TLGO-	74	GND
26	GND	75	OPEN
27	TLDAT12-	76	MODSEL15-
28	TLDAT13-	77,78	+5 MAIN
29	120HZ	79,80	GND
30	TLDAT14-		
31	TLDAT15-	P2-	
32	CRUBIT13	1,2	GND
33	IAQ-	3,4	+5 MAIN
34	CRUBIT15	5	TLAG(OUT)
35	OPEN	6	TLAG(IN)
36	CRUBIT12	7	GND
37	MODSEL2-	8	TLADR14-
38	CRUBIT14	9	TLADR15-
39,40	+12 MAIN	10	TLADR10-
41,42	-12 MAIN	11	TLADR12-
43	MODSEL3-	12	TLADR11-
44	MODSEL4-	13	MODSEL 23-
45	MODSEL5-	14	OPEN
46	MODSEL6-	15	TLADR13-
47	MODSEL7-	16	MODSEL22-
48	MODSEL8-	17	TLADR08-
49	MODSEL9-	18	MODSEL 21-
50	CRUBIT7	19	TLADR09-
51	MODSEL10-	20	TLDAT11-
52	CRUBIT6	21	TLDAT08-
53	MODSEL11-	22	MODSEL 20-
54	CRUBIT5	23	TLDAT10-

Table C-2. 990/10 System Interface (AU 2) Pin Out (Continued)

Pin	Function	Pin	Function
P2-24	INT3-	P2-51	TLADR02-
25	TLADR18-	52	INT7-
26	TLHOLD-	53	TLADR03-
27	TLADR17-	54	INT8-
28	RESTART-	55	TLADR00-
29	TLADR16-	56	INT9-
30	SPARE	57	TLADR04
31	TLADR19-	58	INT10-
32	MODSEL 19-	59	TLADR05-
33	TLDAT09-	60	OPEN
34	MODSEL 18-	61	TLDAT04-
35	TLDAT02-	62	INT11-
36	MODSEL 17-	63	TLDAT05-
37	TLDAT03-	64	INT12-
38	MODSEL-16	65	INT13-
39,40	+12 MAIN	66	INT14-
41,42	-12 MAIN	67	TLDAT00-
43	TLDAT06-	68	INT15-
44	TLADR01-	69	TLDAT01-
45	TLDAT07-	70	SPARE
46	INT4-	71,72	-5 MEMORY
47	TLADR06-	73,74	+5 MEMORY
48	INT5-	75,76	+12 MEMORY
49	TLADR07-	77,78	+5 MAIN
50	INT6-	79,80	GND

C.4 990/10 SYSTEM ENVIRONMENTAL REQUIREMENTS – installed in 7-inch or 12-inch chassis.

- Operating:
  - Temperature: 0 to 50°C
  - Humidity: 0 to 95% non-condensing
- Storage:
  - Temperature: –40 to 70°C
  - Humidity: 0 to 95% non-condensing.

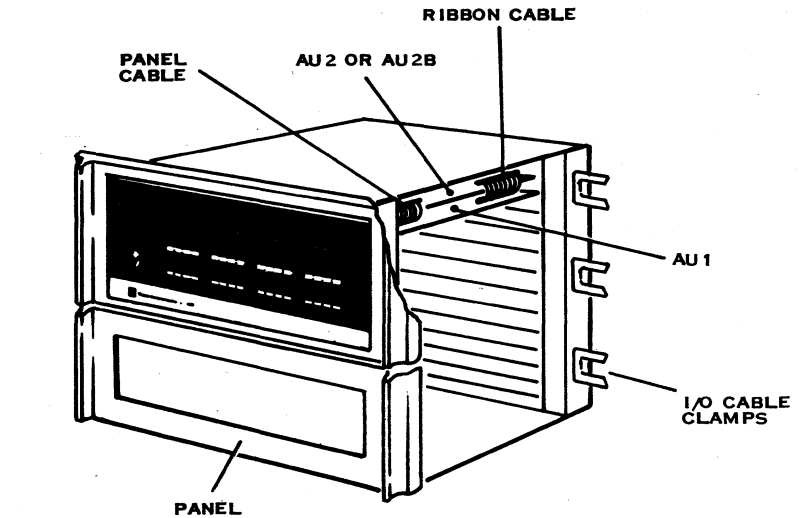


Figure C-1. 990/10 CPU Installed in 13-Slot Chassis

C.5 INSTRUCTION EXECUTION TIMES FOR THE 990/10

Instruction execution times for the 990/10 are a function of:

1. Clock cycle time ≅ 250 nanoseconds
2. Address mode of the operands
3. TILINE memory read and write cycle time
4. Mapped or unmapped instructions.

Table C-3 lists the number of clock cycle times and memory references to execute each 990/10 instruction. The column titled Time lists the number of clocks and memory cycles required without including the multiple address modes. To determine the additional number of clock and memory cycles required for modified addressing, add the appropriate values from the other columns titled Source Address Required, Source Operand Required and Destination Operand Required. An X in these columns indicates that the time for each address mode must also be added. The times for each address mode appear at the end of the table.

The total instruction execution time for an instruction is:

$$T - (C)(T_C) + (M)(T_M)$$

where:

C = total number of clock cycles

T_C = time per CPU clock cycle (250 ns)

M = total number of memory cycles

*T_M = time per memory cycle (725 ns - 990/10 RAM Memory Expansion Board)

As an example, the instruction MOVB with register addressing for both source and destination would be calculated as:

(C,M) = (3,2) + (0,1) + (1,1)

= (Instruction) + (Source operand) + (Destination operand)

= (4,4)

T = 0.250 (4) + 0.725 (4)
= 1.0 + 2.90 = 3.90 μs.

where 0.725 is the TILINE memory cycle time.

If the source was addressed in the symbolic mode:

(C,M) = (5,5)

T = 1.250 + 0.725 (5) = 4.875 μs.

*TILINE Memory Cycle Time (T_M)

- 990/10 RAM Memory expansion board - 725 ns.
- 990 EPROM expansion board - 755 ns.
- 990/10 Error correcting memory module - 825 ns.
- 990/10 Mapping Option - add 75 ns to memory cycle.

Table C-3. 990/10 Execution Times

Instruction	Condition	Time CLK,MEM	Source Address Required	Source Operand Required	Destination Operand Required	Other
A		2,2		X	X	
AB		3,2		X	X	
ABS	Source Positive	5,2	X			Operand Fetch Included in Time Column
ABS	Source Negative	5,3	X			Operand Fetch Included in Time Column
AI		4,4				
ANDI		4,4				
B		2,1	X			
BL		3,2	X			
BLWP		3,6	X			
C		5,1		X	X	
CB		5,1		X	X	
CI		6,3				
CKOF,CKON		5,1				
CLR		2,2	X			
COC,CZC		5,2		X		
DEC,DECT		2,2		X		
DIV	MIN.	41,5		X		

Table C-3. 990/10 Execution Times (Continued)

Instruction	Condition	Time CLK,MEM	Source Address Required	Source Operand Required	Destination Operand Required	Operand
DIV	MAX.	58,5		X		
DIV	Overflow	4,2		X		
IDLE	MIN.	4,1				
INC,INCT,INV		2,2		X		
J–	JUMP	3,1				
J--	Take Next Instruction	2,1				
LDCR		4,2		X		Add 1,0 for Each Bit Output
LDS,LDD		4,7	X			Plus Next Instruction
LI		3,4				
LIMI		3,2				
LMF		3,8				
LREX		6,6				
LWPI		3,2				
MOV		1,2		X	X	Destination Address Only– See "Destination MOV
MOVB		3,2		X	X	
MPY	MIN.	19,4		X		
MPY	MAX.	35,4		X		

Table C-3. 990/10 Execution Times (Continued)

Instruction	Condition	Time CLK,MEM	Source Address Required	Source Operand Required	Destination Operand Required	Other
NEG		3,2				
ORI		4,4				
RSET		5,1				
RTWP		3,4				
S		2,2		X	X	
SB		3,2		X	X	
SBO,SBZ		4,2				
SETO		2,2	X			
SLA	C≠0	3,3				Add 1,0 for Each Bit Position Shifted
SLA	C=0	5,4				
SOC		2,2		X	X	
SOCB		3,2		X	X	
SRA,SRC,SRL	C≠0	3,3				Add 1,0 for Each Bit Position Shifted
SRA,SRC,SRL	C=0	5,4				
STCR	C≤8	18,3		X		Add 1,0 for Each Ext. Bit Input
STCR	C>8	24,3		X		
STST,STWP		2,2				
SWPB		2,3	X			

Table C-3. 990/10 Execution Times (Continued)

Instruction	Condition	Time CLK,MEM	Source Address Required	Source Operand Required	Destination Operand Required	Other
SZC		2,2		X	X	
SZCB		3,2		X	X	
TB		5,2				
X		1,1	X			
XOP	Software	7,7		X		
XOP	Hardware	3,2		X		Plus Hardware Execution Time
XOR		3,2		X		
Interrupt Trap		6,6				
Source Address	T_S=0	0,0				
Source Address	T_S=1	0,1				
Source Address	T_S=2,S=0	1,1				
Source Address	T_S=2,S≠0	3,2				
Source Address	T_S=3	1,2				
Source Operand	T_S=0	0,1				
Source Operand	T_S=1	0,2				
Source Operand	T_S=2,S=0	1,2				
Source Operand	T_S=2,S≠0	3,3				

Table C-2. 990/10 Execution Times (Continued)

Instruction	Condition	Time CLK,MEM	Source Address Required	Source Operand Required	Destination Operand Required	Other
Source Operand	T_S=3	1,3				
Destination Operand	T_d=0	1,1				
Destination Operand	T_d=1	1,2				
Destination Operand	T_d=2,D=0	1,2				
Destination Operand	T_d=2,D≠0	3,3				
Destination Operand	T_d=3	2,3				
Destination MOV	T_d=0	2,0				
Destination MOV	T_d=1	2,1				
Destination MOV	T_d=2,D=0	2,1				
Destination MOV	T_d=2,D≠0	4,2				
Destination MOV	T_d=3	3,2				

APPENDIX D

CRU INTERFACE

APPENDIX D

CRU INTERFACE

D.1 GENERAL

The Communications Register Unit (CRU) is the direct command driven input/output interface for the Model 990 Computer family. The 990 CPU (990/4 or 990/10) originates and controls all data transfers on the CRU. The CRU is implemented using a 1-line serial input bus and a 1-line serial output bus. All data transfers on the CRU are synchronous with respect to the CPU. Table D-1 defines some terms used in this description.

