Customer Engineering Division

WANG COMPUTER SYSTEMS 60 & 80

Volume 1

System Introduction

2ND Edition (Reprint) MAY, 1982

WANG LABORATORIES, INC.

ONE INDUSTRIAL AVENUE, LOWELL, MASSACHUSETTS 01851, TEL. (617) 851-4111, TWX 710 343-6769, TELEX 94-7421

November, 1977

PREFACE

The WCS 60 and 80 systems represent a significant departure from Wang's pre-1977 product lines. At the heart of a WCS 60/80 system, the 2200VS Central Processor operates on a completely different set of concepts. New terminology introduced in this text during the development of each concept must be understood before details of 2200VS hardware are studied.

Publications for WCS 60/80 Service training/support are grouped into four categories: 1)Pretraining, 2)In-class handouts, 3)a Post-training package, and 4), Miscellaneous 2200VS documentation.

Pretraining documentation is primarily intended for individuals scheduled for formal WCS 60/80 training at the Home Office. Each of the publications in the 'Pre-training package' must be read thoroughly before the prospective trainee attends class.

Pre-Training Publications:

1. 2200VS, Volume I - System Introductory Manual (CE)
2. 2200VS Programmers's Introduction (WL#800-1101PI)
3. CDC Microcircuits Manual (OEM)

In-class handouts will be used as training aids for daily classroom and laboratory sessions. The following publications comprise the 'in-class handouts' package, and must be kept in class.

In-Class Handouts:

1) CP Hardware Manual (CE)
2) IOP Master Processor Manual (CE)
3) Parallel Workstation IOP Manual (CE)
4) IOP Tester Manual (CE)

5) Hard/Floppy Disk I/O Manual (CE)

6) 2200VS, Volume II - System Installations, Diagnostics, and Troubleshooting (CE)*

7) 2200VS, Volume III-Preventive Maintenance (CE)*

8) 2200VS Pre-Installation Guide (Site preparation)*

9) 2200VS Principles of Operation (WL#800-1100PO)

10) 2200VS Assembly Language Manual (WL#800-1200AS)

11) 2200VS System Programmer's Guide (WL#800-1103SP)

12) IBM Assembly Language Student Text (IBM #SC20-1646-6)

13) Chaintrain - Logic & Troubleshooting Manual (OEM)

14) Chaintrain - Maintenance Instructions Manual (OEM)

15) Chaintrain - Operation Instructions Manual (OEM)

16) Chaintrain - Principles of Operation Manual (OEM)

17) CDC 75 Meg Disk Manuals (OEM):
    a) CDC Storage Module Drive BK4XX-BK5XX
    b) CDC BK4XX-BK5XX Installation & Checkout
    c) CDC BK4XX-BK5XX Diagrams & Wire Lists

18) CDC 288 Meg Disk Manuals (OEM)
    a) CDC Storage Module Drive BK6XX-BK7XX
    b) CDC BK6XX-BK7XX Installation & Checkout

19) CDC Field Test Unit Manual (OEM)

20) Kennedy Mod. 9100 Vacuum Column Tape Transport Manual (OEM)

21) Kennedy Mod. 9219 Formatter Manual (OEM)

22) Schematics for all Wang circuit boards

The Post-training package can be used by those who have already attended a WCS 60/80 seminar. These publications are as follows:

Post-Training Package:

1) Model 61V Printer Manual (CE)*

2) Zilog Z-80 Manual (CE/OEM)

3) 2200VS Cobol Language Ref. Manual (WL#800-1201CD)

4) 2200VS Executive Introduction Manual (WL#800-1105EI)

5) 2200VS BASIC Language Ref. Manual (WL#800-XXXX)*

*Not published as of November, 1977.

6) 2200VS File Management Utilities Manual (WL#800-1300 FM)

7) 2200VS RPG II Reference Manual (WL#800-XXXX)*

8) CDC 'Hawk' Training Manual (CE/OEM)

9) CDC 'Hawk' Training Manual (CE/OEM)

The final category is rather informal, and consists of certain user manuals, data sheets, product bulletins, competetive profiles, etc. The present list of such publications is as follows:

Miscellaneous Documentation:

1) 2200VS Brochure (WL#700-4507)

2) 2200VS Product Bulletin - No. 155

3) 2200VS/WCS 60/WCS 80 Data Sheet (WL#800-2101, 2201)

4) 2200VS Peripherals Data Sheet (WL#800-2102)

5) 2200VS Languages Data Sheet (WL#800-2201)

6) 2260V Ten-Megabyte Disk Data Sheet (WL#800-XXXX)*

7) 2265V-1 (75-Meg Disk) Data Sheet (WL#800-XXXX)*

8) 2265V-2 (288-Meg Disk) Data Sheet (WL#800-XXXX)*

9) 2231W-1 (120 cps Matrix Printer) Data Sheet (WL#800-XXXX)*

10) 2231W-2 (120 cps Matrix Printer) Data Sheet (WL#800-XXXX)*

11) 2221W (200 cps Matrix Printer) Data Sheet (WL#800-XXXX)*

12) 2261W (240 lpm Matrix Printer) Data Sheet (WL#800-XXXX)*

13) 2263V-1 (400 lpm Train Printer) Data Sheet (WL#800-XXXX)*

14) 2263V-2 (600 lpm Train Printer) Data Sheet (WL#800-XXXX)*

15) 2209V (1600V)I 9-Track Tape) Data Sheet (WL#800-XXXX)*

16) 220V06 Telecommunications Data Sheet (WL#800-XXXX)*

Presently, requests for 2200VS documentation 'packages' can only be honored for prospective WCS 60/80 students.

The remaining 11 items of this memorandum generally describe the contents of existing and proposed 2200VS user manuals.

*Not published as of November, 1977.

1. 2200VS PROGRAMMER'S INTRODUCTION (WL #800-1101PI)

This manual was previously known as the "8300 System Introduction". It preserves all of the material from the original System Introduction with the exception of Chapter 1 (general discussion of system features), which moves to the new "EXECUTIVE INTRODUCTION TO THE 2200VS" (see next page). New material was added on Data Management functions (file types and access methods, shared files, etc.). Specifically, the manual covers the following topics:

Introductory Concepts
Workstation Characteristics
Command Processor
System Utilities
Procedure Language
Data Management Functions

2. 2200VS COBOL REFERENCE MANUAL (WL #800-1201CB)

Reference manual for 2200VS COBOL, with material on multiple indexing, additional workstation support features, shared files, etc.

3. 2200VS PRINCIPLES OF OPERATION (WL #800-1100PO)

General descriptions of machine architecture, machine instruction set, and I/O devices.

4. 2200VS ASSEMBLER LANGUAGE REFERENCE MANUAL (WL #800-1200AS)

Reference text for 2200VS macroassembler.

*Not published as of November, 1977.

5. 2200VS SYSTEM PROGRAMMER'S GUIDE (WL #800-1103SP)

A collection of "all the things you wanted to know about the system but were afraid to ask". Included are descriptions of all super- visor calls (SVC's), system macros, and control blocks.

6. EXECUTIVE INTRODUCTION TO THE 2200VS (WL #800-1105)

A brief, general introduction to major system features intended for the edification of management. This manual will incorporate material from Chapter 1 of the present "8300 System Introduction", as well as the 2200VS Product Bulletin, to provide clear, simplified discussions of concepts such as virtual memory, print spooling, background processing, etc.

7. 2200VS BASIC LANGUAGE REFERENCE MANUAL (WL #800-XXXX)*

Reference manual documenting the elusive 2200VS version of BASIC.

8. 2200VS FILE MANAGEMENT UTILITIES (WL #800-1300FM)

Documentation of the CONTROL, DATENTRY, and REPORT utilities. Oriented towards non-programmers who wish to use these utilities for data entry and report generation.

9. 2200VS SYSTEM MANAGEMENT GUIDE (WL #800-XXXX)*

A manual intended for use by system administrators. This manual covers topics of interest to those responsible for system administrative functions, including:

Security (transplanted from "Console Operator's Guide")
File Backup Procedures
Error Reporting and Generating Formatted Dumps
System I/O Error Log
System Performance Evaluation (when is more memory needed, or another disk, etc.)

10. 2200VS SYSTEM OPERATION GUIDE (WL #800-1102SO)

Previously titled "8300 Console Operator's Guide", this manual covers various topics of interest to the system operator, including:

Print Spooling
Background Processing
Operational Characteristics of all peripherals

11. 2200VS RPG II REFERENCE MANUAL (WL #800-XXXX)*

Reference Manual for RPG II.

DIRECT ALL COMMENTS ON THIS MANUAL TO THE TECHNICAL WRITING STAFF OF WANG LABS' CUSTOMER ENGINEERING DIVISION.

*Not published as of November, 1977.

TABLE OF CONTENTS

SECTION 1 SYSTEM OVERVIEW

SECTION 2 SYSTEM CONCEPTS

SECTION 3 INTRODUCTION TO 2200VS HARDWARE

SECTION 4 WORKSTATION CHARACTERISTICS

SECTION 1 SYSTEM OVER-VIEW

SECTION 1
SYSTEM OVERVIEW

1.1 INTRODUCTION

The Wang 2200VS, also known as the 8300, is an interactive, multiuser, general-purpose computer.

Initially, the 2200VS will be offered in two packaged systems, the WCS/60 and the WCS/80. A standard WCS/60 System will include the 2200VS computer with 64K of memory, a 308K diskette drive, a 10 megabyte disk drive, one workstation, and a 240 lpm printer. A standard WCS/80 will include a 2200VS computer with 256K of memory, a 308K diskette drive, two 75 megabyte disk drives, a workstation, and a 240-line-per-minute printer. Both systems can be expanded with additional memory, more disks, and more workstations. Both are designed to serve, even in their minimal configurations, as complete commercial data processing systems.

1.2 INTERACTIVE OPERATION

The 2200VS, like all previous 2200 systems, is "interactive", allowing users to communicate directly with the system from workstations. The system requests user-specified data and provides useful information in a series of clear, nontechnical prompts displayed on the workstation screen. Such prompts may ask the user to "fill in the blanks" with requested data, or select a desired item from a number of displayed options.

Many other systems require the use of a complex special language for issuing instructions to the system and controlling system functions. On the 2200VS, no special language is required. All system functions are invoked through a special program called the 'command processor'. The user simply chooses the desired function from a displayed menu, then responds to any subsequent prompts asking information for that function. If the user response is erroneous or insufficient, the system returns an error message which identifies the problem, and, in many cases, suggests a possible solution.

Running a program on any existing 2200 processor is a trivial task: The user simply loads the program, then keys RUN and EXECUTE. On the 2200VS, it is scarcely more complicated. The user simply chooses the RUN PROGRAM function from the Command Processor Menu, then types in the program name, disk library, and volume in which it is located (this is the equivalent of performing a LOAD DC operation on the 2200). When this information is entered, the user keys ENTER to begin program execution.

At any point during the execution of a program, a user can interrupt the program by keying HELP. The HELP key is somewhat analogous to the HALT/STEP key on the 2200: it temporarily halts program execution, without destroying any critical information or closing any files. The program can be resumed from the point of interruption with a CONTINUE command. While a program is interrupted, the user has access to all system functions. He can, for example, examine the status of open files or I/O devices, scratch or rename files, or begin debug processing. Once these functions are completed, normal execution of the interrupted program can be continued.

1.3 THE 'COMMAND PROCESSOR'

All user communication with the 2200VS is carried out by issuing interactive 'commands' which direct the Operating System* to perform a variety of tasks. Commands, selected from a 'Command Processor Menu', are used to perform such operations as running programs, setting default parameters, scratching and renaming files, initiating debug processing, and examining and modifying various aspects of the system status.

In general, if any program parameter is inadvertently omitted or specified incorrectly, the Command Processor simply repeats the appropriate screen prompt so that the user may enter the correct value.

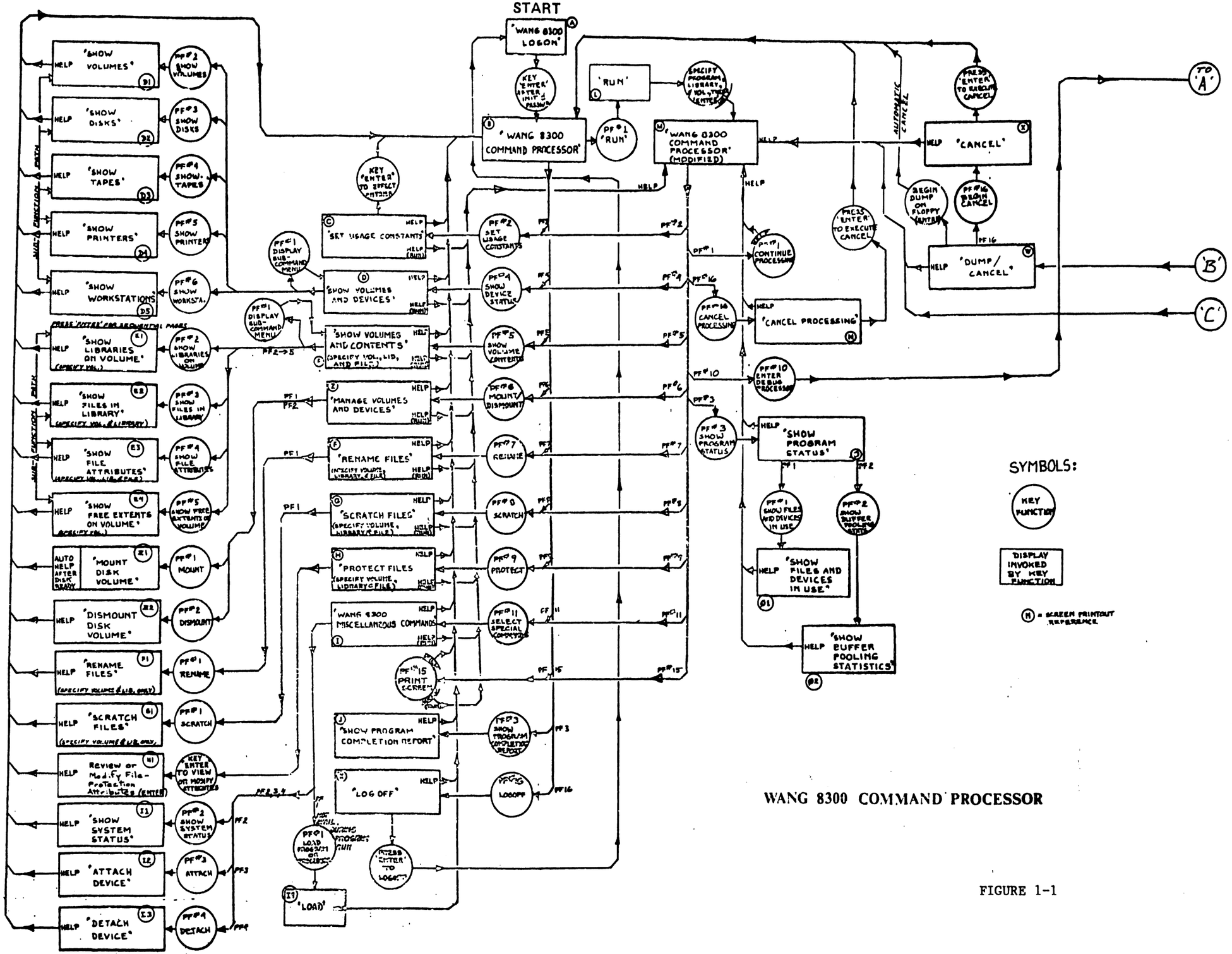
The Command Processor has two slightly different versions of its menu. The "normal" Command Processor menu is displayed whenever no program is running at a given work station. The user is therefore

provided immediate access to the system upon completion of any program, so that a new program or command can be initiated at once. If a running program is interrupted prior to completion, as the result of a fatal execution error, etc., the Command Processor displays a modified Command Processor Menu, which permits the user to either continue program execution or cancel the program.

A program cannot be continued while a selected 'command' from the modified menu is executing. Nor, can a second program be run at the same work station while the first program is interrupted. In order to run a new program, the current program must be terminated, either through normal completion or cancellation by the user. For this reason, the RUN command does not appear in the modified Command Processor Menu. RUN is replaced by CONTINUE.

The diagram on the following page is a map of the 2200VS/8300 Command Processor. Each circle on the map indicates that some keyboard function takes place; each rectangle on the map represents a screen that is presented to the user by the Command Processor. The map takes the workstation operator from LOGON, through all primary functions of the Command Processor, including the Debug Processor. Note that certain screens may be invoked by either the Command Processor or the Modified Command Processor. When the HELP key is used during a program run, the Modified Command Processor is invoked, rather than the Command Processor. The Command Processor map should be used in conjunction with the Programmer's Introduction Manual (WL# 800-1101PI-01) when familiarizing oneself with the 8300 workstation.

*See Section 2; Introductory Concepts.



WANG 8300 COMMAND PROCESSOR

FIGURE 1-1

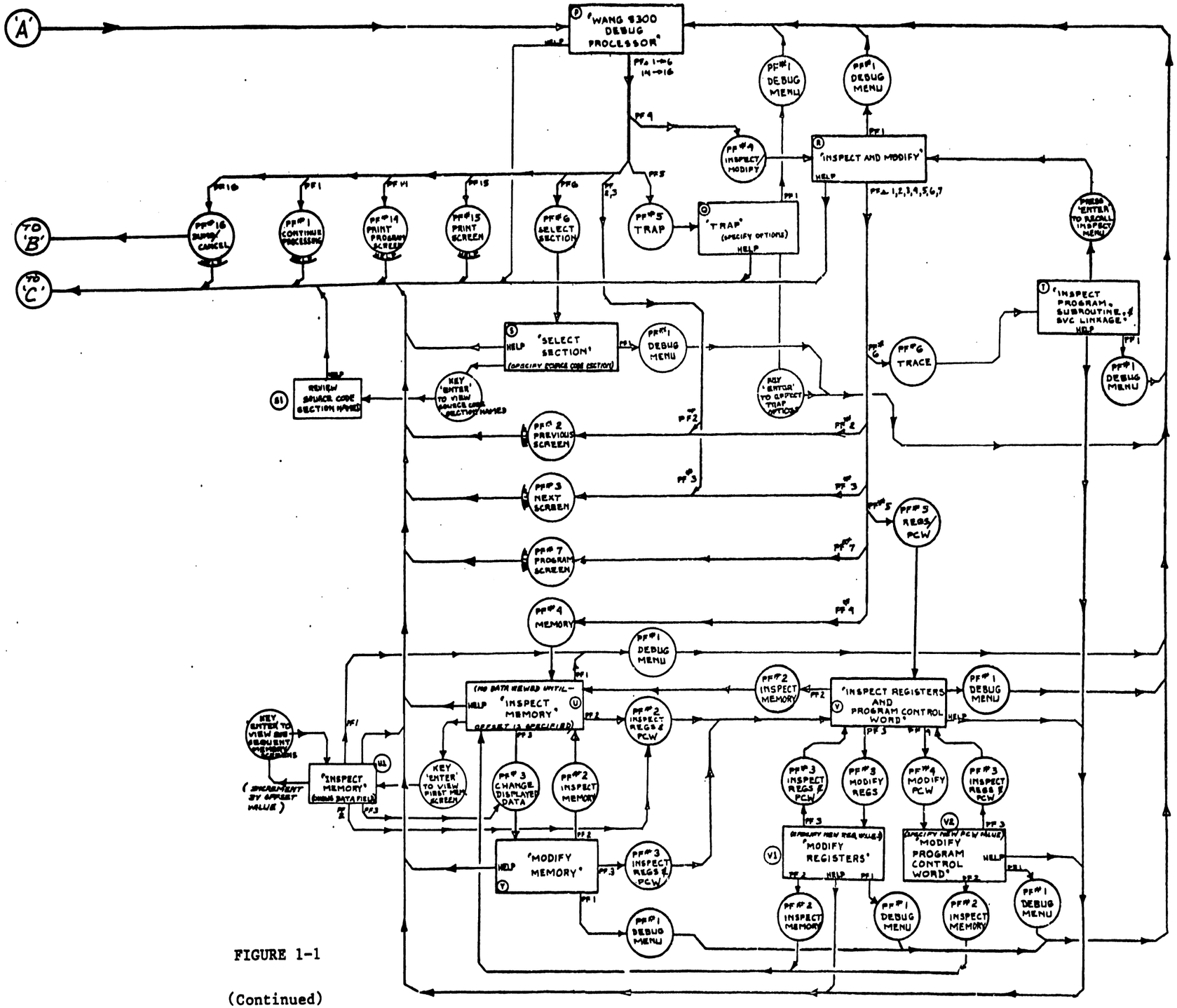


FIGURE 1-1

(Continued)

It is important to understand that, although the system can process two or more tasks concurrently from several work stations, a single work station cannot perform two or more tasks concurrently. Only one program or procedure may be run from each work station at any time.

1.4 MULTIPLE USERS

The 2200VS supports multiple workstations; in a WCS/60 configuration, a maximum of 16 workstations can be supported; in a WCS/80, the maximum is 24. Each workstation can be running its own job concurrently with the others. Since each "job" is actually a separate program, such a system is often referred to as a 'multiprogramming' system. Of course, since there is only one central processor and one memory, it is not possible for two or more programs actually to be executing at exactly the same instant. On the 2200VS, all programs share a common set of resources. Each program (or some portion of it) is kept in its own location(s) in memory. The central processor permits each program to run for a short period of time, then interrupts and permits another program to run for a similar brief period; this process continues indefinitely, or until all programs terminate. Because the central processor is fast and is managed efficiently, the illusion is created that each user's program is running without interruption, simultaneously with all other programs.

The key to the success of such a scheme is the Operating System software, which must manage the use of common resources such as the central processor, memory, and I/O devices, with maximum efficiency. The 2200VS operating system is designed for such efficiency; it is designed to guarantee that each user will have reasonable response time for his program, and also ensure that each program runs without interference from others. The process of sharing a single computer among many users is called 'distributed processing'. Each user can proceed exactly as if he had access to his own private system.

The 2200VS Operating System imposes no special restrictions on the types of jobs which can be run concurrently from each workstation. It is not unlikely, for example, that an installation might have two or more workstations running a large data entry application, while other workstations are running an order-entry program, and still other workstations are being used by programmers for program development. Such a system obviously provides great flexibility for the user, who is not forced to purchase several different systems to perform different jobs.

Typically, in a multi-workstation configuration, one workstation is designated as the 'system console'. In addition to running normal programs, the system console provides a second mode of operation in which it can control a number of special system features not accessible from regular workstations, including 'print spooling' and 'background processing', explained in paragraphs 1.8.4 and 1.8.5.

1.5 MULTILINGUAL SYSTEM

Unlike previous 2200 Systems, the 2200VS supports multiple languages. ANSI COBOL and BASIC, as well as Assembler Language is currently available. Also unlike previous systems, 2200VS languages are 'compiled' rather than 'interpreted'.

COBOL, (Common Business Oriented Language), is one f the most popular programming languages in use today for commercial data processing applications. COBOL programs read like ordinary business English, yet the language provides an array of powerful record formatting, data manipulation, and file handling capabilities which are particularly important for data processing applications. In part, also, COBOL's popularity derives from the fact that it is the only major language subject to an industry-wide standard, administered by the American National Standards Institute (ANSI).

For the programmer who wishes to obtain a greater degree of control over the system and write more efficient programs, the 2200VS also provides an Assembler language. Since the instruction set of a

machine defines the complete set of elementary capabilities provided by the machine, Assembler language provides the programmer with access to the machine's total repertoire of functions.

The 2200VS Assembler also allows a programmer to define a routine consisting of a series of instructions, and assign a name to the routine. The name can then be specified (instead of the entire routine) as a single instruction in a program. Such named routines are called "macros," and the names assigned to them are called "macroinstructions." Because the 2200VS Assembler permits the definition of macros, it is also referred to as a 'macroassembler'.

It is important to note that the machine instruction set of the 2200VS contains all instructions available on the IBM 360, along with most available on the 370. Programmers familiar with IBM 360/370 Assember Language should therefore find 2200VS Assembler easy to learn.

The 2200VS Procedure Language allows users to create special text files which perform many of the operations normally executed interactively by the user at a workstation. Some typical procedures would be: running two or more programs sequentially, supplying run-time parameters to a program, and scratching or renaming programs. Procedures can reduce the number of keystrokes and interactions required of a user who is running a program.

'Compilers', 'Interpreters', and 'Assemblers' are discussed more thoroughly in Section 2.

1.6 LARGE ON-LINE FILES/FILE MANAGEMENT FACILITIES

To support applications with large on-line data base requirements, the 2200VS supports extensive on-line data storage capacity. Both the 10-megabyte and 75-megabyte disk units are supported. In a WCS/60 configuration, the maximum disk storage capacity is 150 megabytes; in a WCS/80, the maximum is 600 megabytes.

The 2200VS 'Data Management System' provides a comprehensive disk file access and maintenance capability. Two types of 'files' are supported on the system: 'sequential files', in which records are stored in the order in which they are written; and 'indexed files', in which records are stored in order of their key values. In both types of files, records may be accessed either sequentially or randomly. A third access mode, dynamic access mode, permits a program to switch back and forth between sequential and random access on the same data file.

The indexed file system permits multiple indexes for a single file. This feature enables a record to be accessed on different keys for different purposes. An employee record, for example, may be accessed by employee name for personnel purposes and by employee number for payroll purposes.

A single file can be shared among several different users. Several users may therefore perform updates and/or inquiry operations on a common file concurrently. In data entry applications, for example, all operators can directly update a single master file. The additional steps of creating temporary files for all operators, and then merging them together are therefore eliminated.

The 'Data Management System' is discussed in more detail in Section 2.

1.7 VIRTUAL MEMORY

1.7.1 GENERAL

To understand the need for virtual storage, one must first understand the characteristics and shortcomings of conventional real storage management.

When a program is entered into a computer, it must be tailored by the programmer to fit within the confines of that computer's physical (real) memory. Secondly, the memory space taken up by that program must be allocated in one piece. Other data or programs cannot be allocated to the same section of real memory. If a program requires (for example) 50,000 bytes of real storage, but during execution uses only 12,000 bytes actively, the remaining 38,000 bytes of real storage are effectively wasted. If a program is too large to fit into the available memory, it must be broken up into a series of modules, or overlays, each of which is small enough to fit in memory. Similarly, if there is not enough memory to hold all of the data used by a program, special procedures must be written into the program to handle the data in smaller chunks.

To further complicate problems, Operating System routines also take real memory space. Real storage tends to become 'fragmented'; that is, a condition occurs where there are many unused storage locations, but these are spread out all through real memory, and there is no single piece of contiguous storage large enough to meet a current demand for real memory space.

Finally, if a memory upgrade is purchased, programs must be revised, sometimes extensively, to take advantage of the added memory. The net effect of these considerations is that programs must be designed to meet the restrictions imposed by a particular machine, rather than to most efficiently deal with the problems these machines are intended to solve.

To provide efficient management of real storage space, memory allocation responsibilities must be taken away from the user. In the 2200VS, this management function is performed by the Operating System.

The process by which blocks of program address space are placed in ('bound' to) a computer's real storage space for execution is called 'relocation'. The 2200VS uses a 'dynamic relocation' process

as the foundation for its virtual memory system. Indeed, one of the most important features of the 2200VS is its 'virtual memory' system. The term 'virtual memory' more specifically refers to a technique of memory management in which the Operating System uses disk storage as an extension of physical memory, automatically ensuring that only those sections of a program and its associated data which are frequently referred to or 'referenced' during program execution are kept in physical memory; less frequently referenced sections are kept on disk until needed.

1.7.2 ADVANTAGES AND DISADVANTAGES OF "VIRTUAL MEMORY"

Virtual memory can best be explained by comparison with many larger computer systems (in the 2200VS's price range) that do not have virtual memory capability. On previous 2200 models, if a programmer wrote a program that exceeded the total available real memory space, he would have had to carefully break the program into a series of modules, or overlays, each of which would have been small enough to fit in real memory. As previously stated, this procedure involves the addition of special software (a 'procedure') to control the overlay process.

On a multiprogramming system, where several users share the same physical memory, this problem becomes even more complex. Each user must know how much of the total memory is available to him. One solution is to divide the total memory into a number of 'partitions' of fixed size, and give each user his own partition. Such systems are called fixed partition systems. They are simple but inefficient, since when a user is not using his entire partition, the unused portion remains vacant, even if a user in another partition requires more memory for his application. A refinement of this technique, called dynamic partitioning makes more efficient use of memory by expanding or contracting the size of each user's partition as his memory requirements increase or decrease (subject, of course, to the requirements of other users). Partitioning systems have commonly been

used on minicomputers (such as certain DEC systems). An alternative to partitioning used on some systems is swapping. In a system that employs swapping, each user has access to all of the available real memory. The system manages this by swapping entire programs in and out of memory from the disk. Each program is brought into memory, permitted to run for a brief interval ('time slice'), and then is swapped out while another program takes its place in memory. One example of a competitive system which utilizes swapping is the IBM System 34.

These systems still share two serious inefficiencies:

1. They force the programmer to tailor programs to fit the available memory.

2. They require the user to modify his software in order to take advantage of additional memory.

These drawbacks are eliminated with a virtual memory system.

In a virtual memory system, the Operating System (rather than the programmer) automatically performs the function of fitting a program to the available real memory space. The Operating System accomplishes this task by first dividing the available physical memory into a number of fixed-size 'page frames'. On the 2200VS, each page frame is 2K bytes in size; thus a 64K system would be divided into 32 page frames. The program address space is correspondingly divided into a number of pages, also 2K bytes in size. When program execution begins, the Operating System loads the first page of the program into an available page frame (2K block) in physical memory. Other page frames may be loaded with the 'first pages' from other programs. If additional page frames are available, the system loads in as many pages of the programs as there are available page frames in memory.

As program execution proceeds, reference may be made (either through the normal sequence of execution or as the result of a branch) to a section of the program not currently located in physical memory. When such a reference occurs, the system automatically interrupts program execution. It then locates the referenced page in virtual memory (on disk), loads that page into an available page frame in physical memory, and resumes execution of the program. This entire process takes place automatically without the user's intervention or knowledge.

Because the Operating System automatically handles the job of overlaying each program and transferring pages in and out of memory as necessary, the amount of physical memory available does not impose any constraint on the program size. The programmer can write his program to meet the needs of the application most efficiently, without regard for the size of that program or the physical characteristics of the system.

The total amount of virtual memory accessible to each user on the 2200VS is one megabyte. Each user has, in effect, his own one-megabyte computer. So long as his program and its associated variable data do not exceed one million bytes in size, there is no need to break it into overlays. Note that the one megabyte figure for each user's accessible virtual memory space is constant, and is not dependent upon the amount of physical memory available, nor upon the number of users on the system. A 64K system with 16 users provides each user with one megabyte, as does a 512K system with only four users.

Although the amount of memory accessible to each user does not vary with the physical memory size or number of users, the system's response time is affected by these factors. The system guarantees each user one megabyte of virtual memory by keeping all unreferenced program pages on a disk, and storing only as many of these pages in physical memory as there are available page frames. If there are many

users on the system and relatively little physical memory, the total number of pages from each program which can be kept in memory at any time will be small. In this case, the system must spend a good deal of time transferring pages in and out of memory as program execution proceeds. (But even in this situation, a virtual memory system would still offer better performance than either a partitioning or a swapping system.) Since the disk I/O operations involved in page transfers are relatively slow compared to actual execution time once a page is in memory, inadequate memory with too many users can degrade system performance.

Conversely, whenever a memory upgrade is purchased in a virtual memory system, all programs experience an immediate improvement in response time with no changes in the software itself. In other types of systems, such performance improvement may be immediate, or may require software modification. Virtual memory is the only system which guarantees automatic performance improvement without software modification when more memory is added.

In summary, 2200VS virtual memory offers the following advantages:

1. Allocation of memory is managed among several users more efficiently than other, less sophisticated memory management techniques.

2. Each user is provided with access to a one-million-byte virtual memory (½ Meg program address space, ½ Meg modifiable data space).

3. Program size is made independent of memory size. Programs can be designed to meet the needs of the application, not the restrictions of a particular machine. The same programs will run on other 2200VS systems with different memory configurations.

4. Memory upgrades will automatically improve performance, without requiring modifications to existing software.

1.8 EXPANDABILITY

The modular design of the 2200VS permits it to be readily expanded with additional physical memory, more disks, and additional workstations and printers. (Expansion can be carried out with no impact on existing software.)

A WCS/60 configuration can be expanded from the minimum 64K of memory to a maximum of 256K, in 64K increments. Disk capacity can be increased from the minimum 10 megabytes to a maximum of 150 megabytes. Up to 16 additional workstations can be added (for a total of 17). High-performance printers are also available.

A WCS/80 configuration can be expanded from the minimum 256K of memory to a maximum of 512K, with disk storage expandable from 150 megabytes up to 600 megabytes. The WCS/80 can support up to 23 workstations.

The user with distributed data processing requirements can therefore, purchase several system configurations of differing size and complexity, and utilize a common set of application software on all systems.

1.9 MAJOR PERFORMANCE FEATURES

In addition to those already mentioned, the 2200VS provides a diversity of features designed to increase total throughput and improve system performance. Among these are: automatic program sharing, independent I/O processors, automatic data compaction, automatic print spooling, and background processing.

1.9.1 AUTOMATIC PROGRAM SHARING

When two or more users are running the same program at the same time, it would be wasteful to keep a separate copy of the program in memory for each user. To avoid such duplication, the system auto-

matically causes several users to share the same copy of a program in memory whenever possible. The amount of memory saved by this feature can be substantial when, for example, a number of users are running a large data entry program, or several programmers are compiling COBOL programs. Program sharing also improves performance for all users by reducing the total number of pages which must be transferred in and out of memory.

1.9.2 INDEPENDENT I/O PROCESSORS

Most commercial application programs spend a good deal of their time performing I/O operations, such as reading and writing disk files, or sending output to a printer. On the 2200VS, I/O operations are handled by independent I/O processors, which control the transfer of data between memory and the various I/O devices. When a program requests an I/O operation, the central processor notifies the appropriate I/O processor, supplies it with any necessary information, then turns its attention to other processing while the I/O operation is carried out. Because each I/O processor can transfer information directly to or from memory without central processor involvement (i.e., 'Direct Memory Access -DMA'), the central processor is able to perform internal processing concurrently with I/O operations. This overlap of I/O processing and internal processing guarantees that maximum use is made of the central processor, and increases overall system throughput.

1.9.3 AUTOMATIC DATA COMPACTION

To conserve disk storage and hasten data transfer, the system provides an option to compress data records automatically before storing them on disk. In the compaction process, characters which are repeated three or more times in sequence are stored as a single character and a repetition count. Data compaction is performed automatically on all print files, and is performed on a data file if the "compressed records" option is specified when the file is

created. Compressed records are automatically expanded to their original format by the system when they are read back into memory, making the entire compaction process completely transparent to the user's software. Data compaction can reduce the disk storage requirements of many files up to 50% and contributes to improved performance by reducing the total number of characters which must be transferred between disk and memory for each record access.

1.9.4 AUTOMATIC PRINT SPOOLING

Print spooling is a technique by which a job scheduled for printing is temporarily stored in a disk file rather than being sent directly to the printer. The 'Print files' thus created on disk are placed in a print queue under the control of the system console. When a printer becomes available, each job is then printed in the order determined by the print queue. Print spooling has the dual benefit of freeing individual workstations from dependence upon printer availability, and enabling the printers to be efficiently scheduled.

In most installations, printers are a common resource shared by all users. If printers are not used efficiently, system performance can be seriously degraded. For example, a user at one workstation who wishes to print only a few pages could be held up for hours while another user is printing a lengthy report. To avoid this situation, the 2200VS system provides this automatic 'print spooling' feature. Print spooling is one function of the 'Data Management System', which in turn is a subset of the Operating System software.

1.9.5 BACKGROUND PROCESSING

Background processing is the automatic execution of batched lower-priority programs whenever there are no higher-priority programs being handled by the Operating System.

Although the 2200VS is designed primarily for interactive operation, it is possible to run jobs which require large amounts of

processor or I/O time, with a minimum amount of operator interaction, on a 'background' basis. Background jobs are run in a batch from the system conscle, rather than from the individual workstations. All workstations therefore remain available for interactive use even while a background job is running.

1.10 USER CONVENIENCE FEATURES

While they may serve as useful first criteria for evaluating a system, performance and throughput do not tell the whole story. A system must also be designed so that its users can make the most effective use of its facilities, without being forced to undergo a long and arduous learning process. The 2200VS is a user-oriented system which offers a multitude of convenience features that make it easier to use by programmers and non-programmers alike. Among these features are: a versatile data entry, file maintenance, and report generation facility; an interactive text editor for entering and editing source programs; an easy-to-use symbolic debug facility for program debugging; and an assortment of system utility programs, including sort, copy and link routines.

