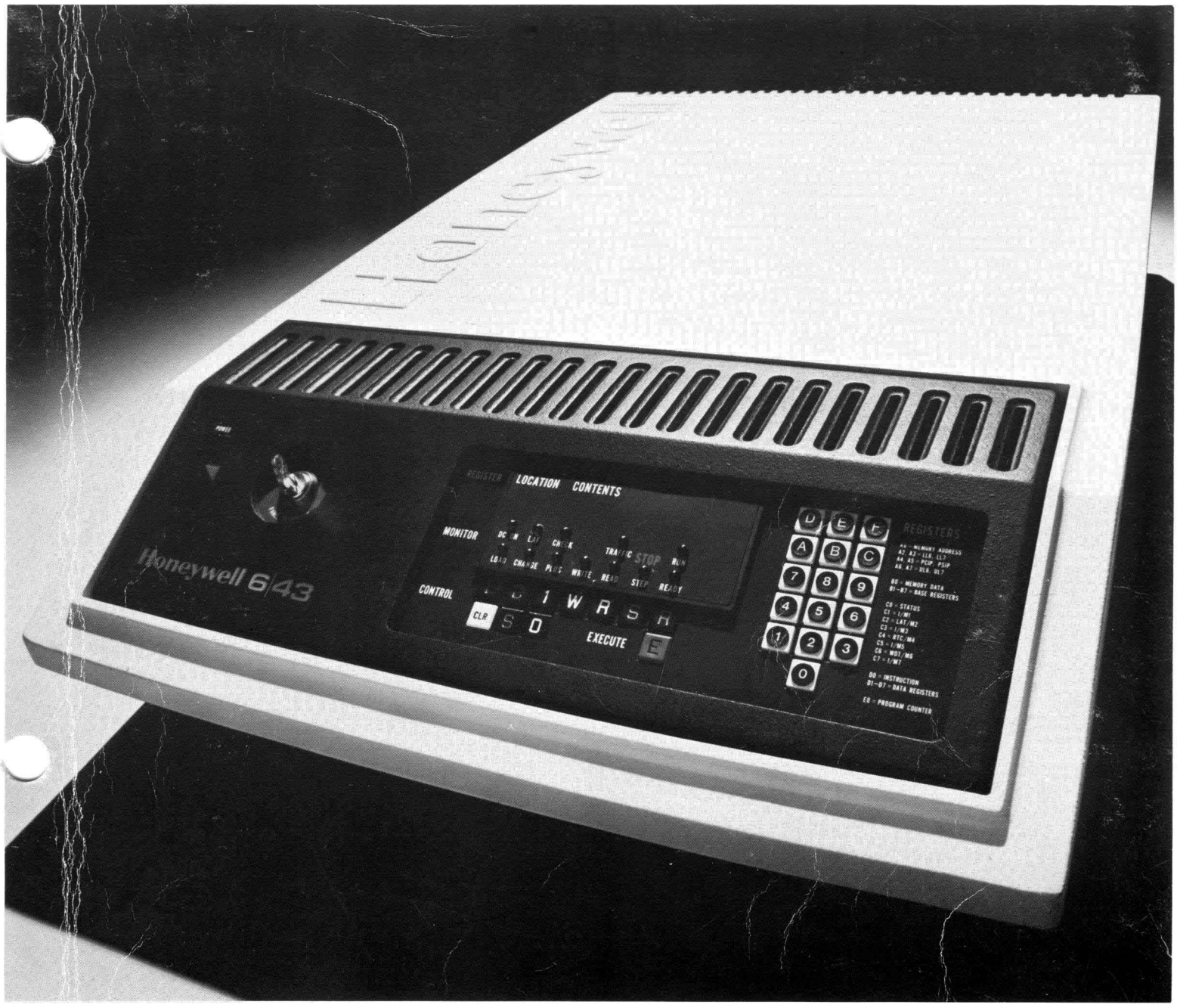


Honeywell Level 6 Minicomputer Handbook

SERIES 60 (LEVEL 6)

MINICOMPUTER HANDBOOK

SUBJECT

Introduction and Overview of the Models 6/34, 6/36 and 6/43 System Components, Architecture, and Software; Programmer-oriented Discussion of Instruction Set and Peripherals

SPECIAL INSTRUCTIONS

This manual supersedes the previous edition, AS22, Rev. 1, dated August 1976. This edition has been so extensively revised that change bars were not used. The Site Preparation Planning and Site Facility Listing sections have been removed from the manual; this information is now contained in the Site Preparation Manual, Order Number AY52.

ORDER NUMBER

AS22, Rev. 2

September 1977

Honeywell

PREFACE

This Handbook introduces Level 6, Honeywell's family of minicomputer systems, and describes their major system elements in straightforward technical detail. Thus, present minicomputer users and others with an engineering orientation can understand the fundamental design philosophy of Level 6 and make a firm personal evaluation of its merits. They will also become reacquainted with Honeywell as a minicomputer manufacturer.

Honeywell has been a minicomputer manufacturer for over 10 years. In that time it has marketed the products of Computer Control Company, which it acquired in 1966, and has developed and introduced extensions of the original line. Computer Control Company in 1965 built one of the first 16-bit minicomputers, the DDP-116. Honeywell has continued with the DDP-516 (1966), the DDP-416 (1967), the H316 (1969), and System 700 (1972), establishing itself as one of the foremost manufacturers of 16-bit systems.

Level 6 is a 16-bit minicomputer system founded on new concepts of simplicity, flexibility, and reliability. With its open-ended system architecture, its modular, highly functional software, and its low-cost peripherals (including diskettes), Level 6 presages a family of machines with wide-ranging suitability for both end-use and OEM applications. And as a partner with Honeywell's Series 60 and a member of the Honeywell Information System, Level 6 enjoys the full field engineering service of one of the country's major computer manufacturers.

The heart of Level 6 is the Megabus, expandable to 23 slots – each with multiple device capability (diskettes, printers, card readers, communication controllers, etc.). Modularity has reached new levels, with implementation of an entire central processor (up to 32,768 words of memory) on a single circuit board. Indeed, a complete minisystem of five boards (central processor, memory, and device and communication controllers), plus power supply, air circulating unit, and control panel, fits into a standard drawer 5¼ inches high.

Lying behind the advanced technology of Level 6 are not only Honeywell's long experience as a producer of 16-bit machines, but a design philosophy emphasizing the use of commercially established elements and a new factory system promoting error-reducing automated procedures. These have combined to lower manufacturing costs to the point at which Level 6 can offer one of the most attractive cost/performance ratios on the market.

There is much more that is newsworthy about Level 6, and this Handbook can only summarize key aspects. For a full account of Honeywell's new minicomputer and its place in your operations, call a Honeywell Marketing Representative.

File No.: 1S03

AS22

CONTENTS

ILLUSTRATIONS

TABLES

SECTION 1

SYSTEM SUMMARY

Level 6 engineering is highly contemporary, beginning with a Megabus permitting high-speed asynchronous data transfer and continuing with MOS and core memory, an extensive use of microprocessors, a large number of program-visible registers, a very large addressing capability, and an instruction set designed for extremely efficient programming. These and other aspects of Level 6 are described briefly in this section of the handbook and taken up in detail in the sections that follow.

MODELS

Four models are presently available: 6/06, 6/34, 6/36, and 6/43.

Model 6/06 is designed for end users, including present users of System 700. In addition to its Level 6 characteristics, Model 6/06 is compatible with System 700, allowing users to operate OS/700 software and DMA peripherals while taking advantage of the large capacity of the Megabus. Model 6/06 is documented extensively (see the *Model 6/06 Programmer's Reference Manual,* Order No. AT25) and will not be treated in this handbook. Table 1-1 compares the three processors discussed in this handbook.

6/34

Model 6/34 includes a central processor mounted in a four-slot Megabus chassis, a memory controller with parity, and an 8192-word MOS Memory-Pac (module). The central processor and directly addressable maximum memory (32,768 words, in modules of 8192 words) each require only one circuit board. Core memory is offered as an option in lieu of the MOS memory. Up to four boards can be included in Model 6/34; thus after the CP and memory boards are plugged in, two boards are still available for the attachment of multiple peripheral devices and/or communication lines. The new, extremely economical Level 6 power supply (including an air circulating unit), a basic or full control panel, multiply/divide hardware, a real-time clock, and a ROM bootstrap loader are included in this model.

6/36

Model 6/36 includes a central processor mounted in a five-slot or ten-slot Megabus chassis expandable to 23 slots; a basic or full control panel; multiply/divide hardware; a real-time clock; and the ROM bootstrap loader. Options include a memory controller with either parity or EDAC[1] memory; 8192-word Memory-Pacs providing a directly addressable maximum of 65,536 words, a core memory controller with 16KW or 32KW, a watchdog timer, and all the options available with Model 6/34. The central processor requires one circuit board; maximum memory requires two boards. Up to 20 additional boards can be attached for controllers of peripheral

TABLE 1-1. MODEL COMPARISON

Description \ Processor	6/34	6/36	6/43
No. of Megabus slots	4	5 or 10	5 or 10
Expansion	–	to 23 slots	to 23 slots
Minimum memory (words)	8K	8K	16K
Maximum memory (words)	32K	64K	1024K
EDAC memory	–	Optional	Optional
Double-fetch memory	–	–	Optional
Watchdog timer	–	Optional	Standard
Scientific Instruction Processor	–	–	Optional
Memory Management Unit	–	–	Optional
Typical instruction times (μ)			
With single-fetch memory	3.2	3.2	2.5
With double-fetch memory	–	–	2.0

[1] Error Detection and Correction

devices and/or communication lines. The Level 6 power supply and air circulating unit are included in the assembly housing the central processor.

6/43

Model 6/43 includes a central processor mounted in a five-slot or ten-slot Megabus chassis (expandable to 23 slots), a basic or full control panel, multiply/divide hardware, a real-time clock, a watchdog timer and a ROM bootstrap loader (see Figure 1-1).

The 6/43 offers parity or EDAC memory, with a maximum of one million words of directly addressable memory. Users may choose different memory controllers to enable a single-fetch or double-fetch capability with both types of memory. Using the single-fetch technique, memory is increased by adding 8K-word Memory-Pacs; with the double-fetch memory, these Memory-Pacs are added two at a time. Core memory is also available as an option. The 8K MOS, 16K MOS double-fetch and 16K/32K core memory may be mixed on the same system, but not on the same controller. Additional options include the Multiple Device Controller, the Multiline Communications Processor, the Mass Storage Controller, the Magnetic Tape Controller, the Scientific Instruction Processor, the Memory Management Unit, and the General Purpose DMA Interface. Also available are the associated Device-Pacs and Communications-Pacs, all Level 6 peripherals and their connector cables, the power supply, cabinetry, and accessories.

CONFIGURABILITY

The physical configurability of these models is extremely flexible, reflecting the variety of uses to which Level 6 can be put. The overriding consideration has been to provide a low-cost, modular, building-block system with minimal space and power requirements. Functional elements plug into or attach to each other without difficulty, needing little time and skill for assembly, disassembly, and replacement. These characteristics make Level 6 attractive to:

1. Original equipment manufacturers, who demand a minicomputer easily embedded in their own systems, operable and durable in a wide range of environments, and including the ready availability of spare parts and simple repair procedures.
2. Users who will place a Level 6 central processor in their own racks or attach it as a freestanding module, as a tool for various technical tasks (e.g., to collect, communicate, control, or manipulate data in factory and laboratory environments).

Figure 1-1. Model 6/43 Configuration

In addition, the peripherals available with Level 6, along with the program development and other software, make Level 6 attractive to:

1. Users requiring a relatively complete system, including I/O equipment, for dedicated applications in factory and other non-office environments.
2. Users requiring turnkey systems in various hardware configurations for use in an office environment.
3. Users requiring a program development capability supporting the above applications and generally used in an office environment.

The packaging of the Level 6 models was designed to satisfy all of these requirements. Available are:

- o Drawer units 5.25 in. (13.3 cm) and 10.5 in. (26.7 cm) high for standard EIA rack mounting
- o Cabinets 60 in. (152 cm) high for central processor, diskette, and other drawers
- o Tabletop configurations, completely enclosed, completely portable, 5.5 in. (13.9 cm) high, 19.5 in. (49.5 cm) wide, 29.7 in. (75.4 cm) deep
- o Office packaging, custom-shaped desk 29.5 in. (74.9 cm) high; 49 in. (124 cm) wide; 33 in. (83.3 cm) deep; integrated keyboard (optional); concealed control panel; 29.5 in. (74.9 cm) high standard EIA rack space. See Figure 1-2.

All provide easy front access and require only front-to-rear airflow for cooling. (In many other minicomputer systems, large backboard areas force designers to use side-to-side cooling with the attendant difficulties of avoiding recirculation in cabinet configurations.)

The software set is also designed to help keep costs down. Every module operates within the basic 8192-word memory of both models, except for the FORTRAN compiler, which requires 16,384 words. Program development requires only the addition of a teleprinter (or keyboard typewriter console) and a dual diskette.

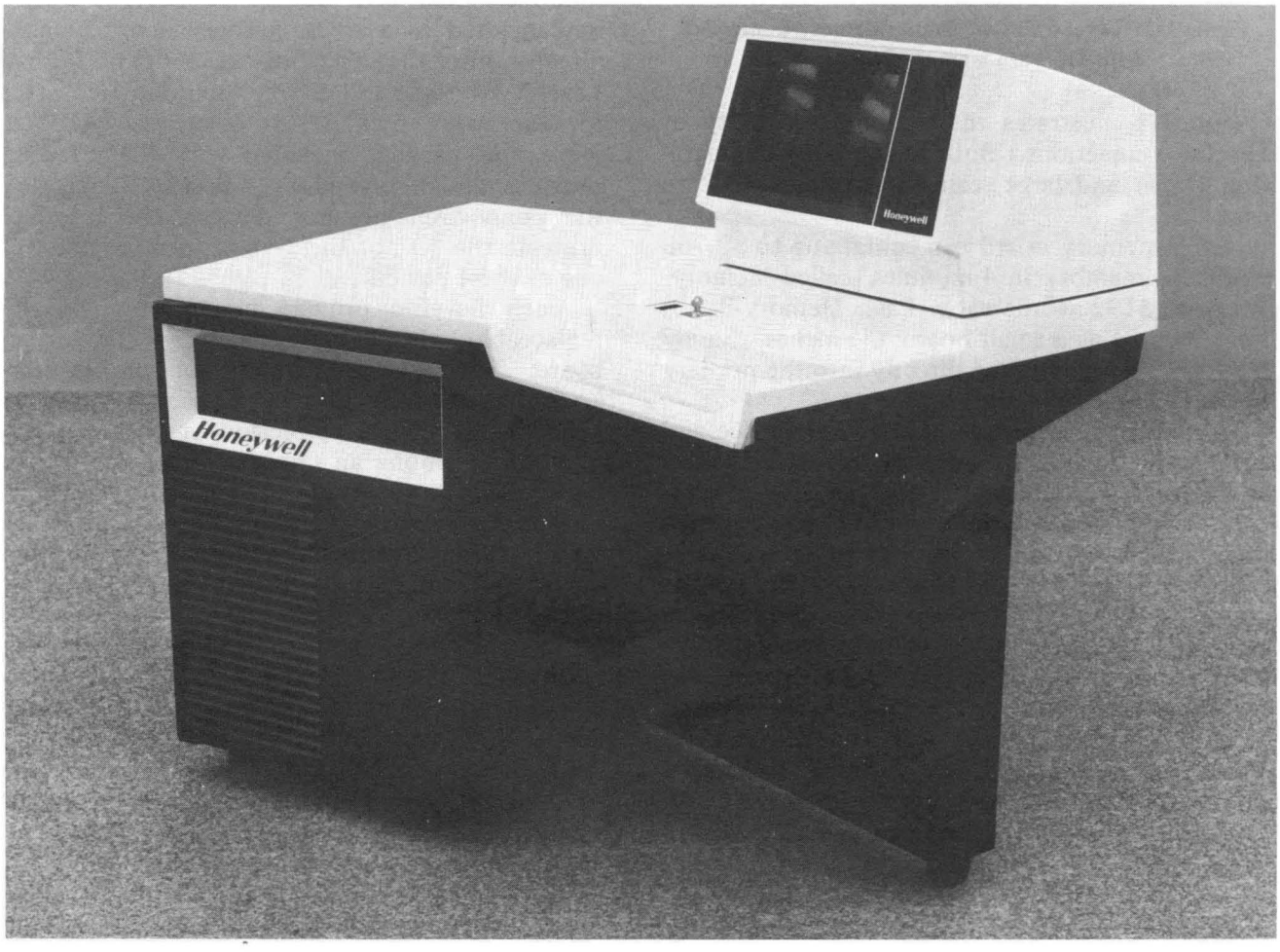


Figure 1-2. Office Packaging

BOARD TECHNOLOGY

Level 6 uses a primary circuit board 15 inches wide by 16 inches deep (see Figure 1-3). The rear of the board is connected to the Megabus, while devices are connected to the front of the board.

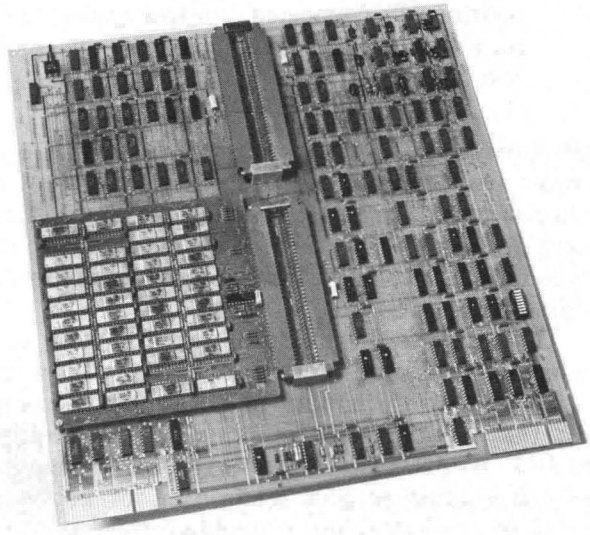


Figure 1-3. Level 6 Circuit Board (Memory Controller with 8K-Word Memory-Pac)

(Figure 1-4 illustrates the extreme simplicity of Level 6 connections.) Both medium scale integration (MSI) and large scale integration (LSI) are used.

Each memory board can contain up to 32,768 words of memory in 4 modules (called Memory-Pacs) of 8192 words each. Each Memory-Pac is implemented as a small board, 6½ inches wide by 6½ inches deep, plugged directly into the primary board.

A similar arrangement is used for device and communication controllers (see Figure 1-5). The Multiple Device Controller (MDC9101) consists of a primary board (containing a microprocessor) onto which up to four device adapter boards (3½ by 11 inches), called Device-Pacs, can be plugged. Each Device-Pac can support an ASR/KSR teleprinter, a teleprinter-compatible CRT display, a keyboard typewriter console, card reader, serial printer, or line printer. The diskette adapter supports either one or two diskette drives. A separate Mass Storage Controller (MSC9101) is used for Level 6 cartridge disk units, up to four of which can be attached to a single MSC. In its control concepts and interface with the Megabus, this controller is similar to the MDC.[2]

[2] See Section 6 for a description of all peripheral devices.

The Magnetic Tape Controller (MTC9101) is a 15- x 16-inch printed circuit board that plugs into a slot on the Level 6 Megabus. The MTC board can accommodate a double-sized magnetic tape Device-Pac and up to two standard printers and/or card reader Device-Pacs. The MTM9101 magnetic tape Device-Pac allows for 7-track NRZI magnetic tape drives while the MTM9102 is the Device-Pac used for 9-track NRZI magnetic tape drives.

The first tape drive in a system attaches via its device-cable directly to the Device-Pac. Up to three additional units can be serially connected. The unit record Device-Pacs are limited to unidirectional models which include the CRM9101 Device-Pac for card readers and the PRM9101 Device-Pac for printers.

The Multiline Communications Processor (MLC9103) consists of a primary board (also containing a microprocessor) with up to four line adapter boards, called Communication-Pacs. Each Communication-Pac handles one or two full-duplex lines.

The General-Purpose Direct Memory Access (DMA) Interface (GIS9001), for users who wish to attach their own devices to Level 6, is also implemented as a single primary board plugged directly into the Megabus. It contains both a Level 6 interface and a large "unpopulated" area for the user's own design interface. (A documentation package included with the interface provides design procedures.) Besides simplifying the connection of user devices, the interface protects the rest of the system from a failure of one of these devices.

Each Level 6 primary board is individually replaceable without disturbance of any other board. Furthermore, the use of the primary board/Pac concept permits the sharing of costly logic (memory error correction, controller microprocessors) among an unusually high number of basic elements (memory, devices) with very flexible configurability. Thus, the Level 6 approach satisfies users' demands for economy in four critical areas: space, power, cabling, configurability.

In addition, board addresses are easy to configure. There are 2^{10} possible addresses. Users can configure an address by setting a hexadecimal rotary switch (establishing an address for each board). This feature is important to all users, and especially to system builders planning to develop software on a large configuration and then use it on many smaller, differently configured systems. Level 6 also includes a device-identification feature, independent of software, so users will not have to keep changing their software when they change an address.

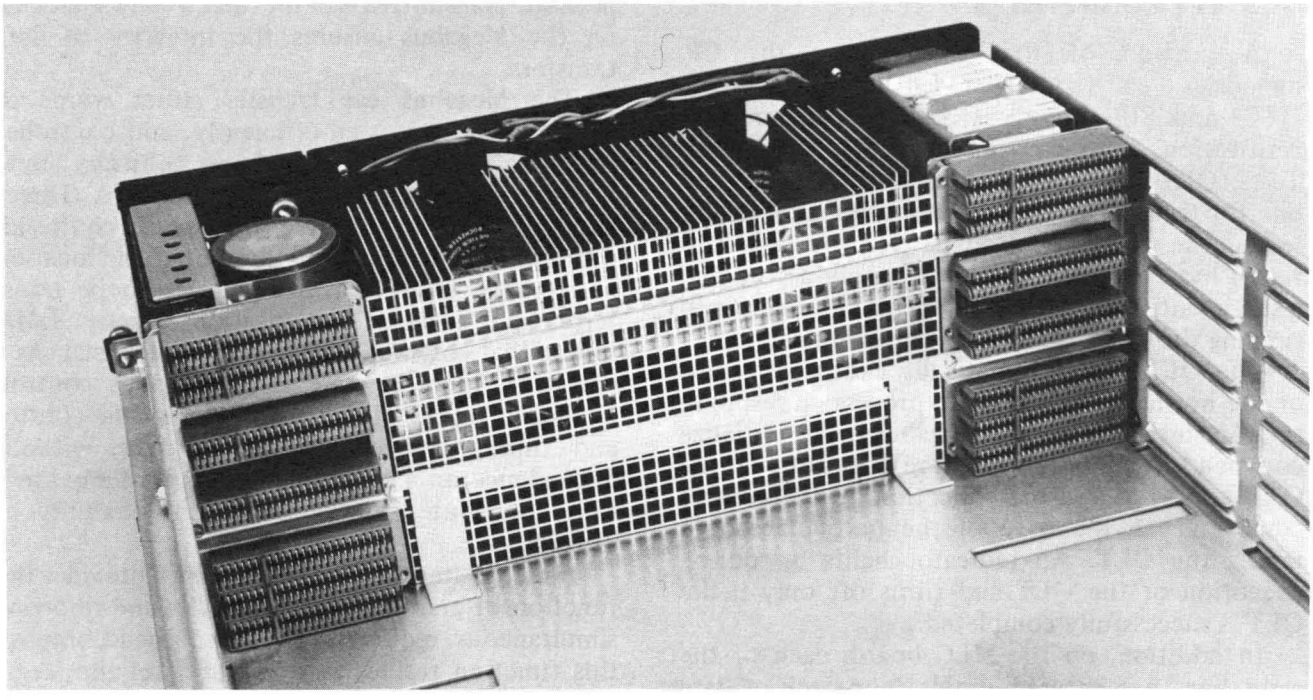
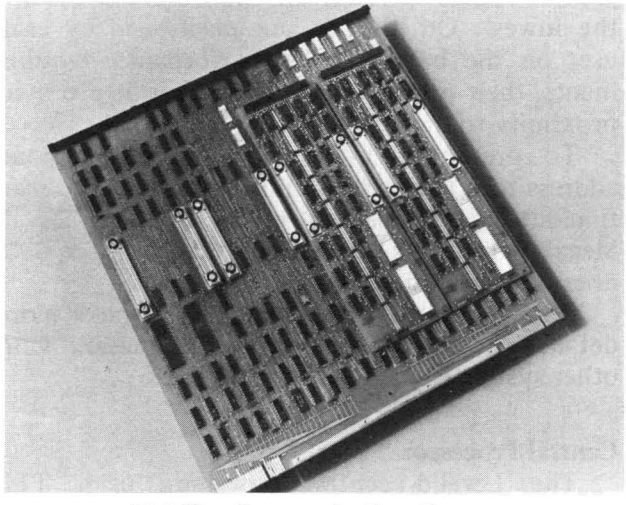
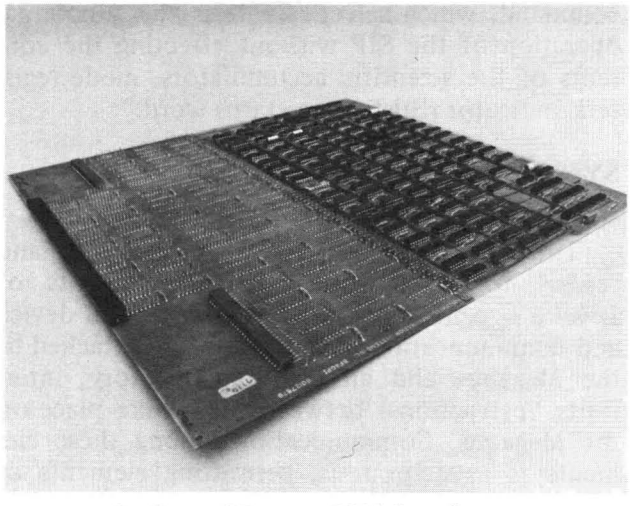


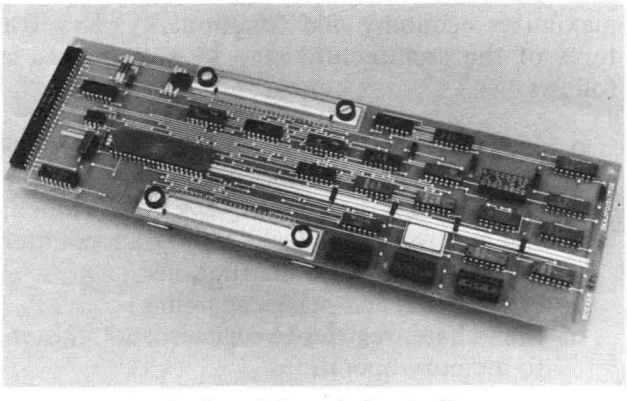
Figure 1-4. Level 6 Megabus and Power Supply



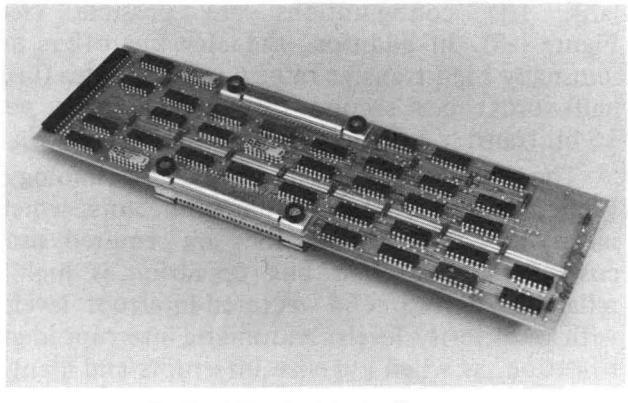
a. Multiline Communications Processor



b. General-Purpose DMA Interface



c. Keyboard-Console Device-Pac



d. Card Reader Device-Pac

Figure 1-5. Typical Controller Boards and Device-Pacs

QUALITY LOGIC TESTS

A portion of the firmware on the CP, memories, and controller (MDC, MLC, MSC, MTC, and SIP) boards is reserved for hardware verification routines called the Quality Logic Tests (QLTs). The QLTs consist of two routines: one for testing the CP and memories; another for testing the various controllers. Their purpose is to verify basic data paths and to supply a go/no-go visual identification of a hardware failure prior to running the test and verification programs.

The QLTs are automatically executed as part of the normal bootstrap load process, in response to a Master Clear on the Megabus or an initialize command (issued by software). Results of the QLT appear as a visual indication on the control panel and on the edge of the particular board failing the QLT. An indicator lights up during execution of the QLT and turns off only if the QLT is successfully completed.

In addition, on the MLC board, each of the eight lines has program-usable loop-back of data transmit to data receive to facilitate diagnosis. The SIP can execute a mini-QLT via a special trap command, which selectively tests the functional operation of the SIP without affecting the contents of the scientific accumulators, mode registers, indicator registers, or status word.

SYSTEM ELEMENTS

Megabus

The Megabus is the heart of Level 6, and central to its performance. All elements of Level 6 – central processor, memory, and device and communication controllers – are attached to the Megabus and all transfers (memory, interrupts, instructions) between them take place on the Megabus. Communication among these elements is asynchronous, permitting elements of different speeds to operate efficiently.

With up to 1024 I/O addresses available, very large I/O configurations are possible (see Figure 1-6). In addition, the Megabus offers an unusually high transfer rate: 6 million bytes (i.e., half-words) per second (300 nanoseconds per 16-bit transfer cycle).

The Megabus is based on TTL technology. Etched wires are used to join connectors, which means that fewer connections are required, user costs are lower, and bus operation is highly reliable. There are 64 vectored-interrupt levels, with 64 priority levels. Automatic interrupt identification, as when a device interrupts and identifies itself to the central processor, is provided. No private wires for interrupts are needed; an interrupt is handled as just one of several types of message transmitted on the bus. Parity checking on the Megabus ensures the integrity of data transfers.

The Megabus can transfer either words or bytes. Memory is used efficiently, and controller transfers can start or end on arbitrary byte boundaries. All transfers are of the DMA (Direct Memory Access) type; each device controller maintains its own information about the location in memory to/from which data is to be transferred and accesses that location directly. DMA means that software involvement is minimal. As a corollary, there is distributed Megabus control. Each unit on the Megabus contains all the control and timing it needs to use the bus, without dependence on a central control unit of any kind. This contributes to the great configurability of Level 6.

A distributed tiebreaking network provides the function of granting Megabus cycles and resolving simultaneous requests. The logic to accomplish this function resides in every unit on the Megabus. Priority is granted on the basis of physical position. In any Level 6 system, memory is granted highest priority and the central processor the lowest. Other units are positioned by each user on the basis of their performance requirements, their priority increasing according to their proximity to memory.

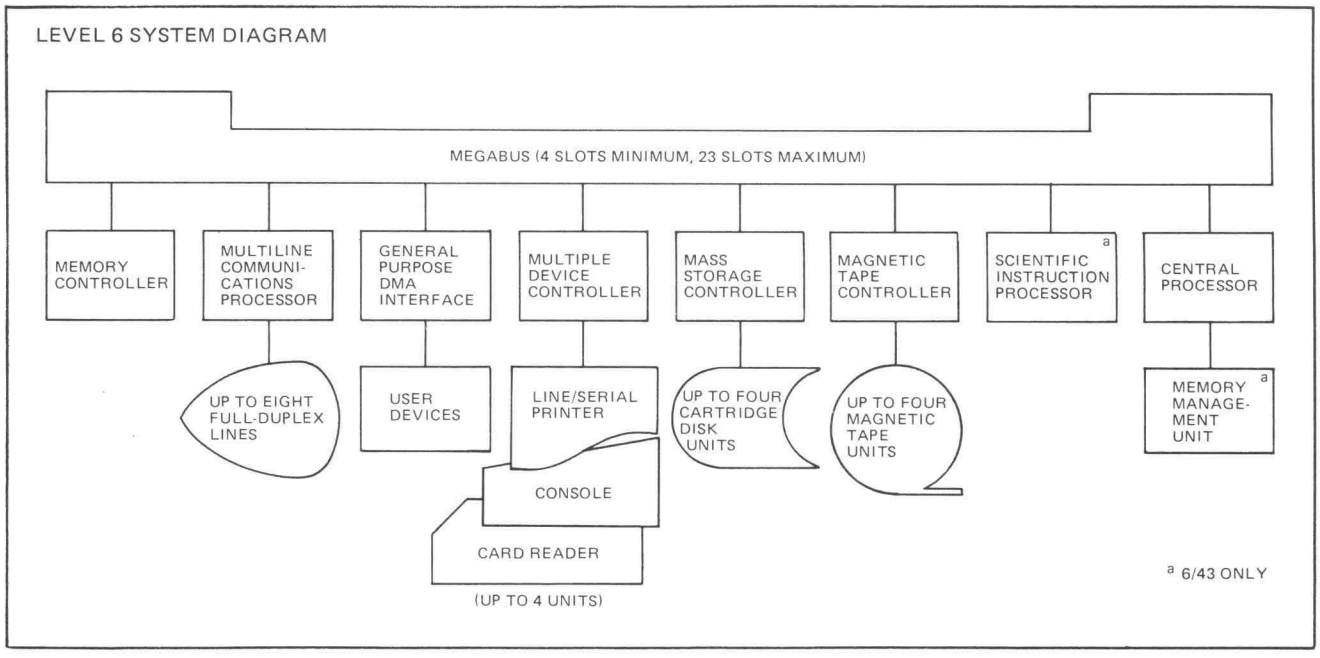
To prevent Megabus hangups, including an address to a nonexistent unit, there is a dead-man timeout by the central processor. It times all Megabus transfers and terminates them if a slave does not respond within 5 microseconds.

For an overview of system logic and a more detailed account of Megabus interaction with other system elements, see Section 2.

Central Processor

The Level 6 central processor, using TTL technology, is an open-ended, powerful, and flexible device. The word size is 16 bits – which maximizes economy and functionality. Key features of the architecture may be summarized as follows:

- 18 (6/34 and 6/36)/26 (6/43) program-visible general registers, including multiple accumulators and multiple address, index, and control registers and a program counter
- Bit, byte, and word instructions
- Bit test, set, and mask capability
- Immediate, register-to-register, and register-to-memory operations
- 64 vectored interrupt levels
- Multiple vectored trap structure
- Hardware-supported context save/restore



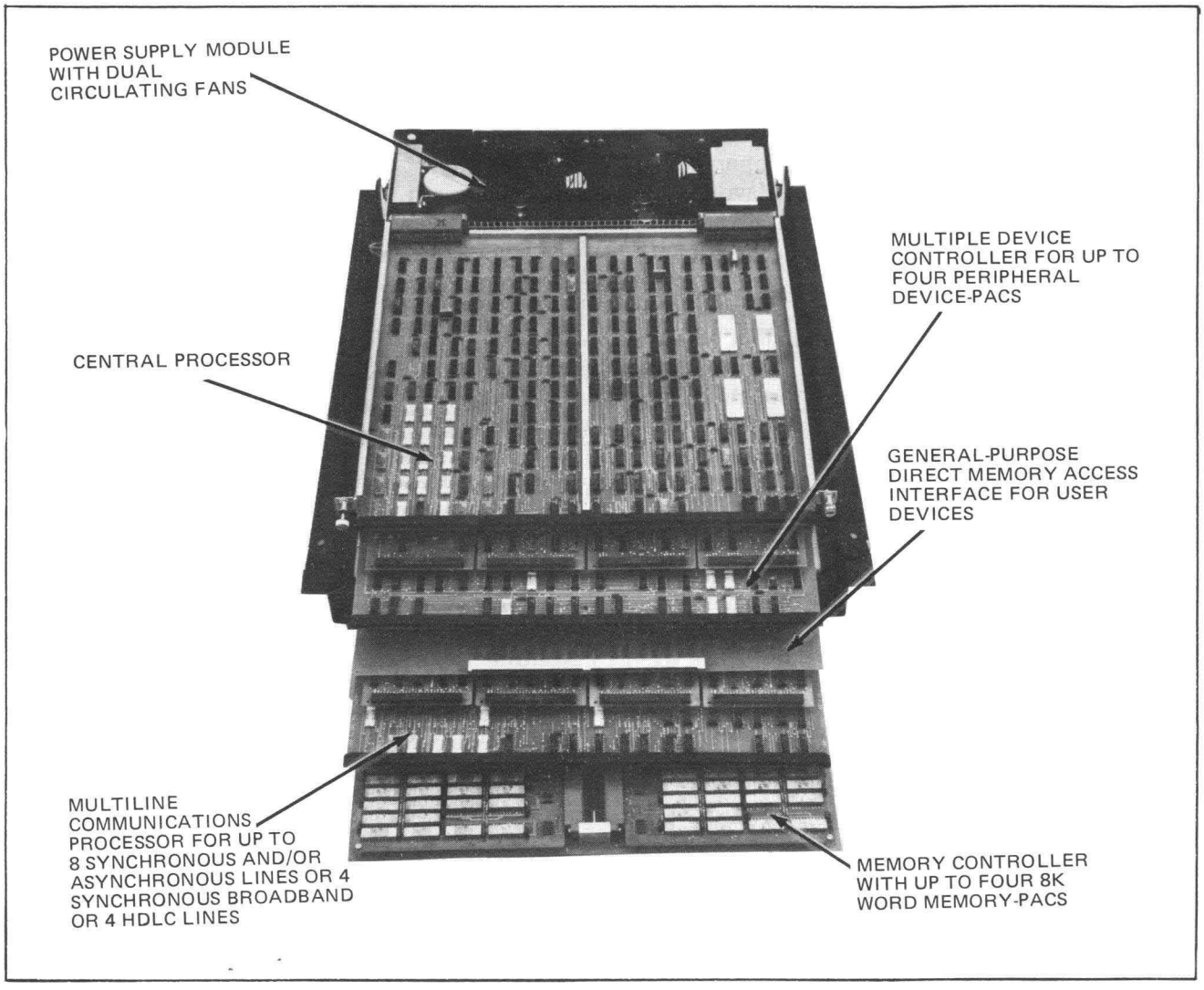


Figure 1-6. Megabus Configuration

- o Multiple addressing modes, including indexing, indirect addressing, base plus displacement, program-counter-relative, auto increment/decrement
- o Permanent bootstrap
- o Power failure detection
- o Real-time clock
- o Watchdog timer (optional on 6/36; standard on 6/43)
- o Automatic restart
- o High configurability
- o Stack/queue (standard on 6/43)

These features are described in detail in Sections 3 and 4. In addition, the central processor control panel is described in Section 5.

The instruction set is programmer-oriented, facilitating the writing of compact, efficient programs and offering the right instruction for each function. The multiple word length capability makes it easy to handle data elements of varying size, while the indexing techniques are completely integrated into the architecture.

The instruction set design strikes a highly effective compromise between two groups of conflicting user requirements: on the one hand the desire for more operation codes, operands, and flexible ways of addressing an operand; on the other hand, the reluctance to devote extra space in costly memory to longer and more exotic instructions, many of which are infrequently used.

The Level 6 approach defines a 16-bit instruction format that is extended in a number of ways. For example, a 16-bit instruction does not contain a direct memory address (though it does allow the programmer to address memory). For a direct address, a field is added to the instruction; this may be a 16-bit direct address. Thus, a 32-bit instruction is achieved as needed. Another method is to use a 16-bit displacement relative to the program counter. The result of this approach is to realize added power only when it is required – without compromising the design of the 16-bit instruction in which most of each program is actually written.

A bootload capability is included on the CP board. It is a ROM stored process and therefore permanent. There is no elaborate load procedure or multiple keying. Users can select any device through the control panel and bootload from there, providing only the device address (if the bootload is not from the default bootload device). Other outstanding characteristics of the central processor are as follows:

The maximum Level 6 memory (2,097,152 bytes on the 6/43) can be *addressed directly* (i.e., without an "indirect address" elsewhere in memory); the complexities and overhead of paging/segmentation units are avoided.

The Level 6 indexing technique permits *addressing and manipulating bits and bytes* in memory. Bit instructions allow testing, setting, resetting, and complementing of addressed bits. Byte instructions allow logical and arithmetic operations. Both facilitate the writing of compact, efficient programs.

The 18 *programmable registers* for the 6/30 models include 7 address registers, 7 data registers, 3 index registers (using data registers), 3 control registers, and a program counter, and for the 6/43 include the above, plus 6 additional control registers, 1 stack address register, and 1 register reserved for future use. This is a large complement for a minicomputer. Programs can be more compact and execution times faster than on machines with fewer registers.

The Level 6 *interrupt structure,* with its associated firmware-level dispatching and automatic context save/restore (see "Executive Modules" in Section 7), efficiently maintains a 64-level priority system while eliminating the need for complex software to perform this function.

Hardware trap support of up to 46 trap conditions allows quick handling of exceptional conditions (e.g., overflow) and permits a source program to run even if a facility it requires is not installed. (See "Trap Manager" under "Executive Modules" in Section 7.)

Level 6 includes instructions that facilitate *stack and queue manipulation,* including subroutine calls, reentrant and recursive programming, and data buffer management.

The various central processor options are listed in Table 1-2.

TABLE 1-2. CENTRAL PROCESSOR OPTIONS

MODEL/OPTION NO.			
6/34	**6/36**	**6/43**	**Description**
PSS9001	PSS9001	PSS9001	Tabletop Memory Save and Auto-restart for up to 64K words of Memory
PSS9002	PSS9002	PSS9002	Rack-Mountable Memory Save and Autorestart for up to 64K words of Memory
CMC9001	CMC9001	CMC9001	Single-Fetch Parity Memory Controller with 8K-word Memory-Pac
CMM9001	CMM9001	CMM9001	8K-word Parity Memory-Pac for CMC9001
–	CMC9002	CMC9002	Single-Fetch EDAC Memory Controller with 8K-word Memory-Pac
–	CMM9002	CMM9002	8K-word EDAC Memory-Pac for CMC9002
CMC9007[a]	CMC9005	CMC9005	Core Memory Controller with 16K words of Parity Core Memory
CMC9008[a]	CMC9006	CMC9006	Core Memory Controller with 32K words of Parity Core Memory
–	–	CMC9501	Double-Fetch Parity Memory Controller with 16K-word Memory-Pac
–	–	CMM9501	16K-word Parity Memory-Pac for CMC9501
–	–	CMC9502	Double-Fetch EDAC Memory Controller with 16K-word Memory-Pac
–	–	CMM9502	16K-word EDAC Memory-Pac for CMC9502

TABLE 1-2 (CONT). CENTRAL PROCESSOR OPTIONS

MODEL/OPTION NO.			
6/34	6/36	6/43	Description
CPF9408	CPF9408	CPF9504	Portable Plug-in Panel
CPF9407	CPF9407	CPF9407	Vertical Panel Mounting Option
–	–	CPF9503	Scientific Instruction Processor
–	–	CPF9501	Memory Management Unit

[a]In lieu of 8KW MOS.

Memory

Level 6 memory offers the reliability, low cost, and high density of MOS components. Battery back-up in the event of power failure is also available as an option (Memory Save and Autorestart).

Level 6 memory is a semiconductor type using N-channel 4096-bit dynamic RAMs as the storage media. Its design emphasizes high reliability, low cost, modularity, and simplified field maintenance. Like other elements of Level 6, the memory is a Megabus-compatible subsystem which may communicate directly with, and in the same fashion as, any other element on the bus. TTL MSI circuitry is used liberally to minimize power and space requirements for the extensive functionality provided. Self-contained on each memory module is Megabus support, refresh, and initialization logic.

Byte-parity memory, offered on all models, includes logic for storing/retrieving two parity bits per word on each 8192-word board module. Thus, the actual word size in memory is 18 bits. The parity bits are returned to anyone reading the memory.

EDAC (Error Detection and Correction) memory, offered on the 6/36 and 6/43, includes logic for six parity bits per word on each 8192-word module. The supplied parity is used together with the data to develop and store an EDAC code. When it is read, memory attempts to correct any internally caused data error, reporting the results over two dedicated leads on the Megabus. Each lead sets a specific status bit depending on whether the error can be corrected or not. EDAC is particularly desirable for large systems, where extended reliability is required.

With either kind of memory, address parity accompanies the most significant eight bits on the address bus. When memory detects an error on these bits, it does not respond; the result is a bus timeout. Note also that each device/communication controller on the Megabus checks parity as it passes through and indicates an error by setting a parity error status bit.

For both EDAC and parity memory, users may choose to use a memory controller with either standard single-fetch or a double-fetch memory technique.Double-fetch memory is available only in 16K increments.

Memory Save and Autorestart guarantees data retention for up to 131,072 bytes of memory for a two-hour period. Support circuit power runs are separated to minimize standby power drain. Electronics within the optional unit maintain battery charge, regulate outputs, and indicate holdup failures. It is available in either tabletop (PSS9001) or standard rackmountable (PSS9002) versions.

Core memory is available as an option on all 6/30 and 6/40 models when a nonvolatile memory is required. It is offered in either 16K- or 32K-word modules and can be intermixed with the standard MOS memory. Core memory systems can preserve their contents for an indefinite length of time during a power failure and automatically restart when power is resumed.

Core memory is packaged on a single, 15- x 16-inch (38- x 41-cm) Core Memory Controller board that requires a single slot on the Level 6 Megabus. All words are 18 bits in length, consisting of 16 data bits and two parity bits. The memory cycle time is 1.2 μs per word.

When intermixed with MOS memory, the core memory should be configured at the low end of memory to preserve the restart location. Core memory can also be intermixed with interleaved MOS memory on a 6/43, thereby offering users maximum throughput as well as the retention features of a nonvolatile memory.

To facilitate multi-user systems, the 6/43 central processor accommodates an optional Memory Management Unit which permits the allocation and assignment of memory among users and provides for segmentation and Read/Write/Execute protection based on four rings of privilege. The unit also provides base relocation and descriptor validation.

Semiconductor Storage

Commercially available 4KN MOS devices are used as the storage elements. Each device is exposed to vigorous functional testing on computerized test equipment. The result is a cost-effective quality product comparing favorably with core technology in all respects. TTL-compatibility offered by N-channel MOS eliminates the need for and liability of the extensive discrete and linear IC circuitry required to interface with core memories. The packing density is superior and power requirements are dramatically reduced: both characteristics again enhance the Level 6 size, weight, and MTBF advantages over core. Furthermore, a straightforward integrated circuit approach simplifies failure diagnosis and repair procedures as compared with core technology. One other significant advantage over core is the increase in memory utilization made possible with the inherently shorter cycle times associated with semiconductor devices.

Packaging

The advantages of Level 6 board technology are significant. Field expansion in small capacity increments is now possible without the cost of repeating overhead circuitry. Sparing costs are minimized through the use of the 8192-word modules as ORUs.[3]

Functionality

As in the case of other Level 6 system elements, the memory functions with minimum Megabus utilization. The interface is totally asynchronous, and after a memory read request is acknowledged the bus is released to other users. Meanwhile the memory completes its internal cycle while retaining the identification of the requestor. As the internal cycle nears completion, the memory requests a Megabus cycle and upon acknowledgment of bus availability sends data to the requestor. Megabus usage is less than half the memory cycle time and so allows simultaneous utilization of two modules at their maximum cycle rates.

Level 6 addressability extends to 16 million bytes; future expansion of the maximum memory now available on Level 6 is foreseeable. Configuration switches mounted on the memory are set with the module identification address to establish each module's position with the address spectrum. (The module's logical position, i.e., address, has no relationship to its physical chassis position.)

Write operations require only one Megabus cycle. Both full and byte write modes are available. Byte writes require a longer internal cycle for EDAC versions,[4] since check bits are now a function of both new and stored data. In EDAC parity checking, "Red" and "Yellow" bus signals are activated for noncorrectable and correctable errors, respectively. Parity on new data is treated as a pseudo data bit by the check bit encoders. If bad parity is received, the "Red" line is activated during the subsequent readout.

A momentary contact switch is placed on the EDAC memory board within easy access behind the swing-out control panel. When the switch is pressed during the specified portion of the diagnostic routine, check bits are forced to all zeros for all new data. With careful selection of data patterns, all elements of the decoding, detection, and correcting logic can be exercised and must be operating properly for the scrambled readout data to match that of a functioning system. Other portions of the EDAC circuitry, including check bit chips, can be checked with special patterns in normal mode.

The sharing of the I/O bus with memory in Level 6 (as opposed to the use of a second bus dedicated to memory) offers many advantages including:

1. A fully asynchronous memory interface is achieved that is fully compatible with all other elements of Level 6. Thus direct memory access is a fundamental ingredient of the architecture. Furthermore, the standard asynchronous interface expands configuration dimensionality by allowing simultaneous use of memory types with varying performance and cost.
2. Bus hardware is minimized, with less than 100 signal paths required for the total function. This feature helps make possible the unique physical configuration of Level 6, providing front access board removal in conjunction with the simplicity of front-to-rear cooling airflow.

[3] Optimum Replacement Units. See "Maintenance" for an explanation of the ORU.
[4] Read operations are also somewhat longer with EDAC versions of memory.

Multiple Device Controller (MDC9101)

Each MDC supports up to four of the following devices, via individual Device-Pacs: (1) a standard Model 33 ASR/KSR teleprinter; (2) a 9600 baud teleprinter-compatible CRT display for console use; (3) a 30- or 120-character-per-second keyboard typewriter console; (4) a 300- or 500-card-per-minute (cpm) card reader; (5) a 60-line-per-minute (lpm) serial printer; (6) a 240/300/440/600 lpm line printer. The MDC also supports one or two diskette units. Through the MDC, all these devices can operate simultaneously.

Multiple MDCs can be attached to the Megabus, and multiple-like devices can be attached to any MDC. Each MDC contains a firmware-driven microprocessor augmented with the circuitry necessary to support the adapter connections. All hardware unique to a specific kind of device is contained in the Device-Pac (which makes it easy for users to change their peripheral configurations). Through its Pac each attached device presents a DMA (Direct Memory Access) interface to the central processor (Figure 1-7). The design of this interface is such that software commands and status pertain to the device itself rather than to the MDC.

Mass Storage Controller (MSC9101)

The MSC provides microprogrammed support of up to four cartridge Disk Units with a total capacity of 44.8 million bytes. General control functions include execution of Megabus sequences, command decoding, data transfer multiplexing between Device-Pacs, and status and control register storage.

The hardware unique to the disk units is localized on a Cartridge Disk Device-Pac similar in concept and performance to the Device-Pacs plugged into the MDC. The Device-Pac performs such functions as device interface dialog control, sync word detection, check word generation/verification, device state monitoring, and bit-to-bit conversion for the set of disk units (1 to 4) attached to the MSC.

Maximum throughput across the MSC/Pac interface is 312,500 bytes per second. Multiple MSCs can be attached to the Megabus. The MSC permits a data transfer operation to take place concurrently with multiple seek operations.

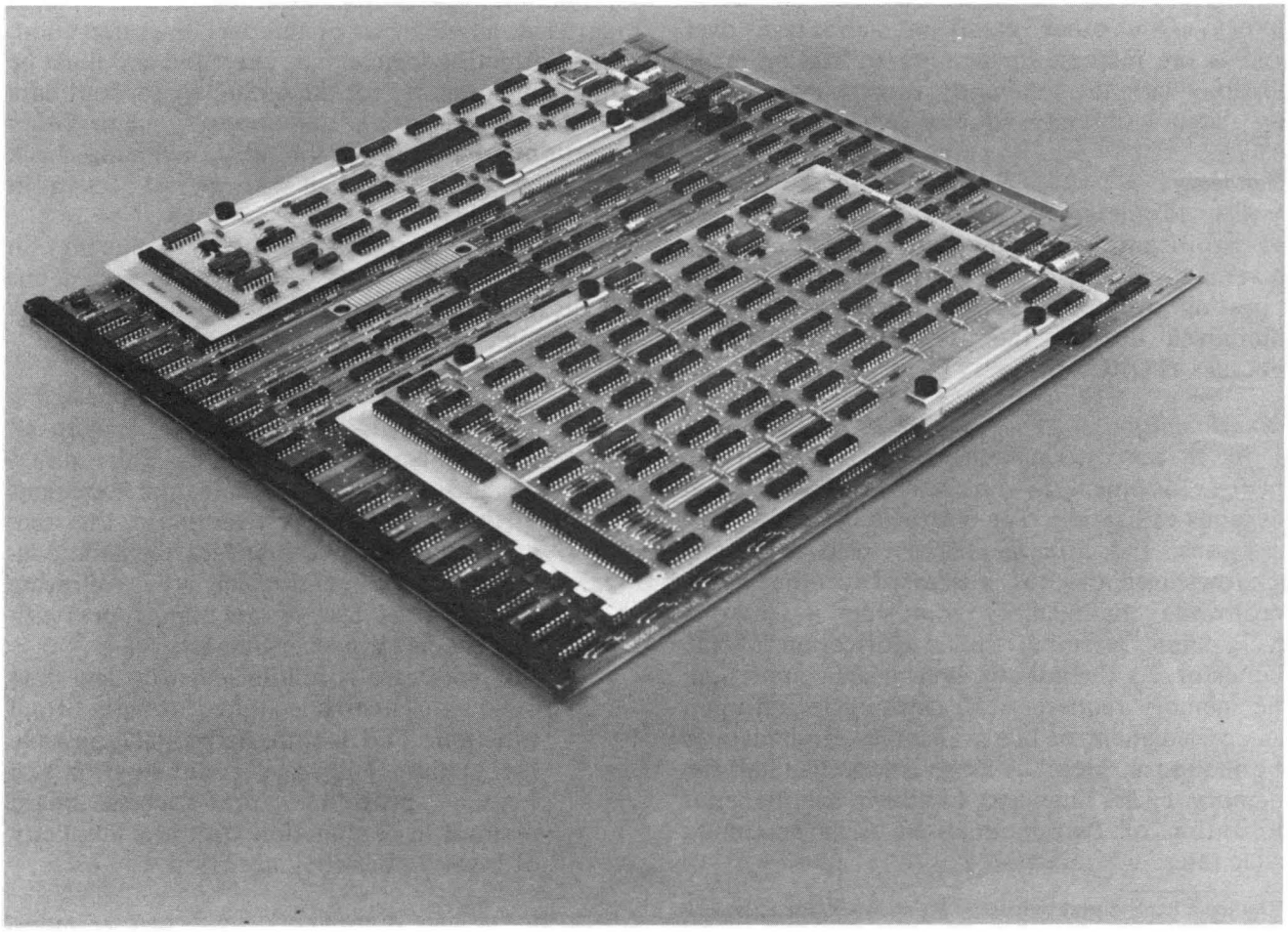


Figure 1-7. MDC with Device-Pacs Attached

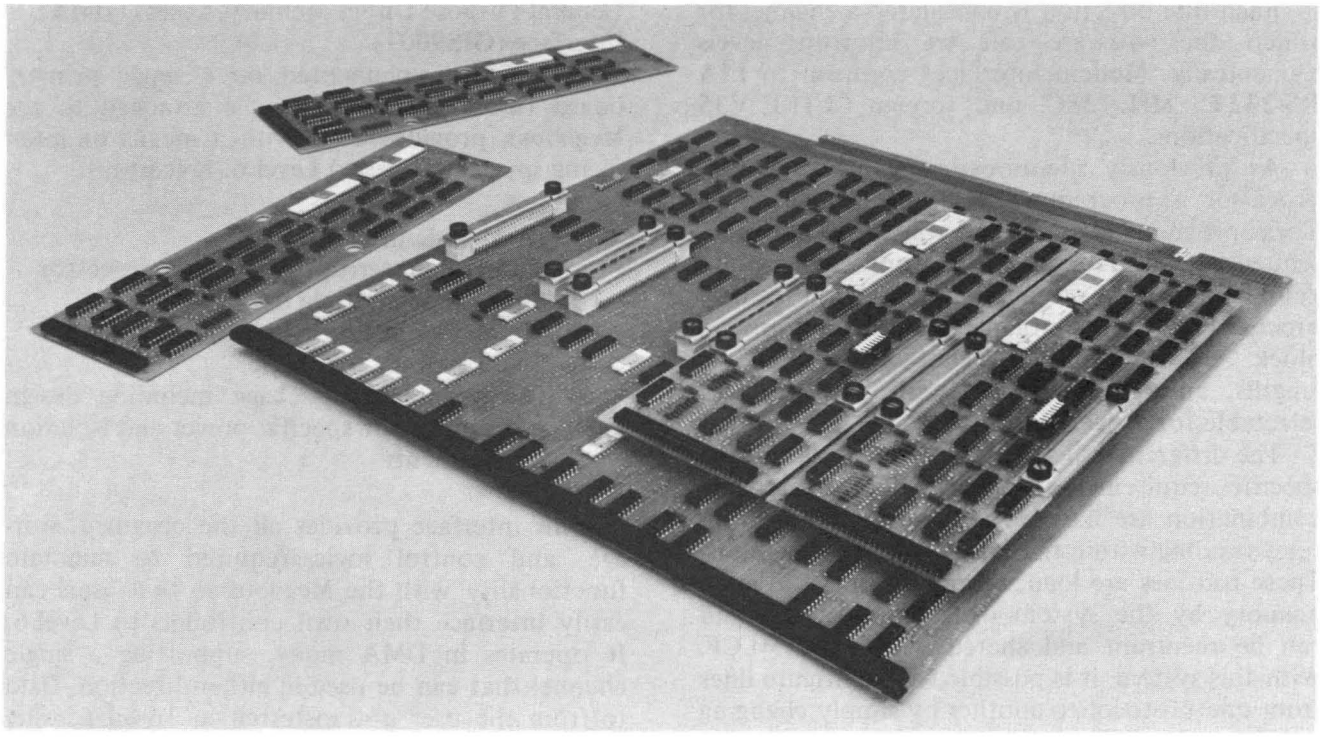


Figure 1-8. MLCP with Communications-Pacs Attached

Multiline Communications Processor (MLC9103)

The microprocessor-based MLCP provides an extremely powerful communications link at moderate cost. Each MLCP can contain up to four Communication-Pacs, and each Pac can handle one or two lines (Figure 1-8). Throughput is up to approximately 16,000 characters per second. The MLCP is unusually flexible regarding line types, speeds, and communications disciplines. For example, it can handle up to eight, full-duplex, low-speed (300 bps or less) and medium-speed (600 to 19,200 bps) lines in one board slot. Alternatively, four broadband or HDLC lines can be attached; within the overall throughput limit, each such line may operate at speeds of up to 72,000 bps. Its 4096 bytes of Random Access Memory (RAM) – 3072 used for its procedure code – permit it to execute complex line-handling procedures with no involvement of the central processor. Key attributes of the MLCP are as follows:

- o Implementation of MLCP on a single primary board with field-configurable, easily changed/replaced Communication-Pacs
- o Operation of up to 8 full-duplex, 7,200 bps lines or one 72,000-bit line per MLCP
- o Program load of configuration and control
- o Program load of data stream control
- o User-programmable to provide for message delimiting, message editing, checking algorithms
- o Hardware checking (e.g., Longitudinal Redundancy Check – LRC)
- o Individual DMA for each line and transmission direction
- o Standard Megabus interface and the following:
  - *Synchronous Communication-Pacs:* one or two full-duplex lines; speeds of up to 19,200 bps; EIA RS-232-C interface.
  - *Asynchronous Communication-Pacs:* one or two full-duplex lines; speeds of up to 19,200 bps; EIA RS-232-C interface, one or two stop bits; selection of speeds by parameter.
  - *Synchronous Broadband Communication-Pac:* one full-duplex line; speeds up to 72,000 bps; broadband CCITT-V35 or Bell 301/303 interface.
  - *Synchronous HDLC Communication-Pac:* one line; speeds up to 19,200 bps.
  - *Synchronous Communications-Pac:* one line; speeds up to 10,800 bps; MIL 188C interface.
  - *Auto Call Communications-Pacs:* one or two synchronous/asynchronous lines.
  - *Current Loop Connection Communications-Pacs:* one or two lines; speeds up to 9600 bps.

Each line direction is considered a channel for which the software can set interrupt levels dynamically. Modem interfaces conform to EIA RS-232-C, MIL 188C, and foreign CCITT V35 specifications.

As previously mentioned, the MLCP microprocessor is programmable at the channel level. These programs can edit and convert prespecified sequences of data. The microprocessor fully delimits the data stream, offloading the central processor of character generation, detection, and block check functions. Data formats, code lengths, and parity characteristics are program selectable for a given line.

The differences between line protocols and the specific requirements of each line adapter/line combination are handled via small software routines residing within the MLCP on a per-line basis. These routines are loaded from an image in main memory by the system operating software and can be reentrant and shared within the MLCP. With this system, it is possible to reconfigure lines from one protocol to another by simply changing a parameter or reloading a line-specific routine.

Magnetic Tape Controller (MTC9101)

The MTC offers Level 6 users the capability of attaching magnetic tape units as well as unit record devices to their system with a single controller. Incorporating sophisticated hardware and firmware, the MTC microprogrammed control supports the connection of one of the following configurations:

(1) up to four 7-track magnetic tape units (MTU9112/9113)
(2) up to four 9-track magnetic tape units (MTU9104/9105)
(3) up to two magnetic tape units, and up to two (same or combination) unit record devices: card readers, serial printers, line printers.

7-track and 9-track magnetic tape units may not be mixed on the same MTC; however, they may be mixed on the same Level 6 system if a second MTC is configured. Multiple MTCs can also be configured if users require extra functionality such as read-write simultaneity, additional tape drives, and additional unit record devices. Tape speeds of 45 ips and 75 ips may be mixed on the same MTC.

The magnetic tape units interface with the MTC via a single NRZI Magnetic Tape Device-Pac which can connect up to four tape units. The unit record devices require their own Device-Pacs.

General-Purpose Direct Memory Access (DMA) Interface (GIS9001)

The GPI, implemented on a single primary board (more than one can be attached to the Megabus), provides users with a means of interfacing special devices to Level 6. It features:

- o Word-wide interface
- o Ability to interrupt the central processor
- o Support of multiple devices
- o Simple control-line interface
- o Spare area for user logic
- o Documentation package including design procedures for specific power and isolation requirements

This interface provides all the circuitry, storage, and control logic required to maintain functionality with the Megabus so that users can easily interface their own controllers to Level 6. It operates in DMA mode, supporting a single channel that can be used in either direction. Data to/from the user is transferred in 16-bit parallel form, controlled by a simple, asynchronous handshake operation. Data transfers between the interface and the Megabus are also in 16-bit parallel form, via DMA control. The transfer rate is 0.5 million bytes/second between memory and user device.

A scratchpad memory (16-bit x 16-word) is integral to the GPI. One cell (16-bit word) is dedicated as a temporary data buffer to isolate the user from the bus. Other cells are used for essential system quantities (e.g., Status, Task Word, Interrupt Control). Nine cells are provided for user application as working registers or as communication locations between the user-written driver and the user's hardware.

Board maintainability is enhanced by the inclusion of loop-back logic. This feature, along with the standard test and verification program supplied, allows a user to exercise data transfers under DMA control, as well as scratchpad memory operation and task flag and interrupt functions.

The GPI board contains both the Honeywell-supplied logic and an area reserved for user logic (see Figure 1-9). The user area is configured with 66 sites for mounting 14- or 16-pin Dual In-line Packages (DIPs). Each DIP station includes wire-wrap pins for the user interconnections. The interface logic and user areas are separated by a row of 70 wirewrap pins; of these, 50 are used to connect the interface signals to/from the GPI logic (the rest are used for power and ground). There are two 56-pin connectors for attaching cables to user devices.

Scientific Instruction Processor (CPF9503)

The Scientific Instruction Processor (SIP) is an optional single-board processor available for 6/40 models. It operates on a powerful set of 30 scientific instructions including arithmetic operations on single-precision and double-precision floating-point operands and single-word and double-word integer operands.

The SIP contains three variable length scientific accumulators which may contain floating-point values of two or four words. Control bits define both the accumulator length and length of memory operands directed to that accumulator.

All operands stored in the SIP are in floating-point format. Operands directed to the SIP from main memory are also in floating-point format, while those from the CP are in integer format. Integer operations directed to the SIP are converted to floating-point values prior to entering into the scientific calculations.

Memory Management Unit (CPF9501)

The Memory Management Unit (MMU) is an option located on the central processor board of a 6/40 model. It provides advanced memory management and protection capabilities and is implemented as an integral part of the 6/40 processor, thus avoiding any performance degradation when memory management is used.

The MMU performs the following functions:

- o Separates memory into 16 or 31 independent segments
- o Relocates each segment independently in physical memory
- o Protects each segment from improper access, based on software-specified attributes

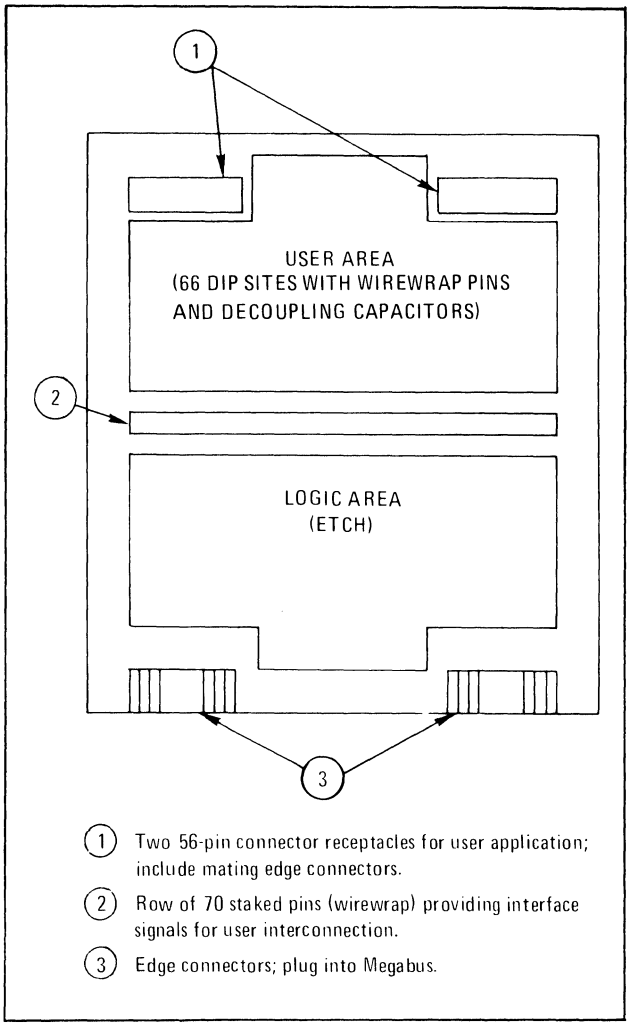


Figure 1-9. Layout of General-Purpose DMA Interface Board

POWER SUPPLY AND FANS

The Level 6 power supply and air-circulating fans are located at the rear of the drawer holding the central processor. Packed into a relatively small volume (about 15 in. x 6 in. x 5 in.) and requiring only a standard, 120-volt, single-phase, 3-wire supply, the power module provides users with unusually economical operation plus full brownout protection (ability to ride through a 3- to 4-cycle loss).

The front panel and chassis wraparound provide complete protection from hazardous voltages. In addition, the ac line input is interlocked so that removal of the power supply automatically removes ac power from the system.

Two fans at the rear of the power supply draw air from the front to the back of the drawer, past the circuit boards and the power module itself. They help keep the system operating through a temperature range of 0°C to 50°C ambient, and a humidity range of 5% to 95%.

PERIPHERAL DEVICES

Level 6 includes a set of economical peripheral devices, all attachable via the MDC, MSC, or MTC, featuring quiet operation suitable for an office-like environment, low power requirements, and sufficient speed for users' application. This section summarizes peripheral features. (For detailed information, see Section 6.)

Programmable Interrupt Levels

Level 6 gives any peripheral device the capability of interrupting at any level, under software control. Interrupt levels can be set by the software dynamically, before each job or before each peripheral instruction sequence, etc. These levels are independent of physical system configuration or device type. Users benefit from an increased flexibility in assigning job priorities.

Teleprinters and Compatible CRT Devices

Teletype Corporation standard Model 33 teleprinters are available in a Level 6 packaging design, in either ASR/KSR (TTU9101/9102) or ASR/KSR (TTU9103/9104) with autoshutdown.

The CRT keyboard consoles are low-cost, teleprinter-compatible devices for users who require a visual medium in addition to their hard-copy teleprinters and typewriter consoles. They feature a separable keyboard (facilitating its use and incorporation into office furniture packages), nondestructive cursor backspace and forwardspace, and a 960-character (12-line) display. A 64-character set (ASCII) is offered with the DKU9101; a 95-character set including lowercase letters is available with the DKU9102.

Keyboard Typewriter Consoles

The TWU9101 keyboard typewriter console prints 30 characters per second, has a line width of 132 print positions, and has a 64-character ASCII code set. The TWU9104/9106 keyboard typewriters have a print speed of 30/120 cps with a 96-character ASCII code set and 132 print positions.

Paper is tractor-fed and the fanfold paper forms provide lengths of 3 to 17 inches and widths of 4 to 15 inches. The console furnishes an original and four clear copies.

Card Readers

Level 6 card readers (CRU9101/9102/9103/9104/9108/9109/9110/9111/9112/9113) handle 80-column cards at 300 or 500 cpm. A Mark Sense capability which allows the reading of mixed punches and IBM-compatible marks is standard on some of the readers. A 51-column card reading capability is optionally available on other readers. The hopper and stacker capacities are 500 cards. The card readers are compact, self-contained, tabletop units.

Serial Printers

The serial printers (PRU9101/9102) furnish an original and up to four clear carbon copies at speeds of 165 characters per second (cps) with 64- and 96-ASCII code character sets, respectively. This speed is equivalent to 60 lines per minute at 132 characters per line.

There are 132 print positions at a horizontal spacing of 10 characters per inch. Characters are formed from a 9 x 7 dot matrix and are equivalent to 10-point type. Vertical spacing is 6 lines per inch and there is a capability for head-of-form, end-of-form, and single-line formats. A standard sprocket paper feed is used.

The self-contained, tabletop-sized serial printers may be made self-standing with a pedestal.

Line Printers

The line printers (PRU9103/9104/9105/9106) operate at 240–600 lpm, using the 64- or 96-character ASCII set. Characters are printed from a drum-mounted, solid font.

There are 136 print positions, and the paper has a slew rate of 20 inches per second. Six-part forms are produced. A Vertical Format Unit provides head-of-form and end-of-form capability in a 66-line format; a 12-channel paper tape option is available for the VFU. A floor-mounted cabinet is standard.

Diskettes

The Single/Dual Diskettes (DIU9101/9102) have a data rate of 31,248 bytes per second (MDC to memory) and a capacity of 256,256 bytes per disk (formatted, 77 tracks). The diskettes (single or dual) are offered in convenient rackmountable or freestanding tabletop versions.

The diskettes use standard (IBM "Type 8") flexible magnetic disks or their equivalent. Data is physically recorded on the disks in an IBM-compatible format, but the contents of the header records and the actual data are under software control and are not restricted to be IBM-compatible. The data representation is in ASCII.

Cartridge Disk Units

The CDU9101/9102/9103/9104/9114/9116 Cartridge Disk Units provide low-cost data storage for users with medium-sized file requirements. Capacity ranges from 2.5 to 11.2 million bytes per unit; random access to 11.2 million bytes requires an average of 47.5 milliseconds. Two record lengths are available: 12 or 24 per track. All records include check words for data integrity. The units are packaged for either rack or cabinet mounting; both versions feature easy top-loading of removable cartridges. And since removable and fixed disks can be mounted on the same spindle, file copying can be accomplished with a single unit.

Magnetic Tape Units

The MTU9104/9105/9112/9113 are compact, self-contained, rack-mountable tape units. The MTU9104/9105 are 9-track, NRZI, 800-bpi tape units with transport speeds of 45/75 ips and transfer rates of 36/60K bytes per second. The MTU9112/9113 are 7-track, NRZI, dual-density (556-/800-bpi) tape units with transport speeds of 45/75 ips and transfer rates of 25/36K characters per second and 41.7/60K characters per second. The units use a vacuum system for uniform tension and tape braking.

SOFTWARE

Level 6 offers two operating systems, Basic Executive System (BES) and Multi-Dimensional Tasking (MDT), both under the control of GCOS – the General Comprehensive Operating System.

GCOS/BES

BES is offered in a basic (BES1) and extended version (BES2). Under GCOS/BES1 the chief system functions are performed, allowing users to concentrate on their individual applications. For example, the executive modules provide a rapid response to time, priority, and I/O interrupts. Their size is minimal and their interface with user programs is straightforward and easily learned. Providing system control and a firm base on which to build applications, their features include: task management – starting, suspending, and resuming specific tasks on the basis of software priority levels; support of the central processor clock to provide both periodic and time-of-day execution; operator interface management; trap management; optional file and data management; and optional dynamic buffer allocation allowing a number of requesting tasks to share a common memory pool.

For program development, GCOS/BES1 includes both a powerful 8K Assembler and an industry-standard 16K FORTRAN compiler. Either language operates in a minimum Level 6 configuration with two diskettes but takes advantage of larger Level 6 systems. In two passes, the Assembler produces relocatable code for a program of up to 32K. The single-pass FORTRAN compiler allows assembly language statements to be embedded in a FORTRAN program.

To assist the program development effort, Level 6 includes a macro preprocessor, a linker that can process external references to build a 64K image file and a productivity-increasing text editor.

Still other software includes loaders that support relocation and accept files from console paper tape, disk, or diskette; online and stand-alone device drivers supporting terminals and peripheral devices, and a configuration load manager.

GCOS/BES2 includes all the capabilities of GCOS/BES1, with significant extensions in communications, executive functions, and program development tools. Among the new communications capabilities are support of BSC teleprinters and teleprinter-compatible CRTs. Added executive functions include a disk-based system loading/activating tasks resident on either diskette or cartridge disk. And the program development enhancements encompass extensions in both the Assembler and the FORTRAN compiler, plus a BASIC interpreter and a COBOL compiler.

GCOS 6/MDT

GCOS 6/MDT offers multidimensional tasking capabilities in a diskette-/disk-based operating system. This system provides executive, file management and communications facilities designed to support multitasking real-time or data communications applications in one or more online streams. Additionally, program development and other batch applications can be performed concurrently in a single batch stream.

The GCOS 6/MDT Executive supports the execution of user application tasks and provides a set of system services which allows users to control the execution of individual tasks and the synchronization of multiple tasks with each other and with time-related events. It controls the loading of user programs, manages requests for available memory, and provides both standard and user-oriented system trap handling routines.

The File Manager offers users an extensive set of logical I/O access methods; it provides device-independent access to any device for sequential files as well as direct and indexed access to direct access mass storage files. In addition, the File Manager automatically manages the space utilization of mounted mass storage volumes which allows users to easily create and expand mass storage files as their online application needs change and grow.

Two levels of communications interface are offered under GCOS 6/MDT. Remote and local terminals may be accessed through the sequential file interface of the File Manager. As an alternative, users can exercise more direct control over their communications environment by using the system's physical I/O interface.

An Operator Control Language processor allows the system operator to dynamically configure, load, and monitor application processes. Online applications may be directed by Operator Control Language commands or by Execution Control Language commands that overlap the Operator Control Language but are submitted from a different source, such as a disk file or remote terminal.

The Software Monitor allocates memory dynamically from pools and can relocate programs at load time. Once an application is loaded into memory, it is dispatched according to its assigned priority levels. When multiple tasks share

a priority level, they are serviced in a round-robin fashion.

A powerful and comprehensive set of Program Development components, utilities and debugging aids for user application development is included in GCOS 6/MDT. While multiple applications are executing in the real-time environment, Program Development can be accomplished in the batch stream. The Execution Control Language is used to control the Program Development process.

A Text Editor for program preparation, updating, and correction is provided as well as a powerful Assembly Language. A Macro Preprocessor complements the power of the Assembly Language. A FORTRAN Compiler, based on the proposed American National Standards Institute 1977 standard, provides for double precision data type, extended I/O edit capabilities and many other FORTRAN language features. GCOS 6/MDT also supports a COBOL Compiler that offers significant functionality and a Report Program Generator (RPG) Compiler that provides users with a versatile problem solving language.

Program Development facilities also include a Linker, which combines the object text output of the language processors producing load modules to facilitate rapid program loading and relocatable code capability.

A wide range of utilities is available including disk sort, debugging aids, program patch, copy/compare, print, dump/edit, file dump, data transcription, and file formatting. Mass Storage, magnetic tape, card and print devices are supported by these utilities.

For a detailed discussion of GCOS/BES and GCOS 6/MDT software, see Section 7.

HUMAN FACTORS ENGINEERING AND INDUSTRIAL DESIGN

Level 6 hardware elements meet OSHA[5] requirements for operator safety and have UL and CSA[6] approval. Safety-interlocked panels help eliminate operator hazards, work surfaces are made of durable materials and finishes, and casters and jacks ease installation.

There are many operator convenience features: "see only when necessary" status displays plus internal illumination to avoid confusion and reading errors; insulation to reduce noise; easy machine loading procedures; high-lubricity feed and stacker work surfaces for efficient document handling; simplified access to card paths to facilitate the removal of damaged cards.

[5] Occupational Safety Health Act

[6] Canadian Standards Association

In addition, system builders benefit from the great flexibility of Level 6 in installation and configuration, such as the need for less power and floor space than in many competitive systems, the easy accessibility of internal components and the overall design for simplicity of operation and maintenance, the greatly reduced requirements for operator training, and the feasibility of personalizing the equipment.

SITE PREPARATION PLANNING

Some users need very little site preparation assistance; others need detailed pre-installation planning. As a major mainframe producer Honeywell has had years of experience in installing computer systems and can provide assistance at whatever level is required. Both end users of Level 6 and system builders incorporating Level 6 components into their own product lines will find Honeywell's knowledge and expertise helpful, either in forestalling difficulties or in solving particular configuration problems.

Honeywell's assistance covers the range of physical, environmental, electrical, and safety requirements. For detailed planning information including configuration, layout, specification, and site facility data, see the *Level 6 Site Preparation* manual, Order No. AY52.

MAINTENANCE

The Level 6 design is based on consolidation of functional system modules on single boards. This not only reduces module interconnections to Megabus and power supply (eliminating intra-module connections), but also enhances system availability. First, the existence of functional boundaries simplifies the procedures required to diagnose faults. In turn, this means that execution of these procedures should require very little downtime (on the order of minutes), and that the attempt to isolate a fault to board level should normally be successful. Third, "repair" of the fault requires simply unplugging the failed unit and plugging in a replacement.

For maintenance purposes, all system modules that can be easily removed and replaced at the customer site are called Optimum Replacement Units, or ORUs. Either Honeywell field engineers or user personnel (nontechnical but trained) will replace an ORU, depending on the maintenance agreement in effect. ORUs include the following:

- Primary boards that connect into the Megabus and similar boards that form part of a

Level 6 component (e.g., Memory- or Device-Pacs).
- o A power supply, control panel, or peripheral device.
- o Fuses for power; air circulating fans.

The actual ORU isolation is carried out in two steps. First the resident hardware/firmware QLT performs a go/no-go test of data paths and basic functions. If required, software T & V tests are performed, completing the ORU isolation and causing the results to be displayed.

The display of results indicates: (1) the unit to be replaced, (2) the case of no fault detected, or (3) the case in which the fault cannot be resolved to an ORU. In the vast majority of cases the result will be isolation of the fault to an easily replaced ORU.

DOCUMENTATION

Level 6 documentation includes a full, integrated set of software, hardware, and other manuals. These are user-oriented, and written for easy operational and reference use. Every major element of Level 6 is documented. (See Table 1-3.)

FUTURE DEVELOPMENTS

Experienced minisystem users will have noticed that the open-ended design of Level 6 holds truly exciting possibilities for hardware or software extensions into many areas, with benefits to themselves in terms of greater functionality and bottom-line economies. Your Honeywell Marketing Representative will keep you abreast of these developments. In addition, Honeywell will continue to make largely invisible improvements in Level 6, whose architecture accommodates the rapidly developing semiconductor technologies. Level 6 is built to last: to provide what you need now and in the future, at a definitive cost/performance advantage over its competitors.

TABLE 1-3. LEVEL 6 DOCUMENTATION

Title	Order Number
PROMOTIONAL	
Level 6 General Brochure	AS57
40/50/60 Leasing Plan Brochure	AT14
Level 6 Spares/Repair Pocket Guide	AT96
Level 6 Spares/Repair Brochure	AU35
Models 6/34 and 6/36 Summary Brochure	AS58
Model 6/43 Brochure	AX44
HARDWARE	
6/34 Brief	AR70
6/36 Brief	AS87
6/43 Brief	AX40
MDC9101 Multiple Device Controller Brief	AR71
MSC9101 Mass Storage Controller Brief	AS59
MLC9103 Multiline Communications Processor Brief	AS80
GIS9001 General-Purpose DMA Interface Brief	AT47
CRU9101/9102/9103/9104 Card Readers Brief	AS77
DKU9101/9102 CRT Keyboard Consoles Brief	AU79
TTU9101/9102/9103/9104 Teleprinters Brief	AR72
TWU9101 Keyboard Typewriter Console Brief	AS75
PRU9101/9102 Serial Printers Brief	AS78
PRU9103/9104/9105/9106 Line Printers Brief	AS79
DIU9101/9102 Diskette Units Brief	AS76
CDU9101/9102/9103/9104 Cartridge Disk Units Brief	AT65
CDU9114/9116 Cartridge Disk Units Brief	AX46
PSS9001/9002 Memory Save and Autorestart Brief	AT10
MOS Memory Brief	AU72
Core Memory Brief	AX51
Control Panels Brief	AW78
Level 6 System and Peripheral Operation Manual	AT04
MLCP Communications Manual	AT97

TABLE 1-3 (CONT). LEVEL 6 DOCUMENTATION

Title	Order Number
HARDWARE	
Office Packaging Brief	AX79
CPF9503 Scientific Instruction Processor Brief	AX43
DCM9110 Communications-Pac (Auto Call Feature) Brief	AX66
CRU9108/9109/9110 and CRU9111/9112/9113 Card Readers Brief	AX71
Ruggedized Level 6 (RL6) Brief	AY06
MTC9101 Magnetic Tape Controller Brief	AX25
MTU9112/9113/ 7-Track Magnetic Tape Units Brief	AX52
MTU9104/9105 9-Track Magnetic Tape Units Brief	AX24
TWU9104/9106 Keyboard Typewriter Consoles Brief	AX53
Site Preparation Manual	AY52
SOFTWARE	
GCOS/BES1 FORTRAN Brief	AS93
GCOS/BES1 Overview Brief	AT05
GCOS/BES1 Executive Brief	AT06
GCOS/BES1 Program Development Brief	AT07
GCOS/BES1 Utility Brief	AT08
GCOS/BES1 Assembler Brief	AT09
GCOS/BES2 Executive Brief	AT77
GCOS/BES2 Program Development Brief	AT78
GCOS/BES2 Macro Preprocessor Brief	AT79
GCOS/BES2 Communications Brief	AT80
Test and Verification Programs for 6/30 and 6/40 Models Brief	AT67
GCOS/BES2 Sort Brief	AX81
GCOS/BES2 Software Overview Brief	AX80
GCOS/BES2 Sort Manual	AV49
GCOS/BES1 Software Overview Manual	AS27
GCOS/BES1 Executive Modules I/O Manual	AS28
GCOS/BES1 Program Development Tools Manual	AS29
GCOS/BES1 Utility Programs Manual	AS30
GCOS/BES1 Assembly Language Manual	AS31
GCOS/BES FORTRAN Language and Subroutines Manual	AS32
GCOS/BES1 Operator's Guide Manual	AS33
GCOS/BES Planning and Building Online Application Manual	AS34
GCOS/BES2 COBOL Reference Manual	AU41
GCOS/BES2 Assembly Language Reference Manual	AU43
GCOS/BES2 BASIC Reference Manual	AU44
GCOS/BES2 Executive and Input/Output	AU45
GCOS/BES2 Operator's Guide	AU46
GCOS/BES2 Utility Programs	AU47
GCOS/BES2 Program Development Tools	AU48
GCOS/BES2 Planning and Building an Online Application	AU49
GCOS/BES2 Software Overview and System Conventions	AU50
GCOS 6/MDT Overview Brief	AX54
GCOS 6/MDT Executive Brief	AX58
GCOS 6/MDT Sort Brief	AX55
GCOS 6/MDT Program Development Brief	AX57
GCOS 6/MDT Macro Preprocessor Brief	AY74
GCOS 6/MDT RPG Brief	AX35
GCOS 6/MDT COBOL Brief	AX59
GCOS 6/MDT FORTRAN Brief	AX62
GCOS 6/MDT Communications Brief	AX56

TABLE 1-3 (CONT). LEVEL 6 DOCUMENTATION

Title	SOFTWARE	Order Number
GCOS 6/MDT Assembler Brief		AX61
GCOS 6/MDT File Management Brief		AX60
GCOS 6/MDT System Control Brief		AX76
L6-L66 File Transmission Brief		AW69
GCOS 6/MDT System Control		AX07
GCOS 6/MDT Program Preparation		AX08
GCOS 6/MDT File Organization		AX09
GCOS 6/MDT Monitor & I/O Calls		AX10
GCOS 6/MDT Overview & User's Guide		AX11
GCOS 6/MDT Assembly Language		AX12
GCOS 6/MDT COBOL		AX13
GCOS 6/MDT FORTRAN		AX14
GCOS 6/MDT Sort		AX15
GCOS 6/MDT RPG		AX16

SECTION 2

SYSTEM ARCHITECTURE

CONNECTABLE UNITS

There are three principal types of units in all Level 6 systems: central processors, input/output controllers, and memories. One or more of each type can be included in a system. All are interconnected by the Level 6 Megabus.

Up to 23 connectable units are supported by the Models 6/36 and 6/43, and up to 4 by the Model 6/34. In this context, a connectable unit generally is synonymous with a board and can be any one of the following:

- o a central processor
- o a memory controller with 8K-32K 16-bit words of memory
- o a multiple device controller (MDC) capable of supporting up to 4 different devices
- o other peripheral controllers
- o a multiline communications processor (MLCP) capable of supporting up to 8 full-duplex lines/4 broadband/4 HLDC lines
- o a Scientific Instruction Processor (6/43)

Several controllers of the same type may be included in a system. Each is identified by a unique module address which is easily field-adjustable. Figure 2-1 shows a 64K Model 6/36 system with 3 MDCs and 1 MLCP. This system could support, for example, 4 diskettes, 2 printers, 4 CRTs, a card reader, a teleprinter, and up to 8 communications lines to remote terminals and/or host processors. Yet only 7 connectable units are required, and a 10½-inch, 10-slot 6/36 chassis would have 3 empty slots for future use. If further expansion is required, additional chassis can be added until the 23-module capacity is reached.

A 6/30 system can have up to 32K (6/34)/64K (6/36) words of memory; a 6/40 system up to 1M words. This is housed on one (or more) 32K word controller board(s) attached to the Level 6 Megabus. However, in certain cases it may be advantageous to split 32K-words of memory between two or more boards, allowing a greater degree of overlap to take place in the system. Figure 2-2 shows two ways of configuring a 48K memory. The standard way is to use two controllers, the first with 32K, the second with 16K. However, the alternate method of putting 16K on each of three boards is allowable as long as the

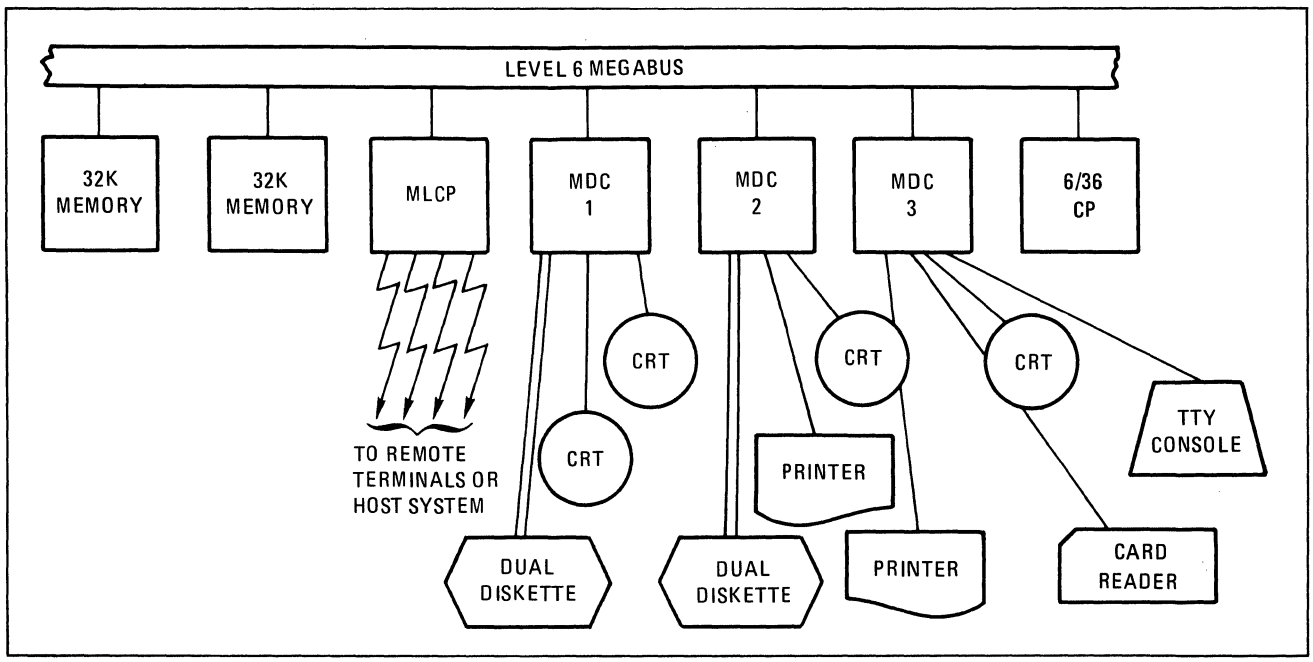


Figure 2-1. Typical 6/36 Configuration Diagram

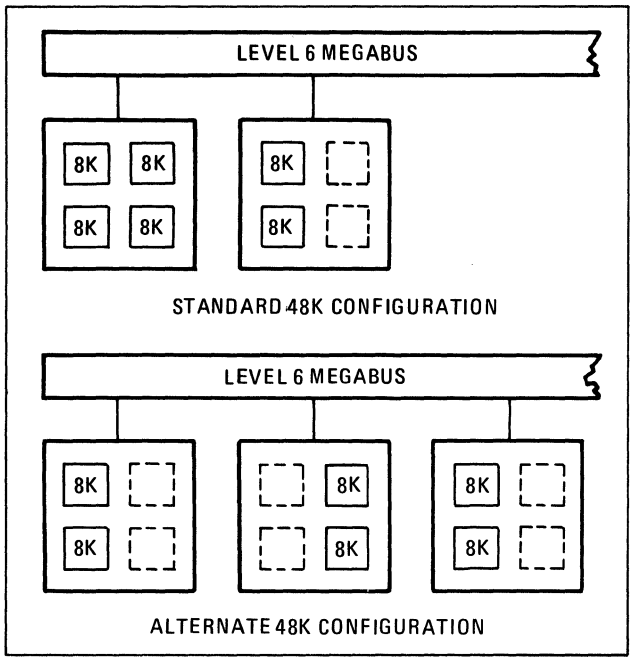
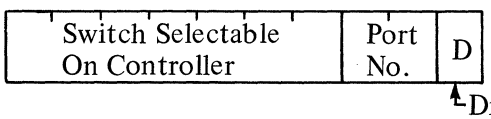


Figure 2-2. Memory Configurations

physical modules on the first two boards do not occupy the same positions. In this case, the same address is set in each of the first two boards, and a different address in the third.