Table D-1. CRU Terminology

Term	Definition
CRU Address	The CRU addresses referred to by this specification are absolute CRU addresses. Note that this is half the value of the CRU base address contained in workspace register 12. A CRU base address of $1E0_{16}$ would correspond to an absolute CRU address of $0F0_{16}$.
CRU Module	A group of 16 adjacent CRU addresses that are defined by a module select signal generated by either an AU or a buffer circuit card. A CRU module does not necessarily correspond to a circuit board as some CRU circuit boards contain 2 CRU modules.
Logic Levels and Loading	All logic levels and normalized loads are those for standard 7400 series logic as defined in the *Texas Instruments TTL Data Book.*
CRU Circuit Boards	A circuit board whose only interface to the computer is through the CRU. CRU circuit boards may be either full-sized or half-sized circuit boards.
CRU Peripheral Devices	A device that is not contained in the main or expansion chassis and interfaces to a CRU circuit board. Examples include line printers, modems, CRT Terminals, and teleprinters.

D.2 SPEED

The 990/10 CPU will operate the CRU at the maximum rate of 4 million bits per second (not including instruction overhead) for both input and output operations in the CPU chassis. To allow sufficient time for signal propagation in the interconnecting cable, the maximum input data rate is reduced to 2 million bits per second in CRU expansion chassis. The 990/4 CPU will operate the CRU at the maximum rate of 1.5 million bits per second (not including instruction overhead) for both input and output operations in any chassis. All 990 CRU devices are designed and tested to work with either the 990/10, the 990/4, or any future 990 AU.

D.3 CHASSIS/POWER SUPPLY INTERFACE SIGNALS

The chassis/power supply CRU interface signals are shown in relationship to the AU or buffer CRU interface in figure D-1. The following paragraphs define these signals.

D.3.1 TILINE POWER FAIL WARNING PULSE (TLPFWP–). The chassis/power supply generates TLPFWP– to indicate that a power failure is imminent. TLPFWP– is wired to the CPU and to all CRU connectors. TLPFWP– duration is 7.0 milliseconds. TLPFWP– is low true and is capable of driving at least 24 normalized TTL loads.

D.3.2 TILINE POWER RESET (TLPRES–). The chassis/power supply generates TLPRES– to indicate that the power supply voltages are unstable. TLPRES– is wired to the CPU and to all CRU connectors. TLPRES– is low true and is capable of driving at least 100 normalized TTL loads.

D.3.3 LOGIC GROUND. All power supply voltages and logic levels are referenced to logic ground. In addition the backplane in the chassis contains a logic ground plane on the connector side of the board to minimize cross talk and noise problems for CRU signals. Logic ground is electrically isolated from chassis or earth ground. This minimizes ground loop problems when two chassis of slightly different earth ground potentials must interconnect.

The fault protection capacitors between earth and logic grounds perform two functions. First, they bleed off high frequency noise on logic ground that would interfere with the operation of high frequency analog devices such as CRT displays. Second, the fault protection capacitors provide protection for the user. Should a potential difference of more than 30 volts DC or 15 volts AC exist between earth and logic grounds, the fault protection capacitors will short, trip the user's circuit breaker and thus prevent a potential shock hazard.

D.3.4 EARTH GROUND. Earth ground is supplied from the ground prong on the chassis power cord. Earth ground is connected to all exposed metal parts of the chassis. Earth ground is not available to any CRU circuit board.

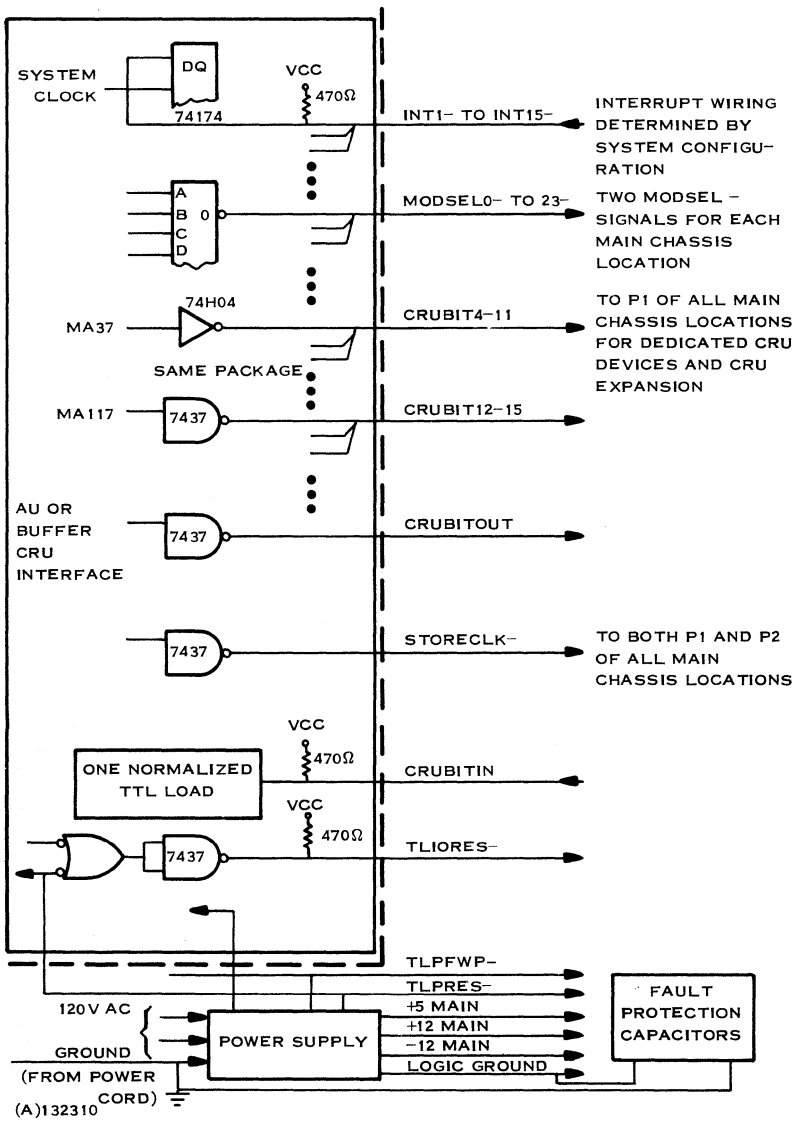


Figure D-1. CRU Interface Signals

D.4 CPU INTERFACE SIGNALS

The CPU originates and controls all data transfers on the CRU. All data on the CRU either originates from or is received by the CPU. In addition, the CPU performs the following CRU functions:

1. Decodes 24 module selects so that all available chassis locations of either a 6-slot chassis or a 13-slot chassis can be utilized for CRU modules without additional hardware.

2. Supplies the complete CRU address bus to connector P1 of all CPU chassis locations so that the CPU chassis can accomodate CRU expander circuit boards and other boards that require dedicated CRU addresses (Memory controller boundary registers, etc.).

3. Has the capability to accept interrupts from CRU modules on a prioritized basis.

Table D-2 summarizes the CRU signals. Refer to table F-1 for the complete pin-out of the General TILINE/CRU card slots.

D.4.1 PROCESSOR INTERRUPTS (INT1- THROUGH INT15-). The CPU board implements at least 7 prioritized vectored interrupts signals which may be used by CRU devices. All interrupt signals are low true and interrupt lines are terminated on the 990 CPU by a 470 ohm pull-up resistor and a flip-flop synchronized to the system clock. The synchronizing flip-flop presents one normalized TTL load to the interrupt signal.

D.4.2 CRU ADDRESS BUS (CRUBIT4- THROUGH CRUBIT15-). The CPU issues a 12-bit address which defines up to 4096 individual bits. The same 12-bit address is used for both input and output operations. Figure D-2 illustrates the field assignments for the 12-bit CRU addresses. All CRU address bus signals are high true. CRUBITS 4 through 11 are capable of driving 12 normalized TTL loads and CRUBITS 12 through 15 are capable of driving 30 normalized TTL loads.

D.4.3 MODULE SELECTS (MODSEL0- THROUGH MODSEL23-). The CPU generates 24 module select signals by decoding CRU bits 7-11. MODSEL0- corresponds to address 000_{16}. All module select signals are low true and are capable of driving 10 normalized TTL loads.

D.4.4 CRUBITOUT. CRUBITOUT is the serial data signal for transfer of data to the addressed CRU bits. CRUBITOUT is synchronized with the positive edge of STORECLK-. The CRUBITOUT signal is high true and is capable of driving 30 normalized TTL loads.

Table D-2. CRU Signals

Signature	CPU Pin	Description		
CRUBITOUT	P1,2-18	CRU Data Out From AU		
CRUBITIN	P1,2-60	CRU Data Into AU		
STORECLK-	P1,2-22	CRU Data Clock		
CRUBIT15	P1,2-34	CRU Bit Address	15	
CRUBIT14	P1,2-38		14	
CRUBIT13	P1,2-32		13	
CRUBIT12	P1,2-36		12	
CRUBIT11	P1-70		11	Used Only by CRU Expander
CRUBIT10	P1-68		10	
CRUBIT9	P1-64		9	
CRUBIT8	P1-62		8	
CRUBIT7	P1-50		7	
CRUBIT6	P1-52		6	
CRUBIT5	P1-54		5	
CRUBIT4	P1-56		4	
MODSEL0-	P1-45	Module Select For 1st 16 I/O		
MODSEL1-	P1-46	Module Select For 2nd 16 I/O		
MODSEL2-	P1-47	Module Select For 3rd 16 I/O		
MODSEL3-	P1-48	Module Select For 4th 16 I/O		
MODSEL4-	P1-23	Module Select For 5th 16 I/O		
MODSEL5-	P1-37	Module Select For 6th 16 I/O		
MODSEL6-	P1-49	Module Select For 7th 16 I/O		
MODSEL7-	P1-51	Module Select For 8th 16 I/O		
MODSEL8-	P1-53	Module Select For 9th 16 I/O		
MODSEL9-	P1-61	Module Select For 10th 16 I/O		
MODSEL10-	P1-67	Module Select For 11th 16 I/O		
MODSEL11-	P1-69	Module Select For 12th 16 I/O		
MODSEL12-	P1-73	Module Select For 13th 16 I/O		
MODSEL13-	P1-74	Module Select For 14th 16 I/O		
MODSEL14-	P1-43	Module Select For 15th 16 I/O		
MODSEL15-	P1-44	Module Select For 16th 16 I/O		
MODSEL16-	P2-38	Module Select For 17th 16 I/O		
MODSEL17-	P2-36	Module Select For 18th 16 I/O		
MODSEL18-	P2-34	Module Select For 19th 16 I/O		
MODSEL19-	P2-32	Module Select For 20th 16 I/O		

Table D-2. CRU Signals (Continued)

Signature	CPU Pin	Description
MODSEL20-	P2-22	Module Select For 21st 16 I/O
MODSEL21-	P2-18	Module Select For 22nd 16 I/O
MODSEL22-	P2-16	Module Select For 23rd 16 I/O
MODSEL23-	P2-13	Module Select For 24th 16 I/O

D.4.5 STORE CLOCK (STORECLK-). STORECLK- is derived from the CPU clock and is normally high. STORECLK- is enabled during write operations only. The positive edge of STORECLK- indicates to the selected CRU bit that the operation is a write operation (Set Bit or LDCR). The STORECLK- signal is capable of driving 30 normalized TTL loads. CRU address bits and MODSEL signals are stable at least 50 ns before STORECLK– goes low.