1.10.1 DATA ENTRY/FILE MAINTENANCE

Included in the 2200VS system software is a package of three programs designed to facilitate the creation and maintenance of data file, and the creation of reports based on such files. A setup utility permits the user to define a data file by specifying the types of data in each record of the file, and to design the screen display used to prompt an operator for information to be entered for each record. A data entry program can then be used to solicit operator input by displaying the defined prompts and accepting and validating entered data. A report utility, intended for use by management as well as programmers, provides great flexibility in the design of custom reports which present information from a data file in a useful and coherent format.

1.10.2 INTERACTIVE TEXT EDITOR

Program development is greatly facilitated on the 2200VS by an interactive text editor. With the editor, a programmer can create and modify program files interactively using any one of the supported languages. Entering program text is as easy as typing it into the display, and editing an existing program is equally simple with the many editing functions provided. Interactive program development permits programmers to work with maximum productivity in the development and maintenance of programs.

1.10.3 INTERACTIVE DEBUG FACILITY

In many cases, the process of identifying and correcting bugs in a program is more time-consuming than the writing process itself. To assist the programmer in this task, the 2200VS supports an easy-to-use interactive debug facility.

The 2200VS Debug Processor permits inspection of program code, and permits inspection and modification of data by memory address. In addition, an easy-to-use 'symbolic' debug feature is provided that displays sections of source code in a program 'window' on the workstation screen, and permits data values to be examined and modified by symbolic data name rather than by address. The 2200VS Debug Processor also includes facilities for examining and modifying internal registers and the Program Control Word. Breakpoints can be set in a program, and another feature allows the user to manually step through program execution.

1.11 ADDITIONAL SYSTEM UTILITIES

A variety of additional system utility programs are provided to support the general programming task. These include, among others, copy, sort and linker utilities. The versatile copy utility permits the user to copy a single program or data file, an entire library of

such files, or a complete disk volume. For data files, the copy utility provides an option to change the file organization from sequential to indexed or vice-versa. The sort utility provides high-speed sort and merge capabilities for both indexed and sequential files, with either fixed or variable-length records. The linker, finally, is used to link together two or more program modules into a single large program, and optionally to remove the symbolic debug information previously inserted in a program for debugging purposes. Other utilities include a translation utility which translates from EBCDIC to ASCII and vice-versa; a special copy utility which copies and automatically translates 2200 program and data files to 2200VS format (and vice-versa); and a display utility, which can be used to display and/or print printer files.

The following list contains the names and descriptions of all 2200VS system utility programs. A detailed description of each system utility is documented in the 2200VS File Management Utilities Manual, WL# 800-1300FM.

ASSEMBLE - Assembles a source program written in 2200VS macroassembler language.

BASIC - Compiles a program written in 2200VS BASIC.

COBOL - Compiles a program written in 2200VS COBOL.

CONTROL - Used to define attributes and validation criteria for a data file.

COPY - Copies files, libraries, or entire volumes from one location to another.

COPY 2200 - Copies and automatically converts files from 2200 standard format to 2200VS format, and vice versa.

DATENTRY - Used to create and update data files.

DISKINIT - Initializes a new disk volume in 2200VS format, with a volume label and Volume Table of Contents.

DISPLAY - Displays the contents of a file on the work station screen.

DUMP - Produces a printed copy of a task dump previously written to diskette with the DUMP AND CANCEL function of the Debug Processor.

EDITOR - Used to enter and edit source program text.

EZFORMAT - Used to create display files for formatting the work station screen.

LINKER - Combines two or more program modules into a single executable program.

LISTVTOC - Produces complete or selective listings of a specified volume's Table of Contents, and examines the VTOC for errors.

PRINT - Prints the contents of a print file.

REPORT - Used to produce customized reports from a data file.

RPG II - Compiles source programs written in 2200VS RPG II.

SORT - Sorts a data file, with an optional capability to merge two or more sorted files.

TRANSL - Automatically translates the contents of a specified file from EBCDIC to ASCII (the code used internally by the 2200VS), or vice versa.

1.12 FILE PROTECTION AND SECURITY

All disk and tape files on the 8300 are classified according to a flexible file protection and security system, tailored at each installation to suit the requirements of the specific applications in use. At each installation, the file protection and security system is under the direct control of the Security System Administrators. The Security System Administrators are specially recognized users who determine the meaning and use of the file protection classes. They are able to access all files on the system, including the System User List and the Special Privilege Program List.

Every program, procedure, and data file on the system can be placed in one of twenty-eight file protection classes. Classes A through Z are used to represent protection classes whose meanings are determined by the Security System Administrators. For example:

Class W - The Workorder File
Class P - The Product File
Class C - The Customer File
Class Q - The Sales Quota File
Class R - The Pension Administration File
Class M - The Payroll File
Class X - Proprietary Programs and Procedures
Class D - Confidential Project Documentation Files

Classes "#" and " " (blank) are reserved for specific uses:

Class " " - Unprotected Files
Class "#" - Private and Security System Administration Files

The class of unprotected files is specified by setting the file protection class to blank. An unprotected file can be accessed by any user of the system. Class "#", unlike the other file protection classes, is used to define one protection class for each user. When specified, Class "#" identifies those files which can be accessed only by the user who created them (and by the Security System Administrators).

Before any user of the system can access a protected file, he must identify himself using the Logon command. At Logon Time, by lookup in th System User List, the user's Logon-ID and Password are validated, and his "access rights", relative to the defined file protection classes, are determined. The access rights are listed in the System User List (for each file protection class) to specify three different levels of privilege in order of increasing responsibility:

1) Execute Only Access (EXEC)
2) Execute and Input Access (READ)
3) Execute, Input, Update, Rename, Scratch and Debug Access (WRITE)

These access rights are checked whenever a user attempts to execute a program or procedure, whenever he attempts to open an existing file, and whenever he attempts to rename or scratch a file.

1.13 RELIABILITY

To ensure the integrity of information stored in memory and on external storage devices (disk or tape), the system provides automatic error detection and correction facilities. In physical memory, all single-bit errors are corrected automatically, while multi-bit errors cause an error indication. Similar checks also are performed on information stored on disk or tape.

1.14 HARDWARE - GENERAL

This section describes the Central Processing Unit, I/O Processors, available options, and peripheral devices for the WCS/60 and WCS/80.

1.14.1 THE CENTRAL PROCESSING UNIT (CPU)

The CPU is the central component of the WCS/60 and WCS/80 systems. It is a compact cabinet which houses the 2200VS computer (including main memory and I/O Processors) and the system diskette drive.

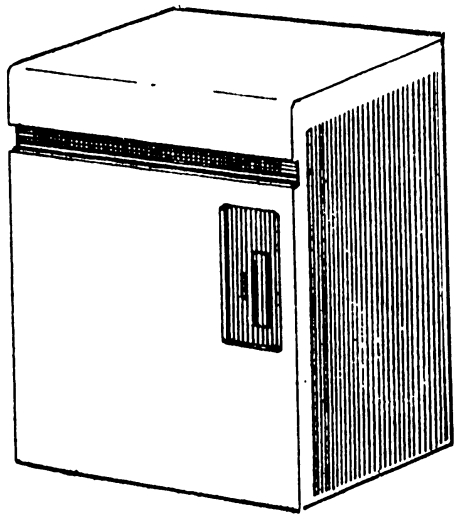


FIGURE 1-2 CENTRAL PROCESSING UNIT (CPU)

THE WCS/60 CPU

The WCS/60 CPU contains the 2200VS computer, the 2270V 308,000-byte diskette drive, and two IOP's:

1) A 22V01 Printer/Workstation IOP, which supports one printer and up to three workstations.
2) A 22V02 Diskette/10 Megabyte Disk IOP, which supports the 2270V system diskette drive and up to three 2260V 10 Megabyte Fixed/Removable Disk Drives.

Up to four additional IOP's support other peripherals and communications options.

Minimum memory for the WCS/60 is 64K bytes. Memory upgrades are available in increments of 64K, to a maximum of 256K bytes.

THE WCS/80 CPU

The WCS/80 CPU contains the 2200VS computer, a 2270V 308,000-byte diskette drive, and three IOP's:

1) A 22V01 Printer/Workstation IOP, which supports one printer and up to three 2246P workstations.
2) A 22V02 Diskette/10 Megabyte Disk IOP, which supports the 2270V system diskette drive and up to three 2260V 10 megabyte fixed/removable disk drives.
3) A 22V04 Removable Disk Pack Disk IOP, which supports a combined total of four 2265V-1 75 megabyte disk drives, and/or 2265V-2 288 megabyte disk drives.

Up to five additional IOP's support other peripherals and communications options.

1.14.2 I/O PROCESSORS (IOP'S)

I/O Processors control the operations of peripheral devices. The following IOP's are available:

22V01 Printer/Workstation IOP.
Supports one printer up to three workstations.

22V02 Diskette/10 Megabyte Disk IOP.
Supports one 2270V 308,000 Byte Diskette Drive and up to three 2260V 10 Megabyte Disk Drives.

22V04 75 Megabyte Removable Disk Drive IOP.
Supports two 2265-1 75 Megabyte Removable Disk Drives. (Used only for WCS/60; only one 22V04 IOP per system is allowed.)

22V05 9 Track Tape Drive IOP.
Supports up to four 2209V 9 Track Magnetic Tape Drives.

22V06 Communications IOP.
Available in two models to support bisynchronous telecommunications in the following combinations.
22V06-1 - Supports one bisynchronous line.
22V06-2 - Supports two bisynchronous lines.

1.14.3 WORKSTATION

The 2246P workstation is the means by which users communicate with the system.

FIGURE 1-3 2246P PARALLEL WORK STATION

The workstation consists of a display screen and keyboard. The 12-inch diagonal CRT display screen has a total display capacity of 1920 characters (24 rows, with 80 characters per row). Characters can be displayed in bright or dim intensity, and the screen can be formatted into 'fields'. The keyboard contains the familiar typewriter-like arrangement of alphabetic, numeric, and special character keys. A strip of 16 Program Function keys is placed along the top of the keyboard.

Workstations can be attached locally or remotely. A local workstation may be a maximum distance of 250 ft. from the CPU. A remote workstation may be located anywhere adjacent to a telephone line; it communicates with the CPU via the 22V06 communications IOP. (See the discussion of the communication IOP at the end of this chapter.)

1.14.4 DISKS

In addition to the system diskette drive, three hard-disk models are available, ranging in storage capacity from 10 megabytes to 288 megabytes.

2270V SYSTEM DISKETTE DRIVE

The 2270V diskette drive, mounted in the Central Processing Unit, holds a single removable diskette.

Each diskette has a storage capacity of 308,000 bytes. Diskettes can be used to store programs or small, transient data files; they also serve as a convenient means of transferring information between two WCS systems, or between a WCS system and a foreign computer system. In addition, all updates to the WCS/60 and WCS/80 system software made by Wang Laboratories will be provided on a diskette.

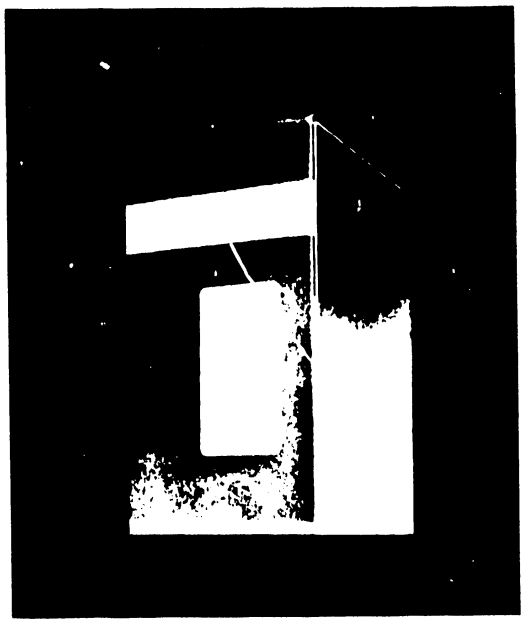


FIGURE 1-4 2270V SYSTEM DISKETTE DRIVE

2260V TEN-MEGABYTE FIXED/REMOVABLE DISK DRIVE

The 2260V provides an approximate storage capacity of 10 million bytes, equally divided between a fixed and removable platter.

Because the disk unit contains both a fixed platter and a removable cartridge, backup operations can be performed easily. Cartridges containing backup file copies or information which is not currently needed can be stored "off-line" and remounted in the disk unit as required.

FIGURE 1-5 2260V FIXED/REMOVABLE DISK DRIVE
10 megabytes

2265V REMOVABLE DISK PACK DISK DRIVE

The 2265V is a high-performance, high-capacity disk unit which provides fast access to large volumes of information.

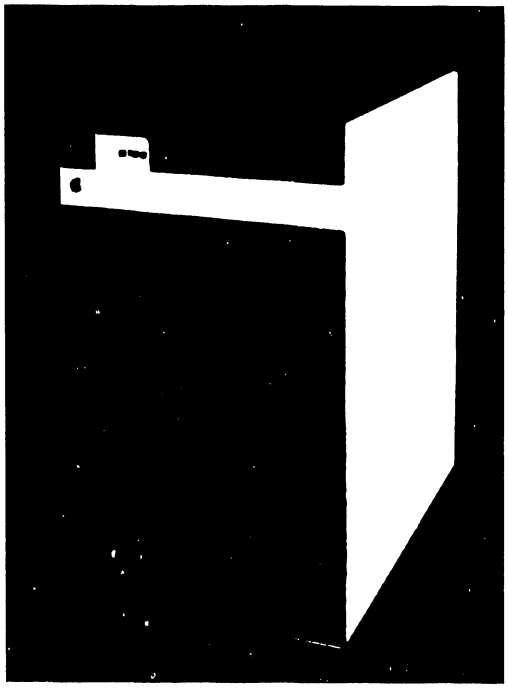


FIGURE 1-6 2265V-1, -2 REMOVABLE DISK DRIVE
75 megabytes (2265V-1)
288 megabytes (2265V-2)

The 2265V is available in two models:

1) The 2265V-1, with a storage capacity of 75 million bytes.
2) The 2265V-2, with a storage capacity of 288 million bytes.

Each model holds a single, removable disk pack. In addition to its high speed and large storage capacity, each model of the 2265V also provides extensive automatic error checking and correction facilities to ensure greater reliability.

On the WCS/60, a maximum of two 2265V-1 75 megabyte drives are allowed. The 2265V-2 288 megabyte drive is not supported on the WCS/60.

On the WCS/80, a combined maximum of eight 2265V disk drives are allowed (2265V-1 and/or 2265V-2).

1.14.5 PRINTERS

A variety of different printers are available for the WCS/60 and WCS/80, offering different speeds and print types. There is no restriction on the printer models which may be attached to a particular system.

2221V MATRIX PRINTER

The 2221V is a versatile matrix character printer. Characters are formed using a 9 x 7 dot matrix (for some characters, a larger 9 x 9 matrix is used for better detail).

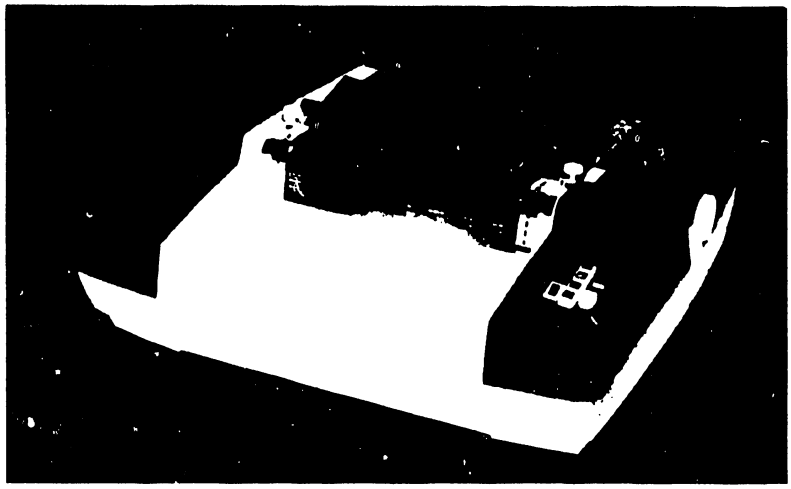


FIGURE 1-7 2221V MATRIX CHARACTER PRINTER
200 characters per second

The 2221V provides a character set of 96 characters, including upper and lowercase and special characters. Multipart forms with up to four carbons plus an original can be handled. Paper in widths from 3 1/2 inches (8.9 cm) to 14.9 inches (37.2 cm) can be mounted. The printer provides automatic vertical formatting, programmable audio alarm, and an expanded-print capability.

The 2221V prints serially at 200 characters per second. The number of lines per minute actually printed varies, according to the line length, from 65 to about 300 lines per minute.

2231V MATRIX PRINTER

The 2231V is an economical matrix printer which offers many of the features of the 2221V, but with a somewhat slower printing speed.

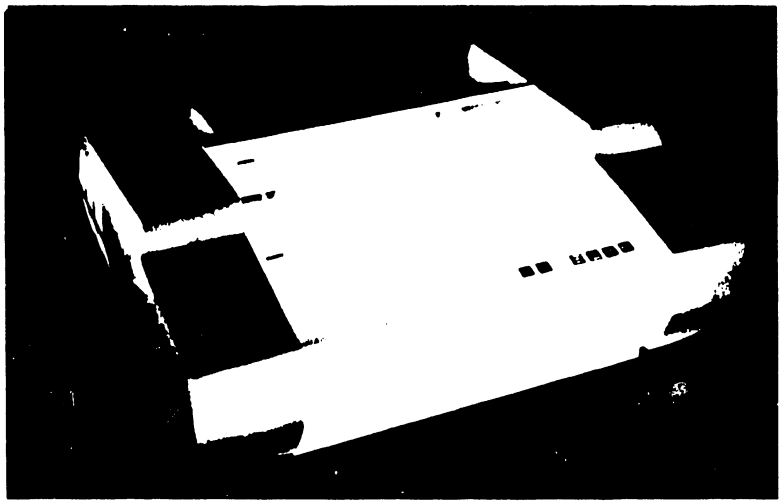


FIGURE 1-8 2231V-1, -2 MATRIX CHARACTER PRINTER
120 characters per second

The 2231V provides a full 96-character set, including upper and lowercase and special characters, using a 7 x 9 dot matrix to form each character. The 2231V is available in two models: the 2231V-1 prints a 112-character line (10 pitch); the 2231V-2 prints a 132-character line (12 pitch). Multipart forms and variable paper widths can be handled, and an audio alarm and expanded print capability are provided (see the 2221V).

The 2231V prints serially at a rate of 120 characters per second. The actual printing speed varies, according to the line length, from 45 to about 250 lines per minute.

2261V MATRIX LINE PRINTER

The 2261V printer is a high-performance matrix line printer which produces quality output at much higher speed than serial printers.

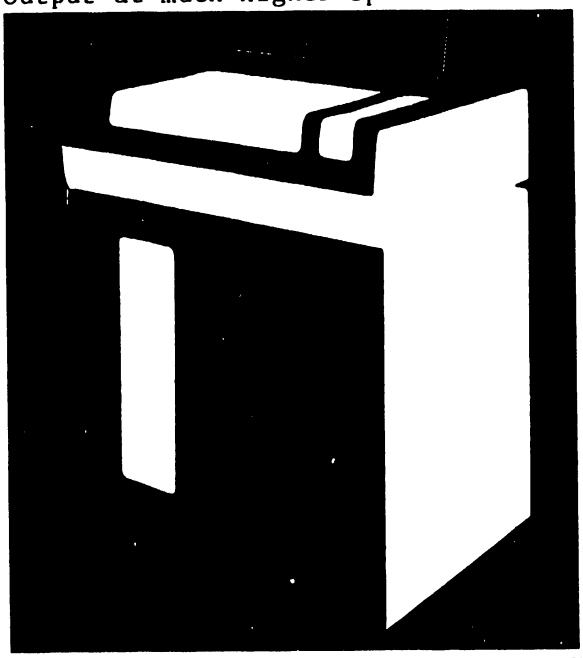


FIGURE 1-9 2261V MATRIX LINE PRINTER
240 lines per minute

The 2261V produces high-quality output using a 9 x 8 dot matrix to form each character. (An 11 x 8 matrix is used for some characters to obtain better detail.) A complete set of 96 characters, including upper and lowercase and special characters, can be printed.

A switch-selectable pitch features enables you to switch from 10 pitch (132-character line) to 12 pitch (160-character line). Line density is also switch-selectable at either 6 line per inch or 8 lines per inch.

An original and up to four carbon copies can be printed, with paper width varying from 3 1/2 inches (8.9 cm) to 14.9 inches (37.8 cm).

The printer offers a number of other useful features, including automatic formatting, expanded print, and a programmable audio alarm.

The 2261V prints bidirectionally, using four matrix impacter printing heads to achieve a print speed of 240 lines per minute, independent of line length.

2263V LINE PRINTER

The 2263V is a solid-character line printer which produces quality printed output at high speed. The 2263V is available in two models: the 2263V-1, with a printing speed of 400 lines per minute; and the 2263V-2, with a printing speed of 600 lines per minute.

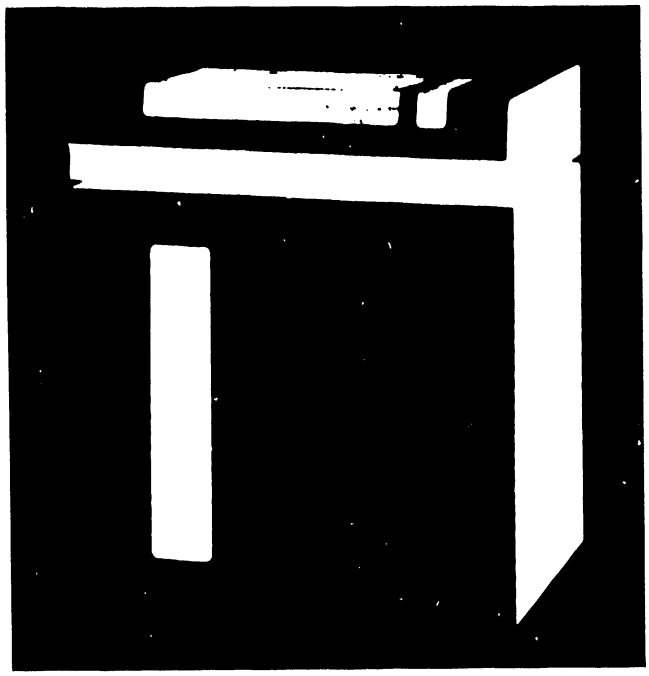


FIGURE 1-10 2263V-1, -2 LINE PRINTER
400 lines per minute (2263V-1)
600 lines per minute (2263V-2)

The 2263V prints one entire line (up to 132 characters) at a time. It can print one original and up to five carbon copies. Paper widths from 3.5 inches (8.9 cm) to 19.5 inches (48.8 cm) can be handled. The printer provides a number of useful features, including an automatic paper puller, static eliminator, and programmable audio alarm. Different typefaces and special character sets (including foreign language character sets) are optionally available.

2281V WHEEL PRINTER

The 2281V produces typewriter-quality output at 30 characters per second.

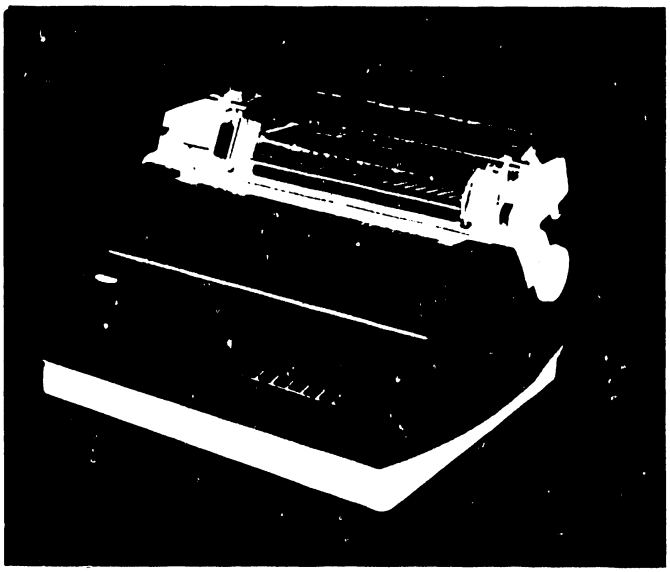


FIGURE 1-11 2281V WHEEL PRINTER
30 characters per second

The 2281V is a bidirectional output writer which utilizes a daisy character wheel with an 86-character set (upper/lowercase and special characters). Character wheels are removeable/replaceable for changing character sets.

The 2281V prints either a 132 or 158-character line at 30 characters per second. Among its features are programmable character underscoring, format tabbing, and color print selection.

1.14.6 TAPE DRIVE

2209V NINE-TRACK TAPE DRIVE

The 2209V Nine-Track Magnetic Tape drive is particularly useful for transferring information between a WCS/60 or WCS/80 and other computer systems.

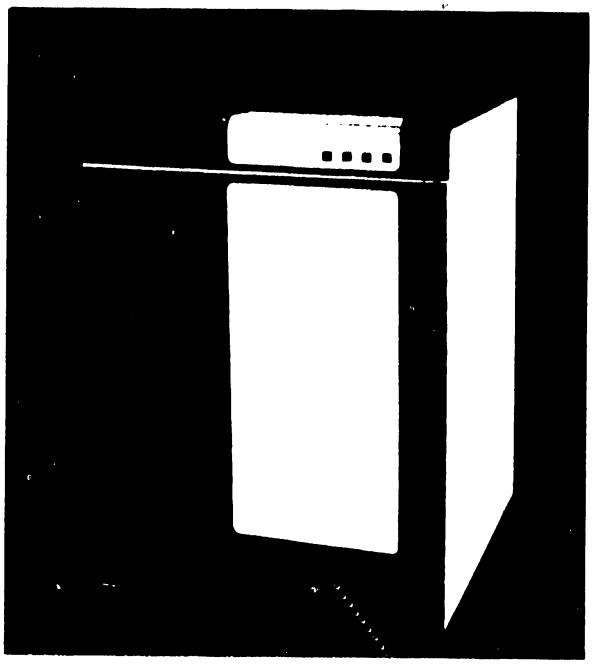


FIGURE 1-12 2209V NINE-TRACK MAGNETIC TAPE DRIVE

The 2209V can read from or write to any 1600 bytes-per-inch (bpi) Phase Encoded magnetic tape. It supports both ASCII and EBCDIC character codes.

The Model 2209V operates in the Phase Encoded mode at a density of 1600 bpi. The drive transports tape at a velocity of 75 inches per second during read and write operations, and up to 200 inches per second during rewind. The unit contains a dual-gap read/write head, full width erase head, tape cleaner, and photo electric sensors to detect reflective tape markers and tape breakage. The Model 2209V provides read-after write verification and automatic correction for single track errors, and for multiple track errors which can be reduced to a single track.

1.14.7 COMMUNICATIONS

22V06 COMMUNICATIONS IOP

The 22V06 Communications IOP is an I/O Processor which supports bisynchronous communication with a variety of line speeds and industry standard protocols. It is available in two models:

1) The 22V06-1 supports a single bisynchronous line.
2) The 22V06-2 supports two bisynchronous lines.

A number of different line speeds are available including 1200, 2400, 4800, and 9600 baud. At least one line can support an automatic calling unit.

A number of industry standard protocols are supported for bisynchronous transmission:

1) 2780/3780 emulation.
2) 3270 emulation.
3) HASP.

On the 22V06-2, each line is independently programmed; thus it is possible to run different protocols on separate lines concurrently from the same IOP.

Remote 2246P workstations can be attached to the system via the 22V06 IOP. Such workstations function as local workstations, enabling users in remote locations to communicate directly with the system and interactively access all system facilities, just as if they were logged on locally.

Wang Computer Systems

Technical Information

DEVICE	GENERAL SPECIFICATIONS	PERFORMANCE SPECIFICATIONS	PHYSICAL SPECIFICATIONS	ENVIRONMENTAL REQUIREMENTS
Model 2246P Parallel Workstation	CRT • 12 inch diagonal display screen • 24 lines, 80 characters per line • Screen can be formatted into discrete fields • Manual controls for contrast and brightness • Bright and dim intensity for instant recognition of field modifiability • User-programmable blinking cursor for next-character position • Anti-glare filter Keyboard • Typewriter-like keyboard • Cursor positioning keys • Character insertion and deletion keys • Numeric keypad • 16 Program Function keys • Programmable Audio Alarm IOP – 22V01	• 2K Random Access Memory	• Physical Dimensions Height – 13 in. (33 cm) Width – 19.5 in. (49.5 cm) Depth – 19.6 in. (49.5 cm) • Weight 40 lb (18.2 kg) • Heat Dissipation 427 BTU/hr • Cable 500 ft maximum	• Power Requirements 115 or 230 VAC (±10%) 50 or 60 Hz (±1 Hz) 125 watts • Operating Environment 50°F to 90°F (10°C to 32°C) 35% to 65% relative humidity
Model 2260V Fixed/Removable Disk	• 10 megabytes/drive • One fixed 5-megabyte platter and one removable 5-megabyte platter per drive for convenient backup and off-line storage • 408 tracks per platter • IOP – 22V02	• Rotational Speed 2400 RPM • Average Access Time 35 ms • Average Latency Time 12.5 ms • Track-to-Track Head Positioning Time 9 ms • Data Transfer Rate 312 kilobytes/sec	• Physical Dimensions Height – 32.5 in. (82.6 cm) Width – 17.5 in. (44.5 cm) Depth – 29 in. (73 cm) • Weight • Heat Dissipation 2043 BTU/hr	• Power Requirements 115 or 230 VAC (±10%) 50 or 60 Hz (±1 Hz) 800 watts start up 425 watts running • Operating Environment 50°F to 95°F (10°C to 35°C) 20% to 80% relative humidity
Model 2265V-1 75-Megabyte Removable Disk Pack Disk Drive	• 75 megabytes/drive • 5 usable surfaces/pack • 823 cylinders/pack • IOP 22V03 for WCS/60 22V04 for WCS/80 • Restriction: Maximum of two drives on WCS/80	• Rotation Speed 3600 RPM • Average Access Time 30 ms • Average Latency Time 8 ms • Track-to-Track Heat Positioning Time 6 ms • Data Transfer Rate 1.2 megabytes/sec	• Physical Dimensions Height – 41 in. (104 cm) Width – 24 in. (61 cm) Depth – 36 in. (91 cm) • Weight 500 lb (227 kg) • Heat Dissipation 2680 BTU/hr	• Power Requirements 110 VAC (±10%) 50 or 60 Hz (±1 Hz) 40 amp start up 8.2 amp operating 1.5 amp standby • Operating Environment 60°F to 90°F (15°C to 32°C) 35% to 65% relative humidity
Model 2265V-2 288-Megabyte Removable Disk Pack Disk Drive	• 288 megabytes/drive • 19 usable surfaces/pack • 823 cylinders/pack • IOP: 22V04 • Restriction: Only for WCS/80	• Rotational Speed 3600 RPM • Average Access Time 30 ms • Average Latency Time 8 ms • Track-to-Track Heat Positioning Time 6 ms • Data Transfer Rate 1.2 megabytes/sec	• Physical Dimensions Height – 41 in. (104 cm) Width – 24 in. (61 cm) Depth – 36 in. (91 cm) • Weight 550 lb (250 kg) • Heat Dissipation 3980 BTU/hr	• Power Requirements 208 VAC (±10%) 50 or 60 Hz (±1/2 Hz) 40.0 amp startup 8.0 amp operating 2.0 amp standby • Operating Environment 60°F to 90°F (15°C to 32°C) 35% to 65% relative humidity
2209V Magnetic Tape Drive	• 25 megabyte capacity (with 2k blocks) • Can hold up to 2400 ft (731.5 m) of tape (standard thickness) • Records at 1600 bpi phase encoded • Holds a standard 10.5 in. (26.7 cm) reel • Dual-gap read-after-write head IOP: 22V05	• Read/Write Speed 75 inches per second • Rewind Speed 200 inches per second • Data Transfer Rate 120 kilobytes per second	• Physical Dimensions Height – available with integral storage cabinet in heights from 34 in. to 84 in. (83 cm to 182.2 cm) Width – 24 in. (61 cm) Depth – 26 in. (66 cm) • Weight 170 lb (77.4 kg) • Heat Dissipation 1623 BTU/hr	• Power Requirements 115 VAC (±10%) 60 Hz ±1 Hz 4.1 amp • Operating Environment 60°F to 90°F (15°C to 32°C) 35% to 65% relative humidity, non-condensing

DEVICE	GENERAL SPECIFICATIONS	PERFORMANCE SPECIFICATIONS	PHYSICAL SPECIFICATIONS	ENVIRONMENTAL REQUIREMENTS
Model 2221V **Matrix Character Printer**	• High quality 9 x 9 and 9 x 7 dot matrix impact printer • Expanded print capability • Up to five-part forms • Up to 132 characters/in • Full ASCII set of 96 characters • Full line buffering • Programmable Audio Alarm • 14.9 in. (37.8 cm) maximum forms width • 3 channel vertical format unit • Bottom load paper feed • IOP: 22V01	• Print Speed 200 characters per second 65 to 300 lines per minute, depending upon length	• Physical Dimensions Height – 12 in. (31 cm) Width – 29 in. (74 cm) Depth – 25 in. (64 cm) • Weight 85 lb (38.6 kg) • Heat Dissipation 1025 BTU/hr	• Power Requirements 115 VAC (±10%) 50 or 60 Hz (±1 Hz) (10°C to 32°C) 35% to 65% relative humidity, non-condensing
Models 2231V-1 and 2231V-2 **Matrix Character Printers**	• High-quality 7 x 9 dot matrix impact printer • Expanded print capability • Model 2231V-1 prints 112 character line • Model 2231V-2 prints 132 character line • Full ASCII set of 96 characters • Up to 5 part forms • Full line buffer • Programmable Audio Alarm • Manual line feed • 2 channel vertical format unit IOP: 22V01	• Print speed 120 characters per second 45 to 250 lines per minute, depending upon length	• Physical Dimensions Height – 10 in. (25 cm) Width – 24 in. (61 cm) Depth – 18 in. (46 cm) • Weight 68 lb (31 kg) • Heat Dissipation 478 BTU/hr	• Power Requirements 115 or 230 VAC (±10%) 50 or 60 Hz (±1 Hz) 140 watts • Operating Environment 50°F to 90°F (10°C to 32°C) 35% to 65% relative humidity
Model 2261V **Matrix Line Printer**	• High Quality 11 x 8 and 9 x 8 dot matrix impact printer • Expanded print capability • Up to 5-part forms • 132 character line (10 pitch format) or 160 character line (12 pitch format) • Full ASCII set of 96 characters • Line density switch selectable at 6 lines/inch or 8 lines/inch • Full line buffering for faster throughput • 14.9 inch (37.8cm) maximum forms width • Automatic formatting • Programmable Audio Alarm • 4 matrix impacter heads, bidirectional printing • 3 channel vertical format unit • IOP: 22V01	• Print Speed 240 lines per minute, independent of characters-per-line and pitch	• Physical Dimensions Height – 36 in. (91cm) Width – 27 in. (68.6cm) Depth – 26 in. (66cm) • Weight 210 lb (94.5kg) • Heat Dissipation 1572 BTU/hr	o Power Requirements 115 or 230 VAC ±10% 50 or 60 Hz (±1 Hz) 460 watts o Operating Environment 50°F to 90°F (10°C to 32°C) 35% to 65% relative humidity, non-condensing
Models 2263V-1 and 2263V-2 **Line Printers**	• Chain Printer • Removable print character links • Optional character sets and foreign language type • Gothic print set • 132 characters per line • 64 ASCII character set (upper case) • Up to 6-part forms • Full-line buffering • 3.5 inch to 19.5 inch paper width • 8-channel vertical format unit • Static eliminator • Automatic paper puller • Programmable Audio Alarm • Precise form positioning • Off-line test capability • Diagnostic panel • IOP: 22V01	• Print Speed – 400 lines per minute for Model 2263V-1 – 600 lines per minute for Model 2263V-2 • Single Line Advance Speed 20 ms • Slew Speed 20 in./second	• Physical Dimensions Height – 42 in. (106.7cm) Width – 36.5 in. (92.7cm) Depth – 32 in. (81.3cm) • Weight 570 lb (258.5kg) • Heat Dissipation 2700 BTU/hr	• Power Requirements 115 or 230 VAC (±10%) 50 or 60 Hz (±1 Hz) 690 watts o Operating Environment 40°F to 95°F (4.4°F to 35°C) 35% to 65% relative humidity, non-condensing

DEVICE	GENERAL SPECIFICATIONS	PERFORMANCE SPECIFICATIONS	PHYSICAL SPECIFICATIONS	ENVIRONMENTAL REQUIREMENTS
Model 2281V Wheel Printer	• Daisy character wheel impact printer • Removable character wheel and interchangeable character sets • Typewriter-like print registration • 86 ASCII character set, both upper and lower case • 132 characters per line (10 pitch format), 158 characters per line (12 pitch format) • Black/red ribbon cartridge • Full-line buffering • Programmable character underscoring • Format tabbing • Top of form switch • Pin feed forms tractor mechanism (optional) • Adjustable platen • IOP: 22V01	• Print Speed 30 characters per second	• Physical Dimensions Height – 14 in. (35.6cm) Width – 24 in. (61cm) Depth – 22 in. (55.9cm) • Weight 37 lb (16.8kg) • Heat Dissipation 850 BTU/hr	• Power Requirements 115 or 230 VAC (±10%) 50 or 60 Hz (±1 Hz) 250 watts • Operating Environment 45°F to 95°F (7°C to 35°C) 35% to 65% relative humidity, non-condensing • Duty Cycle Medium-average actual printing time – up to 4 hours per day
Model 22V06 Communications IOP	• 22V06-1 supports 1 bisynchronous line • 22V06-2 supports 2 bisynchronous lines • Protocols supported: 1) 2780/3780 emulation 2) 3270 emulation 3) remote 2200VS workstation 4) HASP • At least one bisynchronous line supports automatic calling unit	• 1200, 2400, 4800 or 9600 band	• Requires one IOP slot in CPU	

SECTION 2 SYSTEM CONCEPTS

SECTION 2

SYSTEM CONCEPTS

2.1 2200VS - VIRTUAL MEMORY

In small computers with, for instance, 4K of RAM, a programmer often overcomes the physical RAM size limitations with program overlays and small disk records. The smaller the physical memory, the more frequently the disk must be used. Software required to support disk operations adds to program overhead and increases the chance of programming errors. Changing physical memory size causes further problems by necessitating changes ir software.