Input/output controllers are used to control one or more peripheral devices or communications lines. A multiline communications controller can, for example, handle up to 8 full-duplex lines or 16 simultaneous input/output paths. Each path is considered to be a channel and will have a unique channel address. The logical capacity of the system is 1024 channels. Every device will have at least two channels assigned to it, even though both are not always used. Odd numbered channels are used for output devices, even numbered channels for input devices. Thus, a line printer requires two channel addresses, but only the odd numbered channel will be used. Two diskette units on a single controller would have four channel numbers assigned, one for each device in each direction.

The channel number for each data path is a ten-bit value. Of these the high-order bits are factory assigned by means of an adjustable rotary switch which is easily field changeable. The low-order bits identify the "port" on the controller, with the least significant bit the direction bit—odd for output, even for input. A typical MDC device channel number is shown below.



Multiple devices of the same type can thus be assigned to different channels. Channel numbers are a means of addressing input/output units and have nothing to do with either bus priority or interrupt priority levels.

INTERUNIT COMMUNICATION

If a processor wants to store a word in memory, it sends that word together with its memory address down the Megabus to the memory. In this case, it acts as the master and the memory acts as the slave. The transfer of the information from a master unit to a slave unit takes a single *bus cycle*. There are several types of bus cycles, and various units can become masters and various units slaves. The bus cycle described above is called a write cycle and is pictured in Figure 2-3.

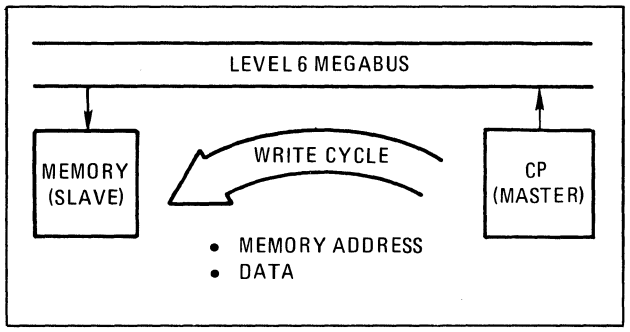


Figure 2-3. Write Cycle

The bandwidth of the bus is 6 million bytes per second, a bus cycle taking approximately 300 ns. The bus is asynchronous in design, permitting units of varying speeds to operate efficiently on the same system. The extremely high speed of the bus allows a great many operations to be multiplexed, thereby giving a high degree of overlap among various system elements.

Megabus Priority

Any unit on the Megabus can become the master. To do so, it must have a higher priority than any other unit simultaneously also trying to become a master. Bus priority is dependent upon the relative position of the unit on the bus. Memories have the highest priority, intervening units have lesser priority, and the central processor has the lowest. While this is shown in the diagrams as the highest priority on the left, in actuality the memories are physically at the bottom of a unit and the central processor is at the top. Therefore, the highest priority devices are those toward the bottom. See Figure 2-4.

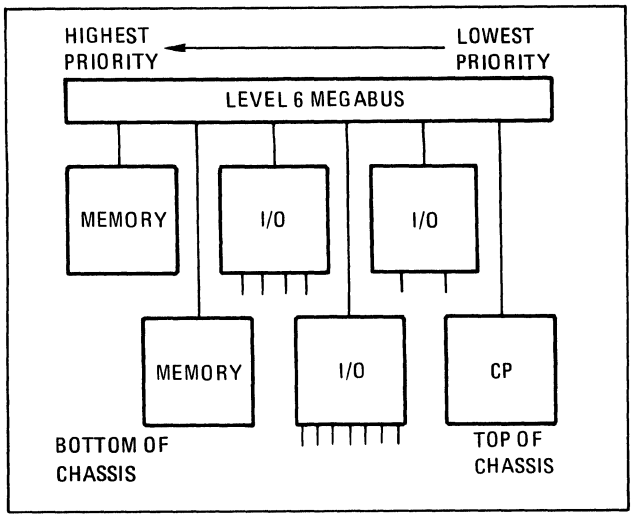


Figure 2-4. Bus Priority

Types of CP and Memory Transfers

The write cycle shown in Figure 2-3 is one type of transfer between a central processor and memory. A write requires a single bus cycle. Another common operation occurs when the central processor wants to read a word of data from memory during either an instruction fetch or an operand fetch. A read operation requires two bus cycles: a read request cycle and a read response cycle. During a read request cycle, the central processor is the master and the memory the slave. The central processor sends a memory address plus its own processor channel number to the memory. Accompanying this transfer is a signal that says a response is requested. Upon receipt of this request, the memory accesses data and initiates a response cycle. For this cycle the memory becomes the master and the CP becomes the slave. The memory puts the data, together with the central processor channel number, on the bus and the CP accepts it.

Figure 2-5 illustrates a typical read operation for a processor with: 1) a single-fetch memory and 2) a double-fetch memory. As can be seen, the single-fetch memory read operation is a two-bus cycle operation, while the double-fetch memory read operation is a three-bus cycle operation and uses two memory cycles. In either case, both units will appear busy to any other unit that tries to address them during this time – with the exception of the central processor, which can accept interrupts from other units, request other cycles, or receive responses between the read request and the read response bus cycles.

The memory does not initiate the read response cycle until after it has accessed the data. This is done during a 650-ns memory cycle (for single-fetch memory) which partially overlaps the two bus cycles (see Figure 2-6); or two 550-ns memory cycles (for double-fetch memory) that partially overlaps three bus cycles. During this time the bus is free to accept requests from other units, interleaving bus cycles and effectively overlapping operations.

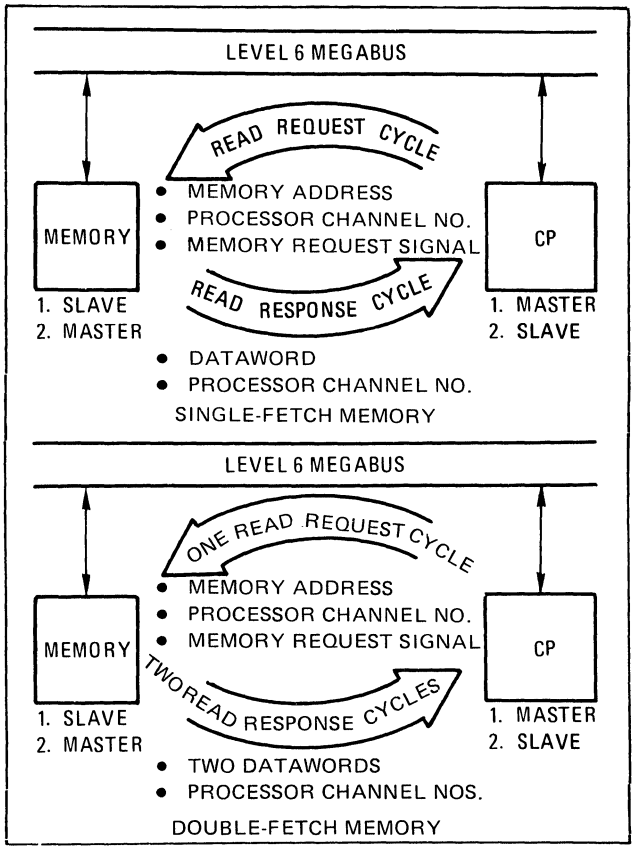


Figure 2-5. Read Operation

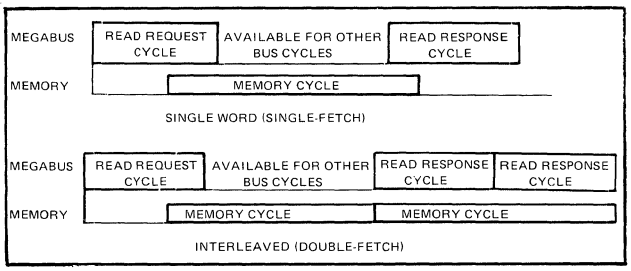


Figure 2-6. Timing of Memory Read Operation

Certain central processor to memory operations operate in read-modify-write mode and require three cycles: a read request, a read response, and a write cycle. Among these are a group of central processor instructions specifically designed to support future multiprocessor operation in that initiation of these instructions locks memory for the full three cycles.[1] This

[1] The six CP instructions that "lock" memory are INC, DEC, LBF, LBT, LBC, and LBS. (See Section 4 for details.)

feature allows one processor to set flags in memory to be tested by another processor, without the chance of the second processor disturbing the flags while the process is taking place.

Input/Output Transfers

All I/O transfers occur in direct memory access (DMA) mode. Once a channel is set up by the central processor, data transfers are effected via the bus independent of the central processor. When a channel wants to input a word to memory, it becomes the master and initiates a write cycle transmitting the data and the memory address to the memory (Figure 2-7). If the channel is connected to an output device and it wants to receive a word from memory, it initiates a read request cycle and then, after the memory has accessed the data, a second bus cycle, the read response, is initiated with the memory as master and the channel controller as slave (Figure 2-8).

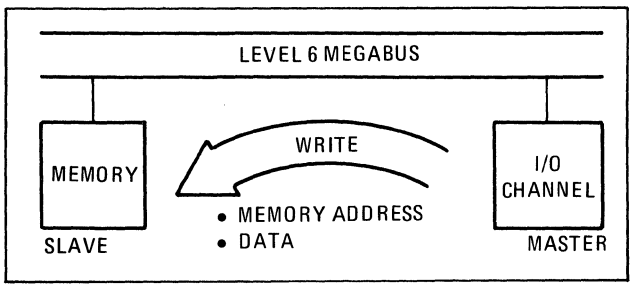


Figure 2-7. I/O DMA Input Operation

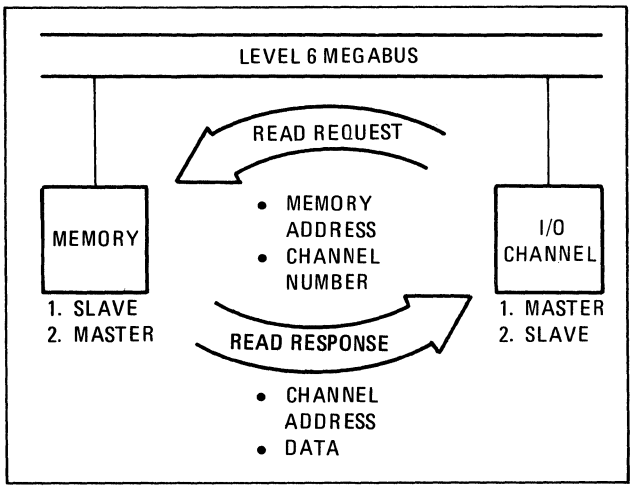


Figure 2-8. I/O DMA Output Operation

Multiple I/O units (including those on the same controller) can be operating simultaneously and can multiplex on a word basis to the same memory as long as the combined data rates do not exceed the maximum memory transfer rate of approximately 1.1 million words per second. As this rate is approached the central processor is locked out since I/O channels have a higher bus priority than the CP. If a second (or third) memory controller is added, true overlap can be attained, with the central processor communicating to one memory unit at the same time as the input/output channel is communicating to the second memory unit.

Input/Output Commands

In order for an input/output channel to initiate a DMA block transfer, it must first be set up by a central processor. This is done via an I/O output command. The central processor sends several words of information to the I/O controller, taking a single I/O write cycle to send each word (see Figure 2-9). The processor sends a word of information together with a channel number and a function code defining what that word of data is. Typical words that must be sent from a central processor to a controller are *task* words identifying the operation to be performed, *configuration* words showing such things as the track and sector of a disk file, *address* words showing the address in memory where the transfer is to start, and *range* words showing how many words or bytes of data are to be transferred.

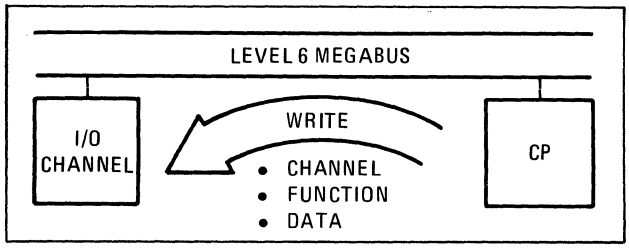


Figure 2-9. I/O Output Command

In addition to I/O output commands, the processor can initiate I/O input commands. These are two-cycle operations in which the processor issues a read request to the I/O channel, sending it both a channel number and a function code, plus its own processor channel number. The read response cycle is then initiated by the I/O channel, with the contents of a particular channel register being sent back to the central processor. Typical registers which are read by the processor are *status* registers showing status and error information, *range* registers typically read after a DMA transfer to determine how many characters have been read in, and *identification* registers showing a fixed ID for a particular device type. The latter allows the software to identify exactly what type of device is on a particular channel.

The two cycles of an I/O input command are shown in Figure 2-10.

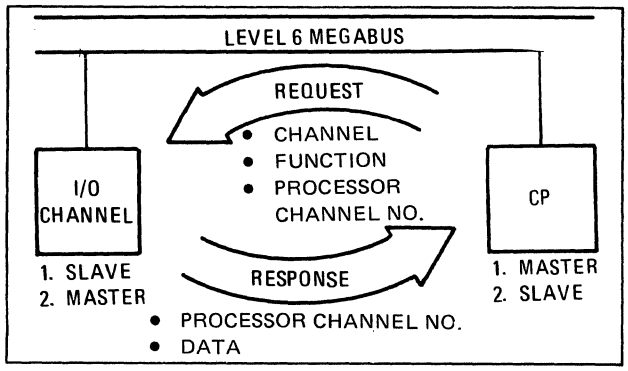


Figure 2-10. I/O Input Command

Interrupt Levels

One I/O output command in particular is important at this point. This is the output interrupt control function. Utilizing this command, a processor can assign an interrupt level to a particular channel. There are 64 interrupt levels. Any channel can be assigned any interrupt level by the central processor. Several can have the same interrupt level if necessary. To assign an interrupt level the processor executes an I/O output command which transmits a data word containing the interrupt level and the CP channel number to the I/O channel. This then becomes the interrupt level at which that channel may in the future interrupt. It should be noted that priority levels are thus software assigned and are completely independent of physical position.

Interrupt Commands

When a channel wants to interrupt because of a certain condition in the controller (typically the end of a block transfer), the channel initiates an interrupt cycle (see Figure 2-11). In this cycle, it places its channel number and its interrupt level on the Megabus and attempts to transmit these to a processor (not necessarily the same one which had previously set up the interrupt level). A register in the processor (the S register) defines the priority level at which the processor is currently operating. Priority levels are assigned such that level 63 is the lowest priority and level 0 is the highest priority (reserved for power failure interrupt). If the level number of the interrupting device is numerically lower (higher priority) than the level of the central processor, an interrupt will occur. If not, then the central processor will reject the interrupt and no more interrupt cycles will be placed upon the bus by the I/O channel until it receives a signal from the central processor stating that the priority level of the central processor is changing and that it should thus reinitiate the interrupt request.

Normally, a central processor cannot receive any bus signals between the time that it issues a read request and the time that it receives a read response bus cycle. An interrupt request is an exception. It can be received by the central processor at any time, including the time between the two cycles of a read operation, although it will not be processed until the instruction is complete.

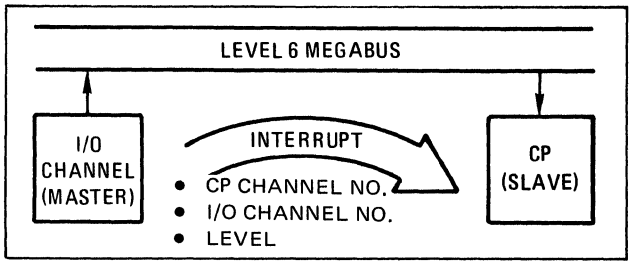


Figure 2-11. Interrupt Cycle

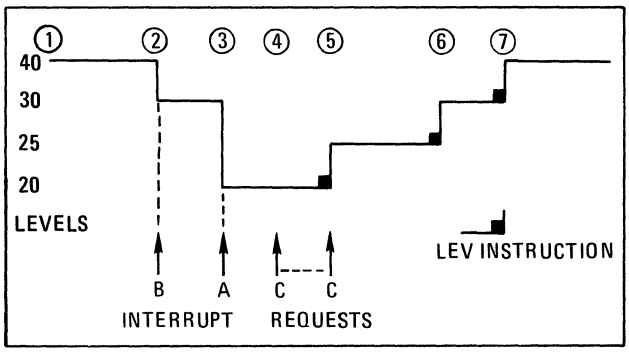


Figure 2-12. Interrupt Action

Interrupt Action

When an interrupt is accepted, the central processor typically assumes the level of the interrupting source. It remains at this higher priority level until interrupted by some still higher priority level, or until the CP finishes the interrupt routine and executes an instruction (the LEV instruction) that returns control to either the previously interrupted level or to an intervening level.

Consider the following (see Figure 2-12):

1. Processor running at level 40 sets up devices A, B, and C at interrupt levels 20, 30, and 25, respectively.
2. Device B requests interrupt and is accepted. Processor now runs at level 30.
3. Device A requests interrupt and is accepted. Processor now runs at level 20.
4. Device C requests interrupt but is not accepted. It will not try again until processor changes levels.

5. Processor finishes processing level 20 and issues LEV instruction. This allows C to request interrupt again; it is now accepted and processor runs at level 25.
6. Processor finishes processing level 25 and issues LEV instruction; no higher interrupts are pending, so it resumes level 30 processing.
7. Processor finishes level 30, issues LEV, and goes back to original level 40.

Context Switching

Each interrupt level has its own individual context save area pointed to by a dedicated vector. Upon interrupt, the CP firmware (after clearing all pending traps at the running level) automatically stores the contents of all active registers in the context save area of the interrupted level and loads the registers as specified by the Interrupt Save Mask 1, 2. The number of registers whose context is saved and restored is under the control of a mask for each interrupt level, with the mask set up by the software. Thus, anywhere from 2 to 18 (for 6/30 models) or 2 to 26 (for 6/40 models) registers may be automatically switched upon initiation of a new level. (For details on registers and the action of interrupts, see Section 3.)

Summary of Bus Operations

Figure 2-13 summarizes the various Megabus operations described. The Megabus itself contains the following 51 information signals:

- o 24 address bits
- o 16 data bits
- o 6 control bits
- o 5 integrity bits

There are also 17 other lines used for timing and other functions. Data lines typically hold a 16-bit data word or, on data requests, the "return address" of the requesting channel. The address lines contain either the channel number of the destination or a memory address as designated by one of the control lines. It should be noted that memory addresses are not used to specify registers in various I/O channels. The latter are specified by a 10-bit channel number and a 6-bit function code, leaving the full range of possible memory addresses for addressing memory. A 32K system will have all 32K words of memory addressable and a 128K system will have 128K words addressable, etc.

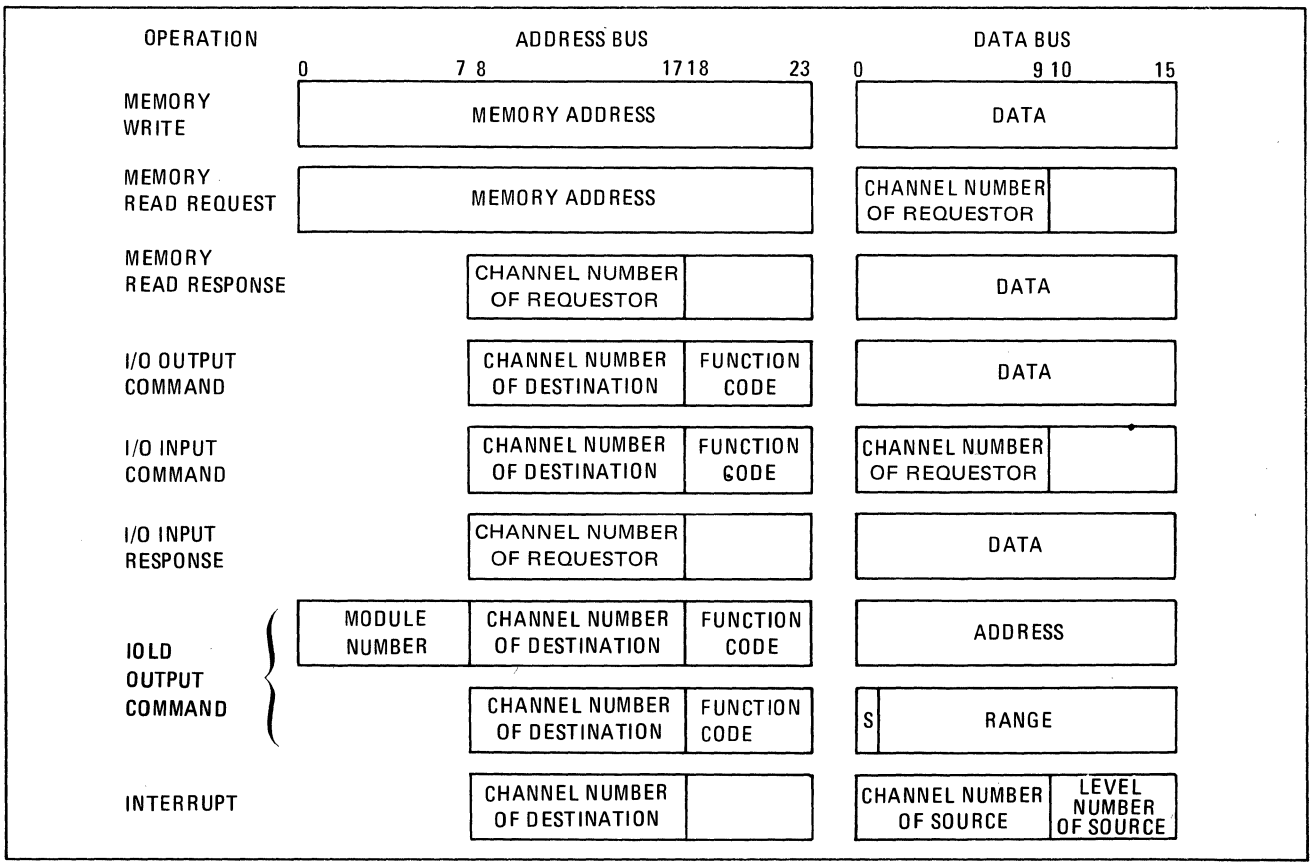


Figure 2-13. Data and Address Bus Formats

MEMORY ADDRESSES

The basic unit of addressing is a word. A 6/36 has 16-bit address registers; therefore, it can address 64K words of memory. For byte addresses, a 17th bit is appended and, therefore, it can directly address 128K bytes of memory. This is also the way a 6/43 works in the short address format (SAF) mode. The 6/43, however, has 20-bit address registers. When in the SAF mode, only 16 bits of these registers are utilized. When in long address format (LAF) mode, all 20 bits are used for word addressing. Thus, the 6/43 can directly address 1 million words of main memory. Again, for byte addresses an extra bit is appended. In this case, a 21-bit address results, allowing the 6/43 to directly address 2 million bytes of memory.

The Level 6 Megabus has 24 address lines utilized to address memory to the *byte* level. Thus, the Megabus has an architecture that is "open-ended" in that it can support processors with a direct addressing capability of over 16 million bytes. Of these 24 lines, 17 are actually used by 6/30 systems and 21 by 6/40 systems.

On word addresses, a 17-bit address is always generated in a 6/30 or in a 6/40 operating in SAF mode. A 6/40 operating in LAF mode will generate a 21-bit address. If a byte-oriented instruction is being executed, the CP informs the memory via a control line on the Megabus (otherwise the memory assumes a word address). The memory then uses the low-order Megabus address bit to determine which byte of a word should be written into (the remaining byte will not be altered). The low-order bit is always ignored for memory reads, the entire word is returned, and the receiver determines which byte to use.

The CP asks for the word in question and performs the byte indexing internally. The memory always returns the complete word.

Through the use of indexing, 17-bit byte addresses are generated in SAF mode, 21-bit in LAF mode. A byte count in an index register is shifted one place to the right before being added to a 16-bit word address of a 6/30 or a 20-bit word address of a 6/40. Thus, if an index register contained the quantity 5 and a word address of 1000 was specified, the fifth byte (starting at zero) would be selected from the left-half of word 1000, or the right byte in word 1002 (see Figure 2-14). Notice that byte addresses are sequential (left to right) in memory and that data is thus in its correct sequence.

A similar scheme is used with *bit instructions* to address bits; in this case a bit count is shifted four places to the right prior to being added to a word address.

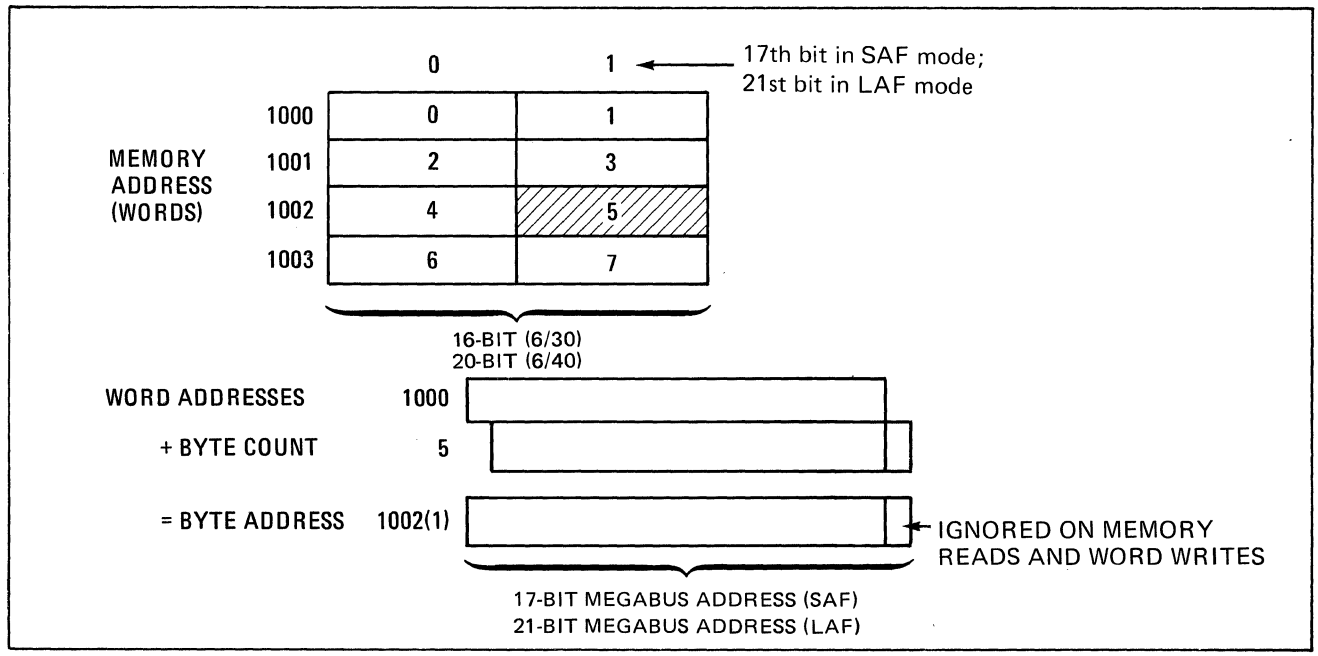


Figure 2-14. Byte Addressing

SECTION 3

CENTRAL PROCESSOR ARCHITECTURE

This section describes the architecture of the 6/34, 6/36, and 6/43 central processors. From a functional point of view the processors are identical. The differences between them are in the amount of memory they can address, the number of registers, the kinds of options, and the number of peripherals each is capable of handling. For the purposes of this discussion they will be treated as one and the same.

REGISTERS

In the *6/30 systems,* there are 18 registers visible to the programmer:

- 7 general registers (R1–R7) – 3 (R1–R3) can be used as index registers
- 7 address registers (B1–B7)
- 1 program counter (P register)
- 1 system status register (S register)
- 1 mode control register (M1 register)
- 1 indicator register (I register)

In the *6/40 systems,* there are 26 registers visible to the programmer. They include all of those found in the 6/30, plus the following:

- 1 stack address (T) register
- 1 RFU address register
- 4 RFU mode control registers (M2, M3, M6, M7)
- 2 mode control registers for the SIP (M4, M5)

All of the 6/30 and 6/40 registers can be accessed via the control panel. Three other registers can also be accessed from the control panel:

- Instruction register
- Memory address register
- Memory data register

(See Section 5 for control panel operation.)

Finally, there is a single 32-bit scientific register that is simulated by software and is used to store floating-point operands. This register is utilized by the scientific instruction set that is automatically trapped to software routines in 6/30 systems. In 6/40 systems, an optional Scientific Instruction Processor is offered that can hold either single-precision (32-bit) or double-precision (64-bit) floating-point quantities.

DATA FORMATS

The word length in 6/30 processors is 16 bits and in 6/40 processors 16 or 32 bits. All hardware registers are 16 bits in length except for the M registers, which are 8 bits (address registers are 20 bits when the 6/40 is in the LAF mode). Within this framework the central processor has the ability to process double-words (32 bits in length), words (16 bits), half-words or bytes (8 bits), and single bits.

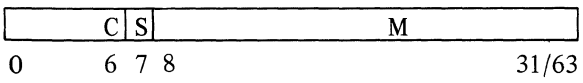
Basically, there are two types of data: signed data used in arithmetic operations; and unsigned data, used for logical quantities, addresses, ASCII characters, or any other type of internal data coding.

Unsigned data is usually expressed in hexadecimal notation. Thus, a 16-bit address can range from $(0000)_{16}$ to $(FFFF)_{16}$. The contents of a byte can be expressed from $(00)_{16}$ to $(FF)_{16}$. A 16-bit word can contain two ASCII characters; if a word contained $(4139)_{16}$, this would represent "A9" in ASCII.

Arithmetic (signed) data is represented in twos complement notation. All arithmetic is performed in binary, with single word (16-bit) values extending from -32,768 to +32,767. A signed value in a byte can range from -128 to +127. Figure 3-1 shows the various data types for both signed and unsigned data.

A byte in memory can represent either an unsigned 8-bit quantity or an arithmetic value with a sign and seven bits. The byte can occupy the left- or right-hand half of a word. However, when the byte is loaded into a register, it will occupy the right-half of the register. The left-half will contain either zeros, if a logical load instruction was used, or the sign extended, if an arithmetic load was used. See Figure 3-2 for examples.

Floating-point values occupy two or four words. The format for a floating-point value is as follows:



Where C is a 7-bit value representing the characteristic expressed as an excess 64 power-of-sixteen exponent.
S is a sign of the mantissa
M is the magnitude of the mantissa

GENERAL REGISTERS

There are seven general registers (R registers) numbered R1 through R7. These registers may be loaded from or stored in memory either a word or a half-word at a time. (If a half-word store is executed, it is the right-half of the register that is stored.) It is possible to load and store two of them (R6 and R7) as a single double-word. Each register can be shifted individually or as pairs (even/odd) and used as operands in arithmetic, logical, and compare operations. In addition, the first three registers (R1 – R3) also double as index registers and may be used to store double-word, word, half-word, or bit counts. These quantities are used to modify the addresses of items in memory. See Figure 3-3.

ADDRESS REGISTERS

In addition to the seven general registers, there are seven address registers (B registers). These are also known as base registers and are numbered B1 through B7. It is a very important concept of Level 6 architecture that the general (R) registers are separate from the address (B) registers. The 6/40 also has a stack address (T) register and another address register reserved for future use.

Address registers in 6/30 systems are all 16 bits in length; however, in 6/40 systems they can

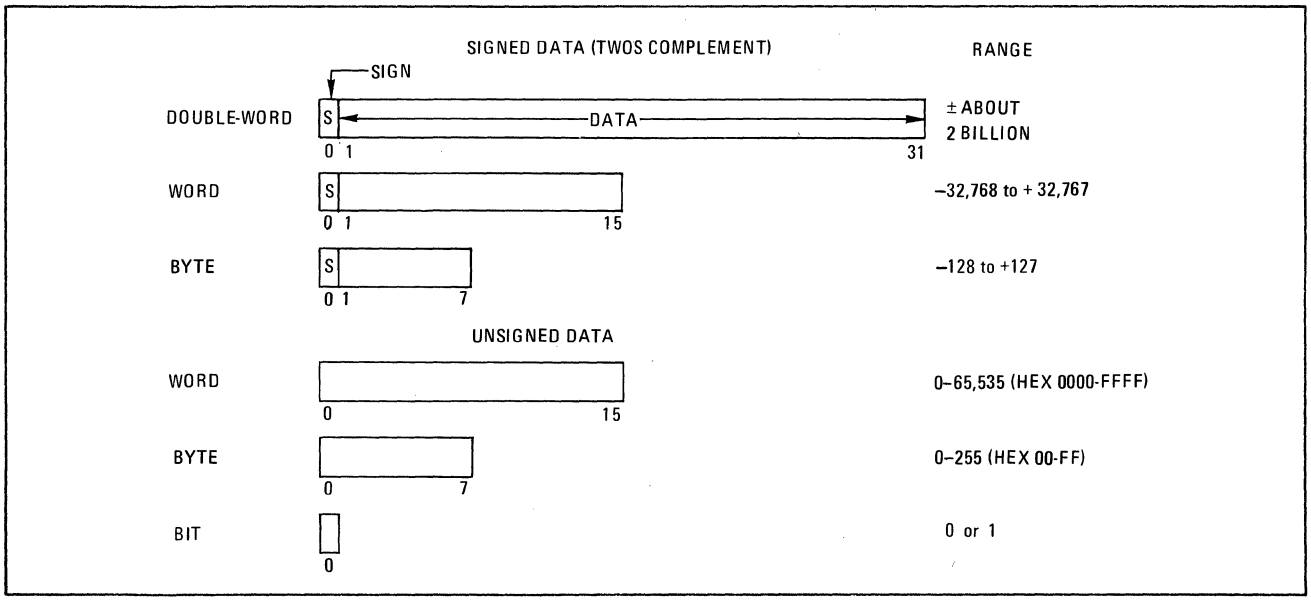


Figure 3-1. Data Formats

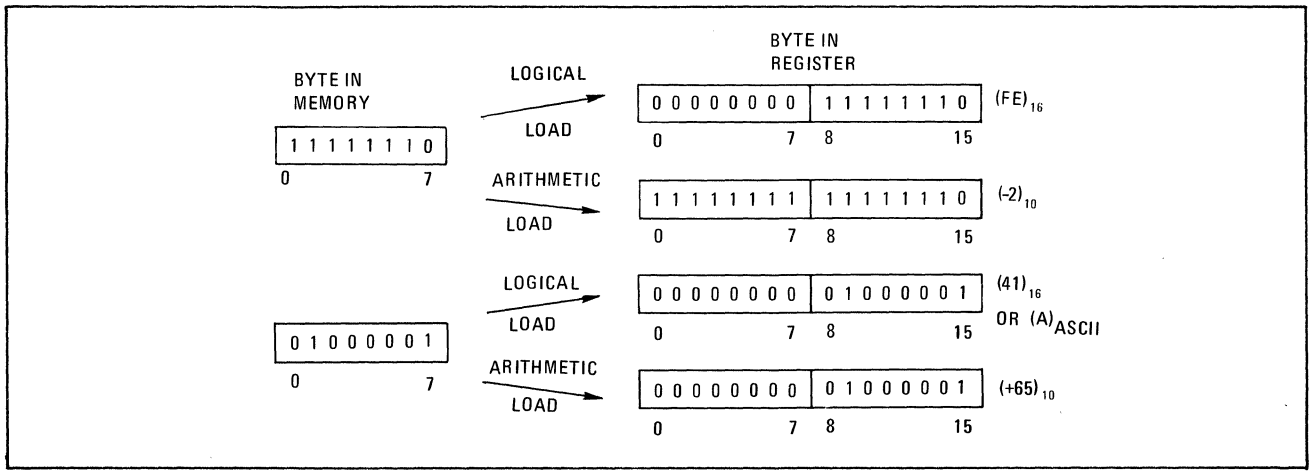


Figure 3-2. Byte Formats

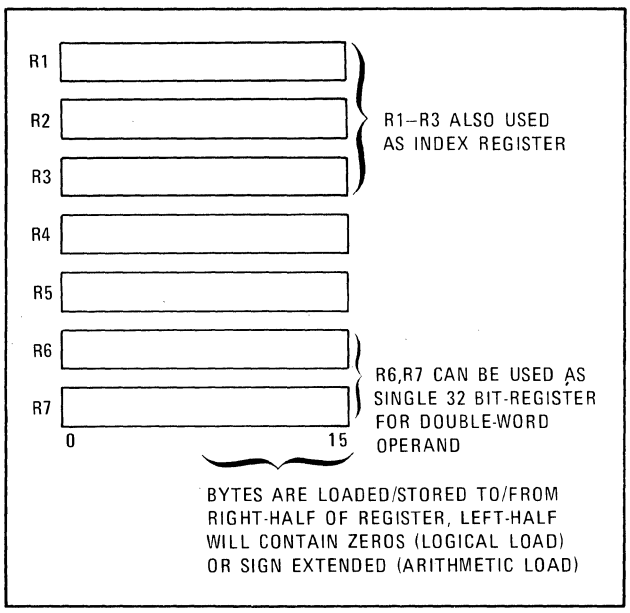
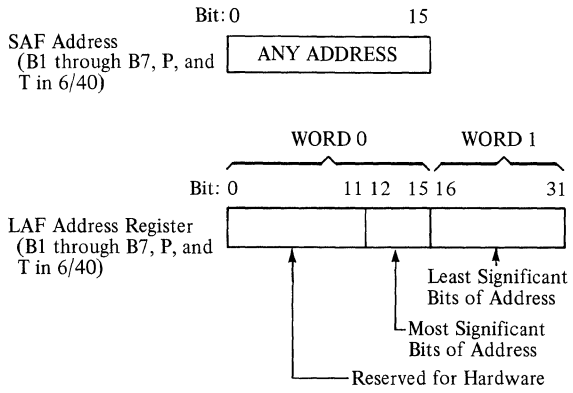


Figure 3-3. General Registers

be either in the Short Address Form (SAF) or in the Long Address Form (LAF). SAF is oriented to systems of 128K bytes (64K words) or less of addressable memory and each word can be accessed through a 16-bit address pointer. LAF is oriented to systems of 2^{20} words and each word can be accessed through a 32-bit address. The length of the direct physical address, however, may not exceed 20 bits. The most significant 12 bits of the 32-bit LAF address are used by the optional memory management unit (MMU).



The seven base registers can be used for formulating addresses by pointing to any procedure, data, or arbitrary location in the system. Address registers typically contain addresses and are not used for arithmetic calculations. However, they do have the capability of being automatically incremented or decremented during instruction execution. This allows them to be used to scan arrays either forwards or backwards and also to conveniently utilize stacks.

Addresses are normally expressed in memory as four or eight hexadecimal digits, with the memory addresses running from 0-0 through F-F. Details of how base registers are used in formulating addresses are given in Section 4.

PROGRAM COUNTER

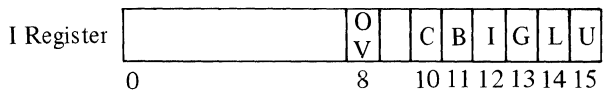
The program counter (P register) is a 16-bit (6/30) or 20-bit (6/40) register which points to the next instruction to be executed. Instruction length is variable. The majority of instructions are one or two words in length, but certain instructions can be as long as six. The 6/30 program counter is always incremented during instruction execution to point to the next sequential instruction, except for branches and jumps. Thus, for example, if a two-word instruction is executed, the program counter will be incremented by two during its execution.

The 6/40 has a pre-fetch or look-ahead capability; therefore, the program counter will point at *two* instructions ahead, rather than the next one.

INDICATOR REGISTER

The indicator register (I register) is a 16-bit register that contains various single-bit indicators. The register format is as follows:

The indicators contained in this register can be grouped as follows:



- o Arithmetic indicators
  - – OV (overflow indicator)
  - – C (carry bit)
- o Comparison indicators
  - – G (greater than indicator)
  - – L (less than indicator)
  - – U (unequal signs indicator)
- o Bit indicators
  - – B (bit test indicator)
- o I/O Indicator
  - – I (input/output indicator)

Arithmetic Indicators

Two indicators are affected by arithmetic and shifting operations: the overflow (OV) indicator and the carry (C) indicator.

The overflow indicator is set when any of the seven general registers "overflows," that is, when an arithmetic result produced is larger than the capacity of the register. For example, adding the

quantity 1 to a register that contains +32,767 $(7FFF)_{16}$ will set overflow because the *arithmetic* capacity is exceeded. The register would contain $(8000)_{16}$, or -32,768 after the addition. If the data were not to be interpreted as signed data, the overflow could be ignored.

The carry (C) indicator, conversely, is set when the *logical* capacity of a register is exceeded. Thus if 3 were added to a register that contained $(FFFE)_{16}$, a carry to the 17th bit would be produced, i.e., the answer would be $(1)(0001)_{16}$. However, if these were arithmetic quantities, $(FFFE)_{16}$ would represent -2, and adding +3 to it would produce $(0001)_{16}$; this is the correct answer and thus the carry could be ignored.

Comparison Indicators

The three indicators which are controlled by compare instructions are the greater than (G), less than (L), and unlike signs (U) indicators. These one bit indicators contain the results of the last compare instruction executed by the computer and in turn can be tested by other central processor instructions (see Branch on Indicator in Section 4).

A comparison is typically executed between a register and a word in memory. If the contents of the register are greater, the G indicator is set; if the contents of memory are greater, the register is less and L is set.

Bit Test Indicator

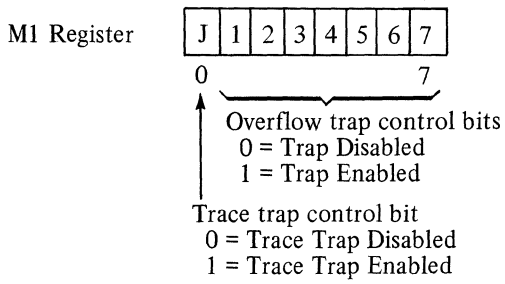
This can be considered a one bit register loaded by load bit instructions. Not only can it be loaded from any bit in any word of memory, but it can also be set if any of a group of bits in a word in memory is set. The selection is done under the control of a 16-bit mask and is very useful for testing bit patterns in memory.

Input/Output Indicator

The I-bit stores an indication of whether the last input/output command was successful. For example, if the central processor issues a command output to a peripheral channel that is busy, the peripheral channel will issue a "NAK," which in turn will cause the I/O indicator to be cleared. Software can then test this indicator and determine what alternate action to take.

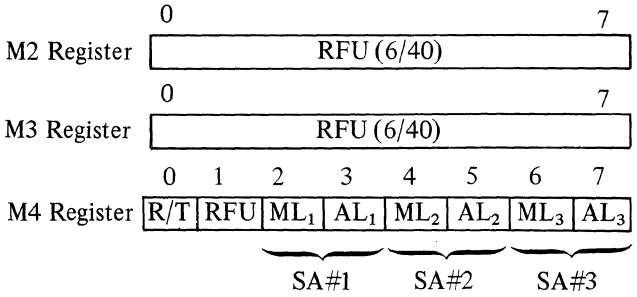
MODE CONTROL REGISTER

Other control registers are the trap enable/mode control registers (M1–M7). These are all 8-bit registers.



The seven overflow trap control bits are associated with the seven general registers, R1–R7. If overflow in any one of these occurs and if the corresponding trap control bit is set, a trap occurs through trap vector 6. This facility saves the programmer from having to test the result of every arithmetic operation for overflow, and yet also guarantees that overflow will not go undetected.

The other bit in this register, bit zero, is the trace trap enable bit (J-bit). When this is enabled, all jumps and branches that are executed in a program will cause a trap to the trace entry location. This bit therefore allows a programmer to trace a code without having to modify its procedure at all.



where:

R/T = Round/Truncate Mode
0 = Truncate
1 = Round

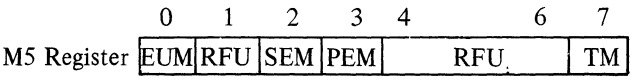
SA1 = Scientific Accumulator 1

SA2 = Scientific Accumulator 2

SA3 = Scientific Accumulator 3

ML = Memory Length – Length of main memory data field associated with this SA
0 = 2 words
1 = 4 words

AL = Accumulator Length – Length of the value in the scientific accumulator data field.
0 = 2 words
1 = 4 words



where:

EUM = Exponent Underflow Trap Mask
- 0 = Trap disable
- 1 = Trap enable

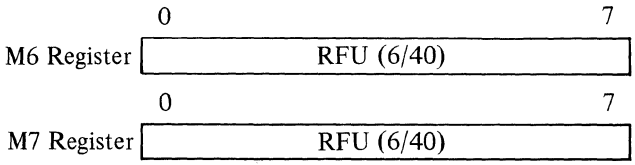
SEM = Significance Error Trap Mask
- 0 = Trap disable
- 1 = Trap enable

PEM = Precision Error Trap Mask
- 0 = Trap disable
- 1 = Trap enable

RFU = Reserved for Future Use – (MBZ)

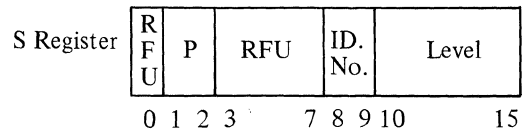
TM = Test Mode Control
- 0 = Test Mode disable
- 1 = Test Mode enable

The M4 (SIP Mode Register) and M5 (SIP Trap Register) are scientific control registers that reside in their associated 6/40 central processor with a copy of their contents in the SIP.



STATUS REGISTER

The third control register is a 16-bit register known as the status register (S register). This register contains three fields, as shown below: the privileged state indicator (P), the processor ID and the interrupt priority level.



Privileged State Indicator

There are two modes of instruction execution: user mode and privileged mode. The privileged field in the S register defines this mode as follows:

P = Privilege State (Ring Number)
- For 6/30 and 6/40 (without MMU):
  - 00 = User state
  - 11 = Privilege state
- For 6/40 (with MMU):
  - 11 = Ring 0 (Privilege)
  - 10 = Ring 1 (Privilege)
  - 01 = Ring 2 (User)
  - 00 = Ring 3 (User)
- NOTE: Privileges and access rights accorded the various rings are in inverse order to the ring number (i.e., Ring 0 is the most privileged).

If bit 1 is not set, the processor is in the user mode and will automatically trap when certain instructions are attempted (bit 2 is ignored). Input/output command instructions plus the interrupt level change instruction are privileged instructions and can be executed only when the privilege mode bit is set. They will be automatically trapped if attempted in user mode. By utilizing this hardware feature, systems can be protected against unauthorized use of input/output by user routines.

Processor ID

This is a 2-bit field that is fixed during system configuration. It is typically zero (a second processor in the system would have an ID of 01). These 2 bits are used as the four least significant bits of the 10-bit channel number for the processor itself. The high 8 bits are always zero. Thus, the processor ID and the processor channel number are for all intents and purposes synonymous.

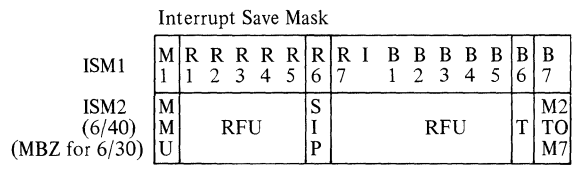
The processor ID in the S register is hard-wired and cannot be changed under program control.

Priority Level

The 6-bit level field defines the interrupt priority level on which the processor is currently executing instructions. Zero is the highest priority level and 63 is the lowest. Upon receiving interrupt requests from other units, it also determines whether the interrupting unit is of higher, equal, or lower priority. Only higher priority interrupt requests are granted. When an interrupt occurs, the level of the interrupting unit replaces the level in the status register of the interrupted level. The old level is always automatically stored. (See Interrupts for further details.)

SUMMARY OF PROGRAM VISIBLE REGISTERS

Thus 18 registers are visible to the 6/30 programmer and 26 to the 6/40 programmer. The registers are shown in Figure 3-4. Of the various registers, two are automatically saved and restored upon interrupt (the S register and the P register). The others have their context stored and restored under firmware control according to a 16-/32-bit mask. The first 16 bits of this mask, also used by the save context (SAVE) and restore (RSTR) instructions, are set up under program control. Its format is as follows:



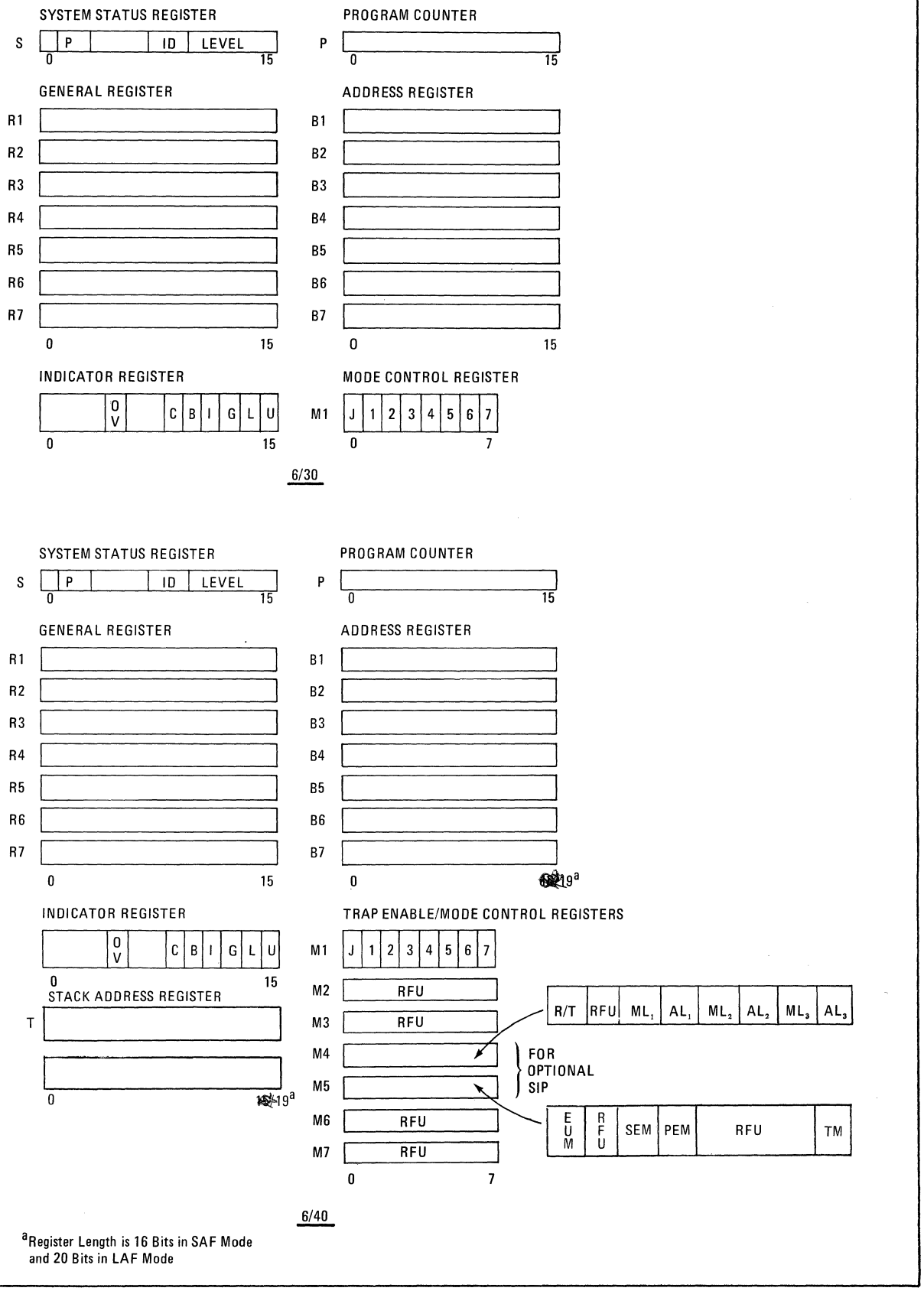


Figure 3-4. Register Complement

INTERRUPTS

There are 64 levels of interrupt, numbered from 0 to 63; level 0 has the highest priority. Clearing the computer puts it into level 0, which in effect makes it uninterruptable.

Associated with each interrupt level is a dedicated memory location which contains the interrupt vector. These locations, extending from address 0080 to address 00BF (00FF for 6/40), contain a pointer to an interrupt save area. The interrupt save area needs only to be set up by those levels that are active in a particular program. Each interrupt save area has five fixed locations and up to 16 more variable locations. These locations are as follows:

1. DEV – in this location the channel number and device ID of the interrupting device are automatically stored.
2. ISM – these two locations contain the 32-bit interrupt save mask. This mask controls which of the registers will be saved in the variable portion of the interrupt save area.
3. Reserved for future use.
4. P – in this word the program counter of the interrupt level is stored. It also acts as a pointer to the interrupt handling procedure for a new level, or upon restoration of an interrupted level, to the location of the next instruction to be executed.
5. S – this is where the status register is automatically stored. Note that when a new interrupt level is set up, the S register is loaded from this location only as far as the privileged mode field is concerned; the level is generated automatically, and the processor ID is hard-wired.

Words 6–21 are locations for saving the machine registers under control of the interrupt save mask. If the interrupt save mask is all zeros, none of these words will be reserved.

The mask bits are scanned from right (bit 15) to left (bit 0), ISM1 first and then ISM2.

It should also be noted that the word prior to the one pointed to by the interrupt vector (that is, the word prior to the device word) is also a dedicated word. This is called the TSAP and points to a trap save area. (See discussion of Traps.)

Figure 3-5 shows this action of an interrupt. Both levels 20 and 30 have had their interrupt save areas set up – level 20s at AB20, as pointed

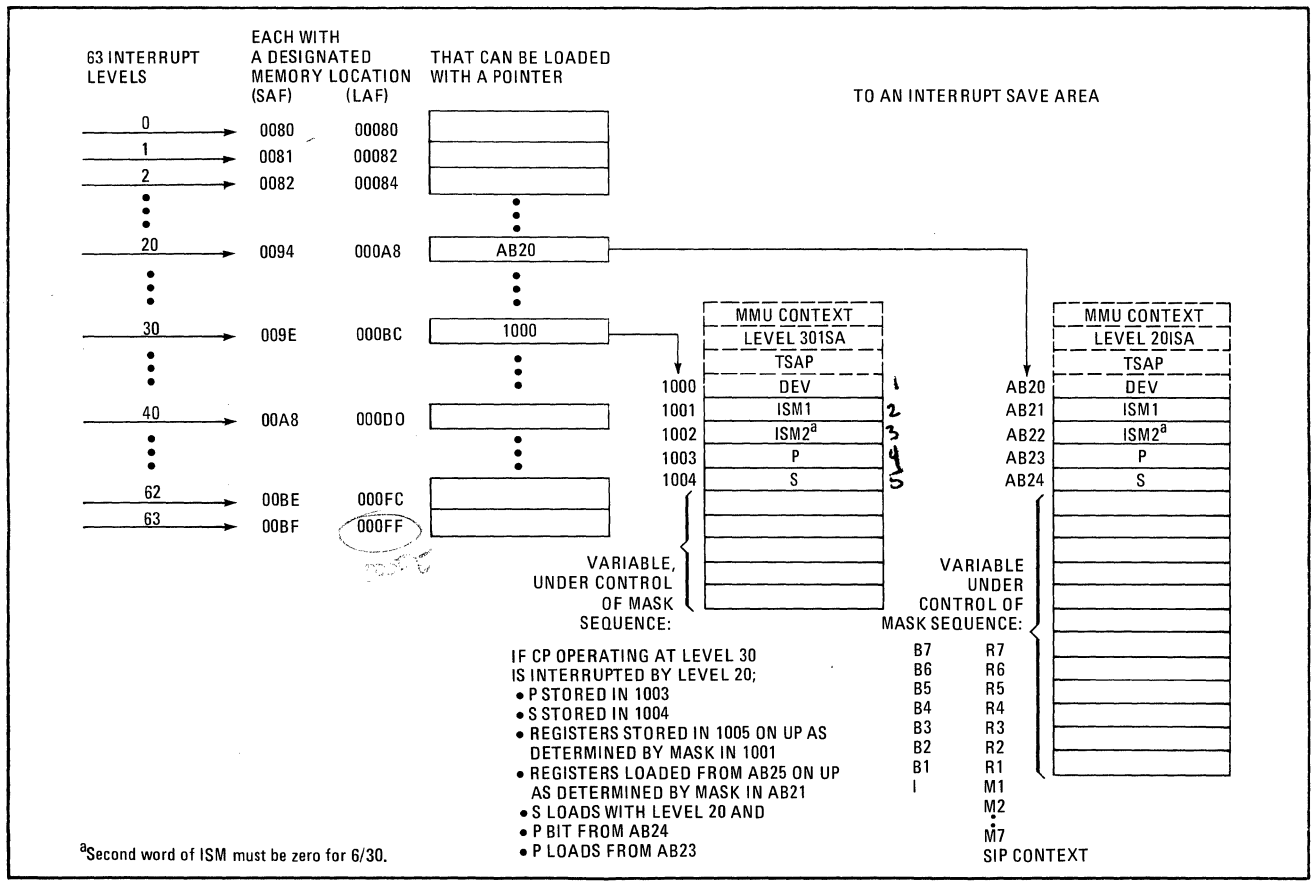


Figure 3-5. Interrupt Action

to by its vector at 0094; and level 30s at 1000, as pointed to by its vector at 009E.

If level 30 is interrupted by level 20, firmware automatically saves the register contents in the level 30 interrupt save area and loads the registers with the values in the level 20 interrupt save area. One of these values is the starting address of the level 20 interrupt subroutine which is automatically loaded (in this example) from location AB23 into P.

Associated with each interrupt level is a dedicated flag bit which is set when the interrupt is initiated. These bits are stored in four dedicated memory locations, 0020-0023. In the example above, both bits 20 and 30 would be set. At the end of the interrupt routine for level 20, a level change instruction (LEV) would be executed. This would clear the bit for level 20 and then scan the table to determine which was the next highest scheduled level. If no intermediate interrupts (such as level 25) were pending, it would scan the table, find bit 30 set, and therefore return to level 30.

The LEV instruction can set or reset activity flags, change the current level, inhibit interrupts, or do a "quick change" without saving and restoring context. By changing the current level to level 3 all device interrupts can be inhibited; level 2 (overflow of trap context scan area), level 1 (watchdog timer runout), and level 0 (incipient power failure) will still be enabled. Table 3-1 shows the assignment of interrupt levels.

TABLE 3-1. EVENT INTERRUPT LEVEL ASSIGNMENT

Event Causing Interrupt	Level Assignment	Comment
Incipient Power Failure	0	Highest priority
Watchdog Timer Runout	1	–
Overflow of Trap Context Save Area	2	–
Real-Time Clock	–	Level is contained in main memory location 0016 (HEX)
Device Requiring Service	–	Level is either fix wired or dynamically controlled by software
LEV Instruction	–	Level specified in instruction

TRAPS

Traps are caused by events such as overflows, parity errors, addressing nonexistent resources, executing a scientific instruction on a 6/30 model or a scientific error on a 6/40 model. A trap can occur at any priority level, and several can be nested at the same level. A trap could be entered at one level, that level interrupted during the execution of the trap routine, and then the same trap routine reentered in the new level. See Table 3-2.

Each type of trap has its own trap vector containing a pointer to the trap-handler procedure. Also utilized is a pointer in location 10 to the next available trap save area. The latter are pooled, and pointers to the next available area are automatically adjusted by firmware. When a trap occurs, some (but not all) register contents are automatically stored in the trap save area. The pointer in the first word of the interrupt save area for the current level is adjusted so that it points to the trap save area; the pointer in the trap save area points to any other traps that occurred at the same interrupt level. Thus, several traps may be nested at the same interrupt level. At the end of the execution of the trap-handler procedure, a return from trap (RTT) must be executed; this does a restoration of the partial context that was stored, unlike the trap save area, and returns all pointers to their original state.

The relationship of traps and interrupts, and their vector linkage are shown in Figure 3-6. The trap vector (TV) points to the trap-handler procedure. The trap save areas contain the following information:

- TSAL – Trap save area link
- I – Contents of indicator register
- R3 – Contents of register R3
- Instr – The instruction causing the trap
- Z – Miscellaneous information
- A – An address related to the trap condition (details depend upon trap type)
- P – The program counter; e.g., on a branch, the location to which the branch would go if it were not trapped
- B3 – The contents of register B3.

Note that the address of this area is noted in the Trap Save Area Pointer (TSAP) in the interrupt save area for the current level.

TABLE 3-2. TRAP VECTORS AND EVENTS

Vector #	Event
Vector #1	Monitor call (MCL instruction)
Vector #2	Trace[a] (debug) or BRK instruction
Vector #3	Scientific operation not in hardware
Vector #4	Reserved for software use
Vector #5	Other operation not in hardware (or undefined)
Vector #6	Integer register overflow[a]
Vector #7	Scientific divide by zero
Vector #8	Scientific exponent overflow
Vector #9	Stack underflow
Vector #10	Stack overflow
Vector #11	Reserved for future use
Vector #12	Reserved for future use
Vector #13	Unprivileged use of privileged operation
Vector #14	Unauthorized reference to protected memory (with optional protection)
Vector #15	Reference to unavailable resources
Vector #16	Program error
Vector #17	Memory or Bus error (parity or non-correctable ECC) detected
Vector #18	Reserved for future use
Vector #19	Scientific exponent underflow[a]
Vector #20	Scientific program error
Vector #21	Scientific significance error[a]
Vector #22	Scientific precision error[a]
Vector #23	Reference to unavailable resources by external processor
Vector #24	Memory or Bus error detected by external processor
Vector #25 through Vector #46	Reserved for future use

[a]If enabled.

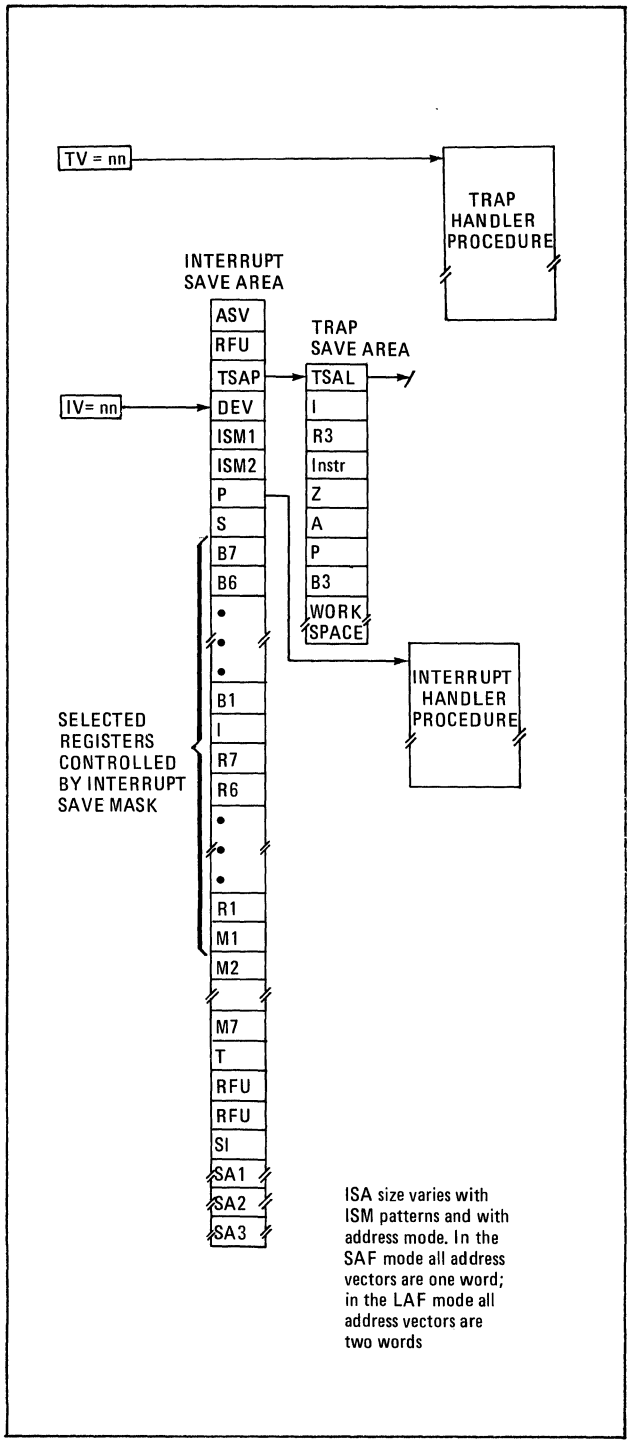


Figure 3-6. Trap Vector and Interrupt Vector Linkage

QUEUE MANAGEMENT

The 6/40 provides a queue capability that allows easy maintenance of ordered lists of "frames" (a frame contains a frame priority number, a next frame pointer, and an associated data structure). Each list is identified by a lock frame which contains a lock word and list head and tail pointers (Figure 3-7).

Four generic instructions are provided to enqueue/dequeue frames from the list. The instructions, which are described in Section 4, are as follows:

- o Queue on Head (QOH)
- o Queue on Tail (QOT)
- o Dequeue from Head (DQH)
- o Dequeue by Address (DQA)

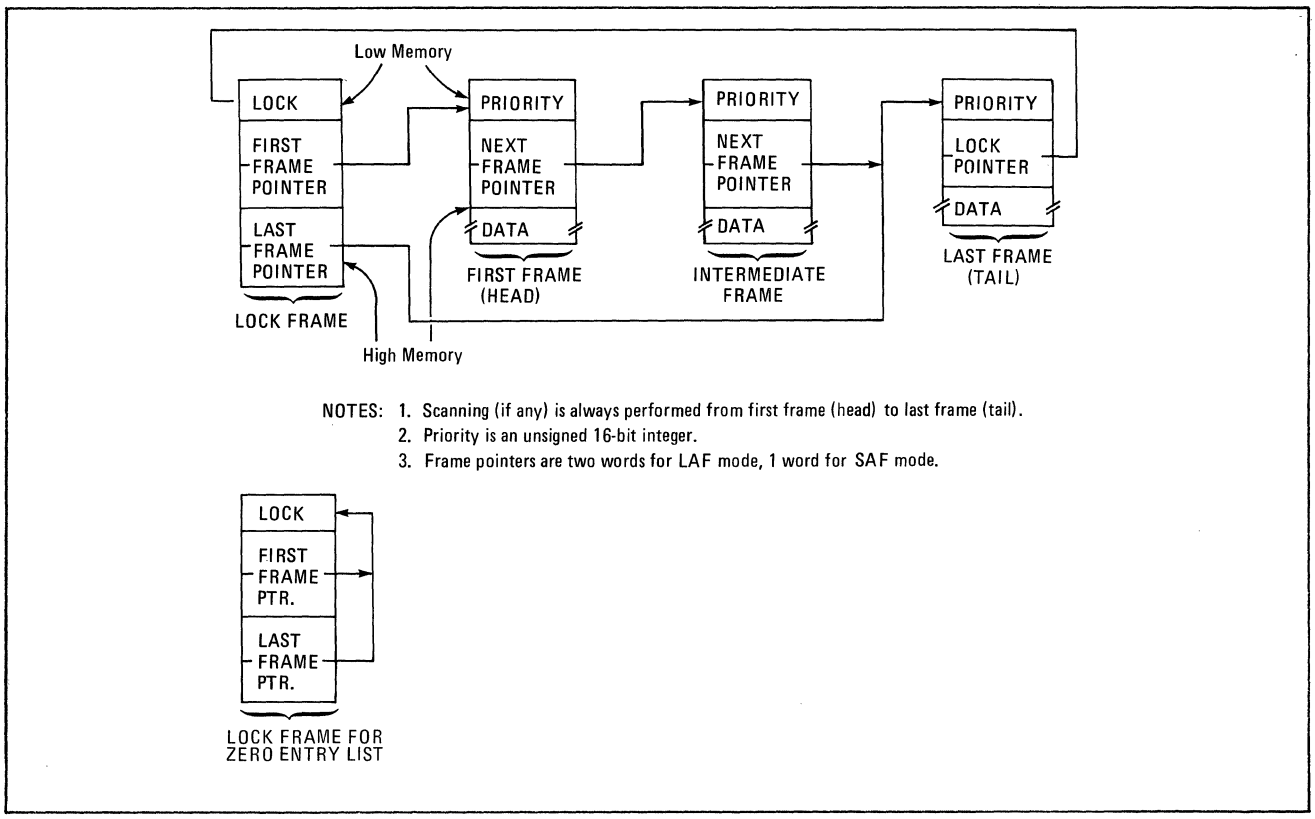


Figure 3-7. Queue Management

The lock word is used to ensure that only one CP is accessing a particular queue at a time. Each instruction causes a fetch of the lock word with a Read-Modify-Write (RMW) cycle. If the low-order bit of the lock is set, the RMW cycle is completed without changing lock, I(C) is cleared, and the next instruction is fetched. If the low-order bit of lock is cleared, the CP completes the RMW cycle, writing ones into the lock word, and initiates execution of the queue/dequeue instruction. Each queue/dequeue instruction causes a scan of the frames from the head toward the tail.

The scan continues until the conditions of the particular command are met (a hit), the last frame is reached without a hit, or an interrupt occurs.

When a hit occurs or if the last frame is reached without a hit, the frame is linked into or out of the list as appropriate, and I(G) and I(L) are left in a known state. In either case, the CP initiates another RMW cycle, writing zeros in lock, sets I(C), then fetches the next instruction. If an interrupt[1] occurs the CP stops the scan, initiates an RMW, writes zeros in lock, clears I(C), leaves I(G) and I(L) undefined and backup P to point to the queue/dequeue instruction, before servicing the interrupt.

During the scan, the effective ring number is moved outward if the frame being scanned is in a lower privilege ring. If an inward move from the effective ring occurs in going to a frame, MMU protection procedures apply.

Software must build the lock frame of each list to be used. A list with no entries is a lock frame in which the first and last frame pointers point to LOCK (see Figure 3-7). The CP leaves the lock frame in the same condition when a frame is unlinked from a single frame list.

STACK MANAGEMENT

The 6/40 provides a single stack capability for each interrupt level. The Stack Address Register (T) points to the first word of the stack header (Figure 3-8).

Four generic instructions are provided to manipulate the stack. The instructions, which are described in Section 4, are as follows:

- o Load Stack Address Register (LDT)
- o Store Stack Address Register (STT)
- o Acquire Stack Frame (ACQ)
- o Relinquish Stack Frame (RLQ)

[1] This includes service of the internal timer at a 120-Hz rate.

These are all two-word instructions having a common first word. Appropriate checks are made for stack overflow/underflow conditions.

The stack header contains four entries, two of which are not used by the 6/40, but must contain null pointers if upward software compatibility is desired.

MW is the number of words allocated to this stack. MW is written by software when the header is created and referenced (but not altered) by hardware.

CW represents the number of words currently consumed in this stack. CW is written by software when the header is created. Thereafter, the value of CW is a hardware responsibility.

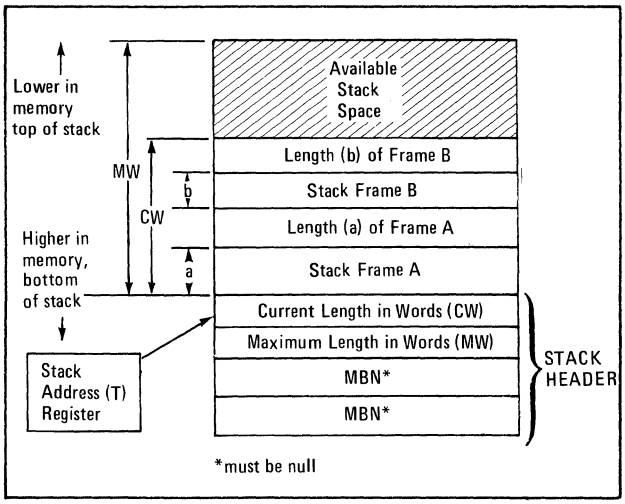


Figure 3-8. Stack Structure

SCIENTIFIC INSTRUCTION PROCESSOR

The Scientific Instruction Processor (SIP) is optionally (CPF9503) available on 6/40 models. The 30 scientific instructions operate on single-precision and double-precision (respectively, double-word and quadruple-word fields) with floating-point, integer and zero formats. These operands come from main memory, CP registers (following their conversion to two- or four-word floating-point quantities) and scientific accumulator (SA) registers. Prior to command execution involving two operands, the SIP tests for unequal length floating-point values. If the operand lengths are not equal, the destination scientific accumulator's length dominates.

Control Registers

There are three scientific control registers in the SIP. Two of these registers, the SIP Mode Register (M4) and the SIP Trap Mask Register (M5) reside in the associated CP with a copy of their contents stored in the SIP. The third register is the SIP Indicator Register, located in the SIP.

Accumulators

The SIP contains three variable length scientific accumulators (SA1, SA2, SA3) that may contain either single-precision (32-bit) or double-precision (64-bit) floating-point quantities.

Automatic Round/Truncate

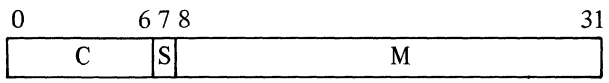
Scientific operations requiring right scaling, which is the process of shifting floating-point mantissa digits one hexadecimal position to the right, inserting a zero digit into the most significant mantissa digit position and increasing the exponent by one, may produce results where nonzero data is shifted off the end of the scientific accumulator into guard digit locations. These guard digits will be truncated from the result if bit zero of mode register (M4) is zero. If bit zero is a one, the result is rounded using the significant guard digits.

Scientific Traps

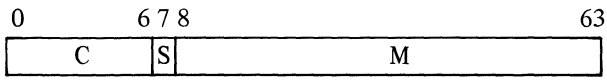
As the SIP is an option of the CP and functions as an extension of it, any SIP exception conditions are reported to the CP in the form of traps rather than interrupts. This trap facility is activated upon detection of specific status conditions during the execution of a scientific instruction. The SIP has eight types of trap vectors. Three of these trap vectors are a function of the data being worked on and are conditional, while two others are not conditional. Two trap vectors are a function of megabus activity; i.e., the detection of an unavailable resource, bus parity, or main memory error. The last identifies program error conditions detected in the SIP.

Data Formats

Floating-point data appears either as a single-precision (32-bit) or double-precision (64-bit) constant, as follows:



SINGLE-PRECISION

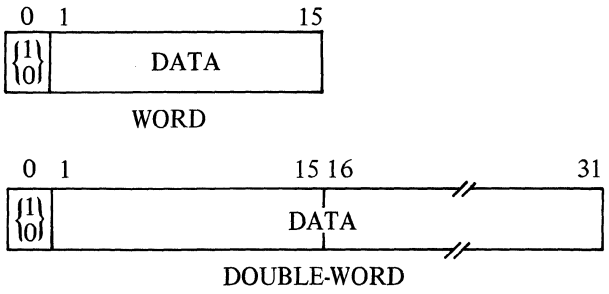


DOUBLE-PRECISION

where:

C = the characteristic (excess 64 power-of-16 exponent) of the number. The characteristic represents exponents with a range from -64 to +63. Since the characteristic has no sign bit, the number 64 (decimal) is effectively added to each exponent, thus allowing a characteristic range of 0 to 127 to represent

exponents with a range of -64 to +63.
S = sign bit (0 = +; 1 = -) of the mantissa.
M = magnitude of the mantissa.
Signed integer data contains a sign (0 = +;1 = -) in bit 0 and the data in the remaining bits. Negative numbers appear in twos-complement form. Word and couble-word formats are permitted, as follows:



Single-word integers must be in general registers R4, R5 or R6 of the CP in order to be processed by the SIP. Double-word integers must be in registers R6 and R7.

MEMORY MANAGEMENT UNIT

The Memory Management Unit (MMU) is optionally (CPF9501) available on 6/40 models. MMU functionality provides for the separation of memory into 16/31 independent segments, the relocation of each segment independently in physical memory, and the protection of each segment from improper access, based on software-specified attributes.

Segmentation

The MMU option divides the million-word address space of the 6/40 system into 16 regions of 65,536 words each, numbered 0 through 15. The first of these regions is subdivided into 16 regions of 4096 words each, numbered 0.00 through 0.15. Each of these regions may contain a segment of program address space. Each segment may range from 256 words in size up to the 4K or 64K size of its associated region, in steps of 256 words.

In Short Address Form (SAF), segments 0.00 through 0.15 are available, corresponding to the SAF address space of 64K. In Long Address Form (LAF), segments 1 through 15 are also available, yielding a total of 31 segments corresponding to the LAF address space of one million words. See Figure 3-9.

Relocation

Through 32-bit "segment descriptors," software can specify the location each segment is to occupy in main memory. This is given as a base (on a 256-word boundary) and a size (in multiples of 256 words). Each segment is relocated independently; segments need not be contiguous, be in order, or have any other particular physical relationship.

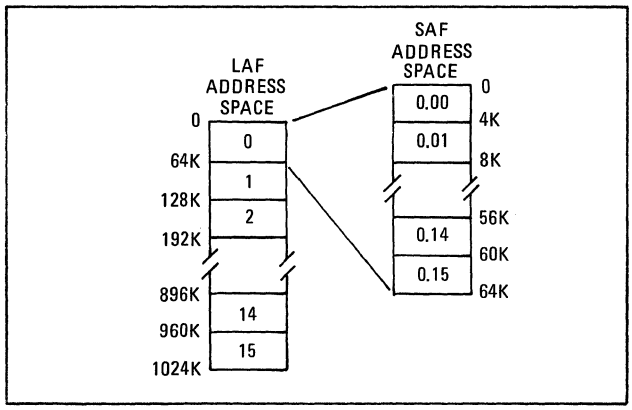


Figure 3-9. Memory Layout with Memory Management Unit Option

Protection

The MMU implements protection using the "ring" approach pioneered by Honeywell's Multics system. At any time, the processor is operating in one of four "rings" of privilege. Each ring, starting with Ring 0 and continuing through Ring 3, is more privileged than the next outer ring. Thus, Ring 0 can be used for the most critical system functions; Ring 1 for less critical system functions; Ring 2 to create a "user supervisor" or for well checked out user programs; and Ring 3 for the majority of user software. Programs operating in one ring cannot access code or data reserved for rings of higher privilege.

The MMU controls three forms of access: *read, write,* and *execute*. Each form of access is limited to programs operating in a certain ring or lower-numbered ring. Access control is implemented on a segment basis, so each segment has its own protection attributes. Access control can support many different objectives:

- *Data protection* – critical system control data can be placed in a user's address space, so that operating system routines executing on a user's behalf can access it while protecting it from user programs.
- *Source data integrity* – data to be read but not modified can be given read, but not write, access.
- *Software protection* – proprietary routines which a user is permitted to execute can be made inaccessible to "read" operations, thus preventing copying.

- *Program checkout* – by making data segments nonexecutable and code segments nonwriteable, many programming errors can be detected quickly.

An additional margin of protection is provided by the segment tables themselves. There is no way a program can attempt to access data not covered by that program's segment descriptors.

Segment Descriptor Format

Each segment is described by a 32-bit segment descriptor (see Figure 3-10).

During program execution, the segment descriptors are contained within the MMU hardware itself. At system initialization, the descriptors are given values which create a "transparent" mode of execution: each virtual address maps into the same real address, and all access modes are permitted in all four rings. Therefore, all programs execute on a system with the MMU exactly as they would on a system without it, until explicit software action changes the MMU tables. This can happen in one of two ways:

- When a level change takes place, if desired. Specifically, bit 0 of the second Interrupt Save Mask (ISM 2) in the Interrupt Save Area (ISA) of a level determines whether the MMU registers are to be loaded when that level is activated. If this bit is 1, the Address Saving Vector in the ISA indicates a 62-word area which contains 31 descriptor images. These images are loaded into the MMU before instruction execution starts at the new level.
- When an Activate Segment Descriptor instruction is executed. This instruction loads a single segment descriptor into the MMU.

Descriptor tables in memory are not referred to during program execution. Therefore, there is no performance degradation with the Memory Management Unit; programs using it operate at the same speed as programs which do not.

As instructions are executed, three types of addressing errors can occur. These are detected by the MMU and result in program traps. They are as follows:

1. Nonexistent segment – an attempt to reference a segment whose validity bit is reset. Causes Trap 15 (nonexistent resource).
2. Access out of bounds – an attempt to reference an address beyond the size established by the segment descriptor. Causes Trap 15 (nonexistent resource).
3. Protection violation – an attempt to perform a type of access not permitted by the appropriate permission field (RP, WP or EP) in the segment descriptor. Causes Trap 14 (protection violation).

Three instructions activate MMU functionality, over and above the relocation and protection functions performed on every memory access. They are:

- ASD (Activate Segment Descriptor)
- VLD (Validate)
- LEV (Level Change)

They are described in Section 4 of this handbook.

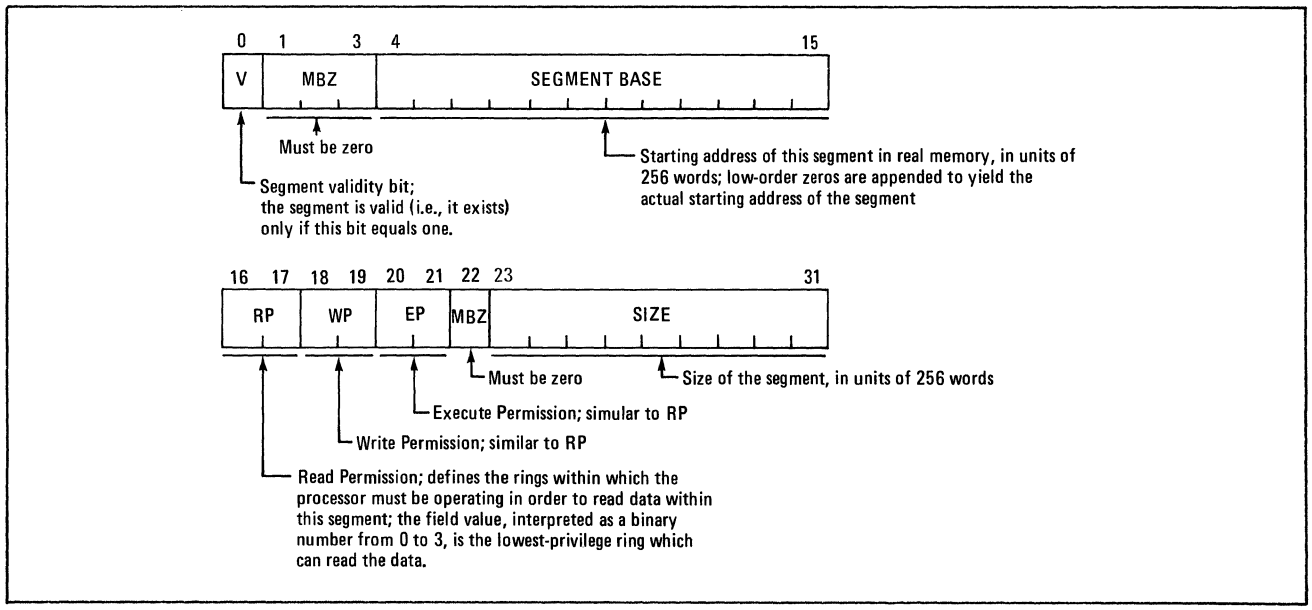


Figure 3-10. Segment Descriptor Format

SECTION 4

INSTRUCTIONS AND ADDRESSING

INSTRUCTION SET SUMMARY

The instruction set consists of over 100 instructions, grouped as follows:

- o Double Operand
- o Single Operand
- o Branch on Register
- o Branch on Indicator
- o Short Value Immediate
- o Shift
- o Generic
- o Input/Output
- o Scientific

These are listed below in Table 4-1 and summarized following the table. The instruction formats for each type and the addressing modes are detailed later in this section.

TABLE 4-1. INSTRUCTION SET SUMMARY

SINGLE OPERAND INSTRUCTIONS

Modify	Description
INC	Increment
DEC	Decrement
NEG	Negate
CPL	Complement
CL	Clear
CLH	Clear halfword
CMZ	Compare to zero
CMN	Compare to null
CAD	Add carry bit to contents

Control	Description
STS	Store S-register
JMP	Jump
ENT	Enter
LEV	Change level
SAVE	Save context
RSTR	Restore context

Bit	Description
LB	Load bit
LBF	Load bit and set false
LBT	Load bit and set true
LBC	Load bit and complement
LBS	Load bit and swap

Double Word	Description
AID	Add double integer (6/43 only)
LDI	Load double integer
SDI	Store double integer
SID	Subtract double integer (6/43 only)

DOUBLE OPERAND INSTRUCTIONS

Word	Description
LDR	Load R register
STR	Store R register
SRM	Store R register through mask
SWR	Swap R register
CMR	Compare contents to R register
ADD	Add contents to R register
SUB	Subtract from R register
MUL	Multiply R register
DIV	Divide R register by contents of location
OR	Inclusive OR with R register
XOR	Exclusive OR with R register
AND	AND contents with R register
AID	Double word add
SID	Double word subtract

Byte	Description
LDH	Load halfword into R register
STH	Store R register halfword
CMH	Compare halfword to R register
ORH	Halfword inclusive OR with R register
XOH	Half-word exclusive OR with R register
ANH	Logically AND halfword with R register
LLH	Load logical halfword into R register

Mode and Base Register	Description
MTM	Modify or test M register

TABLE 4-1 (CONT). INSTRUCTION SET SUMMARY

Mode and Base Register	Description
STM	Store M register
LDB	Load B register
STB	Store B register
CMB	Compare contents to B register
CMN	Compare to null
SWB	Swap B register
LAB	Load effective address into B register
LNJ	Load B register and jump

SHORT VALUE IMMEDIATE INSTRUCTIONS

Instruction	Description
LDV	Load value
CMV	Compare value to R register
ADV	Add value to R register
MLV	Multiply by value

BRANCH INSTRUCTIONS

Branch on Register	Description
BLZ	Branch if R register less than zero
BGEZ	Branch if R register equal to or less than zero
BEZ	Branch if R register equal to zero
BNEZ	Branch if R register not equal to zero
BGZ	Branch if R register greater than zero
BLEZ	Branch if R register equal to or less than zero
BODD	Branch if R register odd
BEVN	Branch if R register even
BINC	Branch and increment
BDEC	Branch and decrement

Branch on Indicator	Description
B	Branch
NOP	No operation
BE	Branch if equal
BNE	Branch if not equal
BAL	Branch if algebraically less than
BAGE	Branch if algebraically greater than or equal to
BAG	Branch if algebraically greater than
BALE	Branch if algebraically less than or equal to
BL	Branch if less than
BGE	Branch if greater than or equal to
BG	Branch if greater than
BLE	Branch if less than or equal to
BSU	Branch if signs unlike
BSE	Branch if signs equal
BCT	Branch if carry
BCF	Branch if no carry
BBT	Branch if bit test indicator true
BBF	Branch if bit test indicator false
BIOT	Branch if I/O indicator true
BIOF	Branch if I/O indicator false
BOV	Branch if R register overflow
BNOV	Branch if no R register overflow

SHIFT INSTRUCTIONS

Shift Short	Description
SOL	Single shift open left
SCL	Single shift closed left
SAL	Single shift arithmetic left
DCL	Double shift closed left
SOR	Single shift open right
SCR	Single shift closed right
SAR	Single shift arithmetic right
DCR	Double shift closed right

Shift Long	Description
DOL	Double shift open left
DAL	Double shift arithmetic left
DOR	Double shift open right
DAR	Double shift arithmetic right

INPUT/OUTPUT INSTRUCTIONS

Instruction	Description
IO	Input/output word
IOH	Input/output halfword
IOLD	Input/output load

GENERIC INSTRUCTIONS

Instruction	Description
HLT	Halt
MCL	Call monitor via trap

TABLE 4-1 (CONT). INSTRUCTION SET SUMMARY

Instruction	Description
RTT	Return from trap
RTCN	Real-time clock on
RTCF	Real-time clock off
WDTN	Watchdog timer on[a]
WDTF	Watchdog timer off[a]
BRK	Break trap
MMM	Memory to memory move (6/43 only)
ASD	Activate segment descriptor (6/43 only)
VLD	Validate access rights (6/43 only)
QOH	Queue on head (6/43 only)
QOT	Queue on tail (6/43 only)
DQH	Dequeue from head (6/43 only)
DQA	Dequeue on address (6/43 only)
RSC	Rescan Configuration (6/43 only)
RLQ	Relinquish stack space (6/43 only)
LDT	Load T Register (6/43 only)
ACQ	Acquire stack space (6/43 only)
STT	Store T Register (6/43 only)

SCIENTIFIC INSTRUCTIONS

Single Operand	Description
SCZD	Scientific compare to zero two words
SCZQ	Scientific compare to zero four words
SNGD	Scientific negate two words
SNGQ	Scientific negate four words

Double Operand	Description
SLD	Scientific load
SST	Scientific store
SCM	Scientific compare
SAD	Scientific add
SSB	Scientific subtract
SML	Scientific multiply
SDV	Scientific divide
SSW	Scientific swap

Scientific Accumulators Branches	Description
SBLZ	Branch if SA less than zero
SBGEZ	Branch if SA greater than or equal to zero
SBEZ	Branch if SA equal to zero
SBNEZ	Branch if SA not equal to zero
SBGZ	Branch if SA greater than zero
SBLEZ	Branch if SA less than or equal to zero

Scientific Indicators Branches	Description
SBL	Branch if less than
SBGE	Branch if greater than or equal
SBE	Branch if equal
SBNE	Branch if not equal
SBG	Branch if greater than
SBLE	Branch if less than or equal
SBPE	Branch if precision error
SBNPE	Branch if no precision error
SBSE	Branch if significance error
SBNSE	Branch if no significance error
SBEU	Branch if exponent underflow
SBNEU	Branch if no exponent underflow

[a]Not available on 6/34.