D.4.6 CRUBITIN. CRUBITIN is the serial data signal for transfer of data from the addressed CRU bit to the CPU. This signal is terminated on the CPU circuit board by a 470 ohm pull-up resistor and one normalized TTL load.

D.4.7 TILINE I/O RESET (TLIORES-). TLIORES- is a normally high signal that goes low for 100-500 nanoseconds when a RESET instruction is executed or goes low whenever TLPRES- goes low. TLIORES- is capable of driving 30 normalized TTL loads.

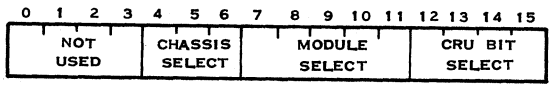


Figure D-2. CRU Address Field Assignments

APPENDIX E

9900 BUS

APPENDIX E

9900 BUS

E.1 9900 BUS

The 9900 Bus provides a high-speed, direct memory access capability for the TMS9900 and the 990/4 microcomputer. The 990/4 utilizes the 9900 Bus for CPU memory and also for memory expansion modules. The bus is a 3-state 16-bit bi-directional bus. DBIN is a logical "0" for a memory write operation. External devices may request the memory bus by setting HOLD- to a logic "0". HOLD- = 0 on processor input causes processor to complete the current cycle and place its output buffers into a high impedance state. HOLDA is set to a Logic "1" to indicate that the processor relinquished the 9900 Bus.

E.2 ADDRESS BUS

The Address Bus is a 3-state, 15-bit bus. This bus is used by the processor to transmit address to the memory. When HOLDA = 1, the processor Address Bus is placed in the high impedance state. The Address Bus is also used by the I/O System.

E.3 CONTROL SIGNALS

The following signals coordinate data transfer with the Data Bus:

- Data Bus In (DBIN) - A 3-state control line supplied by the processor to the memory system. When DBIN=1, memory enables its output buffer for a memory read operation.

- Write Enable (WE-) - A 3-state line supplied by processor to indicate valid memory write data is on the data bus. Assumes high impedance state when HOLDA=1.

- Memory Enable (MEMEN-) - A 3-state control line supplied by the processor to distinguish between memory address and I/O address. MEMEN- is used to generate chip enables for the memory devices. Placed in the high impedance when HOLDA=1.

- Ready - Ready is an input signal to the processor. Ready is generated in response to the memory cycle initiated by the processor. Ready is sampled by the processor during Phase 1 of T2. Refer to timing diagrams figures E-1 and E-2. If memory is not ready, (READY=0), the processor sets WAIT=1 and suspends internal processing until READY=1.

- Wait - Wait is generated by the processor to indicate a non-ready memory response.

- HOLD- - HOLD- is an input to the processor, when HOLD=0, indicating to the processor that an external device has requested the 9900 Bus. When HOLD=0 is received by the processor, Data Bus, Address Bus, WE- and MEMEN- are placed in the high impedance state. Processor suspends internal operation until HOLD- = 1.
- HOLD ACKNOWLEDGE (HOLDA) - HOLDA is set to a logic 1 by the processor in response to HOLD- = 0. HOLDA indicates that the processor buffers driving the Data Bus have been placed in high impedance state. See figure E-3 for timing sequence. HOLDA is wired in series through logic in each external DMA device to establish priority of bus access.