In a virtual memory system, the larger the physical memory, the faster programs can be run. Fewer disk operations are required.

We can explain the above points by example:

2.1.1 A COMPARISON TO EXISTING 2200's

DIM A$(20,20)32, sets up a four hundred (20 x 20) element array (each element 32 characters, max.), using 12.8K bytes of memory (in a 2200 with more than 12K of RAM). On a 4K Model 2200, to accomplish the same, DIM A$(20)32 would be used, adding at least twenty disk operations to keep the array loaded with currently needed data. These extra disk operations must be written into the existing software.

A virtual memory computer does all the disk overlay work for the programmer. If the 4K Model 2200 referenced here had virtual memory, the 12.8K array dimension would work; the excess being put on a disk scratchpad. Whenever an array reference was made beyond the physical memory limits, the 2200 CPU would exchange a portion of current physical memory with a portion of disk data, and then proceed. The user would not be aware of the true memory size, or the number of overlays. The same would be true for long program text.

Although the user is 'not aware' of the specific functions involved in providing him with a virtual memory, larger than the physical memory, an understanding of the paging mechanism is useful. Also, a general awareness of how paging is implemented may help any programmer write more efficient code for the machine.

2.1.2 RELATION OF VIRTUAL MEMORY TO PHYSICAL MEMORY

Physical memory of the 2200VS is limited to a maximum of 512K bytes. However, at the present stage of 2200VS developement, machine instructions can reference any of 1,310,720 one-byte virtual address locations.

'Translation' of virtual addresses to physical ('absolute') addresses is performed by routines of the Operating System. Three key information units recognized by the Operating System during translations are: 'segments', 'pages', and 'page frames'.

A segment is a block of contiguous one-byte virtual memory locations that begin on a decimal virtual address of zero, 1,048,576, or some multiple of 1,048,576. This start-point for each segment is commonly referred to as a '1 Meg Boundary'. Segments 1 and 2 are allocated to each user at LOGON time by the Operating System. Segment zero is shared by every user. Segment 0 begins at virtual address 0, ends at virtual address 262,143, and comprises 128 pages. Supervisory routines and data of the Operating System are in segment zero. Segment 1 for each user starts at virtual address 1,048,576, ends at virtual address 1,572,863, and comprises 256 pages. User programs are in segment one. Segment 2 for each user starts at virtual address 2,097,152, ends at virtual address 2,621,439, and also comprises 256 pages. User data is in segment two. References to segments 3 through 15, presently invalid, are treated as program errors. Note that there are gaps of nonaddressable virtual locations between segments. These gaps are defined as follows:

VIRTUAL ADDRESSES 262,144 - 1,048,575 (inclusive) constitute the gap between segment 0 and segment 1 This gap is somtimes called the 'segment 0 Non-addressable' area.

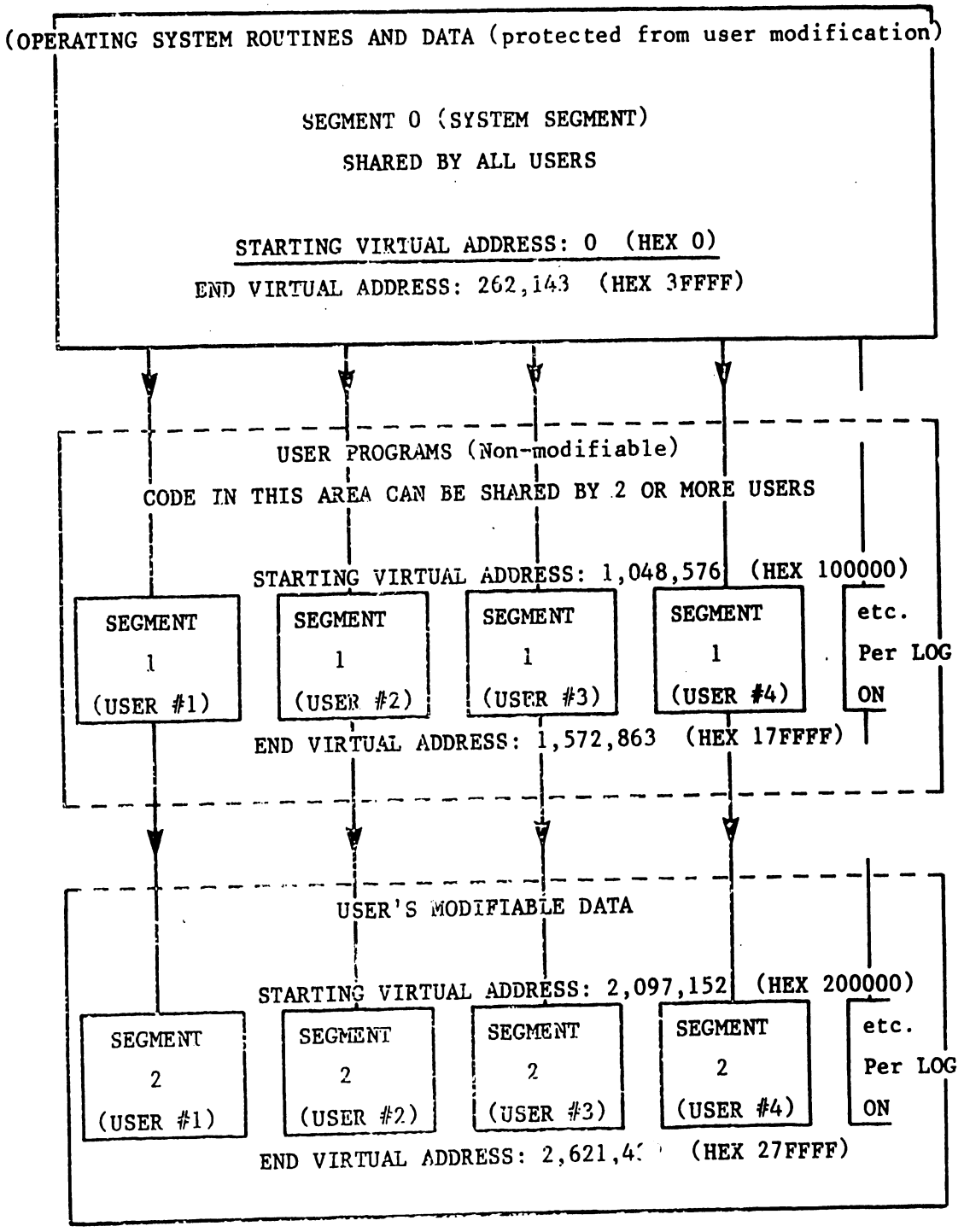
VIRTUAL ADDRESSES 1,572,864 - 2,097,151 (inclusive) constitute the gap between segment 1 and segment 2. This gap is sometimes called the 'segment 1 Non-addressable' area.

Each user's 1 Meg virtual address space (segments 1&2) is assigned to a unique disk space, even though the virtual addresses for every user are identical. Some additional element such as workstation number or task number is provided by the Operating System to identify each user's unique disk space (files).

A 2200VS machine instruction could reference any of 16,777,216 one-byte virtual memory locations; however, at the present stage of 2200VS development, only segments 0, 1, and 2 are allocated to each user at LOGON time, thus accounting for the addressability of only 1,310,720 locations.

The following diagram illustrates the present segment structure of 2200VS virtual memory. Note that this diagram bears no resemblance to physical memory; all blocks represent disk files:

FIGURE 2-1



NOTE:

In the above diagram, it can be seen that each user is allocated approximately 1 Meg of virtual memory space, once logged onto the system. Thus, with four users logged on (for example), the total requirements for virtual storage space is 4¼ Meg. The four-user system described in this example would probably require a 10 Meg hard disk to support minimal configuration demands.

A page is a block of 2,048 contiguous one-byte virtual memory locations that begin at an address of zero, 2,048, or some multiple of 2,048. This start-point for each page is commonly called a '2K boundary'. A page of virtual memory currently residing in physical memory is said to be 'framed', or can be called a page frame. Page frames, therefore, are 2K blocks of contiguous one-byte physical memory locations that begin at a physical (main) memory address of zero, 2,048, or some multiple of 2,048.

A 128K system would, for example, have sixty-four 2K page frames in physical memory. Certain page frames must be occupied by top-priority routines of the Operating System (i.e., the 'paging routines' and other Operating System 'control blocks'); such routines cannot be 'paged-out' of physical memory, and are said to be 'permanently resident', or 'permanently fixed'. Certain other page frames are considered 'temporarily resident' or 'temporarily fixed' if allocated for an I/O DMA*. As soon as the I/O DMA is complete, any page or pages allocated for that I/O DMA become 'replaceable'. Remaining page frames (also replaceable) can be occupied by any other User/Operating System routines and data, as required.

Pages of a program or data may be framed at any position in Physical Memory, and still be executed as if each page were adjacent. Pages need not be contiguous, since each one is linked, or 'threaded' to the next by an address pointer (a 3-byte entry in the Program Control Word**). This concept of initiating a program that begins at any page frame and randomly occupies any number of additional page frames is called 'relocatability'.

Programs which repetitively jump from one page to another will require more frequent disk access. The same is true for data references. A programmer who wants maximum execution speed will try to remain within one page frame as long as possible before branching elsewhere. This desirable quality in structuring programs is called 'locality of reference'.

---

* - I/O DMA: Input/Output Direct Memory Access

** - PCW: Program Control Word

Three 'local page tables' (0, 1, and 2), located in stack (CP LOCAL STORAGE), are also required in the virtual-to physical address translation process. There is one local page table (LPT) allocated for each segment in virtual memory. Each local page table contains one entry for every virtual page in its corresponding segment. Local page table entries are one byte long, and contain either the eight high order address bits of a physical page frame start boundary, or zero. If the entry is zero, the corresponding page is not currently framed; such an entry would thus be called 'invalid'.

At this point, to further understand how virtual addresses are translated into physical addresses, the 2200VS virtual address format must be explained. 2200VS virtual memory addresses are always in the following 24 bit form:

VIRTUAL SEGMENT FIELD	VIRTUAL PAGE FIELD	BYTE DISPLACEMENT FIELD
(Segment Index)	(Page Index)	(Byte Index)
4 bits	9 bits	11 bits

The hardware uses the segment index portion of the virtual address to select one of the three local page tables in the CP stack:

SEGMENT NO.:	SEGMENT INDEX (IN BINARY):
0	0000
1	0001
2	0010

The page index is used as a CP stack address, in order to select an entry from the local page table. Bit four indicates whether the virtual address is valid or illegal. Bits 5 - 12 select the 8-bit stack element:

PAGE TABLE NO.:	LPT ADDRESSES (IN HEX):	NUMBER OF TABLE ENTRIES
0	00-7F	128
1	00-FF	256
2	00-FF	256

In the translation diagram that follows (next page), one may note that when bit 12 of the virtual address is 0, the high order table element is selected; when bit 12 = 1, the low order table element is selected.

The byte index or 'displacement' is carried over to the physical address unchanged. ('Displacement' is discussed in greater detail in subsequent text of this section.)

When no error conditions ('exceptions') are encountered in the translation process, the page table entry and the byte index are 'concatenated', or joined, thus producing the full 19-bit physical memory address.

There are, of course, many combinations of the 13 page and segment bits from the virtual address that cannot be translated into an 8-bit physical page address. In general, when translation is impossible, one of the following errors will occur:

PAGE FAULT-	An error condition indicating that a valid, referenced virtual page does not currently occupy any page frame.
PROTECTION VIOLATION-	An error condition indicating that a write operation was attempted in either segment 0 or segment 1, or that a segment 0 access was attempted by a user.
ADDRESS EXCEPTION-	An error condition indicating that the virtual address referenced is invalid.

The above error conditions will, in turn, cause one or both of the following actions:

PROGRAM INTERRUPT-	The Operating System seizes control and halts processing of that task.
SUPPRESSED OPERATION-	The Operating System inhibits a particular operation, such as a write into a protected area.

An illustration of virtual-to-physical translation follows:

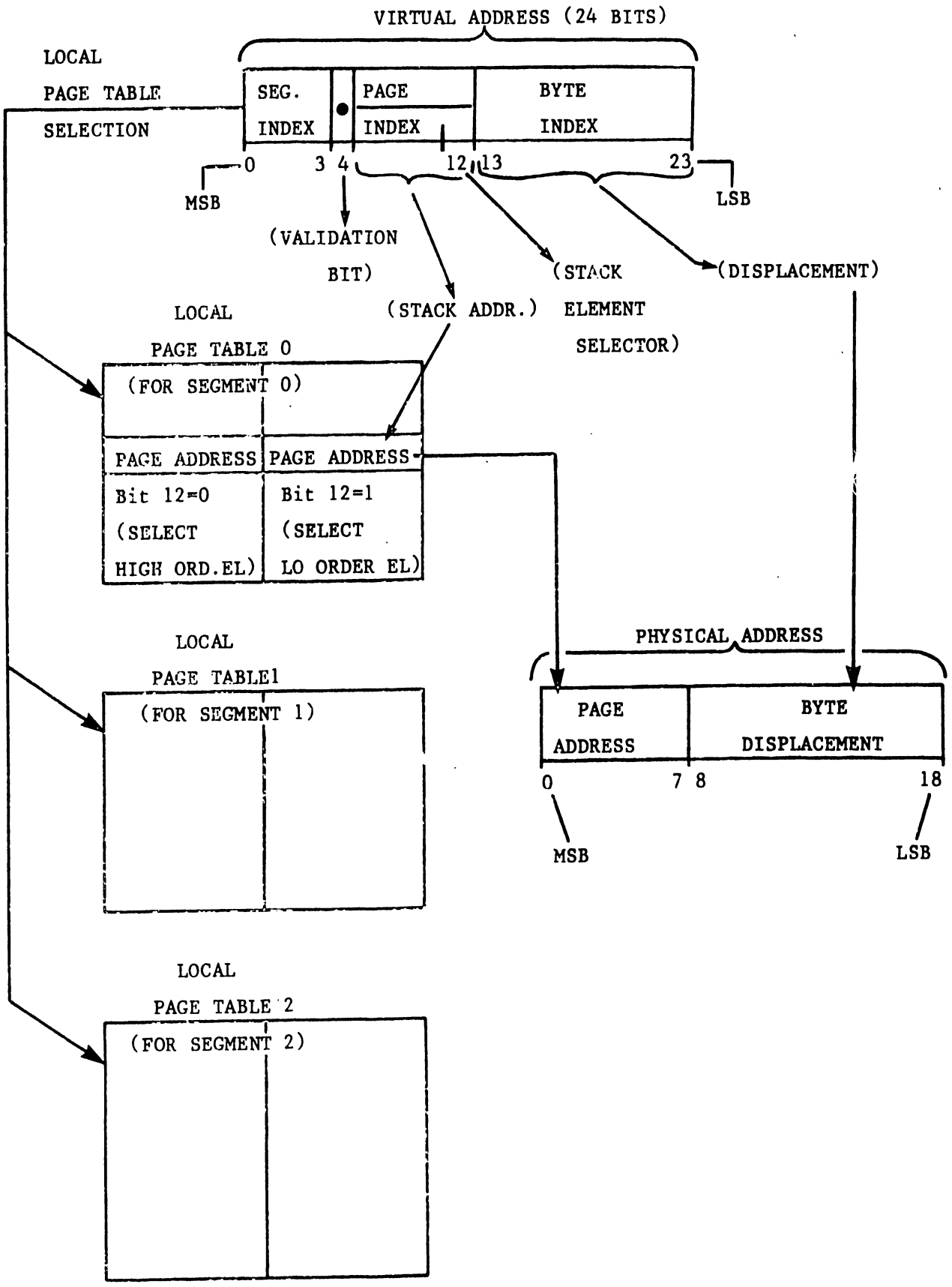
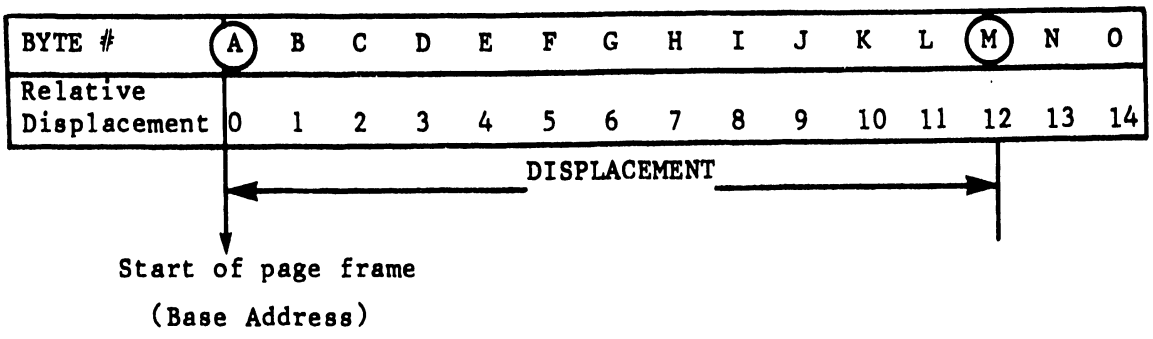


FIGURE 2-2

Note that LPT 0 is permanently resident, and is updated whenever a 'paging task'* is executed for the Operating System itself. Since each user has his own segments 1 and 2, each user also has a unique LPT1 and LPT2. At the end of a user's time slice, that user's LPT1 and LPT2 are swapped out of local storage ('stack') and the next user's LPT1 and LPT2 are brought in to the stack for the duration of his time slice. This swapping of LPTs allows a completely new set of page frames to be addressed from the same set of virtual addresses.

DISPLACEMENT:

Each address within a page frame must be specified relative to the starting, or 'base' address of that page frame. Each new address is 'displaced' a specified number of byte locations beyond the base address. Look at the following example.



In this example, byte A resides at the base address; byte M is displaced by 12 locations. Byte M's 'displacement' therefore equals 12.

The displacement is added to the base address, and the resulting sum corresponds to an 'absolute' or 'true' physical address in main memory. The displacement number, indicated by a field of 11 bits, has 2^{11} = 2,048 possible combinations, corresponding to the 2,048 byte locations per page frame. Displacement, therefore, actually indicates byte address within a page frame.

*Explained in subsequent text.

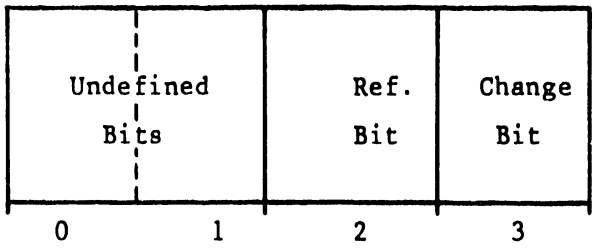
PAGE REPLACEMENTS:

Now, one must ask: "What happens if the next item referenced in a program is located on a page not currently framed"? Keeping track of which pages are framed is another function of the Operating System. Pages are brought into physical memory in a manner that can best be described as 'demand paging'.

When, during translation, a referenced page of memory is found to be missing from physical memory, a 'page fault exception' is reported to the Operating System, which in turn calls for a 'paging task'. The Program Interrupt Service first attempts to locate a page frame, the contents of which may be replaced with the required page from virtual memory (currently on disk). To aid in this determination of which page will be replaced, the 2200VS Operating System uses a 'Least Recently Used' (LRU) algorithm. If a page has not been referenced recently, that page will probably not be needed in the immediate future, and is therefore 'replaceable' according to the LRU. The LRU makes this determination based on an 'Age Count', which is maintained for each page frame.

Age count, and other page frame-related information is held permanently fixed in an area of main memory called the 'main memory Page Frame Table' (PFT; do not confuse with Local Page Tables). The main memory Page Frame Table is maintained as a 'control block' of the Operating System. Each page frame's age count is zeroed by a corresponding 'reference bit', every time that page frame is referenced. This reference bit is maintained in a 'Local Page Frame Table' (LPFT). The LPFT occuppies another portion of the CP stack. One four bit LPFT entry exists for each page frame. The paging routine uses the page address obtained from a Local Page Table to index into the Local Page Frame Table .

LPFT entries have the following page frame status information:



If a page frame has been modified by a write instruction to main memory (WTRAN), the 'change bit' (in the LPFT entry belonging to that page frame) is set. That page no longer matches its original form, still on disk. Thus, during a paging task, before replacing the contents of that page frame, the modified frame must first be rewritten on disk (paged out). This will update the contents of virtual memory. If a currently framed page has not been modified, that frame may be directly overwritten with a new page from disk. Scanning of the Main Memory PFT and the LPFT for 'age count' and 'change bit' status during the LRU routine is performed by the Scan Page Frame Table(s) (SPFT) instruction, which is used exclusively by the Operating System.

There are four 4-bit page frame table entries in each of the sixty-four 16-bit LPFT stack elements.

To summarize LPFT entries:

R-Reference bit

1 = page frame was not referenced by translation microinstruction

0 = page frame was referenced by translation microinstruction (not immediately replaceable)

C-Change bit

1 = contents of page frame was not changed

0 = contents of page frame was changed (page must be recopied to disk)

Once the LRU algorithm determines which page will be replaced, the page being searched by the current task (program) must be located and brought in from disk. To locate that page, the main memory PFT entry for the current (waiting) page frame supplies a three-byte pointer. This pointer is a virtual address, which, when translated, yields the location of a 'File Length and User Block' (FLUB), also in main memory. There is one FLUB held in main memory for each active file in the system. For example, if a 2200VS system has nine users logged on, and each user is executing a task, nine FLUB's are established in main memory, one for each user's task; other FLUBs are established for paging of active Operating System files. Each FLUB contains all primary information necessary to locate the active file on disk. The virtual page address (for the required page) is then used as an index or displacement into that file, thus enabling the missing page to be read and subsequently 'paged in' via I/O DMA to the page frame that has been reserved by the LRU algorithm. This accomplished, the proper LPT is updated, and the current task resumes execution.

Any user or Operating System task that needs a page from disk is forced to wait until the paging task has been completed, and the local and main memory page and page frame tables have been updated. Keeping track of which tasks are held up for 'page-in waits' is a function that is performed by another of the Operating System's control blocks, the 'Page Frame Semaphore Area' (PFSA). The PFSA is permanently fixed in main memory and holds: 1) the virtual page number required for page-in, 2) the number of the page frame that has been selected by the LRU, and 3) the FLUB address required for page-in. Note that item 3 is not used as the actual pointer to the FLUB.

The following flow diagram illustrates a typical paging operation:

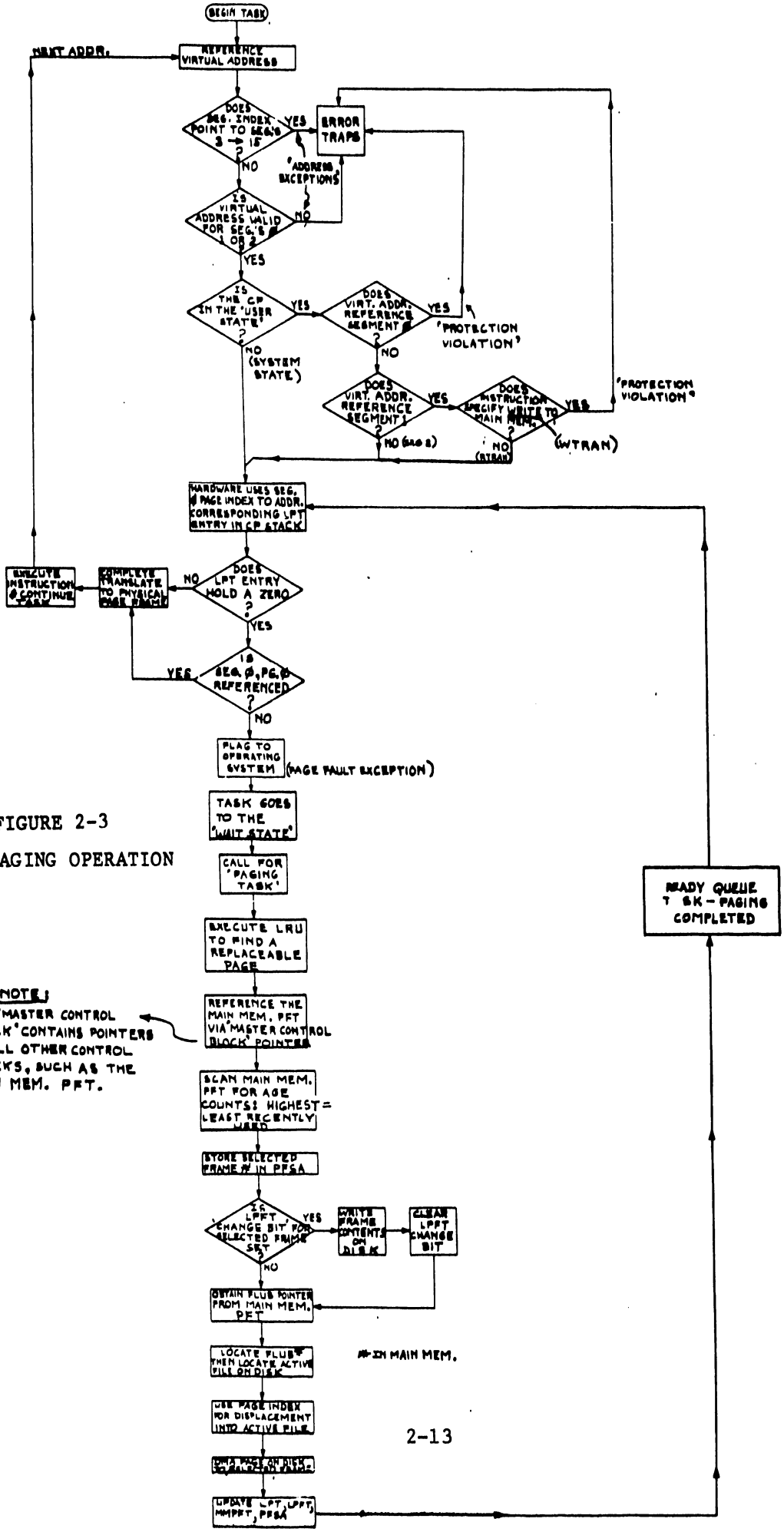


FIGURE 2-3

TYPICAL PAGING OPERATION

2.2 COMPILERS, INTERPRETERS, AND ASSEMBLERS

Any machine using high-level programming languages (BASIC, COBOL, RPGII, FORTRAN, etc.) cannot directly execute any statement written in that language. Each statement must first be transformed, by some means, into a series of executable machine language instructions.

2.2.1 COMPILERS

2200VS Languages are 'compiled'. In a compiler system, a program is first entered in one of the available high-level languages (COBOL or BASIC in the 2200VS) by interacting with a 'text editor' program. Entering text via 'text editor' is very similar to typing text into a word processing machine. The programmer may enter anything he wants. The text editor program causes the machine to blindly accept the user's program text, allowing him to edit his text on entry, and giving no indication of programming errors. This initial entry of high-level user program text is called the 'source program' (also 'source text' or 'source module'), and is not machine executable.

After text has been entered and edited, another function of the text editor program stores this source program on disk. Once on disk, the source program becomes a 'source file', still not machine executable.

Once the source file has been created, the programmer loads a 'compiler' program, written in machine language. The COBOL or BASIC compiler will use the source file as input to produce a 'compiled' version of the user's program in machine language, called the 'object program' or 'object code'. The compiler program then creates an 'object file' or 'program file' on disk, and then generates a printout of source code, object code, syntax errors, and other relevant information for the programmer's use. It is only this 'object' or 'program' file that can be executed on the system; source code cannot be executed.

In a compiler system such as the 2200VS, therefore, two versions of each program are produced: the source program, written in a high-level language; and the object program, consisting of machine language statements and produced by a compiler.

To run the object program, the object file is loaded and run. Errors of execution will show up at this time. On the 2200VS, error messages are in plain English, not number codes. When an error is encountered, the user program goes into a debug mode. Machine code, memory, and registers may be examined and changed. The program can be stepped, rerun, or cancelled.

On the 2200VS, because all programming languages ultimately compile to the same machine code, a program may have different portions written in different languages. It is not unusual to write a machine code routine for something that is not convenient in a high-level language. The various pieces are put together by a system program called a 'linkage editor' which works with the object files.

In a compiler system, a programmer will be more efficient if he makes flow charts and 'desk checks' his code in advance. Since a program is usually run many times after it is written, it is better to slow down the coding process and speed up run time.

2.2.2 INTERPRETERS

An Interpreter is a program that will translate high-level language program statements (source code), as encountered, directly into individual sets of executable machine instructions. An interpreter does not generate the object code for each source statement, and save that object code for later execution. The major disadvantage of an interpretive system is speed. An interpreter may generate results 10 to 20 times slower than the equivalent compiled code. For example, in an interpretive system, a statement describing a 1,000-pass loop must be retranslated and executed for each of the 1,000 passes in that loop. The interpreter is, however, superior to the compiler in that the executing program can easily be interrupted, changed, or resumed.

Most commercially available BASIC language systems, such as Wang's 2200C, S, or T use an interpreter program for execution of source code. A 2200 program would run much faster if all source code were compiled instead of being interpreted. In the loop example above, the loop would be translated to machine code just once, not 1000 times. However, the resulting machine code (object code) would produce the same end result, but much faster.

2.2.3 ASSEMBLERS

An 'assembler', also known as an 'assembly routine' or 'assembly program', is a program designed to convert a set of non-executable symbolic (mnemonic) instructions directly into executable machine language instructions. Assembler language therefore permits a programmer to write machine-level instructions. Each assembler symbolic instruction has a one-to-one correspondence to a machine language instruction. For example, a typical assembly language source program for a Wang Model 700 calculator might look like this:

Step #	Mnemonic Instruction
1	1
2	UP
3	WR AL
4	CR/LF
5	END AL

After 'assembly', the object code would be:

Step #	700 Machine Code
1	0701
2	0604
3	0412
4	0108
5	0413

Notice the one-to-one relationship between assembly language steps and machine code. This is what distinguishes an assembler from a compiler.

Since the instruction set of a machine defines the complete set of elementary capabilities provided by the machine, Assembler language provides the programmer with access to the machine's total repertoire of functions.

Again, it is important to note that the machine instruction set of the 2200VS contains all instructions available on the IBM 360, along with most available on the 370.

The 2200VS Assembler also allows a programmer to define a routine consisting of a series of instructions, and assign a name to the routine. The name can then be specified (instead of the entire routine) as a single instruction in a program. Such named routines are called "macros", and the names assigned to them are called "macroinstructions". Because the 2200VS Assembler permits the definition of macros, it is also referred to as a "macroassembler".

Macroinstructions used in preparing and assembler language source program fall into two categories: 'system macroinstructions', provided by Wang, which relate the object program to components of the operating system; and 'programmer-created macroinstructions', specifically for use in the user program at hand, or for incorporation in a library for future use. All current system macros belong to the library named '@ MACLIB @'; the following is a list of system macros:

AIR	FREEMEM	REWRITE
ALEX	GETBUF	RMSG
AXD1	GETMEM	SCRATCH
AXDGEN	GETPARM	SEND
BCE	IORE	SETIME
BCTBL	KEYLIST	START
BCTGEN	LINK	STMB
CALL	LNKB	SVCE
CANCEL	LOW	SVCT
CHECK	MCB	SYSCODE
CLOSE	MSGLIST	TCB

CMSG	OFB	TIME
CREATE	OPEN	TPLAB
DBTB	PATCH	TPLB2
DELETE	PCEXIT	TQEL
DESTROY	PFB	TS
DPT	PFSA	UCB
ETCB	PFT	UFB
EXTRACT	PFTX	UFB2
FDAV	PT	UFBGEN
FDR1	PUTPARM	VCB
FDR2	PXE	VOL1
FDX1	READ	WAIT
FDX2	REGS	WRITE
FLUB	RENAME	XIO
FMSG	RESETIME	XMBUF
FMILIST	RETURN	XMIT
FREEBUF		

'Programmer-created macroinstructions' are used to simplify the writing of a program and to ensure that a standard sequence of instructions is used to accomplish a desired function. For instance, the logic of a program may require the same instruction sequence to be executed again and again. Rather than code this entire sequence each time it is needed, the programmer creates a macroinstruction to represent the sequence and then, each time the sequence is needed, the programmer simply codes the macroinstruction statement. During assembly, the sequence of instructions represented by the macroinstruction is inserted in the object program.

NOTE:
MORE DETAILED INFORMATION ON SYSTEM
MACROINSTRUCTIONS CAN BE FOUND IN THE
2200VS ASSEMBLER LANGUAGE REFERENCE MANUAL
(WL# 800-1200AS)

The assembler language also contains mnemonic assembler-instruction operation codes to specify auxiliary functions performed by the assembler. These are instructions to the assembler program itself, and with a few exceptions, result in the generation of no machine-language object code by the assembler program.

2.3 THE 'OPERATING SYSTEM'

To introduce this subject, a typical single-user system (the 2200VP) will be discussed.

User 'A', at the console of a 2200, runs a job; other users must wait to use the system. User A's job is typical, consisting of Input, Computation, and Output phases.

During the Input phase, the CPU prompts the operator with questions, waits for responses, and stores these responses in memory to await the Computation phase. Assuming that user 'A' is an efficient typist, and reads prompts quickly, the CPU might receive one data item per second. At that rate, the CPU is idle about 99.99 percent of the time. User 'A' has also pre-punched a card deck for input. At 300 cards per minute, the CPU still remains idle approximately 99.54 percent of the time.

The Computation phase is 100 percent efficient, but brief; however, if any disk access is required, the CPU waits an average of 77 percent of the access time.

A 600 LPM printer leaves the CPU idle 98.5 percent of the time. With a 2201, the Output phase leaves the CPU idle 99.98 percent of the time, not counting carriage returns.

Obviously, no customer would buy a single-user machine for a situation where several users are always waiting to access the system. The above example shows that there is adequate CPU time for all jobs (tasks), without having to delay one task for another.

Another inefficiency in a single user system is memory utilization. The entire 64K memory of a 2200VP is available to only one task, whether the task requires 100 bytes or 60 thousand bytes. Furthermore, even in a large task, only one phase is being executed at a time. The immediate memory requirements of any program are usually small.

Peripheral devices in a single-user system are not utilized efficiently. In the example, the card reader was used only during

user A's Input phase; the printer was used only during user A's Output phase; and the disk was used mainly during user A's Computation phase. Each peripheral was actually idle during most of user A's task.

Another weak area in single-user systems is the management of disk and tape storage facilities. Considerable storage space is wasted if each user decides to have a personal disk or tape. Software overhead devoted to file management in a user's program is also a common weakness.

ANY COMPUTER SYSTEM HAS FOUR MAJOR RESOURCES:

1. Processor time
2. Resident memory space
3. I/O devices
4. External data storage space

If a second task could be run using the idle CPU time and wasted storage space of the first task, the second task would run as if it were the only task being executed. In fact, there is so much waste in almost any single user system task, that the optimum number of tasks that could run efficiently, concurrently, could be much higher. This technique is called 'concurrent processing' or 'interleaving'.

As the number of users on a system increases from zero, the system becomes more and more efficient. All system devices decrease their waste time percentages with more users. Even though the operating system overhead is significant, a 2200VS is far better off than a single user system.

However, as users are added, a breaking point can be reached. In a virtual memory system, the breaking point can be defined as having page faults repeatedly occur during paging operations. When a system reaches this condition, the disk drives become extremely active. So much time is tied up in the operating system paging routine and disk I/O, that little or nothing is getting done on any user task. A system in this condition is said to be 'thrashing'. Thrashing can also be defined as a system condition where the users would be better off on a similar single-user machine.