1. *Double Operand* instructions are memory reference instructions in which the first operand is a register address and the second operand is usually a memory address, although for register-to-register instructions the second address can also specify a register. A typical double operand instruction is an (ADD) instruction, which adds the contents of the addressed memory location (or register) to the general (R) register specified by the first operand. Thus, the instruction ADD $R1, LOC adds the contents of memory location LOC to register R1.[1]

2. *Single Operand* instructions can address memory (or a register) in the same way as double operand instructions, but they do

[1] Instruction examples will be given in Assembly Notation. For details, see the *Level 6 Assembly Language Manual,* Order No. AS31.

not need a register address. A typical single operand instruction is the Clear (CL) instruction, which clears the addressed memory location to zero. In assembly notation, this instruction could be written:

CL LOC

3. *Branch on Register* instructions are similar to double operand instructions in that they must specify a general register, R1 through R7, and also a memory address to which control will be transferred if the tested condition is true. A typical branch on register instruction is Branch if Register Odd (BODD), which might be written:

   BODD $R6, LOC

   This would test register 6 to see whether it were even or odd, and if it were odd the program would branch to location LOC.
4. *Branch on Indicator* instructions are similar to branch on register instructions, but the op code specifies an indicator and no register address is required. A typical instruction is Branch if Greater than (BG), which will branch if the G (greater than) indicator is set. This will be written:

   BG LOC
5. *Short Value Immediate* instructions do not reference memory, but specify a register and an 8-bit immediate operand which is contained in the instruction itself. For example, if it were desired to add the quantity 2 to register R3, the Add Value (ADV) instruction could be used. This would be written:

   ADV $R3, =2
6. *Shift* instructions are used to shift either single general registers or pairs of general registers. The first operand specifies the register itself or, in the case of a double word shift, the right-hand (odd) register of a pair. The second operand usually specifies the number of positions to be shifted. A typical shift instruction is Shift Closed Left (SCL), which rotates the contents of a register "n" positions to the left. For example, to rotate R6 four places to the left, the following instruction would be used:

   SCL $R6, 4

   To rotate both 6 and 7 together, a Double Closed Left would be utilized:

   DCL $R7, 4
7. *Generic* instructions have no variable addresses and need only an op code. Typically, these are control instructions. A typical instruction in this group is Monitor Call (MCL), which generates an automatic trap via vector # 1.
8. *Input/Output* instructions enable the processor to communicate directly with input/output channels by sending the channel either an output command or an input command request (see section 2). A typical I/O command is the I/O Load (IOLD) instruction, which sends *both* an address and a range to the addressed channel. Thus, this instruction has three operands and could be written:

   IOLD ADDR, CHAN, RANGE

   This instruction in machine language, depending upon the address form used, could occupy from 3 to 6 words of memory.
9. *Scientific* instructions are all executed by the SIP when it is configured (optional on 6/40 models; not available on 6/30 models). If the SIP is not configured (or offered), then the scientific instructions are trapped and emulated by software (assuming that the SIP software simulator is configured).

SAF and LAF Mode Impact on Instructions

The operation mode of the CP impacts instructions in two ways.

- Instruction size – this is a factor whenever an instruction specifies an IMA form of addressing. In SAF mode the instruction consists of two words while in LAF mode the instruction consists of three words. Consequently using the wrong instruction size not only results in the erroneous execution of the instruction but also results in mispositioning of the program counter.
- Instruction Execution – this relates to those instructions which operate on base registers or address information since addresses are 16 bits in SAF mode and 20 bits in LAF mode. Consequently when either loading or storing a B register or address information, the correct size storage is required, the following instructions operate on registers or address information: LDB, STB, CMB, SWB, LAB, LNJ, SAVE, and RSTR.

SAF/LAF Independent Code (SLIC)

Two techniques are available to achieve SAF/LAF independence:

- *SAF/LAF Independence by Reassembly* A program must be reassembled for the

addressing mode in which it will execute. Rules required to achieve this are provided in the *GCOS 6/MDT Overview and User's Guide,* Order Number AX11. Refer specifically to "Assembly Language Address-Form Independence" in Appendix A.

- *SAF/LAF Independence at Loading* A program is modified at the time it is loaded for the addressing mode in which it will execute. Detailed rules for writing software in this fashion are described in the *GCOS 6/MDT Assembly Language* manual, Order No. AX12.

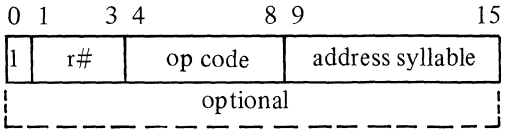
Pre-fetch Capability and Self-Modifying Code

The 6/40 models have a pre-fetch or "look ahead" capability in which two words following the current instruction are pre-fetched to achieve greater processing speed. Therefore, programmers should avoid using an instruction modifying another that closely follows it, since the modification would take place in the memory location from which the instruction has already been pre-fetched.

INSTRUCTION FORMATS AND ADDRESSING MODES

Single and Double Operand Instructions

The format for single and double operand instructions is as follows:



Single Operand Instruction: # bits 1–3 will be zero

Double Operand Instruction: # bits 1–3 will specify a register number from 1–7.

The significance of the bits is as follows: bit 0 is always a 1; bits 1, 2 and 3 are 0 for single operand instructions and define a register number (1–7) in double operand instructions (the op code defines whether this is one of the 7 general (R) registers or one of the 7 address (B)registers); bits 4 to 8 define the operation code; bits 9–15 are the Address Syllable and are used to define either:

- a *location in memory* that contains an operand
- a *register* that contains an operand
- an *immediate operand,* where the operand is contained in the second word of the instruction.

Single and double operand instructions can be either one word or two words in length depending on the addressing mode utilized. Table 4-2 summarizes the addressing modes.

TABLE 4-2. SUMMARY OF ADDRESSING MODES FOR SINGLE AND DOUBLE OPERAND INSTRUCTIONS

Operand Location	Types of Addressing	Instruction Length (Words)[a]	Assembly Example Using ADD Command
Register	Register Addressing	1	ADD $R6, =$R5
Instruction	Immediate Operand	2[b]	ADD $R6, =1000
Memory	Absolute (Immediate Address)	2 (3 LAF mode)	
	• Direct		ADD $R6, <LOC
	• Indirect		ADD $R6, *<LOC
	• Indexed		ADD $R6, <LOC. $R3
	• Indirect Indexed		ADD $R6, *<LOC. $R3
	Base Addressing	1	
	• Direct		ADD $R6, $B7
	• Indirect		ADD $R6, *$B7
	• Indexed		ADD $R6, $B7. $R3
	• Indirect Indexed		ADD $R6, * $B7.$R3
	• Pre-Decrement		ADD $R6, –$B7
	• Post-Increment		ADD $R6, +$B7
	• Auto-Indexed, Pre-Decrement		ADD $R6, $B3.–$R3
	• Auto-Indexed, Post-Increment		ADD $R6, $B3. +$R3

TABLE 4-2 (CONT). SUMMARY OF ADDRESSING MODES FOR SINGLE AND DOUBLE OPERAND INSTRUCTIONS

Operand Location	Types of Addressing	Instruction Length (Words)[a]	Assembly Example Using ADD Command
	Relative Addressing	2	
	• P-Relative Direct		ADD $R6, LOC
	• P-Relative Indirect		ADD $R6, *LOC
	• Base Relative, Direct		ADD $R6, $B7.–5
	• Base Relative, Indirect		ADD $R6, *$B7.7
	• Interrupt Vector Relative		ADD $R6, $IV.7

[a]Add additional word for mask when required.

[b]Three for LDI, SDI, or Scientific.

NOTE: The major breakdowns are register addressing, memory addressing and immediate addressing. An assembly language example for an add instruction is shown next to each mode. For further information on instruction addressing in assembly language, see the referenced assembly language manual for Level 6 systems.

The addressing mode is defined by the address syllable. Instructions that address a register are one word in length. Instructions that utilize an immediate operand are considered to be two words in length, including the operand, since the program counter is incremented by two in order to access the next instruction. And finally, instructions that address operands in memory can be either one or two (two or three for those requiring a mask) words in length depending on the addressing mode used; these are as follows.

Absolute addressing – (also called immediate address mode). In SAF mode, a two-word instruction is used, with the second word containing a 16-bit word absolute address that describes a location from 0 to 64K. In LAF mode, a three-word instruction is used, with the last two words containing a 20-bit word absolute address that describes a location from 0 to 1M. This address can be:

- o a direct address
- o an indirect address
- o a direct address indexed by the contents of R1, R2, or R3
- o an indirect address that is post-indexed by the contents of R1, R2, or R3

Base addressing – one-word instructions that define one of the seven address registers (B1–B7) as containing the address of the operand. The address in the register can be:

- o a direct address
- o an indexed address
- o an indirect address
- o an indirect address post-indexed

In addition, some extremely powerful additional modes of base addressing are provided. These are still all one-word instructions:

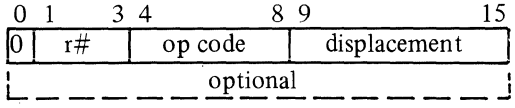
- o Base, pre-decremented (also called push addressing). In this mode one is subtracted from the contents of an address register prior to its being used as an address – unless it is a double-word load or store, in which case two is subtracted.
- o Base, post-increment (also called pop addressing). Here one (or two, as above) is added to the contents of an address register after it has been used as an address.
- o Base, auto-indexed. Here the contents of an index register R1, R2, or R3 are either pre-decremented (push indexed) or post-incremented (pop indexed) before/after being added to the contents of an address register B1, B2, or B3.

Relative addressing – two-word instructions where the second word contains an algebraic

displacement (±32K) relative to either the program counter (P relative), an address register (Base relative), or the interrupt vector for the current central processor level (IV relative). The resultant address can be utilized as either a direct or an indirect address (except for IV relative, which is direct only). This does not change in LAF mode (i.e., the 16-bit displacement is still used).

Branch Instructions

There are two types of branch instructions: branch on register and branch on indicator. The format is as follows:



A branch on indicator instruction has zeros in the register field, bits 1 through 3, while a branch on register instruction will have the number (from 1 to 7) of one of the general (R) registers.

Table 4-3 shows the three types of addressing modes that can be utilized with branch instructions together with the assembler mnemonics for each.

TABLE 4-3. BRANCH INSTRUCTION ADDRESSING FORMS (BG INSTRUCTION SHOWN)

Short Displacement	1 Word	BG >LOC
Long Displacement (P-Relative)	2 Words	BG LOC
Absolute (Immediate Address)	2 Words	BG <LOC

These instructions again can be either single- or double-word instructions. Three addressing modes are possible with branch instructions: displacement, relative, and absolute.

Displacement Addressing

In this mode a displacement is contained within a one-word instruction. The displacement is a 7-bit algebraic quantity that is applied to the contents of the program counter. Utilizing this mode of addressing, the program can branch to 64 locations prior to the instruction or 63 locations after it. Displacements of zero and one are not allowed.

Relative Addressing

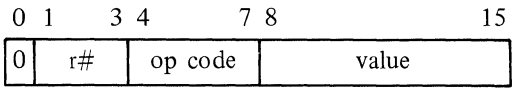
This mode of addressing is identical to the P-relative addressing mode in single and double operand instructions. The second word of the instruction contains a signed, 16-bit value (±32K) displacement from that word.

Absolute Addressing (Immediate Address)

This is also identical to single and double operand instructions. In SAF mode, a two-word instruction is used, with the second word containing an absolute 16-bit word address that describes a location from 0 to 64K. In LAF mode, a three-word instruction is used, with the last two words containing an absolute 20-bit word address that describes a location from 0 to 1M.

Short Value Immediate Instructions

The format for these instructions is as follows:

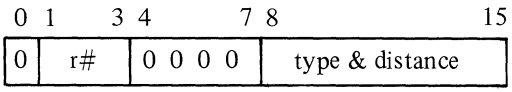


Bits 1–3 must specify a general (R) register number. Bits 8–15 contain an arithmetic value between -128 and +127. This value (with its sign extended) is used as an operand by the instructions that utilize this short value immediate addressing form.

Short Value Immediate	1 Word	ADV $R6,=6

Shift Instructions

Shift instructions have the following format:



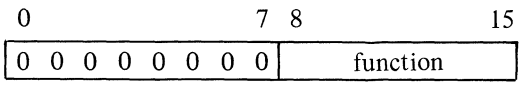
Bits 8–15 are used to specify the type, direction and number of bits to be shifted. If the number of bits field is zero, register R1 will contain the shift distances. Short shifts can specify a distance of up to 15 bits; long shifts, up to 31 bits. Bits 1–3 specify a general (R) register number. If a double shift is to be executed, this field must address the right-hand (odd) register shown below.

Short shift	1 Word	SAL $R5, 6
Long shift	1 Word	DAL $R5, 26

Generic Instructions

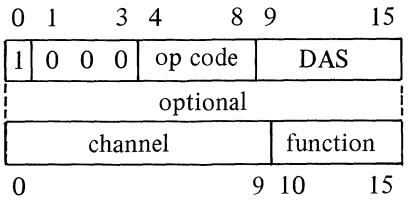
Generic instructions have the following format:

Bits 0–7 must be zeros, while bits 8–15 specify the function.

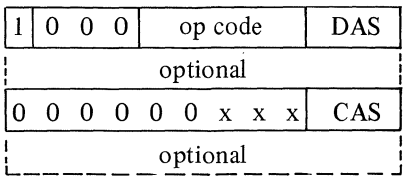


Input/Output Instructions

There are two types of input/output instructions. The first type is used by the input/output word (IO) or by the input/output half-word (IOH) instructions. This is the instruction that is used to place an I/O command on the level 6 Megabus (see Section 2). An I/O Command consists of a channel number, a function on the address bus, and a 16-bit data word on the data bus. The instruction format to do this is as follows:

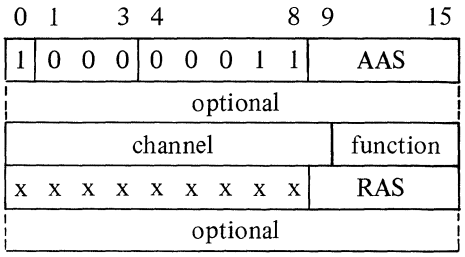


The address of the data word is defined by the Data Address Syllable (DAS) in the least significant 7 bits of the instruction and by a second word if needed. The addressing forms are the same as single word operand addressing and the second word will be needed for absolute addressing, relative addressing, or base plus displacement addressing modes. The last word of the instruction contains the channel address and the function code. If it is desired not to embed the channel number and the function code in the procedure, then the instruction can take the following format:

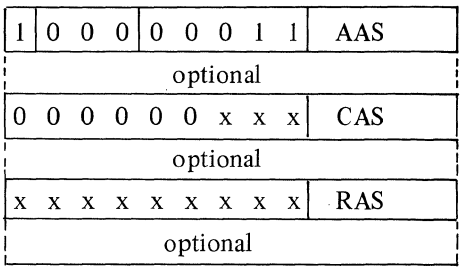


In this case the Channel Address Syllable (CAS) bits point to the location of a word containing the channel and function. Again, a second word may be required to define this address.

The second type of I/O instruction is the IOLD instruction. This is similar except that instead of placing one word of data on the I/O bus it places two words of data. These two words are specifically the address and range that are required to set up a DMA transfer. The format is the same as for the I/O instructions above, except that a third address must be specified. Again, this can be one or two words, depending upon the addressing mode utilized. The two cases are thus as follows, with the first embedding the control in the procedure and the second having the control word nonprocedural:



AAS – address syllable defining buffer address
RAS – address syllable defining location of range



CAS – address syllable defining location of word containing channel and function

Scientific Instructions

Scientific instructions do not have unique formats; they take the formats of either single or double operand instructions. They are not executed by hardware on the 6/30 models, but rather cause traps to unique software routines which execute the instructions. On 6/40 models, an optional Scientific Instruction Processor (SIP) is offered. Two trap handlers, the Floating-Point Simulator, entered via trap vector #3, and the Scientific Branch Simulator, entered via trap vector #5, are available and are described in the *GCOS/BES1/2 Executive Modules I/O Manual.*

INSTRUCTION SET

The instructions that are implemented by hardware are described in detail on the pages that follow.

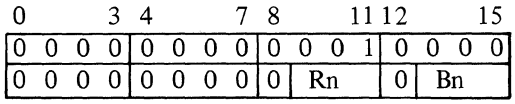
ACQ

Instruction:
Acquire stack space

Restriction:
Operates on 6/43 only.

Type:
Generic

Format:



Description:
Acquires a portion of the remaining available stack space. The amount of space in words to be acquired is specified by Rn, and Bn is loaded with the leftmost address of the newly acquired space. CW is updated to the new stack length.

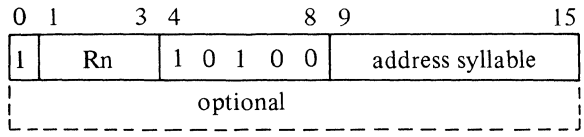
The contents of the I Register are unaffected. Trap vector #10 will occur if MW would be exceeded by the ACQ.

ADD

Instruction:
Add to R register

Type:
Double Operand

Format:



Description:
Adds the word at the effective address to the word contained in the designated R register. The carry (C) and overflow (OV) indicators will be set if carry and/or overflow occurs respectively; otherwise they will be reset to zero. The address syllable can specify a memory location, an immediate operand, or another R register.

Some examples of addition are:

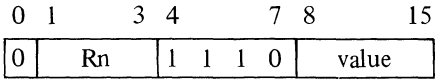
R Register (Before)	Effective Address (Before & After)	R Register (After)	C (After)	OV (After)
$+(32,766)_{10}$	+1	+32,767	0	0
$+(32,767)_{10}$	+1	-32,768	0	1
+1	-2	-1	0	0
-1	+2	+1	1	0
$(FFFF)_{16}$	+2	0001	1	0

ADV

Instruction:
Add value to R register

Type:
Short Value Immediate

Format:



Description:
Adds the value contained in the instruction (with the sign extended) to the designated R register. Overflow (OV) and carry (C) indicators are set/reset according to the results of the addition.

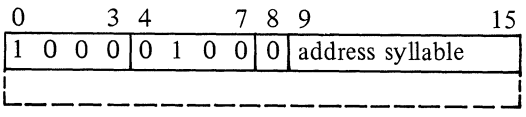
AID

Instruction:
Add double-word integer

Restriction:
Operates on 6/43 only.

Type:
Single Operand

Format:



Description:
Loads the sum of the double-word integer contained in R6 and R7 and the contents of the double-word location specified by the address syllable into R6 and R7.

R6(0) is considered the high-order bit.

R7(15) is considered the low-order bit.

The contents of the I Register are affected as follows:

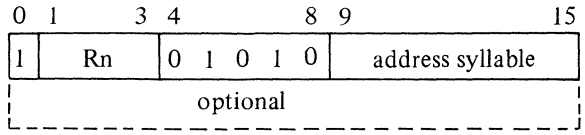
- o If carry, C is set to 1; otherwise 0.
- o If overflow, OV is set to 1; otherwise 0.

AND

Instruction:
AND with R register

Type:
Double Operand

Format:



Description:
Logically ANDs the word at the effective address to the word contained in the designated R register. No indicators are affected. The address syllable can specify a memory address, an immediate operand, or another R register.

The following chart illustrates the result of logically ANDing bits:

First operand bit	0	0	1	1
Second operand bit	1	0	1	0
Result	0	0	1	0

Examples:

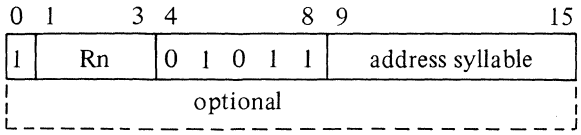
R Register (Before)	Effective Address (Before & After)	R Register (After)
$(ABCD)_{16}$	$(00FF)_{16}$	$(00CD)_{16}$
$(ABCD)_{16}$	$(7777)_{16}$	$(2345)_{16}$

ANH

Instruction:
Logically AND half-word (byte) with R register

Type:
Double Operand

Format:



Description:
Logically ANDs the byte at the effective address (with its sign extended) to the word contained in the designated R register. No indicators are affected. The address syllable can specify a memory address, an immediate operand, or another R register, with the byte being positioned as follows:

– memory location, not indexed	– left byte
– memory location, indexed	
index value even	– left byte
index value odd	– right byte
– immediate operand	– left byte
– R register	– right byte

The following chart illustrates the result of logically ANDing bits:

First operand bit	0	0	1	1
Second operand bit	1	0	1	0
Result	0	0	1	0

Examples:

R Register (Before)	Effective Address (Before & After)	R Register (After)
$(ABCD)_{16}$	$(FF)_{16}$	$(ABCD)_{16}$
$(ABCD)_{16}$	$(77)_{16}$	$(0045)_{16}$

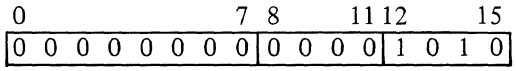
ASD

Instruction:
Activate segment descriptor

Restriction:
Operates only on 6/43 with Memory Management Unit option.

Type:
Generic

Format:



Description:

Other than via a level change, this privileged instruction is the only way MMU segment descriptor tables can be changed.

The ASD instruction loads one segment descriptor into the MMU. Base register B5 specifies the segment, and the descriptor is contained in general registers R6 and R7. The new segment descriptor is effective immediately.

In using this instruction, the programmer must remember that the memory tables associated with each level are not updated by an ASD or by a level change. Therefore, if the new descriptor is to remain valid after an interrupt, it must also be loaded into the MMU image associated with the current level. Normally, to avoid anomalies resulting from interrupts between the two loads, the memory image should be loaded first, and it should be loaded by an (uninterruptible) SDI instruction.

The contents of the I-Register are unaffected. If S(R) $\neq$ 1x, then trap 13 will occur.

B

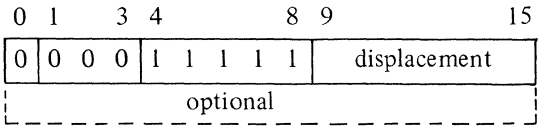
Instruction:

Branch unconditionally

Type:

Branch on Indicator

Format:



Description:

Branches to the specified location unconditionally unless the J bit in the mode control (M1) register is set, in which case it traps to trap vector #2.

BAG

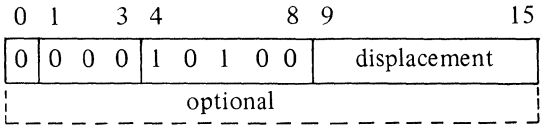
Instruction:

Branch if algebraically greater than

Type:

Branch on Indicator

Format:



Description:

Branches to the specified location if either the greater than indicator (G) or the unlike signs indicator (U), but not both, is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BAGE

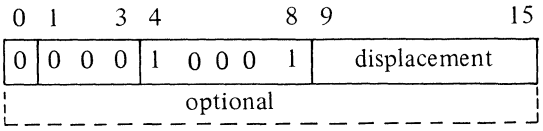
Instruction:

Branch if algebraically greater than or equal to

Type:

Branch on Indicator

Format:



Description:

Branches to the specified location if both the less than indicator (L) and the unlike signs indicator (U), or neither is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BAL

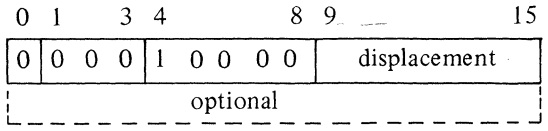
Instruction:

Branch if algebraically less than

Type:

Branch on Indicator

Format:



Description:

Branches to the specified location if either the less than indicator (L) or the unlike signs indicator (U), but not both, is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BALE

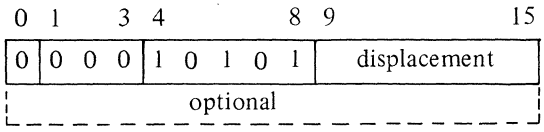
Instruction:

Branch if algebraically less than or equal to

Type:

Branch on Indicator

Format:



Description:

Branches to the specified location if both the greater than indicator (G) and the unlike signs indicator (U), or neither is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BBF

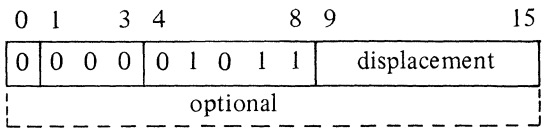
Instruction:

Branch if bit-test indicator false

Type:

Branch on Indicator

Format:



Description:

Branches to the specified location if the bit-test indicator (B) is equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BBT

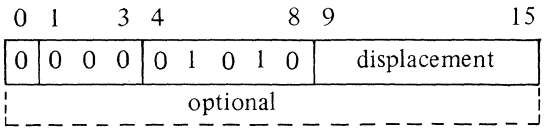
Instruction:

Branch if bit-test indicator true

Type:

Branch on Indicator

Format:



Description:

Branches to the specified location if the bit-test indicator (B) is equal to one. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BCF

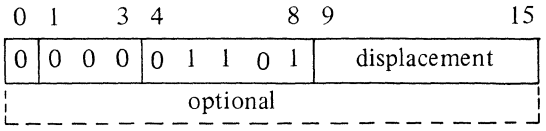
Instruction:

Branch if no carry

Type:

Branch on Indicator

Format:



Description:

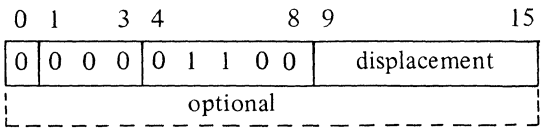
Branches to the specified location if the carry indicator (C) is equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BCT

Instruction:
Branch if carry

Type:
Branch on Indicator

Format:



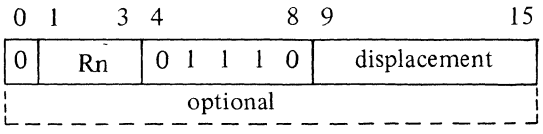
Description:
Branches to the specified location if the carry indicator (C) is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BDEC

Instruction:
Branch and decrement

Type:
Branch on Register

Format:



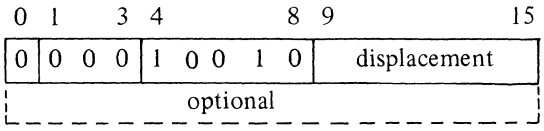
Description:
Subtracts one from the contents of the designated R register and then branches to the specified location if the result is not equal to -1 (i.e., branches unless the register initially contained zero). If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BE

Instruction:
Branch if equal

Type:
Branch on Indicator

Format:



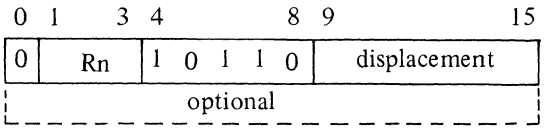
Description:
Branches to the specified location if neither the greater than indicator (G) nor the less than indicator (L) is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BEVN

Instruction:
Branch if R register even

Type:
Branch on Register

Format:



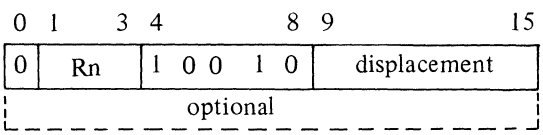
Description:
Branches to the specified location if bit 15 of the designated R register is equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BEZ

Instruction:
Branch if R register equal to 0

Type:
Branch on Register

Format:



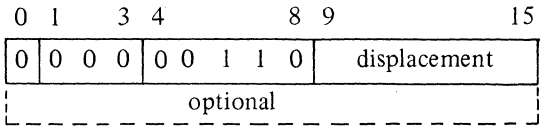
Description:
Branches to the specified location if the contents of the designated R register are equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BG

Instruction:
Branch if greater than

Type:
Branch on Indicator

Format:



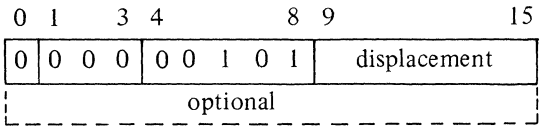
Description:
Branches to the specified location if the greater than indicator (G) is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BGE

Instruction:
Branch if greater than or equal to

Type:
Branch on Indicator

Format:



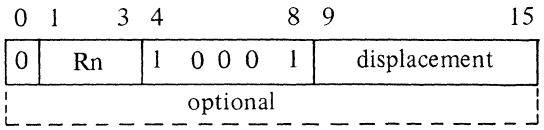
Description:
Branches to the specified location if the less than indicator (L) is equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BGEZ

Instruction:
Branch if R register equal to or greater than 0

Type:
Branch on Register

Format:



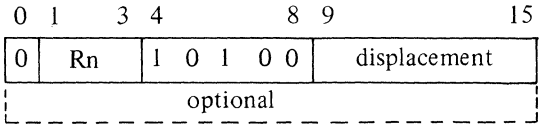
Description:
Branches to the specified location if bit zero of the designated R register is a zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BGZ

Instruction:
Branch if R register greater than 0

Type:
Branch on Register

Format:



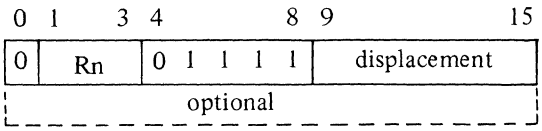
Description:
Branches to the specified location if bit zero of the designated R register is equal to zero and bits 1–15 are not equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BINC

Instruction:
Branch and increment

Type:
Branch on Register

Format:



Description:

Adds one to the contents of the designated R register and then branches to the specified location if the result is not equal to zero (i.e., branches unless the register initially contains -1). If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BIOF

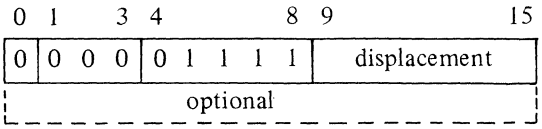
Instruction:

Branch if I/O indicator false

Type:

Branch on Indicator

Format:



Description:

Branches to the specified location if the I/O indicator (I) is equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BIOT

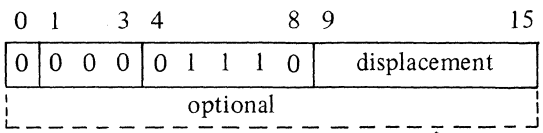
Instruction:

Branch if I/O indicator true

Type:

Branch on Indicator

Format:



Description:

Branches to the specified location if the I/O indicator (I) is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BL

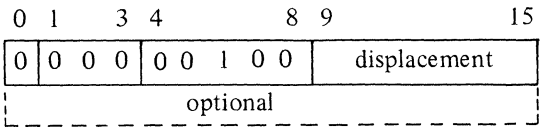
Instruction:

Branch if less than

Type:

Branch on Indicator

Format:



Description:

Branches to the specified location if the less than indicator (L) is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BLE

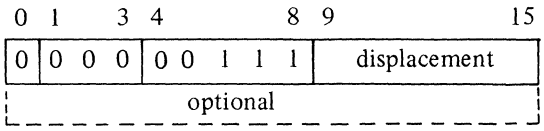
Instruction:

Branch if less than or equal to

Type:

Branch on Indicator

Format:



Description:

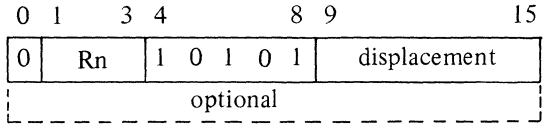
Branches to the specified location if the greater than indicator (G) is equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BLEZ

Instruction:
Branch if R register equal to or less than 0

Type:
Branch on Register

Format:



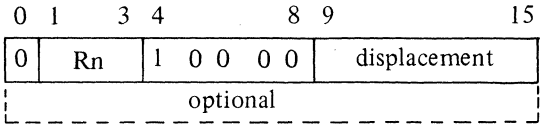
Description:
Branches to the specified location if bit zero of the designated R register is equal to one, or if the contents are equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BLZ

Instruction:
Branch if R register less than 0

Type:
Branch on Register

Format:



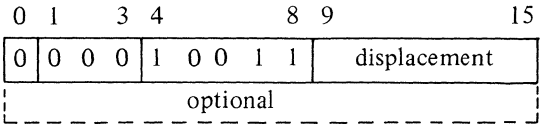
Description:
Branches to the specified location if bit zero of the designated R register is a one. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BNE

Instruction:
Branch if not equal

Type:
Branch on Indicator

Format:



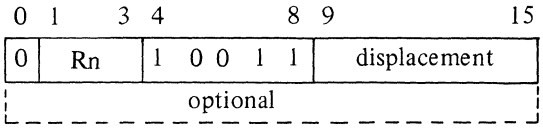
Description:
Branches to the specified location if either the greater than indicator (G) or the less than indicator (L) is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BNEZ

Instruction:
Branch if R register not equal to 0

Type:
Branch on Register

Format:



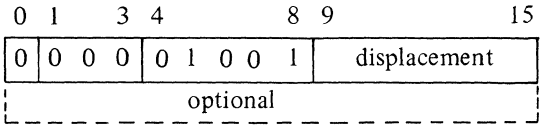
Description:
Branches to the specified location if the contents of the designated R register are not equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BNOV

Instruction:
Branch if no overflow

Type:
Branch on Indicator

Format:



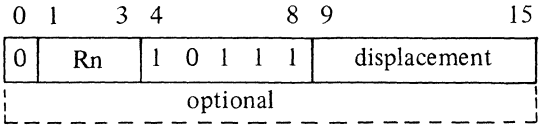
Description:
Branches to the specified location if the overflow indicator (OV) is equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BODD

Instruction:
Branch if R register odd

Type:
Branch on Register

Format:



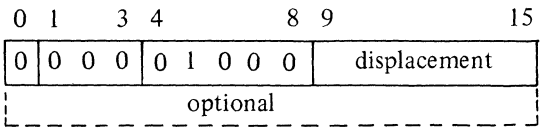
Description:
Branches to the specified location if bit 15 of the designated R register is equal to one. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BOV

Instruction:
Branch if overflow

Type:
Branch on Indicator

Format:



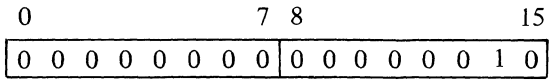
Description:
Branches to the specified location if the overflow indicator (OV) is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BRK

Instruction:
Breakpoint

Type:
Generic

Format:



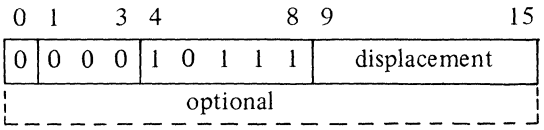
Description:
Causes a trap to trap vector #2; this instruction is used for debugging.

BSE

Instruction:
Branch if signs equal

Type:
Branch on Indicator

Format:



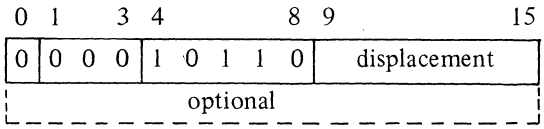
Description:
Branches to the specified location if the unlike signs indicator (U) is equal to zero. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

BSU

Instruction:
Branch if signs unlike

Type:
Branch on Indicator

Format:



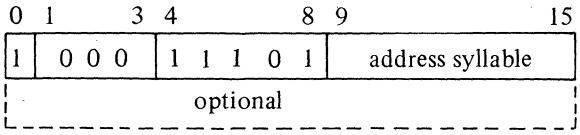
Description:
Branches to the specified location if the unlike signs indicator (U) is set. If the J bit in the mode control (M1) register is set, and if the branch would have taken place, this instruction traps to trap vector #2.

CAD

Instruction:
Add carry bit to contents

Type:
Single Operand

Format:



Description:
Adds the contents of the C bit in the I register to the contents of the location specified in the address syllable. The address syllable can specify a memory location, an immediate operand, or one of the R registers. The contents of the I register are affected as follows:

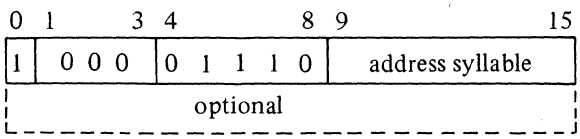
- o If a carry occurs during the operation, the C bit is set to 1; otherwise, it is set to 0.
- o If the result is more than 16 bits long, the OV bit is set to 1; otherwise, it is set to 0. This cannot cause an overflow trap.

CL

Instruction:
Clear

Type:
Single Operand

Format:



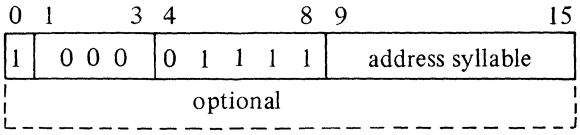
Description:
Stores zeros in the location specified in the address syllable. The address syllable can specify a memory location, an immediate operand, or one of the R registers.

CLH

Instruction:
Clear half-word

Type:
Single Operand

Format:



Description:
Stores 0s in the half-word location specified in the address syllable. The address syllable can specify a memory location, an immediate operand, or another R register, with the byte being positioned as follows:

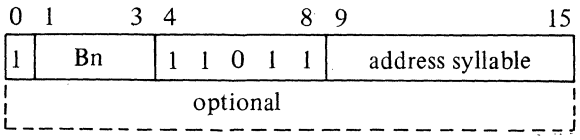
– memory location, not indexed	– left byte
– memory location, indexed	
index value even	– left byte
index value odd	– right byte
– immediate operand	– left byte
– R register	– right byte

CMB

Instruction:
Compare contents to B register

Type:
Double Operand

Format:



Description:
Compares the word in the designated B register with the word in the effective address and sets the G and L indicators according to the results of

the comparison. The address syllable can specify a memory location, an immediate operand, or another B register.

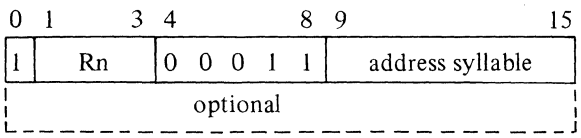
The greater than (G) and less than (L) indicators are set or reset depending on the 16-bit unsigned contents of the two operands. The setting of the unlike signs (U) indicator is not defined.

CMH

Instruction:
Compare half-word (byte) to R register

Type:
Double Operand

Format:



Description:
Compares the word in the designated R register with the byte (sign extended) in the effective address and sets the G, L, and U indicators acording to the results of the comparison. The address syllable can specify a memory location, an immediate operand, or another R register, with the byte being positioned as follows:

– memory location, not indexed	– left byte
– memory location, indexed	
index value even	– left byte
index value odd	– right byte
– immediate operand	– left byte
– R register	– right byte

The greater than (G) and less than (L) indicators are set or reset depending on the 16-bit unsigned contents of the two operands. The unlike signs (U) indicator is cleared if bit zero of both operands are the same, set if different.

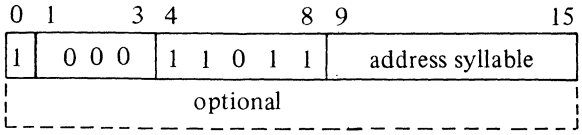
Example: R register contains $(00FF)_{16}$
Effective address contains $(FF)_{16}$
CMH sets L and U, resets G

CMN

Instruction:
Compare address to null

Type:
Single Operand

Format:



Description:
Compares the contents of the location or B register specified by the address syllable to a null address (the address 0).

The contents if the I register are affected as follows:

- o The G bit is set to 0 if the specified address is equal to null; otherwise, it is set to 1.
- o The L bit is set to 0.
- o The U bit is affected, but its value is undefined.

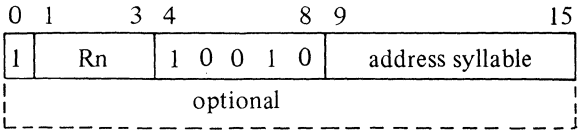
The address syllable can specify a memory location, an immediate operand, or one of the B registers.

CMR

Instruction:
Compare to R register

Type:
Double Operand

Format:



Description:
Compares the word in the designated R register with the word in the effective address and sets the G, L, and U indicators according to the results of the comparison. The address syllable

can specify a memory location, an immediate operand, or another R register.

The greater than (G) and less than (L) indicators are set or reset depending on the 16-bit unsigned contents of the two operands. The unlike signs (U) indicator is cleared if bit zero of both operands are the same, set if different.

Note that this instruction can be used to do either a logical (alphabetical) comparison or an algebraic comparison. The branch instruction that follows performs either an algebraic test or a logical test.

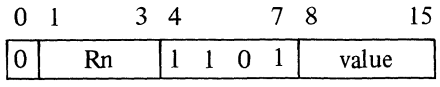
Example: R register contains $(FFFF)_{16}$
Effective address contains $(7777)_{16}$
CMR sets G and U, resets L
BG (Branch if greater) branches
BAG (Branch if algebraically greater) does not

CMV

Instruction:
Compare value to R register

Type:
Short Value Immediate

Format:



Description:
Compares the word in the designated R register with the byte (with its sign extended) contained in the value field of the instruction and sets/resets the G (greater than), L (less than), and U (unlike signs) indicators according to the results of the comparison.

Examples:

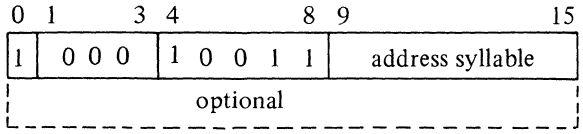
Contents of R register	Value Field in Instruction	G	L	U
$(00AB)_{16}$	$(AB)_{16}$	0	1	1
$(FFAB)_{16}$	$(AB)_{16}$	0	0	0
$(FFAB)_{16}$	$(AA)_{16}$	1	0	0
$(-7)_{10}$	$(-7)_{10}$	0	0	0
$(-7)_{10}$	$(+7)_{10}$	1	0	1

CMZ

Instruction:
Compare to 0

Type:
Single Operand

Format:



Description:
Compares the contents of the location specified in the address syllable to 0. The address syllable can specify a memory location, an immediate operand, or one of the R registers. The contents of the I register are affected as follows:

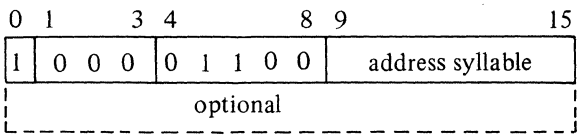
- o If the contents of the specified location do not equal 0, the G bit is set to 1; otherwise, it is set to 0.
- o The L bit is set to 0.
- o If the first bit of the specified location equals 1, the U bit is set to 1; otherwise, it is set to 0.

CPL

Instruction:
Complement

Type:
Single Operand

Format:



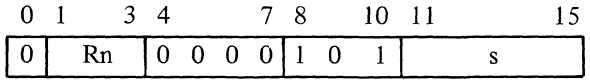
Description:
Ones complements the contents of the location specified in the address syllable. No indicators are affected. The address syllable can specify a memory location, an immediate operand, or one of the R registers.

DAL

Instruction:
Double-shift arithmetic-left

Type:
Shift Long

Format:



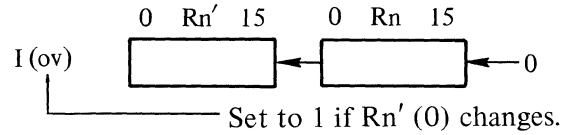
s is number of positions shifted (0-31)

Description:
Shifts the contents of the even-odd R register pair (i.e., R2 and R3, R4 and R5, R6, and R7), identified in the Rn field (which must be odd), left the number of bit positions specified by the s field. The bit positions vacated by the shift are filled with binary 0s. If the s field contains 0s, the shift distance is obtained from bits 11 through 15 of general register R1.

The contents of the I register are affected as follows:

- o If the contents of bit 0 in the even-numbered R register changes, the OV bit is set to 1; otherwise, it is set to 0.

Pictorial Representation:

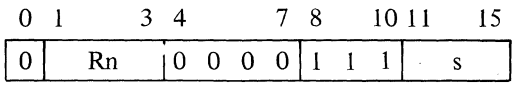


DAR

Instruction:
Double-shift arithmetic-right

Type:
Shift Long

Format:



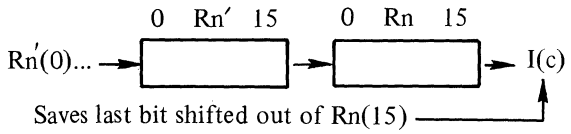
s is number of positions shifted (0-31)

Description:
Shifts the contents of the even-odd R register pair (i.e., R2 and R3, R4 and R5, R6 and R7), identified in the Rn field (which must be odd), right the number of bit positions specified by the s field. The bit positions vacated by the shift are filled with the sign value originally contained in bit 0. If the s field contains 0s, the shift distance is obtained from bits 11 through 15 of general register R1.

The contents of the I register are affected as follows:

- o C-bit contains the last binary digit shifted out of the odd-numbered R register.

Pictorial Representation:

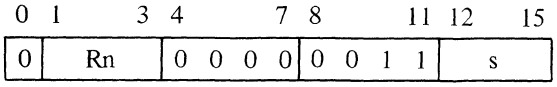


DCL

Instruction:
Double-shift closed-left

Type:
Shift Short

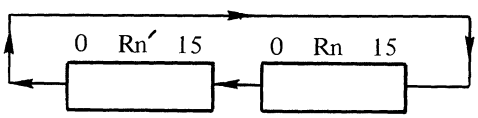
Format:



s is number of positions shifted (0-15)

Description:
Shifts the contents of the even-odd R register pair (i.e., R2 and R3, R4 and R5, R6 and R7), identified in the Rn field (which must be odd), left the number of bit positions specified by the s field. The bits shifted out of the even-numbered R register are placed in the bit positions of the odd-numbered R register vacated as the bits are shifting left. If the s field contains 0s, the shift distance obtained from bits 12 through 15 of general register R1.

Pictorial Representation:

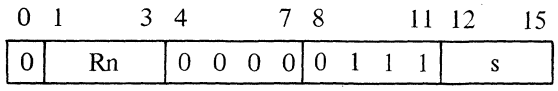


DCR

Instruction:
Double-shift closed-right

Type:
Shift Short

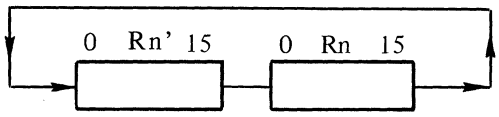
Format:



s is number of positions shifted (0-15)

Description:
Shifts the contents of the even-odd R register pair (i.e., R2 and R3, R4 and R5, R6 and R7), identified in the Rn field (which must be odd), right the number of bit positions specified by the s field. The bits shifted out of the odd-numbered R register are placed in the bit positions of the even-numbered R register vacated as the bits are shifting right. If the s field contains 0s, the shift distance is obtained from bits 12 through 15 of general register R1.

Pictorial Representation:

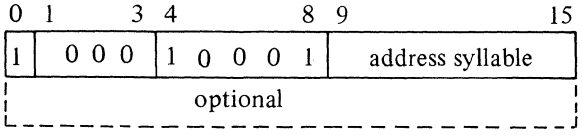


DEC

Instruction:
Decrement

Type:
Single Operand

Format:



Description:
Decrements by 1 the contents of the location or register specified in the address syllable, then copies bit 0 of the addressed word or register into I(b).

This instruction operates in read modify write (RMW) mode, which prevents any other processor in a multiprocessor environment from accessing the location being modified until the modification is completed. The address syllable can specify a memory location, an immediate operand, or one of the R registers.

The contents of the I register are affected as follows:

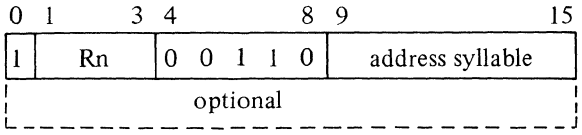
- o If the decrementation causes a carry to occur, the C bit is set to 1; otherwise, it is set to 0.
- o If the value being decremented was -32,768 (-2^{15}), I(OV) is set to 1; otherwise, I(OV) is cleared to 0. However, this can never cause an overflow trap.
- o I(b) is set as described above.

DIV

Instruction:
Divide R register

Type:
Double Operand

Format:



Description:
If the designated R register is R1 to R6, its contents are divided by the word at the effective address and the remainder is lost; if the designated R register is R7, the double-precision value in R6 and R7 is divided by the word at the effective address with the quotient being developed in R7 and the remainder in R6. The address syllable can specify a memory location, an immediate operand, or another R register.

The contents of the I register are affected as follows:

1. I(OV) is set to 1 if
   a. The divisor = 0
   b. The dividend is -2^{15} times the divisor and the divisor is positive.
   c. The quotient is greater than $2^{15} - 1$ or less than -2^{15}.

   Otherwise I(OV) is cleared to 0.

Divide operations that cause I(OV) to be set terminate with all operands unchanged.

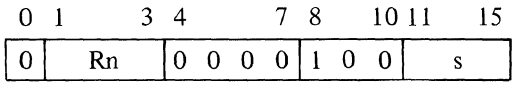
2. The carry indicator (C) is set only if the remainder is not 0 *and* if the remainder is not stored in R6 (i.e., when the first operand does not specify R7). If R7 were specified as the first operand, the remainder would be stored in R6 and the carry bit would be unchanged whether or not the remainder was zero.
3. The carry indicator (C) is cleared if the remainder is zero and is not stored in R6.

DOL

Instruction:
Double-shift open-left

Type:
Shift Long

Format:



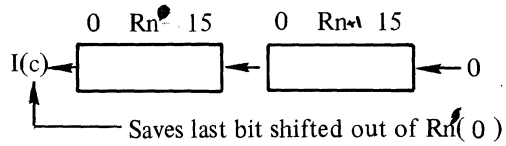
s is number of positions shifted (0-31)

Description:
Shifts the contents of the even-odd R register pair (i.e., R2 and R3, R4 and R5, R6 and R7), identified in the Rn field (which must be odd), left the number of bit positions specified by the s field. The bit positions vacated by the shift are filled with binary 0s. If the s field contains 0s, the shift distance is obtained from bits 11 through 15 of general register R1.

The contents of the I register are affected as follows:

- C-bit contains the last binary digit shifted out of the even-numbered R register.

Pictorial Representation:

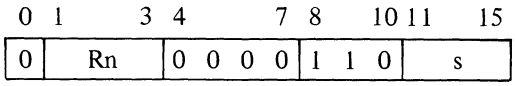


DOR

Instruction:
Double-shift open-right

Type:
Shift Long

Format:



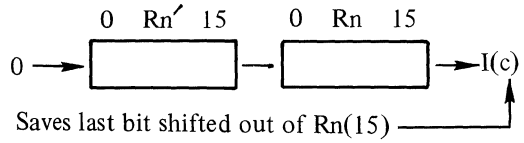
s is number of positions shifted (0-31)

Description:
Shifts the contents of the even-odd R register pair (i.e., R2 and R3, R4 and R5, R6 and R7), identified in the Rn field (which must be odd), right the number of bit positions specified by the s field. The bit positions vacated by the shift are filled with binary 0s. If the s field contains 0s the shift distance is obtained from bits 12 through 15 of general register R1.

The contents of the I register are affected as follows:

- C-bit contains the last binary digit shifted out of the odd-numbered R register.

Pictorial Representation:



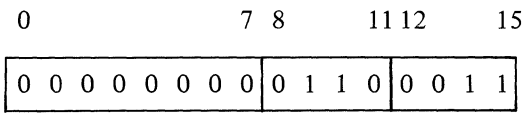
DQA

Instruction:
Dequeue On address

Restriction:
Operates on 6/43 only.

Type:
Generic

Format:



Description:
Unlinks a frame whose priority word address exactly matches B1. B2 points to the lock word.

The contents of the I Register are affected as follows:

- o If the operation is completed, C is set to 1; otherwise 0.
- o If the frame was found and unlinked, L is set to 0; otherwise 1.

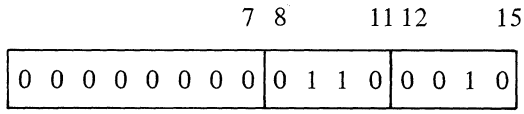
DQH

Instruction:
Dequeue from head

Restriction:
Operates on 6/43 only.

Type:
Generic

Format:



Description:
Unlinks the first frame from the list whose priority value equals or is numerically greater than the contents of R5. B2 is used to point to the lock word, and B1 is returned containing a pointer to the priority word of the unlinked frame. I(G) and I(L) indicate the hit conditions.

The contents of the I Register are affected as follows:

- o If the operation is completed, C is set to 1; otherwise 0.

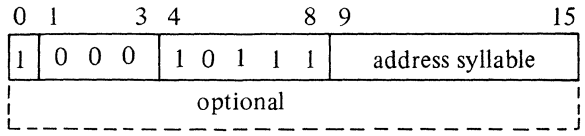
G	L	Condition
0	0	Unlinked frame was first whose priority equalled [R5].
1	0	Unlinked frame was first whose priority exceeded [R5].
0	1	No frame was unlinked, B1 is unchanged, no priority found equal to or greater than [R5].

ENT

Instruction:
Enter

Type:
Single Operand

Format:



Description:
Puts the processor into the unprivileged (user) mode and then jumps to the location specified by the effective address. If the J bit in the mode control (M1) register is on, this instruction traps to trap vector #2.

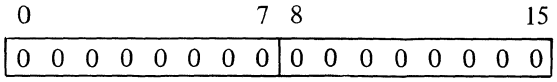
No indicators are affected. The address syllable must specify a memory location, not a register or an immediate operand.

HLT

Instruction:
Halt

Type:
Generic

Format:



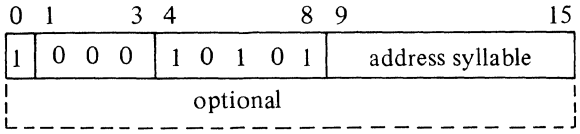
Description:
Stops program execution. HLT state is indicated on the control panel. All interrupts will be honored. The P bit in the S register must be set to 1 (i.e., the central processor must be in the privileged state) for this instruction to be executed. If not, the unprivileged use of a privileged operation is signified by a trap to trap vector #13.

INC

Instruction:
Increment

Type:
Single Operand

Format:



Description:

Copies bit 0 of the contents of the location or register specified in the address syllable into I(b), then increments by 1 the contents of the location or register.

This instruction operates in read modify write (RMW) mode, which prevents any other processor in a multiprocessor environment from accessing the location being modified until the modification is completed. The address syllable can specify a memory location, an immediate operand, or another R register.

The contents of the I register are affected as follows:

- o If the incrementation causes a carry to occur, the C bit is set to 1; otherwise, it is set to 0.
- o If the value being incremented was 32,767, I(OV) is set to 1; otherwise it is cleared to 0. However, this can never cause an overflow trap.
- o I(b) is set as described above.

IO

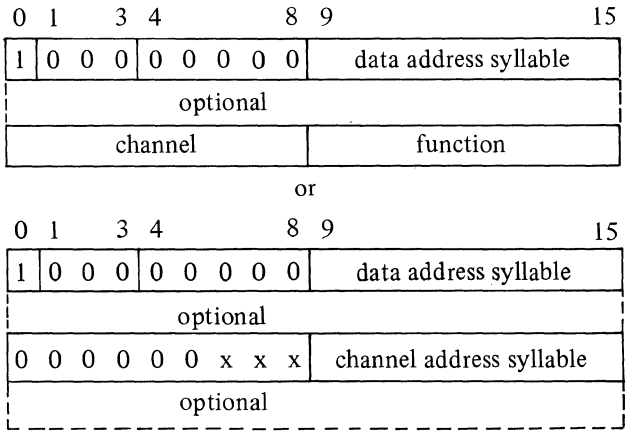
Instruction:

Input/output (word)

Type:

Input/Output

Formats:



Description:

A privileged instruction that can be executed only if the P bit in the s register is set to one; otherwise the instruction traps through trap vector #13.

This instruction initiates an I/O command bus cycle that either outputs a word to a channel or requests the input of a word from a channel. If the channel accepts the I/O command, the input/output (I) indicator is set to a one; if it does not accept the command, the I bit is reset to zero.

The location of the word to be output or of where the word is to be input is described by the data address syllable in the instruction. This address syllable is identical to those in single- and double-operand instructions and can specify a memory location, an immediate operand, or an R register. If a memory location is specified on an output, for example, the data word is first read from the memory to the CP and then output from the CP to the channel. The memory does not send or receive the data directly to or from the channel.

The channel number is a 10-bit value contained in the same word with a 6-bit function code. This word is contained directly in the instruction, or, if the channel number field (bits 0–5) in the instruction is zero, in a location pointed to by the channel address syllable.[2] If the latter is utilized, it is of the general address syllable form and can specify a memory location or an R register.

The channel number can specify an output channel (if it is odd) or an input channel (if it is even). This has nothing to do with whether the I/O instruction initiates an output command or and input command. (It is possible to output to input channels or input from output channels, as well as output to output and input from input.)

The direction of the command is determined by the function code, which also specifies the type of data to be input or output. The function code normally addresses a set up, control, or status register in the channel. However, it could specify a data buffer and thus be used to implement a programmed input/output operation. (The latter technique is not used by standard I/O channels which all work on a DMA basis of I/O.)

[2] On processor to processor transfers, a CAS must be used as the first six bits of a processor channel number are always zero.

Function codes are thus dependent on the design of a particular channel controller. However, the following codes have been assigned and are used by standard channels:

Input codes:
- 02 – Input interrupt control word
- 06 – Input task word
- 0C – Input range
- 10 – Input configuration word A
- 12 – Input configuration word B
- 18 – Input status word #1
- 1A – Input status word #2
- 26 – Input device ID

Output codes:
- 01 – Output control word
- 03 – Output interrupt control word
- 07 – Output task
- 09 – Output address (see note)
- 0D – Output range (see note)
- 11 – Output configuration word A
- 13 – Output configuration word B

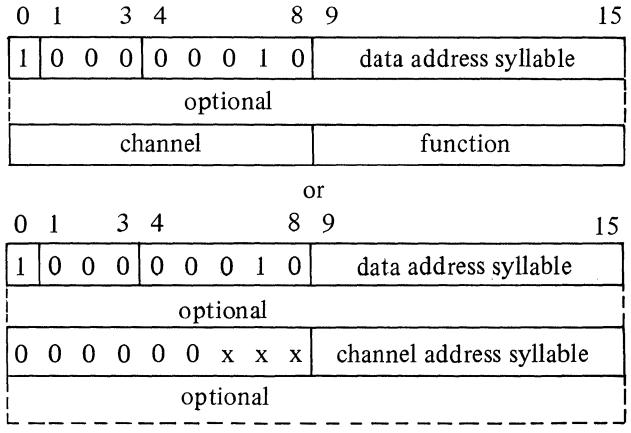
NOTE: The I/O instruction should not use codes 09 and 0D as the address and range should be output with an IOLD instruction.

IOH

Instruction:
Input/output half-word

Type:
Input/Output

Formats:



Description:
A privileged instruction that is identical to the I/O instruction except that it transfers a byte rather than a word. The data address syllable generates a reference to (1) the right byte of an R register or (2) to the left *or* right byte of a word in main memory. If the data address syllable identifies an unindexed memory location, the left byte of that memory location is the target of the input/output operation; if the data address syllable identifies an indexed memory location, the index value is applied (as a byte displacement) to the left byte of that memory location to identify the target byte (which may be either the left *or* right byte of a memory location, depending on whether the index value is even or odd).

In cases where a byte is written into the right half of an R register (input operation), the sign of the byte is extended through bits 0 through 7 (the left byte) of the register. In cases where a byte is written into a memory location (input operation), the other byte of the same word is not modified.

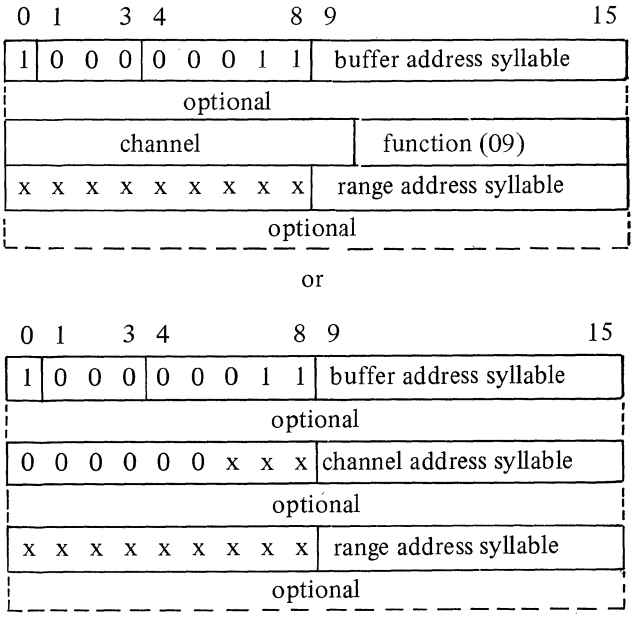
This instruction is designed for programmed I/O operations. It is not currently used by Honeywell implemented controllers.

IOLD

Instruction:
Input/output load

Type:
Input/Output

Formats:



Description:

A privileged instruction that is used to set up a device for DMA operation by outputting an address and a range to it. It is similar in function to two consecutive I/O instructions in that it initiates two I/O output commands, one with a function code of $(09)_{16}$ and one with a function code of $(0D)_{16}$.

The main difference is that it outputs a 17-bit byte address to the DMA address register. The address register is 24 bits in length as opposed to the 16-bit length of the other registers in the channel. To generate a 17-bit word, the IOLD outputs the effective address of the buffer as specified by the "buffer" address syllable, not the contents of the location pointed to by it.

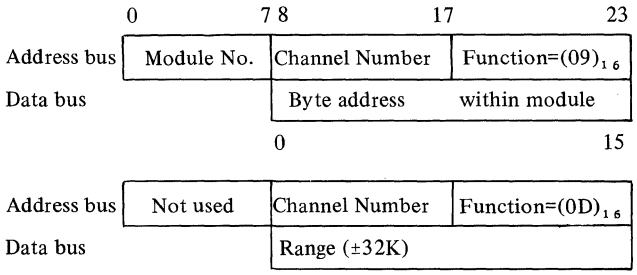
This address is split when put on the bus, with 16 least significant bits being put on the data lines and the high-order bits in the "module" field of the address bus.

The buffer address syllable must define a memory location. If it is not indexed the effective address will define the left byte in a word; if it is indexed it can specify either side.

The channel can either be defined within the instruction or can be pointed to by the channel address syllable. The function code must be $(09)_{16}$.

After the first output command, 04 is automatically added to the function code and another output command is initiated. This command outputs the contents (a 16-bit word) of the location pointed to by the range address syllable. A memory location immediate operand, or R register form of addressing may be used. The range itself is a signed 16-bit value which defines the number of bytes to be transferred.

The formats of the I/O output command bus cycles are as follows:



JMP

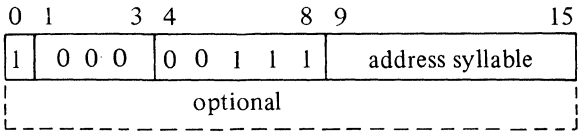
Instruction:

Jump

Type:

Single Operand

Format:



Description:

Jumps to the location specified in the address syllable, unless the J bit in the mode control (M1) register is set, in which case it traps to trap vector #2.

No indicators are affected. The address syllable can specify a memory address, but not a register or an immediate operand.

LAB

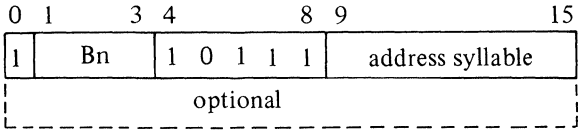
Instruction:

Load effective address into B register

Type:

Double Operand

Format:



Description:

Loads the 16-bit effective address specified by the address syllable into the designated B register. Note that this is the address itself, not its contents. No indicators are affected. The address syllable can specify a memory address or an immediate operand, but not a register.

Example: Assume the following:

Register R3 contains $(0007)_{16}$
Register B5 contains $(B010)_{16}$
Memory location $(B010)_{16}$ contains $(FA03)_{16}$

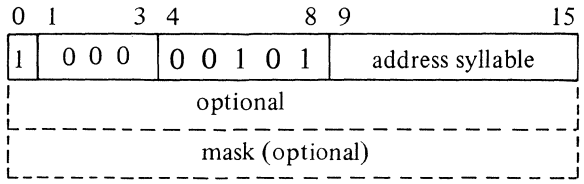
The instruction LAB $B7, *B5.$R3 (indirect through B5 indexed by R3) will load B7 with $(FA0A)_{16}$.

LB

Instruction:
Load bit

Type:
Single Operand

Format:



Description:
Loads the B-bit indicator in the I register with either a single bit or with the result of a test on a group of bits in a word. If any are on, the B bit is set to 1; if none are on, to zero.

To address a single bit, the address syllable may specify an indexed memory address. The index value counts bits relative to bit zero of the effective address.

To test a group of bits in a word, a mask word following the instruction is utilized. Bit positions in the word at the effective address that correspond to one bits in the mask are tested. In this form the address syllable can specify a memory location (unindexed), an immediate operand, or an R register.

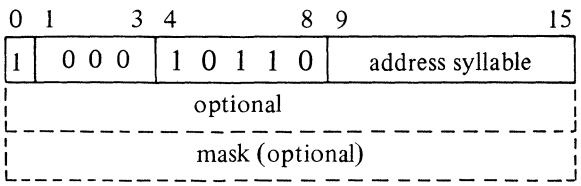
For example, to test bit 13 in a word, indexed addressing with an index value of 13 could be used (or -3, +29, etc.) or unindexed addressing and a mask value of 0004. To test bits 13, 14, and 15, a mask with a value of 0007 should be used.

LBC

Instruction:
Load bit and complement

Type:
Single Operand

Format:



Description:
Loads the B-bit indicator in the I register with either a single bit or with the result of a test on a group of bits in a word. If any are on, the B bit is set to 1; if none are on, to zero. The single bit loaded or the group of bits tested is then complemented.

To address a single bit, the address syllable may specify an indexed memory address. The index value counts bits relative to bit zero of the effective address.

To test a group of bits in a word, a mask word following the instruction is utilized. Bit positions in the word at the effective address that correspond to one bits in the mask are tested. In this form the address syllable can specify a memory location (unindexed), an immediate operand, or an R register.

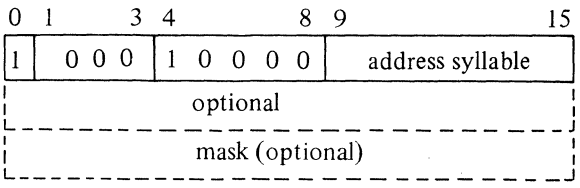
This instruction operates in read modify write (RMW) mode, which prevents any other processor in a multiprocessor environment from accessing the location being modified until the modification is completed.

LBF

Instruction:
Load bit and set false

Type:
Single Operand

Format:



Descripton:
Loads the B-bit indicator in the I register with either a single bit or with the result of a test on a group of bits in a word. If any are on, the B bit is set to 1; if none are on, to zero. The single bit loaded or the group of bits tested are then set to the previous contents of the zero(s).

To address a single bit, the address syllable may specify an indexed memory address. The index value counts bits relative to bit zero of the effective address.

To test a group of bits in a word, a mask word following the instruction is utilized. Bit positions in the word at the effective address that correspond to one bits in the mask are tested. In this form the address syllable can specify a memory location (unindexed), an immediate operand, or an R register.

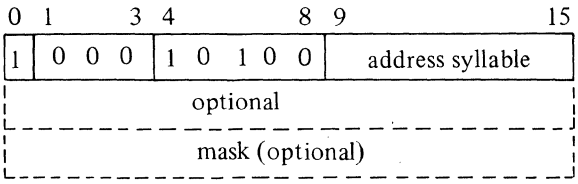
This instruction operates in read modify write (RMW) mode, which prevents any other processor in a multiprocessor environment from accessing the location being modified until the modification is completed.

LBS

Instructon:
Load bit and swap

Type:
Single Operand

Format:



Description:
Loads the B-bit indicator in the I register with either a single bit or with the result of a test on a group of bits in a word. If any are on, the B bit is set to 1; if none are on, to zero. The single bit loaded or the group of bits tested are then set to the previous contents of the B bit.

To address a single bit, the address syllable may specify an indexed memory address. The index value counts bits relative to bit zero of the effective address.

To test a group of bits in a word, a mask word following the instruction is utilized. Bit positions in the word at the effective address that correspond to one bits in the mask are tested. In this form the address syllable can specify a memory location (unindexed), an immediate operand, or an R register.

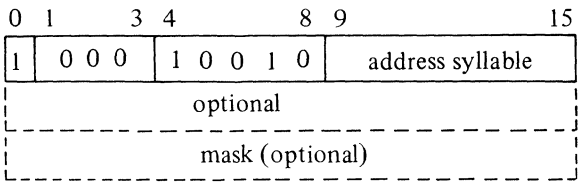
This instruction operates in read modify write (RMW) mode, which prevents any other processor in a multiprocessor environment from accessing the location being modified until the modification is completed.

LBT

Instruction:
Load bit and set true

Type:
Single Operand

Format:



Description:
Loads the B-bit indicator in the I register with either a single bit or with the result of a test on a group of bits in a word. If any are on, the B bit is set to 1; if none are on, to zero. The single bit loaded or the group of bits tested are then set to one.

To address a single bit, the address syllable may specify an indexed memory address. The index value counts bits relative to bit zero of the effective address.

To test a group of bits in a word, a mask word following the instruction is utilized. Bit positions in the word at the effective address that correspond to one bits in the mask are tested. In this form the address syllable can specify a memory location (unindexed), an immediate operand, or an R register.

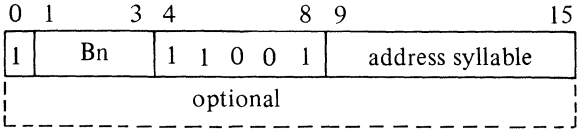
This instruction operates in read modify write (RMW) mode, which prevents any other processor in a multiprocessor environment from accessing the location being modified until the modification is completed.

LDB

Instruction:
Load B register

Type:
Double Operand

Format:



Description:

Loads the designated B register with the word contained at the effective address. No indicators are affected. The address syllable can specify a memory location, an immediate operand, or another B register.

LDH

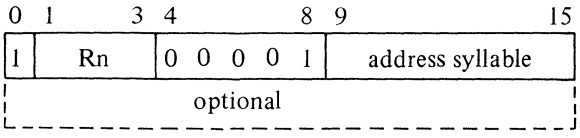
Instruction:

Load half-word (byte) into R register

Type:

Double Operand

Format:



Description:

Loads the byte contained at the effective address into the right half of the designated R register, with the sign being extended into the left half, bits 0–7. No indicators are affected.

The effective address can specify a memory location, an immediate operand, or another R register, with the byte being positioned as follows:

– memory location, not indexed	– left byte
– memory location, indexed	
index value even	– left byte
index value odd	– right byte
– immediate operand–	left byte
– R register	– right byte

Example: Assume that:

memory location 1000 contains $(6789)_{16}$
register R1 contains 0
register R2 contains 1
register B1 contains $(1000)_{16}$

then if LDH $R5, $B1.$R1 is executed: R5 will contain $(0067)_{16}$
but if LDH $R5, $B1.$R2 is executed: R5 will contain $(FF89)_{16}$

LDI

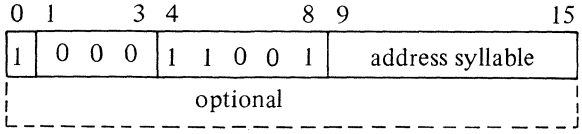
Instruction:

Load double-word integer

Type:

Single Operand

Format:



Description:

Loads the word contained at the effective address into R6 and the following word into R7. No indicators are affected. The address syllable can specify a memory location, an immediate operand, or another R register as follows:

- *Memory location:* The left-hand word of a double-word pair will be addressed. Auto-increment and auto-decrement modes of addressing will add (subtract) two from the base register. A value in the index register will count double-words; it is shifted one place to the left before being applied. Auto-indexing will cause one to be added (subtracted) from the index register.
- *Immediate operand:* This will be a double-word operand and thus the instruction will be three words in length.
- *Register operand:* This form of addressing must address the right-hand register containing a double-word pair, i.e., either R3, which loads the contents of R2 and R3 into R6 and R7, or R5, which selects R4 and R5.

LDR

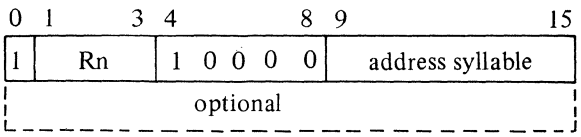
Instruction:

Load R register

Type:

Double Operand

Format:



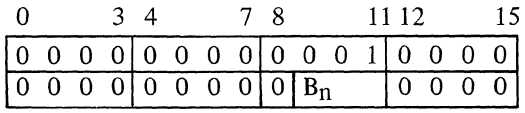
Description:
Loads the designated R register with the word contained at the effective address. No indicators are affected. The address syllable can specify a memory location, an immediate operand, or another R register.

LDT

Instruction:
Load stack address register
Restriction:
Operates on 6/43 only.

Type:
Generic

Format:



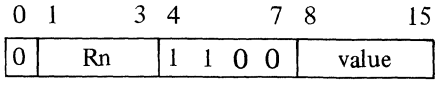
Description:
Loads the contents of Bn into the Stack Address Register (T). The contents of the I Register are unaffected.

LDV

Instruction:
Load value

Type:
Short Value Immediate

Format:



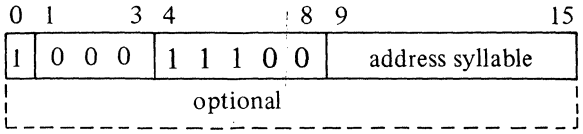
Description:
Loads the byte contained in the value field of the instruction into the right half of the designated R register with the sign being extended into the left hand. No indicators are affected.

LEV

Instruction:
Level change

Type:
Single Operand

Format:

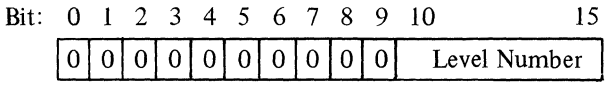


Description:
Sets or resets level activity bits according to the contents of the location indicated by the address syllable.

The LEV instruction, when used on a 6/40 with the MMU, performs the same level management functions performed without the MMU. However, when an MMU is present and a level change takes place, the MMU segment descriptor registers are optionally loaded. In SAF mode, the 16 descriptors of segments 0.00 through 0.15 are loaded; in LAF mode, all 31 descriptors are loaded.

The following bit configurations in the indicated location produce the actions described below.

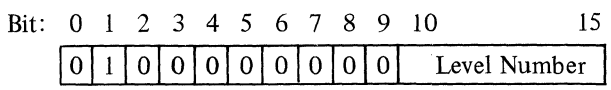
Schedule Interrupt Level, Scan, and Dispatch



The level activity bit for the designated level is set. The level activity bits are scanned and the highest active level ascertained. The context of the current level is saved (unless the current level is the highest active level). The context of the highest active level is restored (again, unless the current level is the highest active level).

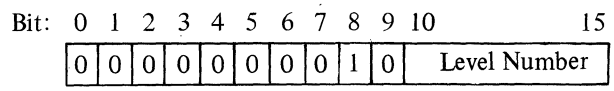
NOTE: Context save/restore is also skipped if the current level and the highest active level, though different, share a common save area.

Schedule Interrupt Level, Defer Interrupt



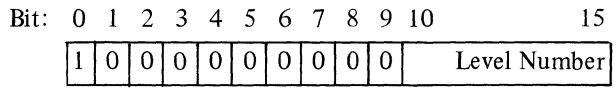
The level activity bit for the designated level is set. Execution continues at the current level.

Inhibit



The level activity bit for the designated level is set. The interrupt vector for the designated level is set equal to the interrupt vector for the current level. Execution of the current task continues at the designated level.

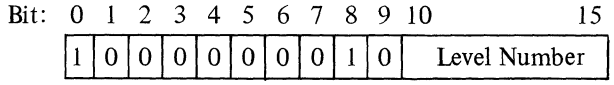
Schedule Interrupt Level, Suspend, Scan, and Dispatch



The level activity bit for the designated level is set. The level activity bit for the current level is cleared. The level activity bits are scanned and the highest level ascertained. The context of the current level is saved. The context of the highest active level is restored.

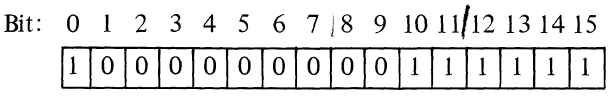
NOTE: Context save/restore is skipped if the current level and the highest active level share a common save area.

Suspend, Inhibit



The level activity bit for the current level is reset. The level activity bit for the designated level is set. The interrupt vector for the designated level is set equal to the interrupt vector for the current level. Execution of the task continues at the designated level.

Suspend, Scan, and Dispatch



Ends execution at the current level. The level activity bit for the current level is cleared. The level activity bits are scanned and the highest active level ascertained. The context of the current level is saved. The context of the highest active level is restored (unless the current level and the highest active level share a common save area).

The address syllable can specify a memory location, an individual operand, or an R register.

The P bit in the S register must be set to 1 (i.e., the central processor must be in the privileged state) for this instruction to be executed. If the P bit is not set to 1, the unprivileged use of a privileged operation is signified by a trap to trap vector #13.

The contents of the S register are affected as follows:

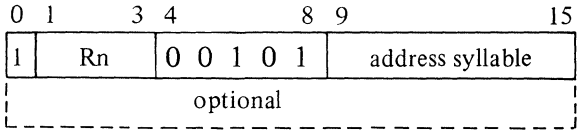
- o Bits 10 through 15 of the S register are set to indicate the priority level at which processing continues after execution of the LEV instruction.

LLH

Instruction:
Load logical half-word (byte) into R register

Type:
Double Operand

Format:



Description:
Loads the byte contained at the effective address into the right half of the designated R register, with the left half, bits 0–7 being cleared to zero. No indictors are affected.

The effective address can specify a memory location, an immediate operand, or another R register, with the byte being positioned as follows:

– memory location, not indexed	– left byte
– memory location, indexed	
index value even	– left byte
index value odd	– right byte
– immediate operand	– left byte
– R register	– right byte

Example: Assume that:

memory location 1000 contains $(6789)_{16}$
register R1 contains 0
register R2 contains 1
register B1 contains $(1000)_{16}$

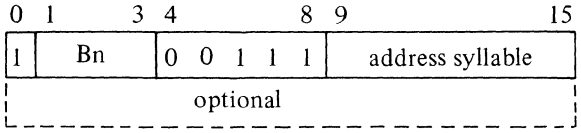
then if LLH $R5, $B1.$R1 is executed: R5 will contain $(0067)_{16}$
but if LLH $R5, $B1.$R2 is executed: R5 will contain $(0089)_{16}$

LNJ

Instruction:
Load B register and jump (Link Jump)

Type:
Double Operand

Format:



Description:
Loads the address of the next sequential instruction into the designated B register and jumps to the location specified by the effective address. If the J bit in the M1 register is on, this instruction traps to trap vector #2.

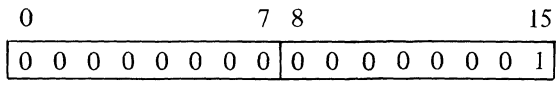
No indicators are affected. The address syllable must specify a memory location, but not a register or an immediate operand.

MCL

Instruction:
Call monitor via trap

Type:
Generic

Format:



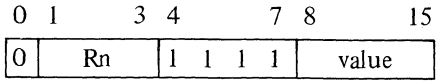
Description:
Calls monitor by a trap to trap vector #1.

MLV

Instruction:
Multiply by value

Type:
Short Value Immediate

Format:



Description:
Multiplies the value in the designated R register by the value (with sign extended) contained in the instruction. A single precision 16-bit product is formed, with overflow bit set if necessary, unless the designated register is R7, in which case a double-precision, 32-bit signed product is developed in R6 and R7.

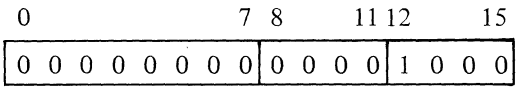
MMM

Instruction:
Memory to memory move

Restriction:
Operates on 6/43 only.

Type:
Generic

Format:



Description:
Moves "n" bytes of memory from one location to another. The address of the first byte to be moved is identified by [B2] + [R2], where R2 contains a signed byte displacement from [B2]. The first byte location to be moved to is identified by [B3] + [R3], where R3 contains a signed byte displacement from [B3]. The number of bytes to be moved is contained in R6. Attempts to move partially overlapped fields causes unspecified results.

The contents of the I Register are unaffected. If R6(0) = 1, Trap 16 occurs. Trap 15 occurs if

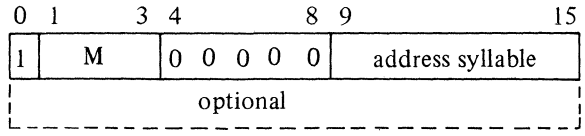
the move would exceed available memory or cause register overflow.

MTM

Instruction:
Modify or test M register

Type:
Double Operand

Format:



Description:

Modifies or tests the contents of the M register identified in the first operand with the contents (mask) of the location specified by the address syllable.

The mask is treated as two 8-bit fields; then, depending on the content of corresponding bits in the two fields (i.e., bit 1 in the first field and bit 1 in the second; bit 2 in the first field and bit 2 in the second; etc.), the corresponding bit in the M register (i.e., if bit 1 in the two mask fields, then bit 1 in the M register) is altered as follows:

- o If bit n in the first mask field is 1, the corresponding bit in the M register is loaded with the contents of the corresponding bit from the second mask field (i.e., M register is modified).
- o If bit n in the first mask field is 0 and the same bit in the second mask field is 1, the corresponding bit in the M register is inclusively ORed. If the result of the ORing is 1, the B bit in the I register is set to 1; otherwise, it is set to 0 (i.e., M register is tested).
- o If bit n in the first mask field is 0 and the same bit in the second mask field is 0, the corresponding bit in the M register is neither modified nor tested.

NOTE: The assembly language instructions LEV, SAVE, and STM store the contents of the M register in a form suitable for reloading by MTM.

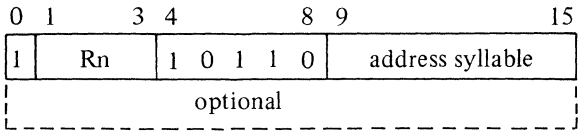
The address syllable can specify a memory location, an immediate operand, or an R register. The register number in the M-number field must be 1; otherwise traps to trap vector #5.

MUL

Instruction:
Multiply R register

Type:
Double Operand

Format:



Description:

Multiplies the word in the designated R register by the word in the effective address. The product is a 16-bit value with the high order bits truncated unless R7 is specified, in which case a double-precision, 32-bit signed product is developed in R6 and R7. Single precision multiplication sets or clears the overflow (OV) indicator depending on the size of the product; double-precision multiplication cannot cause overflow and thus OV is cleared.

The address syllable can specify a memory location, an immediate operand, or another R register.

The contents of the I register are affected as follows:

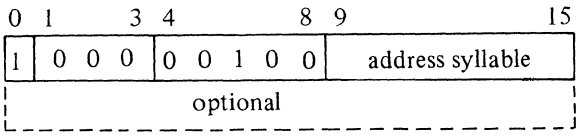
- o If R7 is not specified and the product is more than $2^{15} - 1$ (32,767) or less than -2^{15} (-32768), the OV bit is set to 1; otherwise, it is set to 0.

NEG

Instruction:
Negate

Type:
Single Operand

Format:



Description:

Twos complements the contents of the location specified in the address syllable.

The contents of the I register are affected as follows:

- o If a carry occurs (i.e., the value was zero) during the operation, the C bit is set to 1; otherwise, it is set to 0.
- o If the value complemented was -32768, the OV bit is set to 1; otherwise, it is set to 0. However, this cannot cause an overflow trap.

The address syllable can specify a memory location, an immediate operand, or one of the R registers.

NOP

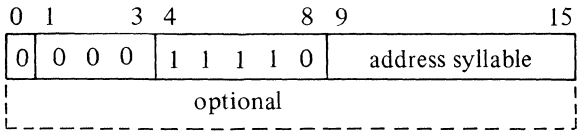
Instruction:

No operation

Type:

Branch on Indicator

Format:



Description:

Performs no operation. In effect, it is the opposite of an unconditional branch (B) instruction.

OR

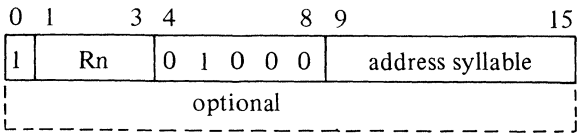
Instruction:

Inclusive OR with R register

Type:

Double Operand

Format:



Description:

Logically ORs the word at the effective address to the word contained in the designated R register. No indicators are affected. The address syllable can specify a memory address, an immediate operand, or another R register.

The following chart illustrates the result of inclusively ORing bits:

First operand bit	0	0	1	1
Second operand bit	1	0	1	0
Result	1	0	1	1

Examples

R register (before)	Effective Address	R register (after)
$(000)_{16}$	$(0007)_{16}$	$(0007)_{16}$
$(FFF0)_{16}$	$(0007)_{16}$	$(FFF7)_{16}$
$(FFF3)_{16}$	$(0007)_{16}$	$(FFF7)_{16}$

ORH

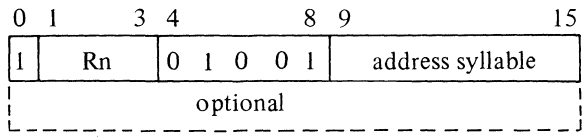
Instruction:

Half-word (byte) inclusive OR with R register

Type:

Double Operand

Format:



Description:

Logically ORs the byte at the effective address (with its sign extended) to the word contained in the designated R register. No indicators are affected. The address syllable can specify a

memory address, an immediate operand, or another R register, with the byte being positioned as follows:

– memory location, not indexed	– left byte
– memory location, indexed	
index value even	– left byte
index value odd	– right byte
– immediate operand	– left byte
– R register	– right byte

The following chart illustrates the result of inclusively ORing bits:

First operand bit	0	0	1	1
Second operand bit	1	0	1	0
Result	1	0	1	1

Examples:

R register (before)	Effective Address	R register (after)
$(0000)_{16}$	$(07)_{16}$	$(0007)_{16}$
$(FFF0)_{16}$	$(07)_{16}$	$(FFF7)_{16}$
$(FFF3)_{16}$	$(07)_{16}$	$(FFF7)_{16}$
$(0007)_{16}$	$(87)_{16}$	$(FF87)_{16}$

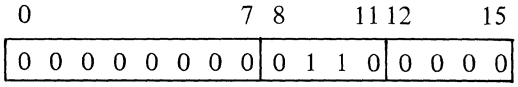
QOH

Instruction:
Queue on head

Restriction:
Operates on 6/43 only.

Type:
Generic

Format:



Description:
Links a new frame into the list before the first frame that has the same priority number, before the first frame that has a numerically higher priority number, or as the last frame if no equal or greater priority is found. B2 is used to point to the lock word of the list, and R5 contains the priority to be assigned to the new frame. The priority word will be loaded with the contents of R5. B1 points to the priority word of the frame to be added. The pointer following the priority will be loaded by the CP with the correct frame pointer.

The contents of the I Register are affected as follows:

- o If the operation is completed, C is set to 1; otherwise 0.

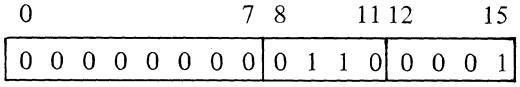
QOT

Instruction:
Queue on tail

Restriction:
Operates on 6/43 only.

Type:
Generic

Format:



Description:
Links a new frame into the list after the last frame that has the same priority number, before the first frame that has a numerically higher priority number, or as the last frame if no equal or greater priority is found. B1, B2, and R5 are used as in QOH. The contents of the I Register are affected as follows:

- o If the operation is completed, C is set to 1; otherwise 0.

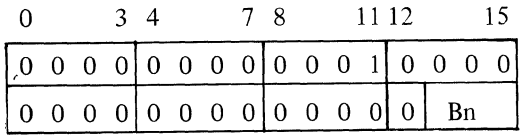
RLQ

Instruction:
Relinquish Stack Space

Restriction:
Operates on 6/43 only.

Type:
Generic

Format:



Description:

Converts a consumed stack frame into available space by updating the current length in words (CW) in the stack header. Bn is loaded with the leftmost address of the frame at the top of the stack. The contents of the I Register are unaffected.

Trap 16 will occur if CW is already zero, or if RLQ would cause CW to go negative.

Trap 9 will occur if CW is reduced to zero.

RSC

Instruction:

Rescan configuration

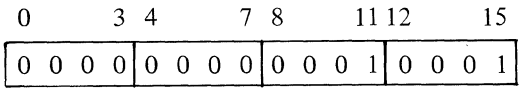
Restriction:

Operates on 6/43 only.

Type:

Generic

Format:



Description:

Causes the CPU to scan a predetermined list of channel numbers to determine the presence/absence of an optional processor. The results of the scan are used to update internal CPU firmware/hardware flags that direct instruction execution (e.g., trap or execution by optional processor).

An identical scan is performed automatically on system power up or initialize. The contents of the I Register are unaffected.

RSTR

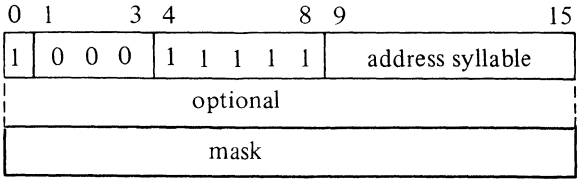
Instruction:

Restore context

Type:

Single Operand

Format:

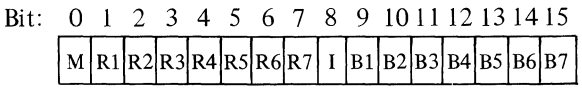


Description:

Restores the registers specified in the mask starting from the location specified in the address syllable.

The address syllable must specify a memory location. The mask specifies which registers are to be restored. If the mask is all zeros, the contents of R1 are used as the mask.

Depending on which bits in the specified mask are set to 1, the registers that can be restored as follows:



This mask should be the same as the one used to save the registers (see the SAVE instruction).

RTCF

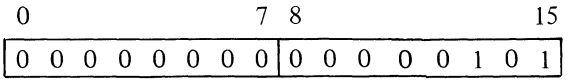
Instruction:

Real-time clock off

Type:

Generic

Format:



Description:

Disables real-time clock. The P bit in the S register must be set to 1 (i.e., the central processor must be in the privileged state) for this instruction to be executed. If not, the unprivileged use of a privileged operation is signified by a trap to trap vector #13.

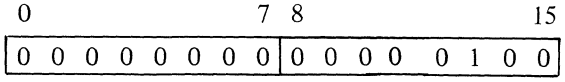
RTCN

Instruction:

Real-time clock on

Type:
Generic

Format:



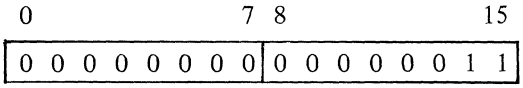
Description:
Starts the real-time clock. The P bit in the S register must be set to 1 (i.e., the central processor must be in the privileged state) for this instruction to be executed. If not, the unprivileged use of a privileged operation is signified by a trap to trap vector #13.

RTT

Instruction:
Return from trap

Type:
Generic

Format:



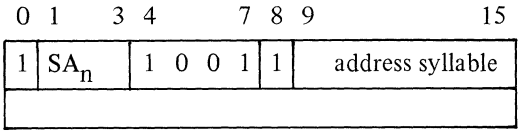
Description:
Restores the registers that were saved in the trap save area when the trap was entered. Leaves the central processor in the privileged state only if both the saved state and the trap handler executing the RTT are privileged. Returns the trap save area block to the trap save area memory pool. Returns control to the next instruction to be executed (determined by the event that caused the trap).

SAD

Instruction:
Scientific add

Type:
Double Operand

Format:



Description:
Adds the floating-point or integer value in the location, scientific accumulator, or R register identified in the second operand to the contents of the scientific accumulator specified in the first operand. The result is saved in the scientific accumulator.