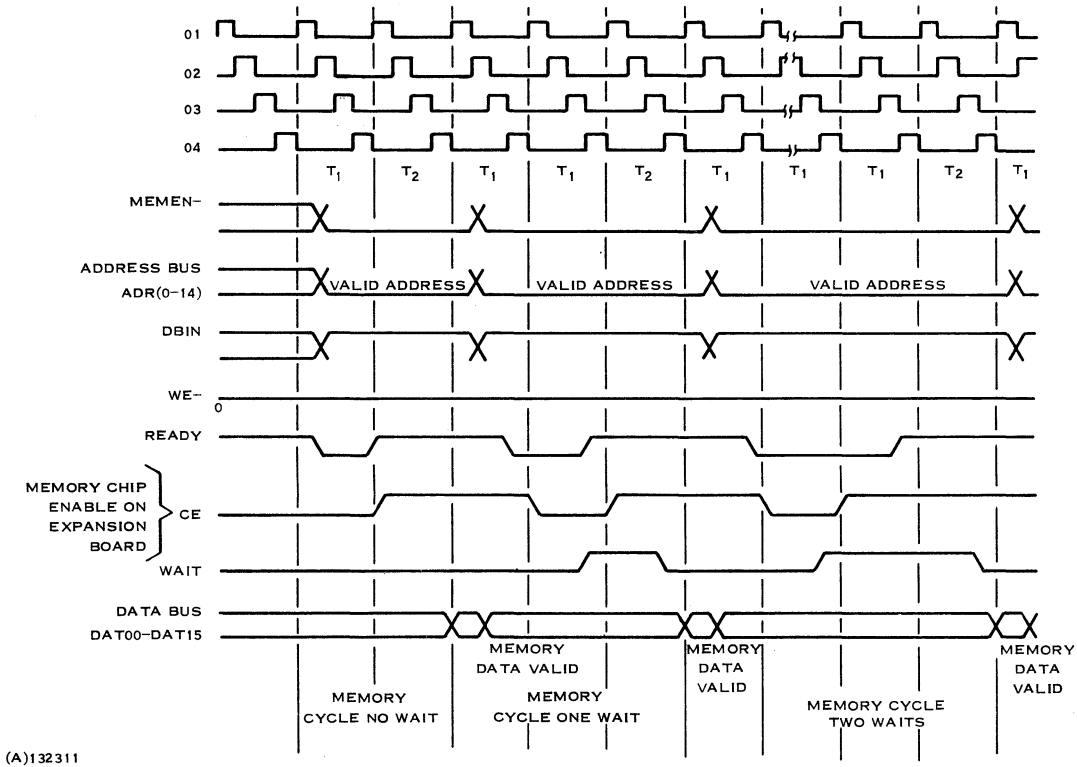


Figure E-1. 9900 Bus Read Cycle, Memory Cycle, One Wait

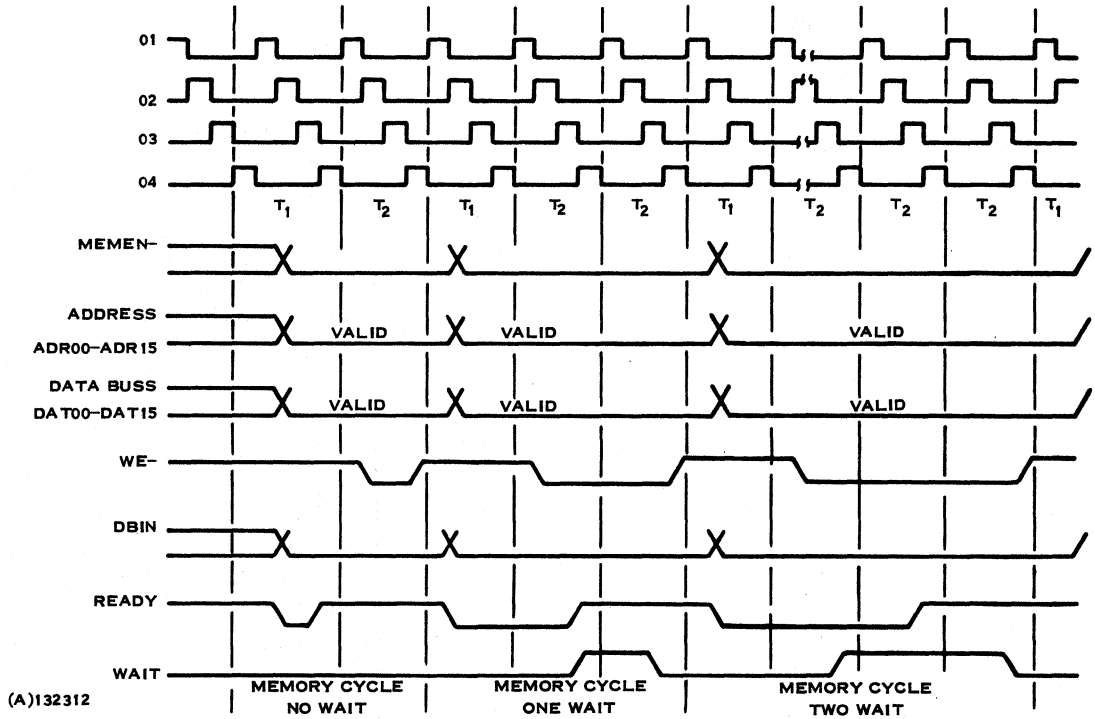


Figure E-2. 9900 Bus Write Cycle

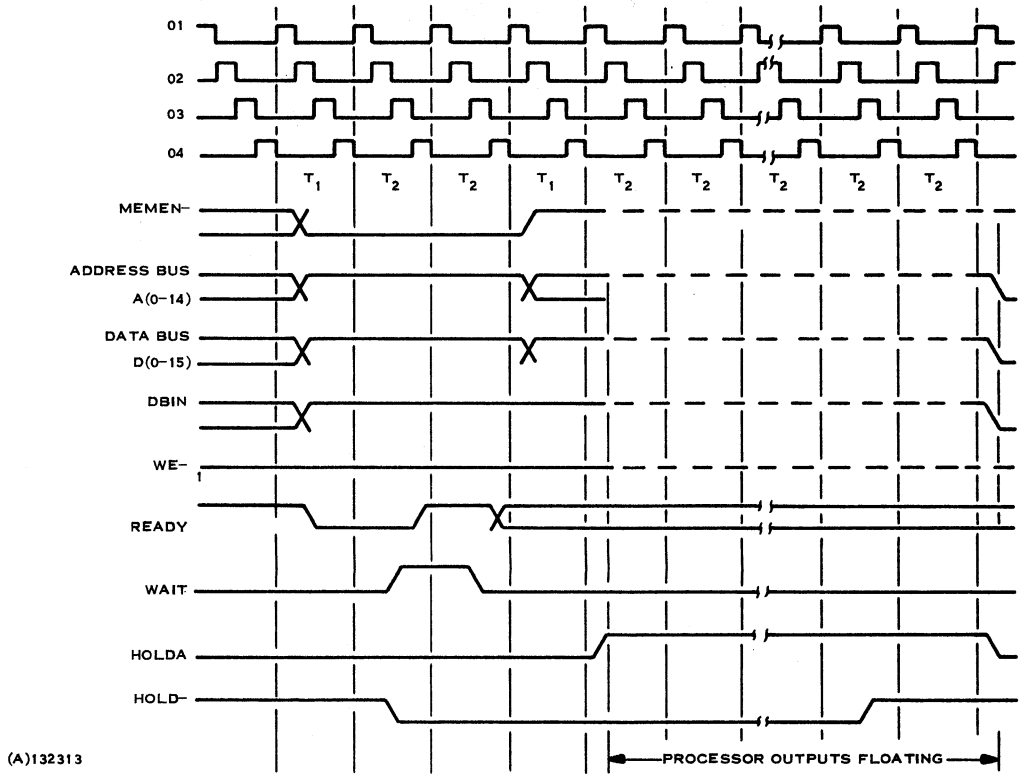


Figure E-3. 990 Bus Hold Operation

APPENDIX F

THE 990/10 TILINE BUS

APPENDIX F

THE 990/10 TILINE BUS

F.1 GENERAL DESCRIPTION

The TILINE is a high-speed 16-bit data transfer bus and associated control lines which serve to transfer data between all high-speed system elements. These elements include the central processor, the memory, and high-speed peripherals such as disc files. The TILINE also serves as a computer-to-computer link and is the backbone of multiprocessor systems.

The TILINE is asynchronous. The speed of data transfers over the TILINE is determined by distance and the speed of the devices interfaced to it. Consequently, system performance can be tailored to the application by suitable choice of elements and can be upgraded easily as needed.

Devices interfaced to the TILINE compete for access on a priority basis. High-speed peripherals are usually assigned highest priority and the central processor is assigned to the lowest. In operation, an efficient cycle-stealing action occurs. The overhead in switching from CPU access to another device is approximately 60 nanoseconds. This allows a very high rate of device switching without sacrificing a great deal of overall data bandwidth.

F.2 APPLICATION

The TILINE is fully implemented on higher capability Model 990 Computers where it is utilized as the sole path of data communication between all high-speed system elements. The CPU, the main memory, and all high-speed peripheral devices such as discs are directly interfaced to the TILINE. Slower peripherals, such as EIA-compatible devices, are interfaced through the Model 990 CRU (Communications Register Unit). As the Communication Register Unit is a direct extension of the Model 990 CPU mainframe, these slower peripheral devices are not part of the TILINE although they share the same backplane as the TILINE. The TILINE bus signals are supplied to every card slot in both the 6-slot and 13-slot chassis.

F.3 TILINE DEVICES

There are two classes of devices that interface to the TILINE: TILINE Master devices that control data transfers, and TILINE Slave devices that generate or accept data in response to some Master. Data transfers in either direction always occur between one Master and one Slave. The CPU is an example of a Master device and a memory module is an example of a Slave device. All Slave devices recognize and are activated by specific addresses. For example, a memory module is activated when some Master device does a read from an address within the boundaries of that memory module. Of course, the system must be configured so that only one Slave recognizes any particular address. In the case of memory modules, pencil switches on the module are set to indicate the starting address and size of the module.

Peripheral controllers are both Master and Slave devices. Control registers addressed as specific memory addresses near the high end of memory constitute the Slave part of the peripheral controller. The registers are loaded by the CPU with memory-to-memory move instructions. The registers specify the parameters of a peripheral data transfer. In the case of a disc, they specify disc address, the number of sectors of data to be transferred, the memory address to which the data is to be transferred and whether the data is to be read or written. One register in each peripheral controller is a status register for that controller. The bits in it indicate information such as "operation complete", "read error", and "illegal command". Other bits in the peripheral controller status register are set by the CPU to command the peripheral to start, stop, clear its interrupt, or reset. All of these registers are addressed by the CPU as consecutive words of memory at some specific address. For each peripheral controller the address of its registers is set by a set of pencil switches. When a peripheral controller is started by the CPU, it transfers data between memory and the peripheral device by cycle-stealing with the CPU and any other Master devices which may be active. When a peripheral controller needs to transfer a word of data over the TILINE, the Master device part of the peripheral controller must gain access to the TILINE and then may address a Slave (such as a memory module) and read from or write to it.

F.4 SIGNAL DEFINITIONS

The sub-sections which follow define the 48 signal lines that comprise the TILINE. These signals are described in three (3) groups according to their function. The three functions are:

- Master/Slave data transfers (40 signals)
- Acquisition of TILINE Mastership (3 signals)
- System control functions (5 signals)

F.4.1 DATA TRANSFER OPERATIONS. The following 40 signals are utilized exclusively for data transfer operations on the TILINE. Thirty-six of these signals are grouped together in two sub-configurations for the transfer of a 20-bit address and 16 bits of data while the remaining four signals are utilized for control of the actual transfer operations. All signals are transmitted and received between a TILINE Master device and a TILINE Slave device as defined in the following paragraphs.

F.4.1.1 TILINE Master to Slave Write Cycle. When a TILINE Master device has access to the TILINE, it may accomplish a memory (Slave) write cycle through the following action. The Master asserts TILINE Go, (TLGO–). At the same time the Master must assert a write command by pulling TILINE Read (TLREAD) low. The Master must also at this time generate valid write data (TLDAT–) and a valid 20-bit address (TLADR–).

All Slave devices interfaced to the TILINE will receive the Go transmitted from the Master. The Slave devices must decode the address to determine which Slave is being

addressed. This is accomplished in the Slave by generating a delayed Go (via a timer circuit) and using it to strobe for a valid address decode. In the case of a memory module, delayed Go and a valid address decode would generate an internal memory start signal. When the Slave device has delayed Go and decoded the address as valid, it must perform a write cycle and then assert TILINE Terminate (TLTM–). At the time the Slave device asserts Terminate, it must have finished using the write data, address, and read/write signals from the TILINE. The action described in the above paragraph happens during "time 1" of figure F-1.

When the TILINE Master device receives the asserted Terminate, it must release Go, Read, address, and write data. This action occurs during "time 2". At this time the Master device may relinquish the TILINE to another Master device. When the Slave device receives the release of Go, it must release Terminate. This is shown as "time 3". When the Master device receives the released Terminate, it may begin a new cycle if it has not relinquished the TILINE to another Master device. This is shown as "time 4".

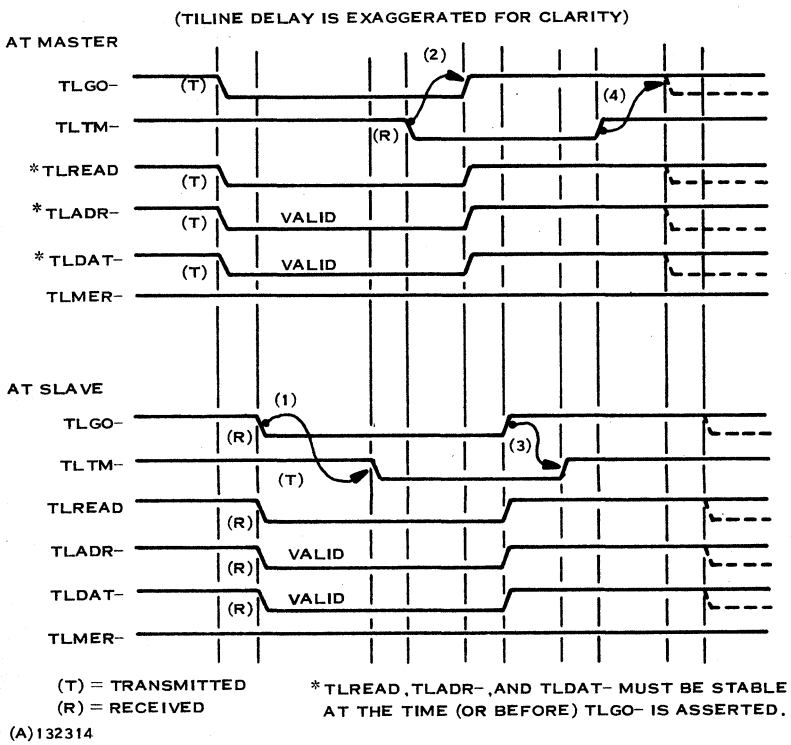


Figure F-1. TILINE Master to Slave Write Cycle

F.4.1.2 TILINE Master to Slave Read Cycle. When a TILINE Master device has access to the TILINE, it may accomplish a memory read cycle through the following action. The Master asserts TILINE Go (TLGO–), and a valid address (TLADR–).

All Slave devices will receive the Go transmitted by the Master. The Slave devices must delay Go and decode the address as for a write cycle. When this is done and the address is decoded as valid, the Slave device will begin to generate read data. In the case of a memory module this means starting a read cycle. When read data is valid, the Slave device must assert Terminate. At the time the Slave device asserts Terminate, it must be finished using the address and read/write signals. If a read error is detected during a read cycle, the Read Error signal (TLMER–) must be asserted by the Slave. This signal must have the same timing that read data would have. This action takes place during "time 1" of figure F-2.