The fundmental cause of thrashing is running too many tasks for the physical memory size. A system on the edge of thrashing can be pushed into that condition by a number of tasks all competing heavily for disk I/O or a bad disk that is consuming time in retries.

DEFINITION:

The Operating System is a software package which interfaces user tasks to the hardware system in such a way as to make optimum use of system resources.

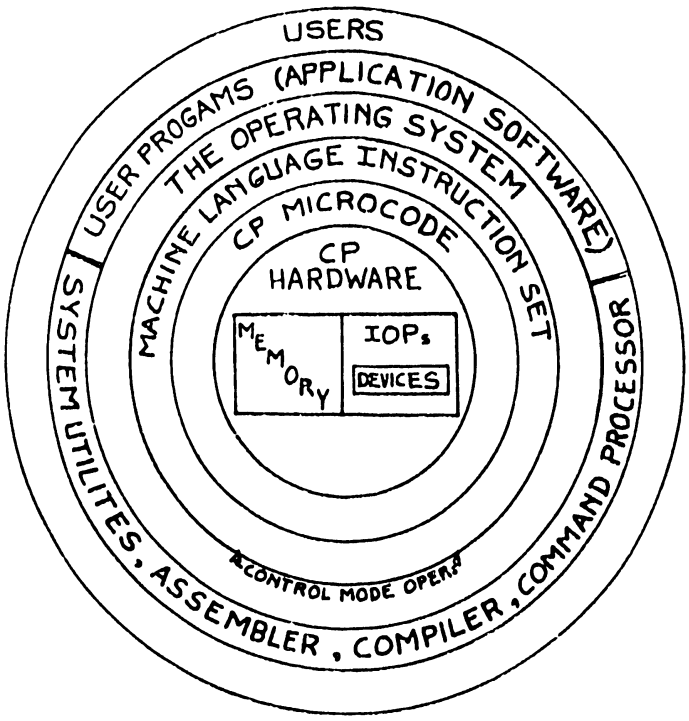
GOALS OF THE OPERATING SYSTEM:

1. Processor Time
   a) Keep the processor computing some user task 100% of the time, switching from task to task to maintain 100% utilization.
   b) Not to delay any task due to unavailability of processor time.

2. Memory Space
   To allocate space for user tasks in such a way as to use all of memory most efficiently.

3. I/O Devices
   a) To keep I/O devices from having any idle time.
   b) To minimize task execution delays caused by busy devices.
   c) To prevent user task conflicts (e.g.: two tasks writing the same file).

4. Data Storage Space
   a) Prevent wasting disk or tape space with unused areas or repetitions of a single character.
   b) Maintain a Volume Table Of Contents (VTOC) for each disk for quick access.



A CONCEPTUALIZATION OF WCS 60/80 OPERATION

FIGURE 2-4

All of these goals are, of course, ideals. In the real world, compromises must be made. The following section examines resource management in more practical terms.

PROCESSOR MANAGEMENT

A user task is submitted to the operating system by the user. The task immediately enters a hold state, waiting for time on the processor. When the task is allowed to begin processing, it is allocated an amount of time called a 'Timeslice'. During this timeslice, the job continues in a RUN state until:

1. The task needs an I/O function or a virtual memory paging operation.
2. A programming error occurs.
3. A higher priority task takes over.
4. The timeslice expires.
5. The task is finished.

If conditions 1, 2, or 3 occur, the task goes into a wait state until the condition is corrected.

Once the condition (1, 2, or 3) is corrected or a task's timeslice has expired, it goes into a READY state and is queued (scheduled) for further processing.

When a task is complete, it is removed from the queueing (scheduling) process and any system resources allocated to it become available for other scheduled tasks.

Only one task is in a run state at any time. All other active tasks are either in the WAIT or the READY state. As soon as any task leaves the RUN state, another READY state task is brought to the RUN state. In this manner, the processor is always doing something useful.

The operating system itself needs processor time to do its functions. However, there are operations it must do that a user task MUST NOT do (such as I/O Control and Memory Paging). To implement such operations, the CPU has two states:

1. Problem State
   User task running; no 'privileged' operations allowed. A 'privileged' operation is any task that is executed only by the Operating System (such as initiating an I/O operation); the user has no access to privileged operations.

2. Supervisor State
   All operations allowed.

When a user task requires a privileged operation, such as I/O, the task must issue a 'supervisor call' and enter the 'wait' state. The Operating System does the privileged operation when expedient and puts the user task back in the READY state.

MEMORY MANAGEMENT:

In a virtual memory system, there are two important considerations in memory management:

1. As many tasks as possible should be in hardware memory ready to process. This is accomplished by loading only those portions of a task which are active at the moment. For example, a program's final print routine would not be loaded until near the end of a task.

2. An algorithm must be used to implement page replacement decisions. The 2200VS uses a Least Recently Used (LRU) algorithm. The operating system occupies certain pages that must not be paged out (e.g., the paging routines). These pages are permanently resident.

I/O DEVICE MANAGEMENT

WORK STATIONS:

The 2200VS is an interactive system, not a batch processor. Because of that, there is no practical way to share a work station among several tasks. Therefore, the idle time of a work station simply has to be tolerated. However, an idle workstation does not slow down the system. Each user work station has 1.25 Meg of virtual memory space allocated to it.

One work station (address 0) is designated as the System Console. Limited control of overall system operation is possible from only this work station. When the Operating System is running, the

'system console' cannot run any user task, nor can its running of the 'System Console Program' be interrupted. The Command Processor cannot be invoked by the system console. The System Console Program consists basically of maintaining a print queue and an I/O error log. The operator of the 'system console' is called the 'system operator'. Other work station operators are usually called users or system users.

The operating system may optionally require that users 'log on' with a prearranged designation and, optionally, a password. This arrangement can be used to control disk file access and/or generate billing for user time.

To be more specific, the Security System consists of:

1) A system user list, with log on I.D., password, special log on procedure, and a three-tiered access classification for all the files on the system.
2) A flexible system of file protection classes, which can be tailored to suit the specific requirements of each installation.

The security system is under the direct control of the security system administrators. These are specially recognized users who are responsible for the meaning and use of file protection classes, and who are able to access all files on the system, including the system user list which identifies all users who have access rights to the system and the special privilege program list.

When a user sits down at a work station, the first display encountered is the LOGON display. LOGON requires the user to enter a USER ID and a PASSWORD; the entered values are then checked against the System-User List, and, if they are located in the list, the user is immediately logged on to the system. Otherwise, LOGON returns a message indicating that an invalid USER ID or PASSWORD has been specified, and the LOGON prompt remains displayed.

NOTE:

To provide an additional measure of security for system users, the PASSWORD is not displayed as it is typed in.

The process of logging a new user on to the system from a particular work station involves automatically resetting all default parameters to the system defaults (in effect wiping out any defaults set by the previous user at that work station), and passing control to the Command Processor. The Command Processor Menu is then displayed on the work station screen.

DISKS

The Operating System relies heavily on disk operations, since disk I/O is faster than any other 2200VS system I/O. The primary considerations for disk I/O operations are 'PRIORITY OF ACCESS' and 'DATA INTEGRITY'.

Virtual Memory Paging is the highest priority disk operation. If this were not so, system performance would be degraded. Also, the operating system will not allow the system disk, containing virtual memory and the Operating System, to be removed from the drive.

The system also makes it impossible to remove or exchange a disk pack that another task is using.

Data integrity is protected by header and Cyclic Redundancy checks. The 75 Meg disks are also written with an ECC (Error Correction Code) that will correct up to 12 bad bits per page. All I/O errors, whether hard of soft, are logged and available for printout via the system console.

PRINTERS

The printer is usually a bottleneck in the system. Direct time sharing of a printer is not possible, due to certain obvious physical

impracticalities. The other extreme is to assign a printer to tasks on a first-come, first-served priority basis. That mode of operation would bring back most of the printer problems associated with a single-user system.

With the 2200VS system, printed output is not obtained directly. The Data Management System intercepts output sent to the printer, temporarily stores it in a disk file for subsequent printing, and informs the System Console Program of its presence. The System Console Program records the names and types of files to be printed in the 'Print Queue', and schedules each job for printout later.

Whenever a print file is created, the Data Management System does two things:

1) It places this file in the 'User's Print File Library'.
2) It passes the name, location, and status (spool or hold) of the print file to the System Console Program. The System Console Program can retrieve any of these files for subsequent printing.

This process is called 'print spooling'. As far as a user program is concerned, if a print task is generated, once that task has been 'spooled', the printing portion of the user task is considered to be already done. Thus, the Operating System is using disk as a 'Virtual Printer'. All user tasks therefore run much faster because of the speed difference between disk and printer.

The printer can be utilized almost continuously, printing files in the order queued. The system operator may adjust the queue sequence manually at the system console to allow for changing job priorities, job length, etc.

The system operator may also release printers to specific tasks for high-priority, on-line use. Anything queued to a released printer will be held until the system reacquires the printer. Releasing and acquiring printers is strictly a software function, requiring no cable changing or switch setting.

DATA STORAGE MANAGEMENT HIERARCHY: FILES, LIBRARIES, AND VOLUMES

The creation and maintenance of files is controlled by the 2200VS Data Management Subsystem. A "file" is a logical unit consisting of one or more records. A file may contain source program text (a "source file") or object program code (a "program file"), or it may contain data records. Files can be opened and named by the user; Data Management automatically handles the complex "housekeeping" chores associated with creating and maintaining an external file. Each file is located within a hierarchical structure consisting of two higher levels: libraries and volumes.

The most comprehensive unit in the file management hierarchy is the volume. A volume is an independent physical storage medium, such as a diskette or a disk pack. The volume name provides a device-independent means of identifying physical storage units. Once a diskette or disk pack has been assigned a volume name, it can be mounted at any available drive unit and accessed by name, without reference to the address or physical characteristics of the disk unit itself.

Immediately below the volume in the hierarchy is the library. A volume may contain one or more user libraries, but a single library may not continue onto a second volume. Each library contains one or more files. Every file must be assigned to a library. Files are not always in one contiguous area on a volume. Sections of each file are put wherever they will fit, in order to gain full utilization of the disk. The 2200VS places no particular restrictions on the types of files placed in a library; a single library may be used for source, program, and data files, or special libraries may be designated for each file type. The conventions governing library usage are completely determined at each individual installation, based on its particular needs and standards.

Duplicate file names cannot be used within the same library, but they may be used in different libraries. Similarly, duplicate library names are not permitted on the same volume, but may be used on separate volumes. Finally, duplicate volume names are allowed but not recommended.

To avoid possible ambiguity, each file name must be qualified with the names of its associated library and volume when the file is opened. Such qualification is not required, however, when running programs from the System Program Library, because the System Library and Volume are used automatically whenever the named program is not located in the user program library, or no user library is supplied. Because all system utilities are stored in the System Program Library, it is never necessary to specify a library or volume name when invoking a system utility program.

SYSTEM PROGRAM LIBRARY

In the special case of program files, an additional level of default is provided by the system. The system program library is used as a default library by the RUN command whenever the named program cannot be found in the specified user program library or whenever no user library is supplied. (Note that the names of the system program library and its associated volume are not displayed in the RUN prompt; the library and volume name fields remain blank.) This feature permits the user to run system utility programs without specifying a library or volume name; without, indeed, ever needing to know those names.

If the user has specified his own default names for Program Library and Volume, he need not change or erase these names when running a system utility. The RUN command first checks the user program library for the specified program. If the program is not found there, RUN then automatically checks the system program library. A system utility program may therefore be run without entering a program library or volume name, whether or not user-specified defaults for these names appear in the RUN prompt.

SECTION 3

INTRODUCTION TO 2200VS HARDWARE

SECTION 3

INTRODUCTION TO 2200VS HARDWARE

3.1 SYSTEM BLOCK OVERVIEW

3.1.1 GENERAL

The WCS-60/80 (2200VS) is a multiple-processor system. At the heart of this system are four major functional elements.

1. Central Processor (CP).
2. Processor Communication Bus (PCB).
3. Main Memory (MM).
4. Main Memory Bus (MMB).

The Central Processor contains facilities for addressing Main (physical) Memory, for fetching or storing information, for arithmetic and logical processing of data, for sequencing instructions in the desired order, and for initiating communication between Main Memory and external devices. In general, the primary task of the Central Processor is to execute the machine instruction set and monitor the results of execution. (There are actually two instruction sets in the Central Processor, the machine instruction set and the micro instruction set. Each machine instruction is actually a microcoded routine, stored in the Central Processor.) Included in the Central Processor are multipurpose registers, Control Memory, and a binary clock. Control Memory provides a storage area for the 2200VS microcode. This memory is also referred to as PROM (Programmable Read-Only Memory), or simply, ROM (Read-Only Memory). The binary clock runs at power line frequency and provides the CP with a date and time of day as a reference for such functions as allocating CP usage.

The Processor Communication Bus controls all operations between the Input/Output processors and the Central Processor. It is the Processor Communication Bus logic which maintains 'hand shake' protocol between IOP's and the CP.

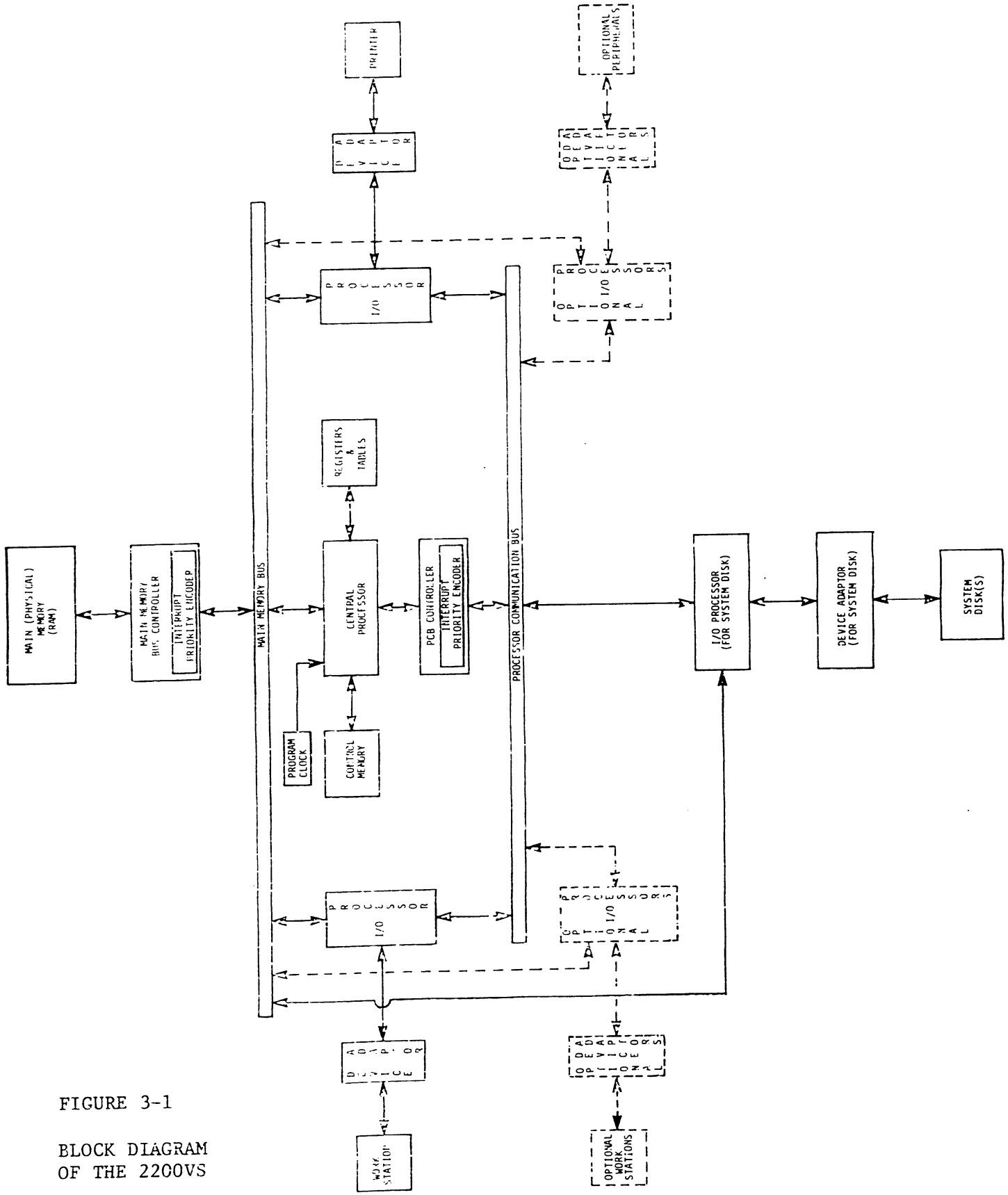


FIGURE 3-1

BLOCK DIAGRAM
OF THE 2200VS

Main Memory is a dynamic random-access memory with automatic error-correction circuitry. All processors in the system have the ability to access Main Memory using the Main Memory Bus. The Main Memory Bus controls all data transfers between IOPs and Main Memory, and between the CP and Main Memory. Direct Memory Access (DMA) is available to each Input/Output Processor and the Central Processor on a priority basis. Processor requests for Main Memory access are sequenced at the start of each Main Memory cycle. The CP is given lowest priority for memory access, due to its ability to use memory cycles that occur between IOP memory accesses. A Main Memory operation consists of a transfer of one or two bytes between Main Memory and a processor. The maximum Main Memory size is 512K (WCS-80). Address translation hardware supports a virtual memory configuration using a 'system disk' which effectively provides each user with one megabyte of memory space.

On the periphery of the Central Processor, one or more subordinate processors (Input/Output Processors-IOPs) receive commands from the CP and control their respective peripherals. Starting and stopping of any IOP is controlled by the CP; however, once initiated, an IOP processes independently of the CP. This allows concurrent I/O processing.

After completion or rejection of any command from the CP, the IOP 'interrupts' the CP to report status. This interrupt capability allows the Central Processor to perform other tasks during I/O 'waits'. For example, a command could be given to one IOP to print data on the printer while, simultaneously, another IOP could be communicating with a work station. Upon completion of their respective tasks, each IOP will request service from the CP on a priority basis, determined by the physical location of the IOP in the computer chassis. Each IOP, while fundamentally the same as other IOPs, is customized for the devices it controls by means of a Device Adapter and a unique microprogram. The Device Adapter is the interface between the IOP microprocessor and the peripheral device.

Each Device Adapter is plugged into its corresponding IOP. There are two basic types of device adapters: one is for work stations and printers; the other is for disks.

The following device adapters are available:

22V01 Printer/Workstation IOP.
Supports one printer and up to three workstations.

22V02 Diskette/10 Megabyte Disk IOP.
Supports one 2270V 315,000-Byte Diskette Drive and up to three 2260V 10-Megabyte Fixed/Removable Disk Drives.

22V04 75/288-Megabyte Removable Disk Drive IOP.
Supports any combination of up to four 2265-1 75-Megabyte Removable Disk Drives and 2265V-2-Megabyte Removable Disk Drives.

22V05 9-Track Tape Drive IOP.
Supports up to four 2209V 9-Track Magnetic Tape Drives.

22V06 Communications IOP.
Available in two models to support bisynchronous telecommunications in the following combinations.
22V06-1 - Supports one bisynchronous line.
22V06-2 - Supports two bisynchronous lines.

3.1.2 DATA ORGANIZATION IN THE 2200VS:

The following data format specifications apply throughout 2200VS system hardware:

8-bit bytes may be handled separately or grouped together in fields. A 'word' is a field of four consecutive bytes and is the basic building block of instructions. A 'doubleword' is a field

consisting of two words and a 'halfword' is a field consisting of two bytes. The location of any field or group of bytes is specified by the address of its leftmost byte, with 'alignment' required for words or doublewords.

Alignment:

Fixed-length fields, such as halfwords and doublewords, must be 'aligned' in main storage on an integral boundary for that unit of information. A boundary is called integral for a unit of information when its storage address is a multiple of the length of the unit in bytes. For example, words (four bytes) must be located in storage so that their address is zero, 4, or a multiple of the number 4. A halfword (two bytes) must have an address that is zero, 2 or a multiple of the number 2, and doublewords (eight bytes) must have an address that is zero, 8, or a multiple of the number 8.

'Boundaries' for halfwords, words, and doublewords can be specified only by the binary addresses in which one, two, or three of the low-order bits, respectively, are zero (Figure 2-2). For example, the integral boundary for a word is a binary address in which the two low-order positions are zero.

Variable-length fields are not limited to integral boundaries, and may start on any byte location.

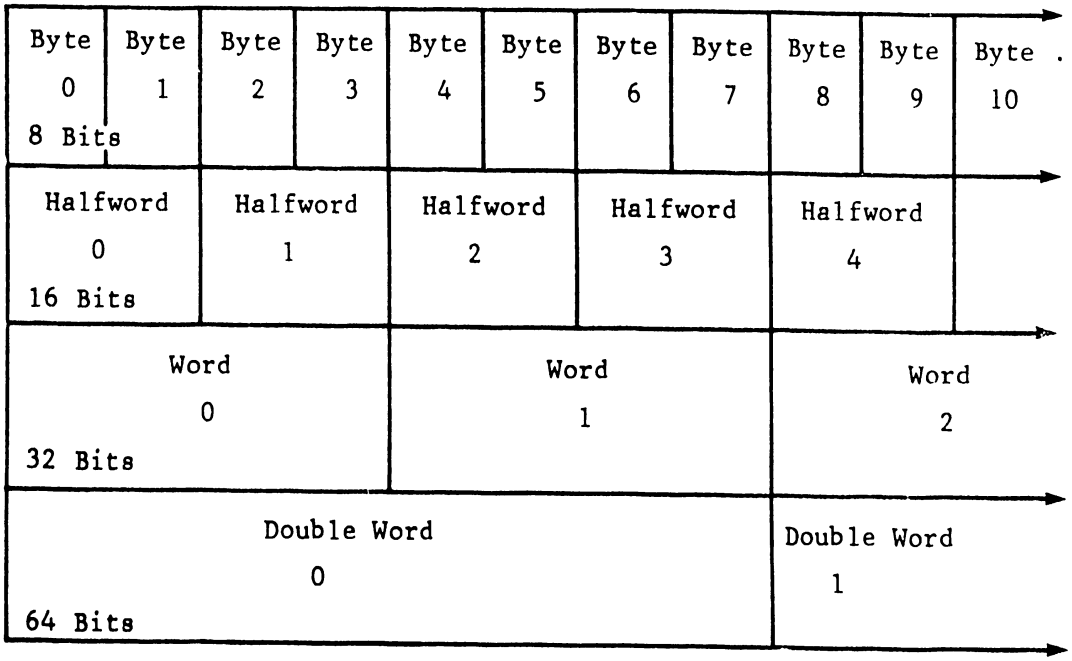


FIGURE 3-2

FIELD ALIGNMENT

Each functional block is discussed in greater detail in Section 3.2.

3.2 THE CENTRAL PROCESSOR

To introduce this section, a cursory presentation of CP hardware is rendered, followed by a more detailed discussion of each CP element in paragraph 3.2.2. Any new vocabulary encountered in 3.2.1 is further explained in 3.2.2.

3.2.1 GENERAL

The Central Processor is the primary controller of the 2200VS. It contains facilities for addressing Main Memory, for fetching and storing information, for arithmetic and logical operations, for sequencing instructions in the desired order, and for initiating communication between Main Memory and external devices. Generally speaking, the primary task of the Central Processor is to execute its instruction set and to monitor and manipulate the results of execution.

The CP has 2 basic modes of operation, each being mutually exclusive of the other:

1. Operation under the 'Operating System'.
2. Control Mode Operation.

Under the Operating System, normal user tasks can be executed by the CP. Under 'Control Mode', normal program execution is halted and certain other facilities are made available. Control Mode interactions are performed exclusively through the System Console, and Control Mode messages are displayed only in the top line of the System Console screen.

Control Mode facilities are divided into two groups, 'debug' and 'load'.

1. Debug - This group contains commands for displaying and/or modifying main memory, general registers, system registers

and the Program Control Word (*PCW). Also included in this group are commands for single step program execution, hard copy dump of memory and registers, and virtual address translation.

2. Load - This group contains commands for initializing the Operating System, loading 'stand-alone' programs, loading diagnostic programs or restarting programs (from an initialized state).

Major elements of the CP are as follows:

MASTER CLOCK:

The function of the master clock is to supply a means of processing microinstructions. The time required to process one microinstruction is referred to as a 'machine cycle'. A further breakdown of a machine cycle occurs in units called 'sub-cycles'.

All instructions are processed in twelve 55 nsec sub-cycle times = 660 nsec., with the exception of the three virtual address manipulation instructions, which use sixteen subcycles (880 nsec) each.

REGISTER STRUCTURE:

a) General Registers

The CP can address information in 16 'general' registers. The general registers are used as index registers in address calculations and as accumulators in fixed-point arithmetic and logical operations. General registers have a capacity of one word (32 bits). These registers are numbered 0-15 and are specified by a four-bit R

---

*PCW - Used to control instruction sequencing, and to hold and indicate the status of the system in relation to the program currently being executed. Further explanation follows.

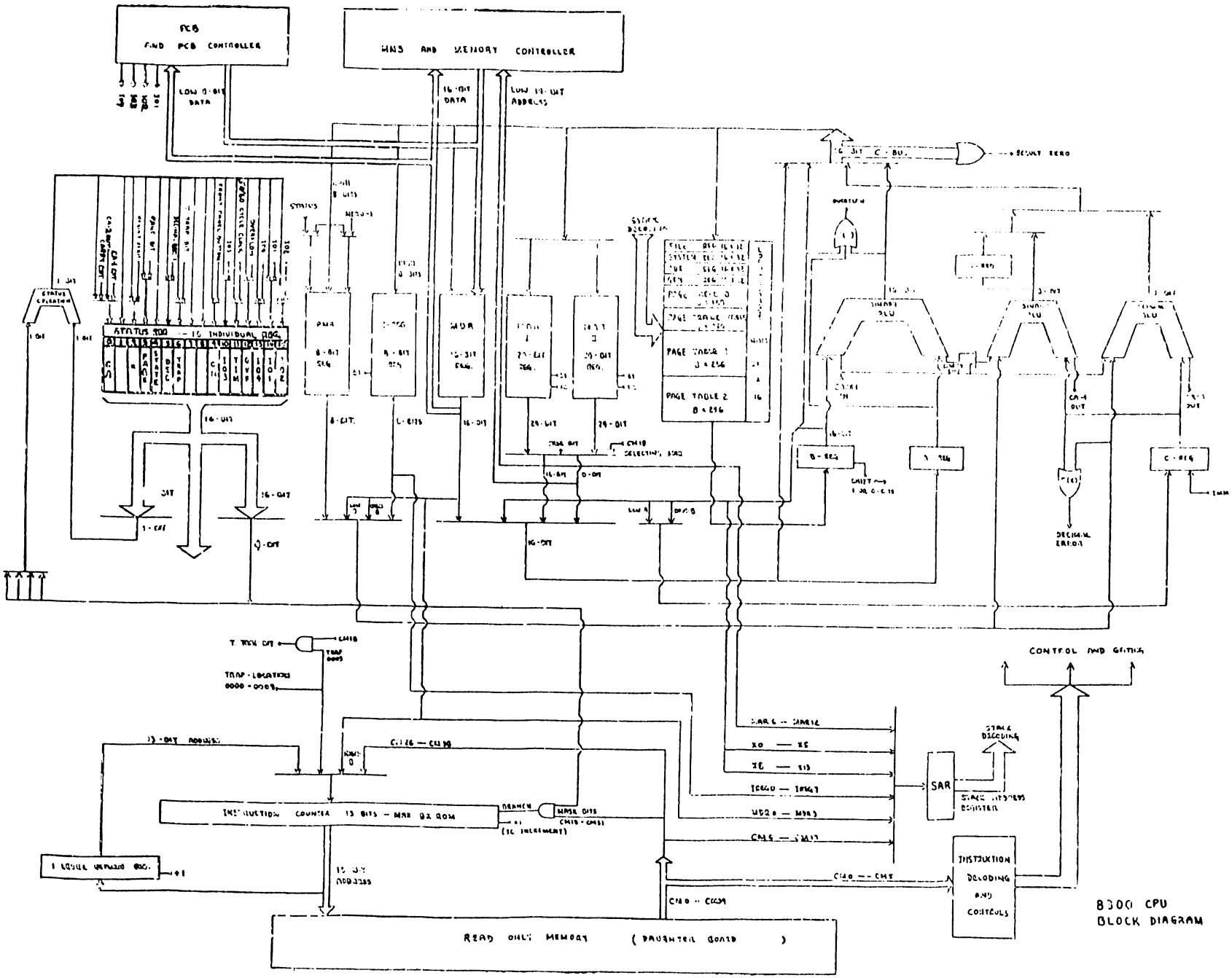


FIGURE 3-3 THE 2200VS CENTRAL PROCESSOR

(Register) field in a microinstruction. Some instructions provide for addressing multiple general registers by having several R fields. The General registers are located in the CP 'stack'. (The CP 'stack' is discussed separately in Section 3.

b) Floating Point Registers

Four floating point registers exist, they are specified as registers 0, 2, 4 and 6. Each FP register is 64 bits in length (one doubleword), and can serve to contain one long floating-point number. These registers are addressed by the floating-point instructions only. The Floating Point Registers are located in the CP 'stack'.

c) System Registers

The system registers provide a means of storing system control information that is used in the execution of the machine instruction set (do not confuse with microinstruction set). The System Registers are located in the CP 'stack'.

d) A-Register, B-Register

These are 16-bit registers, used to hold the 'A' and 'B' operands for the ALU.

c) C-Register

The C-Register holds a B operand for the 8-bit ALUs.

d) Program Mask Register

The PMR is an 8-bit register which holds the Condition Code, system mask bits, and the Instruction Length Code; all of which are discussed in Section 3.2.2.

e) Indirect Register

The IR is an 8-bit register used to store a CP Stack address. The stored value is used for indirect stack addressing operations.

f) Memory Address Register(s)

The CP contains two identical MARs (MAR1 and MAR2). Each MAR can hold either a virtual memory address (used to calculate a physical memory address)or a translated physical memory address. MAR1 and MAR2 are identical in terms of the operations that can be performed on them. The reason for this duplication is related to the time frame during which they are used. As one MAR is being utilized, the other can be loaded with information required for the next operation, thus eliminating CPU 'waits' for MAR availability.

g) Memory Data Register

The MDR is a 16-bit register which is used to transfer data between Main Memory and the CP. Data moving to and from IOPs is also handled by the MDR.

h) Status Register

This Status Register is a 16-bit register which contains a representation of external conditions, 2200VS arithmetic and logical results, and CM microprogram flags (i.e., bits available for microprogram usage).

INTERNAL STACK:

The Internal Stack (also called the 'CP stack' or local storage') is a RAM configured as 512 x 16 bits. A 9-bit address from the Stack Address Register (SAR) is used to select a particular 16-bit stack element. When an 8-bit operation is performed, another address bit selects one byte from the 16-bit stack element. When an address

operand (24 bits) is used, a pair of stack elements is referenced. Many of the stack elements represent a major portion of the CP register structure. Four other important elements of the Internal Stack are the three Local Page Tables and the Local Page Frame Table (see 'Virtual Memory', Section 2).

ARITHMETIC/LOGIC UNITS:

There are three ALUs in the CP:

a) 16-Bit Binary ALU
This is the primary ALU for the CP. It is capable of all arithmetic and logical operations of the 74181 ALU chip.

b) 8-Bit Binary ALU
This is essentially the same as a), but is used for single-byte instructions and as an extension to the 16-bit ALU when manipulating 24 bit virtual addresses.

c) 8-Bit Decimal ALU
This ALU uses BCD, two digits at a time, for execution of decimal microinstructions.

CONTROL MEMORY:

Control Memory (CM) provides a storage area for the CP's micro-program. CM is based on the INTEL 2708 PROM. It is also referred to as 'ROM'. Control Memory is addressed by the Instruction Counter (IC). All CP operations are directed by Control Memory microcode.

BUS STRUCTURES:

a) Processor Communication Bus

The PCB carries commands and status information between CP and IOPs.

b) Main Memory Bus

The MMB is a direct Main Memory access (DMA) channel that can used by every processor in the system.

c) 'C' Bus

The output of the ALU is routed to nearly every register and storage area in the CP via the C Bus.

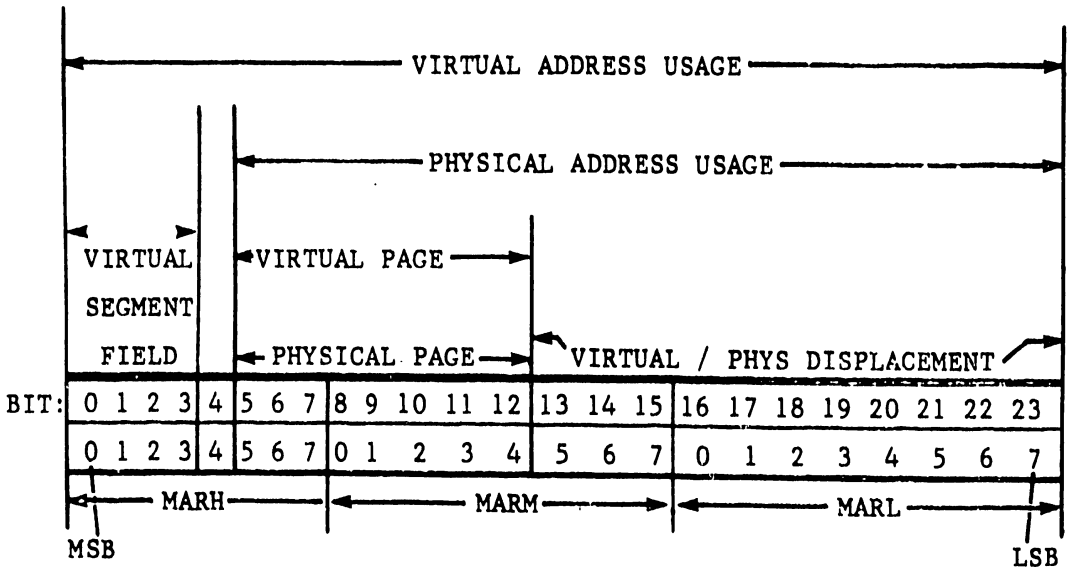
3.2.2 CENTRAL PROCESSOR HARDWARE DETAILS

The CP exists physically on three logic cards in the processor cabinet: CPU #1 card (7301), CPU #2 card (7302), and the PROM board (7107) which is mounted on the CPU #2 card in a piggyback fashion. The MMB and PCB are located on other cards.

REGISTER STRUCTURE:

Memory Address Registers (MAR1, MAR2)

Each MAR is a 24 bit register, logically divided into three parts: MARH, MARM and MARL.



Since the maximum available physical memory is 512K (524,288 bytes), only 19 addressing bits are required for addressing of page frames. The full 24 bits of each MAR can also be used to develop any of 16,777,216 virtual memory addresses, of which, only 1,310,720 are presently valid.

Each MAR is designed so that it may be sequentially incremented (+1 or +2), or decremented (-1). This is accomplished by a 'ripple' operation, specified in the memory field of certain microinstructions.

Ripple '+2':

A ripple +2 operation is used to alter the memory address in halfword increments.

Ripple '+1':

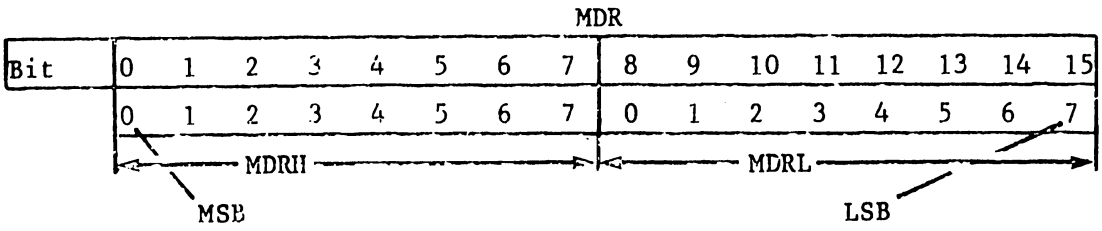
A ripple +1 operation is used to alter memory address in one-byte increments. A ripple +1 operation would be performed under conditions that require access to odd addresses.