This instruction uses the optional Scientific Instruction Processor (SIP). A Floating-Point Simulator is available to allow this instruction to be executed on systems that do not include an SIP. Information on the Floating-Point Simulator is available in the *Executive and Input/Output* manual.

Scientific Indicator Settings are as follows:

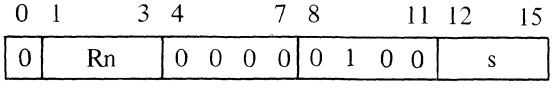
EU: set to 1 on exponent underflow; otherwise, set to 0.
PE: set to 1 if nonzero bits are lost during right shift; otherwise, set to 0.

SAL

Instruction:
Single-shift arithmetic-left

Type:
Shift Short

Format:



s is number of positions shifted (0-15)

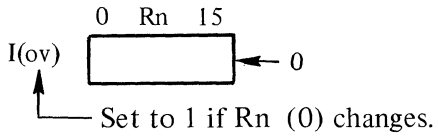
Description:
Shifts the contents of the R register, identified in the Rn field, left the number of bit positions specified in the s field. The bit positions vacated by the shift are filled with binary 0s. If the s field contains 0s, the shift distance is obtained from bits 12 through 15 of general register R1.

The contents of the I register are affected as follows:

- o If the contents of bit 0 in the R register

change, the OV bit is set to 1; otherwise, it is set to 0.

Pictorial Representation:

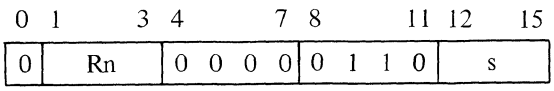


SAR

Instruction:
Single-shift arithmetic-right

Type:
Shift Short

Format:



s is number of positions shifted (0-15)

Description:
Shifts the contents of the R register, identified in the Rn field, right the number of bit positions specified in the s field. The bit positions vacated by the shift are filled with the sign value originally contained in bit 0.

The contents of the I register are affected as follows:

- o C bit contains the last binary digit shifted out of the R register.

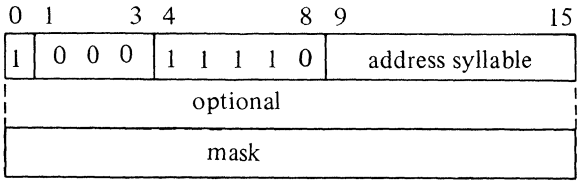
If the s field contains 0s, the shift distance is obtained from bits 12 through 15 of general register R1.

SAVE

Instruction:
Save context

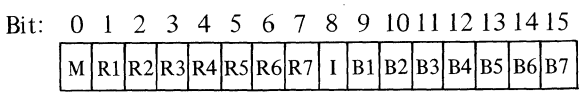
Type:
Single Operand

Format:



Description:
Saves the registers specified in the mask starting at the location specified in the address syllable.

The address syllable must specify a memory address. The mask specifies which registers are to be saved. Each bit in the mask represents a particular register which can be saved, as shown below:



If a mask bit is set to 1, the corresponding register is saved. If a mask bit is 0, the corresponding register is not saved. If the mask is all zeros, the contents of R1 are used as the mask.

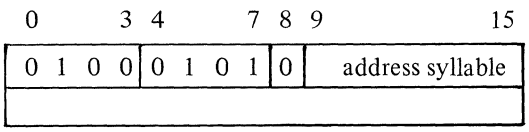
The registers are saved in reverse order. For example, if the mask specified X'CA01' (which, when translated into binary is 1100 1010 0000 0001), indicating that registers M1, R1, R4, R6, and B7 are to be saved, the context save area will contain the registers starting with B7 and ending with M1.

SBE

Instruction:
Branch if equal

Type:
Scientific Indicator Branch

Format:



Description:
Branches to the location specified in the operand if the result of the most recent scientific

comparison sets the SL- and SG-bits of the SI register to 0.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

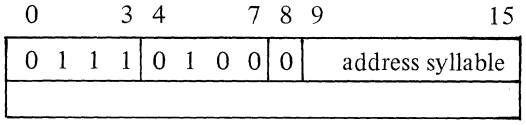
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBEU

Instruction:
Branch if exponent underflow

Type:
Scientific Indicator Branch

Format:



Description:

Branches to the location specified in the operand if the result of the most recent scientific comparison sets the EU-bit in the SI register to 1.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

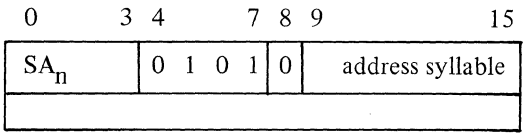
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBEZ

Instruction:
Branch if SA equal to zero

Type:
Scientific Accumulator Branch

Format:



Description:

Branches to the location specified in the second operand if the scientific accumulator identified in the first operand contains a floating-point value algebraically equal to 0.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

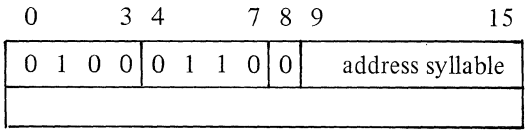
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBG

Instruction:
Branch if greater than

Type:
Scientific Indicator Branch

Format:



Description:

Branches to the location specified in the operand if the result of the most recent scientific comparison sets the SG-bit in the SI register to 1.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch

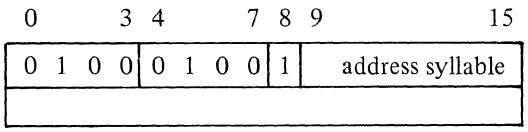
Simulator, if present, is entered via trap vector #5.

SBGE

Instruction:
Branch if greater than or equal

Type:
Scientific Indicator Branch

Format:



Description:
Branches to the location specified in the operand if the result of the most recent scientific comparison sets the SL-bit of the SI register to 0.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

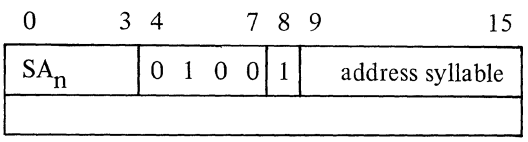
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBGEZ

Instruction:
Branch if SA greater than or equal to zero

Type:
Scientific Accumulator Branch

Format:



Description:
Branches to the location specified in the second operand if the scientific accumulator identified in the first operand contains a nonnegative floating-point value.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

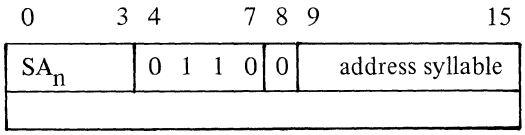
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBGZ

Instruction:
Branch if SA greater than zero

Type:
Scientific Accumulator Branch

Format:



Description:
Branches to the location specified in the second operand if the scientific accumulator identified in the first operand contains a positive floating-point value.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

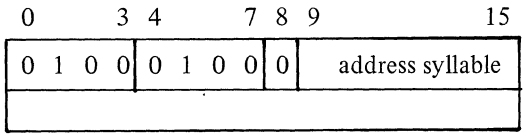
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBL

Instruction:
Branch if less than

Type:
Scientific Indicator Branch

Format:



Description:

Branches to the location specified in the operand if the result of the most recent scientific comparison sets the SL-bit of the SI register to 1.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

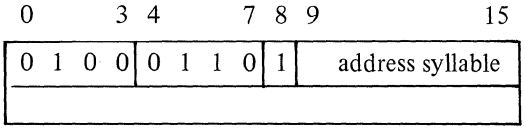
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBLE

Instruction:
Branch if less than or equal

Type:
Scientific Indicator Branch

Format:



Description:

Branches to the location specified in the operand if the result of the most recent scientific comparison sets the SG-bit in the SI register to 0.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

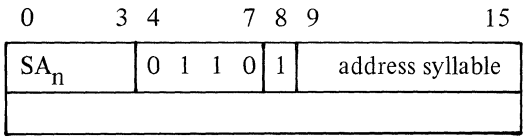
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBLEZ

Instruction:
Branch if SA less than or equal to zero

Type:
Scientific Accumulator Branch

Format:



Description:

Branches to the location specified in the second operand if the scientific accumulator identified in the first operand contains a floating-point value algebraically equal to or less than 0.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

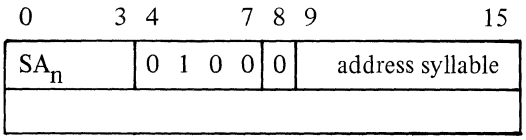
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBLZ

Instruction:
Branch if SA less than zero

Type:
Scientific Accumulator Branch

Format:



Description:

Branches to the location specified in the

second operand if the scientific accumulator identified in the first operand contains a negative floating-point value.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

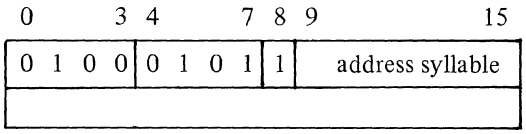
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBNE

Instruction:
Branch if not equal

Type:
Scientific Indicator Branch

Format:



Description:
Branches to the location in the operand if the result of the most recent scientific comparison sets either the SL- or SG-bit of the SI register to 1.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is exeucuted.

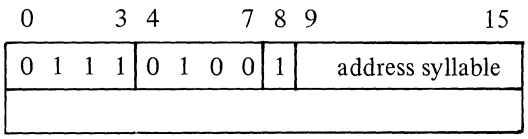
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBNEU

Instruction:
Branch if no exponent underflow

Type:
Scientific Indicator Branch

Format:



Description:
Branches to the location specified in the operand if the result of the most recent scientific comparison sets the EU-bit of the SI register to 0.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

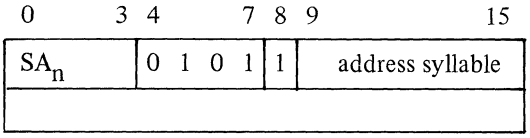
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBNEZ

Instruction:
Branch if SA not equal to zero

Type:
Scientific Accumulator Branch

Format:



Description:
Branches to the location specified in the second operand if the scientific accumulator identified in the first operand contains a floating-point value not algebraically equal to 0.

If branching occurs, the following takes place. If the J-bit of the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case or if the J-bit contains a binary 0, the instruction sequence starting at the

location specified by the operand is executed.

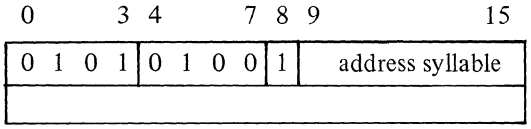
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBNPE

Instruction:
Branch if no precision error

Type:
Scientific Indicator Branch

Format:



Description:

Branches to the location specified in the operand if the result of the most recent scientific comparison sets the PE-bit of the SI register to 0.

If branching occurs, the following takes place. If the J-bit of the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

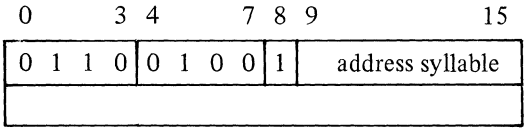
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBNSE

Instruction:
Branch if no significance error

Type:
Scientific Indicator Branch

Format:



Description:

Branches to the location specified in the operand if the result of the most recent scientific comparison sets the SE-bit of the SI register to 0.

If branching occurs, the following takes place. If the J-bit in the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

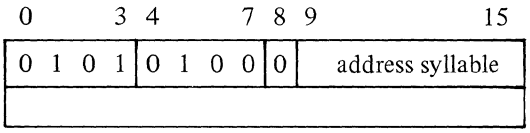
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBPE

Instruction:
Branch if precision error

Type:
Scientific Indicator Branch

Format:



Description:

Branches to the location specified in the operand if the result of the most recent scientific comparison sets the PE-bit of the SI register to 1.

If branching occurs, the following takes place. If the J-bit of the M1 register contains a binary 1, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

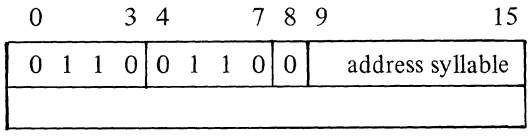
If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SBSE

Instruction:
Branch if significance error

Type:
Scientific Indicator Branch

Format:



Description:

Branches to the location specified in the operand if the result of the most recent scientific comparison sets the SE-bit of the SI register to 1.

If branching occurs, the following takes place. If the J-bit of the M1 register contains a binary 1, the trace procedure is entered via trap vector #2. Upon completion, the trace procedure automatically branches to the address specified by the operand. In this case, or if the J-bit contains a binary 0, the instruction sequence starting at the location specified by the operand is executed.

If the Scientific Instruction Processor (SIP) is not installed on this system, the Scientific Branch Simulator, if present, is entered via trap vector #5.

SCL

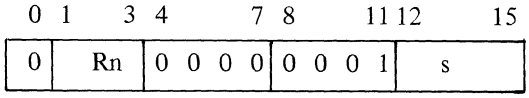
Instruction:

Single-shift closed-left

Type:

Shift Short

Format:



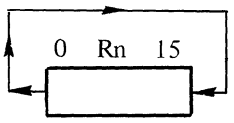
s is number of positions shifted (0-15)

Description:

Shifts the contents of the R register identified in the Rn field left the number of bit positions specified in the s field. The bits shifted out of the register are placed in the bit positions vacated by shifted bits as they are shifting.

If the s field contains 0s, the shift distance is obtained from bits 12 through 15 of general register R1. No indicators are affected.

Pictorial Representation:



SCM

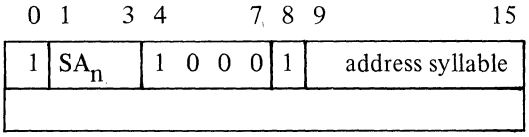
Instruction:

Scientific compare

Type:

Double Operand

Format:



Description:

Compares the contents of the scientific accumulator identified in the first operand to the floating-point or integer value in the location specified in the second operand.

Scientific indicator settings include the following:

SG – Set to 1 if contents of the scientific accumulator are greater than the contents of the location; otherwise, set to 0.

SL – Set to 1 if contents of the scientific accumulator are less than the contents of the location; otherwise, set to 0.

PE – Set to 1 if nonzero bits are lost during right shift for scaling before comparison; otherwise, set to 0.

If the Scientific Instruction Processor (SIP) is not installed on this system, the instruction causes the Floating-Point Simulator, if present, to be entered via trap vector #3.

SCR

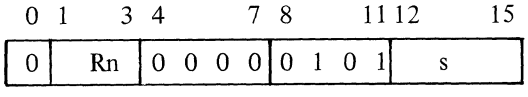
Instruction:

Single-shift closed-right

Type:

Shift Short

Format:



s is number of positions shifted (0-15)

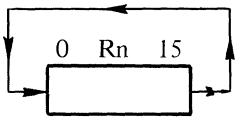
Description:

Shifts the contents of the R register, identified

in the Rn field, right the number of bit positions specified in the s field. The bits shifted out of the register are placed in the bit positions vacated by shifted bits as they are shifting.

If the s field contains 0s, the shift distance is obtained from bits 12 through 15 of general register R1. No indicators are affected.

Pictorial Representation:

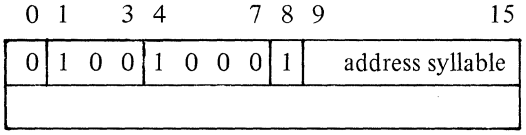


SCZD

Instruction:
Scientific compare to zero (2 words)

Type:
Single Operand

Format:



Description:
Compares the short-precision floating-point value in the specified location or scientific accumulator to 0.

Scientific indicator settings include the following:

SL – Set to 1 if contents of the location are less than 0; otherwise, set to 0.

PE – Set to 1 if nonzero bits are lost during right shift for scaling before comparison; otherwise, set to 0.

If the Scientific Instruction Processor (SIP) is not installed on this system, the instruction causes the Floating-Point Simulator, if present, to be entered via trap vector #3. Since the SI register is not available if the SIP is not installed, the I register is used as a substitute.

The contents of the I register are affected as follows:

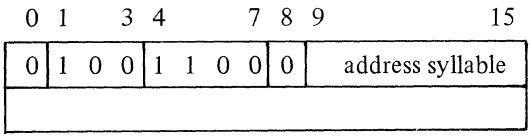
- o If the contents of the specified location do not equal 0, the G-bit is set to 1; otherwise, it is set to 0.
- o The L-bit is set to 0.
- o If bit 7 of the specified location equals 1, the U-bit is set to 1; otherwise, it is set to 0.

SCZQ

Instruction:
Scientific compare to zero (4 words)

Type:
Single Operand

Format:



Description:
Compares the floating-point value in the specified location or scientific accumulator to 0.

Scientific indicator settings include the following:

SL – Set to 1 if contents of the location are less than 0; otherwise, set to 0.

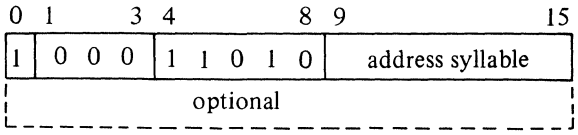
PE – Set to 1 if nonzero bits are lost during right shift for scaling before comparison; otherwise, set to 0.

SDI

Instruction:
Store double-word integer

Type:
Single Operand

Format:



Description:
Stores the contents of R6 and R7 into the effective address and the following word. No indicators are affected. The address syllable can specify a memory location, an immediate

operand, or another R register as follows:

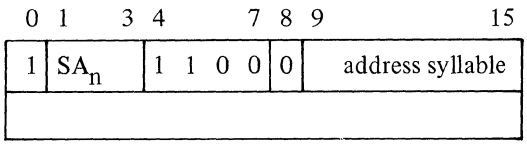
- o *Memory location:* the left-hand word of a double-word pair will be addressed. Auto-increment and auto-decrement modes of addressing will add (subtract) two from the base register. A value in the index register will count double words; it is shifted one place to the left before being applied. Auto-indexing will cause one to be added (subtracted) from the index register.
- o *Immediate operand:* this will be a double-word operand and thus the instruction will be three words in length.
- o *Register operand:* This form of addressing must address the right-hand register containing a double-word pair, i.e., either R3, which causes the contents of R6 and R7 to be stored in R2 and R3, or R5, which selects R4 and R5.

SDV

Instruction:
Scientific divide

Type:
Double Operand

Format:



Description:
Divides the contents of the scientific accumulator identified in the first operand by the contents of the location, scientific accumulator, or R register specified in the second operand. The result is saved in the scientific accumulator (except for the remainder, which is ignored).

If the second operand is =$R4, =$R5, =$R6, or =$R7, the integer value contained in the specific R register is internally converted to floating-point format before it is divided into the S register specified by the first operand.

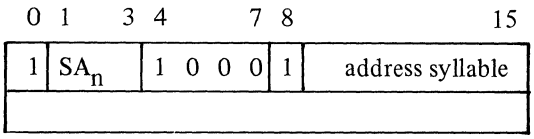
If the Scientific Instruction Processor (SIP) is not installed on this system, the Floating-Point Simulator, if present, is entered via trap vector #3.

SLD

Instruction:
Scientific load

Type:
Double Operand

Format:



Description:
Loads the contents of the location, scientific accumulator, or R register identified in the second operand into the scientific accumulator identified in the first operand.

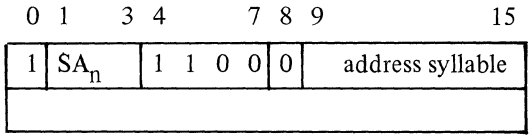
If the Scientific Instruction Processor (SIP) is not installed on this system, the Floating-Point Simulator, if present, is entered via trap vector #3.

SML

Instruction:
Scientific multiply

Type:
Double Operand

Format:



Description:
Multiplies the contents of the scientific accumulator identified in the first operand by the contents of the location, scientific accumulator, or R register specified in the second operand. The result is saved in the scientific accumulator.

If the second operand is an R register, the integer value contained in the specific R register is internally converted to floating-point format before it is multiplied to the S register specified by the first operand.

Scientific indicator settings include the following:

EU – Set to 1 on exponent underflow; otherwise, set to 0.

PE – Set to 1 if nonzero bits are lost during right shift; otherwise, set to 0.

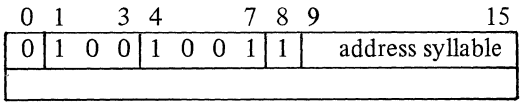
If the Scientific Instruction Processor (SIP) is not installed on this system, the Floating-Point Simulator, if present, is entered via trap vector #3.

SNGD

Instruction:
Scientific negate (2 words)

Type:
Single Operand

Format:



Description:
Negates the short-precision floating-point number at the location or scientific accumulator specified in the operand.

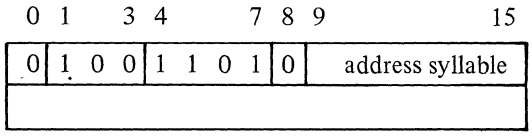
If the Scientific Instruction Processor (SIP) is not installed on this system, the Floating-Point Simulator, if present, is entered via trap vector #3.

SNGQ

Instruction:
Scientific negate (4 words)

Type:
Single Operand

Format:



Description:
Negates the long-precision floating-point number at the location or scientific accumulator specified in the operand.
obtained from bits 12 through 15 of general

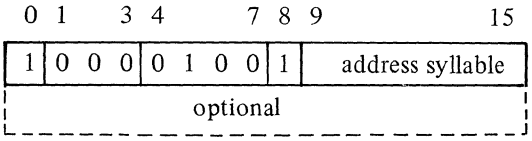
SID

Instruction:
Subtract double-word integer

Restriction:
Operates on 6/43 only.

Type:
Double Operand

Format:



Description:
Loads the difference of the double-word integer contained in R6 and R7 and the contents of the double-word location specified by the address syllable in R6 and R7.

R6(0) is considered the high-order bit
R7(15) is considered the low-order bit.

The contents of the I Register are affected as follows:

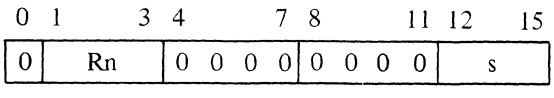
- o If carry C is set to 1; otherwise 0.
- o If overflow OV is set to 1; otherwise 1.

SOL

Instruction:
Single-shift open-left

Type:
Shift Short

Format:



s is number of positions shifted (0-15)

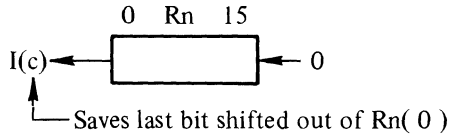
Description:
Shifts the contents of the R register, identified in the Rn field, left the number of bit positions specified in the s field. The bit positions vacated by the shift are filled with binary 0s.

If the s field contains 0s, the shift distance is

register R1. The contents of the I register are affected as follows:

- o C bit contains the last binary digit shifted out of the R register.

Pictorial Representation:

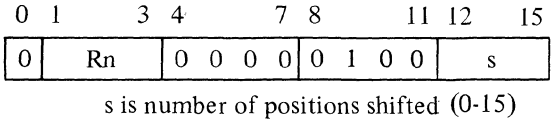


SOR

Instruction:
Single-shift open-right

Type:
Shift Short

Format:



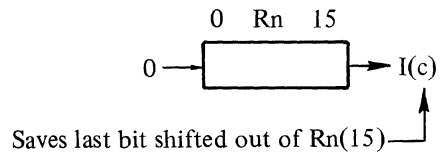
Description:
Shifts the contents of the R register, identified in the Rn field, right the number of bit positions specified in the s field. The bit positions vacated by the shift are filled with binary 0s.

The contents of the I register are affected as follows:

- o C bit contains the last binary digit shifted out of the R register.

If the s field contains 0s, the shift distance is obtained from bits 12 through 15 of general register R1.

Pictorial Representation:

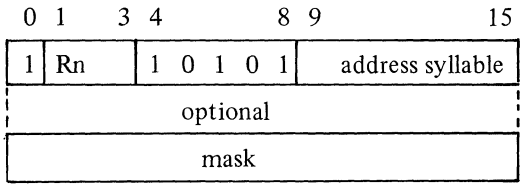


SRM

Instruction:
Store register masked

Type:
Double Operand

Format:



Description:
Stores contents of the designated register in the effective address under control of a 16-bit mask. Every bit (zero or one) in the register with a corresponding mask bit of one is stored; those corresponding to zero bits in the mask are not transferred and the original bits in the effective address remain unchanged.

The mask is contained in the word following the instruction. If it is all zeros, the contents of R1 are used as a mask. If R1 also contains all zeros, the instruction does no operation.

No indicators are affected. The address syllable can specify a memory location, an immediate operand, or another R register.

Example: Contents of:

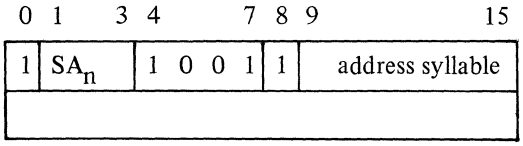
Register	Mask	Effective Address (before)	Effective Address (after)
$(ABCD)_{16}$	$(00FF)_{16}$	$(1234)_{16}$	$(12CD)_{16}$
$(ABCD)_{16}$	$(1111)_{16}$	$(1234)_{16}$	$(0325)_{16}$

SSB

Instruction:
Scientific subtract

Type:
Double Operand

Format:



Description:

Subtracts the contents of the location, scientific accumulator, or R register identified in the second operand from the contents of the scientific accumulator specified in the first operand. The result is saved in the scientific accumulator.

If the second operand is an R register, the integer value contained in the specific R register is internally converted to floating-point format before it is subtracted from the S register specified by the first operand.

Scientific indicator settings include the following:

EU – Set to 1 on exponent underflow; otherwise, set to 0.
PE – Set to 1 if nonzero bits are lost during right shift; otherwise, set to 0.

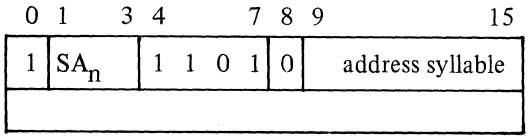
If the Scientific Instruction Processor (SIP) is not installed on this system, the Floating-Point Simulator, if present, is entered via trap vector #3.

SST

Instruction:
Scientific store

Type:
Double Operand

Format:



Description:

Stores the contents of the scientific accumulator identified in the first operand in the location, scientific accumulator, or R register specified in the address expression.

If the second operand is an R register, the floating-point value contained in the specific scientific accumulator is converted to integer format before it is stored into the specified R register.

Scientific indicator settings include the following:

EU – Set to 1 on exponent underflow; otherwise, set to 0.
SE – Set to 1 if resultant floating-point value has a zero fraction; otherwise, set to 0.
PE – Set to 1 if nonzero bits are lost during right shift; otherwise, set to 0.

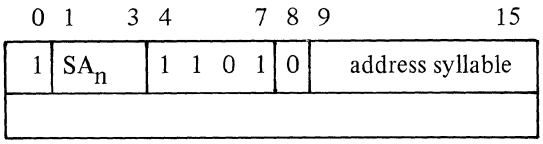
If the Scientific Instruction Processor (SIP) is not installed on this system, the Floating-Point Simulator, if present, is entered via trap vector #3.

SSW

Instruction:
Scientific swap

Type:
Double Operand

Format:



Description:

Swaps the contents of the scientific accumulator identified in the first operand with the contents of the location, scientific accumulator, or R register specified in the address expression.

If an R register is specified as the second operand, the value specified by the first operand is internally converted to integer format, and the value specified by the second operand is internally converted to floating-point. These converted values are then interchanged.

Scientific indicator settings include the following:

EU – Set to 1 on exponent underflow; otherwise, set to 0.
SE – Set to 1 if resultant floating-point value has a zero fraction; otherwise, set to 0.
PE – Set to 1 if nonzero bits are lost during right shift; otherwise, set to 0.

If the Scientific Instruction Processor (SIP) is not installed on this system, the Floating-Point

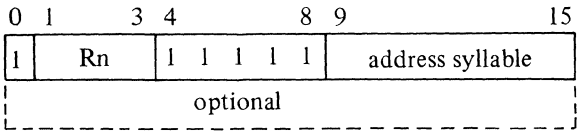
Simulator, if present, is entered via trap vector #3.

STB

Instruction:
Store B register

Type:
Double Operand

Format:



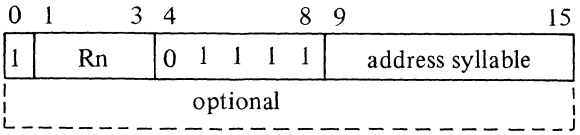
Description:
Stores the word contained in the designated B register into the location specified by the effective address. No indicators are affected. The address syllable can specify a memory location, an immediate operand, or another B register.

STH

Instruction:
Store R register half-word (byte)

Type:
Double Operand

Format:



Description:
Stores the right-hand byte of the designated R register into the half-word specified by the effective address. The other half is not changed. No indicators are affected.

The effective address can specify a memory location, an immediate operand, or another R register, with the byte being positioned as follows:

– memory location, not indexed	– left byte
– memory location, indexed	
index value even	– left byte
index value odd	– right byte
– immediate operand	– left byte
– R register	– right byte

Example: Assume that:

– register B1	contains 1000
– register R1	contains 0
– register R2	contains 1
– register R5	contains (ABCD)
– location 1000	contains $(1234)_{16}$

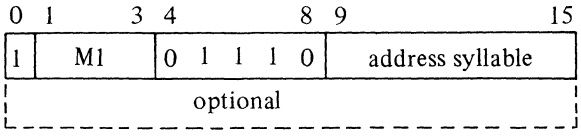
then if STH $R5, $B1.$R1 is executed: location 1000 will contain $(CD34)_{16}$
but if STH $R5, $B1.$R2 is executed: location 1000 will contain $(12CD)_{16}$

STM

Instruction:
Store M1 register

Type:
Double Operand

Format:



Description:
Stores the 8-bit M1 register identified in the first operand in the right half-word of the location specified in the address syllable; the left half-word of the location is filled with 1s.

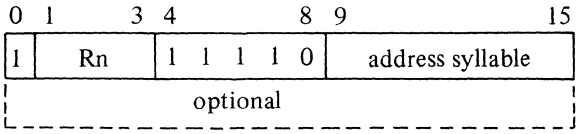
The address syllable can specify a memory location, an immediate operand, or one of the R registers. No indicators are affected. The register number must be 1; otherwise, traps to trap vector #5.

STR

Instruction:
Store R register

Type:
Double Operand

Format:



Description:

Stores the word contained in the designated R register into the location specified by the effective address. No indicators are affected. The address syllable can specify a memory location, an immediate operand, or another R register.

STS

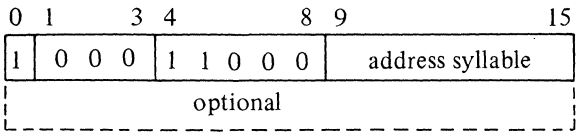
Instruction:

Store S register

Type:

Single Operand

Format:



Description:

Stores the contents of the status register in the location specified by the effective address. No indicators are affected. The address syllable can specify a memory location, an immediate operand, or an R register.

STT

Instruction:

Store Stack Register

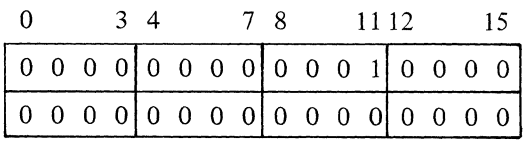
Restriction:

Operates on 6/43 only.

Type:

Generic

Format:



Description:

Stores the contents of the Stack Address Register (T) into B7. The contents of the I Register are unaffected.

SUB

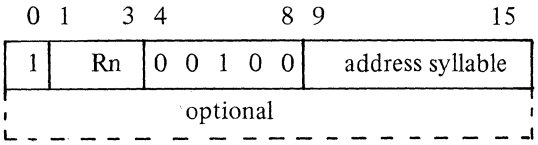
Instruction:

Subtract from R register

Type:

Double Operand

Format:



Description:

Subtracts the word at the effective address from the word contained in the designated R register. The carry (C) and overflow (OV) indicators are set if carry and/or overflow occurs respectively; otherwise they are reset to zero. The address syllable can specify a memory location, an immediate operand, or another R register.

Examples:

R Register (before)	Effective Address (before & after)	R Register (after)	C (after)	OV (after)
-32767	+1	-32768	0	0
-32768	+1	+32767	1	1
-1	-1	0	1	0

SWB

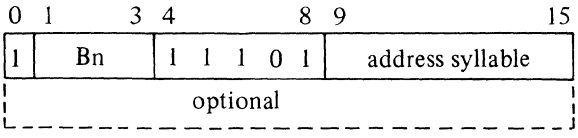
Instruction:

Swap B register

Type:

Double Operand

Format:



Description:

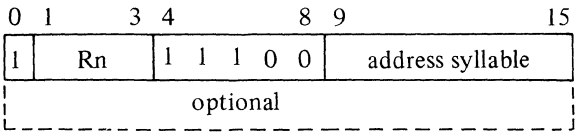
Swaps the word contained in the designated B register with the word at the effective address. No indicators are affected. The address syllable can specify a memory location, an immediate operand, or another B register.

SWR

Instruction:
Swap R register

Type:
Double Operand

Format:



Description:

Swaps the word contained in the designated R register with the word at the effective address. No indicators are affected. The address syllable can specify a memory location, an immediate operand, or another R register.

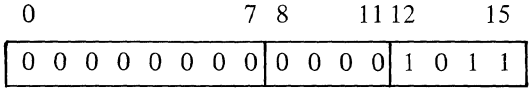
VLD

Instruction:
Validate access rights

Restriction:
Operates only on 6/43 with Memory Management Unit option.

Type:
Generic

Format:



Description:

Determines whether or not certain data accesses are valid without actually attempting them. It accepts a virtual address, interpreted as the start of an area to be verified; a size, in 256-word blocks; and an effective ring number to be used in the validation. These items are loaded into base register B5, general register R3 and general register R5, respectively, before execution. The instruction returns a condition code in R3 as follows:

- -1 – Invalid segment. Segment does not exist or is not large enough for the data area specified.
- 0 – Read access permitted in given ring, write access forbidden.
- 2 – Read access and write access both permitted in given ring.
- -2 – Read access and write access both forbidden in given ring.

This instruction is useful, for example, in a system subroutine that executes at a high level of privilege and has access to critical data. When called on behalf of a user, the system subroutine can validate that user's right to access the data in question by using the VLD instruction.

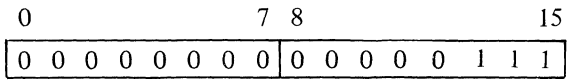
The contents of the I Register are unaffected.

WDTF[3]

Instruction:
Watchdog timer off

Type:
Generic

Format:



Description:

Disables the watchdog timer. The P bit in the S register must be set to 1 (i.e., the central processor must be in the privileged state) for this instruction to be executed. If not, the unprivileged use of a privileged operation is signified by a trap to trap vector #13.

WDTN[3]

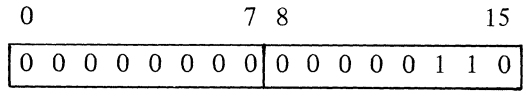
Instruction:
Watchdog timer on

---

[3] Optional on 6/36; standard on 6/40 models.

Type:
Generic

Format:



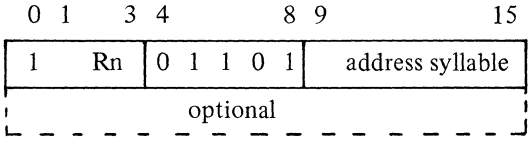
Description:
Enables watchdog timer. The P bit in the S register must be set to 1 (i.e., the central processor must be in the privileged state) for this instruction to be executed. If not, the unprivileged use of a privileged operation is signified by a trap to trap vector #13.

XOH

Instruction:
Half-word (byte) exclusive OR with R register

Type:
Double Operand

Format:



Description:
Logically "exclusive ORs" the byte at the effective address (with sign extended) to the word contained in the designated R register. No indicators are affected. The address syllable can specify a memory address, an immediate operand, or another R register, with the byte being positioned as follows:

– memory location, not indexed	– left byte
– memory location, indexed	
index value even	– left byte
index value odd	– right byte
– immediate operand	– left byte
– R register	– right byte

The following chart illustrates the result of exclusively ORing bits:

First operand bit	0	0	1	1
Second operand bit	1	0	1	0
Result	1	0	0	1

Examples:

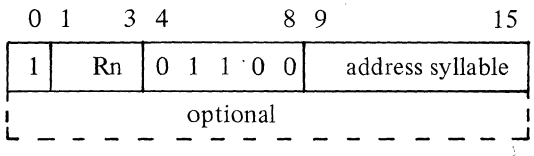
R Register (before)	Effective Address	R Register (after)
$(0000)_{16}$	$(07)_{16}$	$(0007)_{16}$
$(0007)_{16}$	$(07)_{16}$	$(0000)_{16}$
$(FFFF)_{16}$	$(88)_{16}$	$(0077)_{16}$

XOR

Instruction:
Exclusive OR with R register

Type:
Double Operand

Format:



Description:
Logically "exclusive ORs" the word at the effective address to the word contained in the designated R register. No indicators are affected. The address syllable can specify a memory address, an immediate operand, or another R register.

The following chart illustrates the result of exclusively ORing bits:

First operand bit	0	0	1	1
Second operand bit	1	0	1	0
Result	1	0	0	1

Examples:

R Register (before)	Effective Address	R Register (after)
$(0000)_{16}$	$(0007)_{16}$	$(0007)_{16}$
$(0007)_{16}$	$(0007)_{16}$	$(0000)_{16}$
$(FFFF)_{16}$	$(8888)_{16}$	$(7777)_{16}$

SECTION 5

CENTRAL PROCESSOR CONTROL PANELS

This section describes the two kinds of operator control panels -- full and basic -- offered on Level 6 systems, panel options, controls and indicators, and the various functions that the operator can perform with these control panels.

PACKAGING

The physical configuration of the control panel is dependent upon the packaging of the central processor. Table-top models and systems in full (60-inch) cabinets will have protruding panels with the controls and indicators on an incline. Systems in the 30-inch "mini-rack" will have a flat control panel mounted vertically inside the front-door of the cabinet. Systems in the desk-style office packaging may have a flat control panel similar to the mini-rack, or a control panel mounted in the desk top behind the CRT keyboard. In the latter case, only a full control panel is allowed; the others may have either full or basic panels.

OEMs and system builders who wish to install a rack-mountable unit into their own cabinetry will get the protruding inclined panel unless they also order the vertical panel mounting option.

The 6/40 models have more logic and indicators on their control panels than do 6/30 models. An upgrade from a 6/30 to a 6/40 thus requires that the control panel be replaced.

FULL CONTROL PANEL

Systems being used for program development and testing should be equipped with the full control panel (or basic control panel with the portable plug-in panel option). Full panel functionality is also required for maintenance purposes in systems that do not have a system console (CRT, teleprinter, or keyboard printer).

The full panel allows certain CP registers and the entire memory contents to be entered and displayed. It controls, in a step-by-step fashion, the system initialization sequence by single-stepping a program and stopping and starting program execution.

Figure 5-1 shows the full control panel. Table 5-1 lists and describes the various controls and indicators.

Register Display

The registers – 21(6/30) and 26(6/40) – may be accessed from the control panel. A hexadecimal display located in the upper center portion of the control panel, in the area labeled REGISTER, accommodates this capability. The register display is divided into two sections: one labeled LOCATION, the other CONTENTS.

- o LOCATION – A 2-digit hexadecimal display that indicates the coded location of the specific register selected. The selection codes assigned to access the visible registers are listed alphanumerically in Figure 5-2. A complete list of the selection codes is also stenciled on the control panel in the upper right-hand portion of the panel labeled REGISTERS.
- o CONTENTS – A 4-digit (6/30)/5-digit (6/40) hexadecimal display that indicates the contents of the selected register. The specific visible register type/number associated with the assigned selection code is listed in the contents column of Figure 5-2.

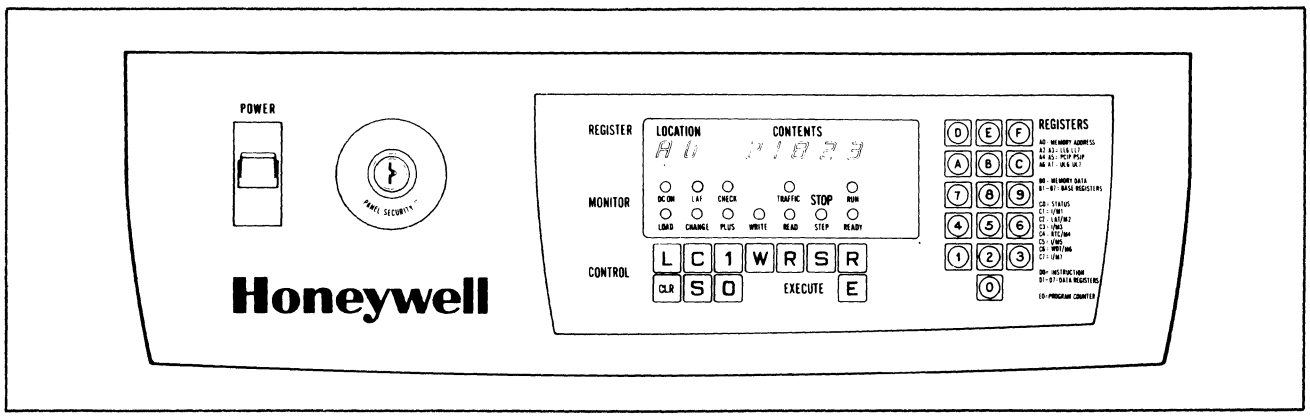


Figure 5-1. Full Control Panel (Shown with 6/40 Panel)

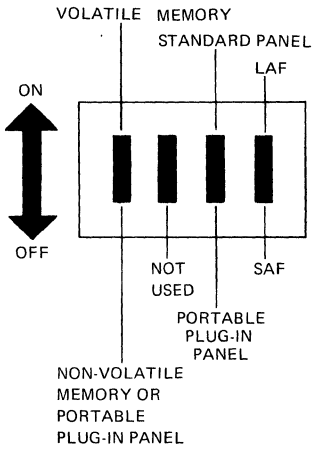
TABLE 5-1. FULL PANEL CONTROLS AND INDICATORS

Control/Indicator	Description
POWER (switch)	– Up for power on; down for power off.
PANEL SECURITY (switch)	– Left (locked) position disables panel switches and the register display (not lit) and push buttons except for POWER; right (unlocked) position enables panel switches/push buttons, and displays.
DC ON (indicator)	– Indicates dc power is applied to the CP.
CHECK (indicator)	– Indicates that one of the individual units (CP, controller, etc.) is executing QLTs or an error was encountered during QLT execution.
TRAFFIC (indicator)	– Indicates the CP is executing instructions other than a Halt.
RUN (indicator)	– Indicates CP is executing any instruction, including a Halt. If TRAFFIC is off and RUN is on, the CP is continually executing a halt.
LOAD (indicator)	– Indicates the CP is in bootload mode.
CHANGE (indicator)	– Indicates control panel is in change mode, i.e., capable of modifying register contents.
PLUS (indicator)	– Indicates CP is in increment memory address mode. When off, memory address register is not modified during memory read or write mode.
WRITE (indicator)	– Indicates control panel is in memory write mode.
READ (indicator)	– Indicates control panel is in memory read mode.
STOP/STEP (indicator)	– Indicates CP is in single instruct mode.
READY (indicator)	– Indicates CP is in ready mode. Pressing the Execute key causes CP to go to run mode.
LAF (Long Address Form)[1] (indicator)	– Indicates CP is operating in the Long Address Form (i.e., 20-bit main memory addressing). It is also lit during system load independent of the setting of the LAF switch. If not lit, CP is operating in the SAF mode.
L (Load) (push button)	– Places the processor in load mode, lights Check indicator, activates QLT, and allows bootstrapping of the bootstrap record into memory. Used in conjunction with Execute so that when Execute is pressed next, the QLT should be executed. Upon subsequent pressing of the Execute key the bootstrapping is actually performed.
C (Change) (push button)	– Places the processor in change mode. In this mode the processor is ready to accept modifications to the contents of the selected register from the control panel. NOTE: Not all visible registers are modifiable.
1 (Plus One) (push button)	– Places the processor in plus 1 mode. In this mode the processor is ready to increment its address register before reading or writing successive memory locations from the control panel. This condition is initiated only after setting the processor to either the read or write mode. When in the plus 1 mode, each pressing of Execute causes the memory address register (A0) to be incremented by 1, prior to its being used.
W (Memory Write) (push button)	– First places the processor in a stop state (if not already in that state); resets the plus 1 mode (if in plus 1 mode); resets the load mode (if in load mode); and places the processor in write mode. In this mode the processor writes the contents of the selected register into the location addressed by the memory address register (A0), when Execute is pressed. (If A0 is selected, the contents of B0 are written.)

TABLE 5-1 (CONT). FULL PANEL CONTROLS AND INDICATORS

Control/Indicator	Description
R (Memory Read) (push button)	– First places the processor in a stop state (if not already in that state); resets the plus 1 mode (if in plus 1 mode); and places the processor in read mode. In this mode the processor reads the contents of the location addressed by the memory address register (A0) into the selected register when Execute is pressed. (If A0 is selected, the contents are read into B0).
S (Stop) (push button)	– Stops instruction execution and places the processor in a stop state. When it is in the stop state, a variety of operating procedures are possible from the control panel. Also, when in the stop state, the processor is automatically in the step mode. In this mode one instruction is executed each time Execute is pressed, thus permitting single stepping through a program.
R (Ready) (push button)	– Places the processor in the ready mode. In this mode the processor is ready to execute a series of instructions constituting a program. If Execute is pressed, the processor enters the run state and commences execution of the program.
CLR (Master Clear) (push button)	– Initiated by pressing Clear but is only effective while in the stop state. Pressing Clear invokes a number of clearing and initializing functions: 1. Clears and sets to 0 the P register, the M1 register (M2-M7), and the instruction register; does not clear to 0, but changes the SIP (P register) and SI register. Modifies the scientific accumulators. 2. Clears all pending interrupts. 3. Resets the real-time clock (RTC) and the watchdog timer (WDT). 4. Sets the ring number to zero and the interrupt priority level to 0. 5. Starts the Quality Logic Test (QLT) in each controller and SIP.
S (Select) (push button)	– Places the processor in select mode. In this mode the desired register that is to be displayed or operated on is selected by keying in the proper selection code from the hexadecimal-pad keys. The select mode may be initiated in any state.
0 (Plus Zero) (push button)	– Resets the plus 1 mode. In this mode the memory address register is not modified during a memory read or write operation.
E (Execute) (push button)	– Performs various execution functions depending on the mode that the processor is in prior to pressing Execute. In the *ready* mode, pressing Execute places the processor in a run state and it executes instructions starting with the instructions in the instruction register. If the instruction register contains a zero, execution begins at the address specified in the P register. Execution continues until a Halt instruction is encountered or Stop, Read, or Write is pressed. In the *step* mode, pressing Execute causes the execution of a single instruction. The processor returns to the stop mode after each single instruction execution. In the *read* or *write* mode, pressing Execute at the appropriate time causes the selected memory location to be displayed or changed. In a *load* mode, pressing Execute causes initiation of a bootstrap operation.
Configuration Switches[1]	– Four tiny rocker switches, located behind the full control panel on the control panel circuit board supply configuration information to the central processing unit. The switch on the extreme left is the volatile memory switch which should be set to "on" if the memory is volatile (i.e., not core and no Memory Save and Autorestart option) or the Memory Management Unit

TABLE 5-1 (CONT). FULL PANEL CONTROLS AND INDICATORS

Control/Indicator	Description
	option is present. In the "on" position an auto bootload will occur on powering up after a power failure. The switch should be set to "off" if core memory is present and no Memory Management Unit option is present *or* if this is a portable plug-in panel. In the "off" position an auto restart will occur on powering up after a power failure (if the Memory Save and Autorestart option is present). The second switch is not used. The third switch should be set to "on" if it is a standard control panel or to "off" if it is a portable plug-in panel. The switch on the extreme right is the LAF switch which should be set to "on" to put the CP in the LAF mode or to "off" for the SAF mode. The system must first be initialized by pressing the Clear key (this prevents the address from being changed while a program is running). After switching modes, the system must be cleared *twice*. 

[1] 6/40 full panel only

The positional format of the location and contents display is represented by key symbols H1 through H7 at the bottom of Figure 5-2.

Hexadecimal-Pad Keys

The set of 16 hexadecimal keys in the right part of the control panel marked REGISTERS is called the hex pad. These keys provide access to the user-visible registers. In the select mode, a hex pad key-in selects the register to be operated on, and the entered digits light up under LOCATION in the register display. In the change mode, a hex pad key-in changes the contents of the selected register, and the entered digits are displayed under CONTENTS in the register display. Each keystroke shifts and loads the selected hexadecimal digit into the least significant hexadecimal position of the selected register; all other digits are shifted to the left.

In the select mode, input from the hexadecimal-pad keys selects the register to be displayed/operated on. This input is simultaneously displayed in the LOCATION field on the display. Hexadecimal-pad keys 8 through F modify the leftmost location character (H1); hexadecimal-pad keys 0–7 modify the rightmost location character (H2).

In the change mode, input from the hexadecimal-pad keys changes the contents of the selected register. This input is simultaneously displayed in the CONTENTS field of the register display. Each key stroke shifts and loads the corresponding hexadecimal character into the least significant hexadecimal position of the selected register and the display. Specifically, each hexadecimal-pad key stroke enters the new character into the H6 (H7) position of the selected register and the CONTENT field of the register display. At the same time, the currently displayed characters (in all four/five positions) shift one

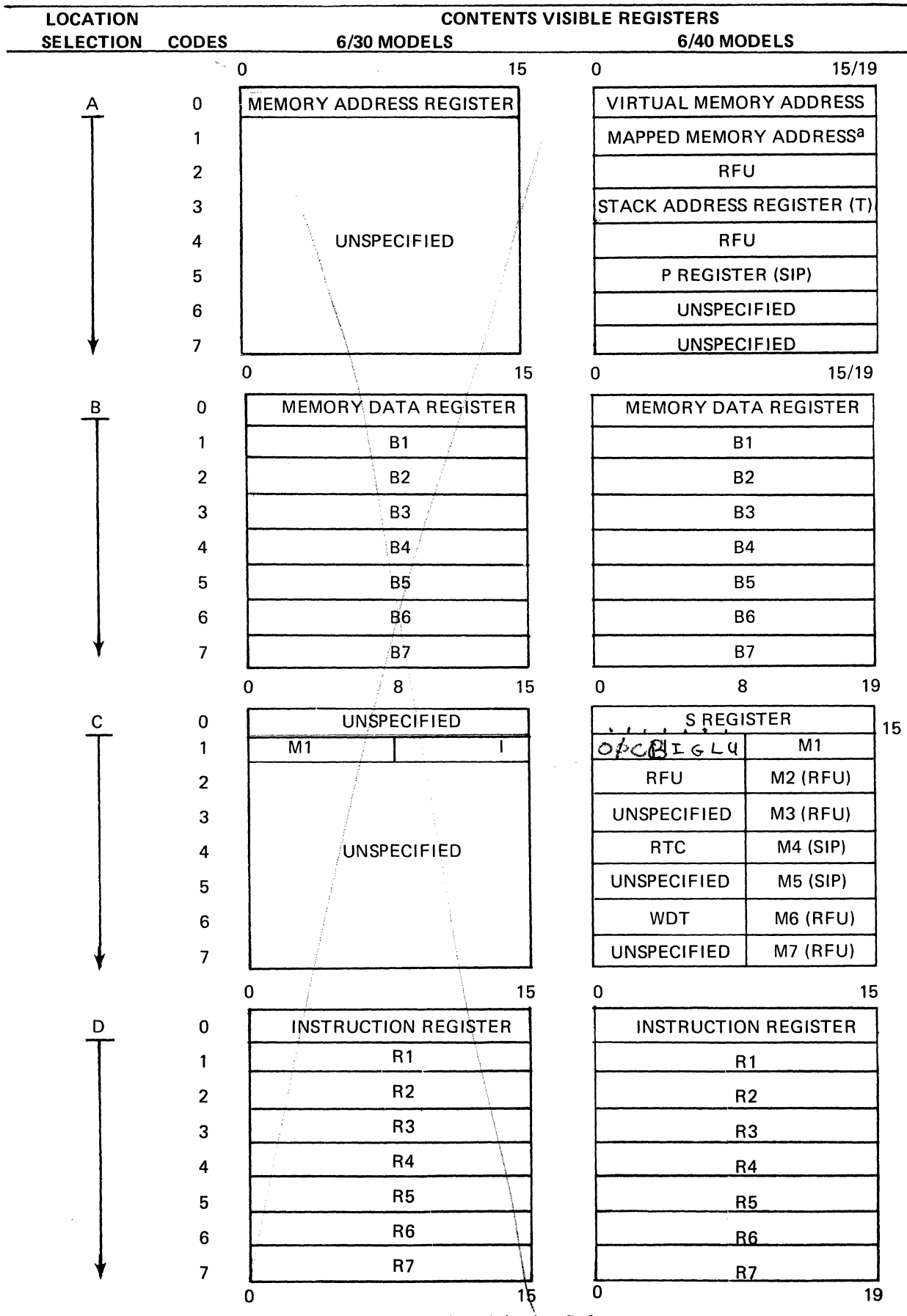
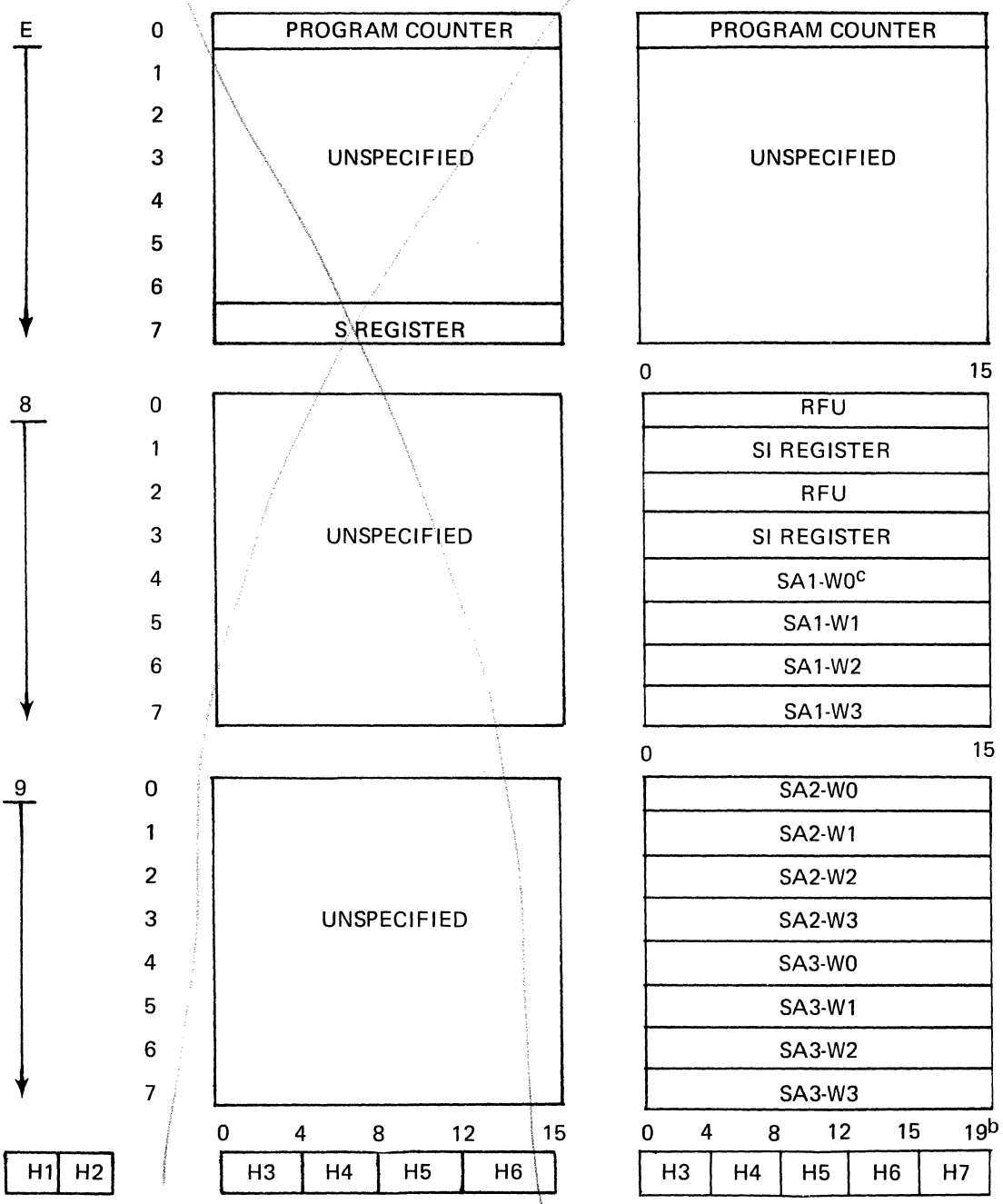


Figure 5-2. Register Selection Codes



[a] If memory management unit (MMU) present; otherwise unspecified.
[b] In the SAF mode, the high-order digit of the 5-digit display must be zero.
[c] Contains exponent, sign, and high-order 8 bits of mantissa.
RFU = Reserved for future use.

Figure 5-2 (cont). Register Selection Codes

position to the left.

The activity resulting from key strokes is illustrated as follows:

$$\text{out} \leftarrow H3 \leftarrow H4 \leftarrow H5 \leftarrow H6 \leftarrow (H7) \leftarrow \text{enter}$$

You may take advantage of this shifting function and key in only a limited number of characters to arrive at the contents desired. For example, if the current CONTENTS display shows 4032 and you wish to change the contents to 3275, then key in only the characters 7,5.

Panel Display Interpretation

The CONTENTS field on the display panel is shown in hexadecimal notation. The 4-(5-) character hexadecimal display value represents the binary value of the 16-(20-) bit visible register. Each hexadecimal character is equivalent to a binary value of four bits. Thus if the display shows a value of (0)4CA2, this hexadecimal value represents the stored binary value of:

$$\left(\frac{0000}{0}\right) \quad \frac{0100}{4} \quad \frac{1100}{C} \quad \frac{1010}{A} \quad \frac{0010}{2}$$

For most registers, the display value is usable in hexadecimal form and does not need to be converted to binary. However, the exceptions are the M1, (and M2–M7 in 6/40), I, and S registers. These registers specify various status, security, and control indicators on a bit basis. Therefore, to properly interpret these registers, you must convert their hexadecimal displays to binary. Refer to Table 5-2 for a list of hexadecimal/binary/decimal conversions for the first 16 digit values.

TABLE 5-2. HEXADECIMAL/BINARY/ DECIMAL CONVERSION

Hexadecimal	Binary	Decimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11
C	1100	12
D	1101	13
E	1110	14
F	1111	15

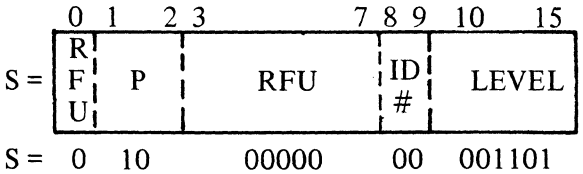
As an example, assume you wish to analyze the contents of the S register, e.g., on a 6/30 model. After you have selected and keyed in the proper selection code (E7), the display panel shows the following hexadecimal character display:

LOCATION	CONTENTS
E7	400D

For meaningful interpretation of its contents, you must first convert the hexadecimal value to binary:

$$\frac{\text{hexadecimal}}{400D} = \frac{\text{binary}}{0100\ 0000\ \ 0000\ 1101}$$

Next, you must overlay this binary value onto the bit fields structured for the S register:



You can now analyze the system status and security indicators:

- o P = Privilege State (Ring Number)
  For 6/30 and 6/40 (without MMU):
  00 = User State
  11 = Privilege State

  For 6/40 (with MMU):
  11 = Ring 0 (Privilege)
  10 = Ring 1 (Privilege)
  01 = Ring 2 (User)
  00 = Ring 3 (User)

  NOTE: Privileges and access rights accorded the various rings are in inverse order to the ring number (i.e., Ring 0 is the most privileged).
- o RFU = reserved for future use.
- o ID# (processor identity) = 00 indicates CP channel 0
- o LEVEL (interrupt priority level) = 001101 when converted to decimal (binary 001101 = decimal 13) indicates the actual priority level.

BASIC CONTROL PANEL

The basic control panel is an important feature for customers who do not want their operators at remote locations to be able to modify the software. It is also less expensive than the full panel. As an option to the basic control panel, Honeywell offers a portable plug-in panel (see "Options"). By plugging this unit into the basic panel, full panel functionality is effected. A single portable plug-in panel can support a multitude of systems with basic control panels.

For security, the basic panel does not allow visibility to CP registers or main memory. Its only control capability is to initiate and control the system initialization procedure. This includes clearing the system, running quality logic tests (QLTs), and invoking either automatic restart (if memory contents are preserved) or bootload (if powering up the system initially or if the Memory Save and Autorestart option is not present). Displays indicate gross system status. The basic panel also provides a connector to facilitate interfacing the portable plug-in panel. Figure 5-3 shows the basic panel layout for 6/30 models and Figure 5-4 for 6/40 models. Table 5-3 describes its controls and indicators. Table 5-4 illustrates and interprets various LED indicator combinations.

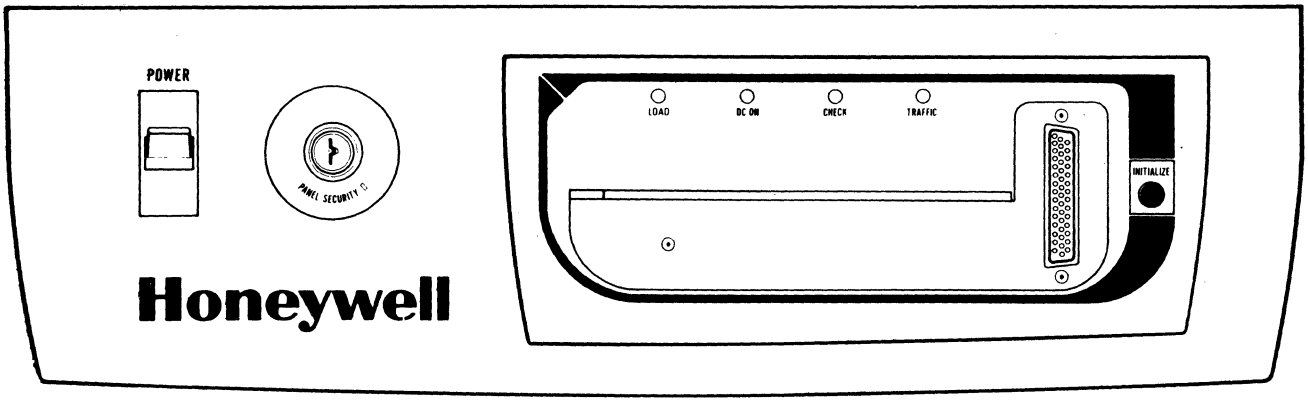


Figure 5-3. Basic Control Panel for 6/30 Models

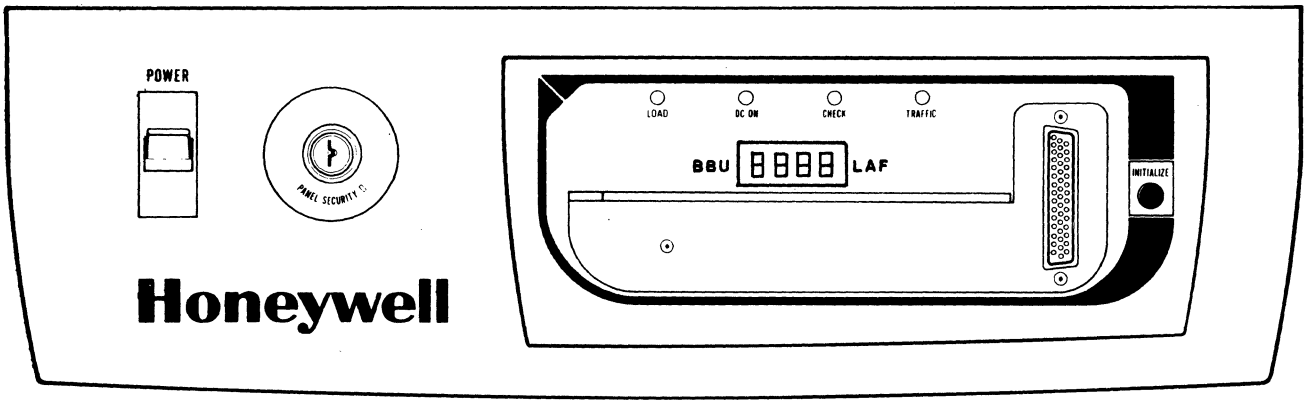
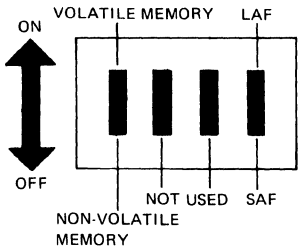


Figure 5-4. Basic Control Panel for 6/40 Models

TABLE 5-3. BASIC PANEL CONTROLS AND INDICATORS

Control/Indicator	Description
POWER (switch)	— Up for power on; down for power off.
PANEL SECURITY (switch)	— Left (locked) position disables panel except for POWER; right (unlocked) position enables panel.
LOAD (indicator)	— Indicates CP is in bootload mode.
DC ON (indicator)	— Indicates dc power is applied to the system.
CHECK (indicator)	— Indicates CP is executing QLTs or an error was encountered during QLT execution.
TRAFFIC (indicator)	— Indicates CP is executing instructions other than a Halt.
INITIALIZE (push button)	— When pressed and the panel is unlocked, the following sequence occurs: o DC clear of system o QLTs are initiated and run o Bootload is performed over channel 0400_{16} into location 0100_{16} o CP begins instruction execution at location 0100_{16}
LAF[a] (indicator)	— Indicates CP is operating in the long address form.

TABLE 5-3 (CONT). BASIC PANEL CONTROLS AND INDICATORS

Control/Indicator	Description
CONFIGURATION SWITCHES[a]	— Four tiny rocker switches, located behind a sliding door on the front of the panel (see Figure 5-4), supply configuration information to the central processing unit. NOTE: If the portable plug-in panel is inserted, it must be removed to access these switches. The portable plug-in panel also has configuration switches which must be set (see Table 5-1). The switch on the extreme left is the volatile memory switch which should be set to "on" if the memory is volatile (i.e., not core and no Memory Save and Autorestart option) or the Memory Management Unit option is present. In the "on" position an auto bootload will occur on powering up after a power failure. The switch should be set to "off" if core memory is present and no Memory Management Unit option is present. In the "off" position an auto restart will occur on powering up after a power.failure (if the Memory Save and Autorestart option is present). The two center switches are not used. The switch on the extreme right is the LAF switch which should be set to "on" to put the CP in the LAF mode or to "off" for the SAF mode. The system must first be initialized by pressing the Clear key (this prevents the address from being changed while a program is running). After switching modes, the system must be cleared *twice*. 

[a]6/40 basic panel only.

TABLE 5-4. BASIC CONTROL PANEL INDICATOR INTERPRETATION

State/Occurrence	Indicators Check	Load	Traffic
Normal Halt State	●	●	●
Normal Run State	●	●	○
Peripheral QLT Fault	○	●	◐
Load Fault	●[a]	○	●
Load in Process	●[a]	○	○
Memory Fault Detected	○	○	●
CP QLT Fault	○	○	○

LEGEND:
○ = ON ● = OFF ◐ = EITHER

[a]Could be ON if a particular peripheral controller QLT has a QLT fault condition stored in it.

OPTIONS

Portable Plug-in Panel

This is a self-contained, full control panel that can plug into any basic panel. It provides security, economy, and flexibility of operation in multi-system environments. See Figure 5-5.

Option No.	Model No.
CPF9408	6/34, 6/36
CPF9504	6/43

Prior to removing the portable panel, the basic panel and then the portable panel are locked (up position). The portable panel may then be removed. If this sequence is not properly followed, the insertion/removal of the portable plug-in panel can cause program disruption.

The portable plug-in panel also has a lock (toggle switch on right-hand side) that must be activated in the proper sequence in conjunction with the lock on the basic panel. Prior to inserting

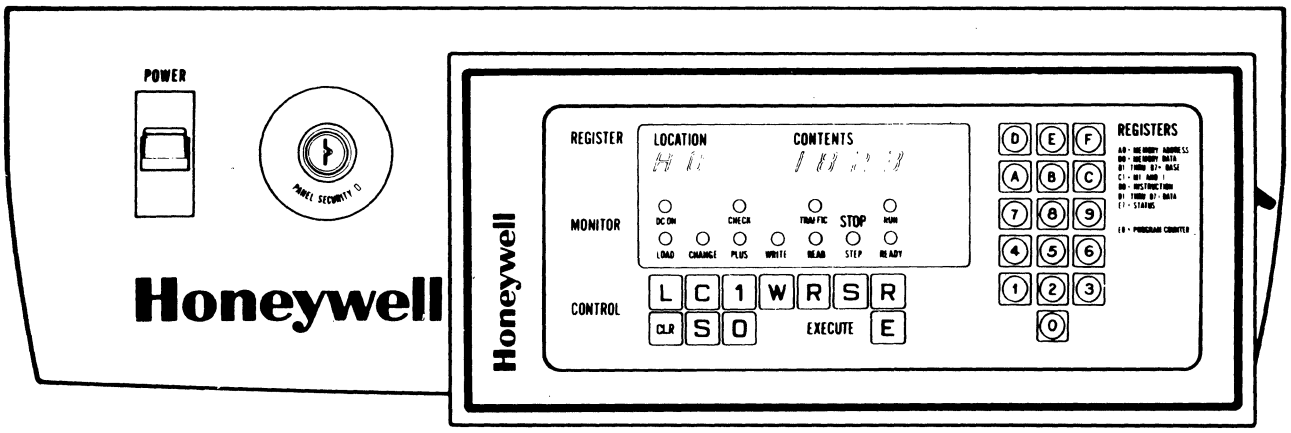


Figure 5-5. Portable Plug-in Panel Option (Shown with 6/30 Panel Plugged into Basic Panel)

the portable panel, the basic panel should be locked. The portable panel with the toggle switch in the "up" (locked) position can then be plugged in. If the "traffic" states between the two panels do not correspond, the portable panel is unlocked by setting the toggle switch to the "down" position and pressing the run or execute key (as appropriate) and relocking it. The basic panel is unlocked followed by the portable panel and then the desired operations are performed.

Vertical Panel Mounting

This is available for any rack-mountable system where physical space limitations exist and replaces the standard inclined panel mounting. It is designed for OEMs with their own cabinetry. See Figure 5-6.

Option No.	Model No.
CPF9407	6/34, 6/36, 6/43

ORDERING INFORMATION

Systems are equipped with either basic or full control panels depending on the processor type number as follows:

	Processor Type No.	
Model No.	With Full Panel	With Basic Panel
6/34	CPS9451/9453	CPS9450/9452
6/36	CPS9460/9461	CPS9462/9463
6/43	CPS9550/9551	CPS9552/9553

CONTROL PANEL OPERATING PROCEDURES

The functions that can be performed from the full control panel are governed by the operation of the processor in its two major states. For example, in relation to program execution, when the processor is in the run state, only the limited actions of displaying registers and executing programs are possible from the control panel. When the processor is in the stop state, the actions possible are much more extensive; they consist of displaying/changing memory, displaying/changing registers, executing single instructions, restarting programs, and master clearing the processor.

Before you perform any operation from the panel (except for power on/off), you must first unlock the control panel using the panel security switch.

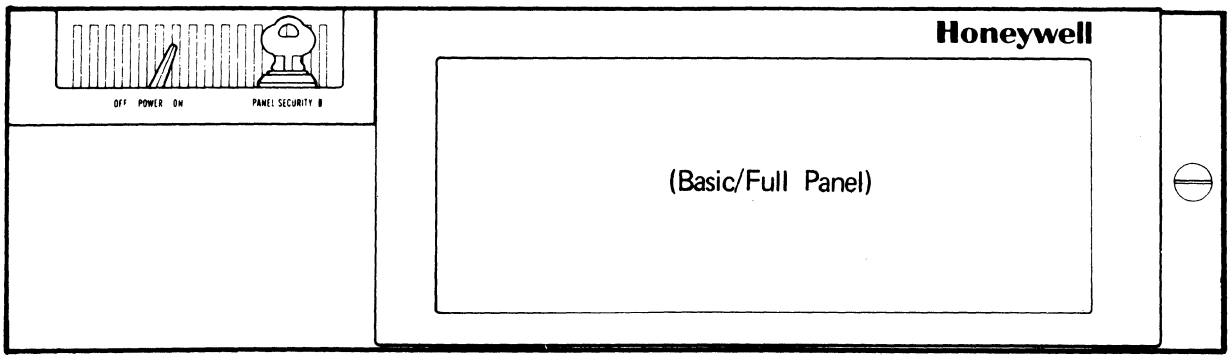


Figure 5-6. Vertical Panel Mounting Option

NOTE: In order for the Autorestart option to be initiated after a power failure, the control panel must be locked. For systems without this option, a reboot-strap will occur after a power failure if the control panel is locked or if the system has a basic control panel either locked or unlocked. In addition to automatically rebootstrapping from the device on channel 0400_{16}, the system will also start executing the program.

Display Memory

Any memory location may be accessed and displayed on the control panel. However, memory can be displayed only when the processor is in a stop state. The memory address register (A0) is the only visible register that can be used to access memory locations from the control panel. Any visible data register may be used for reading/displaying memory data, but by convention the memory data register (B0) is usually used for this purpose to preserve program integrity and will be so used in the procedure.

The following procedure describes a method for displaying the contents of one memory location and, as an option, displaying the contents of subsequent memory locations.

1. Press Read.
   Initially places the processor in a stop state, if the processor is not as yet in this state (the STOP/STEP indicator lights if not already lit). Then the processor is placed in read mode and instructed that the contents of the memory location addressed by the memory address register (A0) are to be displayed. The READ indicator lights when the Read control key is pressed.
2. Press Select.
   Places the processor in select mode as a necessary preliminary to selecting the memory address register.

   NOTE: This step is not necessary unless the CHANGE indicator is lit.

3. Press hexadecimal-pad keys A,0 to enter the 2-digit selection code for the memory address register.
   Digits A0 appear in the LOCATION field of the REGISTER display.
4. Press Change.
   Places the processor in change mode preparatory to keying in the address of the memory location to be displayed. The CHANGE indicator lights when the Change control key is pressed.
5. Key in, via the hexadecimal-pad keys, the 4 or 5-digit hexadecimal value representing the address of the memory location to be read. (For 6/40 models, note that 5 digits must be used even in SAF mode, in which case the most significant digit must be set to zero.)
   This address appears in the CONTENTS field of the REGISTER display.
6. Press Select.
   Returns the processor to select mode as a necessary preliminary to selecting the memory data register. The CHANGE indicator turns off when the Select control key is pressed.
7. Press hexadecimal-pad key B to enter the first digit of the 2-digit selection code for the memory data register. The second digit need not be entered again at this point since the hexadecimal character 0 in position H2 is still actively engaged from step 3.
   Digits B0 appear in the LOCATION field of the REGISTER display.
8. Press Execute.
   The data contents of the selected memory location are loaded into the selected register (B0) and displayed in the CONTENTS field of the REGISTER display.
9. If successive memory locations are to be displayed using the current address of the memory address register as a base, press Plus 1.
   The PLUS indicator lights when the Plus 1 control key is pressed.
10. Press Execute.
    The memory address register is incremented by 1 before accessing memory and the memory data of the succeeding memory location appears in the CONTENTS field of the REGISTER display.
11. Repeat step 10 for each sequential memory location to be displayed.

Change Memory

Any memory location may be accessed and changed from the control panel. However, memory can be changed only when the processor is in a stop state. As mentioned previously, the memory address register (A0) is the only visible register than can be used to access memory locations from the control panel. Any visible data register may be used for writing/changing memory data, but by convention the memory data register (B0) is usually used for this purpose to preserve

program integrity and will be so used in this procedure.

The following procedure describes a method for changing the contents of one memory location and, as an option, changing the contents of subsequent memory locations.

1. Press Write.
   Initially places the processor in a stop state, if the processor is not already in this state. Then the processor is placed in write mode and instructed that the contents of the memory location addressed by the memory address register (A0) are to be changed. The WRITE indicator lights when the Write control key is pressed.
2. Press Select.
   Places the processor in select mode as a necessary preliminary to selecting the memory address register. This step is not necessary unless the CHANGE indicator is lit.
3. Press hexadecimal-pad keys A,0 to enter the 2-digit selection code for the memory address register. Digits A0 appear in the LOCATION field of the REGISTER display.
4. Press Change.
   Places the processor in change mode preparatory to keying in the address of the memory location to be changed. The CHANGE indicator lights when the Change control key is pressed.
5. Key in, via the hexadecimal-pad keys, the 4- or 5-digit hexadecimal value representing the address of the memory location to be changed.
   This address appears in the CONTENTS field of the REGISTER display.
6. Press Select.
   Returns the processor to select mode as a necessary preliminary to selecting the memory data register. The CHANGE indicator turns off when the Select control key is pressed.
7. Press hexadecimal-pad key B to enter the first digit of the 2-digit selection code for the memory data register. The second digit need not be entered again at this point since the hexadecimal character 0 in position H2 is still actively engaged from step 3.
   Digits B0 appear in the LOCATION field of the REGISTER display.
8. Press Change.
   Places the processor in change mode preparatory to keying in the data for the memory location that is to be changed. The CHANGE indicator lights when the Change control key is pressed.
9. Key in, via the hexadecimal-pad keys, the 4- or 5-digit hexadecimal value representing the new data that is to be entered into the memory location to be changed.
   The data entered appears in the CONTENTS field of the REGISTER display.
10. Press Execute.
    When this action is initiated, the new data contents are loaded into the selected memory location.
11. If successive memory locations are to be changed using the current address of the memory address register as a base, press Plus 1.
    The PLUS indicator lights when the Plus 1 control key is pressed.
12. Repeat steps 9 and 10 for each sequential memory location to be changed.
    Note that while in the plus 1 mode (PLUS indicator lit), each pressing of the Execute key causes the memory address register to be incremented by 1 before the new data contents are loaded into the new incremented memory location.

Display Registers

The contents of any one of the 21 (6/30)/26 (6/40) visible registers may be displayed on the control panel. A register may be displayed when the processor is in any state.

The following procedure describes a method for displaying the contents of one register. The same procedure applies regardless of the processor state.

1. Press Select.
   Places the processor in select mode as a necessary preliminary to selecting the register to be displayed.
2. Key in, via the hexadecimal-pad keys, the 2-digit selection code (per Figure 5-2) for the desired register to be displayed.
   The selection code appears in the LOCATION field of the register display. When the desired selection code is keyed in, the data contents of the selected register is displayed in the CONTENTS field of the REGISTER display.

   NOTE: After a register in the run state is selected, its updated contents are displayed continuously (updated every 8 ms).

Change Registers

The contents of 18 of the 21 (6/30)/25 of the 26(6/40) visible registers may be changed from

the control panel (the M, I, S, and T registers cannot be modified from the control panel). A register may be changed only when the processor is in a stop state. The following procedure describes a method for changing the contents of one register.

1. Press Stop (or Read or Write) if the processor is in the run state.
   The STOP/STEP indicator lights when the Stop control key is pressed.
2. Press Select.
   Places the processor in the select mode as a necessary preliminary to selecting the register to be changed.
3. Key in, via the hexadecimal-pad keys, the 2-digit selection code (per Figure 5-2) for the desired register to be changed.
   The selection code appears in the LOCATION field of the REGISTER display. When the desired selection code is keyed in, the current data contents of the selected register are displayed in the CONTENTS field of the REGISTER display.
4. Press Change.
   Places the processor in change mode preparatory to keying in the data to the register that is to be changed. The CHANGE indicator lights when the Change control key is pressed.
5. Key in, via the hexadecimal-pad keys, the hexadecimal value representing the new data that is to be entered into the selected register.
   The data entered appears in the CONTENTS field of the REGISTER display and is shifted into the selected register.

Stop Program Execution

While a program is running, program execution can be stopped at any time by pressing Stop. The STOP/STEP indicator lights and the RUN, READY, and TRAFFIC indicators turn off. When this action is initiated, the processor completes the execution of the current instruction and enters the stop state. After this state is achieved, the following pertinent conditions exist.

1. The processor is automatically placed in a step mode; i.e., ready to execute one instruction at a time.
2. The instruction register (D0) contains the instruction to be executed next.
3. The P register (E0) contains an address incremented by 1 from the location of the instruction to be executed next.

Note that when a program is running and a Halt instruction is encountered (RUN indicator remains lit, but TRAFFIC indicator turns off), the processor does *not* enter the stop state but rather enters idle condition. In this condition, the Halt instruction is continuously re-executing and the processor is subject to external interrupts, etc. You must press the Stop control key in order to set the processor into a stop state.

Execute Single Instruction(s)

When running a program, you may wish to stop processing and step through the execution of one or more instructions. This procedure is accomplished from the control panel, as follows:

1. Press Stop.
   Refer to the previous procedure, "Stop Program Execution," for relevant information concerning processor/panel status after a stop state is attended.
2. Determine whether the processor has stopped at a point (address) from which you wish to begin executing single instructions. Display and view the contents of the P register (E0) using the procedure previously described for displaying registers.
   The P register (E0) contains an address incremented by 1 from the address of the instruction to be executed next. If this address is *one more* than the point at which you wish to start executing single instructions, proceed to step 8. However, if a new starting point is desired, continue to the next step.
3. Press CLeaR.
   Sets the instruction register (D0), the P register (E0), privilege mode, and indicators, etc., to zero as a necessary preliminary to specifying a new starting address.

   NOTE: In an online environment, do not press CLeaR. Instead, select the instruction register (D0) and change its contents to 0000_{16} so as not to affect peripherals.

4. Press Select.
   Places the processor in select mode, as is required before the P register (E0) is selected.
5. Press hexadecimal-pad keys E,0 to enter the 2-digit selection code for the P register. Digits E0 appear in the LOCATION field of the REGISTER display.
6. Press Change.
   Places the processor in change mode preparatory to keying in the address of the instruction to be executed next. The CHANGE indicator lights when the Change control key is pressed.
7. Key in, via the hexadecimal-pad keys, the 4- or 5-digit hexadecimal value representing the address of the next instruction to be

executed.
This address is entered in the P register and appears in the CONTENTS field of the REGISTER display.

8. Press Execute.
An attempt is made first to execute the instruction contained in the instruction register (D0). If the instruction register contains an executable instruction, then this one instruction is executed. After it is executed, the next succeeding instruction is placed in the instruction register, the P register (E0) is incremented by 1, and the processor is returned to the step mode. If the instruction register (D0) contains the value zero, the one instruction addressed by the P register (E0) is fetched and executed. The contents of the P register are incremented by 1 and the instruction addressed by the P register is placed into the instruction register. Finally, the contents of the P register are incremented by 1 again (i.e., the address of the instruction residing in the instruction register, plus 1) and the processor is returned to the step mode.
9. Repeat step 8 for each successive instruction to be executed singly. At any time after executing a single instruction, you may return to the run state (see the next procedure on restarting programs).

Restart Program

A program may be restarted from the control panel at any time and at any point after it has been stopped during execution. However, when you are restarting a program, it is your responsibility to ensure that:

- o No I/O was pending at the time the machine was halted.
- o All registers are stored to the context the program requires.

The restart procedure is as follows:

1. Determine whether the current start address is the point at which you wish to restart the program. Display the contents of the P register (E0), using the procedure previously described for displaying registers. The P register contains an address incremented by 1 from the address of the instruction to be executed next. If this address is *one more* than the point at which you wish to restart the program, then proceed to step 7. However, if a new starting point is desired, continue to the next step.
2. Press CLeaR.
Sets the instruction register (D0) and the P register (E0) to the value zero as a necessary preliminary to specifying a new starting address.
NOTE: Privilege mode, indicators, etc., are all changed by CLeaR and must be re-instated by software.
3. Press Select.
Places the processor in select mode as is required before the P register (E0) is selected.
4. Press hexadecimal-pad keys E,0 to enter the 2-digit selection code for the P register. Digits E0 appear in the LOCATION field of the REGISTER display.
5. Press Change.
Places the processor in change mode preparatory to keying in the address of the instruction from which you wish to restart execution of the program. The CHANGE indicator lights when the Change control key is pressed.
6. Key in, via the hexadecimal-pad keys, the 4- or 5-digit hexadecimal value representing the restart address.
This address is entered into the P register and appears in the CONTENTS field of the REGISTER display.
7. Press Ready.
Places the processor in ready mode; i.e., instructs the processor that program execution is to begin when the Execute key is pressed. The READY indicator lights and the STOP indicator turns off when the Ready control key is pressed.
8. Press Execute.
The processor is placed in a run state and attempts to start executing the program beginning with the instruction contained in the instruction register (D0).
If the instruction register contains an executable instruction, program execution begins with the instruction specified by the instruction register. If the instruction register contains the value zero, program execution begins with the instruction addressed by the P register (E0).
The RUN and TRAFFIC indicators light when the Execute key is pressed. Program execution continues until a software Halt is encountered or the Stop, Read or Write key is pressed.

Master Clear

The master clear procedure is used to set or restore the processor to a standard initialized

state as a prelude to certain functions such as a bootstrap procedure. The following procedure indicates the master clearing operation:

1. Press Stop if the processor is not already in the stop state. Master clear is only operative when the processor is in the stop state. The STOP/STEP indicator lights when the Stop control key is pressed.
2. Press CLeaR.
   The specific functions that are activated by pressing CLeaR are listed in Table 5-1, under the description of the CLeaR control key.

BOARD CHECKING

Boards inside the central subsystem (boards that have microprogramming (firmware) capabilities) have built-in self-checking capabilities and LEDs (Light-Emitting Diodes) for problem indication. When the computer is powered up, the LEDs on the associated boards are illuminated, and after internal checking is finished, the lights are extinguished one after another. Any LED remaining lit indicates a malfunction on that board and the CHECK light on the operator's panel is lit to inform the operator. The operator thereby identifies the board which is malfunctioning. This allows the integrity of the configuration to be quickly checked, ensuring that all boards are inserted properly (filling every consecutive slot) and all terminators have been inserted.

Accessing and Checking Boards

The operator is required to check and identify malfunctioning boards inside the central subsystem. The accessing and checking procedures for both the freestanding (tabletop) and rack-mounted versions of the central subsystem are described below:

Freestanding (Tabletop) Central Subsystem

1. With the power ON, push the right and left cover release levers. (The release levers are located on the left and right side of the base chassis, 6 inches from the front.)
2. Lift the white cover upwards as far as possible. (The cover opens far enough to allow easy access to the boards for viewing and if necessary, removal.)
3. Check the boards in the cabinet for the one with the illuminated LED. (The LED is located in the right front section of the board and is easily viewed by the operator.)
4. Gently apply pressure to the front of the lighted board to ensure that it is positioned and seated properly.
5. Take the applicable action to correct any malfunctioning board (e.g., replace, notify Honeywell FED, etc.).
6. Once the light is extinguished, push the white cabinet cover down to its original position, ensuring that it is locked in place.

Rack-Mounted Central Subsystem

1. With the power ON, grasp the white face plate covering the operator's panel and remove. (The face plate is kept in place by a series of pressure fittings which are easily disengaged.)
2. Once the front of the operator's panel is exposed (black rectangle with the buttons and indicators), the next operation is to unlatch and swing away the operator's panel for access to the boards.
3. To remove the operator's panel:
   a. Unlock the operator's panel. (Turn the set screw, on the extreme lower right section of the operator's panel chassis, counter-clockwise.
   b. Swing the hinged operator's panel to the left. (Once it is opened the operator has easy access to the boards for viewing and if necessary removal.
4. Check the boards in the cabinet for the one with the illuminated LED. (The LED is located in the right front section of the board and is easily viewed by the operator.)
5. Gently apply pressure to the front of the lighted board to ensure that it is positioned and seated properly.
6. Take the applicable action to correct any malfunctioning boards (.e.g., replace, notify Honeywell FED, etc.).
7. Once the light is extinguished, swing the operator's panel closed (to the right) and turn the locking set screw clockwise.
8. Reposition the white face plate cover over the operator's panel and apply pressure to ensure proper positioning.

PHYSICAL CHARACTERISTICS

A variety of cabinet options is available for the central subsystem. All offer easy access and require only front-to-rear airflow for cooling:

- Drawer unit – 5.25 inches (13.33 cm) high for standard EIA rack mounting.
- Cabinet unit – 60 inch (152.4 cm) high for central subsystem, diskette, and other drawers.

- Tabletop configuration – completely enclosed, completely portable, 5.5 inches (13.97 cm) high, 19.5 inches (49.53 cm) wide, 29.7 inches (75.43 cm) deep.
- Office packaging, custom-shaped desk – 29.5 inches (74.9 cm) high; 49 inches (124 cm) wide; 33 inches (83.3 cm) deep; integrated keyboard (optional); concealed control panel; 29.5 inches (74.9 cm) high standard EIA rack space.

SECTION 6

PERIPHERAL DEVICES

This section describes the functional characteristics, features, operational capabilities, and programming for the various peripheral devices listed in Table 6-1. All of the devices are supported by the Level 6 GCOS/BES and GCOS 6/MDT I/O drivers and executive routines. Test and maintenance programs are also provided to assist in fault isolation and maintenance of the equipment.

PERIPHERAL DEVICE CONNECTION

The majority of peripheral devices listed in Table 6-1 are connected to the Level 6 Megabus via a single-board Multiple Device Controller (MDC9101) and appropriate Device-Pacs (i.e., adapters). The MDC is firmware driven and microprogrammed to provide four levels of simultaneity supporting up to four devices in any combination[1] with full capability of Direct Memory Access (DMA). Multiple controllers can be attached to the Megabus (see Figure 6-1).

Cartridge Disk Units connect to the Megabus via the Mass Storage Controller (MSC9101), which supports up to four disk units and requires only one Disk Device-Pac (CDM9101). The 7- and 9-track Magnetic Tape Units connect to the Megabus via the Magnetic Tape Controller (MTC9101), which supports up to four tape units *or* up to two tape units and up to two (same or combination) of the following: serial printers, line printers, and card readers. The MTC requires only one Magnetic Tape Device-Pac (MTM9101 for 7-track NRZI or MTM9102 for 9-track NRZI) and other peripheral Device-Pacs as appropriate.

[1] In a system the diskette device is specifically allocated ports 0 and 1 or ports 2 and 3 or else ports 0, 1, 2, and 3. All other devices can be allocated ports in any combination.