When the TILINE Master device receives the asserted Terminate, it must release Go and the address. At the time the Master device releases Go, it must be finished using the read data and the TLMER– signals. This occurs during "time 2". At this time the TILINE Master device may relinquish the TILINE to another Master device. When the Slave device receives the released Go, it must release Terminate and read data. This is shown as "time 3". When the Master device receives the released Terminate, it may begin a new cycle if it has not relinquished the TILINE to another Master device. This is shown as "time 4".

F.4.2 TILINE BUS ACQUISITION. The three signals, TILINE Access Granted (TLAG), TILINE Acknowledge (TLAK–), and TILINE Available (TLAV), are utilized only by TILINE Master devices. Their purpose is to schedule the next TILINE Master during the last data transfer operation of the present TILINE Master. All TILINE Master devices are connected in order of priority. This interconnection is shown in figure F-3.

Every TILINE Master device has identical Access Control logic. Arbitration of TILINE access among TILINE Master devices is a distributed function. There is no central bus arbitrator and thus the TILINE does not require a CPU to function. All that is required is at least one Master and at least one Slave. This greatly increases application flexibility.

In operation, TLAG propagates access requests in series from high-priority Masters to low-priority Masters. An access request by a high-priority Master is seen by a low-priority Master as TILINE Access Granted (TLAG) low. TILINE Acknowledge (TLAK–) is asserted by a Master preparing for the next available bus cycle. TILINE Available (TLAV) is the indication to the Master asserting TLAK– that it may take access to the TILINE. The Master indicates that it has done so by taking TLAV low. When the Master has finished its data transfer(s) it may relinquish the TILINE by releasing TLAV to indicate to other Masters that the TILINE is now Available. By this scheme, the arbitration for access to the TILINE is overlapped with the preceding data transfer.

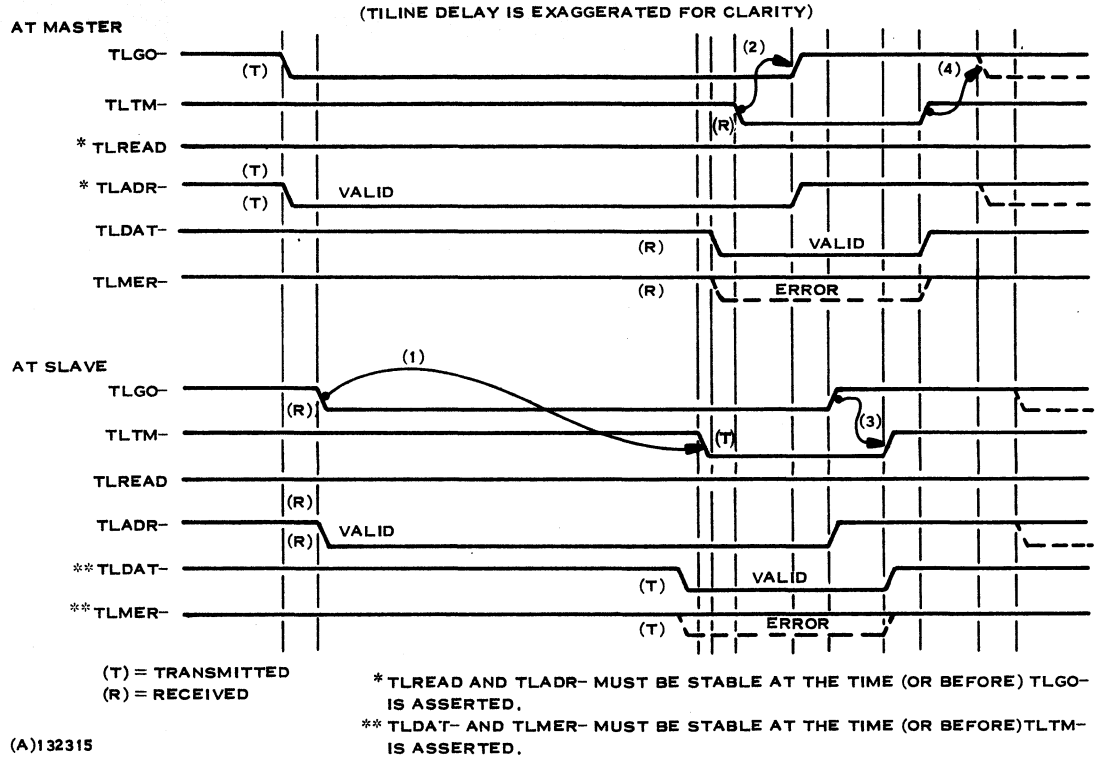


Figure F-2. TILINE Master to Slave Read Cycle

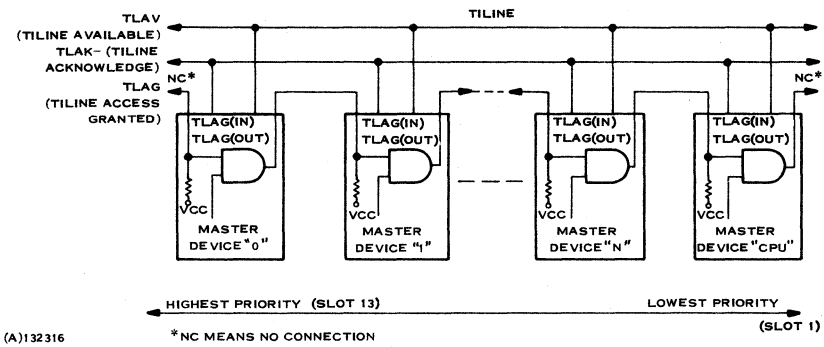


Figure F-3. TILINE Master Device Interconnections

F.4.3 I/O RESET. I/O Reset (TLIORES–) is generated by 990 CPUs during execution of the I/O Reset instruction. Its function is to halt and reset I/O devices. Devices should reset in an orderly fashion in response to this signal and any memory cycle in progress should be completed normally.

F.4.4 POWER FAILURE WARNING. The Power Failure Warning Pulse (TLPFWP–) is generated by the power supply to indicate that a power shut-down sequence is about to occur. This signal goes low at least 7 milliseconds before Power Reset is asserted. The leading edge of this pulse causes the CPU to trap via the power failure trap vector. TLPFWP– will remain asserted until Power Reset is asserted.

F.4.5 POWER RESET. Power Reset (TLPRES–) is a normally high signal that goes low at least 10 microseconds before any dc power voltage begins to fail due to normal shut-down or ac power failure. TLPRES– is generated by the power supply. This signal maintains a low level during and after power failure. During ac power turn-on, TLPRES– will remain at ground until after all dc power voltages are stable.

The purpose of TLPRES– is to reset all device controllers and the CPU during power failure and to directly inhibit all critical lines to external equipment powered by a separate power supply. It is TLPRES– that, for instance, prevents a tape deck from getting a Rewind pulse when the system is powered up and down. When TLPRES– goes high, indicating that power is up and stable, the CPU will perform a power-up interrupt trap.

F.4.6 TILINE WAIT. TILINE Wait (TLWAIT–) is used to resolve certain conflicts which can arise in computer-to-computer communication over the TILINE. It is a normally-high signal generated by TILINE Couplers. TLWAIT– allows TILINE Couplers to exercise a "higher than any" priority on the TILINE.

F.4.7 TILINE HOLD. TILINE Hold (TLHOLD–) is asserted by 990 CPUs when they are executing an ABS instruction. ABS is used as a software " interlock" in multi-processor systems. TLHOLD– is used and propagated by TILINE Couplers in multi-processor systems to ensure the integrity of the interlock.

F.5 GENERAL TILINE SLOT PIN-OUT

Table F-1 lists the pin-out of all of the chassis slots (except slot 1). It includes power pins, the TILINE Bus, the CRU Bus, and Interrupt pins.

F.6 CONFIGURING TILINE MODULES IN A 990 CHASSIS

Slot 1 in either the 6-slot or the 13-slot chassis will normally contain a 990/10 AU2 (or AU2B) board. In configurations without a CPU in a particular chassis, slot 1 of that chassis will contain either a TILINE Coupler board or a CRU Buffer Board. The signal line that establishes positional priority among TILINE Masters is wired along the P2 side of the chassis and is propagated through boards plugged in the chassis. The conductor for this signal (TLAG) shorts across all the P2 connectors (pin 5 to pin 6) except slot 1 (and slot 7 in the 13-slot chassis). The first TILINE master device plugged into the chassis (other than in slot 1) should go into slot 7. Additional TILINE master devices may be plugged into other slots so long as the etch between pins 5 and 6 of that slot is cut.

F.6.1 TILINE COUPLER CONFIGURATIONS. The TILINE link is used to configure a system in which TILINE modules are located in several different chassis. Reasons for configuring multi-chassis systems are:

- TILINE Memory expansions,
- TILINE Peripheral expansion
- Communication between 990/10 CPUs
- Sharing TILINE Peripherals and Memory among several CPUs
- Providing redundant back-up or fail-soft systems.

A TILINE link consists of a TILINE Coupler board in each of two chassis and a TILINE link cable connected between them. The orientation of the link cable establishes the direction in which possible communication deadlocks are broken. It also establishes the direction in which I/O Reset is propagated. For these reasons, some restrictions in multi-chassis configurations are necessary. In the figures below, a TILINE link is shown as a vector with the head of the vector indicating the direction of I/O Reset propagation and the direction of preferred communication in case of deadlock. A chassis is shown as a circle.