Ripple '-1':

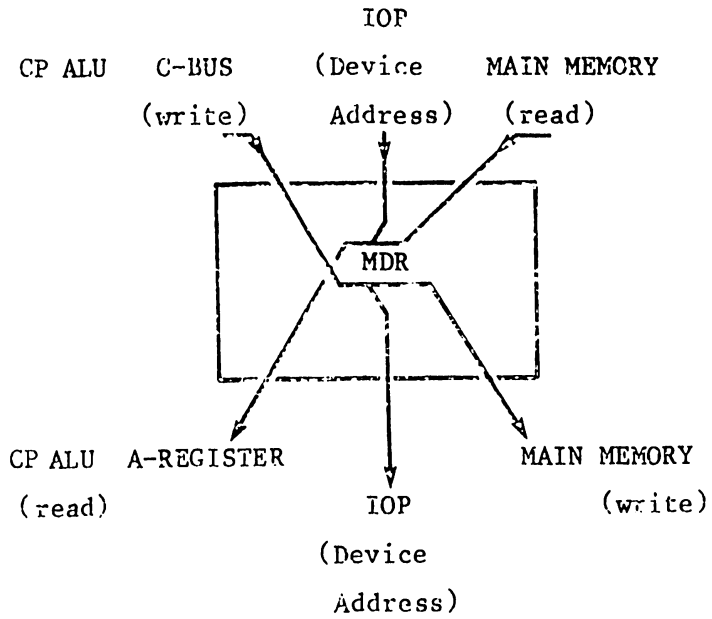
MAR also has the capability of being decremented by one. This operation is required for certain machine instructions such as ADD or SUBTRACT, during which a descending value in MAR is required.

b) Memory Data Register

The MDR is a 16 bit register, logically divided into two parts: MDRH and MDRL.



The primary function of MDR is to transfer data between Main Memory and the CP in the following manner:



It serves as the main processing register in the CP since it participates in the transfer of data between Main Memory, local storage and other CP registers. Data moving to the IOPs is also held by the MDR and sent to the IOP via the PCB.

The output of the MDR is also an input to the ALU multiplexers, thus allowing the MDR to be utilized in 16 bit and 8 bit arithmetic and logic functions.

NOTE:

When communicating with the IOPs, only MDRL0-MDRL7 bits are used; MDRL contains both command and address information. When directed to the CP, these bits contain status and address information necessary for the execution of a machine instruction.

c) Program Mask Register

The PMR is an 8 bit register.

CC		System Mask Bits				ILC	
0	1	2	3	4	5	6	7

The functions performed by its three fields are as follows:

Condition Code (CC) - PMR bits 0 and 1

The code in this field indicates the results of a machine language instruction. The CC is then available to be inspected and used for conditional branching. For instance, the results of a Compare instruction would be placed in the CC field to indicate either a Compare or a Non-Compare condition.

Condition code reflects the result of a majority of the machine arithmetic, logical, and I/O operations. Each of these operations can set the code to any one of four states, and the conditional branch can

specify any combination of these four states as the criterion for branching. For example, the condition code reflects such conditions as non-zero, first operand high, equal, low, overflow, I/O device busy, zero, etc. Once set, the condition code remains unchanged until modified by an instruction that causes a different condition code to be set.

The two bits of the condition code provide for four possible condition code settings: 0, 1, 2, and 3. The specific meaning of any setting depends on the operation that sets the condition code.

Refer to the 2200VS Principles of Operations manual (WL# 800-1100PO) for further explanations of 'condition codes'.

System Mask Bits:

The system mask bits are used to enable/disable the I/O Interrupt, the Clock Interrupt, and the Machine Check Interrupt logic in the CP. (Interrupts are discussed in Section 3.2.4).

Instruction Length Code (ILC) - PMR bits 6 and 7

This field indicates the length of the current machine language instruction.

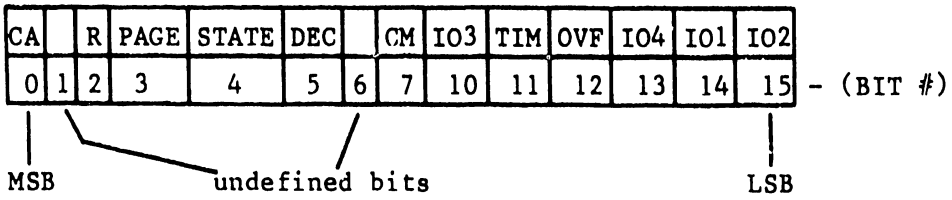
PMR 6-7 = 00	length = 2 bytes
PMR 6-7 = 01 or 10	length = 4 bytes
PMR 6-7 = 11	length = 6 bytes

These bits are used to increment a MAR in order to obtain the next instruction address. The length represented by the ILC is actually added to the address in MAR1 or MAR2 when a machine instruction is fetched.

The entire Program Mask register can also be used by the immediate microinstructions as an 8 bit operand. Also, microinstructions are available to set PMR6&7 from the opcode of a machine instruction in the MDR (MDR bits 0&1).

d) Status Register

The SR is divided into 16 one-bit units:



These 16 one-bit units represent external conditions, CP arithmetic and logical results, and microprogram flags. These bits are available in groups of 4 for testing under a mask with the conditional branch microinstructions. Fifteen of the 16 bits are settable by microinstructions. Note that I/O 3 may not be set by microinstruction. All 16 bits of the SR may be set by hardware.

C.A. - Carry - Status bit 0 (S0)

This bit is used as the "carry-in" and is set as the "carry out" for certain ALU operations.

Status bit 1 (S1) - Not Used

R - Result - Status bit 2 (S2)

This bit is set for all processing operations, all 8 bit and 16 bit moves, 16 bit move indirect, and immediate operations.

S2 = 0 means that: Result = 0
S2 = 1 means that: Result Not = 0

Page - Page bit - Status bit 3 (S3)

This bit is conditioned by the results of a ripple operation performed on either MAR1 or MAR2.

When a ripple operation is called for, the page bit is set to 1. After the ripple has been completed, this bit is checked by the micro-program. When the page bit is set to zero, the updated page address is in a page that is different from the page initially addressed by MAR. If the page bit is set to one, the new address is in the same page initially addressed by MAR.

When page bit = 0, for a:

Positive ripple - The present MAR value points to the first byte of the next page.

Negative ripple - The MAR points to the last byte of the preceding page.

State - Status bit 4 (S4)

This bit represents segment protection information and is used in the WTRAN (Write/Translate) microinstruction.

S4=1 Problem or User State (Segments 0 and 1 are protected)
S4=0 Supervisor or System State (No protect on)

Dec - Decimal Error bit - Status bit 5 (S5)

Set to 1 if an non-decimal digit A_{16}-F_{16} was used as an operand in a decimal add or subtract instruction.

S6 - Translation Trap bit - Status bit 6 (S6)

This bit is set to 0 or 1 when a trap for an invalid virtual address or a page fault is taken (trap 0003). A zero indicates that the contents of MAR 1 caused the trap; a one indicates MAR 2.

CM - Control Mode bit - Status bit 9 (S9)

This bit is set to 1 when the Control Mode button on the front of the processor cabinet is pressed. S9 remains set until the Control Mode button is released.

IO3 - I/O Interrupt bit - Status bit 10 (S10)

This bit is held by the hardware at 0 or 1; it is not settable by the microprogram. (It is held at 1 if one or more IOPs have their PCB request-in lines high; otherwise its held at 0.)

NOTE:

The 'Request-In' lines are used to signal the CP that an IOP wishes to be serviced. These requests are handled on a priority basis by the PCB logic.

TIM - Timer bit - Status bit 11 (S11)

This bit, also called the 'real-time clock tick', is set by the hardware from the AC line frequency every 1/50 or 1/60 of a second, in order to increment the Real Time Clock (RTC).

OVF - Overflow bit - Status bit 12 (S12)

For arithmetic binary operations, when there is a carry-out from the most significant digit position into the + sign position, an 'overflow' condition exists. S12 is set under such conditions.

IO4 - I/O Status bit - Status bit 13 (S13)

S13 is set by the PCB 'Control-In Strobe' line whenever that line is raised by an IOP. The Control-In strobe is sent by an IOP to strobe I/O 1&2 plus the device address into MDRL. The CP microprogram must reset this bit to zero at the end of any PCB operation.

IO1 & IO2 - Status bits 14 (S14) and 15 (S15)

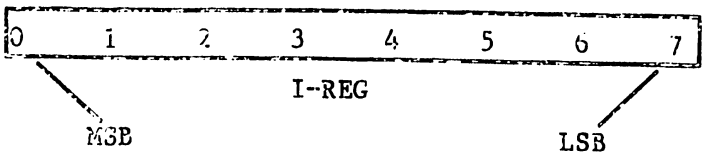
These bits are set by the PCB control lines whenever a 'Control-In Strobe' is issued by an IOP.

IO1	IO2	CONDITION:
0	0	IOP AND DEVICE READY
0	1	DEVICE BUSY
1	0	IOP BUSY
1	1	NOT OPERABLE

e) Indirect Register (I Reg)

The I-Reg is an 8-bit register used to address stack (local storage) indirectly.

Indirect register structure:



There are 3 formats for this register:

1) General Register - Indirect

In this format, a 16-bit register within the 32 x 16-bit General Register area of the stack may be addressed.

2) System Register - Indirect

In this format, a 16-bit register within the 32 x 16-bit System Register area of the stack may be addressed. Four indirect bits (IREG0-IREG3 or IREG4-IREG7) are used along with a low-order address bit supplied by the microinstruction.

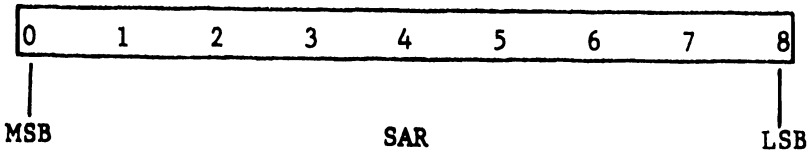
3) Stack - Indirect

Any of the 512 x 16 bit elements within the stack can be addressed by using a high order address bit from the microinstruction along with IREG0-IREG7. This indirect ability is incorporated in several microinstructions which allow the movement of the selected stack element (16 bits) to and from the MDR. These microinstructions may also cause the contents of the IREG to be altered (±1).

INTERNAL STACK:

512 16-bit RAM 'halfwords' comprise the 'local storage' area of the CP (i.e., the 'stack'). Contents of the stack are available only for CP use. The stack is addressed by 9 address lines from the Stack Address Register (SAR). Stack addresses may be loaded into the SAR from the following sources:

MAR6-MAR12 (Memory Address Register)
X0-X5 (B Register output)
X8-X13 (B Register output)
IREG0-IREG7 (Indirect Register)
MDR0-MDR3 (Memory Data Register)
CM6-CM17 (Control Memory bits)



The high order 4 bits SAR0-SAR3 are used to select the group of registers desired:

SAR0-SAR3 = 0000

Selects the file registers. These registers are temporary storage areas for microprogram usage. 32 File registers are made available for CP use.

SAR0-SAR3 = 0001

Selects the system registers. Some elements contained in the system registers are:

Program Control Word Trap Address:

The Program Control Word Trap Address contains information required for proper machine instruction execution (Ref: Section 3.XX)

Virtual Destination Trap Address:

Required by the Operating System for proper execution.

Time of Day Clock Word:

Contains a 32 bit binary quantity, which, translated, represents the time of day for such purposes as billing, logging on/off, etc. Two 16-bit Stack locations are allocated for this purpose.

Clock Comparator:

A programmable, presettable 32-bit register. When time of day equals the value programmed into that register, an interrupt is generated.

Floating Point Registers:

Four floating point registers exist within the System Register section of stack. Each FP register is 64 bits in length and contain one floating-point number. These registers are addressed by floating point instructions only. In order to address the full 64 bits of an FP register, four separate stack addresses are required.

SAR0-SAR3 = 0010

Accesses the Auxiliary registers. These 32 registers are also temporary storage areas for use by the microprogram.

SAR0-SAR3 = 0011

Accesses the General registers. There are 16 general registers that, for machine programs, can be used as index registers in arithmetic and logical instructions, and as accumulators in fixed point arithmetic and logical operations. The registers are 32 bits long, thus requiring 2 stack locations. The general registers are identified by the numbers 0-15 and are specified by a four bit R field in a machine instruction.

For the entire preceeding group of registers in local storage (stack), SAR_0 through SAR_3 generally select a register group; the low order bits (SAR_4 through SAR_8) select the specific register within a group.

Local Page Tables:

The two high order bits of the SAR are used for selecting one of three Page tables:

SAR0-SAR3 = 01XX Local Page Table 0 selected
SAR0-SAR3 = 10XX Local Page Table 1 selected
SAR0-SAR3 = 11XX Local Page Table 2 selected

The lower order bits (SAR3-SAR8) are used to select a 16 bit element within the local page table. Each 16 bit local page table element actually holds two 8-bit entries. The number of local page table entries corresponds to the number of pages in a segment.

Page Table 0 - Contains 128 entries for segment 0.
Page Table 1 - Contains 256 entries for segment 1.
Page Table 2 - Contains 256 entries for segment 2.

Each page table entry is used as a pointer to a physical page frame during translation (see Section 2, 'Virtual Memory').

Local Page Frame Table:

In order to access the Page Frame table, the 3 high order bits of the SAR are required (SAR0-SAR2). See Section 2 for an explanation of the local page frame table.

SAR0-SAR3 = 011X causes the Page Frame table to be referenced.

TABLE 3-1

STACK ALLOCATION

ELEMENTS:	ELEMENT SIZE:	ADDRESSES:	VALUE OF SAR 0- SAR 8
File Registers	(32) x (16)	000 01F	(0) (8) 0000XXXXX
System Registers	(32) x (16)	020 03F	0001XXXXX
Auxiliary Registers	(32) x (16)	040 05F	0010XXXXX
General Registers	(32) x (16)	060 07F	0011XXXXX
Page Table 0	(64) x (16)	080 0BF	01XXXXXXX
Page Frame Table	(64) x (16)	0C0 0FF	011XXXXXX
Page Table 1	(128) x (16)	100 17F	10XXXXXXX
Page Table 2	(128) x (16)	180 1FF	11XXXXXXX

X = Element select field.

ARITHMETIC/LOGIC UNIT:

The CP contains 3 ALUs:

1) 16 bit binary ALU.
2) 8 bit binary ALU.
3) 8 bit decimal ALU.

The carry-out bit from the 16 bit binary ALU can be input to the 8 bit binary ALU, thus combining the two and forming a 24 bit ALU.

The 3 ALUs provide the capability to the CP to process binary integers of fixed length, decimal integers of variable length, and logical information of either fixed or variable length.

The basic ALU path utilizes the 16 bit ALU. A typical operation would entail the referencing of a 16 bit stack element. The output of the stack would be temporarily stored in the B register, a sixteen bit operand register. The output of the B register would be looped back around into the A register multiplexor and gated into the A register.

NOTE:
The 'A' register is a 16 bit operand register.

The stack is again referenced and another 16 bit element is latched into the B register. The ALU now performs its selected arithmetic or logical function and puts the result onto the C bus. The C bus is the main traffic highway of the CP since it has access to all CP registers and all CP registers have access to it. Once on the C bus, a check for an all zero condition on the bus is executed and appropriate status can be reported to the status register.

The 8 bit binary ALU functions similarly except that its operands are:

a) The output of the C register. The C register (no relation to the 'C-Bus') is an 8 bit ALU operand register that may be accessed by immediate microinstructions or the E register.
b) The output from a multiplexor network which may select the Status Register, PMR, I Register, and MDR.

The output of the 8 bit ALU may be latched into the T register which is used for timing considerations before it is placed on the C bus.

The 8 bit decimal ALU is utilized with the decimal microinstructions and has the same operand capabilities as the 8 bit binary ALU.

CONTROL MEMORY (ROM):

The PROM chip used for CP Control Memory is the Intel 2708. PROM can be written into by special equipment, thereby saving the device from obsolescence due to microprogram changes. Each PROM chip has a 1K by 8 bit storage array. Five PROM chips are linked together to obtain a 1K by 40 bit storage array. The CP utilizes three more levels of this 5 chip configuration, thus producing an 4K by 40 bit PROM storage area. Another 4K is available for future expansion.

Addressing of (P)ROM is accomplished by the Instruction Counter (IC), thus providing a means of fetching microinstructions from the ROM in sequential order. After each new microinstruction has been fetched, the Instruction Counter is incremented by +1. The Instruction Counter addresses ROM via 13 address lines.

The sequential incrementing of the IC and fetching of microinstructions can be changed by various conditions. Certain microinstructions have the capability of loading the Memory Data Register (MDR) directly into the IC through the IC multiplexor. Also, while executing branch instructions, a destination address can be loaded into the IC by use of the CM bits. The trap operations may also interrupt sequential incrementing of the IC.

Trap operations are caused by external conditions not initiated within the CP (such as a system power-on). In general, trap operations interrupt the microprogram by forcing an address into the IC.

The following is a list of trap addresses in Control Memory and the reasons for trapping:

0000 - Reserved.

0001 - Power on trap (circuit breaker placed on).

0002 - Load button on front of processor cabinet is pressed.

0003 - Invalid virtual memory address during translation..

0004 - Translation trap due to protection of privileged data.

0005 - Memory trap (bad address).

0006 - Memory trap (bad parity).

0007 - Trap microinstruction for word alignments.

0008 - Location for Branch on Instruction trap.

0009 - Trap microinstruction for CC/Mask.

Instruction Decoding and Controls

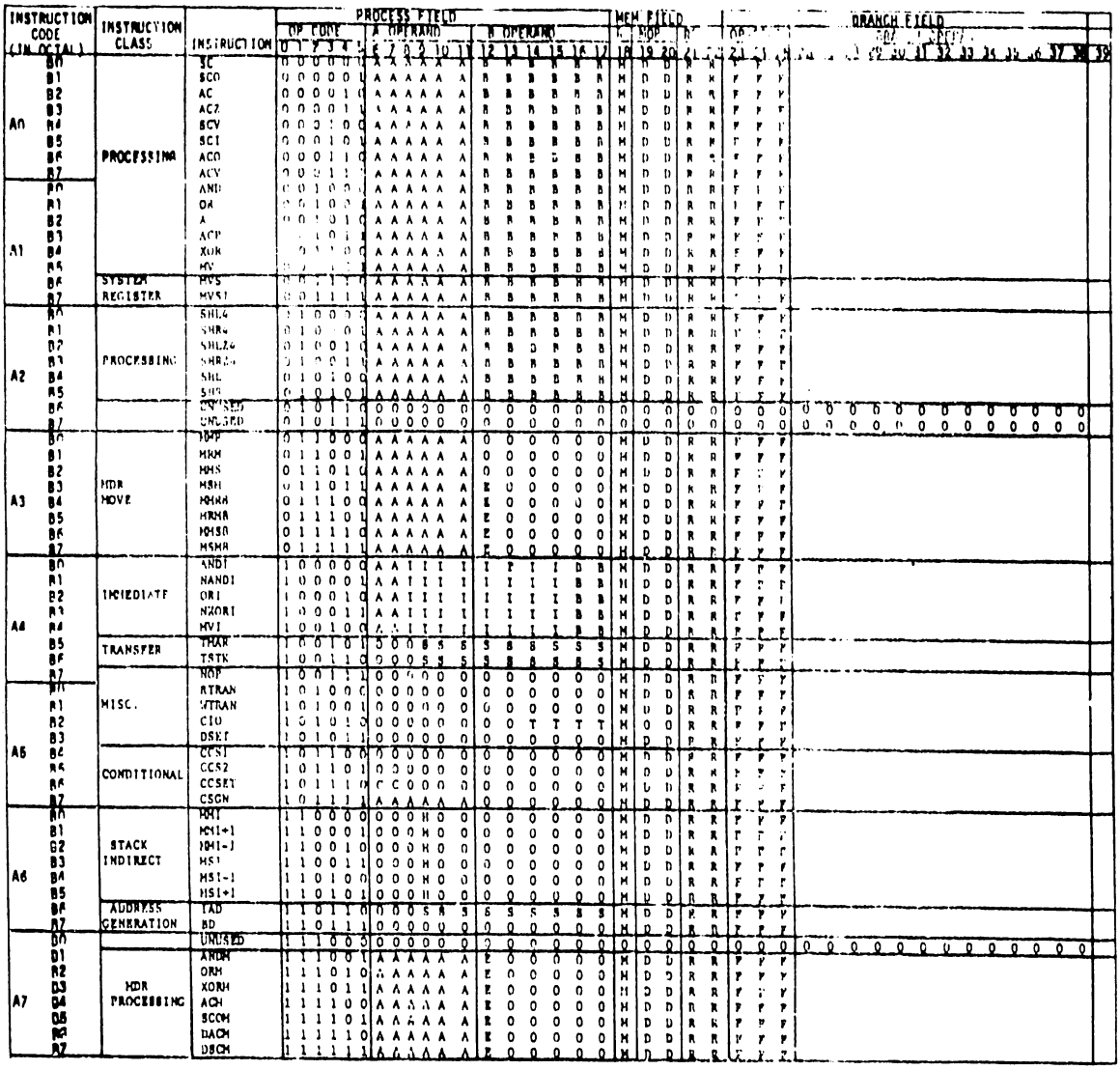
Notice in Figure 3-2 that CM0-CM5 are used by the Instruction decoding and controls. The instruction decoder provides a means of decoding a hardware function from the 6 bit operation code of the microinstruction read from ROM. Each of the 61 operations decoded (three are unused) produce a unique hardware function, such as setting up ALU functions, selection of source/destination registers, and directing the overall flow of data.

3.2.3 INSTRUCTION SETS

The 2200VS CP actually utilizes two sets of instructions: the machine instruction set, and the microinstruction set. The machine instruction set consists of 165 instructions that are nearly identical to the IBM 360/370 instruction set. The CP does not actually execute these instructions. Instead, the CP decodes the operation codes of each machine instruction into an address. This address points to a microcoded subroutine residing in Control Memory. These microinstructions are the actual instructions that the CP will execute.

3.2.3.1 MICROINSTRUCTION FORMAT IN CONTROL MEMORY:

There are 61 unique 40-bit microinstructions used in the CP's Control Memory; they are formatted as follows:



A = A BUS REGISTER
B = B BUS REGISTER
C = CONDITIONAL CODE
D = MEMORY READ/WRITE SPECIFICATION
E = HIGH/LOW STACK BYTE SELECT
F = BRANCH OP-CODE SPECIFICATION
H = HIGH ORDER STACK ADDRESS BIT (A0)
I = IMMEDIATE OPERAND
M = MEMORY ADDRESS REGISTER SELECT
R = MAR FIPPLE SPECIFICATION
S = STACK ADDRESS
T = CIO STROBE SPECIFICATION

TABLE 3-2

THE CP MICRO-INSTRUCTION SET

Bit positions in the microinstructions are referred to as CM0-CM39, as they are read from ROM; CM0 being the most significant bit (MSB) and CM39 being the least significant bit (LSB).

The Process field (CM0-CM17) of the microinstruction contains one of the 61 microinstruction opcodes, the A operand, and the B operand. The A and B operands are used to address various registers.

The Memory field (CM18-CM22) contains 3 elements:

1. R (CM18) is used to select one of the two memory address registers (MAR) that are available in the CP.

0=MAR 1

1=MAR 2

2. MOP (CM19-CM22) contains the memory operation bits. These specify what type of memory operation is to be performed, if any. Memory operations available are:

00 = no memory operation.
01 = read 2 bytes into the memory data register (MDR).
10 = write 2 bytes from MDR into main memory.
11 = write 1 byte from MDR into main memory.

3. ROP (CM21-CM22) are the ripple operation bits which will determine how the Memory Address Register (MAR) is to be incremented or decremented after the current value of the MAR is used for any memory opertion. The various combinations of the ripple bits are:

00 = increase MAR by 1.
01 = increase MAR by 2.
10 = decrement MAR by 1.
11 = no operation.

The Branch field (CM23-CM38) contains one of the following 8 branch opcodes:

000 - B.U. - Branch unconditionally
001 - S.B. - Subroutine branch
010 - B.I. - Branch on Machine Instruction
011 - BT/BF - Branch conditional (True/False)
100 - SR - Subroutine Return
101 - TRP1 - Trap 1 (condition code/mask; called 'TRAP 9' in hardware)
110 - TRP2 - Trap 2 (alignment; called 'TRAP 7' in hardware)
111 - SS - Set Status bit

TABLE 3-3

C.P. HARDWARE INSTRUCTION (MICROINSTRUCTION) SET

MNEMONIC	INSTRUCTION
SC	Binary subtract with carry in
SCO	Binary subtract with carry in = 1
AC	Binary add with carry in
ACZ	Binary add with carry in = 0
SCV	SC with overflow bit set
SCI	Binary subtract (inverted) with carry in
ACO	Binary add with carry in = 1
ACV	AC with overflow bit set
AND	Logical 'AND' (A B)
OR	Logical 'OR' (A+B)
A	Binary add with carry in = 0 and no carry out
ACP	Binary page add with carry in = 0
XOR	Logical exclusive 'OR' (A + B)
MV	Move A-bus to B-bus
SHL4	Shift A-bus 4 bits left
SHR4	Shift A-bus 4 bits right
SHLZ4	SHL4 with 4 bits in = 0
SHRZ4	SHR4 with 4 bits in = 0
SHL	Shift A-bus 1 bit left
SHR	Shift A-bus 1 bit right
MVS	Move system register
MVSI	Move system register indirect
MMR	Move MDR to register
MRM	Move register to MDR
MMS	Move MDR to register
MSM	Move register to MDR
MMR8	Move MDRH to register
MRM8	Move register to MDRH
MMS8	Move MDRH to register
MSM8	Move register to MDRH

ANDI	Logical 'AND' immediate
NANDI	Logical 'AND' immediate. No result stored
ORI	Logical 'OR' immediate
NXORI	Logical exclusive 'OR' immediate. No result stored
MVI	Move immediate
TMAR	Transfer MAR to stack
TSTK	Transfer stack to MAR
NOP	No process operation
RTRAN	Read translate
WTRAN	Write translate
CIO	Control I/O
DSET	'Decode' setup
CCS1	Condition code set by status bits
CCS2	Condition code set by status bits explicitly
CCSET	Condition code set explicitly
CSGN	Condition code set by sign
MMI	Move MDR indirect. IR unchanged
MMI+1	Move MDR indirect. IR incremented 1
MMI-1	Move MDR indirect. IR decremented 1
MSI	Move stack indirect. IR unchanged
MSI-1	Move stack indirect. IR decremented 1
MSI+1	Move stack indirect. IR incremented 1
IAD	Instruction address update
BD	Base displacement address generation
ANDM	Logical 'AND' using MDR
ORM	Logical 'OR' using MDR
XORM	Logical exclusive 'OR' using MDR
ACM	Binary add with carry in using MDR
SCOM	Binary subtract with carry in = 1 using MDR
DACM	Decimal add using MDR
DSCM	Decimal subtract using MDR

3.2.3.2 MACHINE INSTRUCTION FORMAT:

Somewhat in the same manner as microinstruction format, each machine instruction consists of two major parts: (1) an operation code, which specifies the operation to be performed, and (2) the operands that will participate in the instruction.

OPERANDS: - Operands can be grouped in three classes: operands located in registers, immediate operands, and operands in main memory.

Register operands can be located in general, floating point or control registers, and are specified by identifying the register in a four-bit field, called the R field, in the instruction.

Immediate operands are contained within the instruction, and the eight-bit field containing the immediate operand is called the I field.

Main Memory operands may either have an implied length, be specified by a bit mask, or, in other cases, be specified by a four-bit or eight-bit length specification, called the L field, in the instruction. The addresses of operands in main memory are specified by means of a format that uses the contents of a general register as part of the main memory address.

For purposes of describing the execution of instructions, operands are designated as first, second and third operands. In general, only two operands participate in an instruction execution, and the result replaces the first operand. An exception is an instruction with "store" in the instruction name, where the result replaces the second operand.

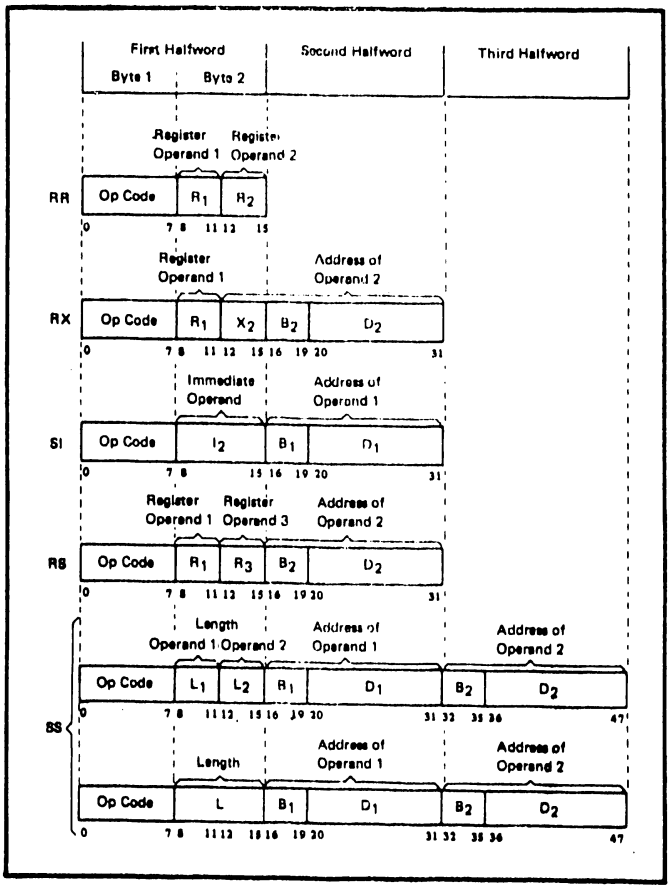


FIGURE 3-4

MACHINE INSTRUCTION FORMAT

TABLE 3-4

MACHINE INSTRUCTION SET

NAME	MNEMONIC	TYPE	CODE	OPERANDS (Assembler Format)
Add	AR	RR	1A	R1,R2
Add	A	RX	5A	R1,D2(X2,B2)
Add Decimal	AP	SS	FA	D1(L1,B1),D2(L2,B2)
Add Halfword	AH	RX	4A	R1,D2(X2,B2)
Add Logical	ALR	RR	1E	R1,R2
Add Logical	AL	RX	5E	R1,D2(X2,B2)
Add Normalized	ADR	RR	2A	R1,R2
Add Normalized	AD	RX	6A	R1,D2(X2,B2)
Add Unnormalized	AW	RX	6E	R1,D2(X2,B2)
And	NR	RR	14	R1,R2
And	N	RX	54	R1,D2(X2,B2)
And	NI	SI	94	D1(B1),I2
And	NC	SS	D4	D1(L,B1),D2(B2)
Bit Reset	BRESET	SI	9D	D1(B1),M1
Bit Set	BSET	SI	9C	D1(B1),M1
Bit Test	BTEST	SI	9E	D1(B1),M1
Branch and Link	BALR	RR	05	R1,R2
Branch and Link	BAL	RX	45	R1,D2(X2,B2)
Branch and Link on Condition Indirect	BALCI	RS	99	M1,R3,D2(B2)
Branch and Link Stack	BALS	RX	81	S1,D2(X2,B2)
Branch On Condition	BCR	RR	07	M1,R2
Branch On Condition	BC	RX	47	M1,D2(X2,B2)
Branch On Condition Stack	BCS	RR	01	M1,S2
Branch On Count	BCTR	RR	06	R1,R2
Branch On Count	BCT	RX	46	R1,D2(X2,B2)
Branch On Index High	BXH	RS	86	R1,R3,D2(B2)
Branch On Index Low or Equal	BXLE	RS	87	R1,R3,D2(B2)
Compare	CR	RR	19	R1,R2
Compare	C	RX	59	R1,D2(X2,B2)
Compare (Floating Point)	CDR	RR	29	R1,R2
Compare (Floating Point)	CD	RX	69	R1,D2(X2,B2)
Compare Decimal	CP	SS	F9	D1(L1,B1),D2(L2,B2)
Compare Halfword	CH	RX	49	R1,D2(X2,B2)
Compare Logical	CLR	RR	15	R1,R2
Compare Logical	CL	RX	55	R1,D2(X2,B2)
Compare Logical	CLI	SI	95	D1(B1),I2
Compare Logical	CLC	SS	D5	D1(L,B1),D2(B2)
Compare Logical Characters Under Mask	CLM	RS	BD	R1,M3,D2(B2)
Compare Logical With Pad	CLPC		E5	D1(L1,B1),D2(L2,B2),L3
Compress String	COMP	SS	F6	D1(R1,B1),D2(R2,B2)
Control I/O	CIO	RR	0C	R1
Convert To Binary	CVB	RX	4F	R1,D2(X2,B2)

NAME	MNEMONIC	TYPE	CODE	OPERANDS (Assembler Format)
Convert To Decimal	CVD	RX	4E	R1,D2(X2,B2)
Convert Floating Point To Integer	CDI		2F	R1,R2
Convert Integer To Floating Point	CID		2E	R1,R2
Decrement and Inspect Semaphore (P)	DSEM	RX	51	R1,D2(X3,B3)
Dequeue	DEQ	RS	A0	R1,D2(B3)
Destack	DESK	RS	A1	R1,D2(B3)
Divide	DR	RR	1D	R1,R2
Divide	D	RX	5D	R1,D2(X2,B2)
Divide (Floating Point)	DDR	RR	2D	R1,R2
Divide (Floating Point)	DD	RX	6D	R1,D2(X2,B2)
Divide Decimal	DP	SS	FD	D1(L1,B1),D2(L2,B2)
Edit	ED	SS	DE	D1(L,B1),D2(B2)
Edit and Mark	EDMK	SS	DF	D1(L,B1),D2(B2)
Enqueue	ENQ	RX	52	R1,D2(X3,B3)
Enstack	ENSK	RX	53	R1,D2(X3,B3)
Exclusive Or	XR	RR	17	R1,R2
Exclusive Or	X	RX	57	R1,D2(X2,B2)
Exclusive Or	XI	SI	97	D1(B1),I2
Exclusive Or	XC	SS	D7	D1(L,B1),D2(B2)
Execute	EX	RX	44	R1,D2(X2,B2)
Expand String	XPAND	SS	F7	D1(R1,B1),D2(R2,B2)
Halt I/O	HIO	RR	03	R1
Halve	HDR	RR	24	R1,R2
Increment and Inspect Semaphore (V)	ISEM	RS	A2	R1,D2(B3)
Insert Character	IC	RX	43	R1,D2(X2,B2)
Insert Characters Under Mask	ICM	RS	BF	R1,M3,D2(B2)
Jump To Subroutine On	JSCI	RX	61	M1,D2(X2,B2)
Load	LR	RR	18	R1,R2
Load	L	RX	58	R1,D2(X2,B2)
Load (Floating Point)	LDR	RR	28	R1,R2
Load (Floating Point)	LD	RX	68	R1,D2(X2,B2)
Load Address	LA	RX	41	R1,D2(X2,B2)
Load and Test	LTR	RR	12	R1,R2
Load and Test	LT	RX	4D	R1,D2(X2,B2)
Load and Test (Floating Point)	LTDR	RR	22	R1,R2
Load Character	LC	RX	62	R1,D2(X2,B2)
Load Complement	LCR	RR	13	R1,R2
Load Complement (Floating Point)	LCDR	RR	23	R1,R2
Load Control	LCTL	RS	B7	R1,R3,D2(B2)
Load Halfword	LH	RX	48	R1,D2(X2,B2)

NAME	MNEMONIC	TYPE	CODE	OPERANDS (Assembler Format)
Load Multiple	LM	RS	98	R1,R3,D2(B2)
Load Negative	LNR	RR	11	R1,R2
Load Negative (Floating Point)	LNDR	RR	21	R1,R2
Load Page Table	LPT0	RS	A3	R1,D2(B2)
Load Page Table	LPT1	RS	A4	R1,D2(B2)
Load Page Table	LPT2	RS	A5	R1,D2(B2)
Load PCW	LPCW	S	82	D1(B1)
Load Physical Address	LPA	RX	B1	R1,D2(X2,B2)
Load Positive	LPR	RR	10	R1,R2
Load Positive (Floating Point	LPDR	RR	20	R1,R2
Move	MVI	SI	92	D1(B1),I2
Move	MVC	SS	D2	D1(L,B1),D2(B2)
Move Numerics	MVN	SS	D1	D1(L,B1),D2(B2)
Move With Offset	MVO	SS	F1	D1(L1,B1),D2(L2,B2)
Move With Pad	MVPC		E2	D1(L1,B1),D2(L2,B2),I3
Move Zones	MVZ	SS	D3	D1(L,B1),D2(B2)
Multiply	MR	RR	1C	R1,R2
Multiply	M	RX	5C	R1,D2(X2,B2)
Multiply (Floating Point)	MDR	RR	2C	R1,R2
Multiply (Floating Point)	MD	RX	6C	R1,D2(X2,B2)
Multiply Decimal	MP	SS	FC	D1(L1,B1),D2(L2,B2)
Multiply Halfword	MH	RX	4C	R1,D2(X2,B2)
Or	OR	RR	16	R1,R2
Or	O	RX	56	R1,D2(X2,B2)
Or	OI	SI	96	D1(B1),I2
Or	OC	SS	D6	D1(L,B1),D2(B2)
Pack	PACK	SS	F2	D1(L1,B1),D2(L2,B2)
Pack and Align	PAL	SS	C4	D1(L1,B1),D2(L2,B2)
Pop	POP	RR	08	S1,R2
Pop Characters	POPC	SS	D8	D1(L,B1),D2(S2)
Pop Halfword	POPH	RR	09	S1,R2
Pop Multiple	POPM	RS	A6	S1,R3,R2
Pop Nothing	POPN	RX	84	S1,D2(X2,B2)
Push	PUSH	RR	0B	S1,R2
Push Address	PUSHA		B0	S1,D2(X2,B2)
Push Characters	PUSHC	SS	D9	D1(L,S1),D2(B2)
Push Multiple	PUSHM	RS	A9	S1,R3,R2
Push Nothing	PUSHN	RX	85	S1,D2(X2,B2)
Reset Reference and Change Bits	RRCB	SI	9F	D1(B1),M2
Return On Condition	RTC	RR	04	M1
Save Then 'AND' System Mask	STNSM		AC	R1,R3,I1
Save Then 'OR' System Mask	STOSM		AD	R1,R3,I1

NAME	MNEMONIC	TYPE	CODE	OPERANDS (Assembler Format)
Set Program Mask	SPM	RR	0D	R1
Shift and Round Decimal	SRP	SS	F0	D1(L1,B1),D2(B2),I3
Shift Left Double	SLDA	RS	8F	R1,D2(B2)
Shift Left Double Logical	SLDL	RS	8D	R1,D2(B2)
Shift Left Single	SLA	RS	8B	R1,D2(B2)
Shift Left Single Logical	SLL	RS	89	R1,D2(B2)
Shift Right Double	SRDA	RS	8E	R1,D2(B2)
Shift Right Double Logical	SRDL	RS	8C	R1,D2(B2)
Shift Right Single	SRA	RS	8A	R1,D2(B2)
Shift Right Single Logical	SRL	RS	88	R1,D2(B2)
Start I/O	SIO	RR	02	R1
Store	ST	RX	50	R1,D2(X2,B2)
Store (Floating Point)	STD	RX	60	R1,D2(X2,B2)
Store Character	STC	RX	42	R1,D2(X2,B2)
Store Characters Under Mask	STCM	RS	BE	R1,M3,D2(B2)
Store Control	STCTL	RS	B6	R1,R3,D2(B2)
Store Diagnostic Data	STDD	S	9B	D1(B1)
Store Halfword	STH	RX	40	R1,D2(X2,B2)
Store Multiple	STM	RS	90	R1,R3,D2(B2)
Subtract	SR	RR	1B	R1,R2
Subtract	S	RX	5B	R1,D2(X2,B2)
Subtract Decimal	SP	SS	FB	D1(L1,B1),D2(L2,B2)
Subtract Halfword	SH	RX	4B	R1,D2,(X2,B2)
Subtract Logical	SLR	RR	1F	R1,R2
Subtract Logical	SL	RX	5F	R1,D2(X2,B2)
Subtract (Floating Point)	SDR	RR	2B	R1,R2
Subtract (Floating Point)	SD	RX	6B	R1,D2(X2,B2)
Supervisor Call	SVC	RR	0A	I
Supervisor Call Exit	SVCX	RR	27	
Test Under Mask	TM	SI	91	D1(B1),I2
Translate	TR	SS	DC	DL(L,B1),D2(B2)
Translate and Test	TRT	SS	DD	D1(L,B1),D2(B2)
Unpack	UNPK	SS	F3	D1(L1,B1),D2(L2,B2)
Unpack Unsigned	UNPU	SS	F4	D1(L1,B1),D2(L2,B2)
Unpal	UNPAL	SS	DB	D1(L1,B1),D2(L2,B2)
Zero and Add	ZAP	SS	F8	D1(L1,B1),D2(L2,B2)
Convert Block-in-extent-list to Block-on-volume	BFBV	RR	0E	R1,R2
Fix Page in Frame	FIX	RS	B2	R1,R3,D2(B2)
Scan Page Frame Table	SPFT	RS	AE	R1,R3,D2(B2)
Set Time Slice	STS	SS	C7	D1(R1,B1),D2(B2)
Unfix Page On Frame	UNFIX	RS	B3	R1,R3,D2(B2)

NOTE that machine instructions for arithmetic and logical operations fall into three classes: fixed-point arithmetic, decimal arithmetic, and logical operations. These classes differ in the data formats used, the registers involved, the operations provided, and the way field length is stated. A more detailed discussion of operation 'classes' is presented on pages 13 through 22 in the 8300 Principles of Operation manual (WL #800-1100PO).