TABLE 6-1. LEVEL 6 PERIPHERAL DEVICES

Device	Description
Card Readers	
CRU9101/9103	300/500 cpm punched card readers
CRU9102/9104	300/500 cpm punched card and mark sense readers
CRU9108/9111	300/500 cpm punched card readers
CRU9109/9112	300/500 cpm punched card and mark sense (IBM mode) readers
CRU9110/9113	300/500 cpm punched card and mark sense (Honeywell mode) readers
Consoles	
TTU9101/9102	10 cps, 72 charcters per line, ASR-/KSR-33
TTU9103/9104	10 cps, 72 characters per line, ASR-/KSR-33 with autoshutdown
TWU9101	30 cps, 132 characters per line, 64-character ASCII set
TWU9104/9106	30 cps/120 cps, 132 characters per line, 96-character ASCII set
DKU9101/9102	960 character display (12 lines, 80 characters per line)
Disk Devices	
DIU9101/9102	Single/dual diskettes, 401,016 bytes/disk (unformatted), 256,256 bytes/disk (formatted)
CDU9101	Cartridge Disk Unit, low-density, 2.5/2.8M bytes, removable disk only
CDU9102	Cartridge Disk Unit, low-density, 5.0/5.6M bytes, fixed and removable disks
CDU9103	Cartridge Disk Unit, high-density, 5.0/5.6M bytes, removable disk only
CDU9104	Cartridge Disk Unit, high-density 10.0/11.2M bytes, fixed and removable disk
CDU9114	Cartridge Disk Unit, low-density, 5.0/5.6M bytes, fixed and removable disks
CDU9116	Cartridge Disk Unit, high-density, 10.0/11.2M bytes, fixed and removable disks
Printers	
PRU9101/9102	Serial Printers, 60 lpm, 64-/96-ASCII character set
PRU9103/9104	Line Printers, 240/300 lpm, 96-/64-ASCII character set
PRU9105/9106	Line Printers, 440/600 lpm, 96-/64-ASCII character set
Magnetic Tape Units	
MTU9104/9105	9-track Magnetic Tape Unit, 45/75 ips
MTU9112/9113	7-track Magnetic Tape Unit, 45/75 ips

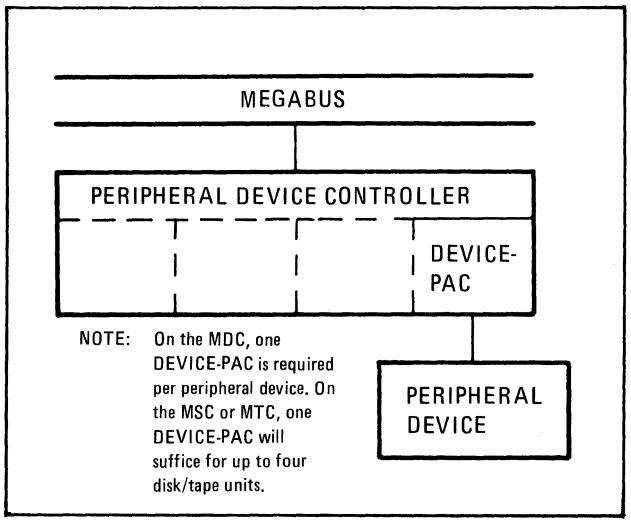
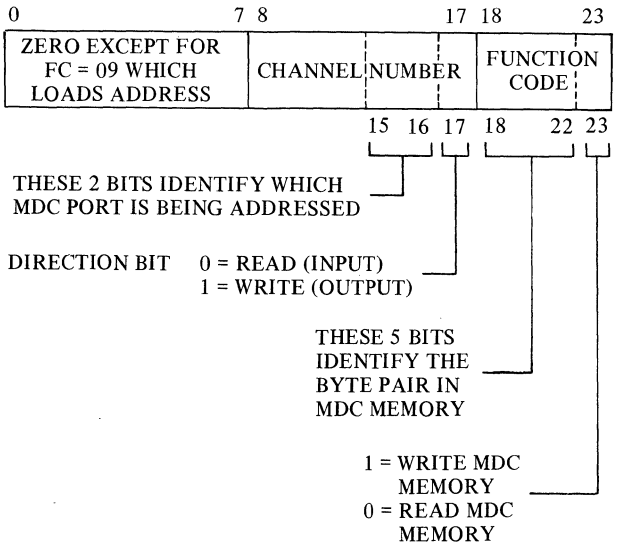
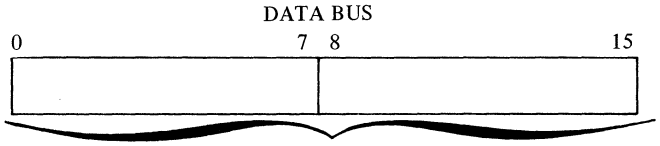


Figure 6-1. Peripheral Device Connection

MDC/MSC/MTC Memory and Command Interpretation

Internal to the MDC/MSC/MTC is a 256-byte Read/Write (R/W) memory (64 bytes of this memory are dedicated to each channel of the controller). The memory is used for all active storage in the controller relative to a specific channel. Included are the DMA address, range, interrupt level, status, and other internal MDC/MSC/MTC working storage. The CP has the ability to read or write any location in the R/W memory as long as the specific channel is not busy. In order to write a memory location, an I/O output command is used, whereas reading is done with an I/O input command. Addressing of the R/W memory relates to the I/O command as follows:





THESE 16 BITS WILL CONTAIN THE BYTE PAIR IN MDC MEMORY

The format shown is for a Write cycle on the bus. For a Read cycle, the memory data will be returned from the controller on a second bus transfer. It can be seen that all of the defined function codes for each peripheral device represents specific controller memory locations.

The general response of the controller/peripheral device to an I/O command is, therefore, only an updating of the designated memory location. Only two exceptions to this exist:

1. Function code 0D which is the range register and comes as part of an IOLD command is an implicit start I/O command. When a controller peripheral device other than a diskette receives this function code, it assumes all the contents of its R/W memory are correct and executes the required operation. This function code also causes the peripheral device to become busy. For diskette devices and devices on other controllers, the Output Task function code is used to initiate peripheral action. This function code (0D) will violate memory protection if used separately from the IOLD command.
2. Function code 01, which is Output Control, causes additional actions as described for each peripheral device.

The controller treats Function code 09 specially, in that a 24-bit address is to be loaded. In this case, the 8-bit module number is written into the controller R/W memory immediately following the location implied by 09 (i.e., 0B).

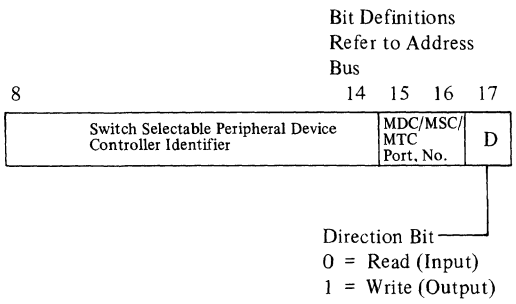
For the IOLD command (FC=09), the direction bit is a 1 for output or a 0 for input. For all other commands the direction bit is ignored.

Bus Responses and Busy Conditions

The IOLD command causes a peripheral device to go busy and it remains busy until after the interrupt has been acknowledged by the CP. If the interrupt level was set to zero thus blocking the interrupts, the peripheral device will go non-busy immediately upon setting the status condition which would have caused an interrupt. During the time the peripheral device is busy, every I/O command is NAKd except for Output Control (Function Code 01).

Channel Number

The Channel Number for a controller is separated into fields as follows:



Bits 8–14 can have any value. The value selected will identify the entire MDC/MSC/MTC and, therefore, will affect other channel number selections for devices also connected to the MDC/MSC/MTC. Bits 15 and 16 identify which Device-Pac slot is occupied. Any or all of the four slots may be occupied. When different devices coexist on the MDC or MTC, the Device-Pac's physical position should be considered because the MDC/MTC services port #0 (bit 15, 16 = 00) at highest priority, and port #3 (bit 15, 16 = 11) at lowest priority. Bit 17 must be 0 for input and 1 for output.

Device Identification Number

In response to Function Code 26, the MDC/MSC/MTC will supply a 16-bit device identification number on the data bus. Table 6-2 lists the device ID numbers for the various peripheral devices.

TABLE 6-2. PERIPHERAL DEVICE ID NUMBERS

Peripheral Device	Device ID Number
Card Equipment	
CRU9101/9102/9103/9104	2008
CRU9108/9109/9110/9111/ 9112/9113	2008
Console Equipment	
TTU9101/9103	2018
TTU9102/9104	2019
DKU9101/9102	201A
TWU9101	2018
TWU9104/9106	201C
Disk Equipment	
DIU9101/9102	2010
CDU9101	2330
CDU9102	2331
CDU9103	2332
CDU9104	2333
CDU9114	2331
CDU9116	2333
Printer Equipment	
PRU9101	2004
PRU9102	2006
PRU9104/9106	2000
PRU9104/9106 with PRF9102	2001
PRU9103/9105	2002
PRU9103/9105 with PRF9102	2003
Magnetic Tape Equipment	
MTU9112	2169 (@ 556 cpi)/ 2171 (@ 800 cpi)
MTU9113	216A (@ 556 cpi)/ 2172 (@ 800 cpi)
MTU9104	2145
MTU9105	2146

Test Mode

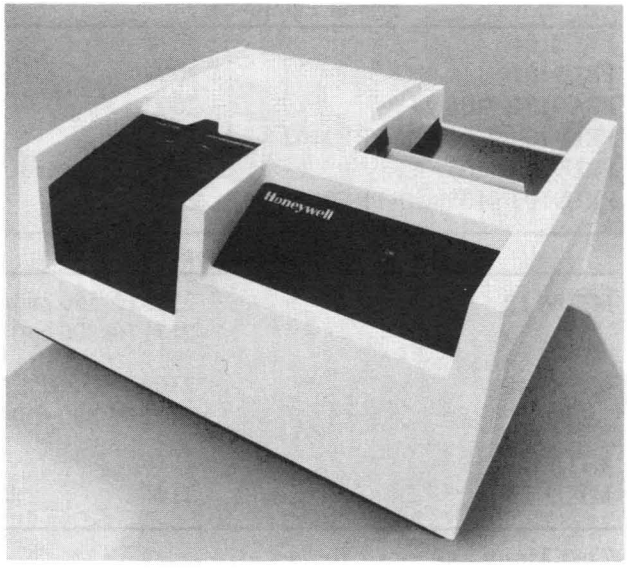
When an Output Control Word command is received with the Test Mode bit on (while the controller is in normal operating mode), the controller will enter Test Mode and stop. Once the controller is in Test Mode, subsequent commands will cause the contents of the data bus to be loaded into the controller instruction register. The data bus contains the complement of a microinstruction. One clock pulse will then be issued to execute the microinstruction loaded. This enables a sequence of microinstructions to be executed in single step mode from a software test routine. Normal mode is resumed when a "reset test mode" microinstruction is executed or when Master Clear is activated on the bus.

All function codes which are received over the bus while the controller is in Test Mode will cause the contents of the data bus to be loaded into the controller instruction register as described above (function code field of the address bus is ignored). Note that response cycles will not be generated for input commands (e.g., Input Status Word A).

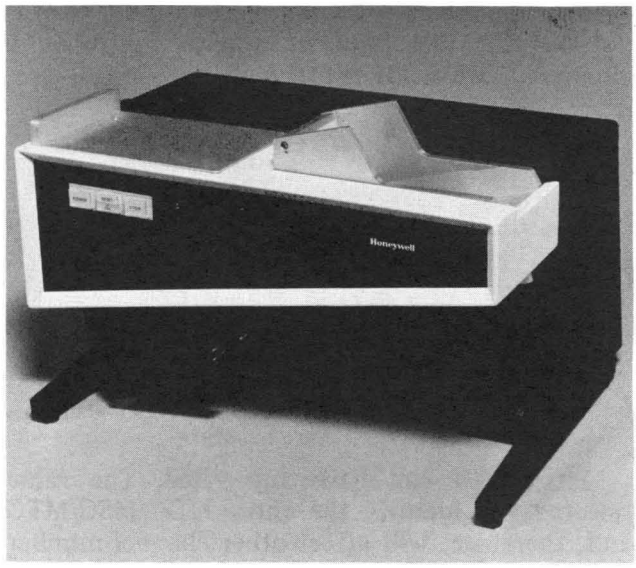
Test Mode operates on an entire MDC/MSC/MTC and, therefore, precludes normal usage of other devices on the controller at the same time.

CARD READERS

Level 6 card readers are compact, self-contained, tabletop units providing economy and versatility of operation (Figure 6-2). Hollerith or binary punched cards can be read at the rate of 300 or 500 cards per minute (cpm). In addition, some of the readers can read 40- or 80-column mark sense cards. A 51-column punched card reading capability is available as an option on some units. The specifications of the various readers offered are summarized in Table 6-3.



CRU9108/9109/9110/9111/9112/9113



CRU9101/9102/9103/9104

Figure 6-2. Level 6 Card Readers

TABLE 6-3. CARD READER SPECIFICATIONS

Characteristics \ Device	Card Readers (CRU) 9101	9102	9103	9104	9108	9109	9110	9111	9112	9113
Speed (cpm)	300	300	500	500	300	300	300	500	500	500
Media:										
80-col. punched cards	X	X	X	X	X	X	X	X	X	X
80-col. mark sense cards	–	X[a]	–	X[a]	–	X[a]	X[b]	–	X[a]	X[b]
40-col. mark sense cards	–	X[a]	–	X[a]	–	–	–	–	–	–
51-col. punched cards	X[c]	X[c]	X[c]	X[c]	–	–	–	–	–	–
Input Hopper Capacity	←		500	→	←			1000	→	
Output Stacker Capacity	←		500	→	←			1000	→	
Device Interface	Each card reader requires its own Device-Pac (CRM9101)									
Data Transfer Mode	Automatic translation via Device-Pac of Hollerith to binary or ASCII									
Reading Technique	Photoelectric, column-by-column, serially									
Card Specification	Standard punched or mark sense cards, 7-3/8 in. x 3-1/2 in. (18.7 cm x 8.9 cm), 0.0077 in. (0.0196 cm) thick; clean and free from excessive curl									
Physical Dimensions										
Height	13.50 in. (34.3 cm)							11.75 in. (29.9 cm)		
Width	19.50 in. (49.5 cm)							23.25 in. (59.1 cm)		
Depth	15 in. (38.1 cm)							20 in. (50.8 cm)		
Weight	35 lb (15.8 kg)							73 lb (32.9 kg)		

[a]IBM mode: marks are arranged on rows in even- or odd-numbered columns. The preprinting of marks must be on the front face of the card.

[b]Honeywell Mode: marks are arranged on 12 rows in even-numbered columns. The preprinting of marks must be on the rear face of the card.

[c]Option CRF9101.

The card readers interface with the central processor by means of a single-board Multiple Device Controller (or Magnetic Tape Controller) via a Card Reader Device-Pac (CRM9101) and a 50-foot cable. A maximum of four card readers are connectable per MDC (or two per MTC).

Features

- o Convenient tabletop size
- o Low cost
- o Choice of 300- or 500-cpm reader
- o Choice of versatile 80-column punched card and 40-/80-column mark sense readers or 80-column punched card readers
- o Status and error indication bits that can be read into the processor under program control for error detection and operator notification
- o Two program-selected punched card reading modes: ASCII with automatic Hollerith to 8-bit ASCII conversion; and Direct (Binary), with 12-bit binary format
- o Test mode for easy maintenance

Operation

All data transfers are under DMA (Direct Memory Access) control, with an end-of-range interrupt after the program selected number of columns have been transferred.

While waiting for a ready indication, the central processor is free to perform other operations. Program interrupt is used to signal the central processor when further device servicing is required.

Data Format

Cards may be read in any of four formats:

- o Binary (Direct Transcription)
- o ASCII
- o Honeywell Mark
- o IBM Mark

Binary Mode

In this mode, the 12 bits designated by each card column are placed on the data bus right-justified, as shown in Figure 6-3. One memory word is required for each card column read. All 2^{12} hole patterns are valid in Binary Mode. The four remaining bits will be zero-filled.

ASCII Mode

In this mode, the 12 bits designated by each card column are converted to a single 8-bit byte and transferred to the bus, two bytes per word (see Figure 6-4). The conversion is a Hollerith to ASCII conversion and is performed by the Device-Pac. The code table of Table 6-4 completely describes the correspondence between the two codes. The bit designations in Table 6-4 are the ASCII standards and relate to the bits on the data bus as shown in Table 6-5. Punches or marks which do not exist in Table 6-4 cause an ASCII error status report as described for bit 8 in Table 6-7.

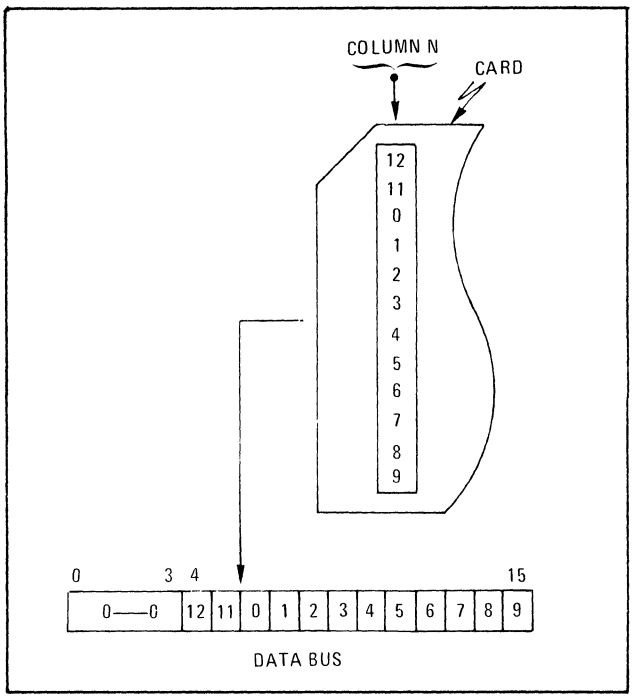


Figure 6-3. Binary Mode Format

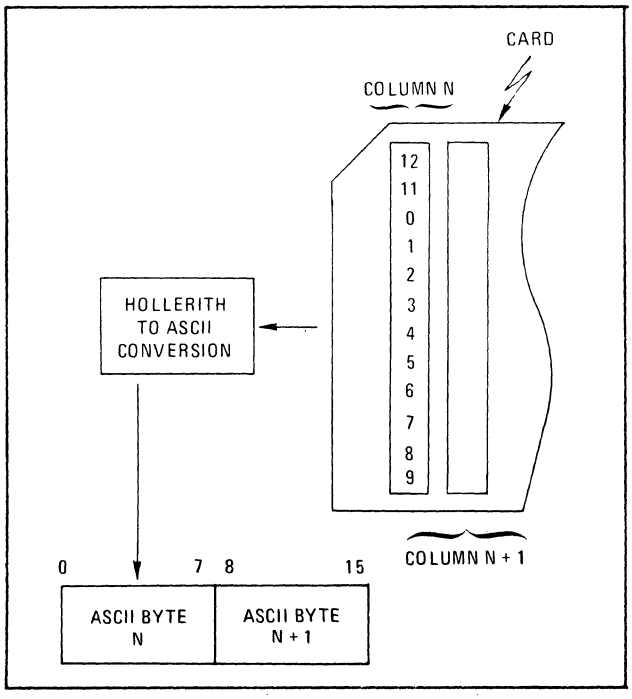


Figure 6-4. ASCII Mode Format

TABLE 6-4. HOLLERITH-ASCII CODE TABLE

b4b3b2b1	Col / Row	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Col / Row
	b8 b7 b6 b5	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1	
0000	0	NUL 12-0-9-8-1	DLE 12-11-9-8-1	SP No Pch	0 0	@ 8-4	P 11-7	\ 8-1	p 12-11-7	11-0-9 8-1	12-11-0-9-8-1	12-0-9-1	12-11-9-8	12-11-0-9-6	12-11-8-7	12-11-0-8	12-11-9-8-4	0
0001	1	SOH 12-9-1	DC1 11-9-1	! ① 12-8-7	1 1	A 12-1	Q 11-8	a 12-0-1	q 12-11-8	0-9-1	9-1	12-0-9-2	11-8-1	12-11-0-9-7	11-0-8-1	12-11-0-9	12-11-9-8-5	1
0010	2	STX 12-9-2	DC2 11-9-2	" 8-7	2 2	B 12-2	R 11-9	b 12-0-2	r 12-11-9	0-9-2	11-9-8-2	12-0-9-3	11-0-9-2	12-11-0-9-8	11-0-8-2	12-11-0-8-2	12-11-9-8-6	2
0011	3	ETX 12-9-3	DC3 11-9-3	# 8-3	3 3	C 12-3	S 0-2	c 12-0-3	s 11-0-2	0-9-3	9-3	12-0-9-4	11-0-9-3	12-0-8-1	11-0-8-3	12-11-0-8-3	12-11-9-8-7	3
0100	4	EOT 9-7	DC4 9-8-4	$ 11-8-3	4 4	D 12-4	T 0-3	d 12-0-4	t 11-0-3	0-9-4	9-4	12-0-9-5	11-0-9-4	12-0-8-2	11-0-8-4	12-11-0-8-4	11-0-9-8-2	4
0101	5	ENQ 0-9-8-5	NAK 9-8-5	% 0-8-4	5 5	E 12-5	U 0-4	e 12-0-5	u 11-0-4	11-9-5	9-5	12-0-9-6	11-0-9-5	12-0-8-3	11-0-8-5	12-11-0-8-5	11-0-9-8-3	5
0110	6	ACK 0-9-8-6	SYN 9-2	& 12	6 6	F 12-6	V 0-5	f 12-0-6	v 11-0-5	12-9-6	9-6	12-0-9-7	11-0-9-6	12-0-8-4	11-0-8-6	12-11-0-8-6	11-0-9-8-4	6
0111	7	BEL 0-9-8-7	ETB 0-9-6	' 8-5	7 7	G 12-7	W 0-6	g 12-0-7	w 11-0-6	11-9-7	12-9-8	12-0-9-8	11-0-9-7	12-0-8-5	11-0-8-7	12-11-0-8-7	11-0-9-8-5	7
1000	8	BS 11-9-6	CAN 11-9-8	(12-8-5	8 8	H 12-8	X 0-7	h 12-0-8	x 11-0-7	0-9-8	9-8	12-8-1	11-0-9-8	12-0-8-6	12-11-0-8-1	12-0-9-8-2	11-0-9-8-6	8
1001	9	HT 12-9-5	EM 11-9-8-1	) 11-8-5	9 9	I 12-9	Y 0-8	i 12-0-9	y 11-0-8	0-9-8-1	9-8-1	12-11-9-1	0-8-1	12-0-8-7	12-11-0-1	12-0-9-8-3	11-0-9-8-7	9
1010	10	IF 0-9-5	SUB 9-8-7	* 11-8-4	: 8-2	J 11-1	Z 0-9	j 12-11-1	z 11-0-9	0-9-8-2	9-8-2	12-11-9-2	12-11-0	12-11-8-1	12-11-0-2	12-0-9-8-4	12-11-0-9-8-2	10
1011	11	VT 12-9-8-3	ESC 0-9-7	+ 12-8-6	; 11-8-6	K 11-2	[12-8-2	k 12-11-2	{ 12-0	0-9-8-3	9-8-3	12-11-9-3	12-11-0-9-1	12-11-8-2	12-11-0-3	12-0-9-8-5	12-11-0-9-8-3	11
1100	12	FF 12-9-8-4	FS 11-9-8-4	, 0-8-3	< 12-8-4	L 11-3	\ 0-8-2	l 12-11-3	| 12-11	0-9-8-4	12-9-4	12-11-9-4	12-11-0-9-2	12-11-8-3	12-11-0-4	12-0-9-8-6	12-11-0-9-8-4	12
1101	13	CR 12-9-8-5	GS 11-9-8-5	- 11	= 8-6	M 11-4	] 11-8-2	m 12-11-4	} 11-0	12-9-8-1	11-9-4	12-11-9-5	12-11-0-9-3	12-11-8-4	12-11-0-5	12-0-9-8-7	12-11-0-9-8-5	13
1110	14	SO 12-9-8-6	RS 11-9-8-6	. 12-8-3	> 0-8-6	N 11-5	^ ② 11-8-7	n 12-11-5	~ 11-0-1	12-9-8-2	9-8-6	12-11-9-6	12-11-0-9-4	12-11-8-5	12-11-0-6	12-11-9-8-2	12-11-0-9-8-6	14
1111	15	SI 12-9-8-7	US 11-9-8-7	/ 0-1	? 0-8-7	O 11-6	_ 0-8-5	o 12-11-6	DEL 12-9-7	11-9-8-3	11-0-9-1	12-11-9-7	12-11-0-9-5	12-11-8-6	12-11-0-7	12-11-9-8-3	EO 12-11-0-9-8-7	15

① may be "|"
② may be "¬"
③ The top line in each entry to the table represents an assigned character (columns 0 to 7). The bottom line in each entry is the corresponding card hole-pattern.
④ All bit designations are in ASCII.

TABLE 6-5. ASCII BIT RELATION TO BITS ON DATA BUS

ASCII	Left Byte	Right Byte
8	0	8
7	1	9
6	2	10
5	3	11
4	4	12
3	5	13
2	6	14
1	7	15

Honeywell Mark Reading Mode

In this mode, marks are arranged on 12 rows in even-numbered columns. The preprinting of marks must be on the rear face of the card. This mode is not program-selectable.

IBM Mark Reading Mode

In this mode, marks are arranged on rows in even- or odd-numbered columns. The preprinting of marks must be on the front face of the card. This mode is not program-selectable.

Instructions

Table 6-6 lists the I/O commands to which the controller/Device-Pac/card readers respond. A detailed description of each command follows this table.

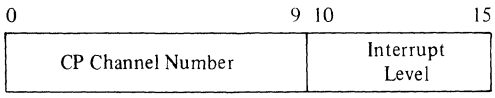
TABLE 6-6. CARD READER COMMANDS

Type	Function Code	Command
Output	03	Output Interrupt Control
	01	Output Control
	09	Output Address and Range
	11	Output Configuration
Input	02	Input Interrupt Control
	08	Input Memory Byte Address
	0A	Input Memory Module Address
	0C	Input Range
	10	Input Configuration
	18	Input Status
	26	Input Device ID (2008)

Output Commands

Output Interrupt Control (03)

Format:

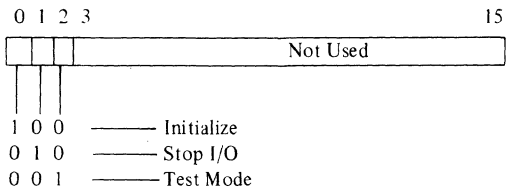


Function:

Loads a 16-bit word into the interrupt control register with the information necessary for generating an interrupt, i.e., the interrupt level and which CP the interrupt is to be issued to. On interrupting, the CP channel number is returned on the address bus while the interrupt level is returned on the data bus.

Output Control (01)

Format:



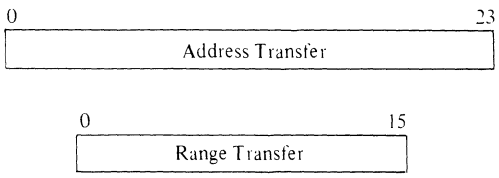
Function:

Loads a 16-bit control word into the referenced channel and is accepted unconditionally regardless of any channel busy status. The channel may respond with a WAIT, but never with a NAK. The individual bits cause the following specific actions to occur:

- o Initialize
  - – Causes the controller to run its resident logic test
  - – Clears all controller Device-Pacs
  - – Clears the Bus Interface
  - – Blocks Interrupts
  - – Resets the Busy condition
  - – Operates on all channels of a controller
  - – Sets card reader attachment to ASCII mode
- o Stop I/O
  - – Resets Busy condition
  - – Causes Interrupt if enabled
  - – Abruptly stops transfer from card reader
  - – Updates status in controller R/W memory
  - – Does not affect other channels
- o Test Mode
  - – Refer to "Test Mode" earlier in this section

Output Address and Range (09)

Format:



Function:

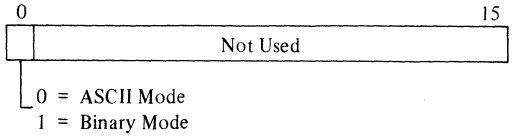
The IOLD command when executed by the processor creates two bus transfers to the controller. The first is the 24-bit address transfer and the second is the 16-bit range transfer. To the controller, these are two separate and distinct bus transfers with two separate and distinct function codes of 09 for the address transfer and 0D for the range transfer. The programmer need only specify the first function code and the processor hardware/firmware automatically calculates the second function code (by adding 04 to it).

- o Address Transfer – A 24-bit quantity transferred to the controller to be used as the starting byte address of the data record in memory where the card data will be stored.
- o Range Transfer – A 16-bit quantity transferred to the controller to be used as the byte range count of the data record in memory where the card data will be stored. Range is specified as twos complement for the controller and must be positive. There-

fore, range is $1 \leqslant r \leqslant 2^{15} - 1$. For the card reader, range should not be greater than the number sufficient to input a single card. If range is set less than that number, the controller will input only the specified number of characters. If the range is set larger than that number, the controller will terminate transfer at end of card rather than at the end of the range.
The range transfer includes the implicit command "Start I/O" and causes the controller to start reading a card and go Busy.

Output Configuration (11)

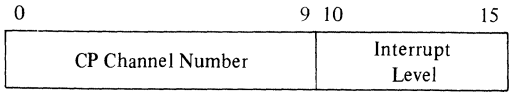
Format:



Function:
Outputs a word to determine in which of two modes the card reader operates.

Input Commands
Input Interrupt Control (02)

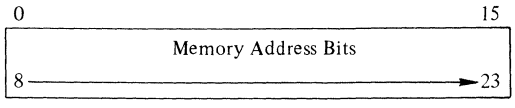
Format:



Function:
Causes the channel to place the contents of its interrupt control register on the data bus.

Input Memory Byte Address (08)

Format:

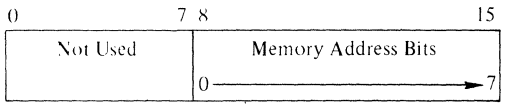


Function:
Causes the channel to place the 16 least significant bits of its DMA address register on the data bus. The bits represent the memory address register bits 8–23.

NOTE: This command will violate memory protection.

Input Memory Module Address (0A)

Format:

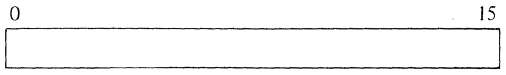


Function:
Causes the channel to place the eight most significant bits of its DMA address register on the data bus. The bits represent memory address bits from 0–7 inclusive.

NOTE: This command will violate memory protection.

Input Range (0C)

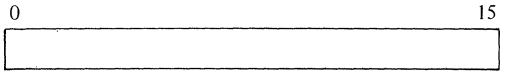
Format:



Function:
Causes the channel to place the contents of its range register on the data bus.

Input Configuration (10)

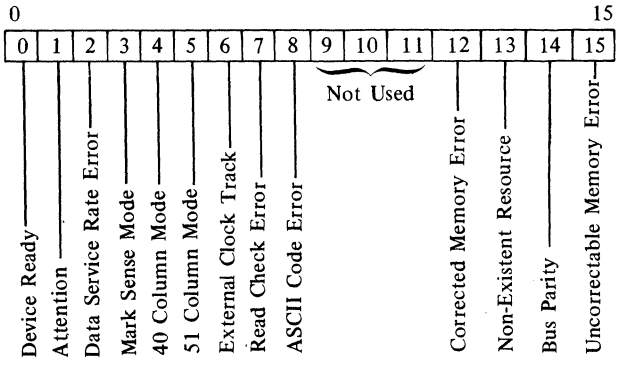
Format:



Function:
Causes the channel to place the contents of its configuration register on the data bus.

Input Status (18)

Format:



Function:
Causes the channel to place the contents of its status register on the bus. Also see Table 6-7.

Input Device ID (26)

TABLE 6-7. STATUS BIT DEFINITIONS – CARD READER

Status Condition	Bit	Definition	Reset By
Device Ready	0	Device is online; cards loaded; no further manual intervention is required to place it under program control. Note that a change of state of this bit will cause the Attention bit (bit 1) to be set resulting in an interrupt (if the interrupt level is nonzero).	A change in condition
Attention	1	Set whenever the Device Ready bit (bit 0 of the status word) changes state. Indicates to software any change of operational status of the device. When set, an interrupt is attempted (if the interrupt level is nonzero). If a previously initiated operation is in progress when a device state change is sensed, the resultant interrupt (with the Attention bit set) serves as notification of both the end of the operation and the device state change.	Input status word 1[a]
Data Service Rate Error	2	Set during a Read/Write operation when the data transfer to/from main memory cannot be maintained at a high enough rate. Either data was lost on input because of failure to keep up with device demands or data was unavailable on output when required by the device.	Next IOLD command[a]
Mark Sense Mode	3	The Mark Sense/STD switch on the Card Reader is in the Mark Sense mode.	A change in switch position
40-Column Mode	4	The 40/80 Column switch on the Card Reader is in the 40-col. position.	A change in switch position
51-Column Mode	5	The 51/80 column switch on the Card Reader is in the 51-col. position.	A change in switch position
External Clock Track	6	The Clock Track/Internal clock switch on the Card Reader is in the Clock Track position.	A change in switch position[b]
Read Check Error	7	The Card Reader failed to meet a light-dark check indicating possible device failure.	Next IOLD command[a]
ASCII Code Error	8	A pattern from card did not translate to ASCII (in ASCII mode only). Data in error forced to all Ones.	Next IOLD command[a]
Corrected Memory Error	12	During execution of previous operation, Main Memory detected and corrected a memory read error. Data delivered to the controller was assumed correct.	–
Nonexistent Resource	13	Set whenever the controller attempts a Write/Read request bus cycle and receives a NAK response. Indicates a possible programming error or illegal IOLD (direction bit 1).	Next IOLD command or input status word[a]

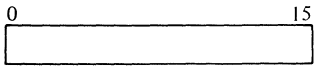
TABLE 6-7 (CONT). STATUS BIT DEFINITIONS–CARD READER

Status Condition	Bit	Definition	Reset By
Bus Parity	14	Set whenever the controller detects a parity error on either byte of the data bus during any Output bus cycle (i.e., odd function code), during a second half memory read cycle, or when a parity error is detected in bits 0–7 of the address bus during an Output Address command.	Input status word 1[a]
Uncorrectable Memory Error	15	During execution of previous operation, Main Memory detected a read error which it could not correct. Data delivered to the controller was incorrect. Will not cause termination of the operation in progress.	

[a]Initialize (Output Control Word) and Master Clear on the bus also reset these status bits.

[b]Applicable to CRU9102/9104 readers only.

Format:



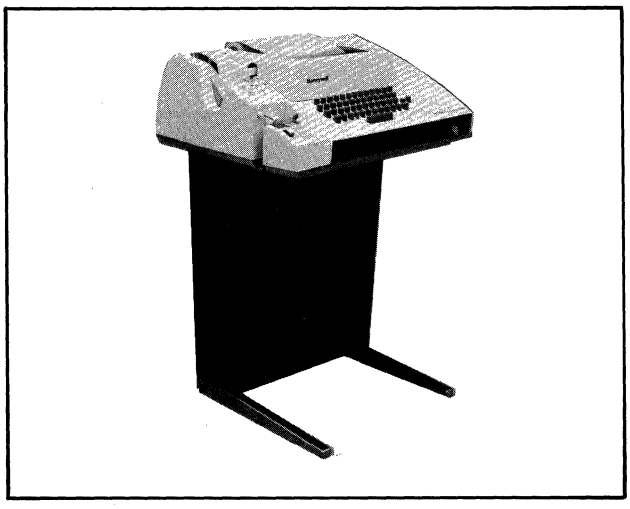
Function:

Causes the channel to place its device identification number (2008) on the bus.

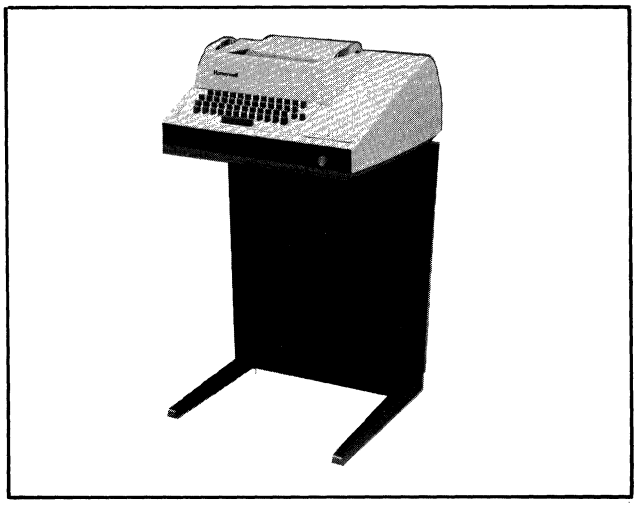
TELEPRINTER CONSOLES

Four teleprinter console devices, Types TTU9101/9103 (ASR-33) and TTU9102/9104 (KSR-33), are available for Level 6 system console and I/O use (Figure 6-5). The TTU9101/9102 differ, respectively, from the TTU9103/9104 in that the latter are equipped with an autoshutdown feature[2] which can extend console life and reduce maintenance costs. The teleprinter consoles print data from or transmit data to the central processor at the rate of 10 characters per second. The TTU9101/9103 can also read and punch paper tape at the same rate. Teleprinter Specifications are listed in Table 6-8.

The teleprinter console interfaces with the central processor by means of a single-board Multiple Device Controller via a Console Device-Pac (KCM9101), and a 26-foot cable. Up to four teleprinters can be connected to a single MDC.



TTU9101/9103



TTU9102/9104

Figure 6-5. Level 6 Teleprinter Consoles

[2]The autoshutdown feature is not currently supported by software.

TABLE 6-8. TELEPRINTER CONSOLE SPECIFICATIONS

Characteristics \ Device	Teleprinter Consoles (TTU) 9101	9102	9103	9104
Speed (cps)				
Print	10	10	10	10
Read/Punch	10	–	10	–
Autoshutdown Feature	–	–	X	X
Printer				
Feed	← Friction →			
Capacity	Paper roll 8-1/2 in. (21.6 cm) wide, 5 in. (12.7 cm) diameter			
Density	Horizontal – 10 characters per inch, 72 characters per line; Vertical – 3 or 6 lines per inch			
Reader/Punch				
Tape	8-level, paper or Mylar-paper combination, 1 in. (2.54 cm) wide			
Capacity	855-ft (260-m) tape roll			
Density	10 characters per inch			
Interfaces				
EIA	Asynchronous data transfer with one or two stop bits; 110 baud operation			
Device	Each teleprinter console requires its own Device-Pac (KCM9101)			
Field Upgradability	Option TTK9101 (Autoshutdown Add-On Kit) enables TTU9101/9102 to be upgraded to TTU9103/9104			
Physical Dimensions (including pedestal)	TTU9101/9103		TTU9102/9104	
Height	32-7/8 in. (83.5 cm)		32-7/8 in. (83.5 cm)	
Width	22 in. (55.9 cm)		18-5/8 in. (47.3 cm)	
Depth	18-1/2 in. (47 cm)		18-1/2 in. (47 cm)	
Weight	56 lb (25.2 kg)		52 lb (23.4 kg)	

Features

- o Printing and paper tape reading and punching in both online and offline modes (TTU9101/9103)
- o Online operation in full-duplex mode
- o Test mode for easy maintenance
- o Autoshutdown feature included with TTU9103/9104

Operation

The teleprinter console control contains two 8-bit registers for full-duplex operation. Data transfers between the teleprinter console and the Level 6 processor are bit-parallel and under DMA (Direct Memory Access) control.

While waiting for the buffer to become ready, the computer is free to perform other operations. A program interrupt signals the central processor when further device servicing is required by the software.

Printer

The console printer operates in DMA mode and can print a number of characters that exceed one line in length with a single output command. Software provides format control of a print line both before and after the line is printed including appropriate electromechanical timing delays. The programmer must include terminal specific format characters and delays if the range exceeds one line.

The line length and control character required at the end of the line are determined by the specific console device chosen. Different consoles require a different number of delay characters to allow a carriage return without losing data. The programmer must provide this variation either by making a delay parameter or by using different software packages. If a display is used as a console, it will have the software appearance of a printer.

Keyboard

The console keyboard may be used by the operator to input a message to a DMA block which had been previously set up by software. The range is typically set larger than the message which is to be entered. Software control exists over the type of character editing which is to be performed by the MDC. It defines:

- o The character that will terminate the operation, when entered.
- o The character that will perform a backspace, when entered.
- o The character that will cause the preceding input message to be discarded.
- o A visible escape character so that arbitrary constants may be entered through the keyboard.

The keyboard may also be used to generate an attention interrupt. The first unsolicited character entered on the console is stored in the MDC and the Attention Status bit is set. The device may be programmed so that when it is in the output mode and the break key is pressed, output is prematurely terminated.

Paper Tape Reader

Software must issue a specific control character to the reader to start it. A DMA operation is then set up to read the tape. The record on tape should be equal to or less than the DMA range. The console can be set up by software to issue, under the following conditions, a control character to the reader to turn it off.

- o End of range
- o Specific character encountered on tape
- o Stop I/O from software

The reader responds to the following ASCII control codes:

DC1 or X-ON – Automatically turns reader on.
DC3 or X-OFF – Automatically turns reader off.

NOTE: These characters must be sent via an output order to the TTY. The reader does not stop upon reading the control character.

Paper Tape Punch

Software must issue a specific control character to the punch to turn it on. A DMA operation is then set up and the contents of the DMA block are punched. Where the record to be punched is greater than the DMA range, software sets up a new DMA block at end of range. The console can be set by software to issue a control character to the punch under the following conditions:

- o End of range
- o Break detected
- o Stop I/O from software

The specific control character is programmable and can be used, for example, to turn off the punch.

The punch responds to the receipt of ASCII control codes as follows:

DC2 or TAPE – Turns punch on.
DC4 or TAPE – Turns punch off.

When sending these control characters to the punch, the software must ensure that they are both followed by at least one DEL, i.e.:

DC2, DEL – Turns punch on. These two characters *are not* punched. All subsequent characters are punched until the punch is turned off.

Data Format

The range of the data buffer in memory is expressed in bytes and is even or odd. The starting address may be on any byte boundary. Data read or written by the MDC is packed two bytes per word. In the case of an odd byte (which may be at the start or end of block), only the designated byte may be transferred. The console is set (by software) to operate in either of two modes:

- o 8-bit direct transcription
- o 7-bit with even parity

If 7-bit mode is selected, the MDC generates the parity on output transfers and checks parity on input transfers. Figure 6-6 illustrates the bit designations. Bit designations relative to the holes in the paper tape are shown in Figure 6-7.

Instructions

Table 6-9 lists the I/O commands to which the MDC/Device-Pac/consoles respond. A detailed description of each command follows.

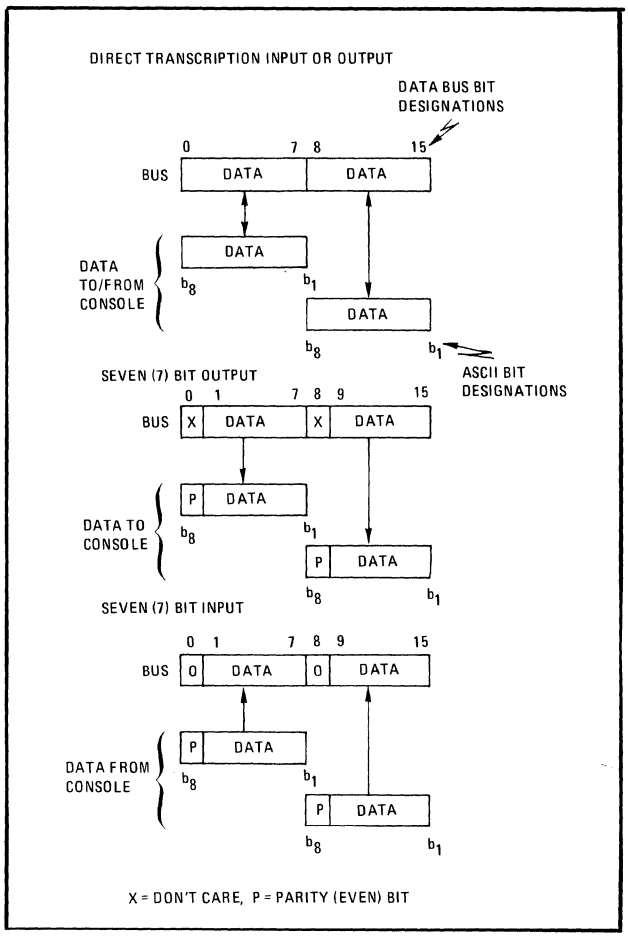


Figure 6-6. Bit Designations

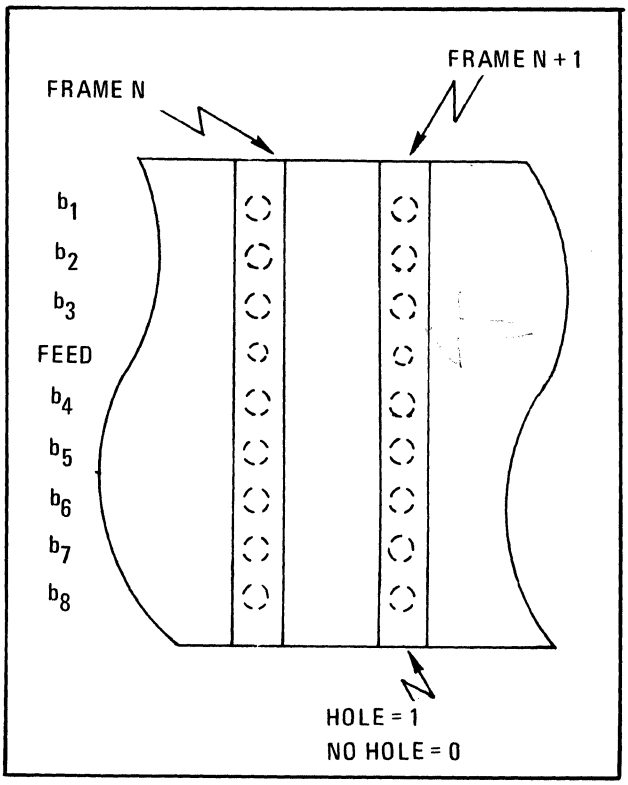


Figure 6-7. Bit Designations on Paper Tape

NOTE: The instructions and programming information presented in this section (exclusive of paper tape operations) also pertains to users of the DKU9101/9102 CRT Keyboard Console Units and the TWU9101/9104/9106 Keyboard Typewriter Consoles.

TABLE 6-9. CONSOLE COMMANDS[a]

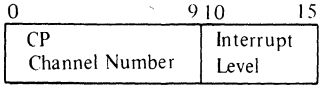
Type	Function Code	Command
Output	03	Output Interrupt Control
	01	Output Control
	09	Output Address and Range
		NOTE: The low order bit of this command specifies the direction of data transfer.
	11	Output Configuration Word A
	13	Output Configuration Word B
	07	Output Task
Input	02	Input Interrupt Control
	0C	Input Range (Residual)
	10	Input Configuration Word A
	12	Input Configuration Word B
	18	Input Status Word
	26	Input Device ID
		TTU9101/9103 – 2018
		TTU9102/9104 – 2019
		TWU9101 – 2018
		TWU9104/9106 – 201C
		DKU9101/9102 – 201A
	08	Input Memory Byte Address
	0A	Input Memory Module Address
	06	Input Task Word
	1A	Input Attention Character

[a]Applicable to TTU9101/9102/9103/9104, DKU9101/9102, and TWU9101/9104/9106.

Output Commands

Output Interrupt Control (03)

Format:



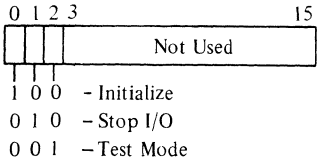
Function:

Loads a 16-bit word into the interrupt control register with the information necessary for

generating an interrupt, i.e., the interrupt level and which CP the interrupt is to be issued to.

Output Control (01)

Format:



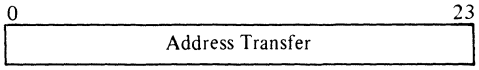
Function:

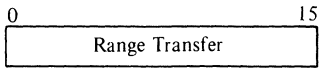
Loads a 16-bit control word into the referenced channel and is accepted unconditionally regardless of any channel busy status. The channel may respond with a WAIT, but never with a NAK. The individual bits cause the following specific action to occur:

- o Initialize
  - – Causes the MDC to run its resident logic test
  - – Clears all MDC Device-Pacs
  - – Operates on all channels of MDC
  - – Makes all channels of MDC non-busy
  - – Blocks interrupts
- o Stop I/O
  - – Causes interrupt when termination is complete (if enabled)
  - – Causes Control Character 3 to be sent
  - – Does not affect other channels
  - – Causes channel to terminate its I/O operation and update status
- o Test Mode
  - – See paragraph entitled "Test Mode."

Output Address and Range (09)

Format:





Function:

The IOLD command when executed by the processor creates two bus transfers to the MDC. The first is the 24-bit address transfer and the second is the 16-bit range transfer. To the MDC, these are two separate and distinct bus transfers with two separate and distinct function codes of 09 for the address transfer and 0D for the range transfer. The programmer need only specify the first function code and the processor hardware/firmware automatically calculates the second function code (by adding 04 to it).

- o Address Transfer – A 24-bit quantity transferred to the MDC to be used as the starting byte address of the data record in memory which is to be transferred.
- o Range Transfer – A 16-bit quantity transferred to the MDC to be used as the byte range count of the data record in memory which is to be transferred. Range is specified as a twos complement integer which must be positive for the MDC. Therefore, for the MDC, range is:

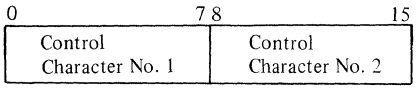
$$1 \leqslant r \leqslant 2^{15}-1$$

The range transfer includes an implicit start I/O command. When received by the MDC it causes the addressed channel to go busy and perform the operation indicated by the previously supplied I/O commands.

The least significant bit (LSB) of the channel number (address bus bit 17) designates direction for this particular DMA transfer. Output transfers have the LSB of the channel number = 1. Input transfers have the LSB = 0.

Output Configuration Word A (11)

Format:



Function:

Outputs a word consisting of two control characters necessary during the execution of an input command.

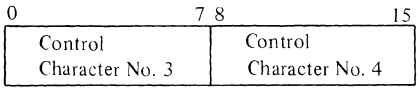
- o Control Character 1, when detected, causes the console to perform a backspace (i.e., the last character entered is eliminated from the input buffer). This is applicable to keyboard operations only.
- o Control Character 2, when detected, causes the console to terminate the input order,

post the appropriate status bit and send an interrupt (if interrupt is allowed).

Following an initialize, Control Characters 1 and 2 are reset to zero.

Output Configuration Word B (13)

Format:



Function:

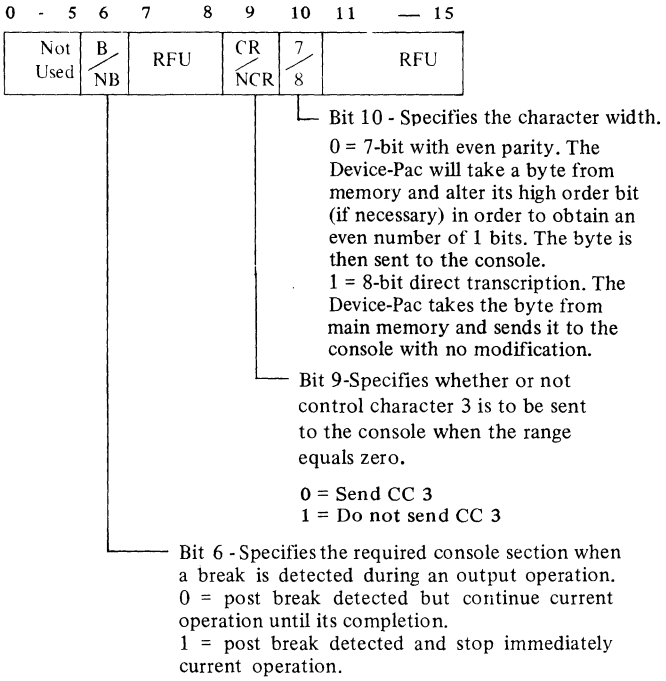
Outputs a word consisting of two control characters necessary during the execution of an input or output command.

- Control Character 3 is used differently as a function of the command type. During the execution of an input command type, detection of Control Character 3 causes the console to terminate the input order. In addition, upon termination of an input order (either normally or abnormally) Control Character 3 is sent to the console to position the carriage to its home position or stop the tape reader.
  During the execution of an output command type, Control Character 3 is sent to the console, if so specified in the command, with the purpose of positioning the carriage to its home position or stopping the paper tape punch.
- Control Character 4 applies to input operations only. When received from the console, this character is discarded by the MDC and the character immediately following is input to memory regardless of its value. Control Character 4 is used as an escape character so that the operator can input a character that was equal to CC1, CC2, or CC3 without causing the actions described for them. If Control Character 4 is set to zero, which it is following an initialize, none of the escape actions described here occur.

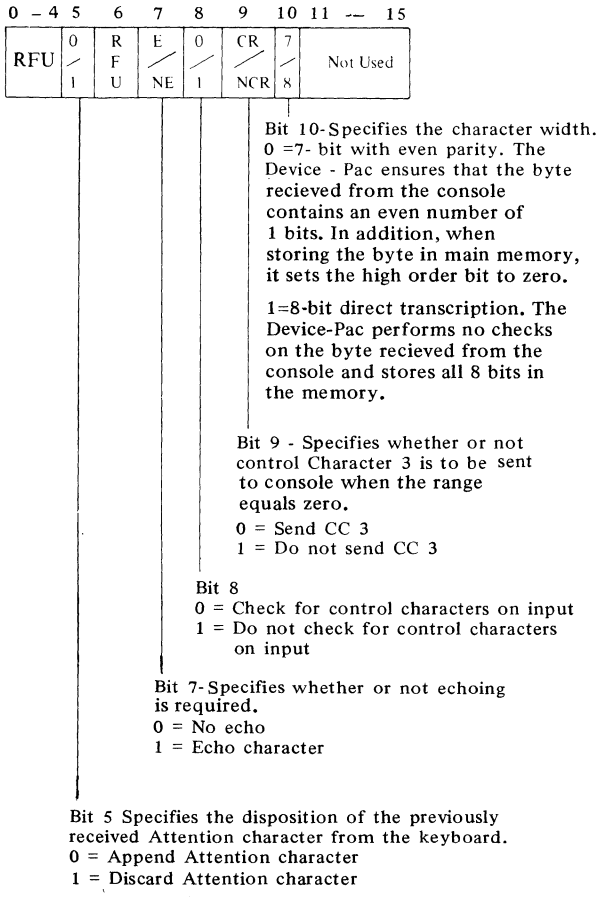
Output Task (07)

Format:

For an *output* operation (print or punch) the task word has the following format:



For an *input* operation (keyboard or paper tape read), the task word has the following format:



Function:

Outputs a word that determines the specific action to be performed. Bit interpretations differ for output and input operations. When this word is received by the MDC it is stored. The next IOLD command which is issued specifies the direction and, therefore, the interpretation. The task word remains stored in the Device-Pac until a new task word is issued or initialize occurs.

An input command is used to either accept a message from the keyboard or read a block (or part of it) from a paper tape mounted on the paper tape reader. For a detailed description of the read keyboard or paper tape operation see "Programming Considerations" later in this section.

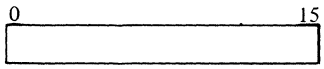
An output command is used to perform one of the following operations:

- o Print/display a message to the operator
- o Format the printer/display
- o Start a paper tape punch
- o Punch paper tape
- o Start a paper tape reader

Input Commands

Input Interrupt Control (02)

Format:

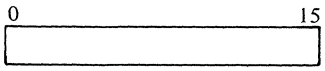


Function:

Causes the channel to place the contents of its interrupt control register on the data bus.

Input Range (0C)

Format:

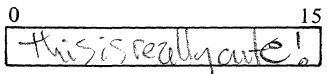


Function:

Causes the channel to place the contents of its channel register on the data bus.

Input Configuration Word A (10)

Format:

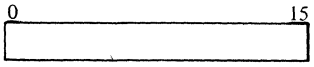


Function:

Causes the channel to place the contents of its configuration word A register on the data bus.

Input Configuration Word B (12)

Format:

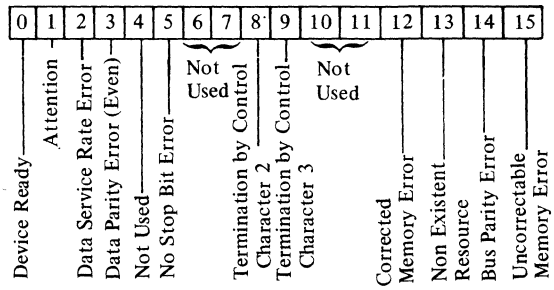


Function:

Causes the channel to place the contents of its configuration word B register on the data bus.

Input Status Word (18)

Format:



Function:

Causes the channel to place the contents of the status register on the bus. See also Table 6-10.

TABLE 6-10. STATUS BIT DEFINITIONS – CONSOLE

Status Condition	Bit	Definition	Reset by
Device Ready	0	In current loop connections, the console is connected by the presence of the output current when the Device-Pac is not busy (i.e., not sending data to the console). In EIA connections, Data Terminal Ready for direct connections or Carrier Detect for data set connection.	Change of condition
Attention	1	Operator pressed a key on the console for software attention.	Input status word command[b]

TABLE 6-10 (CONT). STATUS BIT DEFINITIONS – CONSOLE

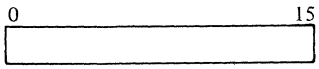
Status Condition	Bit	Definition	Reset by
Data Service Rate Error[a]	2	The MDC/bus/CP have failed to meet the teleprinter speed and data has been lost.	Next IOLD command[b]
Data Parity Error (Even)[a]	3	Device-Pac detected a data parity error. Only applicable if the task word was set to specify 7-bit mode (Bit 10 = 0).	Next IOLD command[b]
No Stop Bit Error[a]	5	A character was received from the console without a stop bit.	Next IOLD command[b]
Termination by Control Character 2	8	Control Character 2 was detected in the input stream and caused a termination.	Next IOLD command[b]
Termination by Control Character 3	9	Control Character 3 was detected in the input stream and caused a termination.	Next IOLD command[b]
Corrected Memory Error	12	The data read from memory was accompanied by a signal indicating an error was corrected.	Next IOLD command[b]
Nonexistent Resource	13	Set whenever the MDC attempts a Write or Read request bus cycle and receives a NAK response.	Next IOLD command[b]
Bus Parity Error	14	Set whenever the MDC detects a parity error on either byte of the Data Bus during any Output bus cycle (i.e., odd function code), during a second-half memory read cycle or when a parity error is detected in bits 0–7 of the Address Bus during an Output Address command.	Input status word command[b]
Uncorrectable Memory Error	15	The data read from memory was accompanied by a signal indicating an error existed that the memory could not correct.	Next IOLD command[b]

[a]These bits relate to the state of the last character received on input and will remain set until the next character is received without these conditions.

[b]Initialize (Output Control Word) and Master Clear also reset these status bits.

Input Device ID (26)

Format:

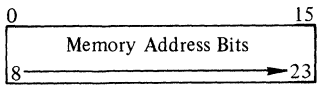


Function:

Causes the channel to place its device identification number (2018 for the TTU9101/9103 and TWU9101; 2019 for the TTU9102/9104; 201A for the DKU9101/9102; 201C for the TWU9104/9106) on the bus.

Input Memory Byte Address (08)

Format:



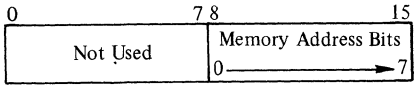
Function:

Causes the channel to place the 16 least significant bits of its DMA address register on the data bus. The bits represent the memory address register bits 8–23.

NOTE This command will violate memory protection.

Input Memory Module Address (0A)

Format:



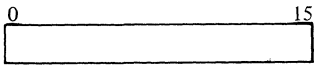
Function:

Causes the channel to place the eight most significant bits of its DMA address register on the data bus. The bits represent memory address bits 0–7.

NOTE: This command will violate memory protection.

Input Task Word (06)

Format:

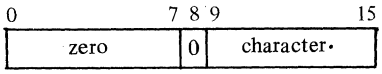


Function:

Causes the channel to place the contents of its task word register on the bus.

Input Attention Character (1A)

Format:



Function:

Causes the software to input the character that was the cause of the previous Attention status. Once a character is placed in this buffer, it remains until another key is pressed after receipt of an IOLD for input mode.

Programming Considerations – Console

The operations required by the software to control the console are the following:

- o Load Configuration
- o Read Keyboard (i.e., read a message entered by the operator through the keyboard)
- o Print/Display (i.e., print or display a message to the operator on the printer
- o Read Status
- o Stop I/O (i.e., stop the I/O operation currently in progress)
- o Attention (i.e., react to the detection of an operator-generated Break request)

Load Console Configuration Operation

Prior to executing a read or print/display operation, the software must preset the console by loading it with the appropriate configuration word(s) to ensure proper operation execution.

The configuration information need be loaded only once before a series of console operations and remains unchanged until a paper tape operation is performed. Thus any console operations which follow at least one paper tape operation must be preceded by a load configuration operation.

Two commands are available to load the console configuration information:

- o Output Configuration Word A
- o Output Configuration Word B

The contents of the configuration words should be as shown in Table 6-11. The console control characters and character set are shown in Table 6-12.

TABLE 6-11. CONTENTS OF CONFIGURATION WORDS – CONSOLE

Operation Performed	Configuration Word A Consists of 2 Bytes		Configuration Word B Consists of 2 Bytes	
	Set Control Character 1 to:	**Set Control Character 2 to:**	**Set Control Character 3 to:**	**Set Control Character 4 to:**
Read	@ (Note 1)	CAN (Note 1)	CR (Note 2)	\ (Note 1)
Print/Display	Not Used Set to 00	Not Used Set to 00	CR (Note 2)	Not Used Set to 00

NOTES: 1. This specific character is chosen by software convention; however, other values are permitted.
2. This specific character is required by the ASR console peripheral. Any other character configured here would be output to the device and would cause an action determined by the specific peripheral used.

TABLE 6-12. CONSOLE CODE SET

Bits b_7 b_6 b_5					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
b_4	b_3	b_2	b_1*	Row ↓ Column →	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	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	—	o	DEL

*b_1 is low order bit

All characters in these two rows + SP (space) and DEL (delete) are non-printing.

"Fold-over Printing" means that lowercase characters received by model 33s are actually printed as their uppercase equivalent. Codes shown in columns 6 & 7 of the chart "fold-over" into columns 4 & 5 respectively, (except for "DEL").

Read Keyboard Operation

A Read operation is initiated by the software to allow an operator to generate a message via the keyboard. The following commands must be issued by the software to the console to initiate the read operation:

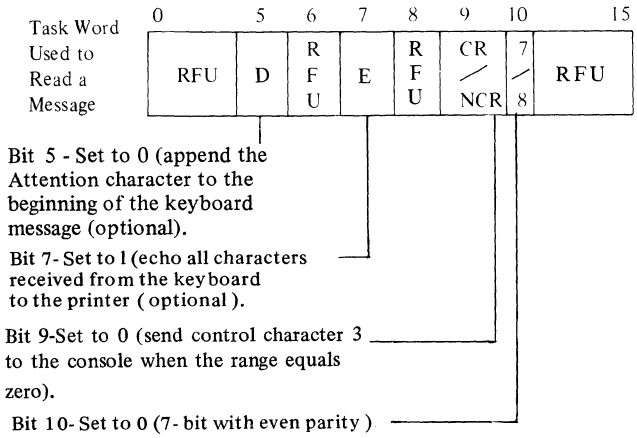
- o Output Task
- o Output Address and Range (IOLD)

The sequence in which the above listed commands must be issued is as shown; otherwise unspecified results occur. However, if a series of Read Keyboard operations is to be performed, then the Output Task command need be sent only once. All subsequent Read operations need only the Output Address and Range command. Typically, the software initiates a Read operation with interrupt allowed and sets the range to 72 characters. Thus it is interrupted only after the receipt of a complete message.

If operating in noninterrupt mode, the software must periodically issue an Input Status Word command and test the I bit of the CP I register. This bit reflects the state of the attachment (i.e., busy or not).

The range should normally be set to 72 but if set to less, then upon detection of range equal to zero, the attachment automatically terminates the order and performs a carriage return.

A Read operation is initiated upon receipt of the Output Address and Range command. The channel number has its lower bit set to zero. It should be set as follows:



The functions performed during execution of the Read operation are as follows:

- o Accept the characters typed by the operator, store them in the main memory buffer, and print them on the printer.
- o Edit the text in conjunction with the software. Specifically, upon detection of the control character:
  @ – Eliminate the last valid entered character from the main memory buffer by updating the range and address pointers. (The deleted data character is not erased from the main memory buffer but is overwritten by the next data character, if any.)
  CAN – Inform the software that the operator desires to cancel the line just entered. The software then reissues the Read operation.
  \ – Discard the control character and input the following character regardless of its value. This is an escape mode which allows the input from the keyboard of a character with a value equal to one of the other three control characters without the usual action being taken.

Print/Display Operation

A Print/Display operation is initiated by the software to output a message to the operator. The following commands must be issued by the software to the console attachment to initiate the Output operation:

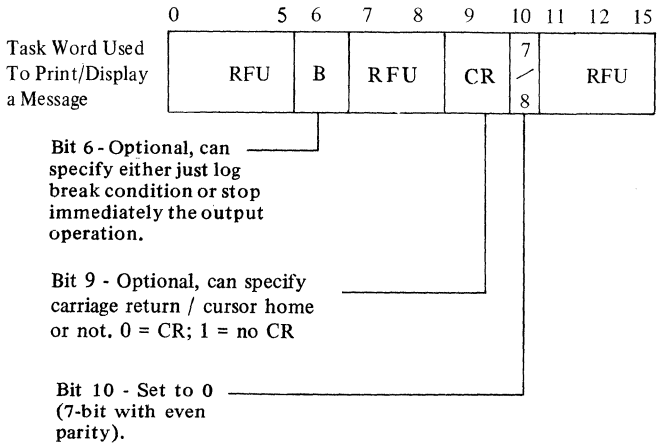
- o Output Task
- o Output Address and Range (IOLD)

The sequence in which the above commands must be issued is as shown; otherwise unspecified results will occur. However, if a series of similar Print/Display operations is to be performed, then the Output Task command need be sent only once. All subsequent Print/Display operations need only the Output Address and Range command.

Typically, the software initiates an Output operation with interrupt allowed and thus is not interrupted until the entire message has been printed/displayed.

If operating in noninterrupt mode, then the software must periodically issue an Interrupt Status Word 1 command and test the I bit of the CP I register. This bit reflects the state of the attachment (i.e., busy or not).

An Output operation is initiated upon receipt of the Output Address and Range command. The channel number has its lower order bit set to one. The Task Word should be set as follows:



The normal mode of usage is to print one line at a time (72 characters or less) and automatically perform a carriage return, if so requested. If it is desired to print more than one line, the test has to contain every 72 characters or less (one line length):

CR, LF, plus delay characters required by the device

Failure to do this results in loss of the data characters beyond character number 72.

Console Read Status Operation

A Read Status operation is initiated by the software in order to determine the outcome of an Input or Output operation or upon receipt of an unsolicited (Attention) interrupt. The following commands are issued by the software to the console:

- o Position the carriage/cursor at the home position upon termination of the Read operation. Then set the attachment in the unbusy state and send interrupt if so required.
- o After a Read Keyboard Operation
  – Input status
  – Input range
- o After a Print Operation
  – Input status
- o Upon receipt of an unsolicited (Attention) interrupt
  – Input Status
  – Input Character

Following a Read Keyboard operation, the software must retrieve the residual range in order to determine the length of the input message.

Typically, the software is interrupt-driven and thus performs a Read Status operation only upon receipt of an interrupt.

If operating in noninterrupt mode, the software performs a Read Status operation following the execution of an input or output operation in order to determine its completion and outcome.

Console Stop I/O Operation

A Stop I/O operation is initiated by the software to terminate gracefully an outstanding Input or Output operation. It is issued via an Output Control command having its bit 1 set to a one. Assuming no hardware faults in the controller, it terminates the current order and enters the nonbusy state. The console also sends an interrupt, if allowed.

Although the Stop I/O can be issued to terminate both Input and Output operations, it is expected that it will be used mainly to terminate outstanding Read Keyboard operations. This results in those cases when no operator or response is detected following the expiration of a software-controlled elapsed time.

Attention

Attention is the mechanism used by the operator to establish a dialog with the system software. It is invoked by the operator, by pressing a key on the keyboard.

Typical usages of this mechanism are:

- To make an inquiry
- To answer a question
- To terminate an output operation
- To direct the system software

The functions performed by the Device-Pac upon detection of an Attention are a function of the Device-Pac state:

- For Device-Pac Ready (not busy)
  - Store the Attention character in the MDC R/W memory in status word 2
  - Send an interrupt to the software, if so allowed
  - Append the Attention character to the beginning of the next input message, if so requested, and echo to the console as verification to the operator.
- For Device-Pac Busy (performing output)
  - Store the Attention character in MDC R/W memory
  - If character is a break, check bit 6 of task word, terminate output and interrupt (bit 6 = 1)
  - Otherwise, just set Attention status and wait for normal termination of output.
- Attention does not pertain to input operations.

The reaction of the system software to an Attention will be:

- Output some sort of an acknowledgement message on the printer or display (e.g., a question mark)
- Issue a read keyboard command to accept the operator's message or command.

The Device-Pac will guarantee that only one Attention interrupt will result for every depression of a key. The operator may cause an Attention interrupt by striking a key on the keyboard when the Console Device-Pac firmware is not processing input or output orders. The software determines that the interrupt was an Attention interrupt by status bit 1 and can determine the specific character input by issuing the Input Attention Character command. If the BREAK key was the cause of the Attention interrupt, the characters will be all zeros and the No Stop Bit status (bit 5) will be set. In cases where more than one key is pressed in succession, the attachment retains only the first character and discards succeeding characters until an input order or Stop I/O command occurs.

When the Device-Pac is in output mode (printing or punching), any key pressed causes the character to be retained in the MDC and the Attention status bit to be set. The interrupt is generated and the output data stream stopped only if the BREAK key was pressed and the output task word has bit 6 set to 1. Otherwise, the software becomes aware of the Attention at the normal Termination interrupt and may determine the character at that time.

Programming Considerations – Paper Tape

The operations required by the software to control the paper tape reader and punch are the following:

- Load Configuration
- Read Paper Tape
- Punch Paper Tape
- Read Status
- Stop I/O

Load Paper Tape Configuration Operation

Prior to executing a Paper Tape Read or Punch operation, the software must preset the console Device-Pac by loading it with the appropriate configuration word(s) to ensure proper execution.

The configuration information need be loaded only once before a series of Paper Tape opera-

tions and remains unchanged until a Console operation is performed. Thus any Paper Tape operations which follow at least one Console operation must be preceded by a Load Configuration operation.

Two commands are available to load the paper tape configuration information:

- o Output Configuration Word A
- o Output Configuration Word B

The contents of the configuration words should be as shown in Table 6-13. The specific characters in Table 6-13 are required by the TTU9101/9103. Any other character configured here would be output to the device and would cause an action determined by the specific peripheral used.

Read Paper Tape Operation

A Read operation is initiated by the software to read either one entire tape block or portion of a tape block. The following commands must be issued by the software to the console to initiate the Read operation:

- o First the reader must be started via an output command sequence which sends one X-ON control character to the reader:
  - – Output Task and
  - – Output Address and Range (IOLD).
- o Then the read order itself is issued via an input command sequence:
  - – Output Task and
  - – Output Address and Range (IOLD).

These commands must be issued in the sequence shown; otherwise unspecified results will occur. In addition, as soon as the first command sequence (that starts the reader) is completed, the second command sequence must be initiated to ensure no loss of information. The time between the two sequences should be less than two character frames (approximately 180 ms).

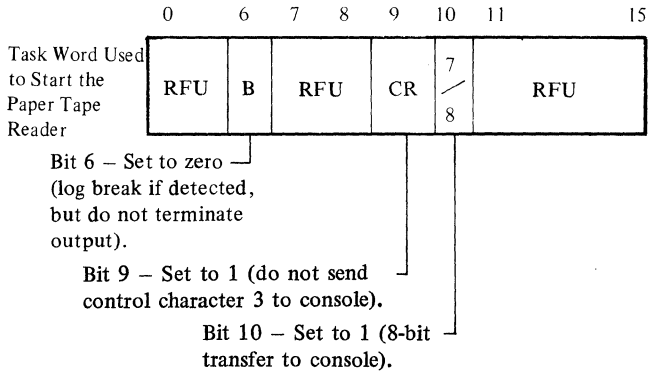
Typically, the software initiates a Read operation with interrupt allowed and reads one entire block. Thus it will be interrupted twice:

- o After starting the reader
- o After one entire block has been read and the tape is stopped in the interblock gap.

If operating in noninterrupt mode, then the software will have to periodically issue an Input Status Word command and test the I bit of the CP I register, to determine whether or not the device is still busy.

In addition, if the software reads a block in 2 or more segments, the reader is not capable of stopping between tape frames. Thus, the software must initiate a new Read operation in less than one character time (approximately 90 ms) or data will be lost.

A Read operation is initiated by starting the reader via an output sequence. It is initiated upon receipt of the Output Address and Range command issued to the output channel (low order bit of channel number is set to 1). It should be set as follows:



The Output Address and Range command should point to one X-ON control character (HEX code = 11).

TABLE 6-13. CONTENTS OF CONFIGURATION WORDS – PAPER TAPE

Operation Performed	Configuration Word A Consists of 2 Bytes		Configuration Word B Consists of 2 Bytes	
	Set Control Character 1 to:	Set Control Character 2 to:	Set Control Character 3 to:	Set Control Character 4 to:
Read	TAPE	TAPE	X-OFF	Not Used Set to 00
Punch	Not Used Set to 00	Not Used Set to 00	TAPE	Not Used Set to 00

The functions performed during the read sequence are:

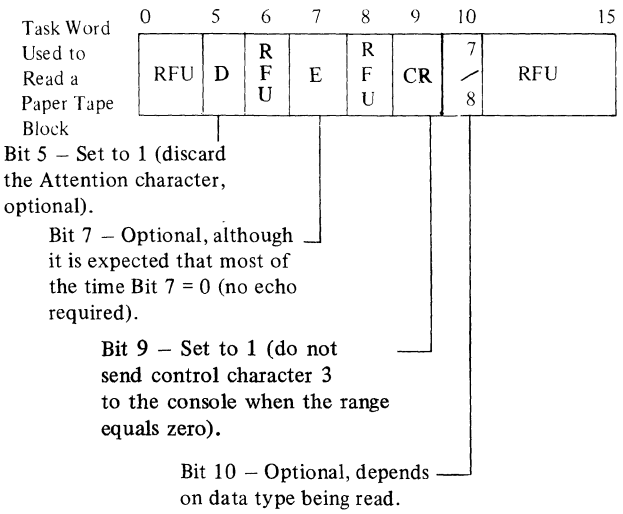
- o Accept the characters read by the reader and store them in the main memory buffer.
- o Print the data read, if so specified.

The functions performed during the start reader sequence are:

- o Send one X-ON character to console (software)
- o If operation terminates successfully, set the console to the unbusy state and send interrupt, if required.
- o If operation terminates because a Stop I/O, Break, or error condition was detected, STOP the reader by sending it Control Character 3 (set to X=OFF).

NOTE: When performing a read operation, the configuration parameters are set to the read state although both an output and input sequence are executed.

Next a read sequence must be issued. The read operation is initiated upon receipt of the Output Address and Range command issued to the input channel. The channel number will have its lower order bit set to zero. It should be set as follows:



- o If Read operation terminates because range = 0, do not stop the reader but set it to the unbusy state and send interrupt, if so required.
- o If Read operation terminates because Control Character 2 (set to TAPE) or Control Character 3 (set to X-OFF) is detected, then stop the reader by sending it Control Character 3. Then set the console to the unbusy state and send interrupt, if so required.

Punch Paper Tape Operation

A Punch operation is initiated by the software to generate either one entire tape block or portion of a tape block. The following commands must be issued by the software to the console to initiate the output operation:

- o Output Task
- o Output Address and Range (IOLD)

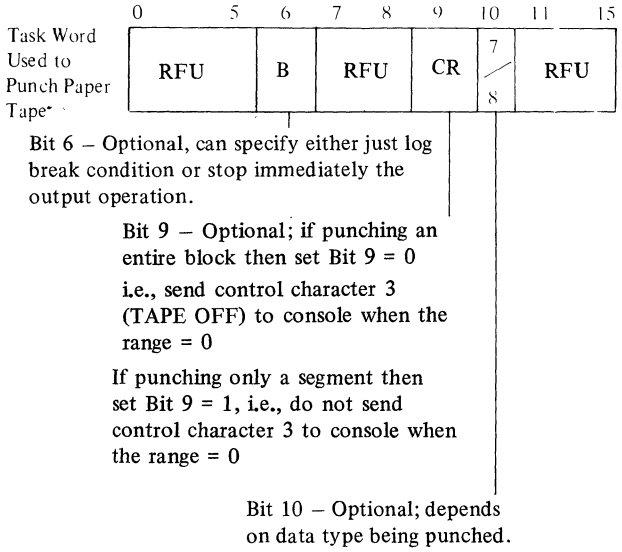
These commands must be issued in the sequence shown; otherwise unspecified results will occur. However, if a series of paper tape punch operations is to be performed, then the output task command need be sent only once. All subsequent punch operations need only the output address and range command.

Typically, the software initiates a punch operation with interrupt allowed and punches one entire block. Thus it will be interrupted only after the entire block has been punched.

If operating in noninterrupt mode, the software will have to periodically issue an Input Status Word command and test the I bit of the CP I register to determine whether or not the device is still busy.

If the software punches a block in 2 or more segments, the punch stays on between segments and thus no console Print/Display operation should be initiated until the punch is stopped (or the data printed/displayed will also appear on the tape).

The Punch operation is initiated upon receipt of the Output Address and Range command. The channel number will have its lower order bit set to 1. It should be set as follows:



The functions performed during the Punch operation are:

- o Start the punch. This is a software function and requires that two control characters (TAPE, DEL) be issued to the console. Normally these are the first two characters in the punch buffer; however, they may be issued separately.
- o Punch the block or a segment of it.
- o Stop the punch if so specified after a normal termination. Then set the attachment to the unbusy state and send it an interrupt if so required.
- o Stop the punch if either a Stop I/O or Break is detected. Then set the attachment to the unbusy state and send an interrupt if so required.

Paper Tape Read Status Operations

A Read Status operation is initiated by the software in order to determine the outcome of an input or output operation. The Input Status Word command is issued by the software to the console.

Typically, the software is interrupt driven and thus performs a Read Status operation only upon receipt of an interrupt.

If operating in noninterrupt mode, the software performs a Read Status operation following the execution of an Input or Output operation in order to determine its completion and outcome.

Paper Tape Stop I/O Operation

A Stop I/O operation can be initiated by the software to terminate gracefully an outstanding Input or Output operation. It is issued via an Output Control command having its bit 1 set to 1. Assuming no hardware faults in the attachment, it terminates the current order and enters the nonbusy state. The attachment also generates an interrupt if allowed.

It is not expected that Stop I/O operations will be used to stop paper tape operations. Nevertheless, there might be situations caused by operator mistakes, (i.e., a paper tape is to be read, but it was not mounted) where the Stop I/O operation might be required.

CRT KEYBOARD CONSOLES

The DKU9101/9102 CRT Keyboard Console Units permit conversational message transfer, status display, and operator control of any Level 6 system. The CRT units (Figure 6-8) display 64 uppercase ASCII characters and 95 uppercase and lowercase ASCII characters, including space, respectively. Utilizing the latest in computer technology, they feature a high level of reliability at a relatively low price. CRT specifications are listed in Table 6-14.

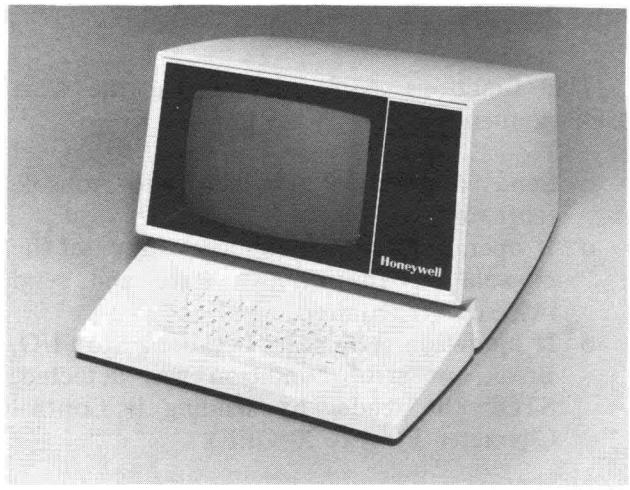


Figure 6-8. Level 6 CRT Keyboard Console

The DKU9101/9102 consists of a CRT display unit with a detachable keyboard unit permitting flexible operation and placement. The CRT interfaces with the Level 6 processor by means of a single-board Multiple Device Controller (MDC9101), a Console Device-Pac (KCM9101), and a 25-foot cable.

Features

- o Displays 960 characters on 12 lines of 80 characters each
- o Roll-up line feed allows data to be entered on the bottom line
- o Keyboard has 60 encoded keys with multikey depression protection (*n*-key rollover)
- o Keyboard generates 97 ASCII codes using SHIFT, UNSHIFT, and CONTROL keys and optionally generates 128 ASCII codes when CAP LOCK key is not enabled
- o BREAK, ESCAPE, and DELETE keys provide flexibility to allow application software interpretation
- o Allows two-way simultaneous transmission on full-duplex link
- o Audible alarm sounds upon BEL code from processor
- o Displays keyboard input upon return (echo) from the connected processor
- o Transmission rate varies by steps from 75 to 9600 bits per second

TABLE 6-14. CRT KEYBOARD CONSOLE SPECIFICATIONS

Characteristics \ Device	CRT Consoles (DKU) 9101	9102
Keyboard	TTY layout; 60 keys; *n*-key rollover; 10 control keys	
Display		
Total number of characters per screen	960	
Total number of characters per line	80	
Total number of lines	12	
Character matrix	5 X 7	
Displayable ASCII characters	97	128
Character size	0.08 in. x 0.16 in. (2.0 mm x 4.0 mm)	
Displayable screen size	54 square in. (348 square cm)	
Device Interface	Each CRT requires its own Device-Pac (KCM9101)	
Physical Dimensions		
Keyboard		
Width	17.9 in. (45.5 cm)	
Depth	8 in. (20.3 cm)	
Height	3 in. (7.6 cm)	
Display		
Width	18.1 in. (46.0 cm)	
Depth	17.5 in. (44.4 cm)	
Height	13.1 in. (33.3 cm)	
Depth of combined display and keyboard	23.7 in. (60.2 cm)	

Keyboard

The console keyboard utilizes solid-state, high-reliability switches as keys. The keyboard permits the entry of variable data and program parameters. The 60 keys can generate 128 characters of the ASCII code set. The characters include 26 alphabetic, 10 numeric, and 32 special symbols. Also included are 10 control keys. See Figure 6-9.

A special entry marker (cursor) appears on the console display screen to indicate the location of the next character to be displayed. The DKU9101/9102 uses the "bottom line entry" approach for the display. The cursor moves only on the bottom line of the display. The bottom line display operates just like the print line of a journal roll, i.e., a line feed causes the entire display page (including the bottom line) to move up one line, leaving the new bottom line blank, or clear. A line return causes the cursor to move to the left margin first display position.

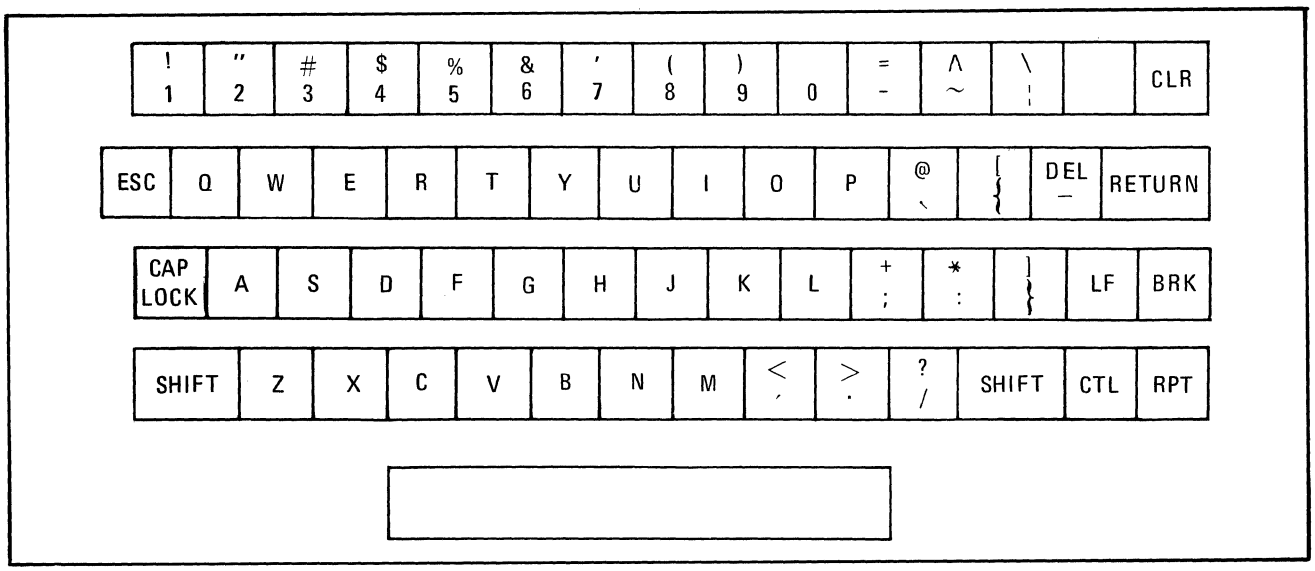


Figure 6-9. DKU9101/9102 Keyboard

Display Screen

Manually entered data and processor-generated inquiries and responses are displayed on a 12-inch CRT. With a display of up to 960 characters (12 lines, 80 characters per line) and a 60-frames per second refresh rate, the display projects clear, bright, easily read information. The operator has access to all character positions within the bottom line.

Two-Way Audible Alarm

An operator-attention alarm sounds whenever the console receives a BEL code or when a character is displayed in the 75th character (or column) position.

Operational Enhancements

Nondestructive cursor movement, backward and forward, a step at a time within the bottom line provides for applications enhancement. The entire screen can be cleared (erased) by pressing a single key.

The transmission rate is variable by steps between 75–9600 bps. Character size may include one or two stop bits as well as a choice for a parity bit to be odd, even, or not needed. An extension 4-wire RS-232-C voltage level interface port is provided to accommodate an auxiliary device, e.g., a serial printer.

Instructions

The instructions and programming information presented for the teleprinters (excluding paper tape operations) are also applicable for the CRT Keyboard Consoles.

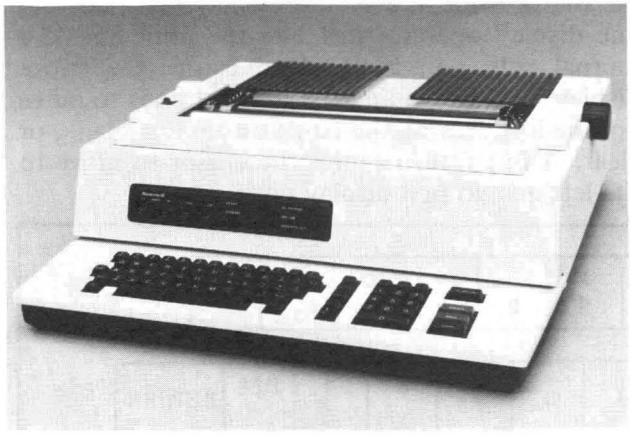
KEYBOARD TYPEWRITER CONSOLES

The TWU9101/9104/9106 Keyboard Typewriter Consoles are available for console and low-speed printer use (Figure 6-10). The TWU9101 has a 64-character ASCII character set and prints at 30 characters per second (cps). The TWU9104 and TWU9106 have a 96-character ASCII character set and print at 30 cps and 120 cps, respectively. All have 132 print positions. Keyboard typewriter console specifications are listed in Table 6-15. Keyboard layouts are shown in Figures 6-11 and 6-12.

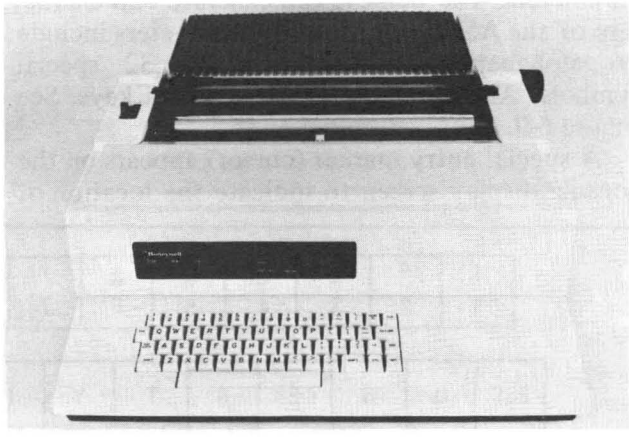
The keyboard typewriter consoles interface with the Level 6 processor by means of a single-board Multiple Device Controller (MDC9101), a Console Device-Pac (KCM9101) and a 26-foot cable.

Features

- Print speeds of up to 30 or 120 characters per second



TWU9101

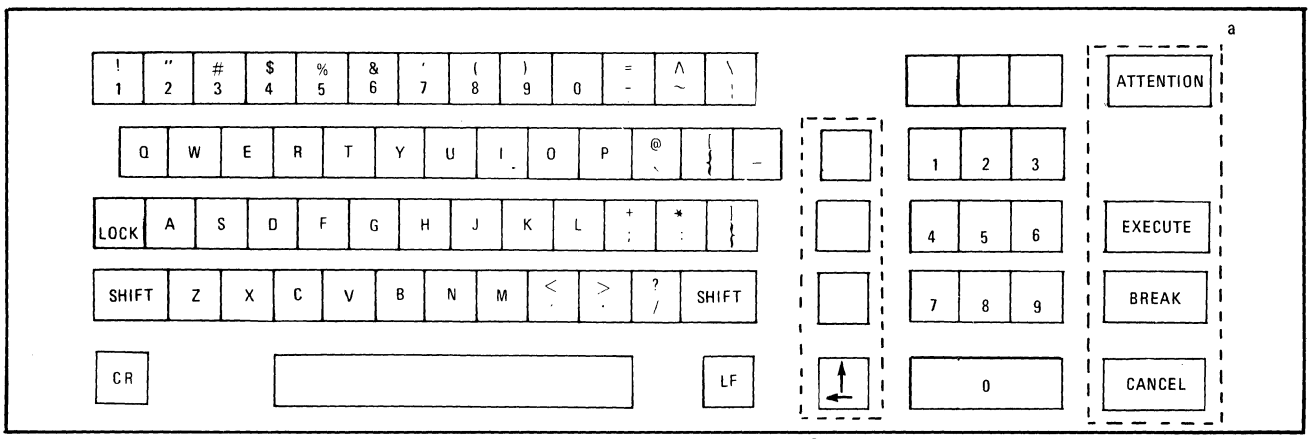


TWU9104/9106

Figure 6-10. Level 6 Keyboard Typewriter Consoles

TABLE 6-15. KEYBOARD TYPEWRITER CONSOLE SPECIFICATIONS

Characteristics \ Device	(TTU) 9101	(TWU) 9104	(TWU) 9106
Print Speed (cps)	30	30	120
Print Format	10 cpi, horizontal; 6 cpi, vertical		
Character Set (ASCII)	64-char.	96-char.	96-char.
Line Feed Time (ms) – nominal	60	60	60
Carriage Return Speed (ms)			
Nominal for 132 columns	380	380	380
Nominal for 80 columns	280	280	280
Print Ribbon	Cartridge type, replaceable by operator		
Matrix Font	7 X 9 dot; equivalent to 10-point type		
Device Interface	Each keyboard typewriter console requires its own Device-Pac (KCM9101)		
Paper Stock	Standard continuous fanfold paper forms with feed holes on each edge with or without margin perforations; 3.0 in. (7.62 cm) to 17 in. (43.2 cm) forms length; 4.0 in. (10.2 cm) to 15 in. (38.1 cm) forms width		
Physical Dimensions			
Height	7.5 in. (19.05 cm)		
Width	22.5 in. (57.15 cm)		
Depth	26.0 in. (66.04 cm)		
Weight	66 lb (29.7 kg)		



[a]These Keys are currently not implemented.