Table F-1. General TILINE/CRU Slot Pin-Out

Pin	Function	Pin	Function
P1-			
1,2	GND	P1-55	TLMER-
3,4	+5 MAIN	56	CRUBIT4
5,6	+12 MEMORY	57	GND
7,8	+5 MEMORY	58	TLAV
9,10	-5 MEMORY	59	GND
11	TLREAD	60	CRUBITIN
12	GND	61	OPEN
13	TLPRES-	62	CRUBIT8
14	TLIORES-	63	TLWAIT-
15	GND	64	CRUBIT9
16	TLPFWP–	65	OPEN
17	GND	66	INTERRUPTA–
18	CRUBITOUT	67	OPEN
19	GND	68	CRUBIT10
20	TLTM-	69	OPEN
21	GND	70	CRUBIT11
22	STORECLK-	71	TLAK-
23	OPEN	72	GND
24	GND	73	OPEN
25	TLGO-	74	GND
26	GND	75	OPEN
27	TLDAT12-	76	OPEN
28	TLDAT13-	77,78	+5 MAIN
29	120HZ	79,80	GND
30	TLDAT14-		
31	TLDAT15-	P2-	
32	CRUBIT13	1,2	GND
33	OPEN	3,4	+5 MAIN
34	CRUBIT15	5	TLAG(OUT)
35	OPEN	6	TLAG(IN)
36	CRUBIT12	7	GND
37	OPEN	8	TLADR14-
38	CRUBIT14	9	TLADR15-
39,40	+12 MAIN	10	TLADR10-
41,42	-12 MAIN	11	TLADR12-
43	OPEN	12	TLADR11-
44	OPEN	13	TLPRES–
45	OPEN	14	TLIORES–
46	MODSELB–	15	TLADR13-
47	OPEN	16	TLPFWP–
48	MODSELA–	17	TLADR08-
49	OPEN	18	CRUBITOUT
50	CRUBIT7	19	TLADR09-
51	OPEN	20	TLDAT11-
52	CRUBIT6	21	TLDAT08-
53	OPEN	22	STORECLK–
54	CRUBIT5	23	TLDAT10-

Table F-1. General TILINE/CRU Slot Pin-Out (Continued)

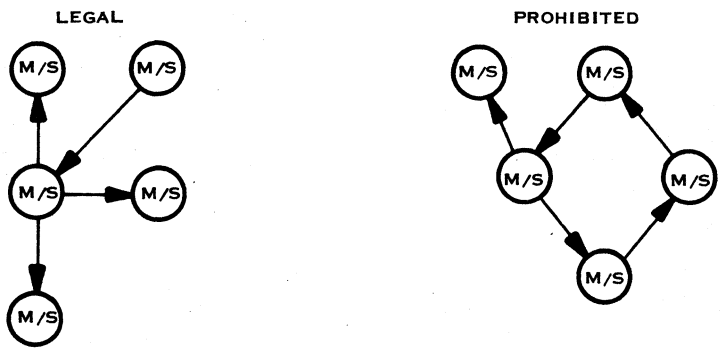
Pin	Function	Pin	Function
P2-24	GND	P2-51	TLADR02-
25	TLADR18-	52	OPEN
26	TLHOLD-	53	TLADR03-
27	TLADR17-	54	OPEN
28	OPEN	55	TLADR00-
29	TLADR16-	56	OPEN
30	GND	57	TLADR04–
31	TLADR19-	58	GND
32	CRUBIT13	59	TLADR05-
33	TLDAT09-	60	CRUBITIN
34	CRUBIT15	61	TLDAT04-
35	TLDAT02-	62	OPEN
36	CRUBIT12	63	TLDAT05-
37	TLDAT03-	64	OPEN
38	CRUBIT14	65	OPEN
39,40	+12 MAIN	66	INTERRUPTA–
41,42	-12 MAIN	67	TLDAT00-
43	TLDAT06-	68	OPEN
44	TLADR01-	69	TLDAT01-
45	TLDAT07-	70	OPEN
46	MODSELB–	71,72	-5 MEMORY
47	TLADR06-	73,74	+5 MEMORY
48	MODSELA–	75,76	+12 MEMORY
49	TLADR07-	77,78	+5 MAIN
50	OPEN	79,80	GND

KEY: ⟶ DENOTES A TILINE LINK

S — DENOTES A CHASSIS WITH SLAVE MODULES

M — DENOTE A CHASSIS WITH MASTER MODULES OTHER THAN TILINE COUPLERS

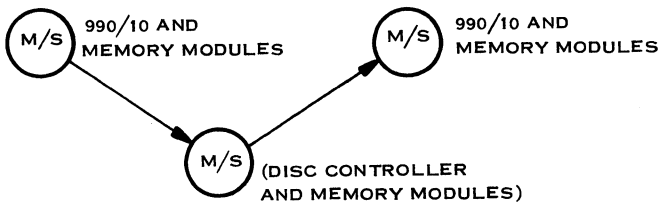
1) IN GENERAL, CLOSED LOOPS OF LINK VECTORS ARE PROHIBITED.



2) HOWEVER, IF ADDRESSING IS RESTRICTED BY CONVENTION OR BY SETTING LIMIT AND BIAS SWITCHES ON THE COUPLERS, CLOSED LOOP ARE PERMITTED SO LONG AS A TILINE NODE (CHASSIS) CANNOT ADDRESS ITSELF.

Figure F-4. TILINE Link Configurations

1) TWO CPU'S SHARING A DISC AND GLOBAL MEMORY.



2) MEMORY EXPANSION

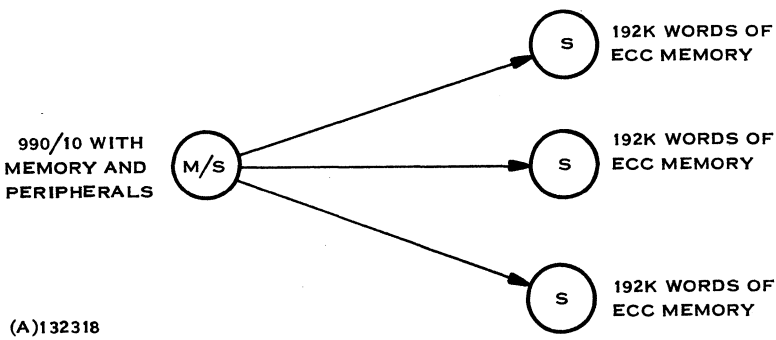


Figure F-5. Some Useful TILINE Configurations

APPENDIX G

CHASSIS SPECIFICATIONS

APPENDIX G

CHASSIS SPECIFICATIONS

G.1 GENERAL

This appendix describes the characteristics of the 990 chassis family. This family consists of three members:

- 13-Slot Chassis - This chassis can be used for computers, communication register unit expansion, and TILINE expansion. This chassis is complete with fans, wiring, power supply, and front panel. This chassis accommodates 13 full-sized logic boards.

- 6-Slot Chassis - This chassis can be used for computers. This chassis is complete with fans, wiring, power supply, and front panel. This chassis accommodates 6 full-sized logic boards.

- OEM Chassis - This chassis is a minimum configuration chassis consisting of a card support and a ba[illegible]el. This chassis accommodates 3 full-sized logic boards.

G.2 FRONT PANEL DESCRIPTION

There are three front panel configurations. These configurations are briefly described in the following paragraphs. Each of these configurations is housed in a 7.00 inch tall injection molded piece part. When used on the 13-slot chassis, the 7.00 inch front panel is accompanied by a 5.25 inch injection molded filler panel. No front panel is used on the OEM chassis.

G.2.1 PROGRAMMER PANEL. This panel gives the user full control of the CPU and enables him to modify memory. Any CPU that will be used for software debugging should have this panel. The programmer panel has the following features:

- Keylock switch with detent located positions for: Power Off - Unlock - Lock. Key may be removed in any position. Unlock position turns power on and gives user control of the machine. Lock position maintains power on and gives control of the machine to the machine.

- 16 Status and Memory Data LEDs

- Fault Indication LED

- Run LED (indicates user or machine control of CPU)

- Idle LED (indicates that machine is processing)

G.2.2 OPERATOR PANEL. This panel gives the user more limited control than the programmer panel over the machine. This panel is also used for chassis not containing a CPU. The operator panel has the following features:

- Keylock Switch with the following positions: Power Off - On - Load. Power Off and On positions are detent located. Load position is momentary. Key may be removed in any position. The On position turns the power on. The Load position maintains power on and executes program in ROM for quick cold starts.
- Power On LED
- Fault Indication LED

G.3 COOLING CAPACITY

The 13-slot chassis and the 6-slot chassis will each support a heat load of 50 watts in each full width board position. The 13-slot chassis is designed to support a heat load of 240 watts in the power supply area. The 6-slot chassis is designed to support a 170 watt heat load in the power supply area. The maximum exhaust air temperature is designed to be 65°C (149°F) for manufacturing and office configurations.

G.4 COOLING REQUIREMENTS OF THE OEM CHASSIS

The OEM chassis is not equipped with fans. Therefore it must be supplied with an average input air flow of 600 feet per minute through the chassis and over the power supply. When so supplied, each full width board position will support a 50 watt heat load and the power supply 170 watt heat load. Maximum exhaust temperature should be 65°C (149°F).

G.5 FILTER MAINTENANCE

The filters are attached to the 13-slot chassis and the 6-slot chassis by four spring clips. Filters may be removed by pulling the filter off the chassis. In the case where the 6-slot chassis is configured as a table top model, it is necessary to first remove the table top cover.

G.6 ENVIRONMENTAL REQUIREMENTS

Operating temperature:	0 to 50°C
Humidity:	0 to 95% non-condensing
Storage temperature:	-40 to 70°C
Humidity:	0 to 95% non-condensing.