INSTRUCTION EXECUTION:

Normally, the operation of the CP is controlled by machine instructions taken in sequence. An instruction is fetched from a location specified by the instruction address in the 'current PCW'. The instruction address is then increased by the number of bytes in the fetched instruction to address the next instruction in sequence. The instruction is then executed and the same steps are repeated using the new value of the instruction address.

A change from this sequential operation may be caused by branching, status switching, interruptions, or manual intervention.

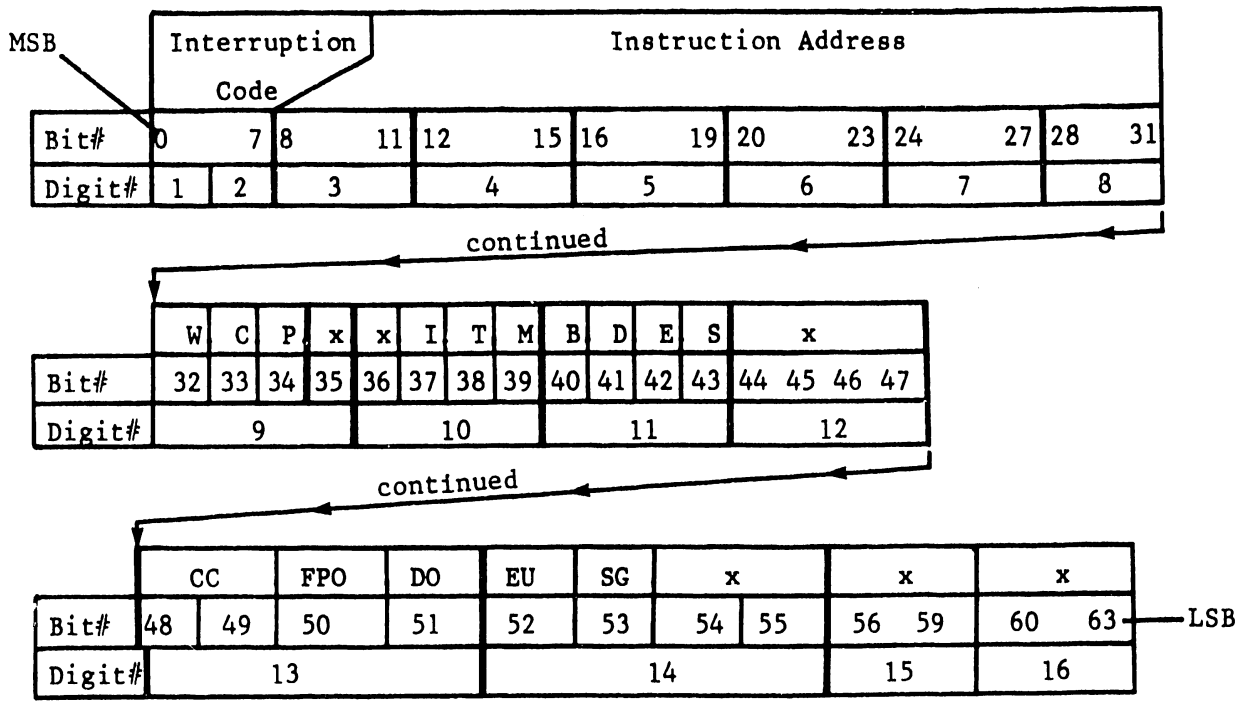
Branching

The normal sequential execution of instructions is changed: a) when reference is made to a subroutine, b) when a two-way choice is encountered, or c) when a segment of coding, such as a loop, is to be repeated. All these tasks can be accomplished with branching instructions. Provision is made for subroutine linkage, permitting not only the introduction of a new instruction address but also the preservation of the return address (multi-level subroutine stacking).

The Program Control Word

The Program Control Word (PCW), 8 bytes long, contains the information required for proper execution of machine instructions. It

includes status and control information, interruption codes and the instruction address. In general, the PCW is used to control instruction sequencing and to hold and indicate the status of the system in relation to the program currently being executed. The active or controlling PCW is called the "Current PCW". By storing the current PCW during an interruption, the status of the CP can be preserved for subsequent inspection and resumption of a task. By loading a new PCW, the state of the CP can be initialized or changed. The PCW is made up of a one byte interruption code, a three byte instruction address, a two byte status field, a one byte program mask field and one byte recovered for later options. The following figures show PCW format.



PCW digits 1 and 2 are defined saisfactorily in the PCW Conversion Chart on page 3-40. For more explicit definitions of PCW digits 9 through 16, see pages 3-41 and 3-42.

TABLE 3-5

PROGRAM CONTROL WORD (PCW)
CONVERSION CHART

BIT # / DIGIT #	8 BIT	4 BIT	2 BIT	1 BIT
1		Floating pt. Exception	Addr. Transl. Exception	Debugging Aid
			STACK OVERFLOW	
2	Digit #2= Programming Errors:			
	1=Operation	4=Protection	7=Data	A=Dec Overfl.
	2=Privileged Op.	5=Addressing	8=Fixed Pt. Ovf.	B=Dec Divide
	3=Execute	6=Specification	9=Fixed Pt. Div.	C=Supv Call R
3	NEXT INSTRUCTION ADDRESS	NIA	NIA	NIA
4	NIA	NIA	NIA	NIA
5	NIA	NIA	NIA	NIA
6	NIA	NIA	NIA	NIA
7	NIA	NIA	NIA	NIA
8	NIA	NIA	NIA	NEXT INSTRUCTION ADDRESS
9	W Wait State	C Control Mode	P Mem. Protect & Privileged Instr. Trap	RESERVED
10	RESERVED	I I/O Interrupt Mask	T Clock Interrupt Mask	M Mach. Check Interpt Mask
11	B PCW Address Compare Trap	D Virtual Address Mod. Trap	E Phys. Address Mod. Trap	S Single Step Trap
12	RESERVED	RESERVED	RESERVED	RESERVED
13	CC Condition Code	CC Condition Code	FPO Fixed Point Overflow Mask	DO Decimal Overflow
14	EU Exponent Under-Flow Mask	SG Significance Mask	RESERVED	RESERVED
15	RESERVED	RESERVED	RESERVED	RESERVED
16	RESERVED	RESERVED	RESERVED	RESERVED

PCW DIGIT #	PCW BIT #	NAME GIVEN IN PCW CONVERSION CHART	MEANING
	32	W	Wait state: 0= Operating state 1= Wait state
	33	C	Control mode: 0= Normal operating mode 1= Control mode
9	34	P	Memory protection violation and priviledged instruction trap: 0= Do not trap on memory protection violation or priviledged instruction 1= Trap on memory protection violation or priviledged instruction
	35		Reserved
	36		Reserved
	37	I	I/O Interruption mask: 0= I/O Interrupts disabled 1= I/O Interrupts enabled
10	38	T	Clock Interruption mask: 0= Clock Interrupts disabled 1= Clock Interrupts enabled
	39	M	Machine check Interruption mask: 0= Machine check Interrupts disabled 1= Machine check Interrupts enabled
	40	B	PCW address compare trap: 0= No PCW address compare trap in effect 1= Trap on PCW address compare equal
11	41	D	Virtual address modification trap: 0= No virtual address modification trap in effect 1= Trap on unequal compare with byte at specified virtual address

PCW DIGIT #	PCW BIT #	NAME GIVEN IN PCW CONVERSION CHART	MEANING
	42	E	Physical address modification trap: 0= No physical address modification trap in effect 1= Trap on unequal compare with byte at specified physical address
	43	S	Single step trap: 0= No step exception 1= Trap after execution of next instruction
12	44-47		Reserved
13	48-49	CC	Condition Code
	50	FPO	Fixed-point overflow mask: 0= Do not interrupt on overflow 1= Overflow will cause interrupt
	51	DO	Decimal overflow: 0= Do not interrupt on overflow 1= Overflow will cause interrupt
14	52	EU	Exponent underflow mask (Floating-point instructions)
	53	SG	Significance mask (Floating-point instructions)
	54-55		Reserved
15	56-59		Reserved
16	60-63		Reserved

3.2.4 INTERRUPTS

3.2.4.1 General

An interrupt is an error condition or a request-for-assistance condition that will cause a break in the normal sequence of instruction execution. Should such a condition occur, the system supervisor seizes control and action is taken to either flag/log/correct the error condition or to service the request for assistance. An interrupt can occur after the execution of one instruction and before the execution of the next instruction. Instructions are said to have been completed, terminated, aborted, or suppressed at the time an interrupt occurs.

An 'interrupt' system permits the 2200VS Operating System to change state as a result of conditions external to the system, conditions in I/O devices, or conditions in the CP itself. Five classes of interrupt conditions are possible: I/O, clock, program, supervisor call and machine check.

Simultaneous requests for interruptions at the end of an instruction are honored in the following order of priority (the conditions are listed in descending order of priority):

Machine Check
Supervisor Call
Program
Clock
Input/Output

3.2.4.2 Types of Interrupts

Machine Check Interruption:

The machine check interruption provides a means for reporting the occurrence of machine malfunctions to the Operating System. Information is provided to assist the Operating System in determining the location of the fault.

The cause of the malfunction is identified by the interruption code. An interruption code of 1 indicates a main memory parity error. An interruption code of 2 indicates an unexpected interruption request from an I/O processor (IOP). A machine check interruption may be masked off by turning off the machine check interruption mask bit in the PCW. A machine check interrupt that has been masked off causes entry to control mode.

Any program or supervisor-call interruptions that would have occurred as a result of the current operation are eliminated. Any instruction in progress when a machine check occurs is aborted.

Supervisor Call Interruption:

The supervisor-call interruption occurs as a result of the execution of the SUPERVISOR CALL instruction. It causes the current PCW and other information to be stored in the system stack in Main Memory and a new PCW is constructed.

The name "supervisor call" indicates that one of the major purposes of the interruption is the switching of the Operating System from problem to supervisor state.

:Program Interruption:

Exceptions resulting from improper use of instructions and data cause a program interruption.

The current instruction is completed, terminated, aborted, or suppressed. Only one program interruption occurs for a given instruction. The occurrence of a program interruption does not preclude the simultaneous occurrence of other program-interruptions.

A description of the individual program exceptions follows. Some of the exceptions listed may also occur in operations resulting from I/O instructions.

Program Interruption Codes

Programming Errors	
Operation	01
Priviledged Operation	02
Execute	03
Protection	04
Addressing	05
Specification	06
Data	07
Fixed Point Overflow	08
Fixed Point Divide	09
Decimal Overflow	0A
Decimal Divide	0B
Supervisor Call Range	0C
Debugging Aids	
PCW Trap	10
Virtual Destination Trap	11
Physical Destination Trap	12
Single Step Trap	13
Address Translation Exception	20
Stack Facility	
Stack Overflow	30
Floating Point Exceptions	
Floating Point Overflow	40
Floating Point Underflow	41
Significance	42
Floating Point Divide	43

Clock Interruption:

The clock interruption provides a means by which the CPU responds to timing conditions set within the system. For example, a clock interruption becomes 'pending' whenever the time-of-day clock value is greater than or equal to the clock comparator value.

Input/Output Interruption:

A request for an I/O interruption may occur at any time, and more than one request may occur at the same time. The requests are preserved in the I/O device until accepted by the processor. More than one event which establishes a pending interrupt may occur at a device. Each such event is recorded at the device and, when the I/O interruption for the device is taken, the stored IOSW (I/O Status Word) reflects the occurrence of all such events. Priority is established among devices so that only one I/O interruption request is processed at at time.

3.3 THE INPUT/OUTPUT PROCESSOR (7110 pc)

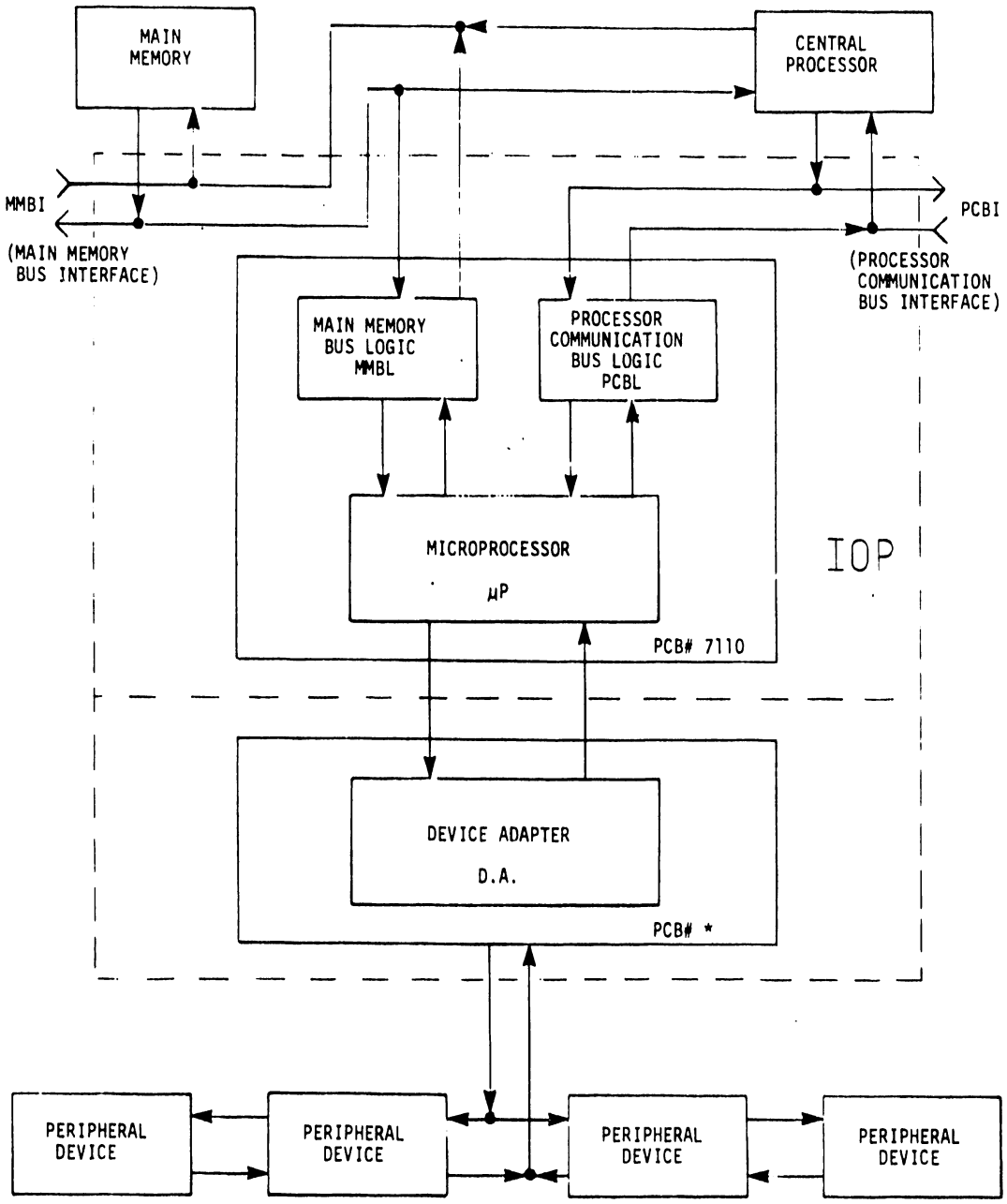
The Input/Output Processor (IOP) is the heart of the interface between the central processor and the user. All transfer of information between the Central Processor (C.P.) and its peripheral equipment is directed through the IOP.

The IOP is used to control input and output requests. Starting of the IOP is under the control of the CP, whereas stopping the IOP can be controlled by the IOP or the CP. Once started, the IOP processes independently of the CP.

Initialization

Initialization of microprograms and hardware elements can be triggered by using either the Power-on or Load buttons on the CP. Initialization involves resetting bus controllers and I/O

FIGURE 3-5 IOP BLOCK DIAGRAM



*DEPENDENT UPON TYPE OF I/O DEVICE ATTACHED

devices to a neutral state and activating the appropriate microprogram routines in the microprocessors (by means of a "trap"). The details of initialization will be defined by the hardware group.

NOTES:

1. An initialize line on the PCB is used to initialize IOP's and I/O devices.
2. Initialization causes all MMB and PCB lines to be 'dropped' (i.e., no bus activity after initialization).

Resetting of the I/O Devices

All I/O devices are reset when the LOAD button is pushed, or when a system power-on sequence is completed. This causes the I/O devices to terminate all I/O operations. Status information and interruption conditions in the devices are reset. Both data transfer operations and control operations are immediately terminated and the results are unpredictable.

Note that each device has a one byte (8 bit) device address. All values from 00 to FF are legitimate device addresses. The current IOP supports four devices; the high-order six bits are therefore the IOP address and the low-order two bits are the device address.

In the following paragraphs, each major functional block of the IOP is discussed.

3.3.1 THE MICROPROCESSOR (MP)

The Microprocessor is to the IOP what the CP is to the overall 2200VS. The Microprocessor consists of a variety of elements which enable the manipulation of I/O data, I/O communication, sequencing of I/O instructions, and the activating of 'interrupt' sequences.

3.3.1.1 Register Structure

Register mobility within the MP is accomplished through two 8-bit paths and a 'path master'; the path master selects the registers specified, places the data in the input bus (called the A-bus), and gates them through a function generator called the ALU. The output of the ALU is placed in the output bus (called the C-bus), the path master then gates the result to the specified register.

The following paragraphs describe these registers, the tasks in which they are involved, and their paths of interaction.

Memory Data Register (MDR):

The Memory Data Register, also called the 'port register', is used as the point of exchange between Main Memory data and the IOP Microprocessor. This register is divided into two 8-bit registers, MDRL and MDRH. MDRL is used as the exchange point between the M.P. and the Processor Communication Bus (PCB) interface. Both registers are used for command and information sequencing at the Device Adapter (DA). Both MDRL and MDRH are used in logical and arithmetic processing.

Memory Address Register (MAR):

The Memory Address Register (MAR) consists of three sections, a 3-bit MARH, an 8-bit MARM, and an 8-bit MARL. These registers are used to point to a predetermined location in Main Memory where some form of data transfer is to take place between Main Memory and the IOP Microprocessor. This register may be incremented or decremented by one or two.

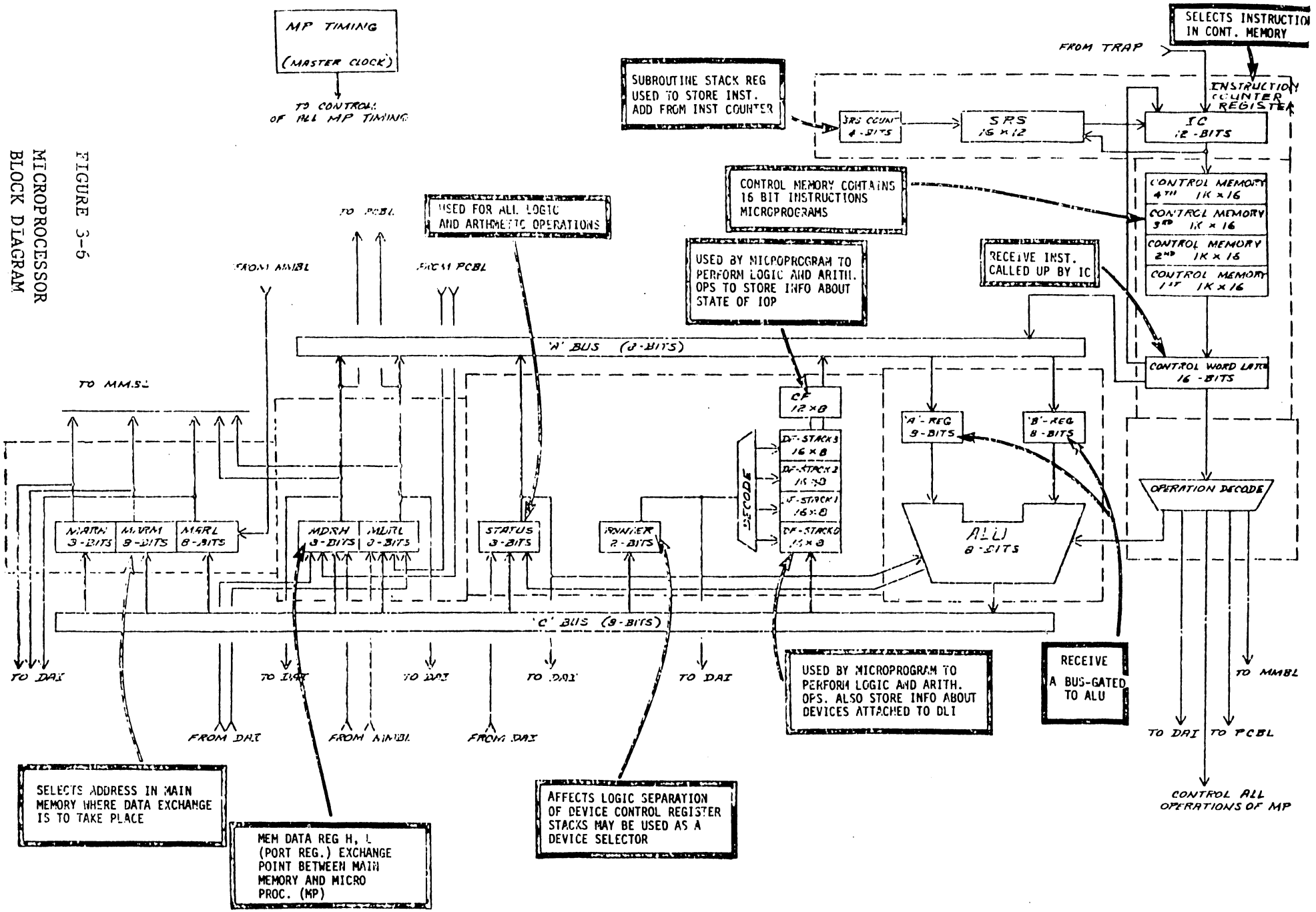


FIGURE 3-6
MICROPROCESSOR
BLOCK DIAGRAM

Establishing a new Main Memory address in IOP MAR may be accomplished by sequentially incrementing or decrementing the MAR value (±1 or ±2), or by parallel loading the MARH, MARM, and MARL eight bits at a time (the IOP uses an 8 bit bus structure, while the CP uses a 16 bit 'halfword' bus structure). Note that the IOP addresses only physical (Main) memory, not virtual memory.

File Control Registers (CF):

The File Control Registers are used by the microprogram to perform logical and arithmetic operations, to store status and control information pertaining to the general state of the IOP.

Device Control Registers (DF):

The Device Control Registers are divided into four groups of 16 eight bit registers each. They are used by the microprogram to perform logical and arithmetic operations and to store status and control information pertaining to the general state of the devices attached to the Device Level Interface (D.L.I.; 'device adapter').

Pointer Register (PT):

The Pointer Register is a 2-bit register that is used to select one of four devices attached to the Device Adapter; or, it can select any one of four sets of Device Control Registers (1 set = 16 registers), each belonging to one of the four devices attached to the Device Adapter.

Status Register (S):

The Status Register is an 8-bit register that can be used for all logical and arithmetic operations, except ADD with carry (AC), in the IOP Microprocessor.

FORMAT:

0	1	2	3	4	5	6	7
U1	IN	SV	RB	U2	C2	C1	CA

U1 - Undefined.

IN - Input - This bit is set to '0' by the CDLI microinstruction (hardware). The DLI sets this bit to '1' when responding to a CDLI microinstruction.

SV - Service - This bit is conditionally set by the PCBI to '0'.

RB - Ready/Busy - This bit is tested by the PCBI.

U2 - Undefined.

C2 - Greater/Less - This bit is set as a result of the compare instruction.

C1 - Equal - This bit is set as a result of the compare instruction.

CA - Carry - This bit is used and set in the add with carry instruction.

Instruction Counter Register (IC):

The IOP Instruction Counter Register is a 12-bit register which selects IOP Control Memory microinstructions. Normally, IOP microinstructions are executed sequentially. The IOP IC can be conditionally incremented by 2 (skip microinstruction), or may be loaded with a value from the IOP Subroutine Stack, from the IOP Instruction Register, or from the IOP Trap Handler.

Instruction Register (IR):

The IR receives the 16-bit instruction code addressed by the IC. Eight of the Instruction Register's bits are sent to the IOP's register-selecting logic (called the 'path master') for immediate data

instructions. Also, 11 bits are made available to the DLI, the MMBI, and the PCBI as control information. The Instruction Register is also called the 'Control Word Latch' (CWL).

Subroutine Stack Registers (SRS):

The Subroutine Stack Registers are a group of sixteen 12-bit registers. Each of the sixteen registers is used to store the address of an IOP microprogram instruction. Each can then be used as a subroutine return address which is loaded into the IOP IC when return conditions are met.

A and B Registers:

These registers receive the IOP's A-bus. Their contents are gated to the IOP ALU.

3.3.1.2 Arithmetic Logical Unit (ALU)

At the center of the microprocessor is the ARITHMETIC LOGICAL UNIT (ALU). The ALU is made up of an input section (2 latches), the ALU itself, an ADD w/carry circuit, and a compare circuit. The output of the ALU is the C-Bus. All arithmetic and logical functions are processed through this network. Certain instructions (CDLI, CMBI, CPBI, SR, SB, and BU) are not arithmetic or logical and do not use the ALU.

The IOP ALU is an 8-bit unit and is fed by two 8-bit input latches, the A-Register and B-Register. Two operands (values to be acted upon) are loaded into the ALU through the A and B Registers. When the ALU is loaded, it processes the two operands according to the controlling microinstruction (example: ADD). The resultant is placed on the C-Bus. The microinstruction also specifies a target register, where the result is to be placed.

3.3.1.3 Control Memory (CM)

IOP Control Memory is a PROM storage block, 4096 x 16 bits. Each of the 4,096 sixteen-bit microcode instructions has 2 parity bits attached, thus making each IOP microinstruction 18 bits wide. These IOP microprograms are used to control each peripheral device and manipulate data in the IOP.

3.3.2 MAIN MEMORY BUS/LOGIC

Main Memory Bus Logic is that portion of the IOP which controls all data transfers between the IOP and main memory. Main Memory Bus Logic (MMBL) resides on the Microprocessor board. The actual operations controlled by the MMBL are: READ 16 bits, WRITE 16 bits, WRITE 8 bits and receive Main Memory error conditions when they occur. When the IOP has data to send to the CP, the data is written into Main Memory locations which begin at an address calculated by the IOP. The CP then reads Main Memory, beginning at that address. The same is true in reverse. The CP would give the addresses of the data needed by the IOP, and the IOP would then read those locations of Main Memory.

Operations involving the IOP and Main Memory are controlled by a "Control (the) Memory Bus Interface" (CMBI) microinstruction. When, during a microinstruction routine, the CMBI instruction is detected, the Main Memory Bus Logic requests a memory cycle from Main Memory. No other microinstructions will be processed by the IOP until the memory transfer is complete. Note that all IOP's in the system will be requesting memory cycles; and when more than one IOP requests access to memory a the same instant, the main memory controller will grant access to Main Memory by a priority basis determined by the physical location of each IOP in the 2200VS chassis.

3.3.3 PROCESSOR COMMUNICATION BUS LOGIC (PCB)

The PCB is that portion of the IOP that controls all communication (handshake, protocol, etc.) between IOPs and the CP. There are only two basic operations performed using the Processor Communication Bus (PCB): 'Command-Out' and 'Grant-Interrupt'. The operations are independent, not time-critical, and do not overlap as far as the PCB hardware is concerned. The operations are basically shoulder tap operations between the CP and an IOP; all decisions and responses involving the state of a particular I/O device are handled by the IOP.

The normal communication between the CP, main memory, and the IOPs is as follows. The CP initiates a Start I/O (SIO) sequence. The IOP receives the SIO and either accepts or rejects it. An IOP would reject the SIO sequence due to that IOP or device being busy.

IOP BUSY

During certain critical operations required to control its I/O devices, an IOP may be unable to accept a SIO or HIO (Halt I/O). This condition is of limited duration and is relatively infrequent. When the IOP is unable to accept a SIO or HIO, a condition code indicating IOP BUSY is returned. The conditions and times when an IOP will respond BUSY are device dependent. Once an IOP has responded to an instruction with an indication of IOP BUSY, it will present an IOP NOW READY interrupt after the BUSY condition clears. Only one IOP NOW READY interrupt will be presented no matter how many SIOs are rejected.

3.3.3.1 Input/Output Interruptions

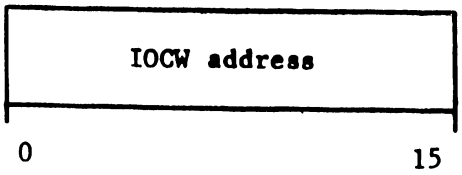
Input/Output interruptions provide a means for the system to change its state in response to conditions that occur in I/O devices and IOP's. These conditions are caused by termination of an I/O operation or by operator intervention at the I/O device.

These conditions cause three types of I/O interruptions, solicited, unsolicited, and IOP NOW READY. A solicited interruption is caused by the completion of an I/O operation initiated by the CP. An unsolicited interruption is caused by operator action at the I/O device such as mounting a disk pack or striking a work station attention key. An IOP NOW READY interruption is caused by an IOP becoming available for acceptance of SIOs and HIOs after having reported IOP BUSY in response to one of these instructions.

If an SIO sequence is accepted by the IOP, the IOP traps to a predetermined location in the IOP microcode. This location (address HEX 04) is a BRANCH microinstruction, directing the microprocessor to address another location in IOP Control Memory. There, a microcoded (subroutine) determines which device attached to a given IOP has been requested for service. The microprogram will then calculate an address in Main Memory where the Input/Output Command Address (IOCA) is located for that particular device.

I/C COMMAND ADDRESS

The I/O command address (IOCA) area starts at main storage location 128 and contains a half-word entry for every possible I/O device address from zero to the highest device address attached to the system (up to 255). The IOP uses the device address received on an SIO instruction as an index into the IOCA area. The IOCA has the address of the I/O command word (IOCW) to be executed.



IOCA Format

The IOCA contains the 16-bit address of the IOCW.

Calculate IOCA address by multiplying the perpheral device address (the 'IOCA index') by 2, and adding that answer to binary 128.

Example of Computation

	(HEX)	BINARY
Device Address =	21	0010 0001
Multiply by 2 =	42	0100 0010
Add 128 (Binary)		+1000 0000
IOCA Address	C2	1100 0010

NOTES:

1. The device address has a six-bit IOP portion and a two-bit peripheral device portion.

   Example:

   IIIIIIPP = 4 DEVICE IOP

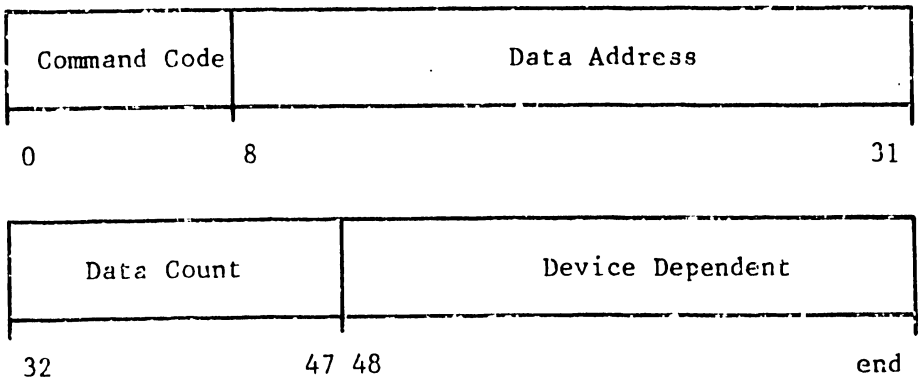
   I=IOP portion of device address (4 or 6 bits) total of 64 4-Device IOPs.P=Peripheral device portion of device address(2 bits) total of 4 devices per IOP.

2. For the purpose of address computation, the IOP portion and the device protion of the address is treated as a single 8-bit address.

This calculated address is moved into the IOP MAR. The IOP microprogram then reads 16-bits from main memory at that calculated address. This results in the IOCA being placed in the IOP MDRH and MDRL registers. The IOCA is an address in Main Memory where the Input/Output Control Word (IOCW) is located. The microprogram does another read from Main Memory at the address specified by the IOCA. The IOCW, read from Main Memory by the IOP, is then stored in the IOP.