Figure 6-11. TWU9101 Console Keyboard

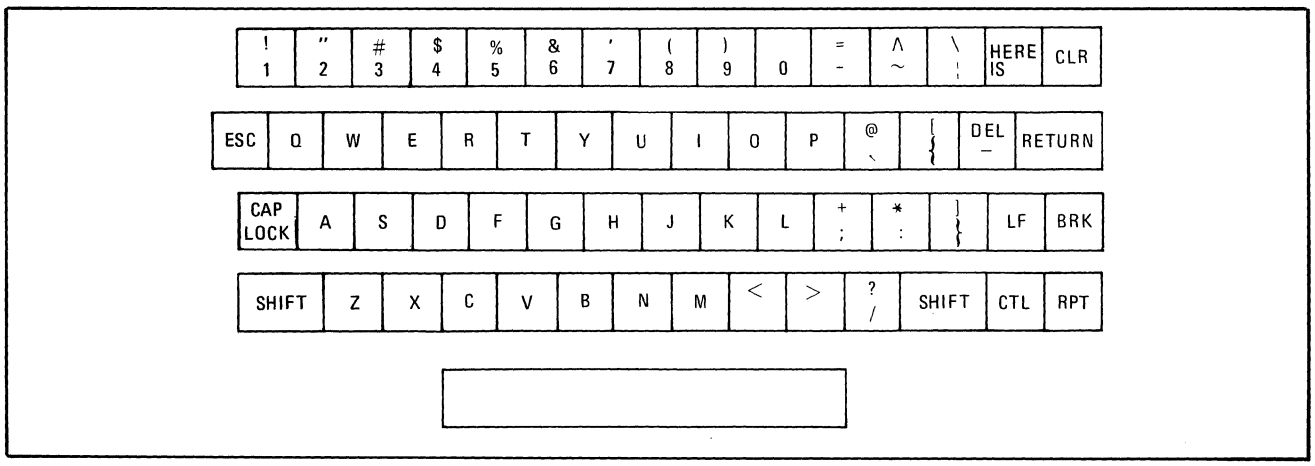


Figure 6-12. TWU9104/9106 Console Keyboard

- o Original and up to four carbon copies
- o Uppercase and lowercase type (TWU9104/9106)
- o Up to 132 print positions
- o Paper-out sensor
- o No fill characters required after CR/LF

Programmed Operations

- o Print message on console typewriter
- o Space paper
- o Accept message from keyboard

Instructions

The instructions and programming information presented for the teleprinters (excluding paper tape operations) are also applicable for the keyboard typewriter consoles.

DISKETTES

The DIU9101 Single Diskette and the DIU9102 Dual Diskette (Figure 6-13) provide low-cost, versatile disk storage. Data is recorded on the magnetic-oxide-coated surface of an 8-inch flexible mylar disk (diskette). Each removable diskette has a formatted data capacity of 256,256 bytes. Diskette specifications are listed in Table 6-16.

A DIM9101 Diskette Device-Pac is required and supports either two DIU9101 Single Diskettes or one DIU9102 Dual Diskette. The diskettes, which are offered in convenient, rack-mountable or freestanding tabletop versions, interface with the Level 6 processor by means of a single-board Multiple Device Controller (MDC9101), a Diskette Device-Pac (DIM9101), a 6- or 15-foot cable, and a power supply. There is a maximum of two Device-Pacs/four diskettes per MDC.

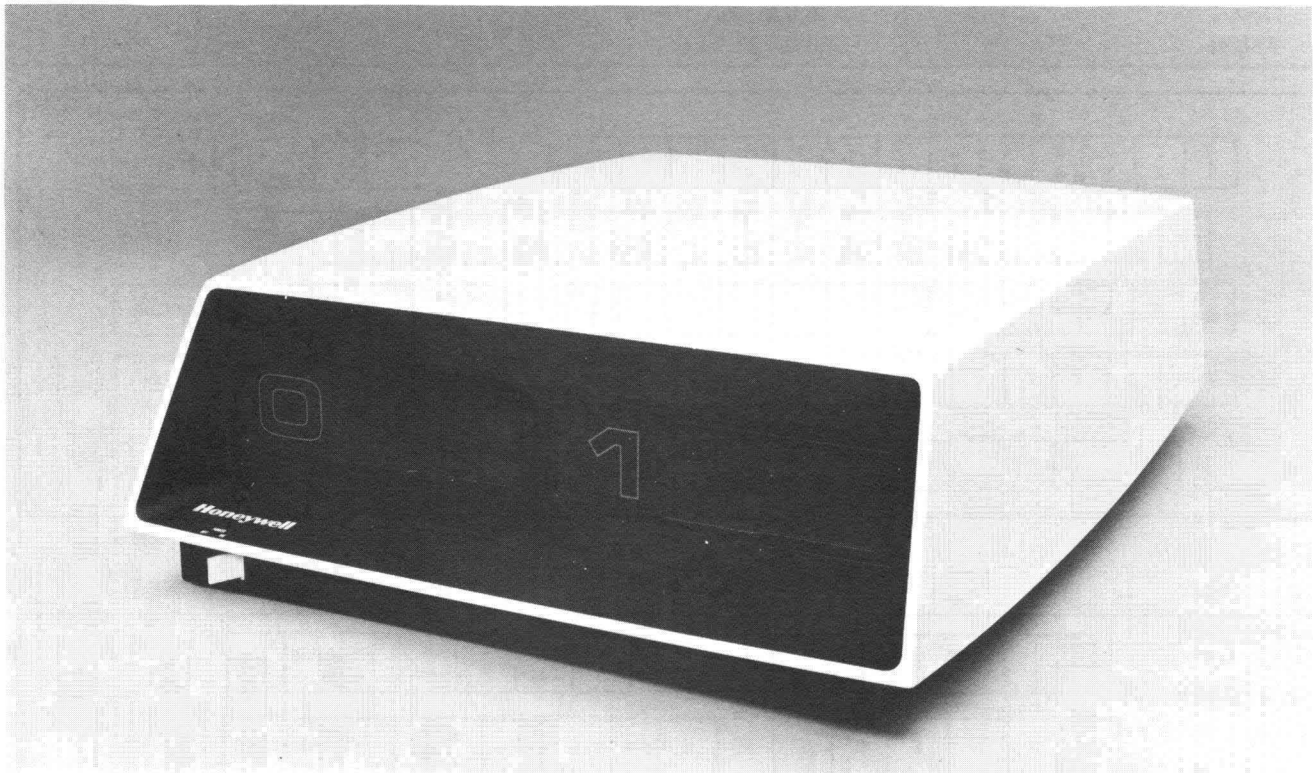


Figure 6-13. Level 6 Diskette Unit

TABLE 6-16. DISKETTE SPECIFICATIONS

Characteristics / Device	Diskettes (DIU) 9101	9102
Spindles	Single	Dual
Data Capacity Bytes/disk (formatted 77 tracks)	256,256	

TABLE 6-16 (CONT). DISKETTE SPECIFICATIONS

Characteristics \ Device	Diskettes (DIU) 9101 9102
Bytes/track (formatted)	3328
Seek Time (ms)	
Minimum	20
Average	260
Maximum	770
Average Latency (ms)	83.33
Transfer Rate	
Bits/second	249,984
Bytes/second	31,248
Words/second	15,624
Diskette Speed (rpm)	360
Tracks per Surface	77
Recording Density (bpi)	3200
Device Interface	One Diskette Device-Pac is required for each DIU9102 (Dual Diskette) or for up to two DIU9101s (Single Diskettes)
Medium	Honeywell type M4101 (or equivalent)
Physical Dimensions	
Height	10.5 in. (26.7 cm)
Width	17.5 in. (44.5 cm)
Depth	22 in. (55.9 cm)
Weight	50.8 lb (23 kg) 61.7 lb (28 kg)

Features

- o Available in configuration of 1–4 diskettes (256KB-1024KB) per MDC
- o Use of flexible disk
  - – Inexpensive media
  - – Simple to load and handle
  - – Easy and convenient to transport
  - – Minimum storage space required
- o Minimum operator intervention necessary
- o Data integrity achieved through the use of error detection code

Operation

While waiting for the diskette to become ready, the CP is free to perform other operations. A program interrupt signals the central processor when further device servicing is required by software.

The recording medium is composed of a single flexible disk of mylar which has a magnetic-oxide-coated recording surface and is packaged in a protective envelope which is never removed. The total disk package is only 8 inches square (Figure 6-14).

When loading the disk, the operator lifts the diskette cover and places the envelope/disk package in position. When the cover is closed, the diskette spindle automatically engages the disk and the diskette is ready for operation.

Associated with each device (spindle) is a set of registers which are loaded by software and specify the parameters required for disk operation. In addition to range and address registers, there are two configuration registers which contain record location and identification information and a task register which contains command codes. To perform a specific operation, software first loads the address, range, and appropriate configuration registers (if any). The task register is loaded last and specifies the operation to be performed. The controller begins command execution when it receives the task word.

Commands addressed to a nonbusy diskette device will always be accepted, but execution may be delayed if a data transfer is being performed on another diskette. All commands addressed to a busy diskette device will be rejected (NAK response on bus) except from an Output Control Word.

Bits 0–7 of configuration register B are treated as a sector number. This field is automatically updated by the controller during Read and Write Data commands (see "Read Data" and "Write Data"). The sector number field of con-

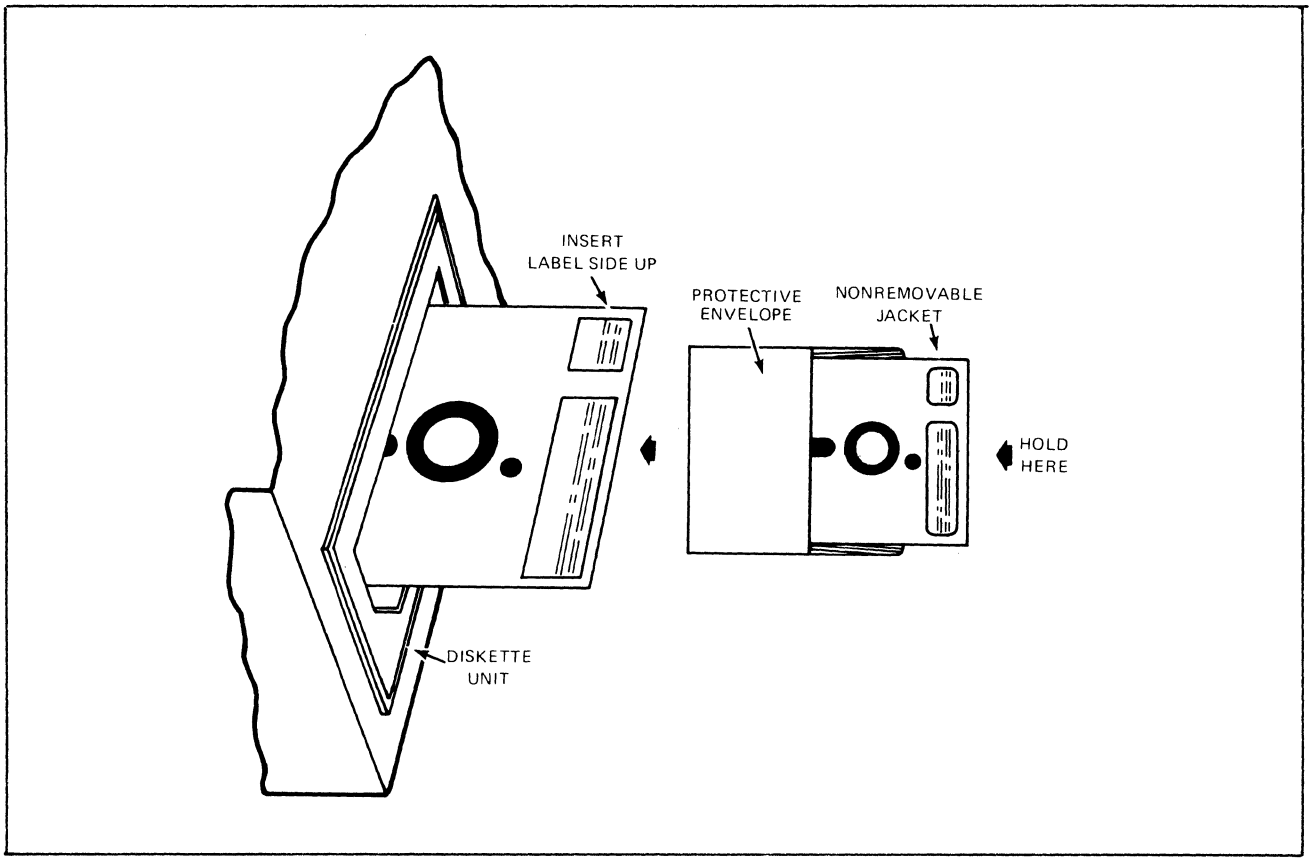


Figure 6-14. Diskette Media Handling

figuration register B is incremented by one at the end of each error free data field and points to the next sector on the track when a Read or Write operation is completed. This feature makes it unnecessary for software to update the configuration register when doing a series of sequential Read or Write Data Commands on the same track. Note that during an extended update Read or Write, records are processed in numerical order (not necessarily the order in which they are physically recorded). An extended operation can, therefore, be used to advantage on files that are recorded using a Sector Identifier (ID) interleaving technique as well as on files that are recorded in numerical order.

If, during a read data operation, a read error is encountered in a data field, the sector number field of configuration register B is not incremented. The command is terminated in this case and the sector number field references the sector in which the error was encountered.

Data Format

Each track on the diskette contains up to 26 equal length sectors of 128 bytes each. There are 77 tracks numbered from 0–76 yielding a total formatted capacity of 256,256 bytes. The data encoding scheme is double frequency recording and each field is preceded by an address mark and followed by a two-byte Error Detector Code (EDC).

The Sector ID fields are software programmable and do not have to be numerically sequential. Note that the third byte of the Sector ID fields are treated by the diskette subsystem as a sector number.

When a track is formatted, up to 26 sectors are written beginning at the index. The sector headers are specified by software and are extracted from memory during the format operation. This allows software to specify any sequence of sector IDs it may require to implement various interleaving schemes. Data is read from memory to format up to 26 sector IDs (see "Format Write"). Data fields will be zero-filled during formatting.

Records may be deleted from any point on a track with the Write Deleted Data command. This operation causes the data field of the addressed record to be updated in the normal way except that a special "deleted data field" address mark is used to identify the field as having been deleted. This record is automatically passed over in subsequent Read Data commands.

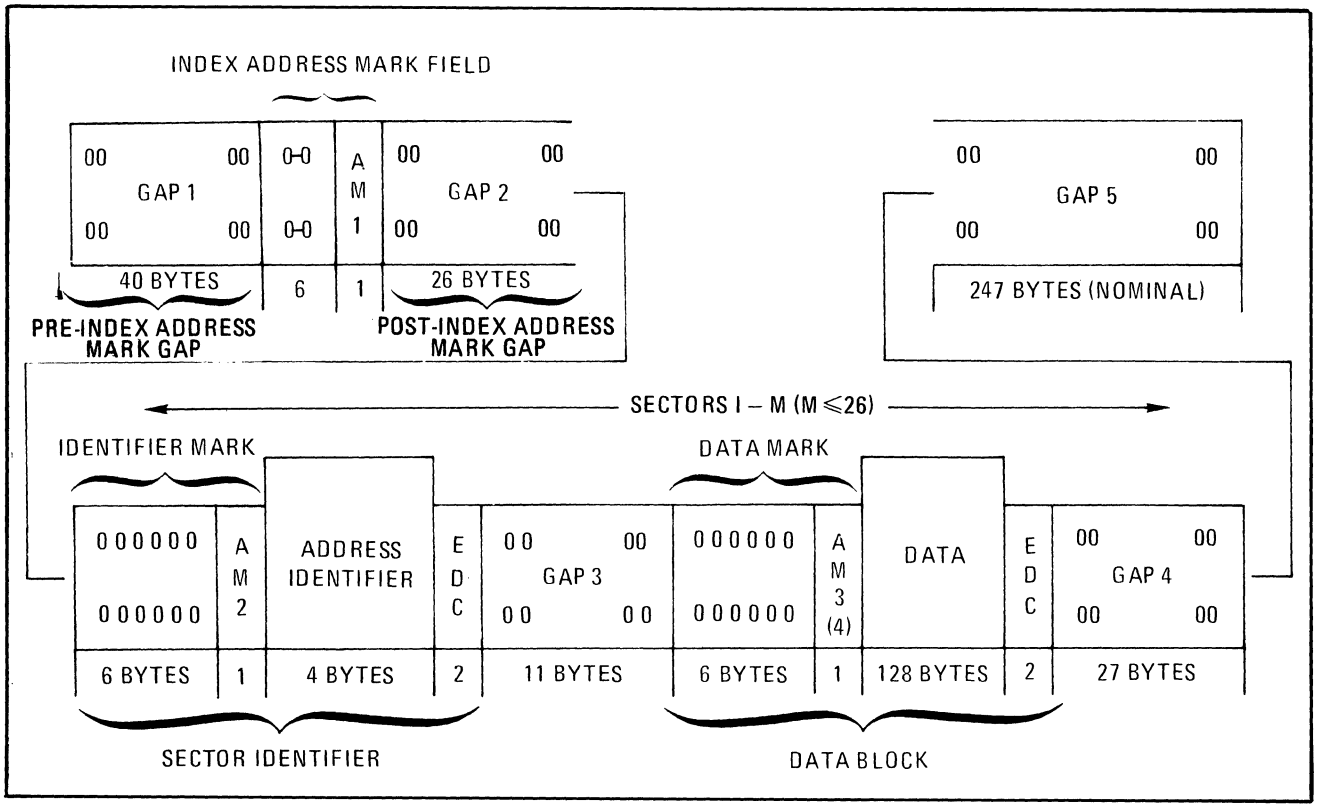


Figure 6-15. Diskette Track Format

Track Format

The diskette track format is illustrated in Figure 6-15.

Pre-Index Address Mark Gap (GAP 1)

This field begins at the leading edge of the Index Pulse and is composed of 40 hexadecimal 00 or FF bytes. This field may, in some cases, be 47 bytes long.

Index Address Mark Field

This is an optional field which, when included, is composed of 6 hexadecimal 00 bytes and one Address Mark byte (AM1). The Index Address Mark byte is illustrated in Figure 6-15. If this field is not included, the corresponding area on the track becomes part of the Pre-Index Address Mark Gap. In this case the Pre-Index Address Mark Gap is 47 bytes long.

Post-Index Address Mark Gap (GAP 2)

This field begins with the first byte after the Index Address Mark field (or is an extension of the Pre-Index Address Mark Gap if the Index Address Mark is omitted) and is composed of 26 hexadecimal 00 or FF bytes.

Sector Identifier

The Sector Identifier field is illustrated in Figure 6-16.

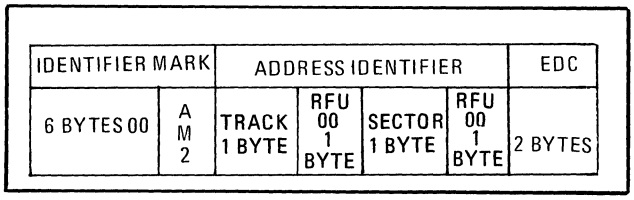


Figure 6-16. Sector Identifier

Identifier Mark

This field is composed of 6 hexadecimal 00 bytes and a Sector Identifier Address Mark (AM2) as shown in Figure 6-16. Any noise areas in a preceding gap caused by write current turn-on or turn-off transients must not extend into this field.

Address Identifier

The first byte of the Address Identifier field (Track Number) is a binary number which represents the logical track number starting with hexadecimal 00 for the outermost track and increasing toward the center of the diskette. For some media, defective tracks may be skipped and the track address numbering continued sequentially with the next good track.

The second and fourth bytes of the Address Identifier field are reserved for future use (must be set to hexadecimal 00 by software).

The Sector Number byte (third byte of the Address Identifier field) is the binary represen-

tation of the sector number. This can be of any value from zero to 255 (hexadecimal FF). The order in which sectors are numbered or recorded on the diskette is not restricted by the hardware.

The last two bytes of the Address Identifier field are EDC bytes. These bytes are hardware generated by shifting serially the bits of the associated field through a 16-bit shift register (initialized to all one bits).

The last two bytes of the Address Identifier field are generated using the bytes of the Sector Identifier starting with the Sector Identifier Address Mark (AM2) and ending with the fourth byte of the Address Identifier field.

Identifier to Data Gap (GAP 3)

This field begins with the first byte after the Sector Identifier EDC bytes and is composed of 11 hexadecimal 00 or FF bytes. These bytes may become ill-defined due to the overwriting process during data field updates.

Data Block

The Data Block field is illustrated in Figure 6-17.

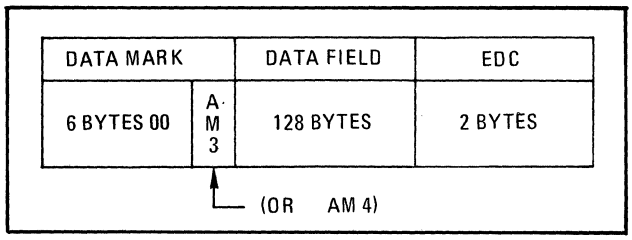


Figure 6-17. Data Block

Data Mark

This field is composed of 6 hexadecimal 00 bytes and an Address Mark byte. There are two types of Address Marks that can occur in front of a data field. The "Data Field Address Mark" (AM3) is for normal data fields. The "Deleted Data Field Address Mark" (AM4) indicates that the data field has been deleted. The format of these two Address Marks is illustrated in Figure 6-15.

Data Field

Data fields are always 128 bytes long. Any write operation that does not write a complete data field results in the rest of the field being filled with hexadecimal 00 bytes. The Format Write operation causes all the data fields on a particular track to be filled with 128 hexadecimal 00 bytes.

The last two bytes of the Data Block are EDC bytes. These bytes are generated using the bytes of the Data Block starting with the Address Mark byte (AM3 or AM4) and ending with the 128th byte of the data field.

Data Block Gap (GAP 4)

This field is composed of 27 hexadecimal 00 or FF bytes. The Data Block Gap is recorded after each Data Block and precedes the following Sector Identifier. After the last Data Block on the Track, this field precedes the Track Gap.

Track Gap (GAP 5)

This field follows the last Data Block Gap on the track and contains hexadecimal 00 or FF bytes until the Index is detected. The length of this gap is dependent on the speed of the particular drive on which the medium is mounted when it is formatted or updated (nominally 247 bytes).

Defective Track Handling

Although new diskette media are shipped free of defective areas, spots on which records cannot be successfully written can develop. If disk errors begin to occur, a decision has to be made regarding replacement of individual diskettes. If diskettes are physically damaged (torn, folded, creased, etc.) or if the recording surface becomes contaminated with a foreign material, the diskette must be replaced. If, however, a bad spot develops due to excessive wear on a particular track, then it may be desirable to flag the affected area as defective in order to keep the diskette in use.

On formatted diskettes, software accounts for defective sectors (or tracks) by allocating space on the diskette around the bad area. For example, if sector 2 of track 4 on a particular diskette is bad, no files will be allocated space in that sector.

Instructions

Table 6-17 lists the I/O commands to which the MDC/Device-Pac/diskettes respond. A detailed description of each command follows this table.

TABLE 6-17. DISKETTE COMMANDS

Type	Function Code	Command
Output	09[a]	Output Address
	0D	Output Range
	11	Output Configuration Word A
	13	Output Configuration Word B
	03	Output Interrupt Control
	07	Output Task Word
	01	Output Control Word

TABLE 6-17 (CONT). DISKETTE COMMANS

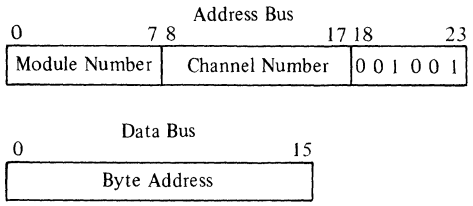
Type	Function Code	Command
Input	0C	Input Range
	10	Input Configuration Word A
	12	Input Configuration Word B
	02	Input Interrupt Control
	26	Input Device ID (2010)
	06	Input Task Word
	18	Input Status Word

[a]Function Code 09 as executed by the CP results in execution of functions 09 and 0D.

Output Commands

Output Address (09)

Format:

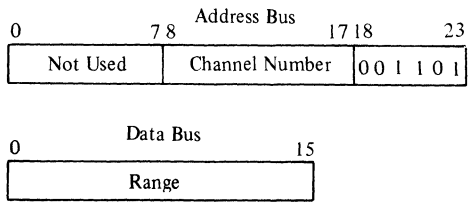


Function:

Loads a 24-bit address into the address register associated with the referenced channel (device). The address refers to the starting (byte) location in main memory where the controller begins input or output data transfers. Bits 0–7 of the address bus (Module Number) are the most significant bits of the address. The data bus contains the 16 least significant bits. Data transfers to or from memory are normally on a word basis but byte mode transfers can occur associated with the first and/or last memory cycle of a particular data transfer if the main memory buffer (identified by this instruction) begins or ends on an odd byte boundary.

Output Range (0D)

Format:

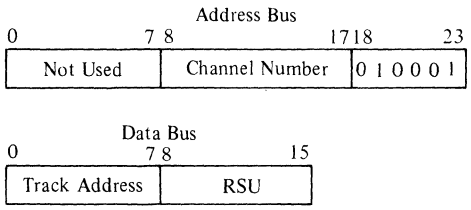


Function:

Loads the Range register associated with the referenced channel. The (16-bit) quantity loaded (data bus) is the number of bytes to be transferred during the data transfer that is being set up. The number is a positive binary quantity (bit 0 must be zero) and is decremented by the controller after each memory transfer. A range of zero results in a premature End-of-Operation for any read or write command that may be subsequently issued (see Output Task Word).

Output Configuration Word A (11)

Format:

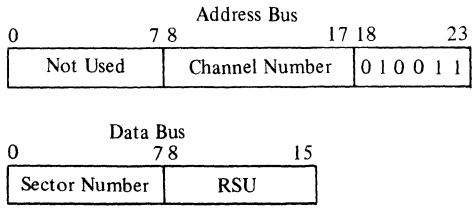


Function:

Loads Configuration Word A for the device corresponding to the referenced channel. The Track Address (bits 0–7) is used as the seek argument during seek operations. The complete word is used as the two high order bytes of a Sector ID field to be searched for during an update Read or Write operation. Bits 8–15 are reserved for software use (RSU).

Output Configuration Word B (13)

Format:



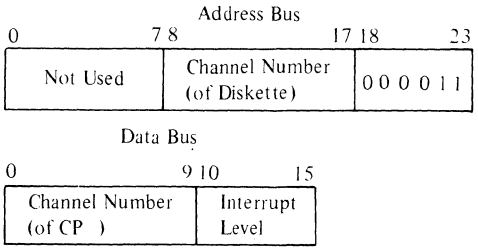
Function:

Loads Configuration Word B for the device corresponding to the referenced channel. This word is used as the two low order bytes of a Sector ID field to be searched for during an update Read or Write operation. The subsystem will treat bits 0–7 of Configuration Word B as a sector number. This number will be incremented after operating on a data field during an update

Read or Write operation. Bits 8–15 are reserved for software use (RSU).

Output Interrupt Control (03)

Format:

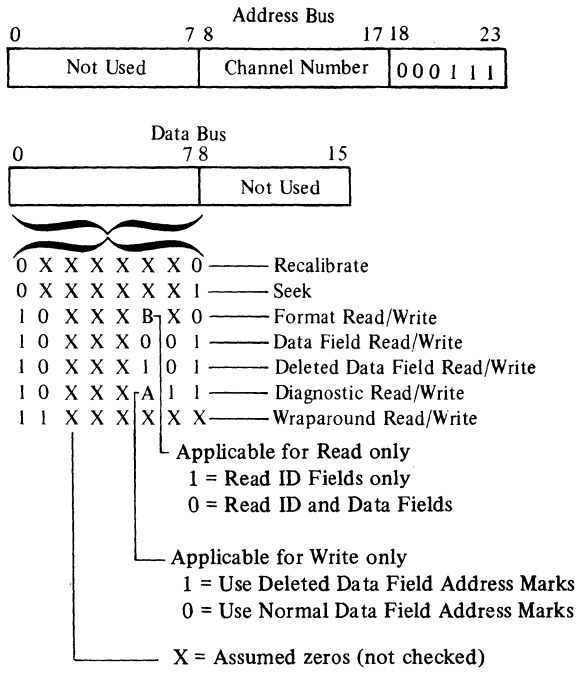


Function:

Loads, for the referenced device, the interrupt level and the channel number of the CP to which subsequent interrupts should be sent. The level number is a 6-bit quantity and is positioned on the data bus as illustrated above. Bits 0–9 of the data bus contain the channel number of the CP to which the interrupt is sent. If an Interrupt Level of zero is loaded, the subsystem does not generate or save interrupts for any events that occur while the Interrupt Level is zero. The Interrupt Level is set to zero whenever the subsystem is initialized.

Output Task Word (07)

Format:



Function:

Outputs a Task Word to the referenced channel. The coding of bits 0–7, illustrated above, represent the operations that are to be performed. When this instruction is accepted, the channel enters the Busy state and execution of the indicated task is immediately begun. All address, range, and configuration information must be loaded prior to execution of this instruction. The direction of data transfer is indicated on the low order bit of the channel number used in the most recent output address instruction.

The individual bits will cause the following to occur:

- Recalibrate: The Recalibrate command causes the channel to move the device's positioner to track zero and reset the Device Fault line on the device interface (via the Fault Reset line). This instruction is intended as an Initialization command to guarantee that the positioner location information in the controller is correct and that device faults are cleared.
- Seek: The Seek command in the Task Word causes the channel to move the device's positioner to the track indicated in Configuration Word A. If the device indicates that it is on track zero when not expected, a cylinder number greater than 76 is specified in Configuration Word A. Or, if the device cannot be positioned at track zero during a Recalibrate command (as indicated by the Track Zero line), then the seek Error Bit is set in the Status Word.
- Format Read: The Format Read command causes the channel to read all Identifier (ID) fields (for Task Word bit 5=1) or all Identifier and Data Fields (for Task Word bit 5=0, including deleted data fields) on a track beginning with the first sector after index and in the order in which they are recorded. Data will be transferred to memory beginning at the memory location specified in the subsystem's memory address register. This address will be the address loaded by the most recent Output Address instruction if no data transfer has occurred since that instruction was executed. If one or more data transfer operations have been executed since the last Output Address instruction, the starting memory address used for this operation will be the byte address immediately fol-

lowing the end of the most recent data transfer executed for this device (either read or write).

Data is transferred until a read error occurs, the range is satisfied, or the entire track is read (index is detected).

Normal range for this command (to read one complete track) is

$$R = (4 + 128) \times 26 = 3432 \text{ bytes}$$

where:
4 = ID
128 = Data
26 = Sectors

If this command is terminated due to end of track before the range is satisfied, the residual range is available via the Input Range command.

A read error in any field transferred to memory causes the operation to be terminated with the Read Error bit set in the Status Word. The Sector ID field in Configuration Word B will point to the record in error. The field in error can be determined through examination of the residual range (if a read error is detected in an ID field the range will have been decremented for the ID field only).

If the required transfer rate is not maintained on the bus (15.6K words per second), the operation is terminated and the Overrun/Underrun bit is set in the Status Word (bit 2).

If the range register is zero when this command is received, the Task is immediately terminated (End-of-Operation). No data is read or transferred.

- Format Write: The Format Write command causes the channel to format the track which is positioned under the Read/Write head when this command is received. Up to 26 equal length sectors are written starting at index. The Sector ID fields are read from memory beginning with the memory location specified in the subsystem's memory address register.

  Data fields are written with normal data field Address Marks and are zero-filled.

  The range to format one complete track is

  $$R = 4 \times 26 = 104 \text{ bytes}$$

  If a range less than 104 is sent for a format write, the track is zero-filled (may result in fewer than 26 sectors). If the range expires after the first word of a Sector ID field, then the second word of that ID is unspecified. If a range greater than 104 is specified, a partial sector might be recorded (a sector can be started in the Track Gap but is not completed if the Index Pulse is detected before the end of the data field). If the range register is zero when this command is received, the task is immediately terminated (End-of-Operation). No data is written.

  If the Sector ID data cannot be read from memory at a sufficient rate, then the operation is terminated and the Overrun/Underrun bit is set in the Status Word (bit 2).

- Read Data: The Read Data command causes the channel to locate the sector defined by the Sector ID image loaded in Configuration Words A and B and to transfer the data field of at least that sector to main memory. Data is transferred to memory beginning with the memory location specified in the subsystem's memory address register and continues until the range is satisfied. When the transfer of the first specified sector field is completed (without error), the Sector Number field of Configuration Word B is incremented by one (module 256). If the initial range is greater than 128, then the sector represented by the updated contents of Configuration Words A and B (A is unchanged) is located and data transfer continues until either the range is satisfied, a read error occurs, or the sector specified by Configuration Words A and B cannot be located on the track (as indicated by the detection of two index marks without a successful compare). If the specified sector cannot be located, then an unsuccessful search is posted in the Status Word (bit 7).

  If a deleted Data Address Mark is encountered during a Read Data operation, that field is spaced over and the Read continued on the numerically next sector (Deleted Field bit set in the Status Word – bit 3).

  If a read error is encountered in a data field, the operation is terminated and the Read Error bit in the status word is set (bit 4). The Sector Number field of Configuration Word B contains the address of the record in error. If a read error is encountered in an ID field, a miscompare result is assumed and the search continues.

The Read Error bit is *not* posted.

If this command is terminated before the range is satisfied, the residual range is available via the Input Range command. If the range register is zero when this command is received, the task is immediately terminated (End-of-Operation). No data is read or transferred.

If the required transfer rate is not maintained on the bus (15.6K words per second) then the operation is terminated and the Overrun/Underrun bit is set in the Status Word (bit 2).

- Write Data: The Write Data command causes the channel to locate the sector defined by the Sector ID image loaded in Configuration Words A and B and to rewrite the data field of at least that sector. The data is read from memory beginning with the memory location specified in the subsystem's memory address register. Rewritten data fields are preceded by normal data field address marks (see Figure 6-15).

  When the transfer of the specified sector is completed, the Sector Number field of Configuration Word B is incremented. If the range is less than 128, the data field is zero-filled. If the range is greater than 128, the sector represented by the updated contents of Configuration Words A and B (A is unchanged) is located and the data field rewritten (preceded by a normal data field Address Mark). This operation continues until either the range is satisfied or the sector specified by Configuration Words A and B cannot be located on the track (as indicated by the detection of two index marks without a successful ID field compare). If the latter event occurs, Unsuccessful Search is posted in the Status Word (bit 7).

  If two or more sectors that are to be updated with a single Write Data command are physically adjacent, they are written consecutively without loss of a revolution. If a read error is encountered in an ID field, a miscompare result is assumed and the search continues. In this case the Read Error bit is *not* posted.

  If this command is terminated before the range is satisfied, the residual range is available via the Input Range command. If the range register is zero when this command is received, the Task is immediately terminated (End-of-Operation). No data is written.

  If the required transfer rate is not maintained on the bus (15.6K words per second), the operation is terminated and the Overrun/Underrun bit is set in the Status Word (bit 2).

- Deleted Data Read/Write: The Deleted Data encoding of the Task Word (1XXXX101) when received on a write channel (low order bit of the channel number equals 1) causes the channel to perform as if the Write Data command was specified except that each data field updated is preceded by a Deleted Data Field Address Mark. Note that the data written in the deleted data fields is read from memory.

  If this Task Word encoding is received on a read channel, the channel performs as if the Read Data command was specified except that only data fields that are preceded by Deleted Data Field Address Marks are read.

- Write Diagnostic: The Write Diagnostic command causes the channel to perform as if the Write Data command was specified except that invalid EDC characters are written at the end of each data field updated (the EDC characters are always zero).

  Note that the data written in the data fields are read from memory and the Address Marks are as specified by bit 5 of the Task Word.

- Read Diagnostic: The Read Diagnostic command causes the channel to read everything on a track (including gaps, address marks, ID, data fields, and EDC bytes) beginning with (and including) the address mark preceding the first Sector ID on the track and ending at index pulse. Meaningful operation of this command can only be guaranteed on tracks which have not been updated since the last Format operation. No EDC checks are done. Data is transferred until the range is satisfied or the entire track is read (index is detected). Normal range for this command (to read one complete track) is

  $R = (7 + 17 + 131 + 33) \times 26 + 247 =$
  5135 bytes

where
7 = Sector ID and Address Mark
17 = Gap
131 = Data Field plus Address Mark
33 = Gap
26 = Sectors
247 = Track Gap

If this command is terminated due to end of track before the range is satisfied, the residual range is available via the Input Range command.

If the required transfer rate is not maintained on the bus (15.6K words per second), the operation is terminated and the Overrun/Underrun bit is set in the Status Word (bit 2).

- Wraparound Read/Write: During a Wraparound Write command, the channel reads one byte from memory (at the address specified in the subsystem's memory address register), shifts the byte through the diskette adapter write logic with an address mark clock bit pattern inserted, and saves the resulting bit pattern (16 bits) in the adapter for use during a subsequent Wraparound Read command. The saved bit pattern is recognizable as an Address Mark by the adapter read recovery and address mark detection logic. The byte supplied by software for the Wraparound Write command must have one bit in the two high order bit positions.

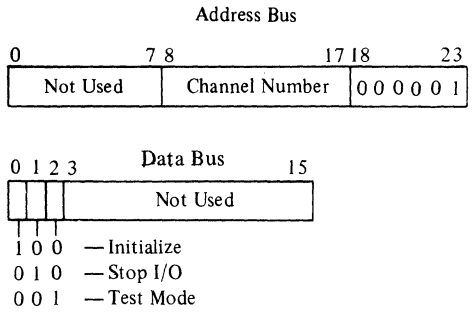
  When a Wraparound Read command is received (immediately following a Wraparound Write), the saved bit pattern from the preceding Wraparound Write command is shifted into the adapter read recovery and address mark detection logic as if it were a bit pattern from the medium. If no errors occur, an Address Mark is detected and the byte of data (with clock bits removed) is returned to main memory at the address specified in the subsystem's memory address register. The byte returned during this operation should be the same as the byte supplied by software in the preceding Wraparound Write command.

  If, during the Wraparound Read operation, an Address Mark is not detected by the adapter, the operation is not terminated. In this case, the adapter continually shifts zeros into the read recovery logic until stopped with a Stop I/O command or an Initialize (Output Control Word or Master Clear).

  A range of one should always be used for these commands. If a zero range is specified, the command is immediately terminated (without being executed). Ranges greater than one for these commands result in different subsystem responses depending on the starting address and data transfer direction and should, therefore, not be used.

Output Control Word (01)

Format:



Function:

Loads a Control Word into the referenced channel. This command is unconditionally accepted by the channel regardless of its Busy status. The individual bits cause the following specific actions to occur:

- Initialize: This command causes the MDC to reset to the same state that it enters after power-up. When an Initialize command is received by the MDC, all of its channels are initialized (regardless of which channel the command was received over). A Recalibrate command is issued for each diskette during execution of this instruction so that the MDC is in a state capable of proper control of diskette devices. Any operations that are in progress in the MDC at the time of the Initialization are abruptly terminated and all registers are initialized. No information about the terminated operations is retained and no interrupts for the operations are generated. The interrupt levels for all channels are set to zero (interrupts blocked).

- Stop I/O: This command causes any operation currently active on the specified channel to be abruptly terminated. If a Data Transfer operation is in progress, it is not completed nor is any error checking done. If a diskette positioner is in motion

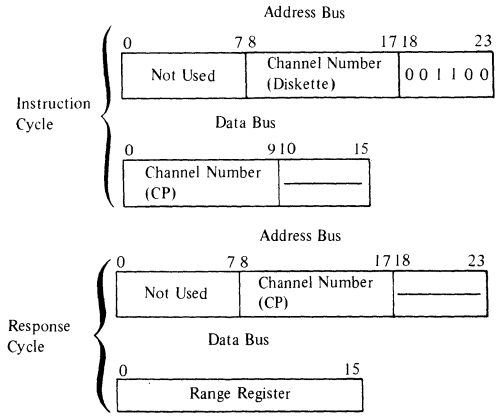
when this command is received, positioner orientation information is lost. An interrupt is generated for the operation terminated by this command as if the operation had come to a normal ending point. Status, Address, and Range information, present in the MDC when this command is received, is retained.

- o Test Mode: Refer to "Test Mode" earlier in this section.

Input Commands

Input Range (0C)

Format:

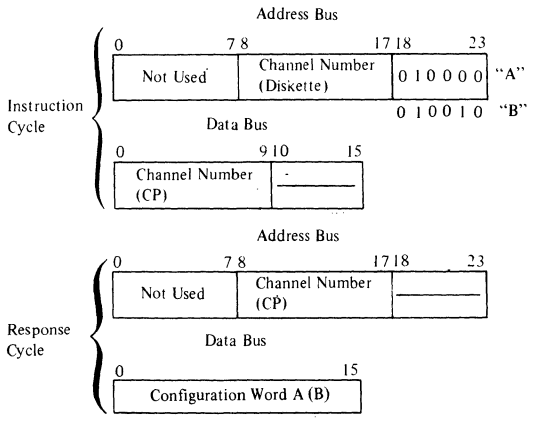


Function:

Causes the contents of the referenced channel's Range register to be transferred to the requesting channel. During the Response cycle (Second Half Read), the MDC returns in bits 8–23 of the address bus the same data that was received in bits 0–15 of the data bus during the Instruction cycle.

Input Configuration A (10)

Format:



Function:

Causes the channel's Configuration Word A (B) to be transferred to the requesting channel. During the Response cycle (Second Half Read), the MDC returns in bits 8–23 of the address bus the same data that was received in bits 0–15 of the data bus during the Instruction cycle.

Input Configuration Word B (12)

Format:

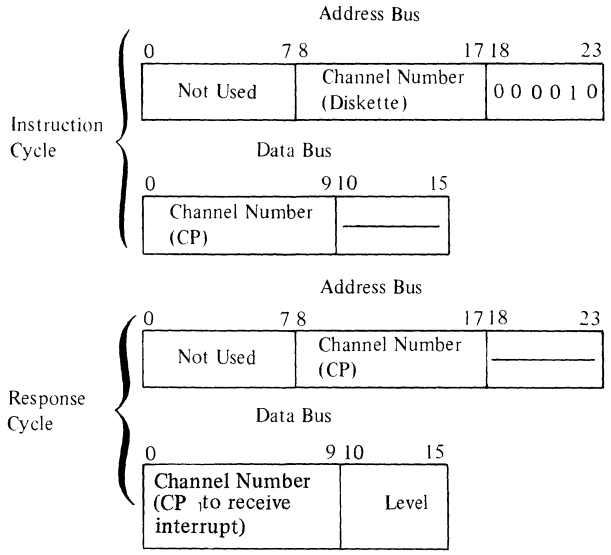
See "Input Configuration Word A"

Function:

See "Input Configuration Word A"

Input Interrupt Control (02)

Format:



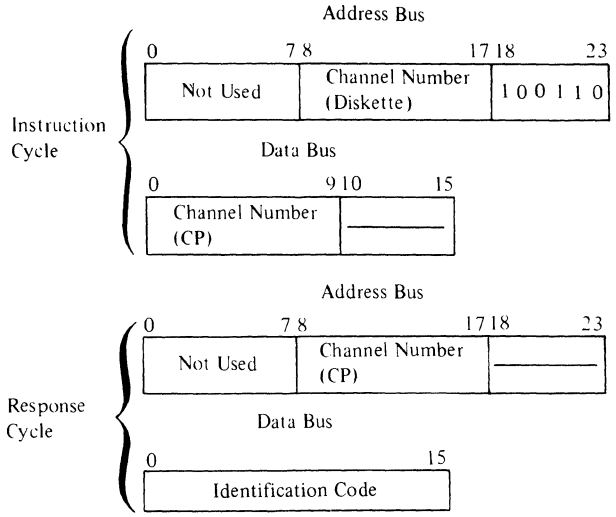
Function:

Causes the channel's Interrupt level to be transferred to the requesting channel. The level value is placed on data bus bits 10–15 (see above) with bit 15 as the least significant bit. This quantity is the value previously received in an Output Interrupt Control instruction or a default value of 00. The default value is the Interrupt level assumed by the channel when initialized. Note that the channel number returned in bits 0–9 of the data bus might be different from the channel number of the CP executing this instruction if more than one CP is attached to the bus.

During the Response cycle (Second Half Read), the MDC returns in bits 8–23 of the address bus the same data that was received in bits 0–15 of the data bus during the Instruction cycle.

Input Device ID (26)

Format:

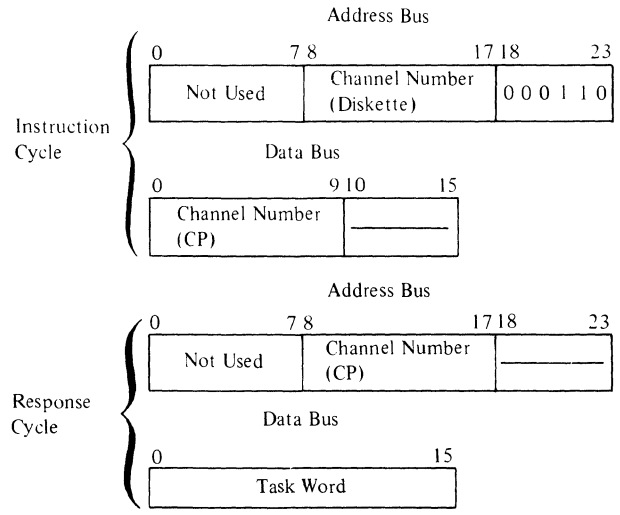


Function:

Causes the referenced channel to transfer its identification code to the requesting channel. The code for this type of diskette is 2010_{10}. During the Response cycle (Second Half Read), the MDC returns in bits 8–23 of the address bus the same data that was received in bits 0–15 of the data bus during the Instruction cycle.

Input Task Word (06)

Format:

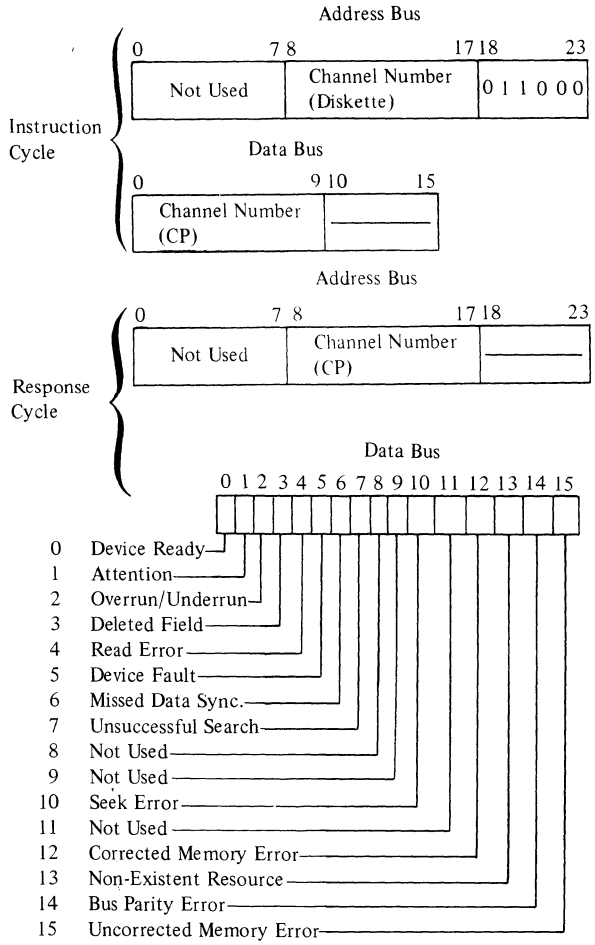


Function:

Causes the Task Word of the referenced channel to be transferred to the requesting channel. The Task Word transferred contains the code for the last operation executed by the channel (unless an Initialize has occurred). During the Response cycle (Second Half Read), the MDC returns in bits 8–23 of the address bus the same data that was received in bits 0–15 of the data bus during the Instruction cycle.

Input Status Word (18)

Format:



Function:

Causes the referenced channel's Status Word to be transferred to the requesting channel. During the Response cycle (Second Half Read), the MDC returns in bits 8–23 of the address bus the same data that was received in bits 0–15 of the data bus during the Instruction cycle. See also Table 6-18.

TABLE 6-18. STATUS BIT DEFINITIONS – DISKETTE

Status Condition	Bit	Definition	Reset by
Device Ready	0	The device is online with the medium loaded and no further manual intervention is required to place it under program control. Note that a change of state of this bit causes the Attention bit (bit 1) to be set resulting in an interrupt (if the interrupt level is nonzero).	A change in condition
Attention	1	Set whenever the Device Ready bit (bit 0 of the status word) changes state. Indicates to software any change of operational status of the device (e.g., load/unload of media). Whenever set, an interrupt is attempted (if the Interrupt level is nonzero). If a previously initiated operation is in progress when a device state change is sensed, the resultant interrupt (with the Attention bit set) serves as notification of both the end of the operation and the device state change.	Input Status Word or Output Task Word[a]
Overrun/ Underrun	2	Set during a Read or Write operation when the data transfer to/from main memory cannot be maintained at a high enough rate (15.6K words per second) during field transfers in word mode). Either data was lost on input because of failure to keep up with device demands or data was unavailable on output when required by the device.	Output Task Word[a]
Deleted Field	3	Set if a Deleted Data Field Address Mark is encountered during a Format Read command or if a data field which would normally be read during a Read Data command is skipped because of a Deleted Data Field Address Mark. Also set if the Deleted Data Field encoding of the Task Word (1XXXX101) is received on a Read channel and a normal data field is skipped during the resulting read operation. Posting of this indication does not cause the operation in progress to be terminated.	Output Task Word[a]
Read Error	4	Set during any Read operation if the EDC Word at the end of a field disagrees with the EDC Word calculated while reading the field.	Output Task Word[a]
Device Fault	5	Set whenever a Fault indication is received from the device.	Recalibrate[a]
Missed Data Sync	6	Set if, after a Sector ID has been detected during a Read operation, the corresponding data field is not detected or if, during a Format Read, two consecutive data fields (or two consecutive ID fields) Address Marks are detected (indicating that a field was missed).	Output Task Word[a]

TABLE 6-18 (CONT). STATUS BIT DEFINITIONS – DISKETTE

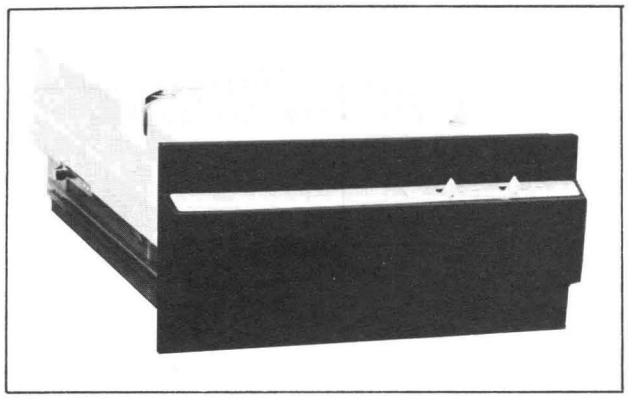
Status Condition	Bit	Definition	Reset by
Unsuccessful Search	7	Set during a non-format Read or Write operation for which the Sector ID specified in Configuration Words A and B cannot be located on the Track.	Output Task Word[a]
Seek Error	10	Set during a Seek operation if the device indicates that it is on track zero when not expected, if a cylinder number greater than 76 is specified in Configuration Word A, or if the device cannot be positioned at Track zero (as indicated by the Track Zero line) during a Recalibrate command.	Recalibrate[a]
Corrected Memory Error	12	During execution of the previous operation Main Memory detected and corrected a memory read error. The data that was delivered to the MDC was assumed to be correct.	Output Task Word[a]
Non-existent Resource	13	Set whenever the MDC attempts a Write or Read request Bus cycle and receives a NAK response.	Output Task Word or Input Status Word[a]
Bus Parity Error	14	Set whenever the MDC detects a parity error on either byte of the data bus during any output bus cycle (i.e., odd function code), during a Second Half Memory Read cycle or when a parity error is detected in bits 0–7 of the address bus during an Output Address command.	Input Status Word[a]
Uncorrected Memory Error	15	During execution of the previous operation, Main Memory detected a read error which the EDAC algorithm could not correct. The data that was delivered to the MDC was incorrect. Will not cause termination of the operation in progress (may result in bad data written on the medium).	Output Task Word[a]

[a]Initialize (output control word) and Master Clear on the Bus also reset these status bits.

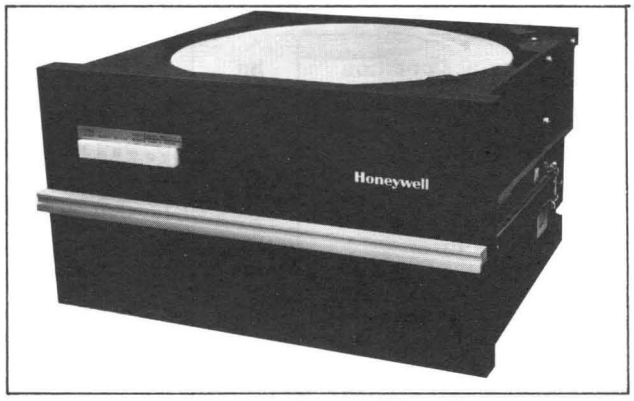
CARTRIDGE DISK UNITS

The CDU9101/9102/9103/9104 and CDU9114/9116 Cartridge Disk Units (Figure 6-18) are random access storage devices with a data storage capacity of up to 11.2 million bytes per unit. Users are offered a choice of low- or high-density units as well as removable only or fixed and removable disks mounted on the same spindle. Fixed-length sectors of 12/24 records of up to 576/256 bytes per track (or larger by linking two or more sectors) can be written. Table 6-19 lists their specifications.

The Cartridge Disk Units interface with the Level 6 processor by means of a Mass Storage Controller (MSC9101), a Cartridge Disk Device-Pac (CDM9102), and an 8- or 20-foot cable. The MSC9101 can connect and control up to four CDU9101/9102 or 9114 low density units or up to four CDU9103/9104 or CDU9116 high density units.



CDU9101/9102/9103/9104



CDU9114/9116

Figure 6-18. Level 6 Cartridge Disk Units

TABLE 6-19. CARTRIDGE DISK UNIT SPECIFICATIONS

Characteristics / Device	Cartridge Disk Units (CDU) 9101	9102 9114	9103	9104 9116
Capacity				
Byters per Sector	256/576	256/576	256/576	256/576
Sectors per Track	24/12	24/12	24/12	24/12
Bytes per Track	6144/6912	6144/6912	6144/6912	6144/6912
Tracks per Cylinder[a]	2	4	2	4
Bytes per Cylinder	12,288/13,824	24,576/27,648	12,288/13,824	24,576/27,648
Cylinders per Unit	204	204	408	408
Bytes per Unit	2.5/2.8M bytes	5.0/5.6M bytes	5.0/5.6M bytes	10.0/11.2M bytes
Units per Controller	4	4	4	4
Bytes per Controller	10.0/11.2M bytes	20.0/22.5M bytes	20.0/22.5M bytes	40.1/45.1M bytes
Disk(s)				
Number	1	2	1	2
Type	Removable	Fixed and Removable	Removable	Fixed and Removable
Density	Low	Low	High	High
Seek Time (ms)				
Same Cylinder[b]	← 0 →			
Track to track	← 9 →			
Average random	← 35 →			
Maximum (408 tracks)	← 65 →			
Average Latency (ms)	← 12.5 →			
Transfer Rate	2.5M bits per second; 312K bytes per second			
Simultaneity	During data transfer on one unit, simultaneous seek operations can be performed on all other units attached to the same controller.			
Device Interface	A single Device-Pac (CDM9101) interfaces up to four disk units.			
Medium	Honeywell type M4024 (or equivalent)			
Controller	MSC9101 controls up to four disk units of the same density.			
Physical Dimensions	*CDU9101/9102/9103/9104*		*CDU9114/9116*	
Height	8.75 in. (22.2 cm)		10.31 in. (26.2 cm)	
Width	17.74 in. (45.1 cm)	18.94 in. (48.10 cm)		
Depth	30 in. (76.2 cm)		30.63 in. (77.80 cm)	
Weight	84 lb (37.8 kg)		150 lb (67.5 kg)	

[a]Fixed and removable tracks on the same cylinder will not be linked by multiple sector data transfers – they are considered logically different media.

[b]Cartridge disk (fixed and removable) cannot be crossed without initiation of a seek operation.

Features

- o Combines transportability of magnetic tape with random access features of disk
- o Ideal for applications requiring fast access to a medium-sized data base; average random access to 10 million bytes is less than 50 milliseconds
- o Maximum total data storage capacity of 40.1/45.1 million bytes; Direct Memory Access transfers data at 156K words/second
- o 200 tpi recording allows 408 cylinders per spindle, each with a capacity of 12K bytes accessible with no head movement
- o Up to four disk units per controller; choice of all high or all low density disk drives
- o Record length can be varied to suit customer application via multiple sectors; all sectors include check words for data integrity
- o Each track can have a maximum of 12 or 24 individually addressable records; software interlace of record addresses minimizes rotational latency time
- o Single removable cartridge disk or removable and fixed cartridge disk units; disk-to-disk copying capability on the same unit

Operation

Depending upon the format selected, each track consists of 12/24 fixed length sectors that are clocked by fixed slots inscribed into the circumference of a disk. Records start at the beginning of a sector, with a maximum of 12/24 records of up to 576/256 bytes each per track. Larger records can be written linking two or more sectors. The maximum record length is 6912 (12 x 576)/6144 (24 x 256) bytes, linking all 12/24 sectors on one track. (See Figure 6-19.)

To seek a particular cylinder and platter, a word containing the platter and cylinder must be output to the controller. A second setup word is also output that contains the track and header (sector number) of the record to be written or read.

Each record consists of two parts:

Header – One 32-bit record identifier
Data – 256- or 576-byte sectors

Each part is followed by a check word that is automatically validated on read operations.

There are four modes of reading or writing:

- o write format
- o write record
- o read record
- o read track (headers or headers and data)

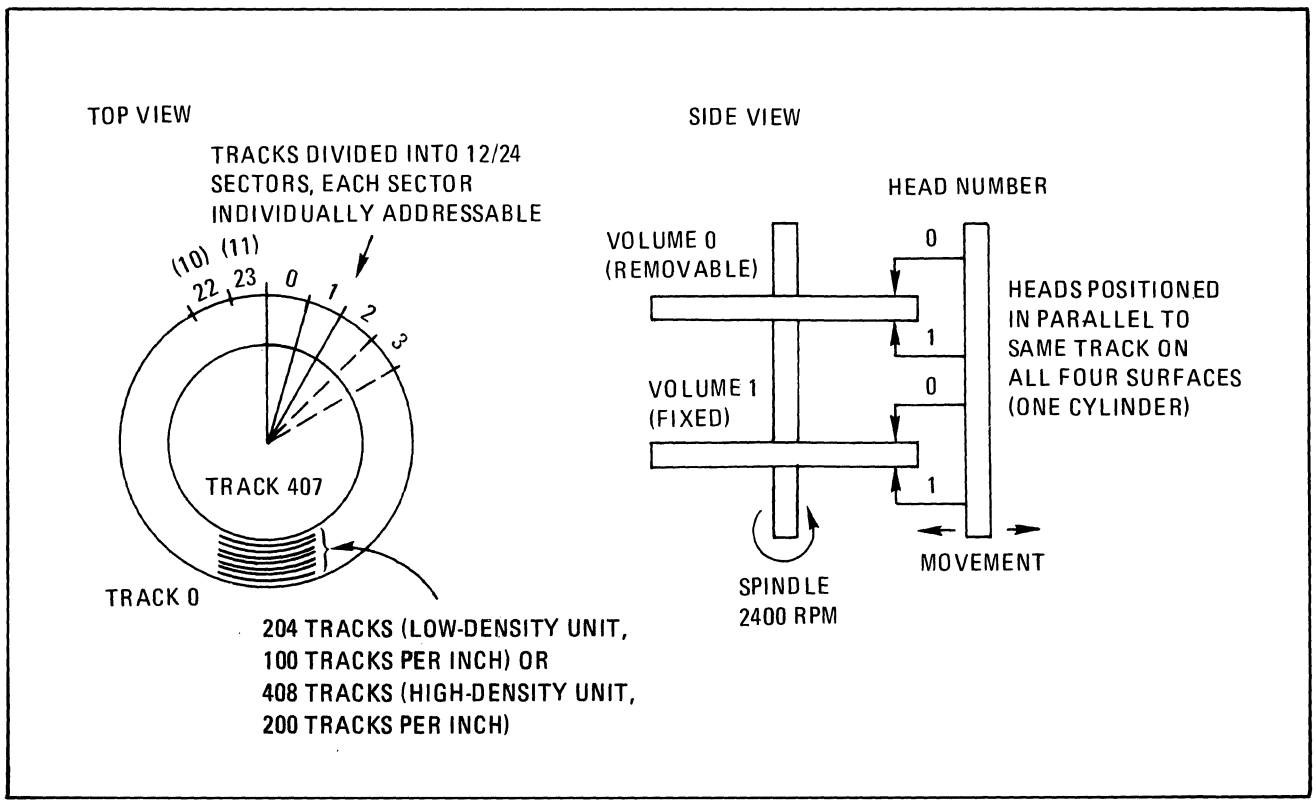


Figure 6-19. Physical Organization of Cartridge Disk

All reads and writes are under control of Direct Memory Access (DMA), with the address and range registers in the controller describing the area in memory that is being written from or read into.

The first operation that must be performed is a Write Format operation which writes 12 or 24 headers, one per sector, on a track, and zero-fills the data fields. A Write Record then writes a single record which may consist of multiple sectors after matching on the proper header. Writing is terminated either by DMA end-of-range or by an unsuccessful search. A Read operation again matches on a header and reads data until an end-of-range is reached. A Read-Track operation reads the header words and data starting with the first sector after an Index Mark (IM) in succession until an end-of-track or end-of-range is reached.

Data Format

Each track on the disk contains 12 or 24 equal length sectors of 256 or 576 bytes, respectively. There are 204 or 408 cylinders per surface and 2 or 4 tracks per cylinder depending on options. The total formatted capacity varies from 2,506,752 to 11,280,384 bytes, again depending on the device and format options used.

The data encoding scheme is double frequency recording and each field is preceded by a sync word and followed by an error detection code (EDC) word and postamble. The header fields are software programmable and do not have to be numerically sequential. Note that the fourth byte of the header fields will be treated by the controller as a sector number.

When a track is formatted, 12 or 24 sectors are written beginning at the sector following the index mark. The sector headers are specified by software and are extracted from memory during the format operation. The data field is zero-filled during formatting.

Track Format

The disk track format is shown in Figure 6-20.

Sector Gap (GAP 1)

The field begins at the leading edge of a Sector Pulse and is comprised of 18 bytes of zeros.

Identifier Sync Word (SWI)

This 2-byte field leading the identifier field consists of the hex characters FAAA.

Sector Identifier (Header)

This is a 4-byte field which is formatted as shown in Figure 6-20. Platter select (bit 4 of byte 1) represents the platter (fixed or removable). The cylinder number is a 9-bit binary number (right-justified in bytes 1 and 2) which represents the logical cylinder number, from 000 for the outermost track, to 203 or 407 for the innermost track. The track number (bit 7 of byte 3) represents the upper or lower surface of the selected platter. The sector number is a binary number (right-justified in byte 4) which represents the sector address.

Error Detection Code (EDC)

The two EDC bytes are hardware generated by use of a half-add algorithm in a 16-bit register.

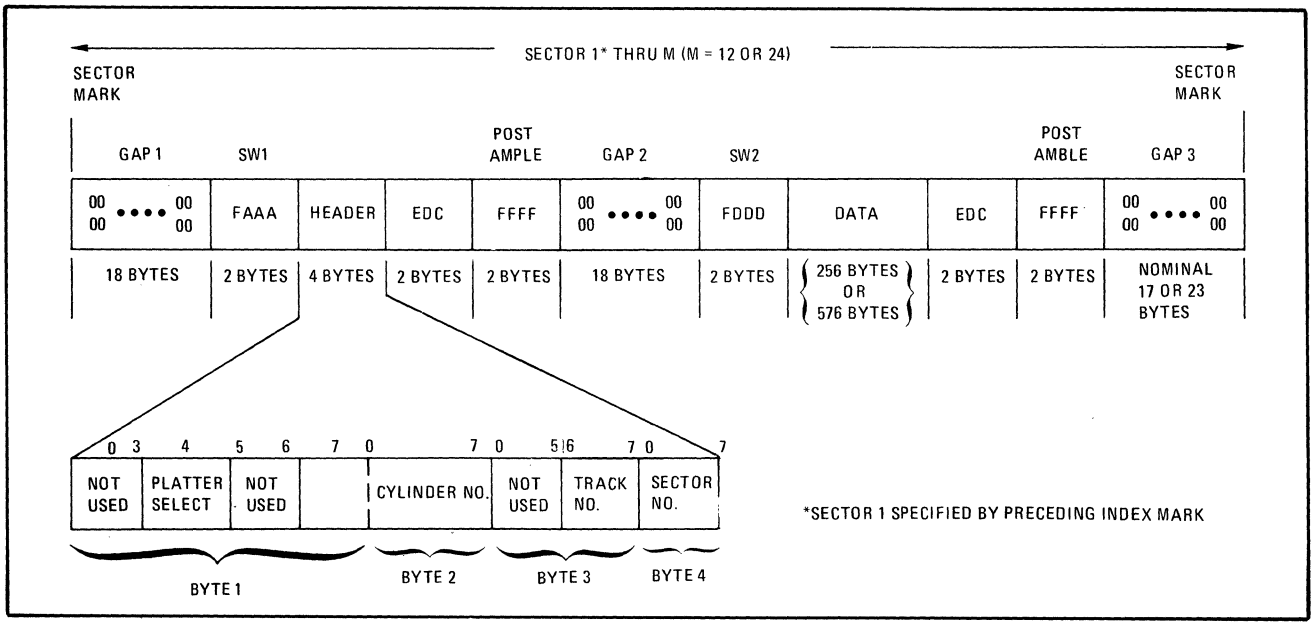


Figure 6-20. Disk Track Format

Data and ID bytes exclusively (no sync words, postamble, or gap bytes) are used for EDC generation.

This polynomial is used for both Sector ID fields and data fields.

Postamble

This 2-byte field follows the EDC bytes and consists of two bytes of all ones.

Identifier to Data Gap (GAP 2)

This field begins with the first byte after the Sector ID Postamble bytes and consists of 18 bytes of zeros. Write transitions may occur in this field as a result of an update write operation.

Data Field Sync Word (SW2)

This 2-byte field leading the data field consists of the hex characters FDDD.

Data Field

Data fields are always 256 or 576 bytes long. Any Write operation that does not write a complete data field results in the rest of the field being zero-filled. The Format Write operation causes all the data fields on a particular track to be zero-filled.

Data Field to Sector Mark Gap (GAP 3)

This field begins with the first byte after the Data Postamble bytes and consists of zeros to the next sector mark. The length of this gap is dependent on the speed of the particular drive on which the medium is mounted when it is formatted or updated (nominally 17 bytes for a 256-byte data field or 23 bytes for a 576-byte data field).

Defective Track Handling

New disk media may be shipped with a minimal number of defective areas. These defective areas, in conjunction with areas which may become defective during the life of the medium, are accounted for by software via a Volume Index of Defective Sectors. When a particular sector (or track) is deemed defective, allocation of space on the disk is made around the bad sector or track.

The controller provides no specific support for the identification or skipping of defective sectors or tracks.

Instructions

Table 6-20 lists the I/O commands to which the MSC/Disk Device-Pac/disk units respond. A detailed description of each command follows this table.

TABLE 6-20. CARTRIDGE DISK COMMANDS

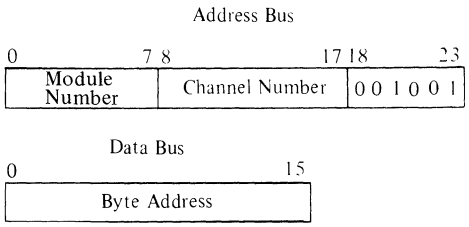
Type	Function Code	Command
Output	09[a]	Output Address
	[illegible]	Output Range
	0F	Offset Range
	11	Output Configuration Word A
	13	Output Configuration Word B
	03	Output Interrupt Control
	07	Output Task Word
	01	Output Control Word
Input	0C	Input Range
	0E	Input Offset Range
	10	Input Configuration Word A
	12	Input Configuration Word B
	02	Input Interrupt Control
	26	Input Device ID
	06	Input Task Word
	18	Input Status Word

[a]Function Code 09 as executed by the CP results in execution of functions 09 and 0D.

Output Commands

Output Address (09)

Format:

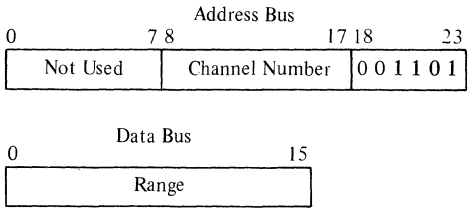


Function:

Loads a 24-bit address into the address register associated with the referenced channel (device). The address refers to the starting (byte) location in Main Memory where the MSC commences input or output data transfers. Bits 0–7 of the address bus (Module Number) are the most significant bits of the Address. The data bus contains the 16 least significant bits. Data transfers to or from memory normally are on a word basis, but byte mode transfers can occur associated with the first and/or last memory cycle of a particular data transfer if the Main Memory buffer (identified by this instruction) begins or ends on an odd byte boundary.

Output Range (0D)

Format:

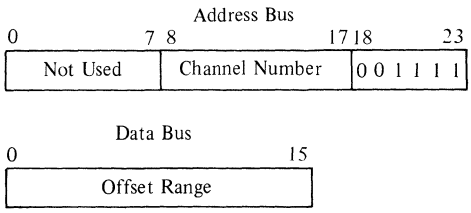


Function:

Loads the range register associated with the referenced channel. The (16-bit) quantity loaded (data bus) is the number of bytes to be transferred during the data transfer that is being set up. The number is a positive binary quantity (bit 0 must be zero) and is decremented by the MSC after each memory transfer. A range and offset range of zero result in a premature End-of-Operation for any Read or Write command that may be subsequently issued (see Output Task Word). Any range register residue is applied to the next command unless reset by another Output Range instruction.

Output Offset Range (0F)

Format:

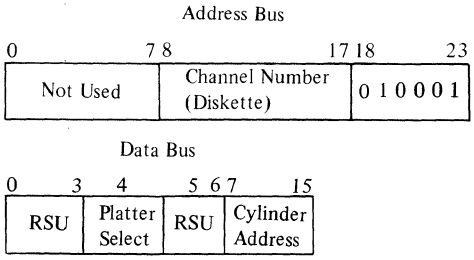


Function:

Loads the Output Offset Range Register associated with the referenced channel. The (16-bit) quantity loaded (data bus) is the number of bytes to be discarded from the beginning of the data transfer prior to the transfer of any data to main memory. The output range is decremented only after the offset range register is decremented to zero. The output offset range is used only in conjunction with read operations (offset range ignored for Write operations) and must be set for each data transfer using an offset. The offset is a positive binary quantity and is decremented by the MSC after each byte read from the disk. Any offset range register residue is applied to the next command unless reset by another Output Offset Range instruction.

Output Configuration Word A (11)

Format:



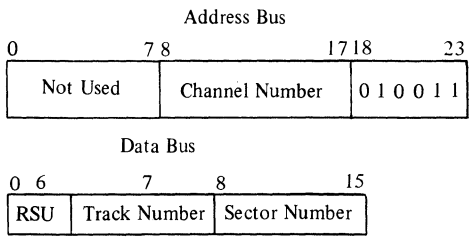
Function:

Loads Configuration Word A for the device corresponding to the referenced channel. The cylinder Address (bits 7–15) and Platter Select (bit 4) are used as the seek argument during Seek operations. The complete word is used as the two high order bytes of a Sector ID field to be searched for during an update Read or Write operation. Bits 0–3 and bits 5–6 are reserved for software use (RSU). The maximum cylinder address permissible is 203 or 407 depending on the device type. Bit 4 provides the platter selection address for any Read or Write operations. The platter selection is as follows:

Bit 4	
0	Removable media
1	Fixed media

Output Configuration Word B (13)

Format:



Function:

Loads Configuration Word B for the device corresponding to the referenced channel. This word is used as the two low order bytes of a Sector ID field to be searched for during a data field Read or Write operation. Bit 7 provides the track address for any Read or Write operations. The track address is as follows:

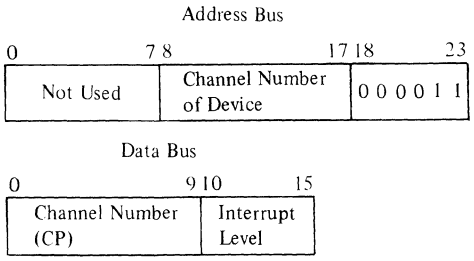
Bit 7	
0	Upper surface
1	Lower surface

The subsystem treats bits 8–15 of Configuration Word B as a sector number. This number is

incremented after operating on a data field during a data field Read or Write operation. Bits 0–6 are reserved for software use (RSU).

Output Interrupt Control (03)

Format:

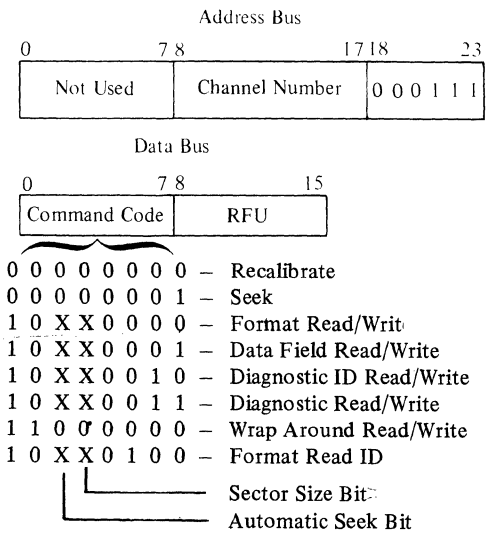


Function:

Loads, for the referenced device, the Interrupt level and the channel number of the CP to which subsequent interrupts should be sent. The level number is a 6-bit quantity and is positioned on the data bus as illustrated above. Bits 0–9 of the data bus contain the channel number of the CP to which the interrupt is to be sent. If an Interrupt level of zero is loaded, the subsystem does not generate or save interrupts for any events that occur while the interrupt level is zero. The Interrupt level is set to zero whenever the subsystem is initialized.

Output Task Word (07)

Format:



Function:

Outputs a Task Word to the referenced channel. The coding bits 0–7, illustrated above, represent the operations that are to be performed. When this instruction is accepted, the channel enters the Busy state and the indicated task is initiated or stacked. All address, range, and configuration information must be loaded prior to execution of this instruction. The direction of data transfer is indicated in the low order bit of the most recent output address instruction. For example, if the data field encoding of the Task Word is received when a Read channel number is indicated, a Read Data command is executed. Note that track selection (which includes the platter and surface) is performed for each media data transfer command prior to initiation of the data transfer and is based on the current contents of configuration words A and B.

In addition to the above functionality, multi-sector data transfer operations (read or write) enable an automatic track and cylinder function. Track switching occurs whenever the last logical sector (sector 23 for 256 byte sectors or sector 11 for 576 byte sectors) on the track has been completed and the range has not expired. Note that track switching is not associated with the Index Mark but with equality with the last sector number, i.e., if the sector number is greater than 11 or 23, track switching will not occur. When the last track of the cylinder has been completed and track switching is attempted, the controller initiates a Seek to the next consecutive cylinder number, selects track number zero, and initiates a search for sector number zero. Note that activities on other channels (including data transfers) may occur during the seek latency period.

Additional considerations are:

1. Data transfer continues until the range expires, an error is encountered, an unsuccessful search occurs or the end of the last cylinder is detected (setting bit 5 of the Status Word).
2. Automatic track switching does not occur for any format operation or any unsuccessful search (detection of two index marks without a successful compare).
3. When track completion is detected without error, the Configuration Words are modified to reflect the next consecutive track and sector zero. In addition, if cylinder completion is also detected, the Configuration Words are modified to reflect the next consecutive cylinder, and track and sector zero. Note that this update occurs only if the range is nonzero at the end of the last sector of the previous track.
4. An attempt to automatically switch off the last track of the last cylinder results in

Status bit 5 being set (except in the case where the range has been decremented to zero).

5. Track and cylinder switching occurs only within the platter selected by the Configuration Words. A multisector operation operates only within the selected platter; it will not link the fixed and removable platters.

Bit(s) 2–4 of the command code have specific meaning for all media data transfers as follows:

1. Bit 2 – Automatic Seek
   If this bit is a logical zero, the data transfer is initiated based on the current cylinder position of the drive.
   If this bit is a logical one, a Seek Cylinder operation, based on the current contents of Configuration Word A, is initiated to the drive. Other channels may be serviced by the MSC during the seek latency period. At Seek completion, no Seek Complete Interrupt is generated to the bus channel and the specified data transfer operation is initiated.
2. Bit 3 – Sector Size
   If this bit is a logical zero, data transfer operations assume that the track selected is formatted with 256-byte sectors.
   If this bit is a logical one, data transfer operations assume that the track selected is formatted with 576-byte sectors.
3. Bit 4 – Automatic RPS
   This bit is ignored.

Bits 2–4 can be set in any combination (they present no interrelationships) for a data transfer operation.

- Recalibrate – The Recalibrate command causes the channel to move the device's positioner to cylinder zero, select track zero of the removable platter, and reset the Seek Timeout line on the device interface. This instruction is intended as an Initialization command to guarantee that the positioner location information in the controller is correct and that the device faults are cleared. Completion of the recalibration operation by the device results in the generation of an appropriate interrupt.

- Seek – The Seek command in the Task Word causes the channel to move the device's positioner to the cylinder and selects the platter indicated in Configuration Word A. If the cylinder specified is greater than 203 or 407 (depending on device) or an error occurs during positioner movement, an error bit is set in the Status Word. Completion of a positioning operation by the device (whether or not any physical movement occurred) results in the generation of an appropriate interrupt.

- Format Read – The Format Read command causes the channel to read all Identifier (ID) and data fields on a track beginning with the first sector after index and in the order in which they are recorded. Data is transferred to memory beginning at the memory location specified in the subsystem's memory address register (after any offset has been exhausted). This address is the address loaded by the most recent Output Address instruction if no data transfer has occurred since that instruction was executed. If one or more Data Transfer operations have been executed since the last Output Address instruction, then the starting memory address used for this operation is the byte address immediately following the end of the most recent data transfer executed for this device (either Read or Write).

  If bit 8 of the command code is a one, then the EDC word of any Sector ID read is ignored. If bit 8 of the command code is a zero, then the EDC word of any Sector ID read is checked. As each sector transfer is successfully completed, the sector number field of Configuration Word B is incremented. Note that this command does not automatically reset the sector number to any preset value but the existing contents are incremented.

  Data is transferred until a read error occurs (except as noted for bit 8), the range is satisfied, or the entire track is read (index is detected).

  Normal range for this command (to read one complete track) is:

  $R = (4 + 256) \times 24 = 6240$ bytes (for 256-byte sectors)

  or

  $R = (4 + 576) \times 12 = 6960$ bytes (for 576-byte sectors)

where:

4 = ID
256/576 = Data
24/12 = Sectors

If this command is terminated due to end of track before the range is satisfied, the residual range is available via the Input Range command (see Input Range). A read error in any field (except as previously noted for bit 8 of the command code = 1) causes the operation to be terminated with the Read Error bit set in the Status Word (bit 4). The Sector ID field in Configuration Word B points to the record in error if Configuration Word B was properly loaded. The field in error can be determined through examination of the residual range (if a read error is detected in an ID field, the range will have been decremented for the ID field only).

If the required transfer rate is not maintained on the bus (156.25K words per second), the operation is terminated and the Overrun/Underrun bit is set in the Status Word (bit 2).

If the range and offset range registers are zero when this command is received, the Task is immediately terminated (End-of-Operation). No data is read or transferred. Track selection for the operation is based on the current contents of the track address of Configuration Words A and B.

- Format Read ID – The Format Read ID command is identical to the Format Read command except that only the IDs of each sector are transferred to memory.
  Normal range for this command (to read one complete track) is

  R = 4 x 24 = 96 bytes (for 256-byte sectors)

  or

  R = 4 x 12 = 48 bytes (for 576-byte sectors)

  where

  4 = number bytes in ID
  24/12 = number of sectors per track

- Format Write – The Format Write command causes the channel to format the track which is positioned under the Read/Write head specified by Configuration Words A and B when this command is received. Either 12 or 24 (as specified by bit 3 of the Task Word) equal-length sectors are written starting at index mark.

  The Sector ID fields are read from memory beginning with the memory location specified in the subsystem's memory address register.
  Data fields are written with normal data field Address Marks and are zero-filled by the MSC.
  The range to format one complete track is:

  R = 4 x 24 = 96 bytes (fors256-byte sectors)

  or

  R = 4 x 12 = 48 bytes (for 576-byte sectors)

  If a range other than that specified is sent for a format write, the remaining portion of the track is zero-filled. If the range was not a multiple of four bytes, the format of the last sector is unspecified. If the range is too large, Status bit 7 is set when an IM is detected.
  If the range register is zero when this command is received, the task is immediately terminated (End-of-Operation). No data is written.
  If the Sector ID data cannot be read from memory at a sufficient rate, the operation is terminated and the Overrun/Underrun bit is set in the Status Word (bit 2).

- Read Data – The Read Data command causes the channel to locate the sector defined by the Sector ID image loaded in Configuration Words A and B and begin to transfer the data field of that sector (at least) to main memory. Data is transferred to memory beginning with the memory location specified in the subsystem's memory address register after any offset has been exhausted (see Format Read) and continues until the range is satisfied. When the transfer of the first specified sector data field is completed (without error), the Sector Number field of Configuration Word B is incremented. If the initial range plus offset range is greater than 256 (or 576 if 576-byte sectors are specified), then the sector on that track represented by the updated contents of Configuration Words

A and B (A is unchanged) is located and data transfer continues with the new sector's data field. This operation continues until either the range is satisfied, a read error occurs, or the record specified by Configuration Words A and B cannot be located on the track (as indicated by the detection of two index marks without a successful compare). If the specified record cannot be located, then an unsuccessful search is posted in the Status Word (bit 7). Note that track and cylinder switching may occur.

Track selection for the operation is based on the current contents of the Track Address of Configuration Word A and B.

If a read error is encountered in a data field, the operation is terminated and the Read Error bit in the Status Word is set (bit 4). The Sector Number field of Configuration Word B contains the address of the record in error. If a read error is encountered in an ID field, a miscompare result is assumed and the search continues. In this case the Read Error bit is posted in the Status Word so that, if the desired record is never located, the operation will be terminated with both the Unsuccessful Search bit and Read Error bit posted in the Status Word indicating that the reason for the miscompare could be a read error in the Sector ID. If the search is eventually successful, the Read Error bit in the Status Word will be reset.

If this command is terminated before the range is satisfied, the residual range is available via the Input Range command (see Input Range). If the range and offset range registers are zero when this command is received, the task is immediately terminated (End-of-Operation). No data is read or transferred.

If the required transfer rate is not maintained on the bus (156.25K words per second), then the operation is terminated and the Overrun/Underrun bit is set in the Status Word (bit 2).

- Write Data – The Write Data command causes the channel to locate the sector defined by the Sector ID image loaded in Configuration Words A and B and to rewrite the data field of at least that sector. The data is read from memory beginning with the memory location specified in the subsystem's memory address register (see Format Read). Rewritten data fields will be preceded by normal data field sync words (see Figure 6-20).

  When the transfer of the specified sector is completed, the Sector Number field of Configuration Word B is incremented. If the range is less than 256 or 576, the data field is zero-filled. If the range is greater than 256 or 576, the sector represented by the updated contents of Configuration Words A and B (A is unchanged) is located and the data field rewritten (preceded by a data field sync word). This operation continues until either the range is satisfied or the record specified by Configuration Words A and B cannot be located on the track (as indicated by the detection of two index marks without a successful ID field compare). If the latter event occurs, Unsuccessful Search is posted in the Status Word (bit 7). Note that track and cylinder switching may occur.

  If two or more records are to be updated with a single Write Data command, they are written in sequence without the loss of a revolution.

  If a read error is encountered in an ID field, that sector is bypassed and the search continues. In this case, the Read Error bit is posted in the Status Word so that, if the desired record is never located, the operation will be terminated with both the Unsuccessful Search bit and the Read Error bit posted in the Status Word indicating that the reason for the unsuccessful search could be a read error in the Sector ID. If the search is eventually successful, the Read Error bit in the Status Word will be reset.

  If this command is terminated before the range is satisfied, the residual range is available via the Input Range command (see Input Range). If the range register is zero when this command is received, the Task is immediately terminated (End-of-Operation). No data is written.

  If the required transfer rate is not maintained on the bus (156.25K words per second), the operation is terminated and the Overrun/Underrun bit is set in the Status Word (bit 2).

  Track selection for the operation is based on the current contents of the Track Address of Configuration Words A and B.

- Write Diagnostic – The Write Diagnostic command causes the channel to perform as if the Write Data command was specified except that invalid EDC characters are written at the end of each data field updated (the EDC characters are always zero). Note that the data written in the data fields are read from memory and normal data field sync words are written (for all fields written by this command).

  Track selection for the operation is based on the current contents of the Track Address of Configuration Word A.

- Read Diagnostic – The Read Diagnostic command causes the channel to read everything on a track (including gaps, sync words, ID and data fields, and EDC bytes) beginning with the first sync word after the next sector pulse detected. Meaningful operation of this command can only be guaranteed on tracks which have not been updated since the last Format operation (updated records may contain write splice discontinuities). No EDC checks are done. Data is transferred until the range is satisfied.

  If the required transfer rate is not maintained on the bus (156.25K words per second), the operation is terminated and the Overrun/Underrun bit is set in the Status Word (bit 2). Track selection for the operation is based on the current contents of the Track Address of Configuration Words A and B.

- Write ID Diagnostic – The Write ID Diagnostic command causes the channel to perform as if the Format Write command was specified except that invalid EDC characters are written at the end of each ID field (the EDC characters are always zero). Note that the data written in the data fields are all zeros (including EDC characters).

- Read ID Diagnostic – The Read ID Diagnostic command causes the channel to perform as if the Read Diagnostic command was specified except that the data transfer begins with the sync word of the first physical sector on the track.

- Wraparound Read/Write – Two wraparound levels are available in the MSC/Device-Pac and are controlled by bit 7 (of the Task Word). If bit 7 is a logical zero, the wraparound level is at the Device-Pac level. If bit 7 is a logical one, the wraparound level is at the MSC level. In either case, functionality is as described in the following paragraphs.

  During a Wraparound Write command, the channel reads 1–8 words from memory (at the address specified in the subsystem's memory address register) and transfers the bytes to the appropriate (MSC or Device-Pac) FIFO (first-in/first-out) buffer.

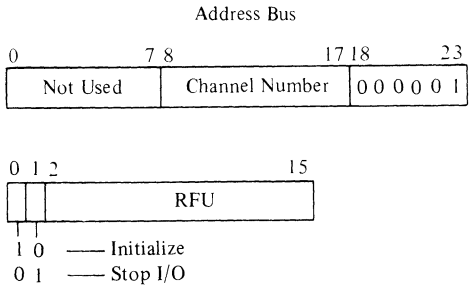
  When a Wraparound Read command is received (immediately following a Wraparound Write), the bytes previously loaded into the specified FIFO buffer by the previous Wraparound Write command are returned to main memory at the address specified in the subsystem's memory address register. The bytes returned during this operation are the same as the bytes supplied by software in the preceding Wraparound Write command. The range specified for the Wraparound Write must be the same as the range specified for the Wraparound Read or the results are unspecified.

  A range of 1–8 words should be specified for these commands. If a range and offset range of zero are specified, the command is immediately terminated (without being executed nor with any status indications). If a range greater than 8 words is specified, the results are unspecified. If bit 7 was a logical one, the overrun bit (bit 2) of the Status Word is set and the command is terminated immediately.

  Execution of a Task instruction to any other channel during a Wraparound sequence results in unspecified results.

Output Control Word (01)

Format:



Function:

Loads a Control Word into the referenced channel. This command is unconditionally

accepted by the channel regardless of its Busy status.

- Initialize – This command causes the MSC to reset to the same state that it enters after power up. When an Initialize command is received by the MSC, all of its channels are initialized (regardless of which channel the command was received over). A Recalibrate is executed on all cartridge disk drives.

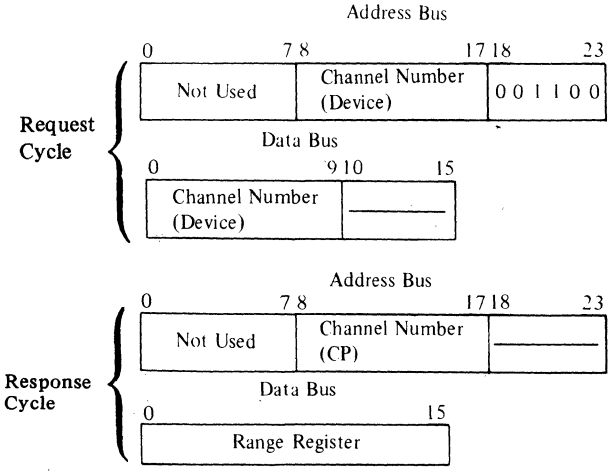
  Operations that are in progress in the MSC at the time of the Initialization are abruptly terminated and all registers are initialized. No information about the terminated operations is retained and no interrupts for the operations are generated. The interrupt level for all channels is set to zero (interrupts blocked).