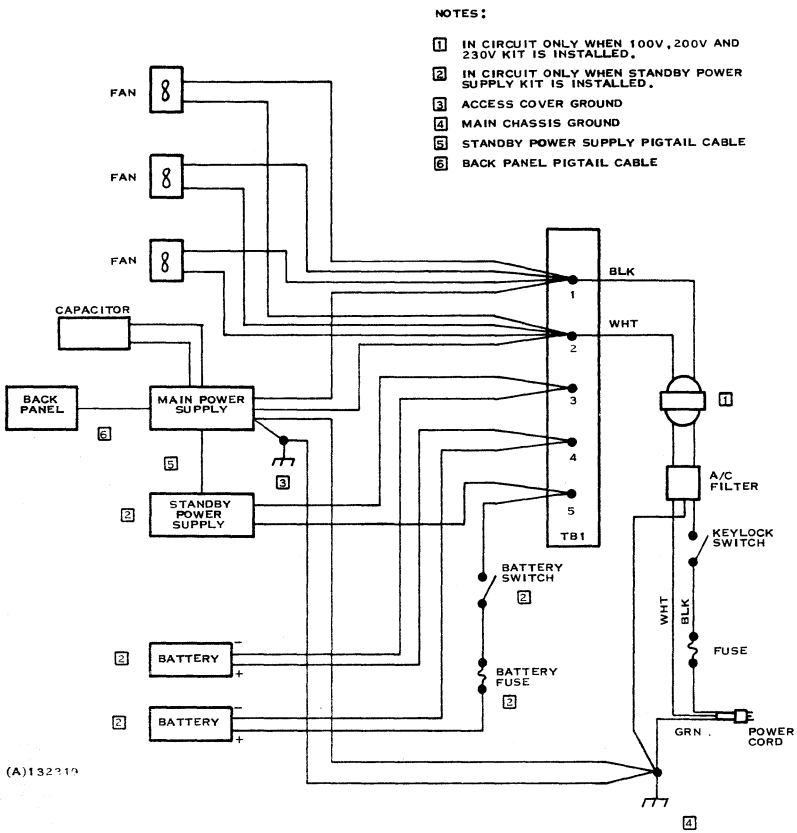


Figure G-1. Chassis Wiring for 6-Slot or 13-Slot Chassis

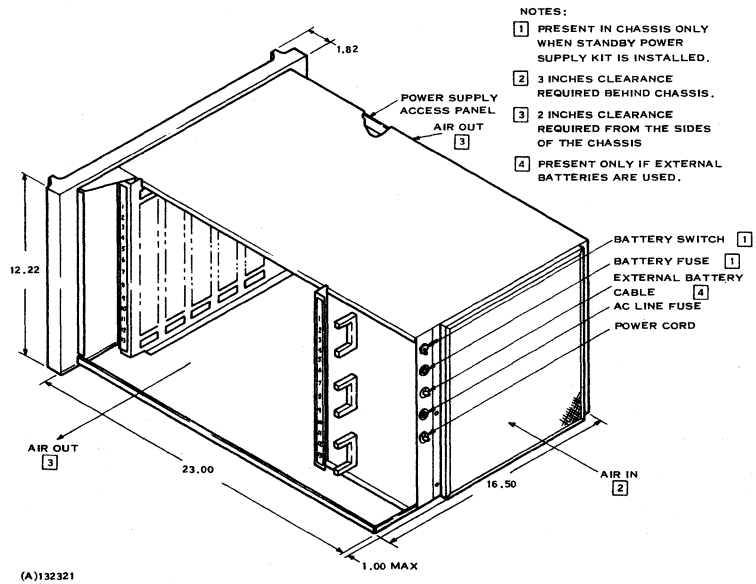


Figure G-2. 13-Slot Chassis Physical Configuration

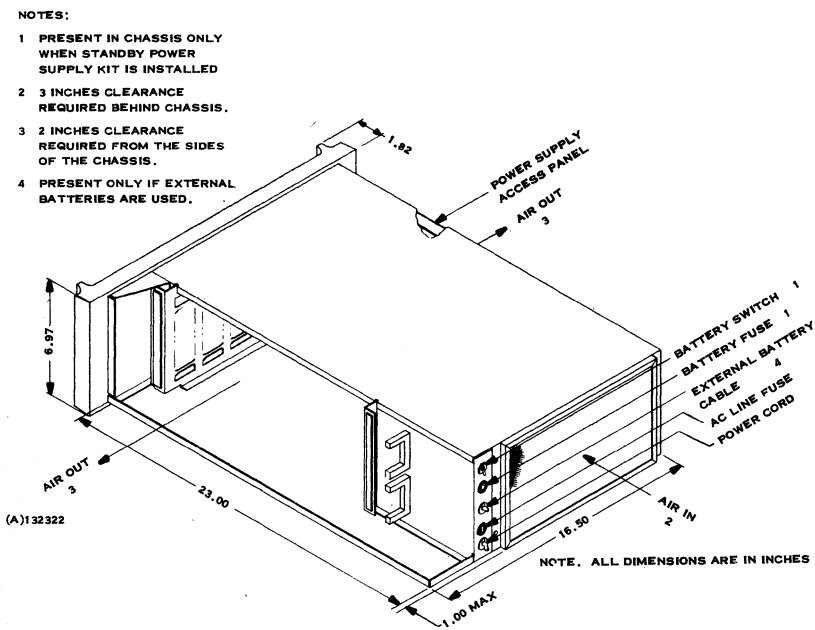


Figure G-3. 6-Slot Chassis Physical Configuration

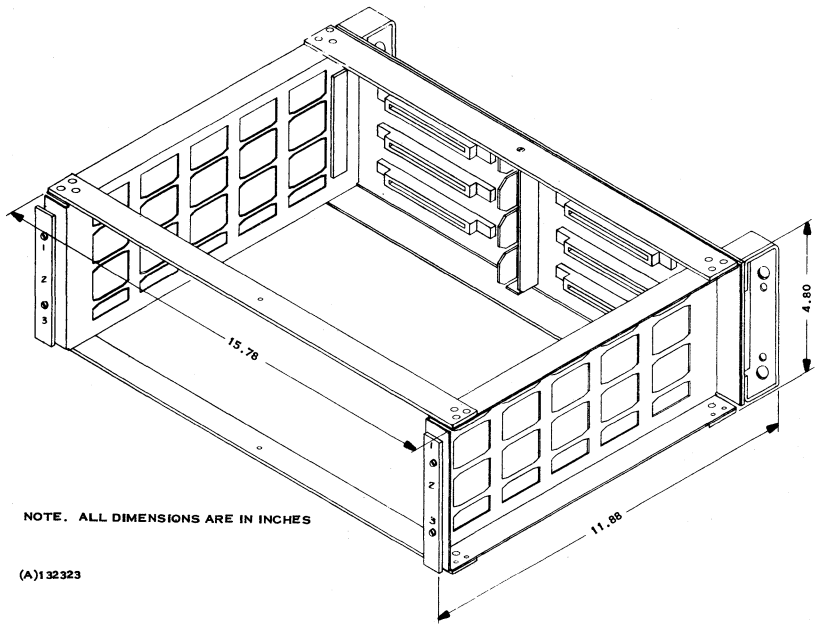


Figure G-4. OEM Chassis Physical Configuration

APPENDIX H

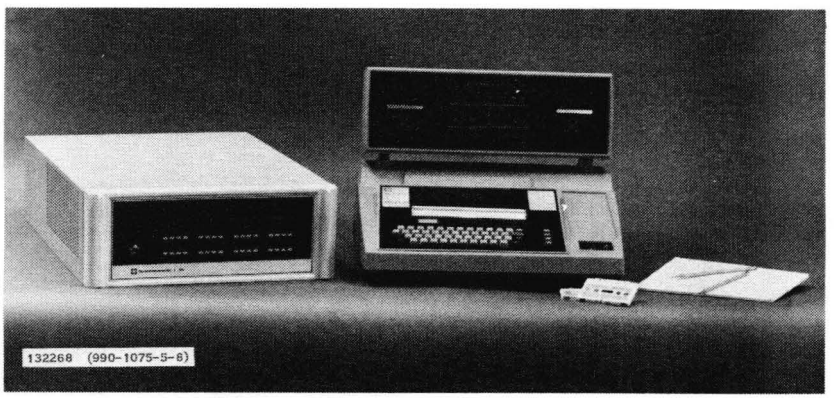
STANDARD SYSTEM CONFIGURATIONS

APPENDIX H

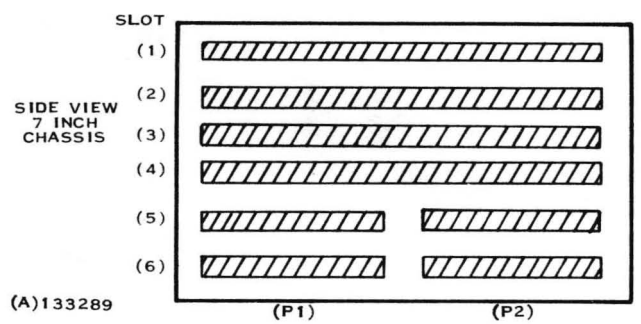
STANDARD SYSTEM CONFIGURATIONS

H.1 GENERAL

This appendix describes the chassis configuration of three standard 990 System Packages. It shows the card position assignments for various peripheral options, how they are addressed and the interrupt level assignments. Standard TI software is designed to operate for these configurations.



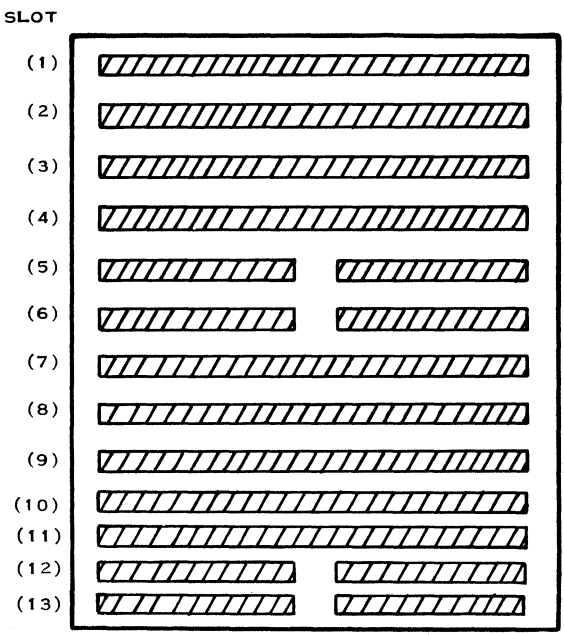
H.2 990/4 PROTOTYPING SYSTEM AND 990/4 PROGRAM DEVELOPMENT SYSTEM



Slot	Board	CRU Base	Interrupt Level
1	990/4	NA	D,1,2,5 (clock)
2	Expansion Memory/Write Protection	$IFA0_{16}$	2
3	913 VDT (option)	$C0/E0_{16}$	3
4	Floppy Disc (option)	$80/A0_{16}$	7
5P1	Line Printer (option)	60_{16}	–
5P2	Card Reader (option)	40_{16}	4
6P1	PROM Programmer (option)	20_{16}	–
6P2	733 ASR	00_{16}	6