The IOCW specifies the command to be executed and contains all information necessary to perform any task involving a peripheral device. Such tasks would include reading from an I/O device, writing to an I/O device, or controlling an I/O device. The IOCW tells the IOP where in main memory any transfer is to begin, how many bytes of data are to be transferred, or in the case of control functions, the initiating of operations with a specific device not involving data transfer (such as a skip to heading on a printer, or a restore on a disk device).

The IOCW consists of a six byte general section and is followed by a device dependent section. The device dependent section can be of any length, but is fixed for each device. The IOCW must be fullword aligned.



I/O COMMAND WORD FORMAT

The fields in the IOCW are allocated for the following purpose:

'Command Code' - Bits 0-7 specify the operation to be performed.

'Data Address' (DA) - Bits 8-31 specify the physical address of an eight-bit byte in main memory, which must be fullword aligned. This byte location is the beginning of the data area for the specified operation, or is the beginning of an Indirect Data Address List, which in turn addresses the data area(s) for the operation. The entire Indirect Address List must reside within the first 64K bytes of physical memory.

'Data Count Field' (DC) - Bits 32-47 specify the number of eight-bit byte locations in memory to be transmitted either to or from the device.

Command Code

The command code, bit positions 0-7 of the IOCW, specifies to the I/O device the operation to be performed.

Bits 0 and 1 of the command code are the command type, and bits 2-7 are the command modifier bits. The following four command types are defined:

1) Reserved - '00'
2) Read - '01'
3) Write - '10'
4) Control - '11'

Commands

'Reserved' - This code is reserved.

'Write' - A write operation is initiated at the I/O device, and data is transferred from main memory to the I/O device. Data in memory is fetched in ascending order of addresses, starting with the address specified in the IOCW.

'Read' - A read operation is initiated at the I/O device, and data is transferred from the device to main memory. Data in memory is placed in ascending order of addresses, starting with the address specified in the IOCW.

'Control' - A control operation is initiated at the I/O device. A control command is used to initiate an operation not involving transfer of data. For most control functions, the entire operation is specified by the modifier bits in the command code. If the command code does not specify the entire control function, the device dependent field of the IOCW can be used. The data address field is always ignored for a control command.

Command Modifier Bits

The use of the modifier bits is device dependent. The modifier bits of the command specify to the device how the command is to be executed. The fifth modifier bits (bit 6 of the command code) is set to indicate Indirect Data Addressing for those devices which support that option.

When the IOCW designated contains an invalid field, an I/O interrupt is generated with the invalid condition indicated in the IOSW.

3.3.3.2 I/O Task Termination/Completion

Upon completion of a task described by an IOCW, the IOP microprogram activates the PCB "Request-In-Line". This request is referred to as a "solicited interrupt". The IOP will leave this request line active until granted service by the CP. The CP will acknowledge this PCB request on a priority basis. When acknowledged, the IOP then writes an Input/Output Status Word (IOSW) into main memory location "0". This IOSW will contain all necessary information as to the 'completion status' of the I/O operation that the IOP just performed.

Normally, an I/O operation is 'completed'; however, the system can force an I/O operation to terminate prematurely under certain other conditions:

Data Transfer Terminations:

When the device accepts a data transfer command, the operation can be terminated by one of the following five conditions:

1. A HALT I/O instruction was issued to the device.
2. The count field in the IOCW has gone to zero. (IOCW exhaustion.)
3. As many bytes have been transferred as are indicated by the sum of the lengths specified in an Indirect Address List. (List exhaustion.)
4. The device has indicated that there is no more data to be transferred. (Data exhaustion.)
5. Hardware malfunction.

The end condition causes the operation to be terminated and an interruption condition to be generated. The status bits in the associated IOSW indicate the reasons for termination. The device can signal termination at any time after initiation of the operation and the signal may occur before any data has been transferred. The duration of data transfer operations is variable and is controlled by the device and its IOP.

HALT I/O Terminations:

If accepted by the IOP, instruction HALT I/O causes the current operation at the addressed device to be terminated immediately.

If an interruption for the addressed device was pending, that interruption remains pending. If an I/O operation was active, the operation is terminated and a completion interruption becomes pending.

Equipment Malfunction Terminations:

When equipment malfunctioning is detected, the recovery procedure and the subsequent states of the devices depend on the type of error. Normally, the device attempts all appropriate error recovery procedures. If the recovery is successful, the I/O operation is completed and the IOSW indicates a soft error. If the recovery is unsuccessful, the operation is terminated, and a hard error is indicated in the IOSW.

An IOSW is stored for every I/O interrupt, and is of the following format:

COMPLETION STATUS	IOP/DEVICE STATUS	RETRY COUNT	DEVICE-DEPENDENT STATUS BYTES (DEVICE PROBLEM DESCRIPTION)	RESIDUAL BYTE COUNT
0 3	4 15	16 19	20 31	32 47

COMPLETION STATUS:

IRQ - Intervention Required

This bit is set with error completion (EC) and without normal completion (NC) to indicate that the device was in a not-ready state when a Start I/O was accepted, or that no device with the specified device number was attached to the specified I/O processor. This condition requires operator intervention to return the device to the ready state.

NC - Normal Completion

This bit is set to indicate completion of an I/O operation without permanent error. An interruption with NC or EC set will occur exactly once for each SIO accepted.

EC - Error Completion

This bit is set to indicate completion with error of an I/O operation. If NC is also set, the operation was successful after at least one retry by the device or IOP. If this bit is set, the errors detected will be indicated in the error status byte or device dependent status bytes, whether or not NC is also set.

Thus we have:

NC	EC	
0	0	Completion not indicated
1	0	Normal completion
0	1	Completion with permanent error
1	1	Completion with corrected error

U - Unsolicited (Attention/Device Now Ready)

This bit is set when the device signals an unsolicited interrupt. An unsolicited interrupt is one not caused by I/O completion. This indicates that either the device has become available for I/O operations or that a user is signalling the CP (attention). This bit is independent of, but may be set with the NC, EC or PC bits on.

PC - IOP Now Ready

This is an indication that an IOP may now accept a SIO. This bit can be set in conjunction with NC or EC (I/O completion) or U (unsolicited). Whenever a SIO is rejected with condition code 2 (IOP BUSY), an interruption with PC set will eventually be presented. If more than one SIO to devices on the same IOP is rejected with condition code 2 without an intervening interruption with PC set, then only one interruption with PC set will be presented.

IOP/DEVICE STATUS:

IC - Invalid Command

This indicates that part of the IOCW or the device dependent control information was invalid (e.g., invalid command code, invalid data address alignment, etc.). This condition also causes hard error to be indicated.

MPE - Memory Parity Error

Memory parity error is indicated whenever there is a parity error while the IOP associated with the I/O device is accessing memory. This is the method by which a machine check is indicated during an I/O operation.

MAE - Memory Address Error

Memory address error is indicated whenever an attempt is made to address outside of the available memory on the machine during an I/O operation. This is the method by which an addressing exception is indicated during an I/O operation.

DM - Device Malfunction

Device malfunction indicates that an equipment error has occurred during an I/O operation or that the I/O operation cannot be completed normally. Device malfunction is not indicated in the case where operator intervention will correct the problem. Thus device malfunction is not indicated when intervention required (IRQ) is set.

DAM - Memory or Device Damage

This bit indicates that the data transfer was interrupted while in process and that either the data at the device or in memory has been changed. This indicates that the receiver of the data trans

mission has unpredictable data, and the data must be retransmitted (if possible) to correct the problem. This may also mean that the device's status has changed (e.g., for a magnetic tape, the tape has been repositioned). This bit will be set only if the hard error indication is set.

IL - Incorrect Length

This bit is set if the length of the data specified in the data count of the IOCW and length of the corresponding item of data at the device were different. If this bit is set, this will cause the error completion bit (EC) to be set. If this bit is set and the device supports storing of the residual data count, a valid residual data count will be stored.

RETRY COUNT:

Self explanatory.

DEVICE DEPENDENT STATUS BYTES:
(DEVICE PROBLEM DESCRIPTION)

Identifies such peripheral device failures as CRC, LRC, 'short sector', sector overrun, compare error, invalid address.

RESIDUAL BYTE COUNT:

This indicates the byte count remaining at the time of I/O completion. Not all devices support storing of the byte count.

The IOCW and the IOSW are two of the most useful pieces of information in the troubleshooting and repair of the system, and/or the various peripheral devices attached to the system.

TABLE 3-6

DISK IOSW CONVERSION CHART

DIGIT # \ BIT	8 BIT	4 BIT	2 BIT	1 BIT
1	I R Q INTERVENTION REQ'D	NC NORMAL COMPLETE	EC ERROR COMPLETE	U UNSOLICITED
2	PC IOP NOW READY	0 ALWAYS ZERO	0 ALWAYS ZERO	0 ALWAYS ZERO
3	IC INVALID COMMAND	MPE MEMORY PARITY ERROR	MAE MEMORY ADDRESS ERR	DM DEVICE MALFUNCTION
4	DAM MEM OR DEVICE DAMAG	IL INCORRECT LENGTH	0 ALWAYS ZERO	0 ALWAYS ZERO
5	SRW SECT REFORMAT-WRITE	HSR HEADER SKIPPED-READ	0 ALWAYS ZERO	0 ALWAYS ZERO
6	IDA INVALID DISK ADDRESS	IDC INVALID DATA COUNT	SO SECTOR OVERRUN	SI SEEK INCOMPLETE
7	WP WRITE PROTECT	NRO NOT RDY DURING OPN	ST SECTOR TIMEOUT	DC DATA COMPARE ERROR
8	IID INVALID SECTOR ID	CRC INVALID CRC OR ECC	0 OVERRUN (DATA FIFO)	ISP SHORT SECTOR
9	RDC RESIDUAL DATA COUNT	RDC	RDC	RDC
10	RDC	RDC	RDC	RDC
11	RDC	RDC	RDC	RDC
12	RDC	RDC	RDC	RDC RESIDUAL DATA CO
13	RETRY SETUP (CDC) 0=NORMAL 1=DATA STROBE EARLY 2=DATA STROBE LATE 3=OFFSET 4=OFFSET+ 5=EARLY & - 6=EARLY & + 7=LATE & - 8=LATE & + 9=HARD ERROR (ECC USED)			
14	RETRY COUNT FOR SETUP IN DIGIT 13			
15				

PCB SIGNALS:

The PCB consists of 28 lines as follows:

1. PCB Grant-Out Strobe line (8). PCB Request-In lines (8; one per IOP).

These 16 lines are used in the PCB Grant-Interrupt operation.

2. PCB Device Address lines (8).

These bi-directional lines are used to transfer an I/O device address.

a. From IOP to CP - Grant-Interrupt Operation. Also Command-Out Operation response.
b. From CP to IOPs - Command-Out operation.

3. PCB Control lines (2).

These bi-directional lines are used to transmit a command from the CP to an IOP or to transmit a response from and IOP to the CP.

a. Command (CP to addressed IOP) Command-Out Operation.
   00 - Alert
   01 - Start
   10 - Stop
   11 - Undefined

   Note: The PCB Hardware is not concerned at all with the above codes. Only the IOP distinguishes between the different commands.

b. Response.
   Command-Out Operation (addressed IOP to CP).
   Grant-Interrupt Operation (selected IOP to CP).
   00 - Available
   01 - Device Busy
   10 - IOP Busy
   11 - Device not operable

   Note: The IOP can generate the IOP busy code based on a simple yes/no condition at the IOP. Other than this case, the PCB hardware is not concerned with these codes.

4. PCB Control-Out Strobe line (1).

A strobe line from the CP to the IOP is used to strobe a command (PCB Control lines) and address during the Command-Out operation.

5. PCB Control-In Strobe line (1).

A strobe line from the IOP to the CP is used to strobe in a response (PCB Control lines) for either PCB operation. (Response includes device address (from IOP to CP), using PCB Device Address lines.)

Summary of PCB Lines

Common bi-directional lines

1. PCB Device Address lines (8)
2. PCB Control lines (2)

Common lines (CP to IOPs)

1. PCB Control-Out Strobe (1)

Common lines (IOPs to CP)

1. PCB Control-In Strobe (1)

Independent lines (one per IOP to CP)

1. PCB Request-In lines (8)
2. PCB Grant-Out Strobe (8)

3.3.4 DEVICE ADAPTER (DLY or DA)

Each Device Adapter basically controls the flow of data and status information between a peripheral device and its IOP.

When data is to be transferred from the IOP Microprocessor to a peripheral device, the DA accepts data from the IOP MP into its Output Register. The output of this register is felt at the Device/Port selection circuitry. The pointer register stores this information which is used to determine which device the data should go to. The data is then gated to the correct port (actual plug on the DA).

When data is being passed from the peripheral device to the IOP Microprocessor, the DA accepts data from the device into its "Input Register" via the Input Data Multiplexer. This multiplexer is controlled by the Pointer register. The output of the Input Register is transferred to MDRH and MDRL.

The microinstruction within the IOP which controls the device is the CDLI instruction (CONTROL (the) DEVICE LEVEL INTERFACE). The CDLI instruction passes a command to a specific device, and that device responds with a completion bit. This microinstruction is sent to a device for the purpose of issuing a command such as a read or write. One variation of the CDLI instruction, referred to as a '5002' CDLI command, is used for the purpose of checking peripheral device status. This gives the IOP Microprocessor the capability of scanning any number of devices to find out their current status (ready, not ready, busy, out of paper, etc.).

3.3.5 THE IOP MP MICROINSTRUCTION SET

The Instruction Set is the set of microinstructions which control the activity of the microprocessor. These microinstructions, in a logical sequence, make up the microprograms stored in Control Memory. A list of the sixteen microinstructions used in the WCS 60/80 IOP follows. It is for general information only, as a more detailed explanation will be included during training:

Instructions 1 through 10 are considered "logical and arithmetic". Instructions 11 and 12 are considered "conditional", while instructions 13 through 15 are considered "unconditional". "Interface Operations" is the class name given to instructions 16 through 18. These eighteen instructions control all processing within the IOP, and are what gives the IOP the capability of processing independently from the CP.

TABLE 3-7

IOP MICROINSTRUCTION SET

		BIT NUMBER → 0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	AND	0	0	1	1	1	0	A	A	A	A	A	B	B	B	B	B
2	OR	0	0	0	0	1	0	A	A	A	A	A	B	B	B	B	B
3	XOR	0	0	0	1	0	0	A	A	A	A	A	B	B	B	B	B
4	A	0	0	0	1	1	0	A	A	A	A	A	B	B	B	B	B
5	AC	0	0	0	1	1	1	A	A	A	A	A	B	B	B	B	B
6	MV	0	0	1	0	1	0	A	A	A	A	A	B	B	B	B	B
7	C	0	0	1	1	0	1	A	A	A	A	A	B	B	B	B	B
8	LM	0	0	1	0	0	0	A	A	A	A	A	S	S	S	-	-
9	MVI	1	0	0	I	I	I	I	I	I	I	I	B	B	B	B	B
10	ORI	1	0	1	I	I	I	I	I	I	I	I	B	B	B	B	B
11	ST	1	1	1	I	I	I	I	I	I	I	I	B	B	B	B	B
12	SF	1	1	0	I	I	I	I	I	I	I	I	B	B	B	B	B
13	B	0	1	1	0	IC0	IC1	IC2	IC3	IC4	IC5	IC6	IC7	IC8	IC9	IC10	IC11
14	SB	0	1	1	1	IC0	IC1	IC2	IC3	IC4	IC5	IC6	IC7	IC8	IC9	IC10	IC11
15	SR	0	1	0	1	1	-	-	-	-	-	-	-	-	-	-	-
16	CMBI	0	1	0	0	0	-	-	-	-	-	-	D	M	M	R	R
17	CPBI	0	1	0	0	1	-	-	-	-	-	-	C	C	C	C	C
18	CDLI	0	1	0	1	0	C	C	C	C	C	C	C	C	C	C	C

Values for B B B B B:

0 0 0 0 0 to 0 1 0 1 1 - CFO to CFB

0 1 1 0 0 - PT

0 1 1 0 1 - ST

0 1 1 1 0 - MDRH

0 1 1 1 1 - MDRL

1 0 0 0 0 to 1 1 1 1 1 - DFO to DFF

Values for A A A A A:

Same as for B B B B B but with PT disallowed.

Values for S S S:

0 0 1 - MARH

0 1 0 - MARM

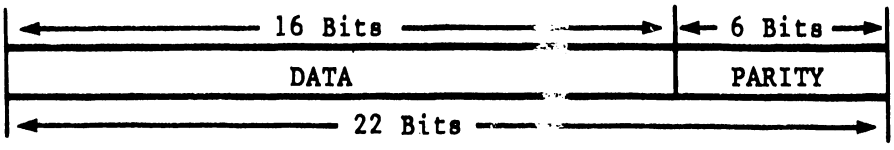
1 0 0 - MARL

3.4 MAIN MEMORY

A minimum of one, and a maximum of eight RAM cards (7104), 64K each, comprise the 2200VS Main (physical) Memory. Maximum memory size is actually 524,288 bytes. Note, however, that 'K' only expresses memory size as a decimal equivalent of 2^X. Thus, 524,288 bytes is described as 512K (2^{19}).

Memory cards are loaded with dynamic RAM, each chip having a capacity of 4K. 2200VS RAM is addressed in 16 data bit 'halfwords'. The lowest order address bit to RAM selects which 8-bit byte of the 16-bit halfword is to be used in 8-bit instructions. This selection is performed on the 7103 card. Each halfword also has six parity bits attached to it (explanation follows).

Main Memory data format is as follows:



Error Correction

An 'Error Correction Code' (ECC) gives 2200VS hardware the capability of correcting single-bit errors in data that is read from RAM (Main Memory); no error correction occurs on RAM writes.

The correction of one-bit RAM READ errors is transparent to the user; however, ECC facilities may be hardware-disabled in a 2200VS for RAM diagnostic procedures. If, during any 16-bit READ from Main Memory, more than one data bit is bad, the Central Processor 'traps' to a predetermined error routine address in Control Memory.

A simplification of 2200VS RAM/ECC circuitry follows:

MAIN MEMORY - SIMPLIFICATION OF ECC HARDWARE

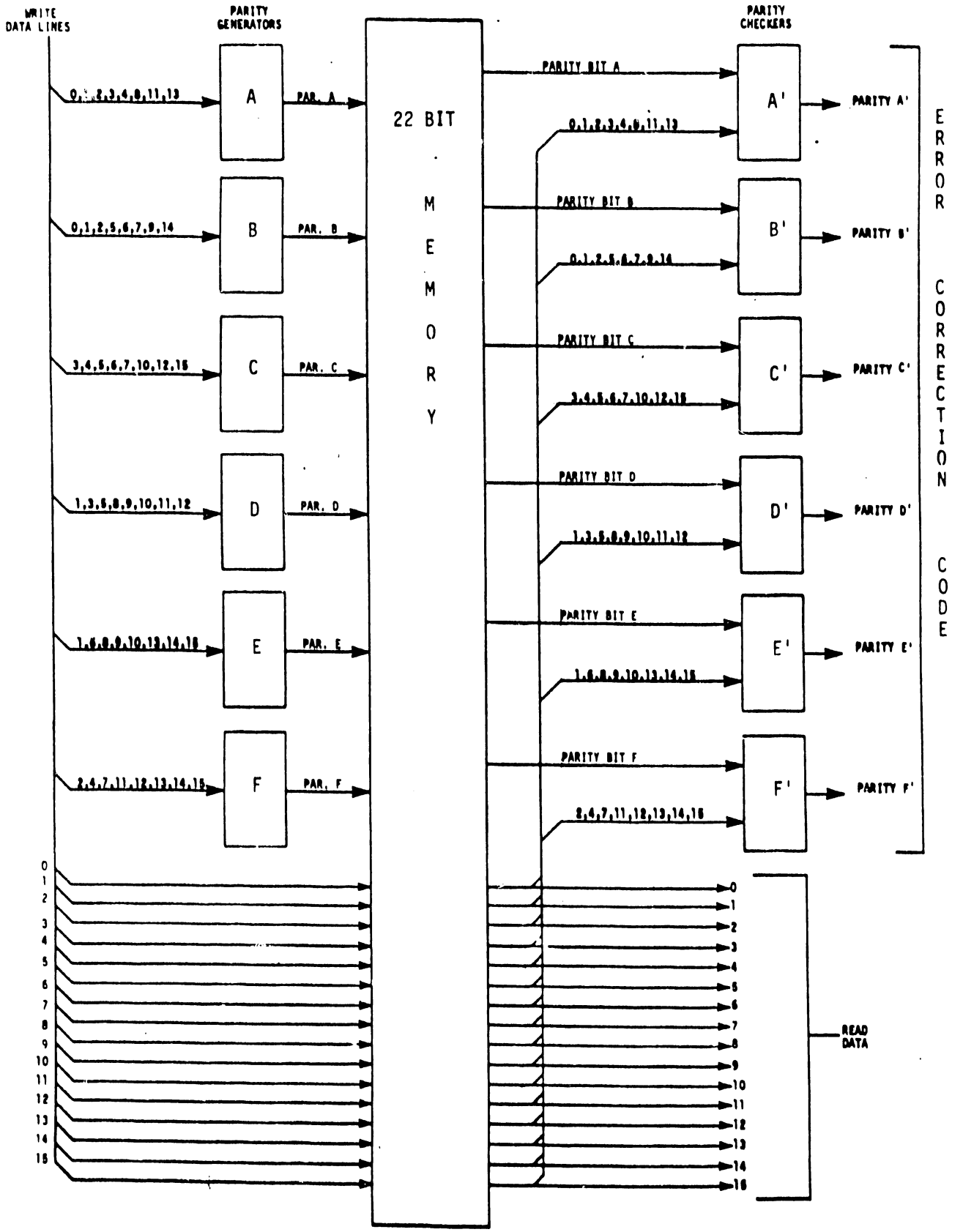


FIGURE 3-7

Before a halfword is written into memory, a unique combination of eight data bits in that 16-bit data pattern is presented to a write-parity generator. There are six such write-parity generators (named 'A' through 'F'); therefore, odd-parity is similarly generated for 5 other unique combinations of the 16 data bits prior to a RAM write. The resulting six parity bits, plus the initial data halfword are written into RAM (22 bits total).

When either the CP or an IOP calls for that same halfword, all 22 bits are read. The 16 bits of RAM data (of the 22 bits read) are latched into a register. Those same 16 bits are presented in unique combinations of 8 data lines plus one parity bit to six 9-bit RAM even/odd parity-check chips.

NOTE that the same data bit combination presented to write-parity generator 'A' (for instance) is also presented to a corresponding 9-bit parity checker 'A '. Also, each RAM write-parity bit generated (parity bit 'A', for instance) is presented to that same even/odd parity checker ('A '). Similar circuit operation occurs for 'B' thru 'F' and 'B ' through 'F '.

An actual ECC from the outputs of the six even/odd parity checkers ('A ' through 'F ') is also latched, then decoded into one of 16 error-correcting bits. One error correction bit exists for each data line. If any data bit is bad when read from RAM, when that data bit and its corresponding error-correcting bit are EXORed, the data bit is inverted. All of the above actions occur prior to loading either the CP MDR or an IOP MDR; i.e., all system processors receive corrected data only.

At this point, further explanation of ECC operational theory can be illustrated using a chart:

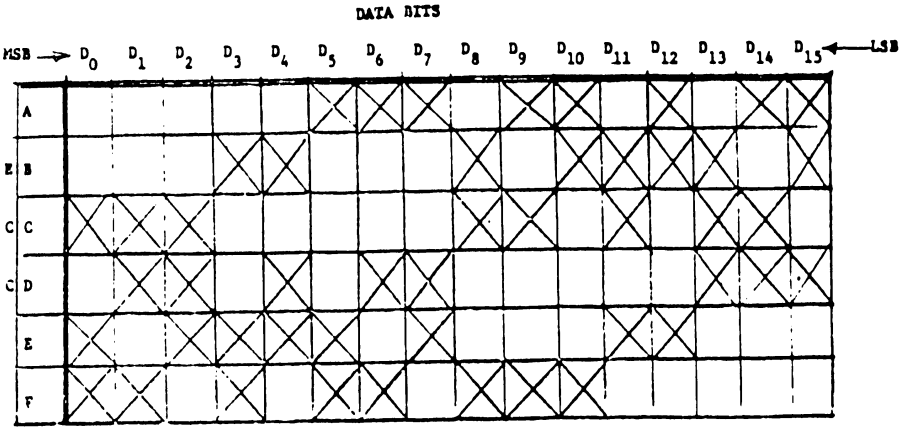


FIGURE 3-8 ECC CHART

In each vertical column, note that any given data bit is associated with exactly three ECC bits. Each of the 16 data bits has its own, unique three-bit ECC. Note, however, that the actual decoding of ECC hardware requires six bits ('A ' through 'F ').

An 'X' in any box of the chart indicates an unused chart position (corresponding, in circuitry, to a data bit line not connected to any given even/odd parity-check chip).

Error correction can be demonstrated by example:

EXAMPLE: Generate the ECC for, and correct a one-bit error in the hexadecimal number CA26 (1100 1010 0010 0110 in binary) recalled from RAM.

Let us assume that the Hex CA26 previously written in RAM is read as Hex CAA6. The 16 bits comprising the binary equivalent of Hex CAA6 are presented to the even/odd parity checkers. Parity checkers A, D, and E indicate even parity. The error code compare logic identifies ECC A B C D E F (combination ADE in the ECC chart) as belonging to data bit 8. Bit 8 is complemented and Hex CAA6 becomes Hex CA26.

DATA BITS

	MSB	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9	D_{10}	D_{11}	D_{12}	D_{13}	D_{14}	D_{15}	LSB
	A	1	1	0	0	1	X	X	X	1	X	X	0	X	1	X	X	1*
E	B	1	1	0	X	X	0	1	0	X	0	X	X	X	X	1	X	1
C	C	X	X	X	0	1	0	1	0	X	X	1	X	0	X	X	0	0
C	D	1	X	X	0	X	0	X	X	1	0	1	0	0	X	X	X	1*
	E	X	1	X	X	X	X	1	X	1	0	1	X	X	1	1	0	0*
	F	X	X	0	X	1	X	X	0	X	X	X	0	0	1	1	0	0
		1	1	0	0	1	0	1	0	1	0	1	0	0	1	1	0	

BAD BIT

*PARITY ERROR

FIGURE 3-9

MAIN MEMORY READ LOGIC

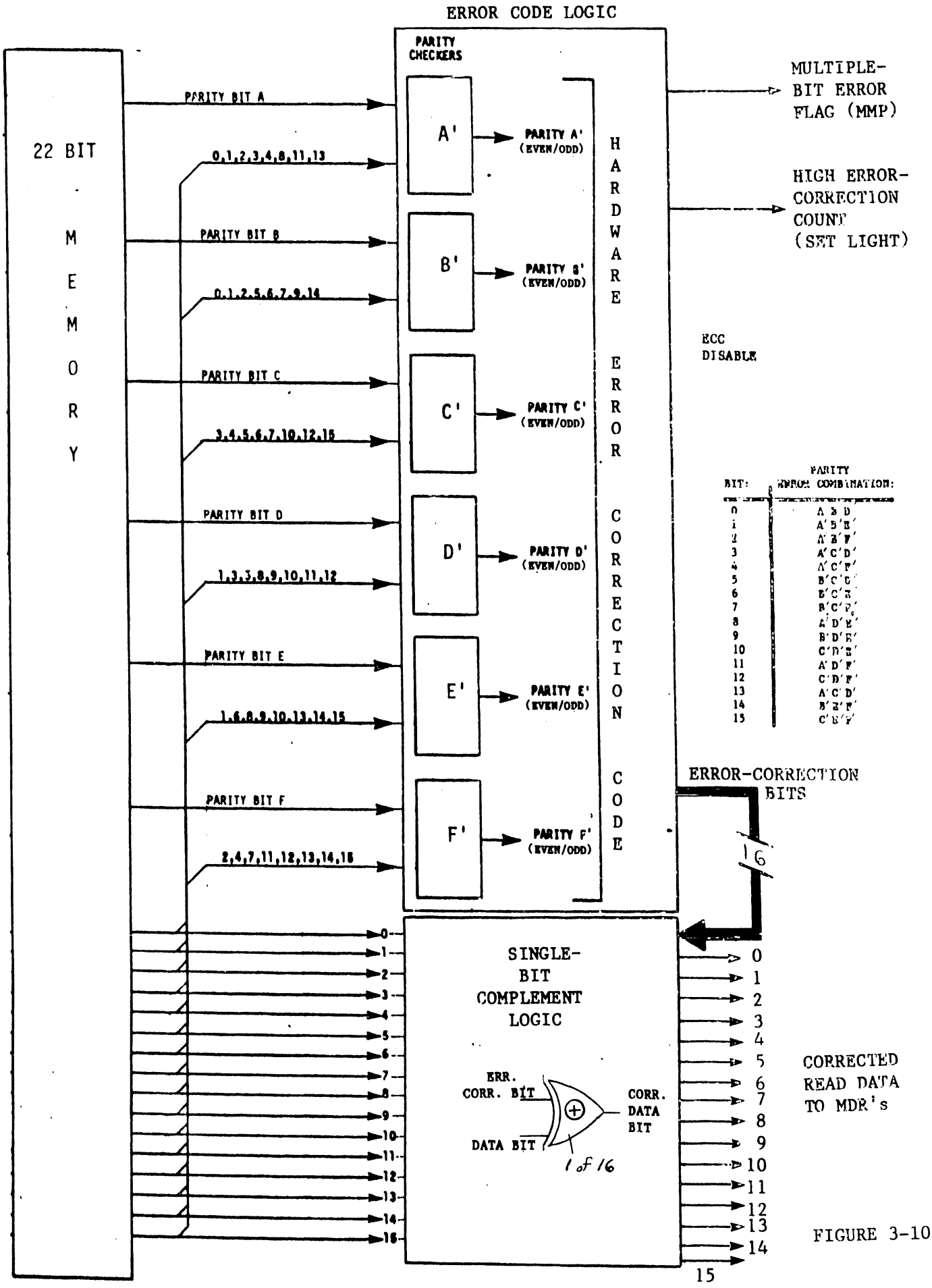


FIGURE 3-10

If more than one bit has been changed, the ECC will not match any of the sixteen 3-bit patterns shown in the ECC table. A multiple-bit error flag signal, MMP, generates an interrupt to the Central Processor to identify a major memory fault.

One section of the 2200VS ECC logic not shown in this discussion contains a counter which keeps track of the number of single bit errors that have occurred. When that counter exceeds 65,536 counts, a light on the processor panel is lit. This light informs the system user that a Main Memory problem exists in the system.

Whatever the nature of the single-bit failure, the bit is wrong and must be inverted. It is important to stress that the error is not corrected in memory. It is corrected only on the memory output lines. A correction in memory would require an additional memory cycle and an attempted correction would probably fail for the same reason the data was incorrect the first time.

DIRECT MEMORY ACCESS

Direct Memory Access (DMA) is available to each of the 8 I/O Processors (IOP) and the CP on a priority basis. Priority encoding logic on the 7303 card gives highest priority to the IOP in slot number one. The remaining IOP slots are from 2 to 8, in order of priority. The CP has the lowest priority due to its ability to steal memory cycles.

The disk IOP always goes in slot number one because of the high data transfer rate of the disk. The IOP gets serial data directly from disk and converts to parallel data. The IOP has only a small buffer. If memory cycles were not instantly available, the IOP buffer would run out of room (disk 'overrun'). If the IOP buffer gets overrun, a retry must be performed on the next revolution of the disk. Retries slow down system operation.

There are eight request lines and nine grant lines on the 7103. An IOP (or the CP) raises its request line when it wants a memory cycle and waits for its grant line to go true. The CP has no request line, as such; instead, the MMB controller looks at the MOP field of the CP microinstruction to see if a memory operation is called for.

When a processor is granted a memory cycle, address and read/write control signals are sent via the Main Memory Bus (MMB). Data is transferred on the MMB's memory data lines.

During a refresh cycle, all priority encoding is disabled. Thus the refresh cycle has, in effect, top priority. The same row address is refreshed at the same time on every memory chip on every board. In a 512 K system, for instance, 1,308 RAM chips are all having one row refreshed at the same time.

3.5 DISK STORAGE PHYSICAL DESCRIPTION

Disk volumes on the Wang 2200VS system are divided into logical 256-byte sectors, numbered in ascending order, from zero. Actual disk sectorization is implemented in 256 or 2,048-byte sectors, depending on device type. Although actual sectors may be addressed by certain I/O commands, the Operating System I/O and Paging routines always address 2,048-byte 'blocks' of disk storage.

'Files' on such a volume are recorded in one or more contiguous areas called 'extents'. Each 'extent' spans one or more consecutively numbered blocks (pages). The presence of a file on a volume is indicated in a 'volume table of contents' (VTOC), which can be located through the 'volume label'. 'Extents', 'VTOC', 'volume label', and other terms concerning disk storage are discussed in subsequent text.

3.5.1 VOLUME LABEL

The volume label occupies sector 1 (the second sector) of any disk volume. It contains the name of the volume (the volume serial number), the location (extent descriptions) of the volume's table of contents, and other descriptive information defining the size and physical organization of the volume.

3.5.2 VOLUME TABLE OF CONTENTS

The volume table of contents on a disk volume has blocks of four types:

1. Available Space Blocks.
2. High-level Index Blocks.
3. Low-level Index Blocks.
4. File Descriptor Blocks.

The first block of the volume table of contents is an 'available space block' (FDAV). Any search for available space on the volume (for a newly-allocated file or an additional extent) begins by searching this block, followed by its chained blocks, for a sufficiently large area of space.

The second block of the volume table of contents is a chain of 'top-level index blocks' (FDX1). Contained in each FDX1 is a series of library names and a pointer to the 'low-level index block' (FDX2; the third block of the VTOC). Each FDX1 points to one or more chained FDX2s. Each FDX2 contains a file name and a pointer to one or more chained 'file descriptor' blocks (FDR1); the fourth block of the VTOC. Each FDR1 contains a library and file name, plus pointers to the initial 'extents' occupied by a particular file.

When a file is initially allocated space, an attempt is made to acquire a single extent of sufficient size on the specified volume. If such an extent is not available, up to 3 extents may be allocated initially.

When a file is enlarged, and thereby exceeds the capacity of extents previously allocated for it, the system allocates an additional extent. This may require that an additional FDR be allocated to contain the additional extent information. FDRs describing additional areas occupied by a file are referred to as FDR2's. A file may encompass a maximum of 13 extents (described in one FDR1 record and one FDR2 record).

3.5.3 EXTENTS

Each block on a volume, with the exception of the first block and blocks containing the volume table of contents, is part of an 'extent' (defined contiguous area) of either free space or file space. Extents of free space are recorded in available space records of the volume table of contents. Extents of file space are recorded in 'File Descriptor Records' (FDRs) in the volume table of contents, where each FDR is associated with a particular file.

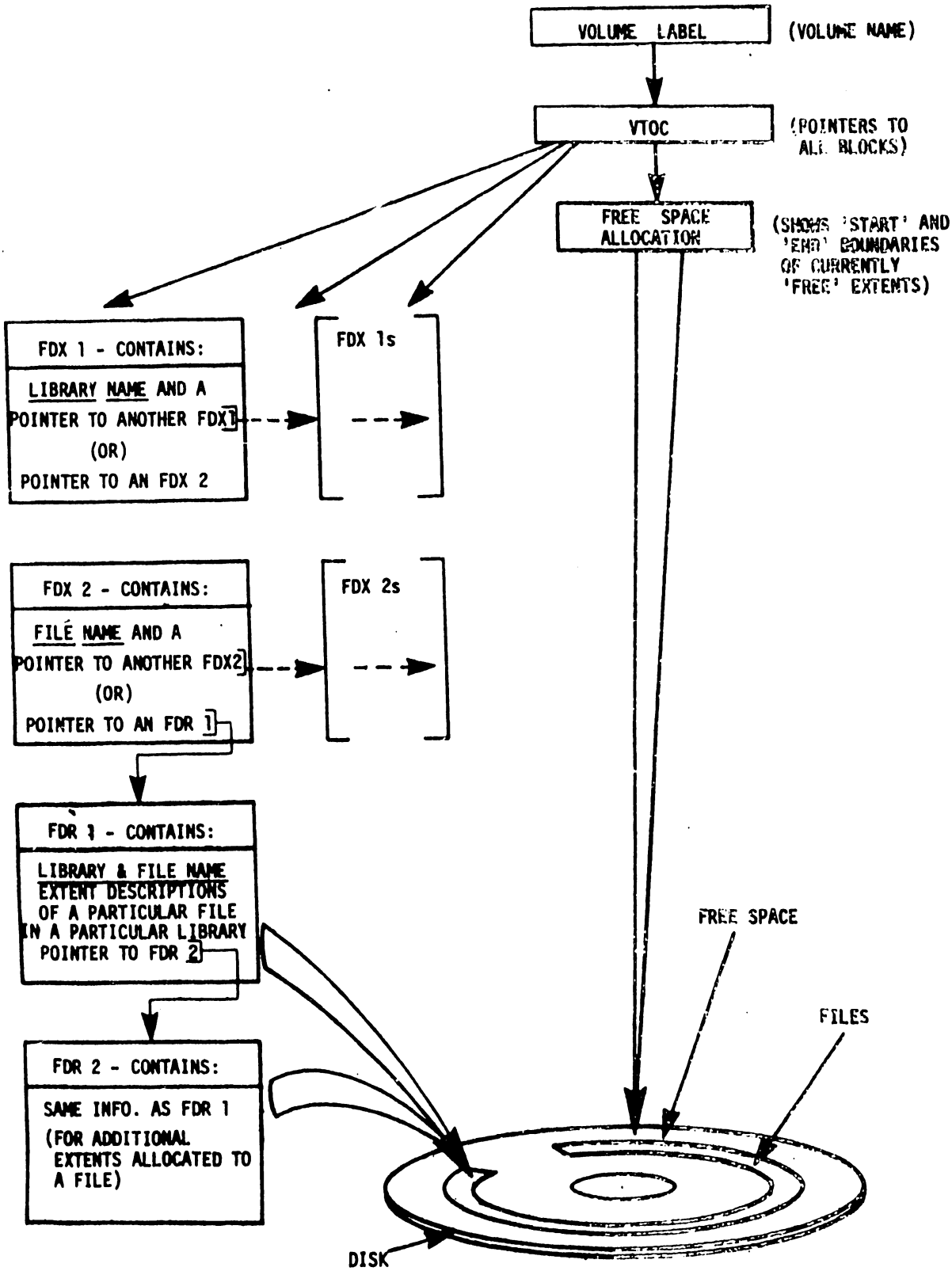


FIGURE 3-11

DISK STORAGE

SECTION 4

WORKSTATION CHARACTERISTICS

SECTION 4

WORKSTATION CHARACTERISTICS

4.1 INTRODUCTION

The workstation for the Wang 2200VS has two main parts, the display screen (the 'CRT') and the keyboard. This device is designed both to simplify the operator's job and to reduce processing time required by the CP to handle workstation I/O.