- Stop I/O – This command causes any operation currently active on the specified channel to be abruptly terminated. If a data transfer operation is in progress, it is not completed nor is any error checking done. An Interrupt is generated for the operation terminated by this command as if the operation had come to a normal ending point. Status, Address, and Range information, present in the MSC when this command is received, is retained. Note that execution of this command may result in invalid data on the media (if a Write operation was in progress) or a device fault (if a Seek operation had been initiated and not completed prior to a subsequent operation).

Input Commands

Input Range (0C)

Format:



Function:

Causes the current contents of the referenced channel's range register to be transferred to the requesting channel.

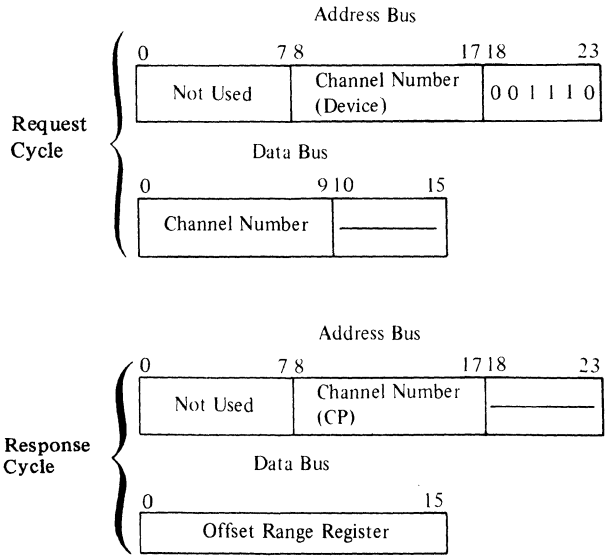
During the Response cycle (Second Half Read), the MSC returns, in bits 8–23 of the address bus, the same data that was received in bits 0–15 of the data bus during the Request cycle.

After the completion of a data transfer operation, the contents of the range register reflect the status of that transfer with respect to the physical sector(s) read.

- If the contents is a positive value greater than zero, the length of the physical sector(s) was less than the sum of the original offset range and range.
- If the contents is zero, the length of the physical sector(s) was equal to or greater than the sum of the original offset range and range.

Input Offset Range (0E)

Format:



Function:

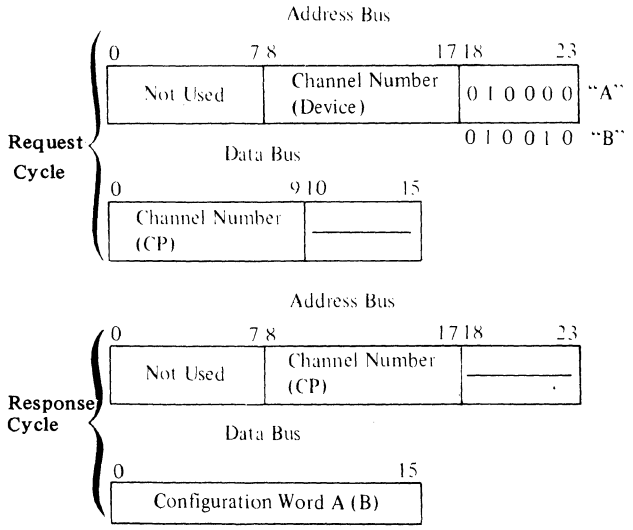
Causes the current contents of the referenced channel's offset range register to be transferred to the requesting channel. During the Response cycle (Second Half Read), the MSC returns, in bits 8–23 of the address bus, the same data that was received in bits 0–15 of the data bus during the Request cycle.

After completion of data transfer operation, the contents of the offset range register reflect the status of that transfer with respect to the physical sector(s) read.

- o If the contents is a positive value greater than zero, the length of the physical sector(s) was less than the original offset range.
- o If the contents is zero, the length of the physical sector(s) was equal to or greater than the original offset range.

Input Configuration Word A (10)

Format:



Function:

Causes the channel's Configuration Word A (B) to be transferred to the requesting channel. During the Response cycle (Second Half Read), the MSC returns, in bits 8–23 of the address bus, the same data that was received in bits 0–15 of the data bus during the Request cycle.

Input Configuration Word B (12)

Format:

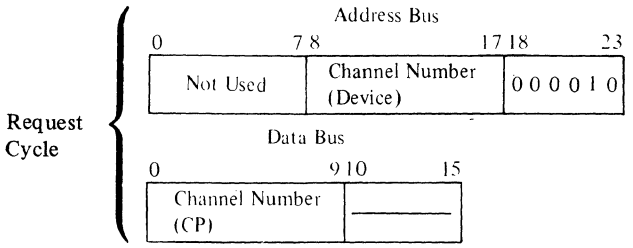
See "Input Configuration Word A"

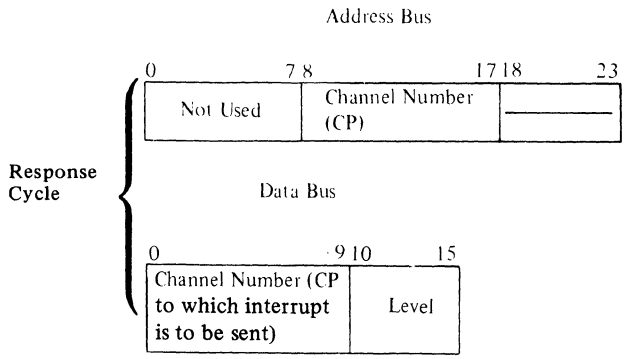
Function:

See "Input Configuration Word A"

Input Interrupt Control (02)

Format:





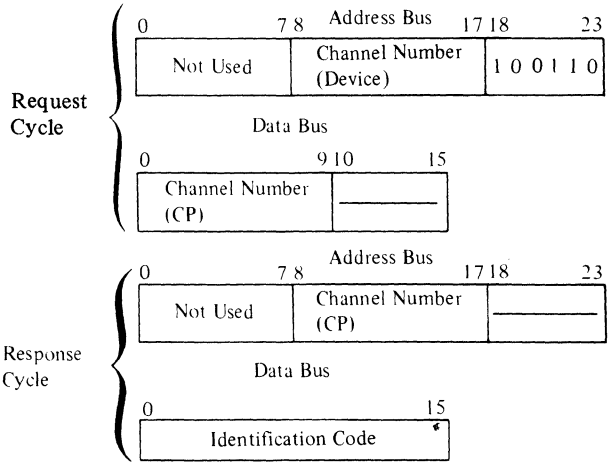
Function:

Causes the channel's interrupt level to be transferred to the requesting channel. The level value is placed on data bus bits 10–15 (see above) with bit 15 as the least significant bit. This quantity is the value previously received in the Output Interrupt Control instruction or a default value of 00. The default value is the Interrupt level assumed by the channel when initialized. Note that the channel number returned in bits 0–9 of the data bus might be different from the channel number of the CP executing this instruction if more than one CP is attached to the Bus.

During the Response cycle (Second Half Read), the MSC returns, in bits 8–23 of the address bus, the same data that was received in bits 0–15 of the data bus during the Request cycle.

Input Device ID (26)

Format:



Function:

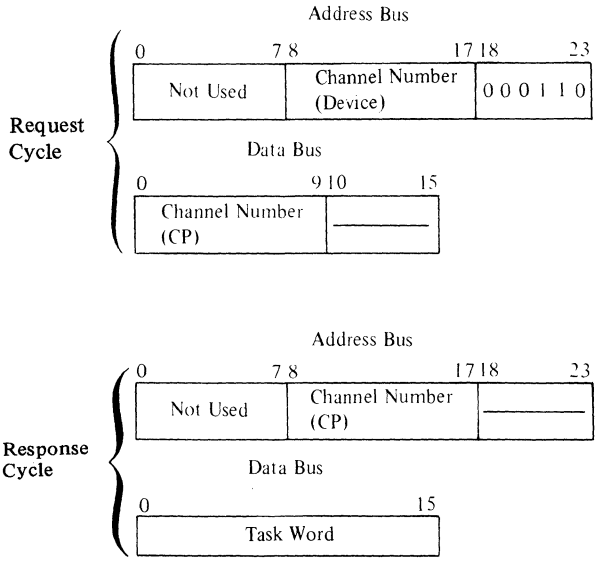
Causes the referenced channel to transfer its identification code to the requesting channel. Depending on the device accessed, one of the following codes is returned.

Code (Hex)	Model
2330	100 tpi, removable disk only (CDU9101)
2331	100 tpi, fixed and removable disks (CDU9102/9114)
2332	200 tpi, removable disk only (CDU9103)
2333	200 tpi, fixed and removable disks (CDU9104/9116)

During the Response cycle (Second Half Read), the MSC returns, in bits 8–23 of the address bus, the same data that was received in bits 0–15 of the data bus during the Request cycle.

Input Task Word (06)

Format:



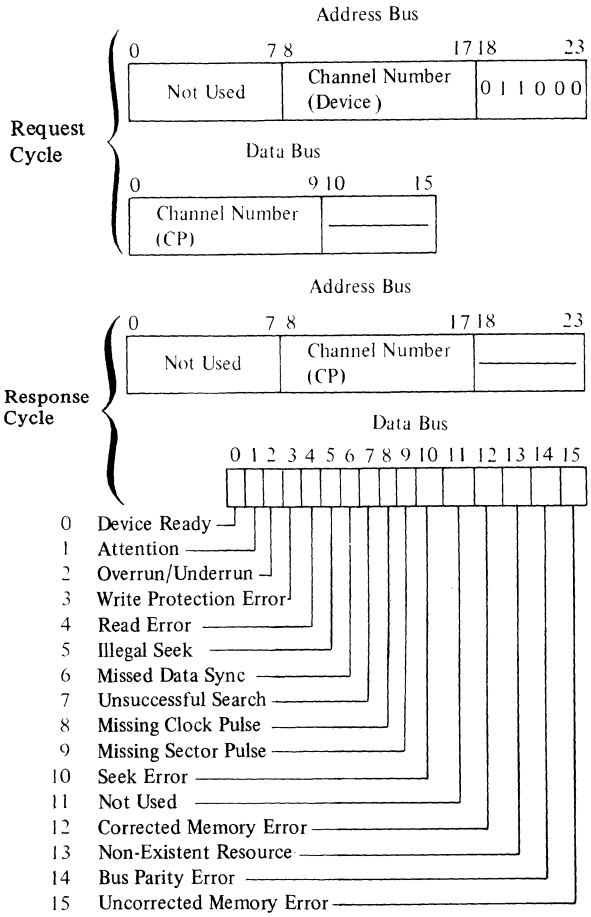
Function:

Causes the Task Word of the referenced channel to be transferred to the requesting channel. The Task Word transferred contains the code for the last operation executed by the channel (unless an Initialize has occurred).

During the Response cycle (Second Half Read), the MSC returns, in bits 8–23 of the address bus, the same data that was received in bits 0–15 of the data bus during the Request cycle.

Input Status Word (18)

Format:



Function:

Causes the referenced channel's Status Word to be transferred to the requesting channel. During the Response cycle (Second Half Read), the MSC returns, in bits 8–23 of the address bus, the same data that was received in bits 0–15 of the data bus during the Request cycle. See also Table 6-21.

TABLE 6-21. STATUS BIT DEFINITIONS – CARTRIDGE DISK

Status Condition	Bit	Definition	Reset by
Device Ready	0	The device is online with the medium loaded and no further manual intervention is required to place it under program control. Note that a change of state of this bit will cause the Attention bit (bit 1) to be set resulting in an Interrupt (if the interrupt level is nonzero).	A change in condition

TABLE 6-21 (CONT). STATUS BIT DEFINITIONS – CARTRIDGE DISK

Status Condition	Bit	Definition	Reset by
Attention	1	Set whenever the Device Ready bit (bit 0 of the status word) changed state. Indicates to software any change of operational status of the device (e.g., load/unload of media). Whenever set, an interrupt is attempted (if the interrupt level is nonzero). If a previously initiated operation is in progress when a device state change is sensed, the resultant interrupt (with the Attention bit set) serves as notification of both the end of the operation and the device state change.	Input Status Word or Output Task Word[a]
Overrun/ Underrun	2	Set during a Read or Write operation when the data transfer to/from Main Memory cannot be maintained at a high enough rate (156K words per second during field transfers in word mode). Either data was lost on input because of failure to keep up with device demands or data was unavailable on output when required by the device.	Output Task Word[a]
Write Protection Error	3	Set if an attempt is made to perform any Write operation on a protected surface (i.e., Write Protect is set on the device). Operator intervention is required to reset the Write Protect condition of the device.	Output Task Word[a]
Read Error	4	Set during any Read operation if the EDC word at the end of a field disagrees with the EDC word calculated while reading the field.	Output Task Word[a]
Illegal Seek (100 tpi device only)	5	Set if bit 7 of Configuration Word A is equal to a one during execution of a Seek command.	Output Task Word[a]
Missed Data Sync	6	Set if, after a Sector ID has been detected during any Read operation, the corresponding data field is not detected.	Output Task Word[a]
Unsuccessful Search	7	Set during a nonformat Read or Write operation for which the Sector ID specified in Configuration Words A and B cannot be located on the track. Also set during a Format Write operation if the track has been formatted.	Output Task Word[a]
Missing Clock Pulse	8	Set if the controller detects a missing clock pulse during Write operations. Note that a missing clock pulse during a Read operation results in a Read Error.	Output Task Word[a]
Missing Sector Pulse	9	Set if the controller detects no sector pulse for a period of 1.5 ms. Normally indicates that one of the fixed surfaces (tracks 2 or 3) has been selected on a device which has no fixed platter.	Output Task Word[a]

TABLE 6-21 (CONT). STATUS BIT DEFINITIONS – CARTRIDGE DISK

Status Condition	Bit	Definition	Reset by
Seek Error	10	Set during a Seek operation if the MSC receives a seek error indication from the device. Occurs if the device does not successfully complete a Seek operation, or if an attempt is made to Seek beyond the cylinder limits.	Recalibrate[a]
Corrected Memory Error	12	Indicates that during execution of the previous operation Main Memory detected and corrected a memory read error. Data delivered to the MSC was assumed to be correct.	Output Task Word[a]
Nonexistent Resource	13	Set whenever the MSC attempts a Write or Read Request bus cycle and receives a NAK response.	Output Task Word[a]
Bus Parity Error	14	Always zero (not supported by the MSC).	–
Uncorrected Memory Error	15	Indicates that during execution of the previous operation Main Memory detected a memory read error which the EDAC algorithm could not correct. Data delivered to the MSC was incorrect. Will not cause termination of the operation in progress (may result in bad data written on the medium).	Output Task Word[a]

[a]Initialize (output control word) and Master Clear on the Bus also reset these status bits.

SERIAL PRINTERS

The Types PRU9101/9102 Serial Printers (Figure 6-21) are self-contained, tabletop-sized printers designed for low-cost printing requirements. The printers have 64- and 96-ASCII code character sets, respectively. Both operate at 165 characters per second and print at line speeds of 60 lines per minute (lpm) for 132 chars./line and 200 lpm for 20–30 chars./line. Serial printer specifications are listed in Table 6-22.

The printers interface with the central processor by means of a single-board Multiple Device Controller (or Magnetic Tape Controller), a Printer Device-Pac (PRM9101), and a 50-foot cable. The printers can be made self-standing via Option PRF9101, the serial printer pedestal.

Features

- o Uppercase/lowercase print capability with PRU9102
- o Separate offline test mode for easy maintenance

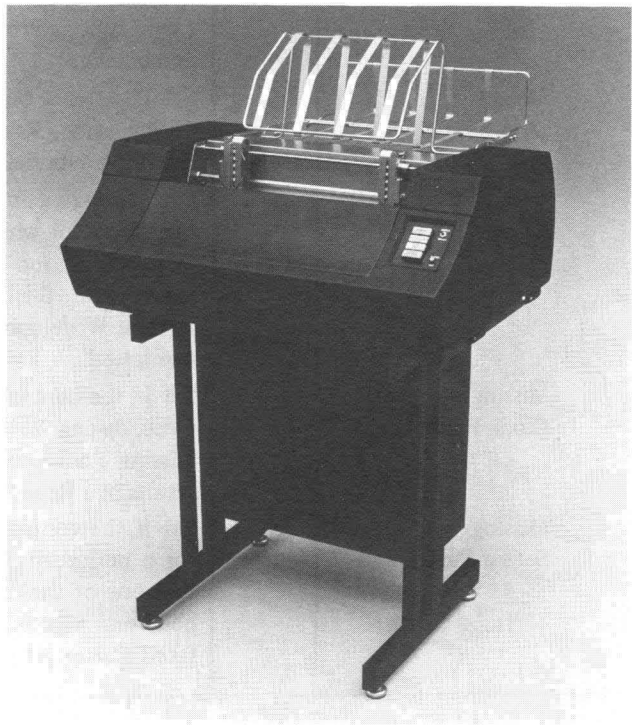


Figure 6-21. Level 6 Serial Printers

TABLE 6-22. SERIAL PRINTER SPECIFICATIONS

Characteristics \ Device	Serial Printers (PRU) 9101	9102
Print Speed (lpm)	60	
Columns per Line	132	
Print Format	10 cpi, horizontal; 6 cpi, vertical	
Character Set (ASCII)	64-character set	96-character set
Matrix Font	9 x 7 dot; equivalent to 10-point type	
Programmed Operations	Print only; space only; space and print	
Device Interface	Each serial printer requires its own Device-Pac (PRM9101).	
Print Ribbon	M3918	
Paper Stock	Standard fanfolded and edge-punched, 4.4 in. to 14.8 in. (11.2 cm to 37.6 cm) width, 15 lb (6.8 kg) minimum; multicopy – 12 lb (5.5 kg) maximum; with carbon – 5 parts – 6 lb (2.7 kg)	
Physical Dimensions		
Height	11.10 in. (28.2 cm)	
Width	28.8 in. 73.2 cm)	
Depth	19.2 in. (48.8 cm)	
Weight	118.3 lb (53.2 kg)	

- o Line speeds of 60 lpm at 132 characters per line; 200 lpm at 20–30 characters/line
- o Original and up to four carbon copies
- o Vertical forms tape permitting various form sizes

Operation

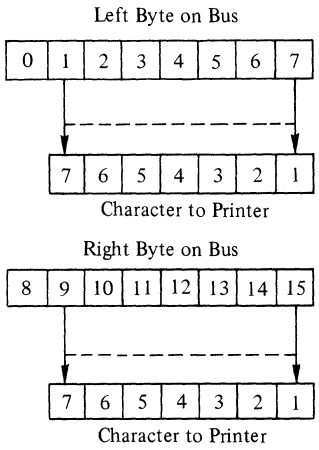
Data to the serial printer must be in standard ASCII 7-bit format. A maximum of 132 characters per line and only one line at a time can be printed per operation. All data transfers are under DMA (Direct Memory Access) control.

Spacing to top-of-form and testing for end-of-form are controlled via holes punched into a paper tape. The operator may, therefore, change the spacing by simply punching a new paper tape.

Data Handling

The range of the data buffer in memory is expressed in bytes and should be equal to or less than one print line. The number of bytes transferred may be even or odd and the starting address of the buffer may be on any byte boundary. The controller fetches words from memory but sends only the specified bytes to the printer.

The data to the printer corresponds to the bus data in the following way:



The eighth bit (bit 0 or 8 on the bus) is a "don't care" bit so that the same character is printed regardless of the state of that bit.

The 64- and 96-character ASCII sets are shown in Table 6-23. The controller/Printer performs limited code translation on the data. Figure 6-22 is a diagram of the data manipulation by the Device-Pac. The data buffer specified by the software is code-converted so that it is printed regardless of the number of printable characters supported by the printer (64 or 96). Control commands are supplied by the software driver separately from the data buffer and are inserted

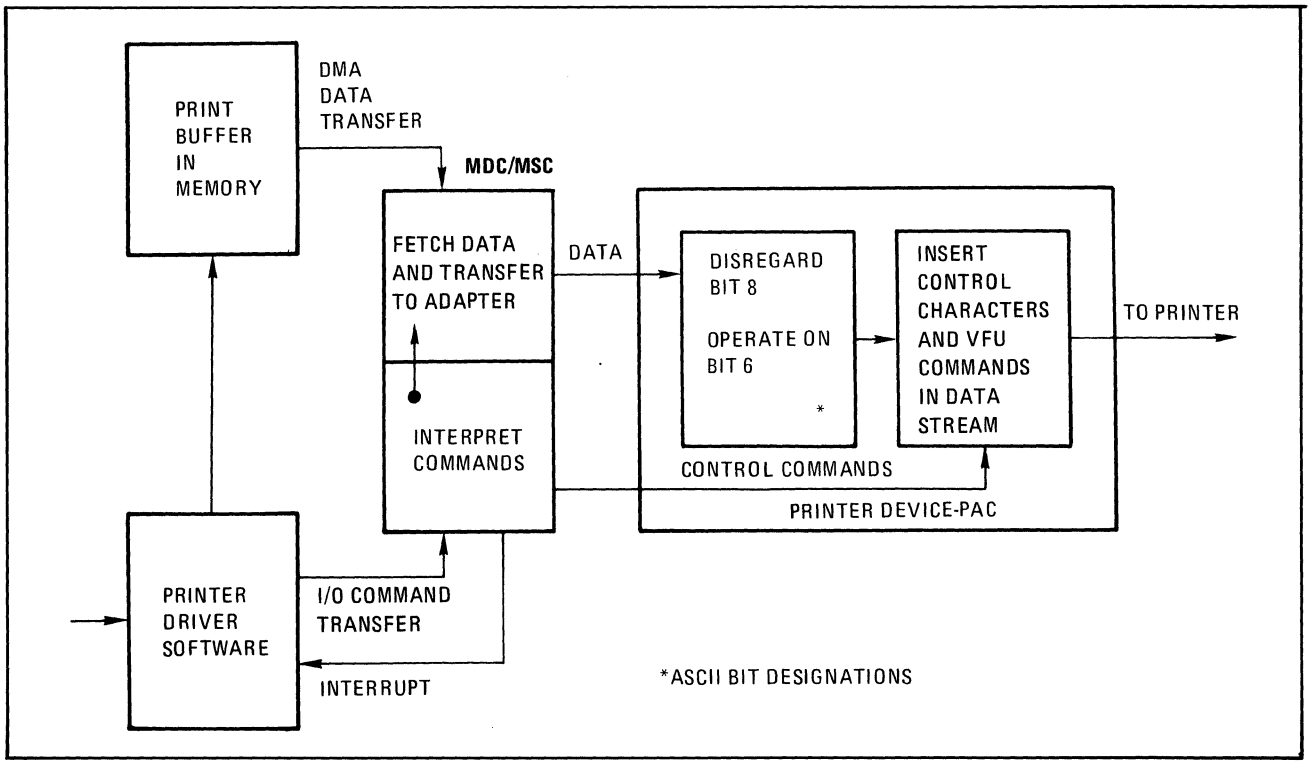


Figure 6-22. Data Manipulation of Printer Device-Pac

in the data stream after the code translation has taken place.

Code translation is limited to bit 6 (Bus bit 2 or 10) and is as follows:

64-Character Set	if b7, b6 = 00, set to 01 if b7, b6 = 11, set to 10
96-Character Set	if b7, b6 = 00, set to 01

TABLE 6-23. 96 CHARACTER ASCII SET

				B8	0	0	0	0	0	0	0	0
				B7	0	0	0	0	1	1	1	1
				B6	0	0	1	1	0	0	1	1
				B5	0	1	0	1	0	1	0	1
B4	B3	B2	B1	Col / Row	0	1	2	3	4	5	6	7
0	0	0	0	0	Space	0	Space	0	@	P	'	p
0	0	0	1	1	!	1	!	1	A	Q	a	q
0	0	1	0	2	"	2	"	2	B	R	b	r
0	0	1	1	3	#	3	#	3	C	S	c	s
0	1	0	0	4	$	4	$	4	D	T	d	t
0	1	0	1	5	%	5	%	5	E	U	e	u
0	1	1	0	6	&	6	&	6	F	V	f	v
0	1	1	1	7	'	7	'	7	G	W	g	w
1	0	0	0	8	(	8	(	8	H	X	h	x
1	0	0	1	9	)	9	)	9	I	Y	i	y
1	0	1	0	A	*	:	*	:	J	Z	j	z
1	0	1	1	B	+	;	+	;	K	[	k	|
1	1	0	0	C	,	<	,	<	L	\	l	|
1	1	0	1	D	-	=	-	=	M	]	m	~
1	1	1	0	E	.	>	.	>	N	^	n	^
1	1	1	1	F	/	?	/	?	O	-	o	Note

NOTE: Serial Printer (PRU9102) prints this as a space. Line Printers (PRU9103/9105) prints this as a box.

Instructions

Table 6-24 lists the I/O commands to which the controller/Device-Pac/printers respond. A detailed description of each command follows this table.

NOTE: The instructions and programming information presented in this section also pertain to PRU9103/ 9104/9105/9106 Line Printers.

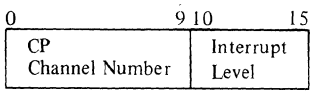
TABLE 6-24. PRINTER COMMANDS

Type	Function Code	Command
Output	03	Output Interrupt Control
	01	Output Control
	07	Output Task Word
	09	Output Address and Range
Input	02	Input Interrupt Control
	06	Input Task Word
	08	Input Memory Byte Address
	0A	Input Memory Module Address
	0C	Input Range
	18	Input Status Word 1
	26	Input Device ID

Output Commands

Output Interrupt Control (03)

Format:

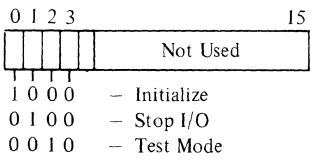


Function:

Loads a 16-bit word into the interrupt control register with the information necessary for generating an interrupt, i.e., the interrupt level and which CP the interrupt is to be issued to.

Output Control (01)

Format:



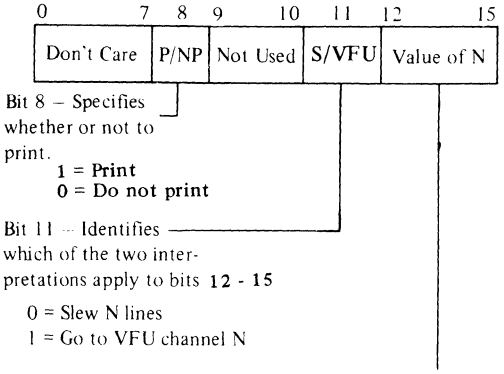
Function:

Loads a 16-bit control word into the referenced channel which is accepted unconditionally regardless of any channel Busy status. The channel may respond with a WAIT, but never with a NAK. The individual bits cause the following specific actions to occur:

- o Initialize
  - – Causes the controller to run its resident logic test
  - – Clears all controller Device-Pacs
  - – Operates on all channels of controller
  - – Makes all channels of controller nonbusy
  - – Blocks interrupts
- o Stop I/O
  - – Makes channel nonbusy (causes interrupt if enabled)
  - – Does not affect other channels
  - – Causes channel to terminate its I/O operation and update status
- o Test Mode
  - – See "Test Mode" earlier in this section.

Output Task Word (07)

Format:



Bits 12 - 15 Specify a number which may be either a VFU channel or the number of lines to be spaced. Bit 11 identifies which of the two interpretations apply to bits 12 - 15 . The maxium value of N is 15 for a spacing operation. For a VFU operation, the maxium value of N is 12. Printers without VFU go to top-of-form by being commanded to go to VFU channel 0.

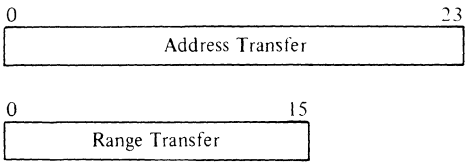
The device ID number can be used to determine which of the printers is installed.If VFU command is issued,where a VFU number is too large,the VFU command is ignored and the data is printed with a single line space.

Function:

Outputs a word that determines the specific printer action to be performed.

Output Address and Range (09)

Format:



Function:

The IOLD command when executed by the processor creates two Bus transfers to the controller. The first is the 24-bit address transfer and the second is the 16-bit range transfer. To the controller, these are two separate and distinct Bus transfers with two separate and distinct function codes of 09 for the address transfer and 0D for the range transfer. The programmer need only specify the first function code and the processor hardware/firmware automatically calculates the second function code (by adding 04 to it).

- o *Address Transfer* – A 24-bit quantity transferred to the controller to be used as the starting byte address of the data record in memory which is to be transferred.
- o *Range Transfer* – A 16-bit quantity transferred to the controller to be used as the byte range count of the data record in memory which is to be transferred. Range is specified as a twos complement integer which must be positive for the controller. Therefore, for the controller, Range is

$$1 \leqslant r \leqslant 2^{15} - 1$$

Range should be restricted to the line length of the printer attached. The reaction of the printers to a buffer in excess of this is:

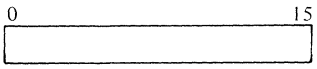
1. Line Printer – Characters in excess of 136 are output by the printer Device-Pac, but discarded by the printer.
2. Serial Printer – Characters in excess of 132 are overprinted starting with the first column.

The range transfer includes the implicit command Start I/O and causes the controller to start printing and to go Busy.

Input Commands

Input Interrupt Control (02)

Format:

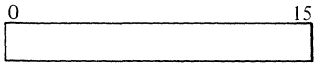


Function:

Causes the channel to place the contents of its interrupt control register on the data bus.

Input Task Word (06)

Format:

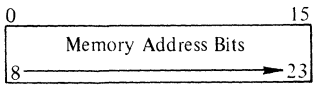


Function:

Causes the channel to place the contents of its task register on the bus.

Input Memory Byte Address (08)

Format:



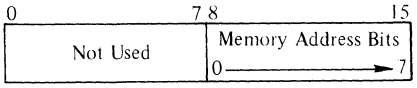
Function:

Causes the channel to place the 16 least significant bits of its DMA address register on the data bus. The bits represent the memory address register bits 8–23.

NOTE: This command will violate memory protection.

Input Memory Module Address (0A)

Format:



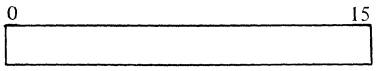
Function:

Causes the channel to place the eight most significant bits of its DMA address register on the data bus. The bits represent memory address bits from 0–7 inclusive.

NOTE: This command will violate memory protection.

Input Range (0C)

Format:

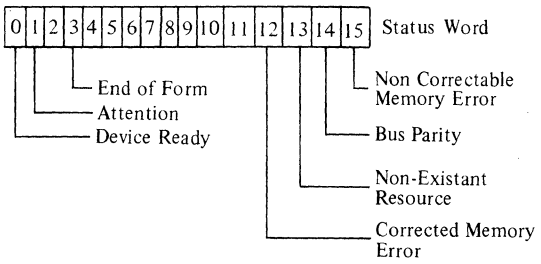


Function:

Causes the channel to place the contents of its range register on the data bus.

Input Status Word (18)

Format:



Function:

Causes the channel to place the contents of its status register on the bus. Also see Table 6-25.

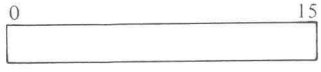
Input Device ID (26)

TABLE 6-25. STATUS BIT DEFINITIONS – SERIAL AND LINE PRINTERS

Status Condition	Bit	Definition	Reset by
Device Ready	0	Printer power on. No faults. Paper loaded. No open interlocks. Switches in correct positions.	A change in conditions
Attention	1	Device ready condition has changed.	Input Status word clears this bit.[a]
End of Form	3	Printer has detected end-of-form either by VFU, if equipped, or otherwise by preconfigured line count.	Next IOLD Command[a]
Corrected Memory Error	12	Memory read response was accompanied by a signal indicating the error was corrected.	Next IOLD Command[a]
Nonexistent Resource	13	A NAK was received from memory. Indicates possible programming error (nonexistent memory).	Next IOLD Command[a]
Parity	14	Parity was incorrect on a bus transfer toward the controller.	Next IOLD Command[a]
Uncorrectable Memory	15	Memory read response was accompanied by a signal indicating an error existed that could not be corrected.	Next IOLD Command[a]

[a]Initialize (output control word) and Master Clear on the Bus also reset these status bits.

Format:



Function:

Causes the channel to place its device identification number on the data bus.

- o Serial Printers – 2004 (PRU9101)
  2006 (PRU9102)
- o Line Printers – 2000 (PRU9104/9106)
  2001 (PRU9104/9106 with PRF9102)
  2002 (PRU9103/9105)
  2003 (PRU9103/9105 with PRF9102)

LINE PRINTERS

The PRU9103/9104/9105/9106 Printers (Figure 6-23) are designed to provide Level 6 users with high-quality printed output at medium to high print rates of 240 to 600 lines per minute (lpm). The PRU9104/9106 and PRU9103/9105 have 64- and 96-ASCII character code sets, respectively. All printers offer 136-column printing and provide for vertical spacing of 6 or 8 lines per inch. Format flexibility can be greatly enhanced via Option PRF9102 Vertical Format Unit which enables paper spacing to be controlled by a 12-channel paper tape. The flexibility of uppercase and lowercase printing is available with the PRU9103 and 9105. Line printer specifications are listed in Table 6-26.

The printers interface with the Level 6 processor by means of a single-board Multiple Device Controller (or Magnetic Tape Controller), a Printer Device-Pac (PRM9101), and a 50-foot (max. length) cable.

Figure 6-23. Level 6 Line Printers

TABLE 6-26. LINE PRINTER SPECIFICATIONS

Characteristics \ Device	Line Printers (PRU) 9103	9104	9105	9106
Print Speed (lpm)	240	300	440	600
Columns per line	← 136 →			
Line Advance Speed (ms per line–maximum)	← 50 →			
Paper Slew Speed (ips–minimum)	← 20 →			
Vertical Spacing	6 or 8 lines per inch; Option PRF9102 VFU Format Unit			
Printable Characters	96	64	96	64
Device Interface	Each line printer requires its own Device-Pac (PRM9101).			
Print Ribbon	M3910 or M3916			
Paper Stock	Standard fan-folded and edge-punched, 4.0 in. to 15.98 in. (10.2 cm to 40.6 cm) wide, with 11.0 in. (27.9 cm) between folds. When VFU option is present, distance between paper folds may be varied from 4.01 in. to 17.99 in. (10.2 cm to 45.7 cm) at 8 lines per inch, or 4.01 in. to 23.97 in. (10.2 cm to 60.9 cm) at 6 lines per inch. Weight for single copy 14.93 lb (6.72 kg) bond (min.) For up to six copies–12.22 lb (5.5 kg) bond with carbon (min.) Maximum form thickness–0.023 in. (0.58 mm)			
Physical Dimensions				
Height	45 in. (114 cm)			
Width	33 in. (83.8 cm)			
Depth	22 in. (55.8 cm)			
Weight	340 lb (153 kg)			

Features

- o Original and up to five copies
- o Simplified operation – fault indicator panel for rapid identification of operator-correctable problems
- o Easy paper loading with front-access, swing-open, drum gate mechanism
- o Full line (136 characters) buffer, loaded at 5–10 μs per character under DMA control
- o Byte-mode addressing to conserve memory
- o 6 or 8 lines per inch selection
- o Separate offline test mode for easy maintenance
- o Uppercase/lowercase print capability with PRU9103/9105

Operation

Printing operations for printers without the VFU option include:

- o Print without spacing
- o Space 1 to 15 lines
- o Space 1 to 15 lines and print
- o Space to head-of-form and print

For printers with the VFU option the following operations can also be performed:

- o Space to channel 1 (or channels 2 through 12) and print
- o Space to channel 1 (or channels 2 through 12) and do not print

All data transfers are under DMA (Direct Memory Access) control. While the printer is busy, the computer is free to perform other operations.

Instructions

The instructions and programming information presented for the serial printers are also applicable for the line printers.

MAGNETIC TAPE UNITS

The MTU9104/9105 and MTU9112/9113 Magnetic Tape Units are compact, self- contained, rack-mountable tape units (Figure 6-24). The MTU9104/9105 are 9-track, 800 bits per inch (bpi) units with transport speeds of 45 and 75 inches per second (ips), respectively. The MTU9112/9113 are 7-track, 556/800 bpi units with transport speeds of 45 and 75 ips, respectively. The recording technique used is NRZI (non-return-to-zero-invert). Table 6-27 lists the tape unit specifications.

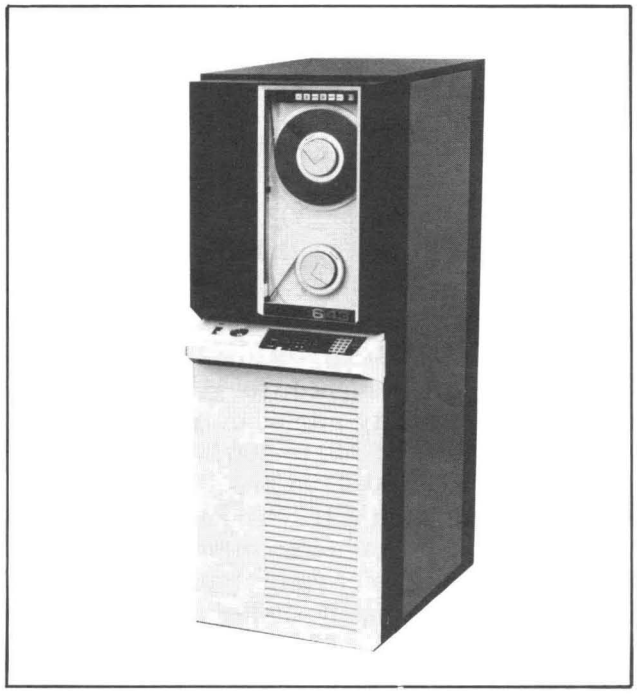


Figure 6-24. Level 6 Magnetic Tape Unit

TABLE 6-27. MAGNETIC TAPE UNIT SPECIFICATIONS

Characteristics \ Device	Magnetic Tape Units (MTU) 9104	9105	9112	9113
Number of Tracks	9	9	7	7
Tape Density (bpi)	800	800	556/800	556/800
Read/Write Speed (ips)	45	75	45	75
Rewind Speed (ips)	200	250	200	250
Transfer Rate	36K bytes/ second	60K bytes/ second	25/36K char./ second	41.7/60K char./ second
Interrecord Gap	0.60 in. (1.5 cm)	0.60 in.	0.75 in. (1.9 cm)	0.75 in.
Vacuum Column	Single	Dual	Single	Dual
Device Interface	MTM9102	MTM9102	MTM9101	MTM9101
	Only one NRZI Device-Pac is required for up to four 7- or 9-track tape units.			
Tape	2400-foot reels of ½-inch Mylar-base certified for 800 bpi; reel diameter of 10½-inch; IBM-compatible reel hubs			
Physical Dimensions				
Height	24 in. (61 cm)			
Width	19 in. (48.3 cm)			
Depth	(overall) 15.4 in. (39.1 cm)			
Weight	110 lb (49.5 kg)	120 lb (54 kg)	110 lb (49.5 kg)	120 lb (54 kg)

The Magnetic Tape Units interface with the Level 6 Megabus by means of a Magnetic Tape Controller (MTC9101) and a single 9-track NRZI Device-Pac (MTM9102) or single 7-track NRZI Device-Pac (MTM9101).

The MTC9101 offers the capability of attaching magnetic tape units as well as unit record devices to the system with a single controller. Incorporating sophisticated hardware and firmware, the MTC9101 microprogrammed control supports the connection of one of the following configurations (maximum is four devices):

- o Up to four 7-track MTU9112/9113 magnetic tape units
- o Up to four 9-track MTU9104/9105 magnetic tape units
- o Up to two or three magnetic tape units, and up to one or two (same or combination) of the following Level 6 unit record devices:
  - – card readers
  - – serial printers
  - – line printers

A combination of 7-track and 9-track magnetic tape units may not be used on the same MTC9101; however, they may be used on the same Level 6 system if a second MTC9101 is configured. Multiple MTC9101s can also be configured if extra functionality such as read-write simultaneity, additional tape drives, and additional unit record devices is required. Tape speeds of 45 ips and 75 ips may be mixed on the same MTC.

Features

- o Choice of 45/75 ips, 7-/9-track tape units
- o Dual-density, 556/800 bpi on 7-track tape units
- o Simple loading and operating procedures
- o Vacuum tape cleaner
- o Quick-release reel hold-down hubs
- o Rack-mountable in either an expansion cabinet or in the central processor cabinet (see Figure 6-24)

Operation

All input or output data transfers take place directly between the MTC and memory. A single tape unit can be reading or writing simultaneously with other units rewinding and with unit record device I/O operations. Only one tape device can be transferring data at one time.

Devices attached to the MTC are independently software addressable via switch selectable channel numbers by which I/O commands may be issued or status information retrieved.

All MTC devices operate in DMA mode; therefore, to set up a data transfer, I/O commands must be issued to the appropriate channel. The MTC and the device then perform the specified operation and at completion issue an optional interrupt. Status information is retrieved following completion of the operation.

Some of the items necessary to set up a channel for I/O commands include: DMA range, DMA address, CP channel number, interrupt level plus device specific configuration commands. The status report includes Bus status, MTC status, and device status pertinent to the previous operation.

The device specific functionality of the MTC is provided in part by the Device-Pac and in part by the firmware located in the MTC ROM. The firmware serves to interpret I/O commands for the device, handles DMA transfers, assembles the status bits, and accomplishes a large part of the dialog with the device.

Overall operation of the devices attached to the MTC is simultaneous. DMA transfers are interleaved on the Bus as are I/O commands and interrupts. MTC firmware manages this interleaving and assures that the devices are served with maximum speed to maintain their rated performance.

The devices attached to the MTC each have their own software-resettable interrupt level. This permits software to select different priority levels for device interrupts depending upon speed or application of the device, or to alter the interrupt level of the device from time to time.

The MTC9101 maintains 32 registers (16 bits per register) for each device. The address of each of the various registers in the MTC is a combination of 2 bits of the channel number and the high-order 5 bits of the function code used to write into or read from a particular register (see Figure 6-25). For example, the Configuration Word for MTC device 2 is MTC register 48 (hex).

Complete software visibility to the MTC registers is provided for diagnostic purposes. An Output Bus sequence addressed to one of the devices causes the information on the Data Bus (16 bits) to be loaded into the device specific register specified by the device port number and the high-order 5 bits of the function code.

The Output Address command is a special case. When an Output Address command is executed (on Port 0, for example), MTC register 04 (hex) is loaded with the low-order 16 bits of the address. The high-order 8 bits of

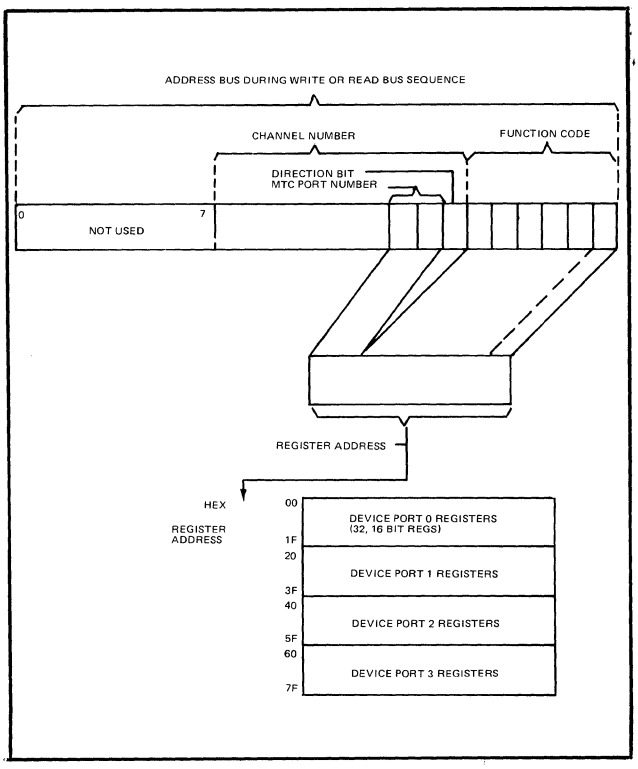


Figure 6-25. MTC Device-Specific Registers and Addressing

MTC register 05 are loaded with the high-order 8 bits of the address.

An Input Bus sequence addressed to a device causes the register specified by the Port number and the high-order 5 bits of the function code to be returned to the requester via the Data Bus (during the second half read cycle).

Function Code for
Configuration Word = 0100X (X = Read/Write bit)
Device Number = 010
Register Number = 010 01000 = 48 (Hex)

Magnetic Tape

The physical layout of half-inch magnetic tape is illustrated in Figure 6-26. A full reel of tape has a nominal recording length of 2400 feet (732 m); the entire length of the tape is oxide coated. Beginning- and end-of-tape sensing is controlled by reflective markers affixed to the Mylar-base side of the tape. The beginning-of-tape (BOT) spot is attached approximately 16 feet (4.88 m) from the physical beginning of the tape, and the end-of-tape (EOT) spot is attached approximately 25 feet (7.62 m) from the physical end of the tape.

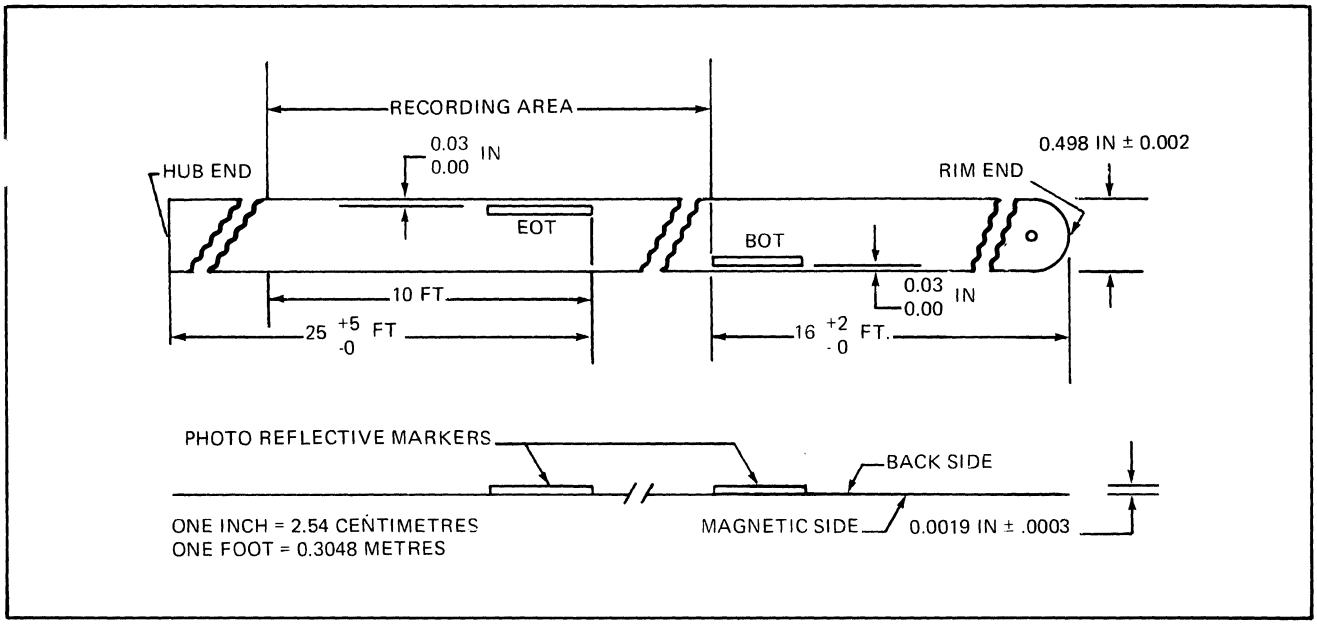


Figure 6-26. Magnetic Tape Layout

Beginning and End of Tape

In order to ensure maximum reliability in the storage of data, an erased area must be recorded in the vicinity of the beginning-of-tape (BOT) marker that is affixed near the reference edge at the start of every tape, and an unrecorded area must be left in the vicinity of the end-of-tape (EOT) marker affixed on the opposite edge of tape at the trailing end of a tape reel.

Beginning-of-Tape Gap

An erased section of tape is required surrounding the BOT marker. This serves as a defined area within which reading can start. This section begins a minimum of 1.3 in. (3.3 cm) before the hub end of the BOT marker and extends a minimum of 3.0 in. (7.6 cm) past the hub end of the BOT marker. This erased section totals about 4.3 in. (10.9 cm).

Data Blocks

The data is formatted and recorded on the tape in blocks. The exact configuration of a block depends on whether the tape is in 7-track or 9-track format. On 9-track tape, each block consists of the data, a cyclic redundancy Check (CRC) character, and a Longitudinal Redundancy Check (LRC) character. The CRC character is positioned four character times after the final data character, and the LRC character must occur four character times after the CRC. A nominal spacing of 0.6 in. (1.5 cm) is required between blocks.

On 7-track tape, each block consists of data, followed by an LRC character four character times after the final data character. Nominal spacing between blocks is 0.75 in. (1.9 cm).

Standard block formats used for NRZI recording are shown in Figures 6-27 and 6-28 (data block and tape mark block formats respectively).

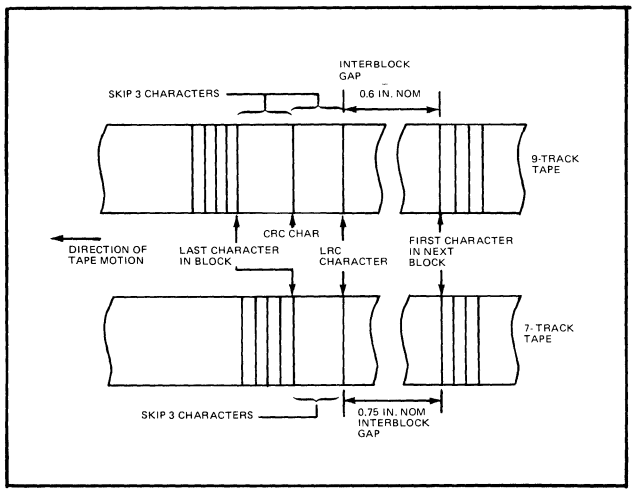


Figure 6-27. Data Block Formats

Interblock Gaps

Interblock gaps are areas without data (i.e., all tracks restored to the dc-erased polarity) placed between data blocks or records. The length of the gap is 0.75 in. (1.9 cm) nominal (0.6 in./1.5 cm minimum) for 7-track subsystems and 0.60 in. (1.5 cm) nominal (0.5 in./1.3 cm minimum) for 9-track subsystems. The maximum length should not exceed 25 feet (7.6 m).

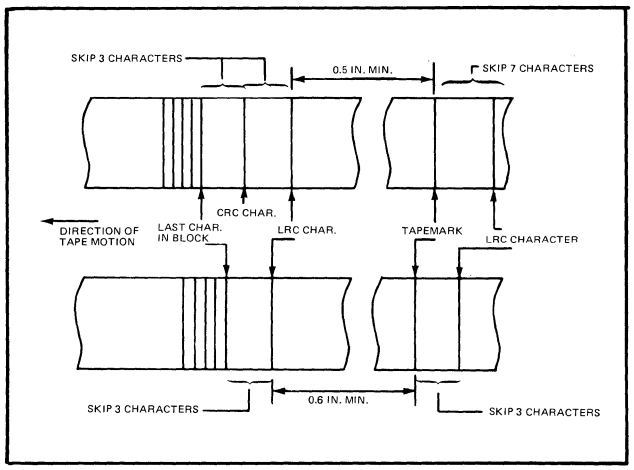


Figure 6-28. Tape Mark Block Formats

Tape Marks

Figure 6-28 shows the format of the Tape Mark for 7-track and 9-track tapes. The distinguishing feature of the block is that it is a single, specific character block with a check character.

NRZI Data Recording Format

In NRZI coding, a logic One bit appears on the interface lines as a low voltage level and a logic Zero as a high voltage level. However, on the tape, a logic One bit is recorded as a flux change and a logic Zero bit as no change. The direction of the change is immaterial; however, the initial flux change direction after BOT conforms to ANSI standards.

Check Characters

The NRZI format provides for both vertical and longitudinal parity checks. In the 9-track system an additional check called the Cyclic Redundancy Check character is used. Refer to Figures 6-27 and 6-28 for the location of the check characters.

- *Vertical Parity* – 7-track and 9-track tape subsystems use 6 and 8 tracks, respectively, for recording data. The remaining track carries parity information. When performing a Write operation, one parity bit is generated, either odd parity or even parity based on a previously stored Configuration Word, to accompany each character written on tape. The 9-track subsystems read and write odd parity only while 7-track subsystems allow the selection of odd or even parity to be recorded or read.

  When performing a Read (including Read-After-Write) operation, the parity read is checked against the parity created from the data portion of the character read. A mismatch causes a Vertical Redundancy Check (VRC) error condition to be set in Status. Note, however, if an even number of bits in the data character are "dropped" or "picked," a VRC error will not be detected; thus additional checking facilities are necessary.
- *Longitudinal Parity* – A Longitudinal Redundancy Check (LRC) character is written following the data portion of each block. It is separated from the end of the data or CRC in each block. This character is made up on a per-track basis. The LRC character written is one calculated so that an even number of "one" bits, including those of the data and LRC character, is recorded in each track of the block. On reading, this is checked and an error is detected if the count is odd in any track. This possibility of detecting an erroneous block still exists if an even number of bits in a given track of a block is dropped or picked. However, when this test is combined with the vertical parity test, the probability of not detecting an error is reduced.
- *Cyclic Redundancy Check (CRC) Character* – In the 9-track system another check character is written. This character is derived with relatively complex logic and, along with the LRC character and vertical parity, minimizes the possibility of undetected errors. The CRC character bits of a Tape Mark Block are all zeros. This check character follows the last data byte of the block by four cell positions.

Data Formats

7-Track

Information can be written or read in two modes:

- Byte Mode – Transfers 12 bits (of 16) of a data word to or from the tape as shown in Figure 6-29 (the four nonsignificant bits – bits 0, 1, 8, and 9 of the memory word – are ignored on writes and zero-filled on reads from tape).
- Pack Mode – Transfers three characters between memory and the tape. On writing tape, the controller generates two bits (zeros) to complete the six bits of the third character. On reading tape, the controller strips off these two bits (Figure 6-26).

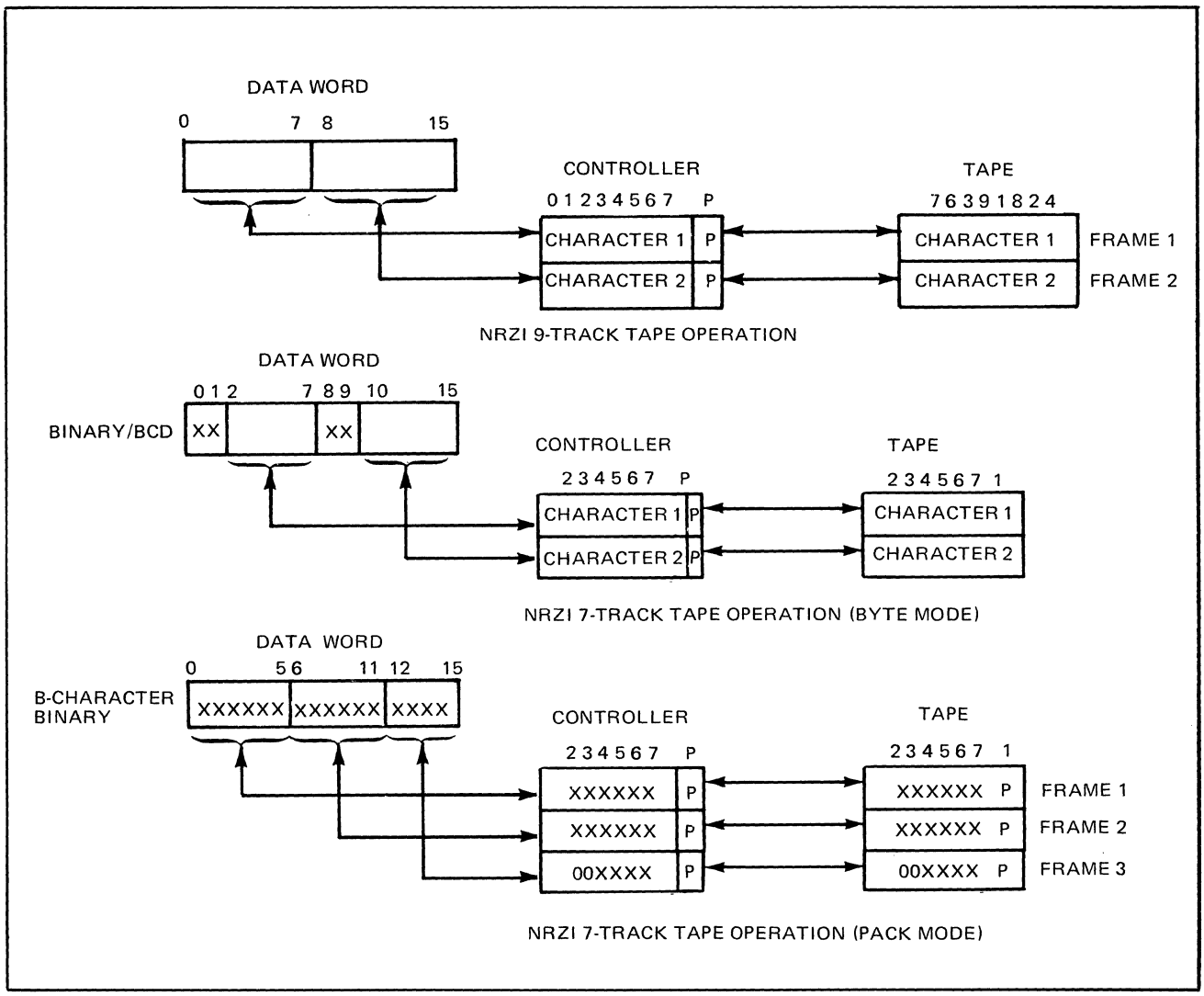


Figure 6-29. Data Formats

Even parity is used in the BCD (byte) mode and odd parity is used in the binary modes. In either mode, longitudinal even parity is indicated for each track at the end of a record. Since the file mark (17_8) is a single frame record, the file mark longitudinal parity frame is identical to the file mark itself.

To accomplish IBM tape code compatibility in the BCD mode, the octal value 00 is converted to the octal value 12 when writing. Conversely, when reading, the octal value 12 is converted to the octal value 00.

9-Track

Data being written on or read from tape is handled on a byte basis. All 16 bits of a data word are transferred to or from the tape as shown in Figure 6-29. Odd parity is written on tape and is checked when read.

Instructions

Table 6-28 lists the I/O commands to which the MTC/Magnetic Tape Device-Pac/tape units respond. A detailed description of each command follows this table.

TABLE 6-28. MAGNETIC TAPE COMMANDS

Type	Function Code	Command
Output	09[a]	Output Address
	0D	Output Range
	11	Output Configuration Word
	03	Output Interrupt Control
	07	Output Task Word
	01	Output Control Word

TABLE 6-28 (CONT). MAGNETIC TAPE COMMANDS

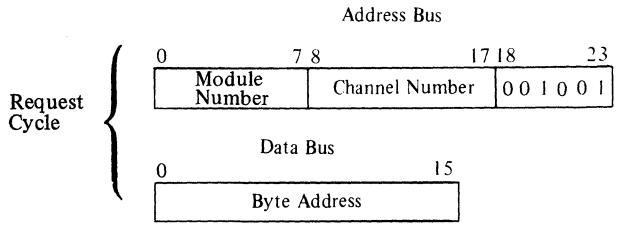
Type	Function Code	Command
Input	0C	Input Range
	10	Input Configuration Word
	02	Input Interrupt Control
	26	Input Device ID
	06	Input Task Word
	18	Input Status Word 1
	1A	Input Status Word 2

[a]Function Code 09 as executed by the CP will result in execution of functions 09 and 0D.

Output Commands

Output Address (09)

Format:

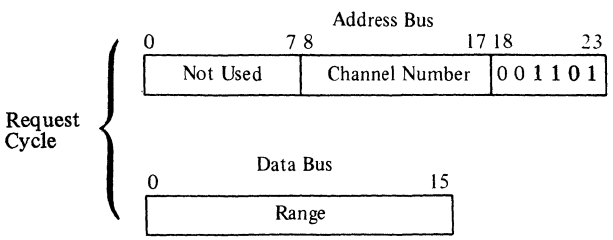


Function:

Loads a 24-bit address into the address register associated with the referenced channel (device). The address refers to the starting (byte) location in main memory where the MSC will commence input or output data transfers. Bits 0-7 of the Address Bus (Module Number) are the most significant bits of the Address. The Data Bus contains the 16 least significant bits. Data transfers to or from memory will normally be on a word basis but byte mode transfers can occur associated with the first and/or last memory cycle of a particular data transfer if the main memory buffer (identified by this instruction) begins or ends on an odd byte boundary.

Output Range (OD)

Format:

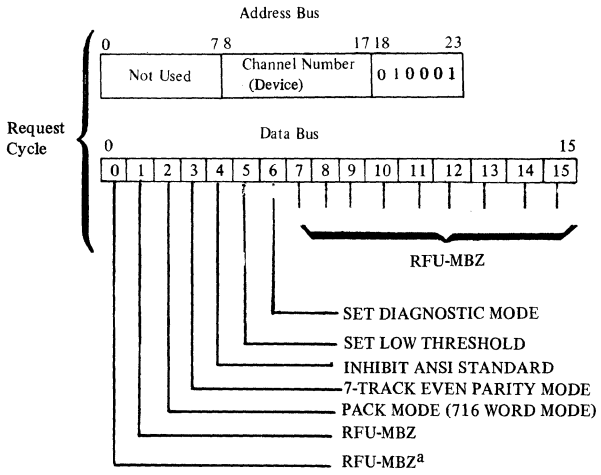


Function:

Loads the Range register associated with the referenced channel. The (16 bit) quantity loaded (data bus) is the number of bytes to be transferred during the data transfer that is being set up. The number is a positive binary quantity (bit 0 must be zero) and is decremented by the MTC after each memory transfer. A range of zero results in a subsequently issued Read Forward order to perform the equivalent of a Forward Space Block order transferring no data to memory. A range of zero results in a subsequently issued Write order setting the Operation Check bit of Status Word 1, no data transfer, no tape motion initiated, and termination of the order. Any address and range register residue is applied to the next command unless reset by another Output Range instruction.

Output Configuration Word (11)

Format:



[a]Reserved for future use must be zero.

Function:

Loads the Configuration Word for the device corresponding to the referenced channel.

- o Pack Mode – This bit set to one puts the NRZI 7-track subsystem in Pack Mode. In this mode, data transfers to and from memory start and end on byte boundaries and the range is specified in bytes. However, if the range contents specify an odd number of bytes, the Operation Check bit in Status Word 1 is set, no tape motion is initiated, and the read or write order is terminated. For a write command, the

7-track subsystem generates three characters with odd parity on tape for each word obtained from memory. The high-order twelve bits of the word are written on tape as characters 1 and 2. The remaining four bits of the word are recorded in the low-order locations of character 3 with the remaining two bit locations zero-filled. For a read command, a data word is transferred to memory for each three characters read from tape or until the range register has been depleted to zero.

NOTE: The NRZI 9-track subsystem ignores this bit.

- o 7-Track Even Parity Mode – This bit set to one puts the NRZI 7-track subsystem, if not in Pack mode, into even (vertical) parity mode. Even parity accompanies each data character written on tape and VRC logic checks for incorrect even parity during read and Read-After-Write operations. Status Word 2 bit 5 is set if errors are detected. This bit set to zero puts the subsystem into the normal odd parity mode.

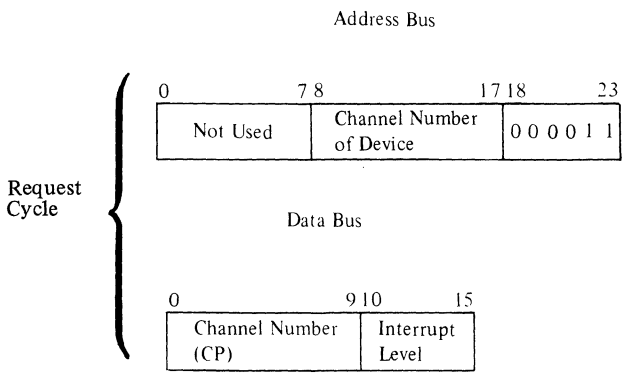
NOTE: The NRZI 9-track subsystem ignores this bit.

- o Inhibit ANSI Standard – This bit set to zero requires that the ANSI Standard, write data blocks of no less than 18 bytes from memory, be adhered to; otherwise an Operation Check occurs. Also, if during reading, 11 or less characters from tape are detected, they are interpreted as noise, Status Word 1 bit 4 is set, and the search continues for the next block on tape. This bit set to a one inhibits the ANSI Standard and writing a minimum data block of one or more characters is allowed. Also allowed is the reading of one or more character data blocks.
- o Set Low Read Threshold – This bit set to one allows the read threshold detection to be reduced during read operations only.
- o Set Diagnostic Mode – This bit set to one puts the 7- or 9-track subsystems into the Diagnostic mode. In this mode:
  - – Subsequently issued write orders generate even vertical parity on 9-track tape (for use in diagnosis; i.e., in the generation on tape of data and check characters with bad parity and blank characters and gaps of specific lengths).
  - – Subsequently issued write orders to 9-track devices generate incorrect CRC characters on tape (for use in diagnosis).
  - – Subsequently issued non-forward motion (Output Task Word) command transfers data stored in the write FIFO adapter buffer to the adapter DLI transmitter/receiver logic and wraps the data around and into the read FIFO adapter buffer.

This bit set to zero puts the NRZI tape adapter subsystem attached to the MTC into its Normal Mode (non-Diagnostic Mode).

Output Interrupt Control (03)

Format:

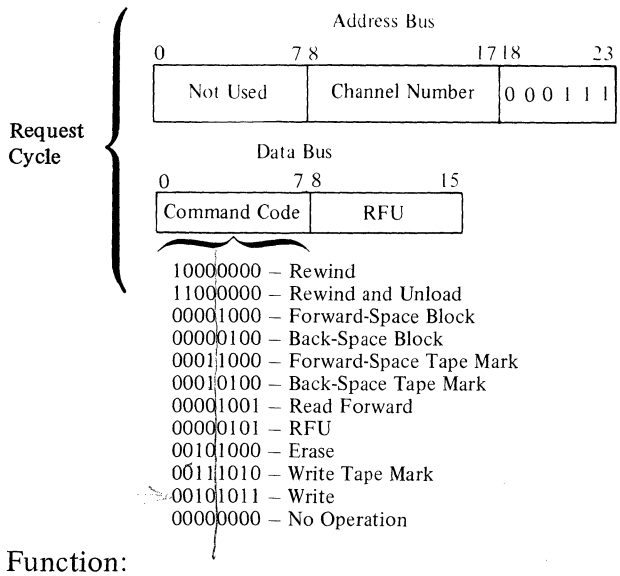


Function:

Loads, for the referenced device, the interrupt level and the channel number of the CP to which subsequent interrupts should be sent. The level number is a 6-bit quantity and is positioned on the data bus as illustrated. Bits 0-9 of the data bus contain the channel number of the CP loading the interrupt level. If an interrupt level of zero is loaded, the subsystem will not generate or save interrupts for any events that occur while the interrupt level is zero. For example, if the attention bit in Status Word 1 is set to one with a stored interrupt level of zero, the subsystem will not generate an interrupt on the bus. The interrupt level is set to zero whenever the subsystem is initialized.

Output Task Word (07)

Format:



Function:

Outputs a Task Word to the referenced channel. The coding bits 0-7, illustrated above, represent the operations that are to be performed. When this command is accepted, the channel enters the Busy state. All configuration, addresses and range information must be loaded prior to execution of this command. The direction of data transfer indicated by the low-order bit of the most recent Output Address command must agree with the direction of transfer (read or write) specified by command code of the Output Task Word. If it does not, Status Word 1 bit 11 Operation Check will be set and a normal termination of the command without data transfer and tape motion will result. These commands addressed to a device not in the online state result in the setting of an Operation Check bit prior to a normal termination of the order.

- Rewind – This command rewinds the tape to the BOT marker if the most recent Output Task command was not a "write order" (Write, Write Tape Mark, Erase). If otherwise, an erase in the forward direction over approximately one inch of tape precedes the rewind operation. The drive remains in the Busy state until the completion of the rewind operation or the setting of a Time Out check. The subsequent ten commands will be NAKed. If the tape on the drive is at BOT when this order is issued, tape motion is not initiated and a normal termination of the order results. Note that the rewinding of a drive via the Rewind Button on the drive does not put the device in the Busy state but activates rewinding (Status Word 2 bit 1), which affects the states of the Device Ready and Attention bits of Status Word 1. When the manually initiated rewind is complete, the Rewinding status condition resets, Device Ready changes state, and the Attention bit is set again.

Note: A rewind of tape without any erase operation may be initiated manually via the rewind switch on the transport.

- Rewind and Unload – This command implements a Rewind command and, following detection of the BOT marker, activates the drive's unload sequence prior to termination of the order. If the tape on the drive is at BOT when this command is issued, then only the unload sequence is initiated prior to termination of the command. The unload sequence puts the selected tape device into the offline state, extinguishes the online indicator and moves tape in the reverse direction until it is wound off the take-up reel. The tape drive can manually be placed into the online state only from the device control panel.

Note: A tape drive may be put in the offline state without any erase action or tape motion by manually switching the Online/Offline switch on the transport to the Offline position.

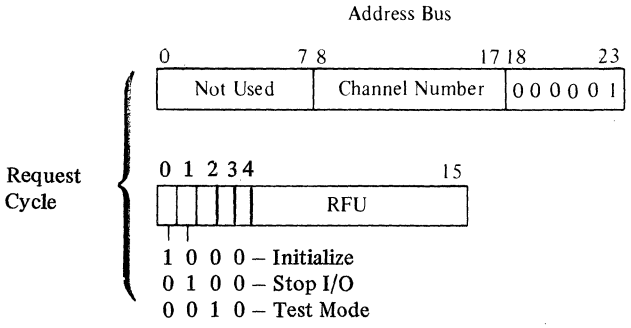
- Forward-Space Block – This command results in the drive spacing forward over the next block on tape if the most recent Output Task command was not a "write order." If otherwise, an erase in the forward direction over approximately one inch precedes the forward-space block operation. The order terminates when tape is positioned in the next interblock gap or as a result of a Time Out Check.
- Back-Space Block – This command results in the drive spacing back over the previous block on tape if the most recent Output Task command was not a "write order." If otherwise, an erase in the forward direction over approximately one inch precedes the back-space block operation. The order terminates when tape is positioned in the previous interblock gap, or when the tape is positioned at BOT, or as a result of a Time Out Check. If this command is issued when

the tape is positioned at BOT, tape motion is not initiated and the order is terminated.

- Forward-Space Tape Mark – This command results in the drive spacing forward over one or more blocks until a Tape Mark or EOT status is detected on tape if the most recent Output Task command was not a "write order." If otherwise, an erase in the forward direction over approximately one inch precedes the forward-space tape mark operation. The command terminates when tape is positioned in the interblock gap following the block containing a Tape Mark or a data block when EOT (Status Word 1 bit 7) is set or as a result of a Time Out Check.
- Back-Space Tape Mark – This command results in the drive spacing back over one or more blocks until a tape mark is detected on tape if the most recent Output Task command was not a "write order." If otherwise, an erase in the forward direction over approximately one inch precedes the back-space tape mark operation. The command terminates when the tape is positioned in the interblock gap preceding the block containing the tape mark or when the tape is positioned at BOT or as a result of a Time Out Check. If this order is issued when the tape is positioned at BOT, tape motion is not initiated and a normal termination of the order follows.
- Read Forward – This command results in the drive reading forward over the next block on tape if the most recent Output Task command was not a "write order." If otherwise, an erase in the forward direction over approximately one inch precedes the read forward operation. The order terminates when the tape is positioned in the next interblock gap or as a result of a Time Out Check. The format of the data transferred from tape to memory is a function of the stored Configuration Word. In addition to reading data, vertical parity, longitudinal parity, and the cyclic redundancy check character are read and integrity checks are made.
- Erase – This command results in the drive erasing tape in the forward direction producing a 2-inch gap on the tape, if the most recent Output Task command was a "write order." If otherwise, an erase in the forward direction over approximately two inches precedes the normal 2-inch erase operation. The channel of the device remains busy for the duration of the erase command and terminates normally.
- Write Tape Mark – This command results in the drive erasing tape in the forward direction producing a 2-inch gap on the tape followed by the recording of a Tape Mark block if the most recent Output Task command was a "write order." If otherwise, an erase in the forward direction over approximately two inches precedes the write tape mark operation (2-inch erase plus write tape mark block). The tape mark format for 9-track tape is shown in Figure 6-29. An attempt to write a Tape Mark Block on a tape unit in Write Protect results in no tape motion initiated or block written and the activation of the Operation Check bit of Status Word 1. The order terminates when the tape is positioned in the gap beyond the Tape Mark Block.
- Write – This command results in the drive writing, in the forward direction, a data block of the format, and equal to or greater than the minimum block length allowed, as specified by the Configuration Word most recently issued to this addressed channel if the most recent Output Task command was a "write order." If otherwise, an erase in the forward direction over approximately two inches precedes the write operation. In addition to writing data, vertical parity, longitudinal parity, and a cyclic redundancy check character (9-track only) are written in the data block and integrity checks are made. An attempt to write a data block of less than the minimum length or to write to a drive in Write Protect results in no data transfer, no tape motion initiated, and the activation of the Operation Check bit of Status Word 1. The order terminates when the tape is positioned in the gap beyond the last data block written.
- No Operation – This command or Output Task command code results in no data transfer, no tape motion initiated, the normal reset of Status Word bits upon reception of an Output Task command, and a normal termination of the command.

Output Control Word (01)

Format:



Function:

Loads a Control Word into the referenced channel. This command will be unconditionally accepted by the channel regardless of its Busy status.

- o Initialize – This command will cause the MTC to reset to the same state that it enters after power up or Master Clear. When an initialize command is received by the MTC all of its channels are initialized (regardless of which channel the command was received over).

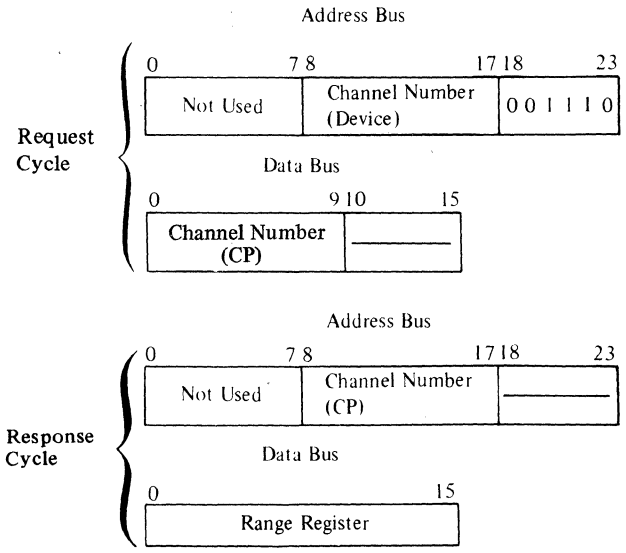
  Operations that are in progress in the MTC at the time of the Initialization will be abruptly terminated and all registers will be initialized, including control registers device select and control signals to the drives. No information about the terminated operations will be retained and no interrupts for the operations will be generated. The interrupt level for all channels will be set to Zero (interrupts blocked).
- o Stop I/O – This command causes any operation currently active on the specified channel to be abruptly terminated. If a data transfer operation is in progress, it will not be completed. An Interrupt will be generated for the operation terminated by this command as if the operation had come to a normal ending point. Status, Configuration, and Range information, present in the MTC when this command is received, will be retained.
- o Test Mode – When an Output Control Word command is received with the Test Mode bit on (while the MTC is in normal operating mode), the MTC will enter Test Mode. Once the MTC is in Test Mode, subsequent commands cause the contents of the Data Bus to be loaded into the MTC instruction register (the Data Bus contains the microinstruction). One clock cycle is then issued to execute the microinstruction loaded. This enables a sequence of microinstructions to be executed in a single step mode from a software test routine.

  Test Mode operates on an entire MTC and, therefore, precludes normal usage of other devices on the MTC at the same time. Normal mode is resumed when a "reset test mode" microinstruction is executed or when Master Clear is activated on the Bus. All function codes which are received over the Bus while the MTC is in the Test mode will cause the contents of the Data Bus to be loaded into the MTC instruction register as described above (function code field of the Address Bus is ignored). Note that Response cycles will not be generated for Input commands.

Input Commands

Input Range (OC)

Format:



Function:

Causes the current contents of the referenced channel's Range Register to be transferred to the requesting channel.

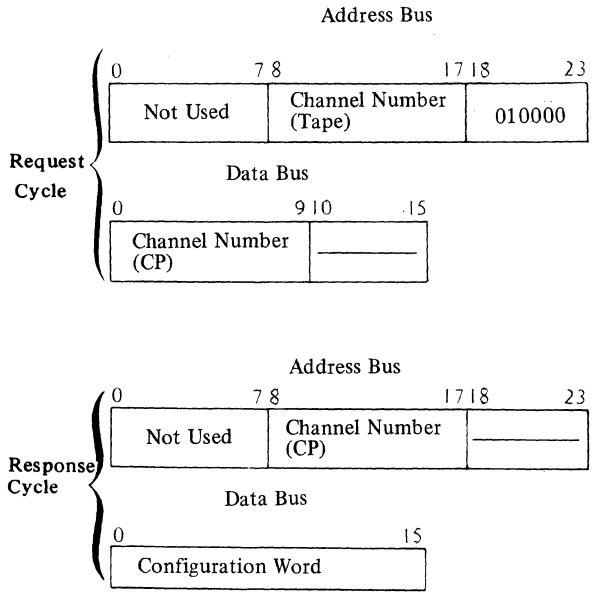
During the Response cycle (Second Half Read), the MTC will return in bits 8-23 of the Address Bus the same data that was received in bits 0-15 of the Data Bus during the Instruction Cycle.

After the completion of a read operation, the contents of the Range Register reflect the status of that transfer with respect to the physical block read.

- o If the contents are a positive value greater than zero and bit 8 of Status Word 1, Unequal Length Check is set to a logical One, the length of the physical block was less than the range.
- o If the contents are zero and bit 8 of Status Word 1 is equal to One, the length of the physical block was greater than the original range.
- o If the contents are zero and bit 8 of Status Word 1 is equal to Zero, the length of the physical block was equal to the original range.

Input Configuration Word (10)

Format:



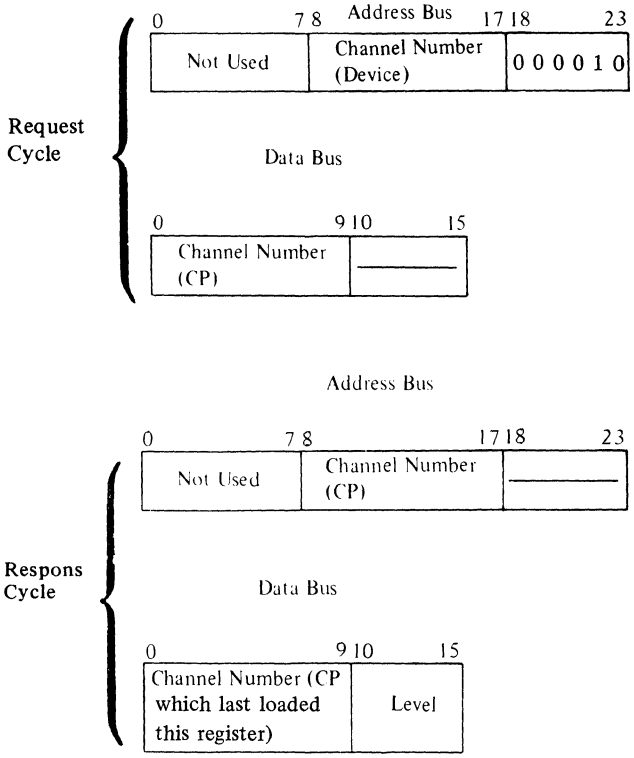
Function:

Causes the channel's Configuration Word to be transferred to the requesting channel.

During the Response cycle (Second Half Read) the MTC will return in bits 8-23 of the Address Bus the same data that was received in bits 0-15 of the Data Bus during the Instruction cycle. After the completion of a read operation, the contents of the Configuration Word register reflect the status of that transfer with respect to the physical block read.

Input Interrupt Control (02)

Format:



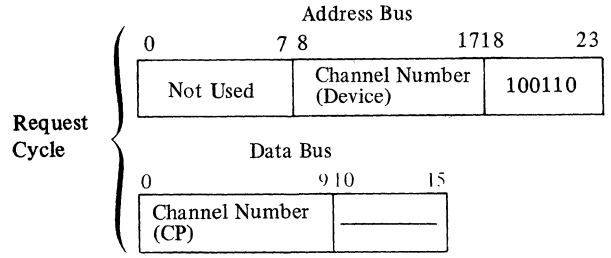
Function:

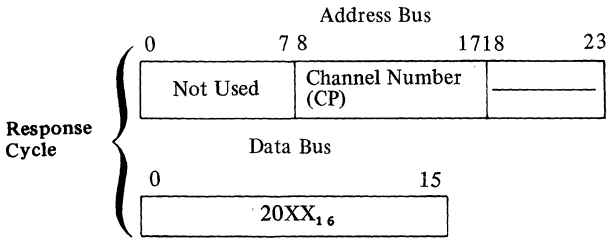
Causes the channel's interrupt level to be transferred to the requesting channel. The level value will be placed on Data Bus bits 10 through 15 (see above) with bit 15 as the least significant bit. This quantity is the value previously received in the Output Interrupt Control instruction or a default value of 00. The default value is the interrupt level assumed by the channel when initialized. Note that the channel number returned in bits 0-9 of the Data Bus might be different from the channel number of the CP executing this instruction if more than one CP is attached to the Bus.

During the Response cycle (Second Half Read), the MTC will return in bits 8-23 of the Address Bus the same data that was received in bits 0-15 of the Data Bus during the request cycle.

Input Device ID (26)

Format:





Function:

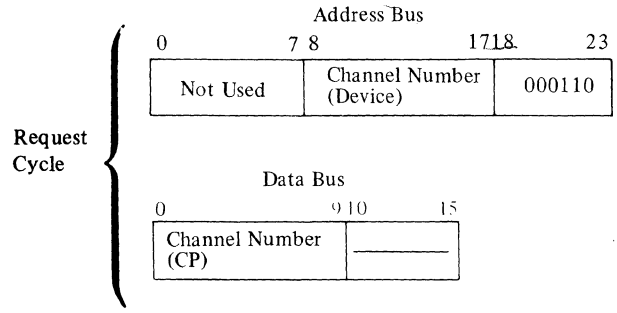
Causes the referenced channel to transfer its identification code to the requesting channel. The codes for each type of tape device are defined as follows:

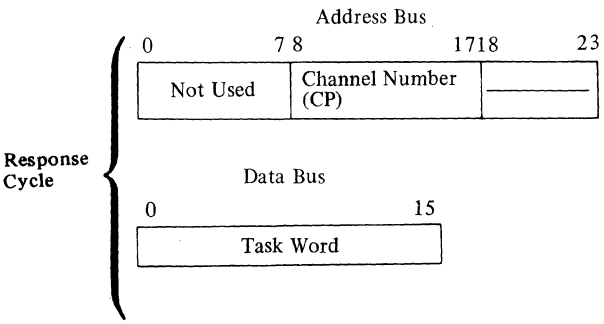
- Bits 0 - 7 = 20_{16} identifies MTC
- Bits 8 - 9 = 01 identifies magnetic tape subsystem
- Bit 10 = x identifies 7-track or 9-track tape
  - 0 9-track tape subsystem
  - 1 7-track tape subsystem
- Bit 11 - 13 = xxx identifies tape densities that the device (channel) can accommodate
  - For 9-track tape subsystems
    - 1 RFU
    - 1 RFU
    - 1 800 cpi
  - For 7-track tape subsystem
    - 1 800 cpi
    - 1 556 cpi
    - 1 RFU
- Bits 14 - 15 = xx identifies tape drive speed
  - 00 RFU
  - 01 45 ips
  - 10 75 ips
  - 11 RFU

During the Response cycle (Second Half Read), the MTC returns in bits 8-23 of the Address Bus, the same data that was received in bits 0-15 of the Data Bus during the Request cycle.

Input Task Word(06)

Format:





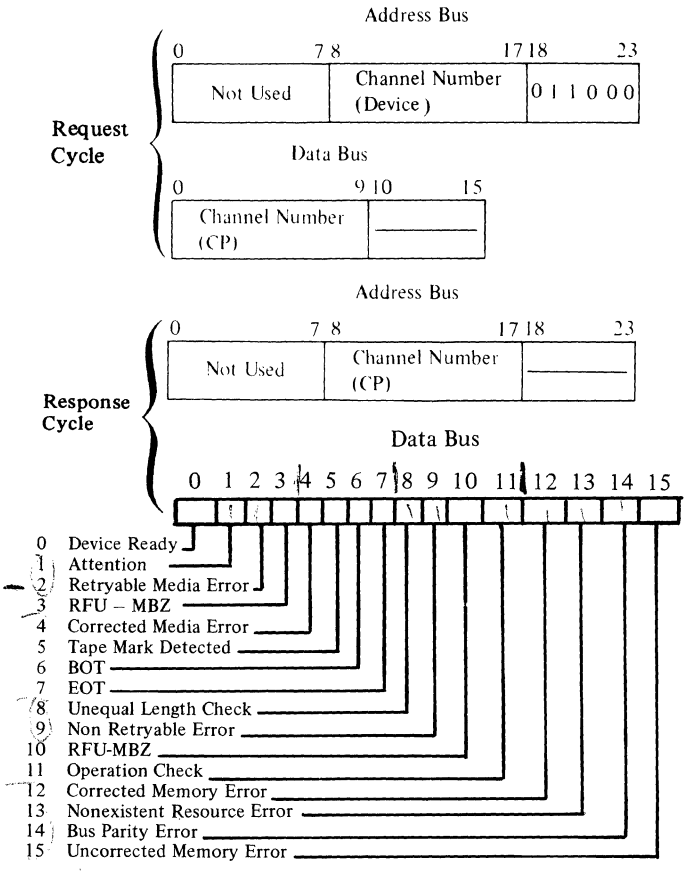
Function:

Causes the Task Word of the referenced channel to be transferred to the requesting channel. The Task Word transferred will contain the code for the last operation executed by the channel (unless an Initialize has occurred).

During the Response cycle (Second Half Read), the MTC will return in bits 8-23 of the Address Bus the same data that was received in bits 0-15 of the Data Bus during the Request cycle.

Input Status Word 1 (18)

Format:



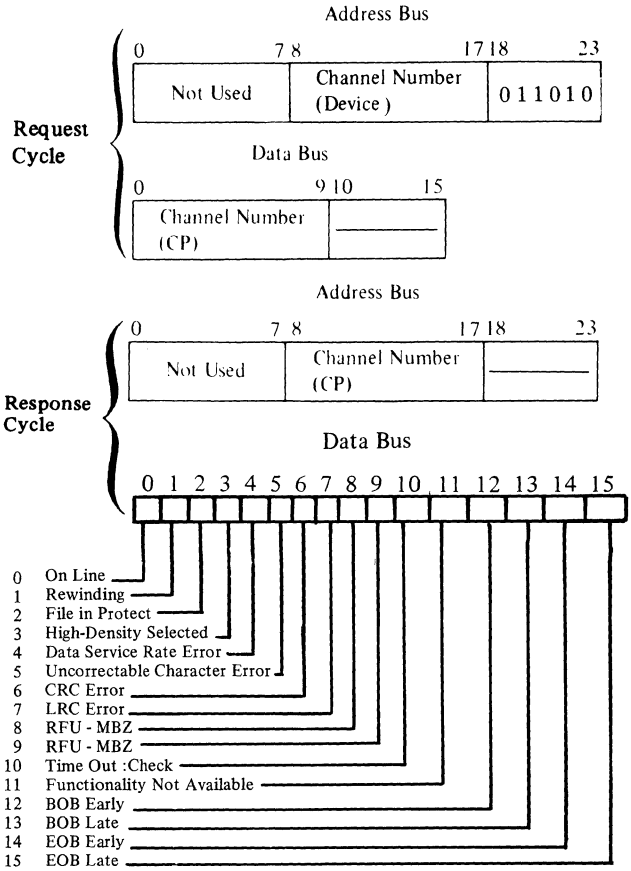
Function:

Causes the referenced channel's Status Word 1 to be transferred to the requesting channel.

During the Response cycle (Second Half Read), the MTC will return in bits 8-23 of the Address Bus the same data that was received in bits 0-15 of the Data Bus during the Request cycle. Bits 0-7 of the Address Bus and the parity bit associated with these bits are the same data received during the Request cycle. See Table 6-29.

Input Status Word 2 (18)

Format:



Function:

Causes the referenced channel's Status Word 2 to be transferred to the requesting channel.

During the Response cycle (Second Half Read), the MTC will return in bits 8-23 of the Address Bus the same data received in bits 0-15 of the Data Bus during the Request cycle. Bits 0-7 of the Address Bus and the parity bit associated with these bits are the same data received during the Request cycle. See Table 6-30.