H.3 990/10 PROGRAM DEVELOPMENT SYSTEM

SIDE VIEW
12 INCH
CHASSIS

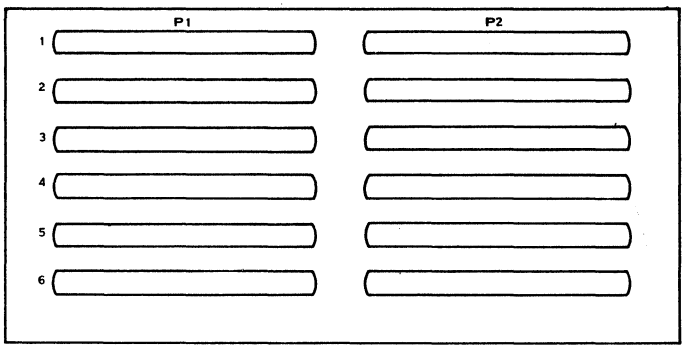


(A)133288

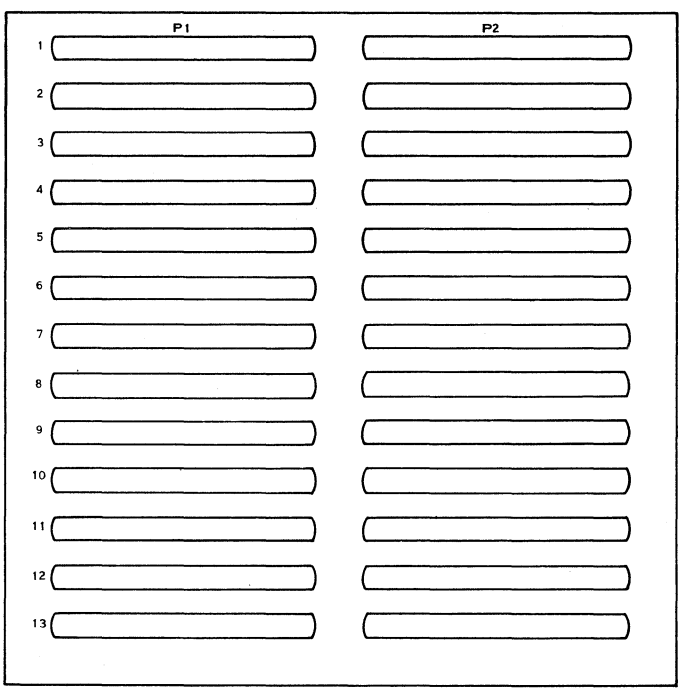
Slot	Board	CRU Base TILINE Base	Interrupt Level
1	990/10 AU2(B)	N/A	0,1,2,5 (clock)
2	990/10 AU1	N/A	
3	Memory	N/A	2
4	Memory	N/A	2
5	Memory (option)	N/A	2
6	Memory (option)	N/A	2
7	DS31 Disc	$FFC00_{16}$	13
8	913 VDT #3 (option)	$140/160_{16}$	9
9	913 VDT #2 (option)	$100/120_{16}$	10
10	913 VDT #1	$C0/E0_{16}$	11
11	Floppy Disc (option)	$80/A0_{16}$	7
12 P1	Line Printer (option)	60_{16}	14
12 P2	Card Reader (option)	40_{16}	4
13 P1	PROM Programmer (option)	20_{16}	15
13 P2	733 ASR	00_{16}	6

ALPHABETICAL INDEX

CHASSIS LAYOUT WORKSHEET

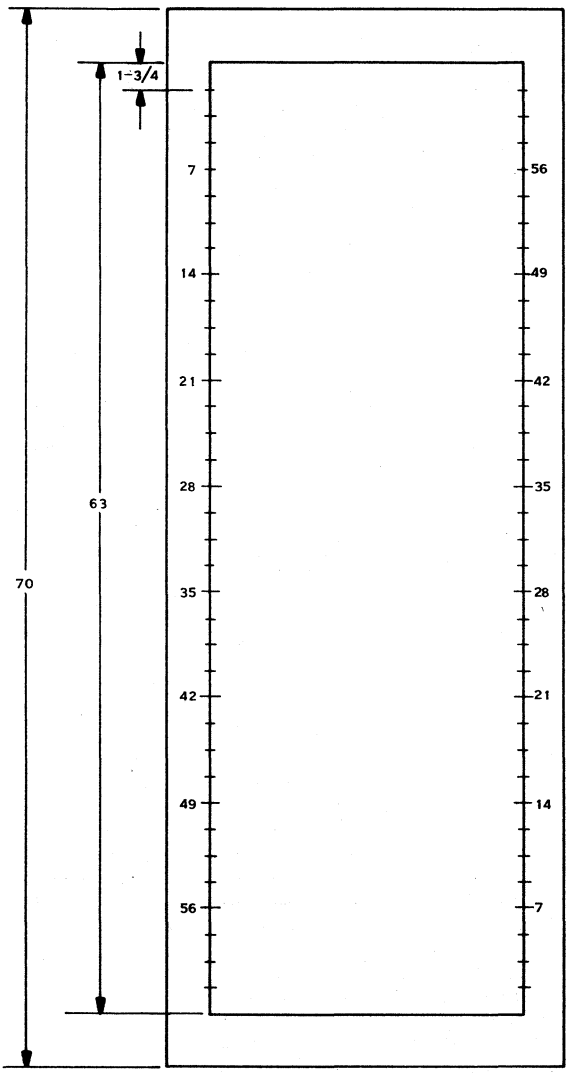


SIX-SLOT CHASSIS



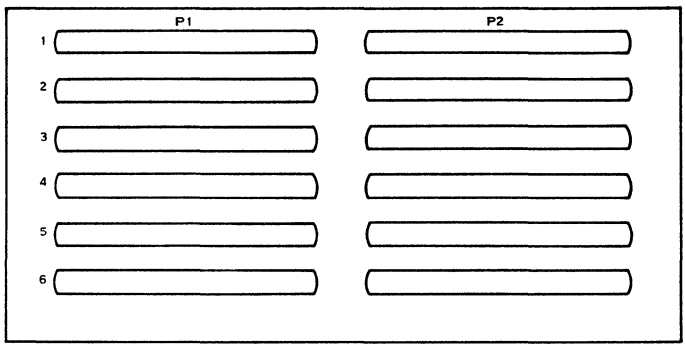
THIRTEEN-SLOT CHASSIS

RACK-MOUNT CABINET WORKSHEET

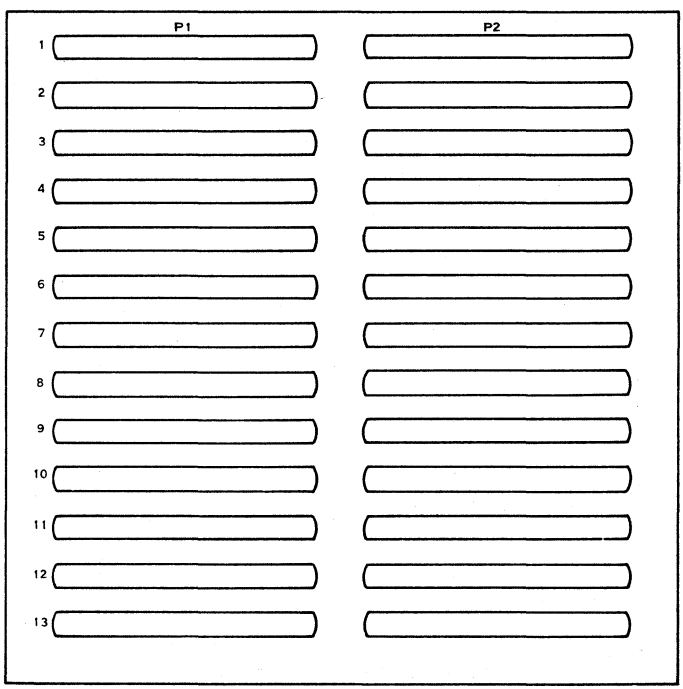


NOTE: ALL DIMENSIONS ARE IN INCHES.

CHASSIS LAYOUT WORKSHEET

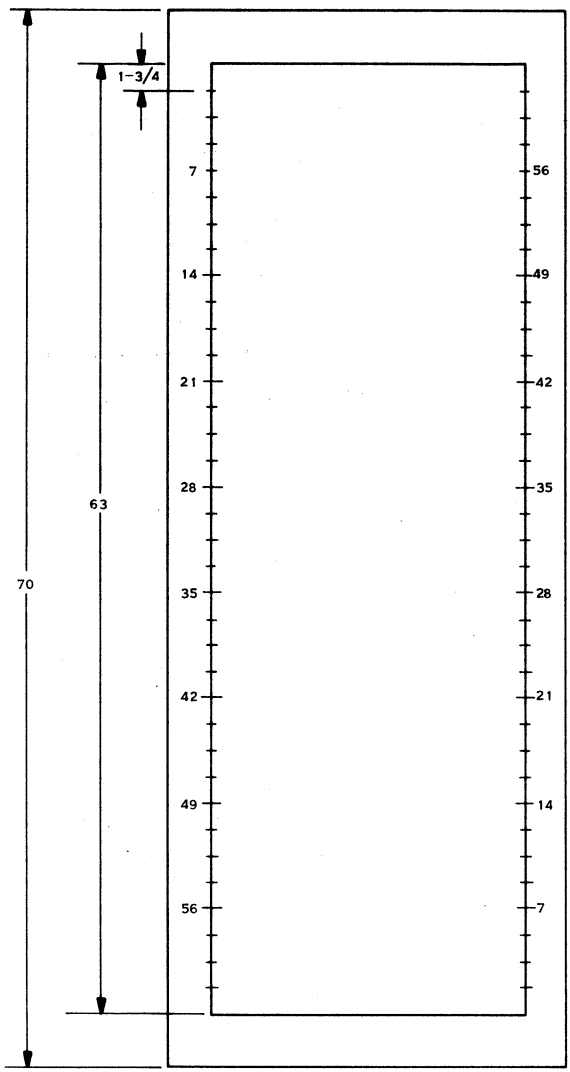


SIX-SLOT CHASSIS



THIRTEEN-SLOT CHASSIS

RACK-MOUNT CABINET WORKSHEET



NOTE: ALL DIMENSIONS ARE IN INCHES.

NOTES

NOTES

TEXAS INSTRUMENTS
INCORPORATED
DIGITAL SYSTEMS DIVISION
POST OFFICE BOX 2909 • AUSTIN, TEXAS 78767

TI-439-25M-5/76

PRINTED IN U.S.A.