TABLE 6-29. STATUS BIT DEFINITIONS – WORD 1

Status Condition	Bit	Definition	Reset by
Device Ready	0	Unit is online with tape loaded, is not rewinding, and no further manual intervention is required to place it under program control. This bit will be zero, if either Status Word 2, bit 0 is a zero or Status Word 2, bit 1 is a one.	A change in condition
Attention	1	Indicates an event has occurred at the unit which requires software action. This event, moreover, was not related to a current task, but rather was unsolicited. This bit will be set whenever the device changes its ready condition as a result of a non-software initiated command (i.e., enter or leave the online state, rewinding state, or media loaded state). Attention status may occur following a software initiated Stop I/O or initialize command if the device was performing a Rewind or Rewind and Unload Instruction. Whenever the Attention bit is set, an interrupt is attempted (if the interrupt level is nonzero). If a previously initiated operation is in progress when a device state change is sensed, the resultant interrupt (with the Attention bit set) will serve as notification of both the end of the operation and the device state change.	Initialize, Input Status Word 1, or Output Task Word command
Retryable Media Error	2	Indicates a data error has occurred and will be set whenever Status Word 2 bit 4, 5, 6 or 7 is active.	Initialize or Output Task Word command
Reserved for Future Use	3	Must be zero.	–
Corrected Media Error	4	Indicates an error condition was detected on tape; however, the data read is not lost. For this subsystem, the detected condition is a noise area within an interblock or BOT gap. A noise area may comprise 1 to 11 detectable frames of magnetic transitions on the media when the subsystem is functioning in the minimum data block mode. In the nonminimum data block mode, all detected frames of magnetic transitions are processed as a block.	Initialize or Output Task Word command
Tape Mark	5	Indicates a Tape Mark has been detected during the execution of a Write Tape Mark, Forward-Space Tape Mark, or a Back-space Tape Mark order. This status bit will also be active if the block encountered during execution of a Forward-space/ Back-space/Read block instruction is a Tape Mark.	Initialize or Output Task Word command
BOT (Beginning of Tape)	6	Indicates the BOT marker is positioned at the BOT sensor. A backspace or rewind order issued to a unit with tape at BOT will result in no tape motion initiated and a normal termination of the order.	
EOT (End of Tape)	7	Indicates the EOT marker is positioned at, or has passed beyond, the EOT sensor. This status bit will remain active until the EOT marker passes back over the sensor as a result of a Tape Backward Motion command (e.g., back-space, rewind).The state of this status bit has no effect on forward motion commands, except the Forward-Space to Tape Mark in which forward motion is terminated in the next interblock gap.	
Unequal Length Check	8	Indicates that for the previous Read operation, the physical block was either greater or less than the value in the range register, at the beginning of the Read operation. This bit, a 1 and a residue in the range register, indicates that a short block was transferred. This bit active and a range register contents of zero indicate that a long block was transferred.	Initialize or Output Task Word command

TABLE 6-29 (CONT). STATUS BIT DEFINITIONS – WORD 1

Status Condition	Bit	Definition	Reset by
Non-Retryable Error	9	Indicates that the position of media under the tape read/write and erase heads is unknown. This bit will be set when a write order RAW[a] failure occurs (i.e., the detection of magnetic transitions on tape before the start or following the completion of a recorded data block or the failure to detect magnetic transitions in the area where a data block is being written or the failure to detect an NRZI density identification area on tape when writing an NRZI tape). This bit will also be set when an erase order RAW[a] failure occurs (i.e., the detection of magnetic transitions in the area on tape being erased) or when, during a read order, a split block is detected. A split block is a data block in which its beginning and end positions cannot be guaranteed detectable because of a detected unrecorded area within the block. This status bit also becomes active when Status Word 2 bit 10 (Time Out Check) is set.	Initialize or Output Task Word command
Reserved for Future Use	10	Must be zero.	–
Operation Check	11	Indicates a write type order (Write, Write Tape Mark, Erase) was issued to a tape drive in Write Protect (see state of Status Word 2 bit 2); that upon acceptance of an Output Task Word data transfer command, the direction of data transfer is not the same as that specified by the direction bit of the channel number issued by the previous Output Address command; that upon acceptance of an Output Task Word data transfer command, the contents of the range register is zero (for Read or Write) or less than 18 (for write) when the subsystem is in the ANSI configuration mode; or that a command (other than No Operation) was issued to a channel on which the device is in the offline or Not Ready state.	Initialize or Output Task Word command
Corrected Memory Error	12	Indicates that during execution of the previous operation, main memory detected and corrected a memory read error. The data that was delivered to the MTC was assumed to be correct.	Initialize or Output Task Word command
Nonexistent Resource Error	13	Indicates the MTC attempted a Write or Read request bus cycle and received a NAK response. Occurrence of this condition does not cause a termination of the operation in progress; however, it can result in bad data being written on the tape.	Initialize, Input Status Word 1, or Output Task Word command
Bus Parity Error	14	Indicates the MTC detected a parity error on either byte if the Data Bus during any output bus cycle (i.e., odd function code), during a second-half memory read cycle or when a parity error is detected in bits 0-7 of the Address Bus during an Output Address command. Occurrence of this condition does not cause a termination of the operation in process; however, it can result in bad data being written on tape.	Initialize, or (error free) Input Status Word 1 command
Noncorrectable Memory Error	15	Indicates that during execution of the previous operation, the main memory detected a memory read error which the EDAC algorithm could not correct. The data that was delivered to the MTC was incorrect. Occurrence of this condition does not cause a termination of the operation in progress; however, it can result in bad data being written on tape.	Initialize or Output Task Word command

[a]Read After Write

TABLE 6-30. STATUS BIT DEFINITIONS – WORD 2

Status Condition	Bit	Definition	Reset by
Online	0	Indicates the unit is online to the subsystem via the ONLINE/ OFFLINE switch on the unit. The unit can also be put into offline status via the Rewind and Unload instruction.	–
Rewinding	1	Indicates the unit is processing a rewind operation either via a command issued by the subsystem or the REWIND switch on the unit.	–
File in Protect	2	Indicates the unit is in write protect (i.e., the write permit ring is not in position on the mounted file reel).	–
High Density Selected	3	Indicates that the device high/low density selector switch is set to the high position. For example, a 556/800 cpi 7-track NRZI tape device set to high density will set this indicator and read and write tape at 800 cpi. A 9-track NRZI tape device which processes tape at 800 cpi (only) will set this indicator to zero.	–
Data Service Rate Error	4	Indicates that during a Read or Write operation, data transfer to/from main memory and the unit via the MTC could not maintain the rate in demand. Either data was lost on input because of failure to keep up with device demands or data was unavailable on output when required by the device. The detection of this error condition does not affect the execution of the data transfer operation in process.	Initialize or Output Task Word command
Uncorrectable Character Error	5	Indicates that during a Read or Write operation either a Vertical Redundancy Check (VRC) error and/or a dropped character error was detected. In a VRC error, one or more data characters were detected with incorrect vertical parity. Data character parity is odd unless bit 3 in the stored configuration word is set. In a dropped character error, one or more data characters following the first contiguous segment of data characters in the block were not read.	Initialize or Output Task Word command
CRC (Cyclic Redundancy Check) Error	6	Indicates that during a Read or Write operation (9-track NRZI only), the tape CRC character failed to compare with the reconstructed value.	Initialize or Output Task Word command
LRC (Longitudinal Redundancy Check) Error	7	Indicates that during a Read or Write operation, an incorrect longitudinal parity was detected for any track read. The LRC character written at the end of each block will result in even track parity.	Initialize or Output Task Word command
Reserved for Future Use	8	Must be zero.	–
Reserved for Future Use	9	Must be zero.	–
Time Out Check	10	Indicates that during a Read or Space operation, an excessive delay prior to detecting a block has occurred and has resulted in a termination of the order. This delay is equivalent to passing over approximately 25 feet of tape. This bit is also set during a Rewind or Rewind and Unload operation following an excessive delay without the device entering a Rewind sequence or at BOT. Positioning of the tape under the read/write and erase heads is unknown. Activation of this bit during a Rewind and Unload operation indicates that an excessive delay has been detected without the device entering the offline state.	Initialize or Output Task Word command

TABLE 6-30 (CONT). STATUS BIT DEFINITIONS – WORD 2

Status Condition	Bit	Definition	Reset by
Functionality Not Available	11	Indicates for the subsystem specified herein that the Output Task Word – Read Backwards order is not available. A termination of the order without tape motion takes place.	Initialize or Output Task Word command
Beginning of Block (BOB) Early	12	Indicates that the block written on tape was detected by the RAW circuitry to have begun earlier than that specified for the selected device.	–
Beginning of Block (BOB) Late	13	Indicates that the block written on tape was detected by the RAW circuitry to have begun later than that specified for the selected device.	–
End of Block (EOB) Early	14	Indicates that the block written on tape was detected by the RAW circuitry to have terminated earlier than that specified for the selected device.	–
End of Block (EOB) Late	15	Indicates that the block written on tape was detected by the RAW circuitry to have terminated later than that specified for the selected device.	–

SECTION 7

SOFTWARE

Level 6 offers three operating systems:

- o GCOS/BES1
- o GCOS/BES2
- o GCOS 6/MDT

GCOS/BES1

GCOS/BES1 performs the chief system functions, enabling the user to concentrate on applications. Figure 7-1 provides an overview of the various modules, languages, programs, and development tools.

Program Development Tools

The program development tools operate in an offline environment and include: a Command Processor which interacts with the Loader to bring the other system modules into memory, two language processors (a FORTRAN Compiler, an Assembler for the Level 6 Assembly Language), an Editor for correcting the source text of programs written in either language, a Linker which converts object modules from the Assembler or the Compiler into an executable form, a Macro Preprocessor that provides for definition of macro calls/routines and expansion of macro calls within an assembly language source program, and a Cross-Reference Program, which is a utility that relates the symbolic tags of an assembly program to the listing line numbers where they appear. Figure 7-2 summarizes the sequence of events that might take place during the development of an application program.

Command Processor

The Command Processor loads all system and application programs via console commands and initiates the following functions:

- o Places information in the Loader communication block.
- o Builds an Attach Table.
- o Builds an Argument List.
- o Sets addresses in the appropriate registers for the Argument List and the program's return address.

The Command Processor responds to these console commmands by providing the Loader with the precise directions necessary to load the

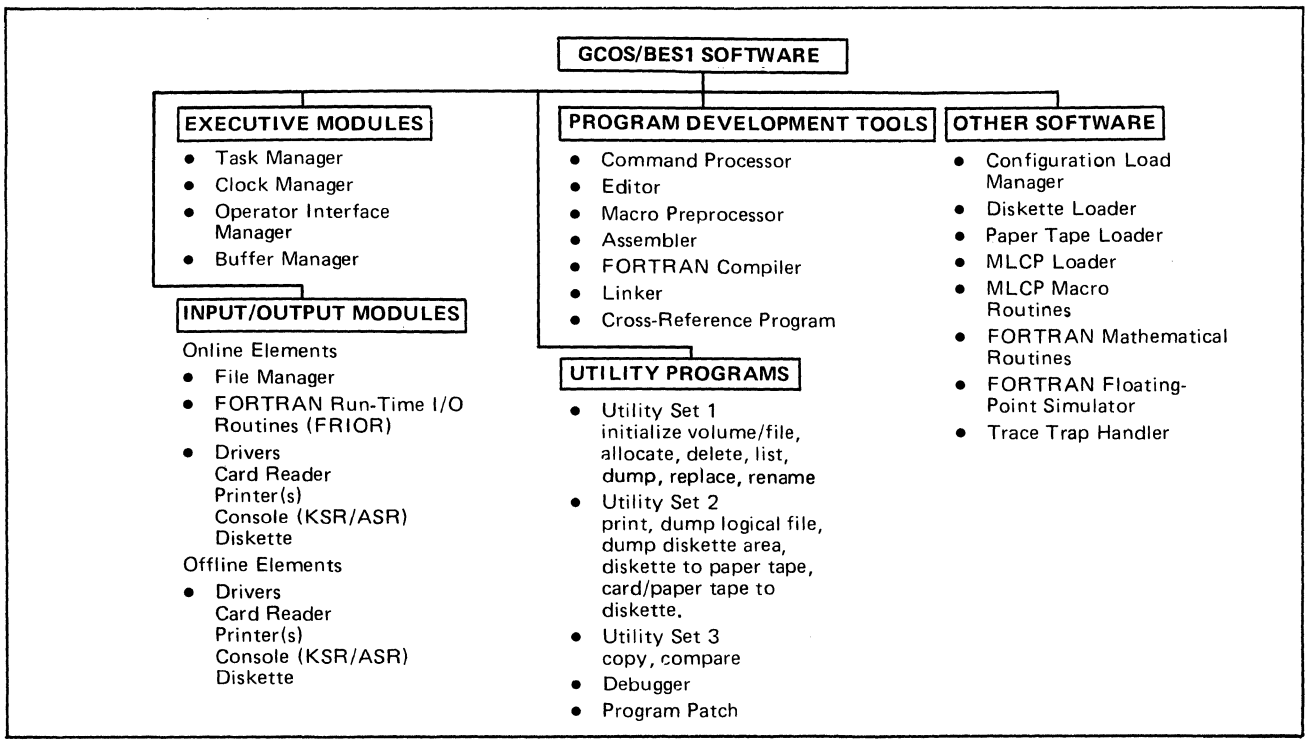


Figure 7-1. Software Overview

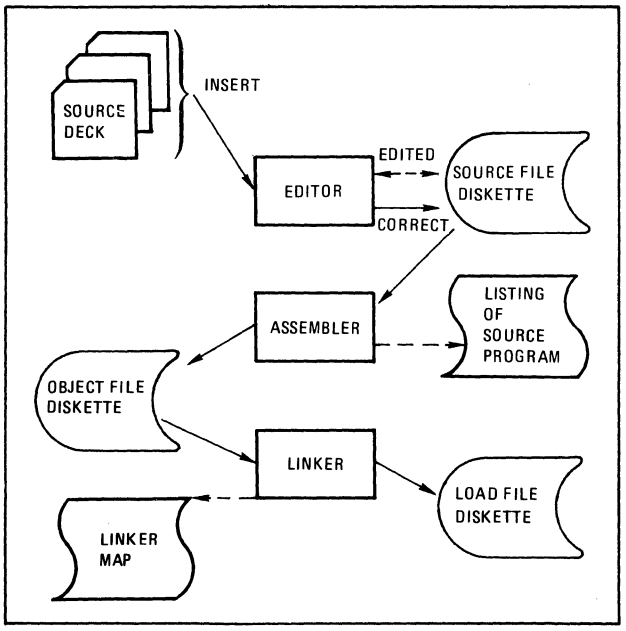


Figure 7-2. Program Development Sequence

requested system or application program. The five commands are:

- Load Initialize – specifies a relocation factor for the program being loaded and provides a post-load option of either halt or execute.
- Attach – provides the name, logical file number, symbolic device name, and channel number for each file used by the program to be loaded. One Attach command is necessary for each file, including the file with the program to be loaded. The Command Processor uses the information from this statement to build the Attach Table.
- Detach – deletes entries from the Attach Table, initializes all except the standard system entries, and "repacks" the table by altering the table displacement in the first word of the Attach Table.
- Load – provides the member name of the load module to be loaded and whatever arguments are needed by the module. The Command Processor uses the information from this command to build the Argument List and sets a bit in the Loader indicator word of the Loader communication block to indicate that there are arguments to be passed.
- Print Attachments – causes a typeout of all or specified current entries in the Attach Table. If the command input device is the system console or another KSR-like device, the typeout occurs on that device. If the command input device is a card reader or disk device, the typeout occurs on the system console.

The Command Processor also places the address of the control word of the Argument List into register B7 and sets a value in register B5 for the return address.

Editor

The Editor is a system program that enables correction of the source text of application programs written in either the Level 6 Assembly Language or FORTRAN. Editor accepts its source file input from diskette and writes the corrected source program out to diskette. The command file that directs Editor's operation can come from a KSR, paper tape on an ASR, or a card reader.

If the source program to be corrected is not already on diskette, the file can be created by using the INSERT command, followed by the text for the file and the appropriate commands to terminate the insert and close the file. The *Program Development Tools* manual details all the Editor commands for locating, substituting, deleting, and inserting statements in source programs.

Once written and edited, an application program must be assembled or compiled using one of the system programs.

Macro Preprocessor

The Macro Preprocessor (MPP) is a system program that provides for:

- Definition of macro routines
- Definition of macro calls
- Expansion of macro calls within an assembly language source program

A macro call is an abbreviation of a sequence of assembly language instructions (the macro routine) which, when encountered by the MPP within a source program, causes the inclusion of the macro routine in the source program at the point where the macro call occurred. Macro calls may be nested (occur within a macro routine) or recursive (a nested call that requests the routine in which it appears).

The MPP can process macro routines which appear at the beginning of the source program in which the related macro calls are being expanded, or it can retrieve macro routines from the library files in which they have been placed by the action of the Editor or Utility Set 2.

The MPP operates in a minimum of 8K words of memory, but can use up to 64K words, if

available. It requires at least one diskette which may be used for both input and output, and a KSR; it can use three input diskettes as well as an output diskette if these are available.

Assembler

The Level 6 Assembler is a system program that processes Assembly Language source code, translates these statements into object code acceptable to the Linker, and produces a listing of the source statement and their associated machine code equivalents, suitably flagged for error conditions.

The Assembler is a two-pass program that runs in a minimum configuration of 8K words. The first pass generates the symbol table in the Assembler's resident table area; the second pass generates the object module and a listing if requested.

The Assembler accepts the input commands that control its operation from a KSR; it accepts source statement input from disk. The Assembler output, both the object module and the list, can be written to diskette, as well as assigned directly to a printer.

FORTRAN Compiler

The FORTRAN Compiler translates source statements written in that language into either an object module ready for processing by the Linker or a file of assembly language instructions for further processing by the Level 6 Assembler. The compiler is a single-pass processor that operates in 16K words of memory.

The compiler is invoked by a command entered through the console; it accepts its source input from diskette and produces its object module or assembly language source module on diskette. The Compiler also produces a listing of the source text with embedded diagnostics and a memory map which is directed either to a diskette (for printing later) or to a printer.

Two sets of routines are available for use with FORTRAN programs: basic external functions and intrinsic functions. The basic external functions are frequently used mathematical routines; the intrinsic functions are a set of fundamental operations that are difficult to express in FORTRAN statements and, hence, are written in Assembly Language for use with FORTRAN programs.

Linker

The Linker converts object modules that are the output of the compiler or the Assembler to a format acceptable to the diskette loader. It resolves external references and can process one or more object modules to produce a single load module or several load modules in one execution.

The Linker is controlled by command statements that are entered through a console. The Linker input and output files must be on diskette. Linker also produces listings containing a link map and error messages which can be output to a console printer, line printer, or spooled to diskette for later printing.

Cross-Reference Program

A Cross-Reference Program, provided for use with Assembly Language source programs, reads the source statements and produces an alphabetized listing of cross-references (i.e., symbol definitions and where they are used) for the program. This Cross-Reference Program keeps a tally of the number of times each symbolic tag is used, and which statements the symbol occurs in so that when a problem arises, all corrections to a particular entity can be made at one time.

Utility Programs

The system programs designated as utilities provide a variety of services for volume preparation and maintenance, file handling, and data transfer from one type of medium to another, as well as program debugging and patching of either object or load modules on diskette files. These utility programs only execute in the offline environment and require the Command Processor. The following descriptions are very brief, for complete details see the *GCOS/BES Utility Programs* manual.

Utility Set 1

Utility Set 1 performs diskette volume and file preparation and maintenance. One of its functions is to initialize new diskette volumes for use with the diskette drive and to allow the volume to be processed by the system file handling routines. Initialization consists of writing the bootstrap record, the loader, and volume directory information on track zero of cylinder zero of the volume.

Other functions include:

- Allocating space for new files
- Deleting files or members
- Initializing partitioned files
- Listing contents of volume directories and member names of partitioned files
- Dumping information from a diskette to a memory area or from memory to diskette or to a console, line, or serial printer
- Replacing one memory value with another
- Renaming a volume or a file

Utility Set 2

Utility Set 2 provides the following data transfer functions:

- o Printing a file or member
- o Dumping a logical file or member
- o Dumping a physical diskette
- o Transferring file or member from diskette to paper tape
- o Transferring card or paper tape file to diskette

Utility Set 3

Utility Set 3 provides for copying and verifying diskette data, including:

- o Copying a volume, file, or member from one diskette to another
- o Comparing the new volume or file (but not a partitioned file or member) with the original to verify the accuracy of the copy operation

Debugger

Debugger is an interactive utility used for program testing and error correction. The console dialog consists of commands submitted by the operator and responses displayed on the console printer in the form of informational and error messages. Debugger can display and modify memory addresses and register contents. The "breakpoint" features of the utility causes the automatic activation of Debugger when a specified operation code occurs in the program being executed. When the utility terminates, it automatically transfers control back to the executing program.

Program Patch

The Program Patch utility allows the alteration of either object or load module text. Patches can be created, added, deleted, and listed by using the appropriate commands to the utility.

Executive Modules

The Executive software is a basic set of support facilities enabling development of application programs around such system-supplied features as clock-initiated tasks, automatic task scheduling and dispatching based on a flexible priority/interrupt structure, and the trapping of certain conditions arising out of program execution, without adding to software overhead. Figure 7-3 shows the interrelationships of the Executive modules.

Task Manager

This Executive component, supported by firmware and hardware functions, schedules, dispatches, and synchronizes tasks. To maintain the various task states shown in Figure 7-4, a number of data structures are defined when the online environment is configured. These structures are:

- o Priority level activity indicators
- o Interrupt vectors
- o Interrupt save area (ISA)
- o Request queues

Priority Level Activity Indicators

Each of the 64 priority levels is associated with an activity indicator bit in a 4-word contiguous area in a dedicated main memory location. Each bit in the 4-word area indicates whether or not its respective priority level is active. Bit 0 of the first word indicates activity of a priority level zero task when this bit is on (equal to 1); bit 15 of the fourth word indicates activity of a priority level 63 (system idle) task. When the level is inactive, its respective bit is equal to zero.

Interrupt Vectors

Interrupt vectors are pointers to priority level specific entries in the interrupt save area (ISA). Each vector is one word long and contains either zeros, if no task is using that particular priority level, or the address of the location of the ISA related to that task's level. There is one vector for each of the 64 priority levels and the 64-word area is in a dedicated main memory location.

Interrupt Save Area

This structure, residing in a nondedicated area of main memory, is where the context of an interrupted task is saved, and from which that context is restored when the task is restarted. A task's context consists of its registers and other necessary information for restarting the task. It is possible to select the number of registers that are saved/restored. There is one ISA for each priority level in use except for level 3, the inhibit level. (See Figure 7-5.)

Request Queues

These structures coordinate requests for the execution of tasks. The request queues, one for each priority level, consist of header tables and request blocks threaded together in a forward direction. (See Figure 7-5.)

The Request Queue header tables contain pointers to the request blocks for each priority level. The Start-Of-Queue (SOQ) header table contains pointers to the first request block associated with each priority level. Each request block in turn points to successive request blocks for that level's task, until a block is reached that

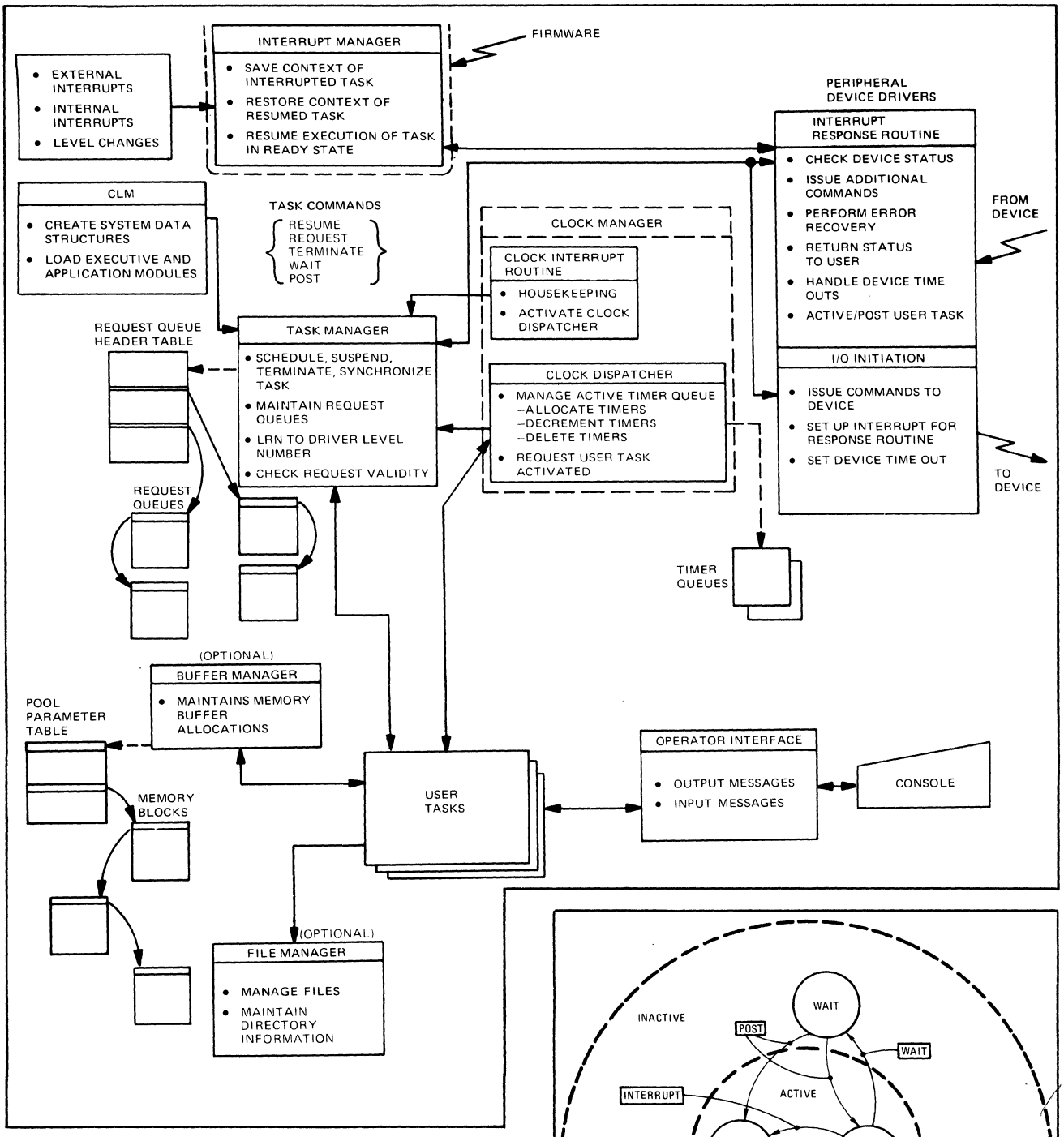


Figure 7-3. Interrelationships of the Executive Modules

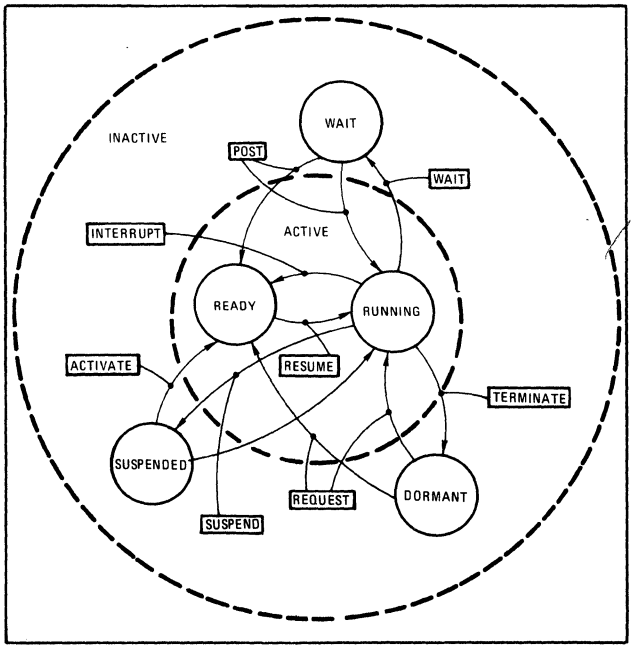


Figure 7-4. Task States

contains zero in the "next entry" pointer, indicating that the last block in the queue has been reached. The End-Of-Queue (EOQ) header table points directly to the last block in the queue.

The request blocks themselves are variable length areas of contiguous main memory. They contain information for task management such as the task's level and status, and other data in the form of temporary storage or values needed by the task.

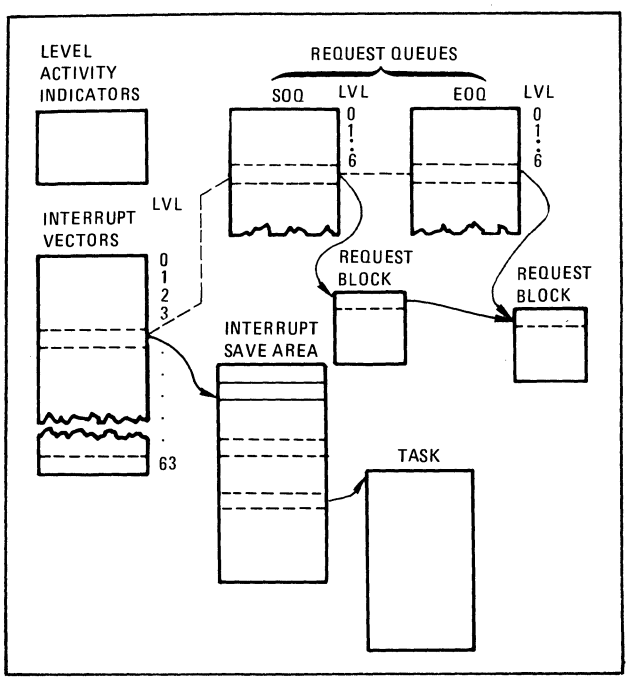


Figure 7-5. Task Management Data Structures

Management of Task States

The task states shown in Figure 7-4 are managed by interactions of software routines with hardware and firmware functions. The hardware in task management saves and restores task contexts.

Firmware functions that alter task states are:

- o Activate – mark the level as active (followed by Resume).
- o Suspend – mark the current level as inactive (followed by Resume).
- o Resume – examine the level activity indicators for the highest active one and start task.
- o Inhibit – mark a dedicated, high priority level as active and immediately start a task on this level. The running task continues on this level with no context Save or Restore.
- o Enable – return a task to its normal level from the inhibit level.

The software functions that alter task states are:

- o Request – supports the capability to request the activation of a task.
- o Wait – provides for a running task to be put in the wait state to allow the completion of another task that was called by the running task.
- o Post – provides for the reactivation of a waiting task, and the termination of the task being waited for.
- o Terminate – ends a running task and unlinks the associated request block.

Clock Manager

This Executive component controls the connecting, disconnecting, and processing of the clock timer blocks and the system control structure used for the timing of tasks by the Executive software.

The clock timer blocks are defined by the user according to application requirements and then are maintained by the Clock Manager and used to activate tasks at given priority levels after an elapsed time interval or at some regular time interval.

The Clock Manager subroutines provide the following services:

- o Connect the clock timer blocks.
- o Disconnect the clock timer blocks.
- o Convert the time of day to ASCII format.
- o Convert a millisecond value to a clock-compatible value.
- o Convert the data to ASCII format.

Each subroutine provides a return status indication when it finishes execution.

Operator Interface Manager

The Operator Interface Manager controls all operator dialog with software in an online environment. It presupposes a keyboard-send-receive (KSR) device that not only prints out messages to the operator, but also accepts information from a keyboard.

The component uses three kinds of messages:

- o Information message – consists of an automatic line feed, message text, and automatic carriage return.
- o Prompter message – solicits an operator response, and consists of a line feed, text, and question mark.
- o Reply message – enables operator response to a request for information or other specific input.

The component uses three kinds of messages:

Buffer Manager

This Executive component manages the system and user-requested memory space. The memory space used for buffer areas is defined when the online environement is configured. (See also "Configuration Load Manager" later in this section.)

When the Buffer Manager receives a request for buffer space, it searches the predefined control structures and related buffer pools for the appropriate space. Even if a request is made for a block size that is not specifically defined, the request can be filled with a block of the next larger defined size.

When a block of the appropriate size is found, the Buffer manager then supplies the requesting program with the address of the block along with a return status indicating a normal operation. If the request cannot be honored, the returned status either indicates invalid size, if the requested buffer was larger than the largest defined buffer block, or that the pool of the requested size is empty (i.e., no blocks are available).

After a program has finished using a buffer area, it can indicate that it no longer needs the space, and the Buffer Manager reconnects the area to the pool, thus making it available for use again.

Input/Output Modules

The software components that provide data input and output services consist of several groups of modules; some are provided for online environments exclusively, others execute in an offline environment.

The modules that provide online input and output services are a File Manager, a set of FORTRAN Run-Time Input/Output Routines (FRIOR), and a set of device drivers. The offline input/output services are provided by a set of device drivers.

File Manager

The File Manager consists of a set of service routines that provide access to volumes and files in an online environment. Among the routines provided by the File Manager are those to open, read, write, and close files, as well as routines to supply file status information to position files. The File Manager also returns error information for the various functions it performs.

FORTRAN Run-Time I/O Routines (FRIOR)

These routines provide for data transfer, device implementation, and the processing of FORTRAN format statements; since these routines require the services of the File Manager, they are used only with programs that are executed in an online environment. These routines provide for the reading and writing of formatted and unformatted records and they contain data conversion routines to edit integer, real, logical, character, and Hollerith data for formatted input and output. The FRIOR produce diagnostic messages to inform the user of inappropriate or inconsistent commands.

Device Drivers

These device-specific software components perform all data transfers between system and application programs and their respective devices. There are drivers for all Honeywell-supplied input and output devices operating in online environements.

The drivers are linked with the Executive modules and receive requests for service from the Task Manager. Online device drivers run at the priority level of the requested device.

Line and Serial Printer Driver

This component drives both line and serial printers. Its functions are:

- o Initiating output on individual printers
- o Reporting errors and status information
- o Timing to detect a device failure
- o Replacing tab characters with spaces

Card Reader Driver

This component drives card readers. Its functions are:

- o Initiating input on individual devices
- o Suspending itself to wait for interrupt from device
- o Checking device status
- o Timing to detect a device failure
- o Reporting device errors and status

Diskette Drivers

This component manages data transfers to and from diskettes. The driver is interrupt driven and requires the Task Manager. The driver must be permanently resident; it does not verify the volume identification. (It is the user's responsibility to ensure that the correct volume is mounted.) Diskette driver functions perform the following:

- o Manage all standard data transfers
- o Provide callable functions for reading and writing data

The driver performs neither diagnostic read/write nor write-deleted-data functions; all formatting must be done offline, using the utilities.

Console (KSR/ASR) Drivers

Teletype device drivers that manage data transfer exist as separate, device-specific modules. The user links the particular module(s) for the device to be used.

The driver for the KSR device provides keyboard and printer functions. The driver for the

automatic-send-receive (ASR) device consists of the KSR driver linked to modules that provide paper tape read and punch functions.

The online drivers (KSR and ASR) respond to requests from the Task Manager and run at the priority level assigned to the device.

Briefly, the functions of the teletype drivers are to:

- o Initiate I/O on individual functional units
- o Report errors and status information
- o Detect device failure/inactivity during input
- o Report use of Break key to operator interface or user-supplied routine (online)

In addition, the ASR drivers provide these functions:

- o Provide "escape" control characters for paper tape
- o Perform checksum calculation and verification for paper tape
- o Provide format and format checking for paper tape
- o Support paper tape punch in automatic mode

Other Software

Configuration Load Manager

This component builds and initializes the online system and starts execution. The Configuration Load Manager (CLM) uses information supplied to it to define system characteristics and build some of the data structures that the Executive software uses to control the processing of tasks.

CLM accepts commands that set up data structures for the Task Manager, devices, and trap handler, as well as the clock variables and a list of load modules to be included in the complete system. The CLM performs in two phases: the configuration phase and the loading phase. During configuration, CLM creates the system data structures and stores them in main memory.

During the loading phase, the various Executive and application modules are brought into memory. Each module contains permanently resident code and some temporary code for initialization of the module. The temporary code is overlaid by the permanent code of the next loaded module.

During this phase, defined symbols are added to the symbol table and references are resolved. Once a load module is in memory, control is transferred to its initialization code; whatever initialization is required by the module is performed. Control is then given back to the Loader to begin loading of the next module. This process continues until all modules are loaded, at which time, control is given to the highest priority Executive module and execution begins. See Figure 7-6 for memory layouts during the two phases.

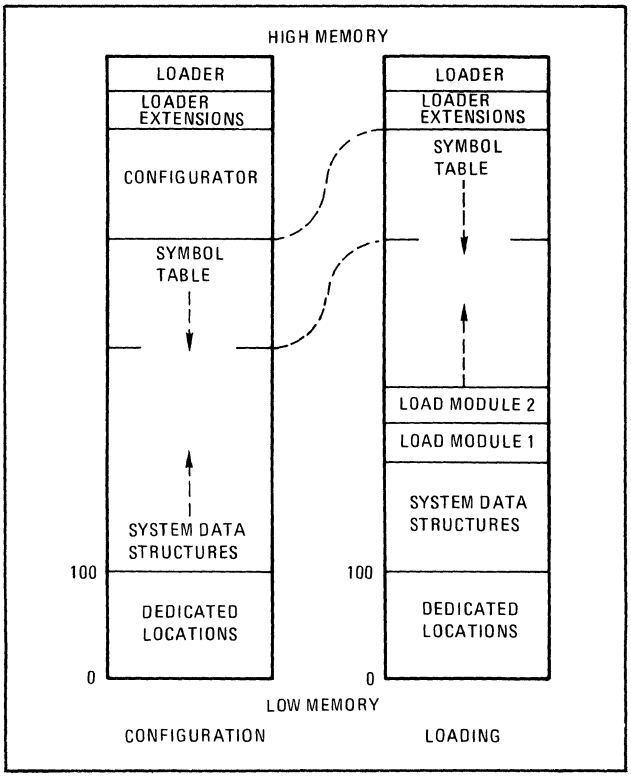


Figure 7-6. Memory Layouts During Operation of CLM

Loaders

Loaders are device-specific software components that load executable modules into memory from their respective devices and turn control over to the module. Loaders are available for bringing modules into memory from diskette as well as paper tape.

Diskette Loader

The Diskette Loader resides on the system load module diskette. It is brought into memory by a firmware bootstrap routine that causes the loader to read into memory and turns control over to the loader itself. The Diskette Loader includes four routines: loader initialization, Command Processor calling sequence, main loader, and miniloader.

At the entry point of the loader initialization routine, control is returned to the loader by the bootstrap routine. When the loader initialization routine finishes processing, the area it has occupied becomes available for use as the loader

buffer area. The functions of the loader initialization routine are to:

- o Save the bootstrap address for the miniloader.
- o Store the HALT address in the loader communication area.
- o Set values for the diskette loader and the processor type in the loader indicator word.
- o Load the trap handling routine that initializes the hardware-dedicated location.
- o Transfer control to the Command Processor calling sequence.

The Command Processor calling sequence primes the load parameters in the loader communication area to load the Command Processor. Its functions are to:

- o Initialize the relocation factor to zero.
- o Initialize the program loading channel to the bootstrap channel.
- o Set the file and member names for the Command Processor.
- o Transfer control to the main loader.

The main loader routine searches for and loads the requested member. It reads the member, a sector at a time; analyzes the item type of the sector; and relocates and distributes the code. The main loader also resolves backpatch chains and relocates global addresses depending on the item type. When the loader finishes processing, it turns control over to the designated entry point of the loaded module.

The miniloader routine, which at some times during processing may be the only resident part of the loader, is the basis of the software bootstrap routine. The miniloader contains:

- o A basic diskette I/O routine.
- o Loader communication area (displacement of loader entry points, Attach Table reference point from high memory, loader parameters, and halt addresses).

The miniloader reloads the boot loader at the bootstrap address, which in turn loads the diskette loader.

Paper Tape Loader

The paper tape loader resides on paper tape and is brought into memory by a firmware bootstrap routine that causes the loader to be read into memory and turns control over to the loader itself. The paper tape loader consists of two routines: the loader initialization routine and the main loader.

The loader initialization routine is entered from the bootstrap routine. After it has executed, the area the initialization routine has occupied is available as a buffer area for the main loader. The loader initialization routine has these functions:

- o Save the bootstrap channel number.
- o Store the halt address for conditions requiring operator intervention.
- o Load the trap handler module that initializes the dedicated hardware locations.
- o Branch to the entry point of the main loader when all functions are complete.

The main loader routine loads the requested load module from paper tape into memory. The main loader reads the load text, a block at a time; analyzes the item type; distributes the code; relocates addresses; and resolves backpatch chains. When the loading is completed, the main loader turns control over to the loaded program.

FORTRAN Mathematical Routines

Level 6 software includes a large set of FORTRAN mathematical routines. These intrinsic functions are available in object module format so they can be linked on an as-needed basis to perform a variety of calculations on behalf of a FORTRAN program. Some of the calculations performed by these routines are:

- o Converting to and from integer and real values
- o Truncating
- o Determining nearest whole number
- o Transferring a sign
- o Choosing the largest/smallest value
- o Finding the length of an entity, the square root, the natural logarithm, the common logarithm
- o Computing selected plane and spherical trigonometric functions

See the *GCOS/BES FORTRAN* manual for details about these routines.

Trap Handling

A trap is a conditional jump made to a predefined location in response to some event that occurs during program execution. Unlike interrupts, which are responses to events that are either unrelated to, or at least asynchronous with, the currently running program, trap conditions are caused by the running program. Series 60 (Level 6) hardware can recognize and respond to

a variety of trap conditions that may arise during program execution.

When a trap condition occurs, hardware/firmware stores relevant information in a trap save area and then examines the appropriate trap vector (a specific memory location) to ascertain whether it points to a dedicated software routine (called a trap handler), which can respond to the trap condition. If no such trap handler is present, the central processor halts. If, conversely, a trap handler is present, the trap handler is invoked; the trap handler processes the trap condition and returns control to the program whose execution caused the trap. The invocation and operation of the trap handler are invisible to the program that caused the trap.

Trap handlers can be user-written or GCOS/BES-provided. The latter category comprises two trap handlers: the Trace Trap Handler and the FORTRAN Floating-Point Simulator.

Trace Trap Handler

The Trace Trap Handler maintains a history of specific system information such as the contents of the program counter (P register), the contents of the system status register (S register), the contents of a prespecified memory location, and the contents of selected data and address registers. The information is gathered each time a BRK (Breakpoint) instruction is executed and each time a Branch or Jump instruction is executed while bit 0 of the mode control register (M1) is set to 1. A register save mask is used to indicate the registers whose contents are to be saved by the Trace Trap Handler.

FORTRAN Floating-Point Simulator

The FORTRAN Floating-Point Simulator permits execution of compiler-generated scientific instructions on machines that do not have the hardware scientific option installed. (See the *GCOS/BES Executive and Input/Output* manual for details.)

GCOS/BES2

GCOS/BES2 includes all the capabilities of GCOS/BES1 with significant extensions in communications, executive functions, and program development tools. Among the new communications capabilities are support of a BSC link to a Host Computer and an operator interface. Added executive functions include a disk-based system for loading/activating tasks resident on either diskette or cartridge disk. And the program development enhancements encompass extensions for both the Assembler and the FORTRAN Compiler. There are still other capabilities in such areas as file management, utilities, job stream management, and teleprinter/console display support.

Program Development Job Stream

The program development system can operate in either an interactive (single user) or a batch mode. Batch mode operation is a new feature added to BES2 that allows program development jobs to be predefined and stored in a file on cartridge disk, diskette, or cards. The Command Processor executes the job by reading that file, thus eliminating operator intervention except where it is desired, or required, to mount volumes, etc.

COBOL Compiler

The COBOL Compiler translates COBOL source statements into an object module ready for processing by the Linker. The compiler is a multipass processor that operates in a minimum of 16K words of memory.

The compiler is invoked by a command entered through a console. It accepts its source input from disk and produces its object module on disk. It also produces a listing of the source text (with embedded diagnostics and a memory map) that is directed either to a disk for printing later, or to a printer.

Additionally, COBOL offers support of relative files and random access capability, the CALL statement, and debug lines.

A description of the COBOL language statements is contained in the *COBOL* manual.

BASIC Interpreter

The BASIC Interpreter provides interactive facilities to create, modify, store, retrieve, and execute programs written in the BASIC language. An important part of the Interpreter's program execution function is its ability to process disk data files that have been prepared by either BES BASIC or BES FORTRAN programs. It operates in 16K words of memory.

By fully utilizing the program and file processing facilities available in the BASIC Interpreter, most of the more elementary data processing and problem-solving requirements and applications can operate within a self-contained BASIC-oriented environment. Both program libraries and data files can be created and maintained on disk for this purpose.

The Interpreter is invoked by a command entered through a console; input is from a console or disk. Diagnostics and execution results are displayed on the console.

A description of BASIC program preparation, execution, and language statements is contained in the *BASIC* manual.

FORTRAN Enhancements

The FORTRAN Compiler provides a number of significant enhancements; most are related to FORTRAN ISA extensions for process control applications. For example, the ISA executive interface (CALL, START, TRNON, WAIT), ISA bit string manipulation (IOR, IAND, IEOR, NOT, ISHFT, IBTEST, IBSET, IBCLR), and the ISA date and time functions are now available.

Executive

A number of enhancements in the online execution area are included. Specifically, tasks can have overlays that are disk-resident and initiated by the root task. The operator's console can be attached through an MLCP. The task management capability is also extended with such features as simultaneous multitasking per priority level and more complex WAIT functionality.

Communications

Communications is supported at two different levels. At the logical I/O level, a programmer can treat a communications terminal as if it were a file (e.g., using OPEN, CLOSE, READ, WRITE). At the physical I/O level, more detailed control of the terminal and line is available. Teletype Model 33 and 35 terminals are supported, along with Teletype-compatible CRTs. BSC (2780) is also supported (ASCII nontransparent and EBCDIC transparent modes).

Utilities

BES2 includes two additional utilities:

- Dump Edit – prints from diskette a memory dump previously dumped there.
- Bootstrap Generator – creates a bootstrap record containing parameters to be subsequently used during bootstrapping and loading.

MLCP Software

The Honeywell-supplied software support for the Multiline Communications Processor consists of the MLCP Loader and the MLCP macro routines.

MLCP Loader

The MLCP Loader resides in Main Memory linked to a user-written program, and is responsible for loading the channel control programs and control structures into the random access memory (RAM) portion of the MLCP, for the purpose of processing the communications data stream of an application.

MLCP Macro Routines

There are a number of macro routines available for writing the channel control programs and other control structures for handling communications data streams. These macro routines are processed by the Macro Preprocessor prior to assembly. For a detailed description of the MLCP software, refer to the *MLCP Programmer's Reference Manual.*

GCOS 6/MDT

Level 6 GCOS Multi-Dimensional Tasking (MDT) is a disk/diskette-based operating system with executive, file management and communications facilities that support multitasking real-time or data communications applications in one or more online streams. In addition, program development or other batch type applications can be performed concurrently in a single batch stream. Figure 7-7 provides an overview of the various modules, languages, programs, and development tools.

Program Development

The program development process (as shown in Figure 7-8) enables the creation of source units via punched cards or the Editor. A source unit comprises source statements written in assembly language, FORTRAN, COBOL, or RPG. If desired, source units can be altered by the Editor. Source units must be converted to object units by a language processor (e.g., the Assembler, FORTRAN Compiler, COBOL Compiler, or RPG Compiler). If assembly language source statements contain one or more macro calls, the source text must be processed by the Macro Preprocessor before it can be processed by the Assembler. The Macro Preprocessor replaces each macro call with a sequence of statements known as a macro routine. Macro Preprocessor output is called an expanded source unit. The Cross-Reference Program can be run to obtain a list of all symbolic names in an assembly language source unit, and to determine whether any of the symbols are undefined, multiply-defined, or defined and not used. If necessary, corrections can be made by using the Editor. Separately assembled and/or compiled object units must be linked by the Linker to form a bound unit. A bound unit comprises a root, or a root and one or more overlays. A root is the portion of a bound unit that is loaded into memory when the Loader is requested to load a bound unit. The root

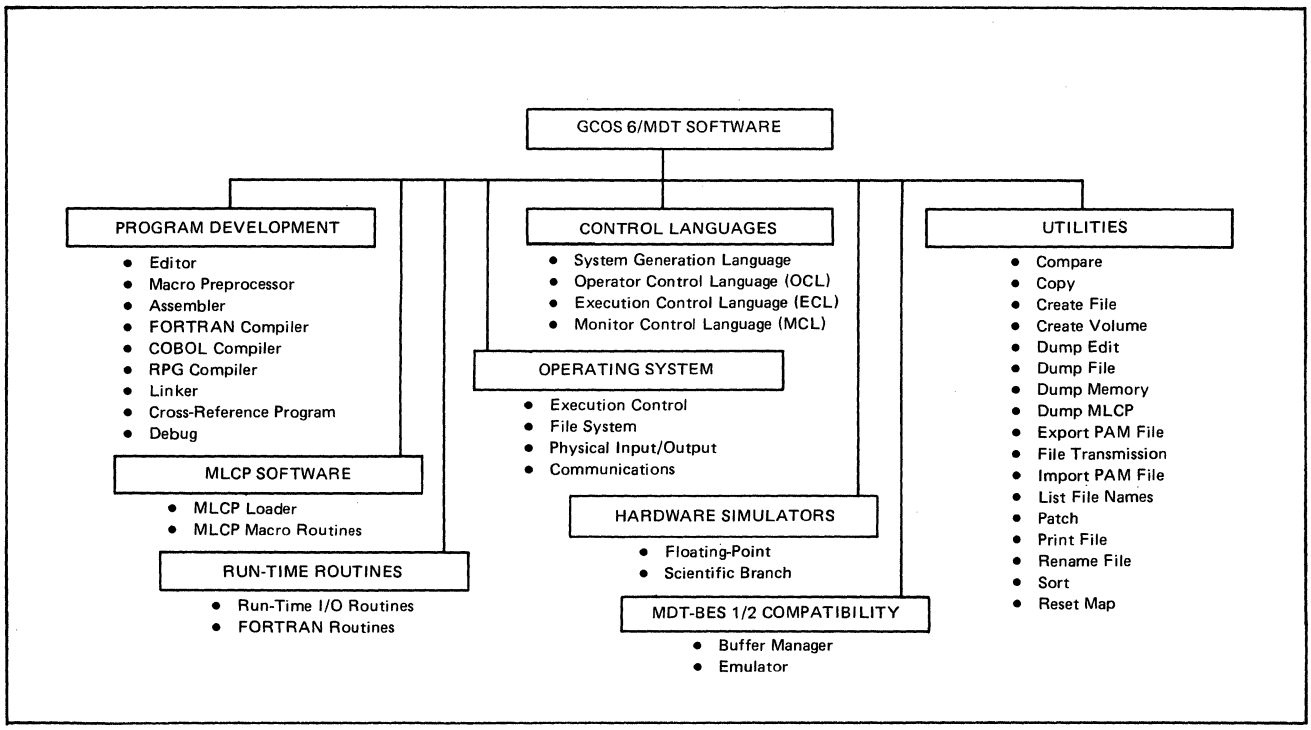


Figure 7-7. GCOS 6/MDT Software Overview

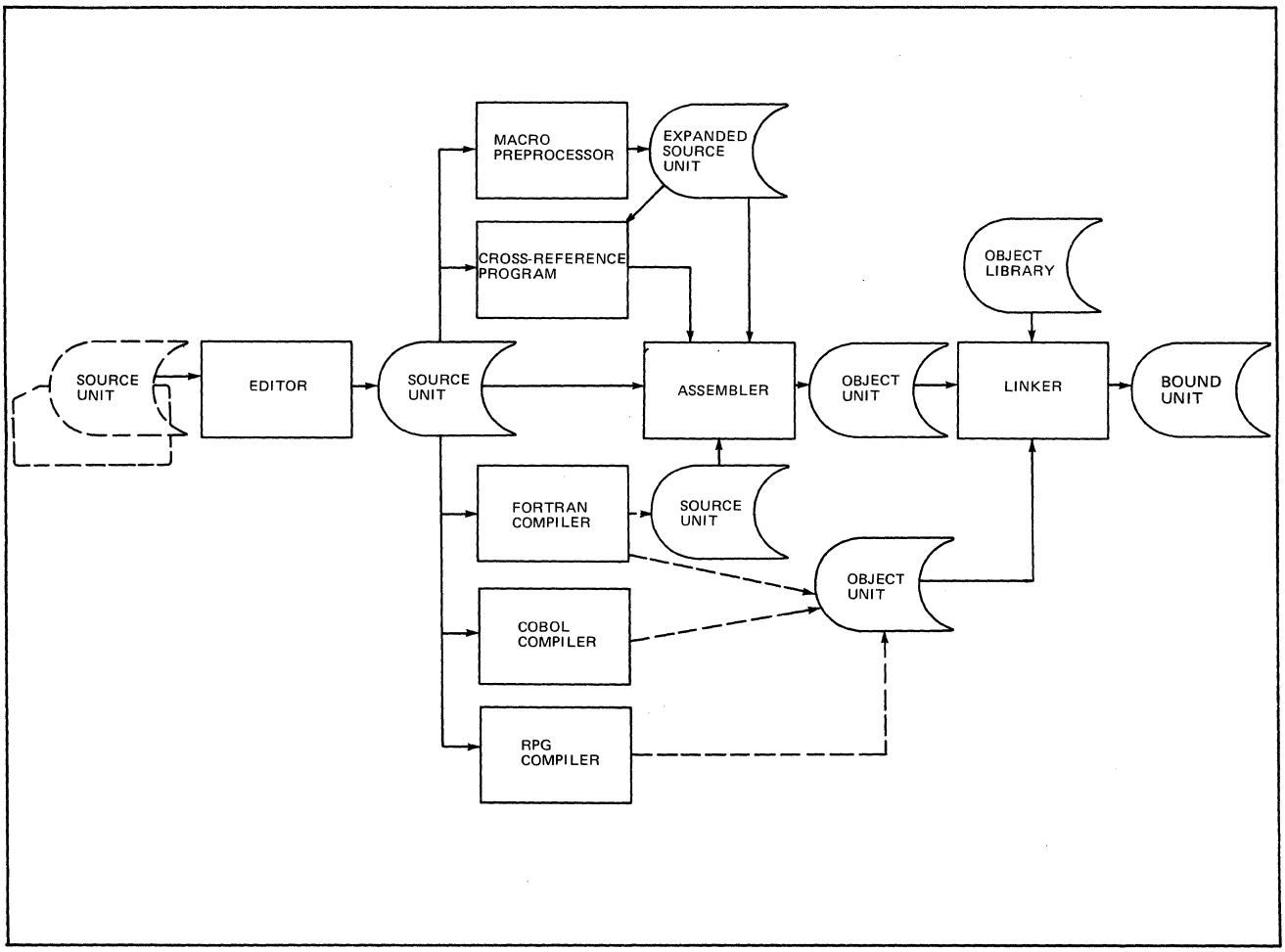


Figure 7-8. Program Development Process

remains in memory as long as there are tasks executing on its behalf. An overlay is loaded into memory whenever it is required.

Editor

The Editor creates and/or updates source statements that are contained in source modules. The statements can be written in COBOL, FORTRAN, RPG, or assembly language. Editing is controlled by directives entered through any interactive or sequential input device on the system.

Editor allows unlimited forward and backward scanning of files and allows files to be created and portions moved via auxiliary buffers. Programs can be prepared and compiled concurrently by running Editor as on online task and scheduling its output for later compilation in the batch stream.

Many powerful editing commands are available:

- o INSERT – adds source lines.
- o DELETE – deletes source lines.
- o PRINT – lists all specific lines.
- o SUBSTITUTE – corrects source lines.
- o READ – incorporates one source module within another.

Macro Preprocessor

The macro preprocessor is a one-pass processor which operates in 9K of main memory, although up to 64K will be used if available. Input is a source unit with macro calls generated as command statements in addition to the macro definitions that were inserted.

Assembler

The Assembler processes source statements written in symbolic language, translating them into relocatable object code and producing a listing of the source program along with its associated assembly information. The Assembler is a nonoverlaid two-pass processor which operates in 9K of memory. During the first pass, the Assembler constructs its symbol table in a memory-resident table area. During the second pass, it generates the object text and/or listing. The assembly listing reflects each source record and the corresponding machine code generated. Multiple users can assemble programs concurrently; however, multiple copies of the program must reside in memory.

In addition, the Assembler can use many of the operating system facilities, such as:

- o Supporting remote terminals
- o Providing for dynamic use of memory. Above a minimum requirement, the Assembler tailors itself to operate in whatever memory is available to it through the operating system.

FORTRAN Compiler

The powerful FORTRAN Compiler is based on the proposed 1977 standard developed by the American National Standards Institute.

The FORTRAN compiler is a fast, one-pass processor which operates in 32K words of main memory, accepting source statements and producing object text and/or a listing of FORTRAN source statements with diagnostics and a memory map. Optionally, the FORTRAN compiler can be directed to generate assembly language statements instead of object code. Using this feature, programmers can embed assembly language statements in FORTRAN statements and/or modify the generated code before assembling.

Extensions to the FORTRAN Compiler for GCOS 6/MDT provide for double-precision data type, extended I/O edit capabilities, and the alternate RETURN statement. Programs generated by the compiler may be used with the Scientific Instruction Processor (optional on Model 6/43).

COBOL Compiler

The previously announced COBOL Compiler, which is based on the 1974 standard developed by the American National Standards Institute and Series 60 standards, has been significantly extended in its support of file handling for sequential, relative, and indexed files, 3-dimensional tables and indexing, CALL/CANCEL capability, DISPLAY and COMP-1 data, full ANS editing, 21 verbs, and communications through file management facilities.

RPG Compiler

The Level 6 RPG Compiler is a compatible language subset of other Series 60 (Levels 62 and 64) compilers, as well as System/3 and System/32.

The RPG Compiler translates RPG source statements of a source unit into a set of object units consisting of a root plus multiple overlays. The Compiler also produces a file containing Execution Control Language (ECL) commands and Linker directives; user-written Linker directives are thus unnecessary to create an executable bound unit. The Compiler is a multi-pass program. Significant features include: lookahead, control levels, and matching fields on input, table and array processing, and editing, detail, and total time functions on output.

Linker

The Linker is a single-pass component that accepts object units produced by the Assembler or by FORTRAN, RPG, or COBOL compilers and joins them to produce one or more bound units. The Linker:

- o Resolves symbolic external references.
- o Links long address form (LAF) as well as short address form (SAF) object units.
- o Accepts directives from any sequential input device.
- o Handles multiple input files.
- o Provides for automatic linking.
- o Produces a link map (optional).

Cross-Reference Program

The Cross-Reference program reads in an Assembly Language source unit, finds all labels, sorts them into alphabetic order, and lists them. On the output listing, beside each label is the number of the line where the label is defined and, sequenced numerically, the number of each line that contains a reference to this label. Duplicate and undefined labels are flagged.

Debug

Debug is used to test programs at the assembly language level. Hexadecimal patches can be made to the program. Debug is invoked differently from other program preparation programs in that it executes as the lead task of a specific online task group, and interfaces with the programmer through the operator's terminal or a file of Debug directives.

Operating System

The operating system can be described as containing software for execution control, the file system, physical I/O, and communications.

Execution Control

The Monitor contains software to execute requests for Monitor functions and to maintain the control tables that are necessary for the orderly processing of requests. These functions are obtained through ECL and OCL commands, monitor and I/O service macro calls, and statements in higher level languages.

- o Task Manager – handles task requests to activate, wait, or terminate tasks.
- o Clock Manager – handles all requests to control tasks based on real-time considerations, and requests for the time-of-day and date in ASCII format.
- o Memory Manager – controls dynamic requests for memory or to return memory to memory pool.
- o Trap Manager – handles an executing program's transfer of execution control to a predefined trap location.
- o Operator Terminal Manager – manages all messages sent simultaneously by multiple task groups to the operator's terminal or from the operator's terminal to a task group.
- o Loader – loads the root and overlays of a bound unit dynamically from disk.
- o Execution Control Language Processor – processes ECL commands. It is the lead task of the batch task group.
- o Operator Control Language Processor – processes OCL commands. It is a special invocation of the ECL processor that runs under the system task group.

File System

The file system is based on a tree-structured hierarchy. Software functions are provided to create or maintain this directory structure, locate a file by its pathname, create and maintain data files, control concurrent use of files, and provide for the logical transfer of records between an application program and an external device. These functions are available through the ECL commands, or, for an assembly language program, through monitor control macro calls.

The file system handles input/output functions of each of the different supported devices, including communications. For disk, it provides four file organizations (sequential, relative, indexed, and fixed-relative) and access to them. (Fixed-relative is compatible with BES1/2 files.) Sequential access is provided for magnetic tape, communications, printer, card reader, and terminals, in addition to mass storage. All unit record devices can be treated as a sequential file.

The languages COBOL, FORTRAN, and RPG use logical file organizations based on the physical organizations listed above. The language reference manuals provide the correspondence between a language's logical file and the system's physical file organization; each language provides statements for accessing the logical files.

An assembly language program may also access the physical files. Files can be accessed through File and Data management macro calls to the Monitor or through the physical I/O drivers.

Physical Input/Output

An assembly language program can use physical input/output driver software which

works at the hardware physical level. Each peripheral and communications device type has a driver which is a reentrant procedure controlling one or more devices. The driver executes as a task at a different priority level for each device being controlled.

Communications

Communications software allows the user to communicate through the file system or physical I/O, with remote KSRs, CRTs, VIP 7700s, and other computers (using the BSC protocol). Communications with these terminals is also available through COBOL and FORTRAN. The interface defined for peripherals is the same interface used for communications.

The components of the communications software are the following:

- o Communications Supervisor – validates, queues and dequeues user requests for service, activates the phone monitor, and services timeouts for all line protocol handlers.
- o Phone Monitor – provides data set control for the detection of connect and disconnect requests by terminal users.
- o Line Protocol Handlers – provide error recovery procedures, process messages, interrupts, and timeouts as well as positive (ACK) and negative (NAK) control signals indicating the error status of a transmission.
- o Poller – (used only with VIPs) maintains the polling schedule for terminals and queues and dequeues all polling requests.

MLCP Software

The MLCP loader and MLCP macro routines are available to the communications system programmer to program the Multiline Communications Processor (MLCP). For a detailed description of the MLCP software, refer to the *MLCP Programmer's Reference Manual.*

MLCP Loader

The MLCP Loader resides in main memory linked to a user-written program and is responsible for loading the channel control programs and control structures into the random access memory (RAM) portion of the MLCP for the purpose of processing the communications data stream of an application.

MLCP Macro Routines

There are a number of macro routines available for writing the channel control programs and other control structures for handling communications data streams. These macro routines are processed by the Macro Preprocessor prior to assembly.

Run-Time Routines

Run-Time I/O Routines

The FORTRAN run-time I/O routines provide for data transfer, peripheral or communications device manipulation, and the processing of data as specified in FORTRAN FORMAT statements. These routines use the File System to accomplish open, close, and position file functions, and to read and write formatted and unformatted records. They contain data conversion routines to edit integer, real, logical, and character data for formatted input and output. Only those routines required by a particular FORTRAN program are linked to form the bound unit.

The COBOL run-time I/O routine provides a logical I/O interface for the transfer and processing of data at program execution time. The routine is linked with the program's object unit, and uses the File System to open, close, and position files, and to read and write records to peripheral or communications devices.

The FORTRAN and COBOL routines produce diagnostic messages to inform the programmer of inappropriate or inconsistent input/output statements.

FORTRAN Run-Time Routines

MDT software includes a large set of FORTRAN mathematical and bit-string manipulation routines. These intrinsic functions are available in object module format, so that they can be linked on an "as-needed" basis to perform a variety of operations on behalf of a FORTRAN program. Operations performed by these routines include:

- o Conversion to and from integer and real values
- o Truncation
- o Determining the nearest whole number
- o Transferring a sign
- o Choosing the largest value or the smallest value
- o Finding the length of a character entity, the square root, the natural logarithm, or the common logarithm
- o Computing selected plane and spherical trigonometric functions
- o A 16-bit quantity manipulation on values of integer data. (The boolean operations of Inclusive OR, Exclusive OR, Product, and Complement are provided as well as the

ability to shift the bits of an integer value, Clear or Set a single bit, and test the value of a bit of an integer.)

- Three tasking subroutines to request task management capabilities:
  - START subroutine initiates execution of an executable program after a specified time delay
  - TRNON (Turn-On) subroutine initiates execution of an executable program at a given time of day
  - WAIT subroutine suspends the execution of an executable program for a specified period of time
- Two time subroutines
  - CALL TIME allows the program to obtain the current hour of the day, minute of the hour, and second of the minute of local time when the call is made
  - CALL DATE allows the program to call the years since 0 A.D. (i.e., this year, 1977), the month of the year, and the day of the month of the date when the call is made

FORTRAN routines are available to implement the management of tasks. Functions are provided to:

- Build task control blocks
- Initiate a task after a designated period of time
- Suspend a task
- Return a task control block

Hardware Simulators

Floating-Point Simulator

The Floating-Point Simulator provides software simulation of floating-point instructions (add, subtract, multiply, divide, compare, load, store, swap, and negate) that are generated by the FORTRAN compiler or the Assembler.

Scientific Branch Simulator

The Scientific Branch Simulator provides software simulation of floating-point branch instructions (branch on bit settings of scientific indicator register or scientific accumulator values).

MDT–BES1/2 Compatibility

Many areas of MDT and BES are fully compatible (e.g., source and object text programs and data management functions), while others require conversion aids. A Honeywell-supplied Accommodation Package is provided to resolve the interface to system services and register usage differences between BES and MDT.

Calls to the BES Buffer Manager are handled by including the Buffer Manager as well as the Accommodation Package in the MDT system. The importing of BES object text programs into the MDT environment is provided by the IM–PAM utility.

Utilities

The following utility programs offer a variety of services. They normally execute as batch programs.

- Compare – compares two mass storage or magnetic tape files or volumes.
- Copy – copies a disk or magnetic tape file or volume.
- Create File – creates a disk file.
- Create Volume – formats a disk or tape volume with a root directory.
- Dump Edit (DPEDIT) – edits and prints out a disk file containing a dump of main memory that was obtained through the MDUMP utility.
- Dump File – dumps a specified disk or magnetic tape file by logical record in both alphabetic and hexadecimal. It can also perform a physical dump.
- Dump Memory (MDUMP) – dumps the contents of memory to a disk file. The Dump Edit utility is then used to print the dump.
- Dump MLCP (DUMCP) – dumps the contents of memory of the Multiline Communications Processor (MLCP) to a designated output device.
- Export PAM File – unloads a MDT sequential file into a BES1/2 PAM file member.
- File Transmission – transmits to a Level 66 or receives a Level 66 file.
- Import PAM File – converts a BES1/2 PAM file member into an MDT sequential file.
- List File Names – lists file entries in a specified disk directory.
- Patch – provides hexadecimal patches for an object unit or bound unit.
- Print File – prints the indicated file on a printer.
- Rename File – renames a disk file.
- Sort – sorts records of an ASCII file based on the character string contents of up to eight keys. Key fields can be sequenced in either ascending or descending order.

- Reset Map – recreates the volume allocation map on a disk volume to reflect the existing file directory information. Used for recovery.

Control Languages

The GCOS/6 MDT operating system is controlled by four distinct language sets; System Generation Language, Operator Control Language, Execution Control Language, and Monitor Control Language. Collectively referred to as the MDT Control Language, they combine a simple command format with commonly used default conditions to provide a concise yet flexible means of communicating orders to the operating system. In addition, the Operator Control Language and Execution Control Language provide a simple method of incorporating user-supplied commands.

System Generation Language

Several areas of the GCOS 6/MDT operating system require specialization to handle a user's particular configuration and application environment. The operating system software, as delivered, incorporates a general set of system generation directives that facilitate the initial operation of GCOS 6/MDT. This allows the user to execute a series of system runs that create a new set of system generation directives that are stored and subsequently used to initialize the final operating system. Particular system areas that may require specialization are:

- Device specification – the most important part of system generation is the specification of the intended system configuration. During system generation, specific directives provide this definition.
- Memory utilization – GCOS 6/MDT memory pools are identified at system generation. Specific memory directives describe both the online and batch areas of memory to be used as memory pools as well as the rules for sharing those areas.
- Executive specialization – the operating system may be tailored to fit users' requirements through the use of these directives during system initialization.
- Communications configuration – definitions of communications devices, lines, and various protocol handlers are specified using these commands.

In all cases, the system provides defaults for the various areas of specialization. Only those user requirements that differ from the most general conditions require the use of system generation directives. Once created, the directives are stored and are automatically executed by system initialization.

Execution Control Language

Any task group in GCOS 6/MDT can be controlled using the Execution Control Language. (The batch task group uses the Execution Control Language *only*.)The Execution Control Language command processor reads commands (statements) from the designated sequential input device (card reader, disk device, or terminal device) and starts tasks corresponding to the specified functions. The various areas of control visible through the Execution Control Language are:

- Task group and task management – permit the start of application task groups in the online dimension and cause tasks within the group to be activated.
- Batch submission – enables the user to submit batch requests for execution in the batch stream.
- File management – commands are available to handle all file-related functions as well as to assign files to applications for processing.
- Utilities and languages – all system utilities, Compilers, and the Assembler are executed using the Execution Control Language.

Operator Control Language

While the Operator Control Language cannot compile or assemble programs, operator commands allow the same interaction with the system as is possible with the Execution Control Language. There are, however, a few very important additions. The general areas in which the operator is allowed more control are:

- Online Task Group Control – the operator can install and delete user task groups and/or application task groups in the online dimension as well as interrogate their status and/or suspend or abort them.
- Batch Stream Control – control of batch stream processing is an operator responsibility. If a batch dimension was configured at system generation, processing by the batch task group can be started, suspended, and/or halted by the operator.
- System Resource Control – devices are assigned to the online or batch dimensions by the operator.

Monitor Control Language

Monitor Control Language (MCL) macro calls are available to an assembly language program to perform a wide variety of Monitor functions, similar, in some instances, to those functions accessible through ECL. MCL macro calls are used to control execution and to control directories and files.

APPENDIX A

INSTRUCTION TIMINGS

6/30 MODELS

Instruction timings for the 6/30 models are listed in Tables A-1 through A-5. Processor timings can vary ±20%.

TABLE A-1. INSTRUCTION AND OPERAND FETCH TIMES (6/30)

	Op codes/Fetch Times (μs)						
Addressing Forms	LDB, SWB, CMB, LDR, SWR, CMR, ADD, SUB, MUL, DIV, OR, XOR, AND, INC, DEC, CAD, CMZ, CPL, NEG, LEV, MTM	STB, STR, STM, CL, SAVE, JMP, RSTR, LNJ, ENT, LAB	LDH, LLH, CMH, ORH, XOH, ANH		STH, CLH	LB, LBF, LBC, LBS, LBT	LDV, CMV, ADV, MLV
			+ Operand	- Operand			
IMA, P + D, B + D[a]	3.72	2.77	4.01	3.72	2.77	3.72	Total Fetch and and Execution times for immediate operands are shown in Table B-2.
IMA + X[a]	4.01	3.06	4.59	4.30	3.35	5.17	
B[a], IMO	2.77	1.82	3.06	2.77	1.82	2.77	
B + X[a]	3.06	2.11	3.64	3.35	2.40	4.22	
↓B, B↑	3.06	2.11	3.35	3.06	2.11	3.06	
B + ↓X, B + X↑	3.35	2.40	3.93	3.64	2.69	4.51	
R Register	1.53	1.53	2.49	2.21	1.53	1.53	
B Register	1.53	1.53	–	–	–	–	

[a]If indirect add 1.24 μs to time shown.

TABLE A-2. EXECUTION TIMES (6/30)

Op Codes	Address Syllable Times (μs)	
	R Register B Register	Any Other
INC, DEC, NEG, CPL, CL, STR, STB, STS	0.58	1.16
CLH, STH	1.16	1.16
STM	0.87	1.45
SWR, SWB	1.16	1.74
CAD	0.58/0.87	0.58/1.45
LBS (I[B] = 1), LBT	1.16	1.74/3.07
LBS (I[B] = 0), LBF, LBC	1.45	2.03/3.36
JMP, ENT	–	0.58
LNJ	–	0.29
LAB	–	0.0
ADD, SUB	0.38	
LDR, LBD, LDH, OR, XOR, AND, ORH, XOH, ANH	0.29	
CMR, CMB, CMH	0.38/0.67	
CMV[a]	1.91/2.20	
CMZ	0.67/0.96	
LEV	4.60/88.20	
SAVE	13.21/22.71	
RSTR	12.63/34.30	
MTM	1.74	
MUL (# ≠7)	11.20	
MUL (# =7)	10.73	
MLV[a] (# ≠7)	12.73	
MLV[a] (# =7)	12.26	
DIV (# ≠7)	13.22/14.56	
DIV (# =7)	13.22/14.27	
LLH	0.58	
LB	0.87	
ADV[a]	1.91	
LDV[a]	1.82	
CMN	To be supplied	
LDI		
SDI		
SRM		
IOH		
LEV		

[a]Total Fetch and Execution Time

TABLE A-3. SHIFT AND GENERIC INSTRUCTION TIMES (INCLUDES FETCH) (6/30)

Shift Instruction Times (μs)		
SOL	1.53 + 0.29d	(See Notes 1 and 2)
SAL	1.82/2.11 + 0.29d	
SCL	1.53 + 0.29d	
DCL	1.82 + 0.29d	
DOL < 16	1.82 + 0.29d	
DOL ⩾ 16	2.11 + 0.29d	
DAL < 16	1.82/2.11 + 0.29d	
DAL ⩾ 16	2.11/2.69 + 0.29d	
SOR	1.53 + 0.29d	
SAR	1.53 + 0.29d	
SCR	1.53 + 0.29d	
DCR	1.82 + 0.29d	
DOR < 16	1.82 + 0.29d	
DOR ⩾ 16	2.11 + 0.29d	
DAR < 16	1.82 + 0.29d	
DAR ⩾ 16	2.11 + 0.29d	
Generic Instruction Times (μs)		
HLT	2.11	
MCL, BRK	16.75/17.07	
RTT	11.85	
RTCN, RTCF	2.11	
WDTN, WDTF	2.11	

NOTES: 1. d = shift distance; $0 \leqslant d \leqslant 15$
2. These are times for procedural shifts. For nonprocedural shifts (d = 0) add 0.96 μs to the above times.

TABLE A-4. R-BRANCH, I-BRANCH EXECUTION TIMES (INCLUDES FETCH) (6/30)

Test Success and Addressing Mode / Op Code	Unsuccessful (μs)	Successful (μs)		
		d ≠ 0, 1 (P + d)	d = 0 (IMA)	d = 1 (P + D)
BLZ, BGEZ, BEZ, BNEZ, BGZ, BLEZ, BEVN, BODD, BINC, BDEC	1.82	2.40/2.69 min/max	3.35	3.35
B, BOV, BNOV, NOP, BAL, BAGE, BE, BNE, BAG, BALE, BL, BGE, BG, BLE, BSU, BSE, BCT, BCF, BBT, BBF, BIOT, BIOF	1.53	2.11/2.40	3.06	3.06

TABLE A-5. I/O INSTRUCTION TIMES (6/30)

Address Syllable Form	I/O Instruction Times (μs)		
	IO		
	Input	Output	IOLD
IMA[a], P + D[a], B + D[a]	10.60	10.94	15.18
IMA + X[a]	11.18	11.52	16.34
B[a], IMO	8.70	9.04	12.33
B + X[a]	9.28	9.62	13.49
↓B, B↑	9.28	9.62	13.20
B + ↓X, B + X↑	9.86	10.20	14.36
IV + D	13.66	14.00	19.77
RA	6.22	5.93	8.32

[a]If indirect in any address syllable, add 1.24 μs for each indirection.

6/40 MODELS

Instruction timings for the 6/40 models are listed in Tables A-6 through A-8.

The times given are for register addressing (SAF mode) utilizing a double-fetch EDAC memory. The minimum times are for the case where the instruction pre-fetch buffers are filled prior to attempting to execute the instruction. The maximum times involve empty pre-fetch buffers. The typical times are a statistical average – due to the nature of the 6/43, the instruction times may vary since the pre-fetch buffers may be full, empty, or half-full. The times do not take into account self-generated conflicts such as memory busy. Processor timing can vary ±20%.

TABLE A-6. INSTRUCTION TIMINGS (6/40)

Instructions		Times (μs)			
Mnemonic	Description	Minimum	Maximum	Typical	Notes
ACQ	Acquire Stack Space				TBS
ADD	Add to Reg	.79	1.47	1.01	
ADV	Add Value to R Reg	.79	1.47	1.01	
AID	Add Integer Double	1.16	1.84	1.38	
AND	And with R Reg	.66	1.34	.88	
ANH	And Half-word	.89	1.57	1.11	
ASD	Activate Segment Descriptor				TBS
B	See Separate Listing For all Branch Instructions				
CAD	Carry Add	.66	1.34	.88	
CL	Clear Memory	.89	1.57	1.11	
CLH	Clear Memory Half-word	1.42	2.10	1.64	
CMB	Compare B Reg	1.07	1.75	1.29	
CMH	Compare Half-word	1.32	2.00	1.54	
CMN	Compare with Null	.89	1.57	1.11	
CMR	Compare R Reg	1.05	1.73	1.27	
CMV	Compare with Value	1.05	1.73	1.27	
CMZ	Compare with Zero	1.30	1.98	1.52	
CPL	Complement	.89	1.57	1.11	
DAL	DBL Shift Left	.72+.16d	1.4+.16d	.94+.16d	1,2,3
DAR	DBL Shift Right	.72+.16d	1.4+.16d	.94+.16d	1,2,3
DCL	DBL Shift Closed Left	.68+.2d	1.36+.2d	.90+.2d	4

TABLE A-6 (CONT). INSTRUCTION TIMINGS (6/40)

Instruction Mnemonic	Description	Times (μs) Minimum	Maximum	Typical	Notes
DCR	DBL Shift Closed Right	.68+.2d	13.6+.2d	.90+.2d	4
DEC	Decrement	.89	1.57	1.11	
DIV	Divide	11.81/11.96	12.49/12.64	12.03/12.16	7
DOL	DBL Shift Open Left	.18+.16d	.84+.16d	.4+.16d	5,6
DOR	DBL Shift Open Right	.18+.16d	.84+.16d	.4+.16d	5,6
DQA	Dequeue on Address				TBS
DQH	Dequeue on Head				TBS
ENT	Enter				TBS
HLT	Halt				TBS
INC	Increment	.89	1.57	1.11	
IO	Input/Output				TBS
IOH	Input/Output Half-word				TBS
IOLD	Input/Output Load				TBS
JMP	Jump	1.53	2.21	1.75	9
LAB	Load EA into B Reg				
LB	Load Bit	1.32	1.98	1.54	
LBC	Load Bit and Complement	1.83	2.49	2.05	
LBF	Load Bit and Set False	1.86	2.54	2.08	
LBS	Load Bit and Swap	1.73	2.41	1.95	
LBT	Load Bit and Set True	1.64	2.32	1.86	
LDB	Load B Reg	.66	1.34	.88	
LDH	Load Half-word	.89	1.57	1.11	
LDI	Load Double-word Integer	1.09	1.77	1.31	
LDR	Load R Reg	.66	1.34	.88	
LDT	Load Stack Register				TBS
LDV	Load Value	.66	1.34	.88	
LEV	Level Change				TBS
LLH	Load Logical Half-word	.89	1.57	1.11	
LNJ	Load and Jump	1.53	2.21	1.75	9
MCL	Monitor Call				TBS
MLV	Multiply By Value	7.85/7.96/7.28	8.53/8.64/7.96	8.07/8.18/7.50	8
MMM	Memory to Memory Move	4.28+.67n	4.76+.67n	4.5+.67n	9
MTM	Modify M Reg				TBS
MUL	Multiply	7.86/7.29	8.54/7.97	8.08/7.51	7
NEG	Negate	.89	1.57	1.11	
NOP	No Operation	.45	1.13	.67	
OR	OR with R Reg	.66	1.34	.88	
ORH	OR Half-word	.92	1.60	1.14	
QOH	Queue on Head				TBS
QOT	Queue on Tail				TBS
RLQ	Relinquish Stack Space				TBS
RSTR	Restore Context	10.7+1.2d	11.75+1.2d	10.9+1.2d	
RTCF	Real-Time Clock Off				TBS
RTCN	Real-Time Clock On				TBS
RTT	Return From Trap				TBS

TABLE A-6 (CONT). INSTRUCTION TIMINGS (6/40)

Instructions		Times (μs)			
Mnemonic	Description	Minimum	Maximum	Typical	Notes
SAL	SGL Shift Left	.72+.16d	1.4+.16d	.94+.16d	3,4
SAR	SGL Shift Right	.72+.16d	1.4+.16d	.94+.16d	3,4
SAVE	Save Context	10.4+.4n	11.08+.4n	10.6+.4n	
SCL	SGL Shift Closed Left	.5+.18d	1.18+.18d	.72+.18d	4
SCR	SGL Shift Closed Right	.5+.18d	1.18+.18d	.72+.18d	4
SDI	Store Double-word Integer	1.32	2.0	1.54	
SID	Subtract Integer Double	1.16	1.84	1.38	
SOL	SGL Shift Open Left	.5+.18d	1.18+.18d	.72+.18d	4
SOR	SGL Shift Open Right	.5+.18d	1.18+.18d	.72+.18d	4
SRM	Store Reg Masked	1.05	1.73	1.27	9
STB	Store B Reg	.89	1.57	1.11	
STH	Store Half-word	1.42	2.10	1.64	
STM	Store M Reg	1.12	1.80	1.34	
STR	Store R Reg	.89	1.57	1.11	
STS	Store S Reg	.89	1.57	1.11	
STT	Store Stack Register				TBS
SUB	Subtract	.79	1.47	1.01	
SWB	Swap B Reg	1.12	1.80	1.34	
SWR	Swap R Reg	1.12	1.80	1.34	
VLD	Validate				TBS
WDTF	Watchdog Timer Off				TBS
WDTN	Watchdog Timer On				TBS
XOH	Exclusive OR Half-word	.89	1.57	1.11	
XOR	Exclusive OR	.66	1.34	.88	

NOTES:
1. Numbers given are for $0 \leq d \leq 15$. For $d > 15$ use 1.28+.16d.
2. For non-procedural shifts add .84 μs.
3. For overflow conditions add .3 μs.
4. For non-procedural shifts add .64 μs.
5. Numbers given are for $0 \leq d \leq 15$. For $d > 15$ use 1.0+.16 (d-16).
6. For non-procedural shifts add .8 μs.
7. Numbers are given in the order of single precision/double precision.
8. Numbers are given in the order of single precision-no overflow/single precision-overflow/ double precision.
9. Another minimum address form for the instruction was used since register addressing is not applicable to the instruction.

TBS = To be supplied

TABLE A-7. BRANCH TIMINGS (6/40)

Op Code	TIMES (μs)					
	Unsuccessful			Successful		
	Minimum	Maximum	Typical	Minimum	Maximum	Typical
BDEC, BEVN, BEZ, BGEZ, BGZ, BINC, BLEZ, BLZ, BNEZ, BODD	.63	1.53	.76	.81	1.71	.97
B, BAG, BAGE, BAL, BALE, BBF, BBT, BCF, BCT, BE, BG, BGE, BL, BLE, BNE, BNOV, BOV, BSE, BSU	.49	1.37	.62	.67	1.55	.80

To obtain timings for other addressing forms, observe the following guidelines:

1. For memory addressing modes except IMO and Push/Pop, add (in μs):

Store Instruction	*Read Modify Write Commands*	*Load Bit Commands*	*All Other*
Word – .19	1.19	.68	Non-LDI .98
Byte – .03			LDI 2.46

2. For displacement modes, add .18 μs.
3. For indexing mode, add (in μs):
   - Word – .34
   - Byte – .5
   - Bit – 1.14
   - Auto Indexed – .54
4. For indirect mode, add .98 μs.

To obtain timings for single-fetch memories, use the following table, where T is the corresponding time listed for a double-fetch memory.

SINGLE-FETCH MEMORIES	Minimum	Maximum	Typical
Register Addressing	T	T+ .78	T+ .26
Store Instructions			
Word	T+ .97	T+1.77	T+1.24
Byte	T+ .81	T+1.61	T+1.08
Read Modify Write Instructions	T+2.07	T+2.77	T+2.30
Load Bit Instructions	T+1.56	T+2.26	T+1.79
LDI Instructions	T+3.34	T+4.04	T+3.57
All Other Addressing Types	T+1.86	T+2.56	T+2.09

1. For displacement modes, add .18 μs.
2. For indexing modes, add (in μs):
   - Word – .34
   - Byte – .5
   - Bit – 1.14
   - Auto Indexed – .54
3. For indirect modes, add 1.08 μs.

TABLE A-8. SCIENTIFIC INSTRUCTION TIMINGS (6/40)

Instruction Mnemonic	Instruction Address Format	Overall Instr. Time (μs)	Model 6/40 Time (μs)	SIP Time (μs)	Available SIP/CP (for overlap)
SML	SA_D op SA_D	8.15	2.03	6.52	1.63/6.12
	SA_Q op SA_Q	11.45	2.03	9.82	1.63/9.42
	R op SA	8.60	2.59	6.97	1.63/6.01
	RR op SA	12.40	2.95	10.77	1.63/9.45
	M_D op SA_D	10.04	3.66	8.21	1.83/6.38
	M_Q op SA_Q	14.57	3.66	12.74	1.83/10.91
SDV	SA_D op SA_D	7.17	2.03	5.54	1.63/5.14
	SA_Q op SA_Q	14.52	2.03	12.89	1.63/12.49
	R op SA	7.62	2.59	5.99	1.63/5.03
	RR op SA	15.47	2.95	13.84	1.63/12.52
	M_D op SA_D	9.06	3.66	7.23	1.83/5.40
	M_Q op SA_Q	17.64	3.66	15.81	1.83/13.98

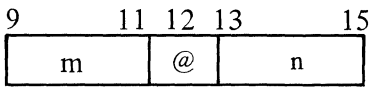
TABLE A-8 (CONT). SCIENTIFIC INSTRUCTION TIMINGS (6/40)

Instruction Mnemonic	Instruction Address Format	Overall Instr. Time (μs)	Model 6/40 Time (μs)	SIP Time (μs)	Available SIP/CP For Overlap
SCM	SA_D op SA_D	3.27	2.03	1.64	1.63/1.24
	SA_Q op SA_Q	3.27	2.03	1.64	1.63/1.24
	R op SA	3.72	2.59	2.09	1.63/1.13
	RR op SA	4.22	2.95	2.59	1.63/1.27
	M_D op SA_D	5.16	3.66	3.33	1.83/1.50
	M_Q op SA_Q	6.39	3.66	4.56	1.83/2.73
SB–	Branch on Indicator	3.23	3.23	0.95	1.63/0.0
SB–	Branch on Accumulator	3.23	3.23	0.95	1.63/0.0
SLD	SA_D SA_D	2.28	2.03	0.65	1.63/0.25
	SA_Q SA_Q	2.28	2.03	0.65	1.63/0.25
	R SA	2.73	2.59	1.10	1.63/0.14
	RR SA	3.23	2.95	1.60	1.63/0.28
	M_D SA_D	4.17	3.66	2.34	1.83/0.51
	M_Q SA_Q	6.00	3.66	4.17	1.83/2.34
SST	SA_D SA_D	2.28	2.03	0.65	1.63/0.40
	SA_Q SA_Q	2.28	2.03	0.65	1.63/0.40
	SA R	2.79	2.79	1.10	1.63/0.0
	SA RR	3.55	3.55	1.60	1.63/0.0
	SA_D M_D	4.43	3.66	2.60	1.83/0.77
	SA_Q M_Q	5.73	3.66	3.90	1.83/2.07
SSW	SA_D SA_D	2.78	2.03	1.15	1.63/0.75
	$SA_Q$$SA_Q$	2.78	2.03	1.15	1.63/0.75
	R SA	3.79	3.79	1.60	1.63/0.0
	RR SA	4.55	4.55	2.10	1.63/0.0
	M_D SA_D	6.67	3.66	4.84	1.83/3.01
	M_Q SA_Q	9.20	3.66	7.37	1.83/5.54
SAD	SA_D SA_D	3.75	2.03	2.12	1.63/1.72
SSB	SA_Q SA_Q	3.57	2.03	1.94	1.63/1.54
	R SA	4.20	2.59	2.57	1.63/1.61
	RR op SA	4.52	2.95	2.89	1.63/1.57
	M_D op SA_D	5.64	3.66	3.81	1.83/1.98
	M_Q op SA_Q	6.69	3.66	4.86	1.83/3.03

op – Operation, SLD (Scientific Load)
SA_D – Scientific Accumulator (Double-word length)
SA_Q – Scientific Accumulator (Quadruple-word length)
SA – Scientific Accumulator (Double- or quadruple-word length)
M – Main Memory Location (Double-word length)
M_Q – Main Memory Location (Quadruple-word length)
R – Central Processor Register (Single integer length)
RR – Central Processor Registers (Double integer length)

APPENDIX B

ADDRESS SYLLABLE REFERENCE TABLE



where:

- m = address modifier
- @ = indirect addressing bit
- n = register number (value between 0 and 7, inclusive)

The single and double operand instructions generate address references through a field called the Address Syllable (AS). The format of the address syllable is as follows:

Table B-1 is a representation of the complete address syllable. An explanation of the mnemonics for the various address forms available appears at the bottom.

TABLE B-1. ADDRESS SYLLABLE REFERENCE TABLE

n = 0			n > 0			
m	@ = 0	@ = 1	@ = 0	@ = 1		
0	IMA	*IMA	Bn	*Bn		
1	IMA+D1	*IMA+D1	Bn+D1	*Bn+D1		
2	IMA+D2	*IMA+D2	Bn+D2	*Bn+D2		
3	IMA+D3	*IMA+D3	Bn+D3	*Bn+D3		
4	P+DSP	*(P+DSP)	Bn+DSP	*(Bn+DSP)		
5	RFU	RFU	Bn register or Dn register	n=1, 2, or 3 Bn +↓D1		n=5, 6, or 7 B(n-4) +D1↑
6	RFU	RFU	↓Bn	n=1, 2, or 3 Bn +↓D2	RFU n=4	n=5, 6, or 7 B(n-4) +D2↑
7	IMO	IV+DSP	Bn↑	n=1, 2, or 3 Bn +↓D3		n=5, 6, or 7 B(n-4) +D3↑

Notation	Description
DSP	16-bit signed displacement that follows the instruction
*	Indirect operator (≠@)
+D	Specifies indexing
↑	Auto-increment (B↑ or D↑ indicates postincrementation)
↓	Auto-decrement (↓B or ↓D indicates predecrementation)
IMA	Immediate Address
IMO	Immediate Operand
IV	Interrupt Vector
B	Base register
D	Operand register
P	Program counter; for the purpose of P relative addressing, P points to the word containing the displacement
()	Logical binding
[]	Contents of
+	Addition operator
-	Subtraction operator
x	Multiply operator
←	Is replaced by
EA	Effective Address
IEA	Intermediate Address

CUT ALONG LINE

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form

TITLE: SERIES 60 (LEVEL 6)
LEVEL 6 MINICOMPUTER HANDBOOK

ORDER NO. AS22, REV. 2

DATED SEPTEMBER 1977

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be promptly investigated by appropriate technical personnel and action will be taken as required. If you require a written reply, check here and furnish complete mailing address below. ☐

FROM: NAME _____ DATE _____

TITLE _____

COMPANY _____

ADDRESS _____

_____

CUT ALONG LINE

PLEASE FOLD AND TAPE –
NOTE: U. S. Postal Service will not deliver stapled forms

FOLD ALONG LINE

FIRST CLASS
PERMIT NO. 39531
WALTHAM, MA
02154

Business Reply Mail
Postage Stamp Not Necessary if Mailed in the United States

Postage Will Be Paid By:

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTENTION: PUBLICATIONS, MS 486

FOLD ALONG LINE

Honeywell

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

20582, 3478, Printed in U.S.A.

AS22, Rev. 2