4.2 THE CRT

4.2.1 THE SCREEN AND CURSOR

The display screen can display up to 24 rows of 80 characters each. Each character position can display any of the characters defined in the workstation display character set. A 'cursor' indicates where on screen the next character entered from the keyboard will be stored. The cursor is displayed on screen only when data can be keyed by the operator. If the cursor is not displayed, the keyboard is locked. This has no effect on the display or the computer interface with the workstation, but does directly effect data entry from the keyboard. Each position of the screen is referenced by its row (1-24) and column (1-80) numbers. The first position of the screen (upper left corner) is called row one, column one. The columns are numbered from left to right and the rows from top to bottom. Position two, for instance, is the second character (col. #2) from the left on the first line (row #1).

4.2.2 SCREEN FORMATTING

An important feature of the workstation is its ability to set off any portion of the display screen into special 'fields' of character groups. These 'fields' control operation of both keyboard entry and CP/workstation I/O communications. A field is defined as the group of characters that exist between one 'field attribute' character and the next.

4.2.3 FIELDS

Fields can be of any length from zero to 80 characters:

Selected field: This field has been modified by user data entry at the workstation.

Underscore field: The characters in this field are underscored when displayed on the screen.

Intensified display field: The characters in this field are displayed at a higher intensity than those in a low intensity display field.

Low intensity display field: The characters in this field are displayed at normal intensity.

Blinking display field: The characters in this field are displayed alternately in the intensified display/display mode. The display will change intensity at a fixed rate of about 3 times a second.

Nondisplay field: The characters in this field are not displayed. This field will look as if it contained all blanks.

Unprotected field: Any or all of the positions of this field can be changed by the operator. (Also called the modifiable field.)

Protected field: No position of this field can be modified by the operator.

Alphanumeric field: Allows keyin of any character on the keyboard.

Uppercase shift field: Letters are displayed and stored only as uppercase. This is without regard to whether shift or lock are depressed. All other keys respond to the shift and lock keys as they normally would.

Numeric only field: Only the characters 0-9, decimal point (.), or minus (-) may be entered into this field or the keystroke is ignored and the alarm sounds.

Reserved: This is not a valid field at this time. It is intended for addition of later options. Its use may yield unpredictable results.

All characters in a field have the same attributes, as defined by the field attribute character preceding the field.

4.2.4 FIELD ATTRIBUTE CHARACTERS

An 'assumed' field attribute character exists just before the first character in each row and just after the last character in each row. Assumed field attribute characters do not take up space on the screen. They have a default value of low intensity, protected, and alpha-numeric upper/lower case. All other field attribute characters display as a blank, no matter what their value or in-screen position.

The possible attributes are defined in the following table.

Bit	Field Description
0	Must be set to one
1	Selected-Field Tag for READ ALTERED and WRITE SELECTED
2	One for underscore

Table 4-1

DISPLAYABLE CHARACTERS

NOTE: b_7 always equals zero*	b_6 →	0	0	0	0	1	1	1	1
	b_5 →	0	0	1	1	0	0	1	1
	b_4 →	0	1	0	1	0	1	0	1

b_3	b_2	b_1	b_0	High-Order Digit → / Low-Order Digit ↓	0	1	2	3	4	5	6	7
0	0	0	0	0		â	SP	0	@	P	`	p
0	0	0	1	1	♠	ê	!	1	A	Q	a	q
0	0	1	0	2	▶	î	"	2	B	R	b	r
0	0	1	1	3	◀	ô	#	3	C	S	c	s
0	1	0	0	4	→	û	$	4	D	T	d	t
0	1	0	1	5	←	ä	%	5	E	U	e	u
0	1	1	0	6	|	ë	&	6	F	V	f	v
0	1	1	1	7	••	ï	'	7	G	W	g	w
1	0	0	0	8	↗	ö	(	8	H	X	h	x
1	0	0	1	9	↘	ü	)	9	I	Y	i	y
1	0	1	0	A	^	à	*	:	J	Z	j	z
1	0	1	1	B	■	è	+	;	K	[	k	ø
1	1	0	0	C	!!	ù	,	<	L	\	l	£
1	1	0	1	D	↕	Ä	-	=	M	]	m	é
1	1	1	0	E	ß	Ö	.	>	N	↑	n	ç
1	1	1	1	F	¶	Ü	/	?	O	←	o	¢

*Bit combinations 10000000 through 11111111 are field attribute characters.

3-4 Display control
- 00 Intensified display
- 01 Low intensity display
- 10 Blinking display
- 11 Nondisplay

5 Protect bit
- 0 Modifiable field
- 1 Protected field

6-7 Valid keyable data specification
- 00 Alpha-numeric upper and lower case
- 01 Alpha-numeric upper case shift
- 10 Numeric only
- 11 Reserved

4.2.5 TABS

There are ten (10) program settable tabs. These can be set to any column of the workstation's screen (1-80). They do not take a screen location and are not displayed. They allow forward tabbing operations to stop at locations within modifiable fields. A tab position is specified by column number, and affects that column of every row in which the specified column is modifiable. Tabs have no effect within protected fields or during back tab operations. When the workstation is powered on, all of the tabs are cleared.

4.2.6 AUDIO INDICATORS

Audible alarm: A short tone is sounded whenever an illegal keying operation is attempted. This can be caused by the operator attempting to enter data into a protected field, trying to move the cursor past the end of the screen with a field sensitive key or trying to enter data when the keyboard is locked.

Mechanical clicker: This is a small relay attached to the keyboard. It clicks each time a key is struck.

4.3 THE KEYBOARD

4.3.1 CURSOR POSITIONING KEYS

A. Nonfield sensitive

This set of keys will position the cursor, but are not affected in any way by fields and field attribute characters. They can position the cursor to any location of the screen. There are four keys in this group. These are:

(up arrow) - This key will position the cursor in the same column but up one row. If the cursor started in the first row, it will be positioned in the same column, but in the last row.

(down arrow) - This key will position the cursor in the same column but in the next row. If the cursor started in the bottom row of the screen, it will be positioned in the same column but on the first row of the screen.

(left arrow) - This key will move the cursor one position left in a row. If the cursor was at the start of a row, it will move it to last position in the preceding line. If the cursor is in the first location of the screen, it will be positioned to the last position of the screen.

(right arrow)- This key will move the cursor one position right in a row. If the cursor was at the end of a row, it will be positioned to the first position of the next row. If it was at the last position of the screen, it will be positioned to the first position of the screen.

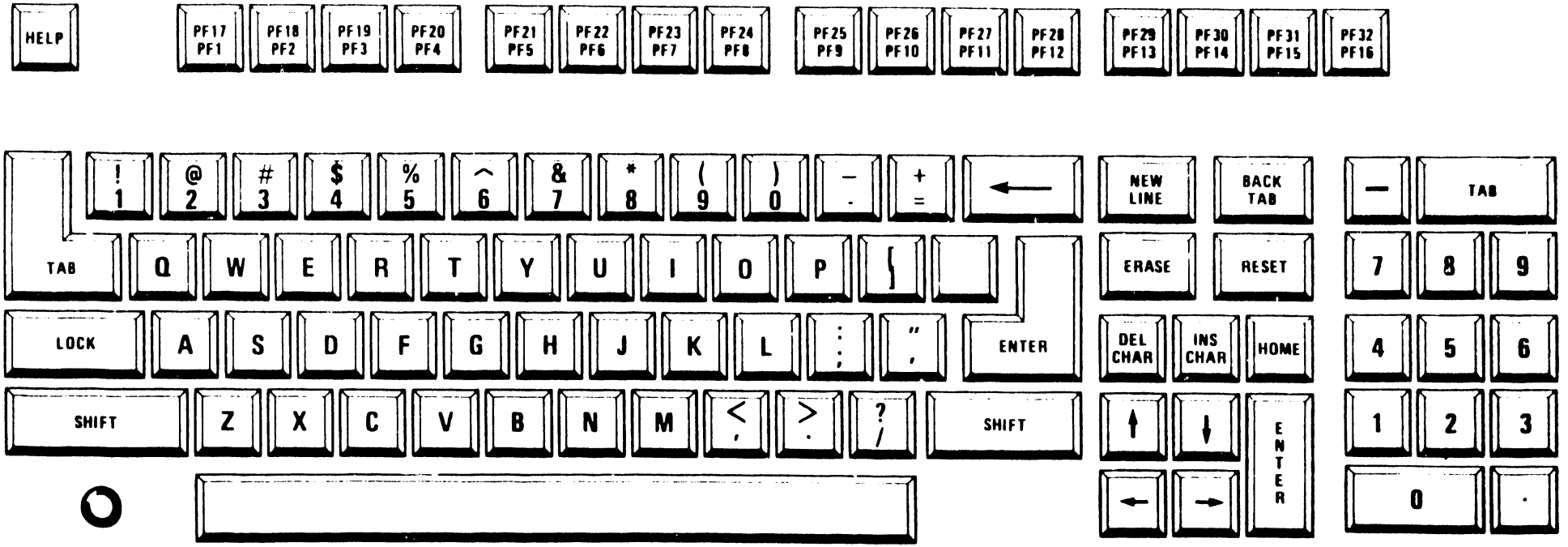


FIGURE 4-1
THE WORKSTATION KEYBOARD

NOTE:
Space Bar, as well as ↑, ↓, ←, and → keys provide Auto Repeat Feature

B. Field sensitive cursor positioning

The following keys will normally move the cursor two or more positions with one depression of a key. These keys are used to position to the start of a field or a new line. The keys can be used to simplify data entry. They will position the cursor to a modifiable position.

These keys are sensitive to modifiable positions. None of these keys modify any position. The four keys of this set are:

(TAB) - This key will position the cursor to the next tabable position. A tabable position is either the first character of a modifiable field or a tab column within a modifiable field, or the first character of a protected numeric-only field. If there is no next tabable position, the alarm will sound and the cursor will not move.

(BACK-TAB) - This key will position the cursor to the closest first byte of a modifiable field preceding the current curcor location. If the cursor is in a modifiable field and in other than the first location, the cursor will be positioned to the start of that field. If there is no preceding modifiable location, the alarm will sound and the cursor will not move.

(NEW LINE) - This key will advance to the first position of the next line. It then will position the cursor to the first position that is modifiable at or following the start of the line. This key may cause the cursor to be positioned several lines from the original position. If at or following the start of the next line there is not a modifiable location, the alarm will sound and the cursor will not move.

(HOME) - The HOME key will position the cursor to the first modifiable location on the screen. If there is no modifiable location on the screen, the alarm will sound and the cursor will not move.

4.3.2 DATA ENTRY KEYS

None of the keys discussed to this point change the contents of any screen display position. The data entry keys' sole function is to enter data into positions of the screen. For all of these keys the cursor must be in a modifiable field. If the cursor is not in a modifiable field, the key stroke is not honored, and the alarm will sound. The keys in this group are:

The normal character keys: These include the letters, numbers and special characters. These keys will enter characters the same as a typewriter (with the use of LOCK and SHIFT). If any of the characters other than (0-9) (-) (.) are struck in a numeric attribute field, the same action as for a protected field is taken. If the field is an upper case character attribute field, lower case letters will be interpreted as upper case letters.

When the cursor is in the last position of a field and one of these keys is struck, the character will be entered into the location and the cursor will be positioned to the next modifiable location. This may involve the skipping of the field attribute character or skipping of several lines.

If the cursor is in currently at the last modifiable location on the screen, the keystroke will be honored, the alarm will sound, and the cursor will not be moved.

ERASE - This key will set the cursor location and all following locations of the field it is in to blank characters. Any locations preceding the cursor will not be changed. The cursor does not move.

INS (insert) - Striking this key will place a blank at the cursor location and shift all of the characters in the current field starting with the one at the cursor location up to but not including the last character in the field, right one character. The last character in the field will be lost. If the last character in the field is not a blank or a pseudo blank, no screen location is changed, the alarm sonds and the cursor does not move. Pseudo blanks are the characters hex 0B and hex 05 in a modifiable field.

DEL (delete) - This key deletes the character at the cursor location and moves the characters in the field after the cursor location to a screen address location one less. The last character moved will be the last character in the field. The last character moved will be replaced with a blank. If the cursor is not in a modifiable field, the key will not be honored and the alarm will sound. This key is reciprocal in action to the INS key.

4.3.3 SPECIAL KEYS

There are four keys that do not seem to fit other categories. These are:

SHIFT - This key basically has the same effect as SHIFT does on a typewriter. On keys with an upper and lower character on the key face it will always select which character is entered. It will have absolutely no effect on entering letters in an uppercase attribute field. These will be entered as uppercase whether this key is depressed or not. Striking this key when the SHIFT light is on will cause the SHIFT light to be turned off and will unSHIFT the keyboard.

LOCK - Striking this key will turn on the SHIFT light. The workstation then behaves as if the SHIFT key was being held continuously depressed. Striking the SHIFT key will turn off the SHIFT light, returning the keyboard to an unSHIFTed state. Normal power on sequence will cause the device to be in an unLOCKed state.

4.3.4 KEYS COMMUNICATING WITH THE COMPUTER

This set of keys cause an interrupt to be presented to the computer. All keys except the HELP key are locked after striking any computer communication key, and the alarm will sound . The cursor is removed from the screen during execution of these operations. The keys are:

HELP - This key acts the same as the ENTER key. It is intended for Operating System use. The shift key does not affect its action. The only time the key cannot be honored is when an unsolicited interruption is pending for the same device. At any other time the key will be honored. This includes both the times when the keyboard is locked for any of the data entry keys and during a READ or WRITE to the workstation. A HELP key struck while a READ or WRITE is in progress will result in a separate attention interruption occurring after the READ or WRITE completion interruption.

PF1-PF32 (program function) - These keys act the same as the ENTER key except for the AID byte value telling which key was struck. There are 16 keys; the lower case values for these keys represent PF1-PF16, and the shifted (upper case) values PF17-PF32.

ENTER - This key is the normal way to terminate data entry and request the program to process the data. The shift key does not affect the action of the ENTER key. The ENTER key is not honored when the keyboard is locked for data entry keys.

APPENDIX A

GLOSSARY

APPENDIX A
GLOSSARY

ABEND - Abnormal end of job.

ABEND dump - A display of register contents, storage contents, and any pertinent information that the system can provide at the point where a job cannot be allowed to continue execution because of the occurrence of an exceptional condition.

absolute address - See explicit address.

absolute assembler - An assembler that calculates absolute memory addresses for each source program instruction and data item.

absolute expression - An expression whose value is not affected by program relocation. It can represent an absolute address.

absolute loader - A loader routine with error-checking capability that determines if the program it loads is a correct sequence of bytes for a previously written valid object program. Programs and data are recorded in a strict format after other systems software has calculated all storage addresses.

access method - A technique for moving data between main storage and input/output devices.

actual address - Same as absolute address.

address - The value by which a programmer references a storage location.

address constant - A constant requested by the programmer and defined by the assembler to contain a complete storage address.

address space - The complete range of addresses that is available to a programmer.

address translation - (1) The process of changing the address of an item of data or an instruction to the address in main storage where it is to be loaded or relocated. (2) In virtual storage systems, the process of changing the address of an item of data or an instruction from its virtual storage address to its real storage address.

algebraic shift - The type of shift in which all bits do not participate equally. The left-most bit is treated as the sign.

algorithm - A preset procedure designed to create a step by step solution to a problem.

alignment - See boundary alignment.

allocate - To assign a resource for use in performing a specific task.

alphabetic character - The characters A through Z and @, #, and $.

alphameric characters - The characters A through Z, digits 0 through 9, and @, #, and $.

ALU - Arithmetic and logic unit. The portion of the hardware that handles arithmetic operations and logical operations such as comparisons.

American National Standards Institute - An organization sponsored by the Business Equipment Manufacturers Association (BEMA) for the purpose of establishing voluntary industry standards. Abbreviated ANSI.

analog - A computer that performs mathematical operations on data received that is converted into electrical impulses. Receives its data in a continuous stream.

ANSI - Abbreviation for American National Standards Institute.

argument - That portion of an element in a search reference table that is checked for a match to the argument being searched for. It is the key to each element.

arithmetic and logic unit - See ALU.

ASA - American Standards Association. A former name for the American National Standards Institute.

ASA control characters - Characters placed in the first byte of an output record destined for the printer. It is not printed itself, but is used to control the spacing of the lines; single spacing, double spacing, or eject.

assemble - The translation of a source module in the assembler's symbolic language to an object module in machine language.

assembler - A program that performs the translation of an assembler source module to a machine language object module.

assembler language - A source language that includes symbolic machine language statements in which there is a one-to-one correspondence with the instruction formats and data formats of the computer.

assembler listing - See listing.

assembly - The output of an assembler.

assembly, conditional - Typically used to facilitate tailoring of programs to varying system configurations by including only those code segments required to handle existing devices.

assembly-output language - An optional symbolic assembly-language listing of the object-code output from a high-level language compiler. Can be quite helpful as a debugging tool because it shows exact machine code in a readable format.

assembly time - The time at which an assembler translates the symbolic language statements into their object code form (machine instructions).

asterisk - Refers to the current value of the location counter when used where a relocatable value is expected. A special character (*) that denotes a comment statement (full card comment) when it appears in column one of a source statement.

A-type constant - See address constant.

automatic data processing - See data processing system.

auxiliary storage - Data storage other than main storage; for example, storage on magnetic tape or direct access device.

base address - The beginning address for resolving symbolic references to storage.

base register - A general purpose register that has been designated and contains the base address to be used in resolving symbolic references to storage locations.

batch processing - See stacked job processing.

batched job - A job that is grouped with other jobs as input to a computing system. Synonymous with stacked job.

benchmark - A test point for facilitating measurement of product performance. Typically, a program or set of programs run on several computers for purposes of comparing speed, throughput and ease of conversion.

binary-coded decimal character code - A set of 64 characters, each represented by six bits. See also extended binary-coded decimal interchange code.

binary number system - A number system containing 2 symbols; 0 and 1. Base 2.

bind - To fix or assign a value to a symbol, parameter, or variable.

binding time - The point in time when a value is fixed or assigned to a symbol, parameter, or variable.

bit - A term generally used to refer to a binary digit.

blank character - On input, a blank will be converted to the ASCII representation of a blank a hexadecimal 20.

block - See physical record.

blocking - Combining two or more logical records into one physical record or block.

blocking factor - The number of logical records combined into one physical record or block.

block length - The number of bytes in a physical record or block.

block size - Same as block length.

boundary - See boundary alignment.

boundary alignment - The position in main storage of a fixed-length field, such as a halfword or doubleword, on an integral boundary for that unit of information. A halfword boundary is a storage address that is evenly divisible by two and a doubleword boundary is a storage address that is evenly divisible by eight.

branch - An instruction that changes the sequence of instruction execution.

branch table - A table in which each element is a branch instruction.

branch target - The subject instruction of the branch instruction. The next instruction that will be executed if the branch is taken.

breakpoint - A specific place in a program or subroutine that facilitates debugging by requesting interruption for manual evaluation and/or modification before continuing execution. (See also set breakpoint.)

buffer - An area that data may be read into, while processing continues. Also the I/O area used by the data management routines.

bug - A problem in a program which prevents it from executing successfully. It can be a syntax error, an error at execution time, or an error in the logic of the solution to the problem.

byte - A sequence of eight adjacent binary digits that are operated upon as a unit and that constitute the smallest addressable unit in the computer's storage (BAU).

call - See subroutine call.

call by name - Passing the addresses of the parameters to a subroutine.

call by value - Passing the actual values of the parameters to a subroutine.

called routine - A subroutine which is called or receives control.

calling routine - A subroutine which calls or passes control to another routine.

calling sequence - The set up of parameters and actual branch which transfers control.

card field - One or more consecutive card columns assigned to data of a specific nature. For example, card columns 15-20 can be assigned to identification.

carriage-control character - See ASA control character.

central processing unit - That part of the computer system that keeps track of the next instruction to be executed, and interprets and executes all instructions. It can be abbreviated CPU.

chain - (1) Any series of linked items. (2) Referring to the sequential processing of successive program segments, each of which depends on the previous segment for its input.

chained list - A means of connecting a collection of data items when they are not in contiguous areas of storage. The connection is made through addresses kept in each item or block. See headers.

character - An 8-bit code represented in a byte, making 256 different bit combinations possible.

character expression - A character string enclosed by apostrophes. The enclosing apostrophes are not part of the value represented.

character set - A fixed group of graphic representations, called characters.

closed loop - A group of instructions that are repeated indefinitely. Same as infinite loop.

collating sequence - A logical sequence used to order items of data.

comments field - The fourth field of an assembler language statement. It follows the operand field preceded by a blank. It is not checked for syntax errors in the assembler's scan of the statement.

comment statement - A statement used to include information that may be helpful in running a job or reviewing a listing. It is noted to the assembler by the appearance of an asterisk in column one of the statement.

comparison - The examination of the relationship between two items of data. It is usually followed by a decision.

compiler - A computer program that translates high-level source code into machine-language code by selecting appropriate machine-language subroutines and performing the necessary linkage to generate a single object program.

concatenated data sets - A group of logically connected data sets that are treated as one data set for the duration of a job step.

condition-controlled loop - A loop in which the decision to stop execution is based on the occurrence of a particular condition.

conditional assembly - An assembler facility for altering at pre-assembly time the content and sequence of source statements that are to be assembled.

conditional assembly instruction - An assembler instruction that performs a conditional assembly operation. Conditional assembly instructions are processed at pre-assembly time.

conditional branch - A branch instruction in which a test for a particular condition is made and if the condition is met, the branch is taken.

conditional jump - Same as conditional branch.

condition code - A code that reflects the result of a previous arithmetic or logical operation.

constant - A fixed or invariable value or data item.

contiguous - Physically adjacent-e.g., consecutive bytes in storage. For example, the byte with the address 2 follows the byte with the address 1. Bytes addressed 1 and 2 are contiguous.

control - Part of data processing system that determines the order for performance of basic functions.

control character - See carriage-control character.

control routine - A routine (effectively part of the machine) that controls the loading and relocation of other routines, sometimes employing instructions not available to the user. (See also monitor.)

control routine, interrupt - A control routine that responds to interrupts. It stores information on the interrupted environment, evaluates the interrupt to determine appropriate reaction, and eventually returns control to the interrupted routine.

control section - That part of a program specified by the programmer to be a relocatable unit, all elements of which are to be loaded into adjoining storage locations. It is abbreviated CSECT.

corner cut - A corner removed from a card for orientation purposes.

count-controlled loop - A loop which is executed a finite number of times. A special case of a condition controlled loop, count reached being the condition.

counter - A location, storage or register, in which a programmer keeps a count of the number of times a particular event has occurred.

CPU or CP - Central processing unit.

cross assembler - An assembler used in one computer to generate object-code instructions for another computer. Frequently used in conjunction with a down-line load capability for remote control of an unattended microprocessor. (See also resident assembler.)

cross compiler - A compiler that runs on one computer system but generates machine code for another computer system. Typically it runs on a large computer and generates code for a microcomputer, speeding up software development.

cross-reference table - A table produced by the assembler from information encountered in the source module. It contains each symbol, attribute, statement numbers of where the symbol is defined and every statement in which the symbol appears in the operand.

CSECT - Abbreviation for control section.

DASD - See direct-access storage device.

data - Characters that are capable of having meaning assigned to them, by a programmer, for a particular purpose.

data base - A collection of data fundamental to an enterprise.

data conversion - The process of changing data from one form of representation to another.

data file - A collection of related data records organized in a specific manner. For example, a payroll file (one record for each employee, showing his rate of pay, deductions, etc.). See also data set.

data management - A major function of operating systems that involves organizing, cataloging, locating, storing, retrieving, and maintaining data.

data medium - See medium.

data organization - The arrangement of information in a data set; for example, sequential organization.

data processing - The handling of data to produce desired results.

data processing system - A network of machine components capable of accepting information, processing this information according to a plan (a program) and producing the desired results.

data protection - A safeguard that prevents the loss or destruction of data.

debugging statement - Logical extensions to programming languages or compiler options that facilitate detection of program errors at run time. Examples of debug aids include a printout of program identifier cross-reference; a printed trace of variable value changes and/or flow of execution logic from routine to routine; the ability to alter or insert statements at run time and selective execution capability.

decimal, binary coded (BCD) - A numbering system that represents each decimal digit by four binary digits, with each place value equal to 8^2, 4^2, 2^2, 1^2, reading from left to right.

decimal number system - A number system containing 10 symbols; 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9. Base 10.

decision - See branch.

decision table - A table showing conditions that can be present in a particular situation, and the resultant actions taken.

default value - The choice among exclusive alternatives made by the system when no explicit choice is specified by the user. For example, keyword parameters on a macro call.

define the file - See DTF.

delimeter - A character or location that groups or separates words or values in a line or statement. For instance, column 72 is the delimeter for an assembler language statement, and a comma is the delimeter that separates each operand in the statement.

delimeter statement - A job control statement used to mark the end of data (/*), or the end of the job (// in OS-based systems and /& in DOS-based systems).

demand paging - In virtual storage systems, transfer of a page from external page storage to real storage at the time it is needed for execution.

desk checking - Debugging a program at coding time. Involves use of flowchart, commenting code, undefined symbol check, and a one for one check of keypunching.

device independence - The ability to request I/O operations without regard for the characteristics of specific types of input/output devices. See also symbolic unit name or logical unit name.

device type - The general name for a kind of device; for example, 1403, 3330, or 3400. See also group name.

diagnostic error message - Error messages produced by the assembler following the listing of the source module, explaining problems it has noted as it scanned the instructions.

diagnostics - Error statements produced by executive routine that tell the programmer of a specific problem. (See also routine, diagnostic.)

digital computer - A computer that operates directly on the data it receives. The data is in discrete pieces rather than a continuous stream.

direct access storage device - A device in which the access time is effectively independent of the location of the data. Abbreviated DASD.

direct reference table - A table in which each element contains only a function portion. The key is used to directly reference a particular element rather than searching a series of arguments in each element.

displacement - Positive number which can be added to the contents of the base register to calculate an effective address.

documentation - Supporting information about a program, such as comments, flowcharts, and writeup.

double threaded - A type of chained list or queue that has a chain of addresses pointing back to the previous blocks as well as forward to the next blocks. See single threaded.

doubleword - A contiguous sequence of 64 bits or 8 bytes of storage. It is capable of being addressed as a unit by referencing its first byte which has an address that is evenly divisible by eight.

downtime - The period of time in which the system or a particular device is inoperative.

driver - A program or routine that controls either external devices or other programs.

DSECT - Abbreviation for Dummy Section. Also referred to as Dummy Control Section.

dummy control section - A control section that an assembler can use to format an area of storage without producing any object code. Abbreviated DSECT.

dump - A display of the contents of storage as well as register contents and other pertinent information.

dump, memory - A printout, generally in hexadecimal format, of the contents of all memory areas currently assigned to the program (includes both program and data areas). THis output then serves as a diagnostic tool to facilitate troubleshooting.

duplication factor - A value that indicates in a DC statement the number of times that the data specified immediately following it is to be generated.

dynamic address translation - In virtual storage systems, the change of a virtual storage address to a real storage address during the execution of an instruction. Also a hardware facility that performs the translation. Abbreviated DAT.

dynamic relocation - A type of relocation which fixes the time of binding to the latest possible point - when it is loaded. Not until a portion of code is needed at execution time is a relative address translated to a real storage address.

EBCDIC - Extended binary coded decimal interchange code.

edit - The process of inserting characters into an output field to create more legible reports.

effective address - An actual real storage address. A displacement added to the contents of a base register and an index register if one is present.

electronic data processing - Data processing using electronic equipment.

element - A discrete portion of a table that is referenced by its location in relation to the beginning of the table.

emulation - Techniques of software or microprogramming that permit one computer to execute the machine-language code of another computer. Typically used to minimize reprogramming during conversion from one system to another.

end-of-file - Condition reached when all the records have been read in a sequential input file. Abbreviated EOF.

end-of-file-mark - A code that signals that the last record of a file or data set has been read. Abbreivated EOF.

entry code - The code that handles standard linkage conventions as a routine first receives control, the storing of register contents, establishing addressability, and preparation of a new save area.

entry name - A name within a control section that defines an entry point and can be referred to by any control section.

entry symbol - An ordinary symbol that represents an entry name or control section name.

EOF - Abbreviation for End-of-file.

EQU - Abbreviation for equate.

equate - An assembler pseudo op that allows the assignment of a value to a symbol. The symbol can be either absolute or relocatable depending on the value assigned.

Error condition - The state that results from an attempt to execute instructions in a computer program that are invalid or that operate on invalid data.

ESD - External symbol dictionary.

establish addressability - The process of informing the assembler which register it can use as a base register and what value will be in that register. Also the filling of that register with the promised value at execution time.

E-time - See execution time.

even-odd coupled register pair - Two consecutive registers, the first having an even number and the second the next higher numbered register. For example, registers 4 and 5, or 8 and 9.

exception - See error condition.

excess sixty-four binary notation - A binary notation in which the characteristic component of a floating-point number is represented in storage.

execute - To carry out an instruction or perform a routine.

execution time - The time during which an instruction is decoded and performed. See also instruction time. Abbreviated E-time.

explicit address - An address in which the base register and displacement are coded in the instruction by the programmer rather than coding a symbol and letting the assembler substitute the base register and displacement.

explicit length - A length, in bytes, specified in the operand it refers to rather than letting the implied length of the symbol in that operand apply. Generally used in SS-type instructions.

expression - A term or arithmetic combination of terms representing a value.

extended binary coded decimal interchange code - A set of 256 characters, each represented by eight bits. Abbreviated EBCDIC. See also binary coded decimal character code.

extended mnemonic - Special mnemonic opcodes that make it easier for the programmer to specify branching instructions. The mnemonic used not only states that this is a branch instruction, but also the mask to be used to determine what conditions.

external page storage - In virtual storage sytems, the portion of auxiliary storage that is used to contain pages.

external page table - An extension of a page table that identifies the location on external page storage of each page in that table.

external reference - A reference to a symbol that is defined as an external name in another module. Also, a symbol that is not defined in the module that references it.

external storage - Same as auxiliary storage.

external symbol - A control section name, entry point name, or external reference that is defined or referred to in a particular module. An ordinary symbol that represents an external reference.

external symbol dictionary - Control information, associated with an object or load module, that identifies the external symbols in the module. Abbreviated ESD.

externally referencable symbol - See entry symbol.

EXTRN - External reference declarative.

fail soft - A method of system implementation that prevents irrecoverable loss of computer usage due to failure of any system resource. It provides for graceful degradation of service.

fetch - To locate and retrieve something from storage. For example, the next sequential instruction or a word of data for a register.

fetch protection - A storage protection feature that determines the right to access storage by matching a protection key associated with a fetch reference to storage.

field - A specific group of contiguous bytes in a record which are treated as a unit.

FIFO - A technique for handling a chained list on a first-in-first-out basis.

file - A collection of related records treated as a unit.

firmware - Software instructions committed to a read-only memory control block. Can increase a computer's instruction set by having the ROM code convert extended instructions into sets of actual machine instructions.

fixed-length data - Data of a specific length (two, four, or eight bytes) that reside on integral boundaries (halfword, fullword, and doubleword, respectively).

fixed-length record - A data set in which a logical record contains the same number of bytes as every other record in the data set.

floating-point arithmetic - An arithmetic technique in which the computer maintains decimal point location (as opposed to fixed-point arithmetic). (See also subroutine package, floating point.)

fixed-point binary number - Occupy fullwords and halfwords. In each case the first bit in the field is the sign (0 is positive, 1 is negative). A negative number is stored in two's complement form.

floating-point number system - A number system in which very large and very small numbers can be represented because the decimal point can be moved.

flowchart - A pictorial method of displaying the steps involved in the logic of a solution to a problem.

foreground/background programs - In a multiprogramming environment, those programs that require real-time response are high priority (foreground) tasks which utilize system resources on demand. Conversely, background tasks, typically batch processing jobs, execute only during idle times and must always yield to demands from foreground programs.

fragmentation - Inability to assign real storage locations because the available spaces, though many, are smaller than needed.

full-line comment - A source statement with an asterisk (*) in column one. It is not scanned by the Assembler and can be used to document the program.

fullword - See word.

function - That portion of an element in a search reference table that is referenced for information once the correct element has been found.

fwb - Abbreviation for fullword boundary.

garbage - Data to which no meaning has been assigned for this particular usage.

general purpose register - See register.

generate - To produce assembler language statements from the model statements of a macro definition when the definition is called by a macro instruction.

GET - To obtain a logical record from an input file.

group name - A generic name for a collection of I/O devices, for example, DISK or TAPE.

guard digit - Used in execution of short form add, subtract, and divide floating-point operations. One spare hexadecimal digit which serves as extra (seventh) digit to improve precision.

halfword - A contiguous sequence of 16 bits or two bytes, which are capable of being treated as a unit. The first byte of the halfword occupies a storage location whose address is evenly divisible by two.

hard copy - A printed copy of machine output in a visually readable form; for example, printed reports, listings, and documents.

hardware - The mechanical equipment necessary for a computing system.

header statement - The MACRO statement which indicates the beginning of a macro definition to the assembler.

header - Contains the address of the beginning of a chained list. A single-threaded list has one header and a double-threaded list has two headers (second header contains the address of the last element in the chain). Same as queue control words.

hexadecimal number system - A number system containing 16 symbols; 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, and F. Base 16.

hexadecimal shorthand - A means of referring to the contents of a byte as two hexadecimal digits rather than eight binary digits.

high-level language - A language that allows programmers to specify problem-solving procedures in a notation more familiar than the computer's machine code. Such programs must be fed into a compiler or interpreter for translation into machine executable code. Examples include Fortran, Cobol, Algol, Basic and APL.

high-order - Leftmost. For instance, bit 0 in a register.

hit - See match.

host computer - (1) The master or controlling computer in a multicomputer network. (2) A computer that prepares programs to be run on another computer system.

housekeeping - Operations or routines that do not contribute directly to the solution of the problem, but do contribute directly to the operation of the computer.

hwb - Abbreviation for halfword boundary.

immediate addressing - An addressing mode in which the instruction contains the operand value in the address field.

immediate data - One byte of data that appears in the instruction itself rather than the symbolic name of the one byte of data. The data is immediately available from the instruction.

implicit address - A symbolic reference to storage that must be converted into its explicit base-displacement form before it can be assembled into the object code of a machine instruction.

implied length - The length associated with a symbol. This length will be used on a variable length operand if a length is not explicitly specified in the operand.

Indexed addressing - A method of computing storage addresses by adding an index value to a previously determined base address to produce a new address.

indexing - A technique of address modification implementated by the use of general purpose registers referred to as index registers.

index register - A register whose contents are added to the address derived from a combination of a base address with a displacement or an implicit address converted to a base and displacement.

indirect addressing - A method of storage addressing in which an addressed location contains an address rather than data. Quite often, several levels of indirect addressing may occur before the sought-after data item is obtained.

infinite loop - Same as closed loop.

information - Data to which a meaning has been assigned for this particular usage.

initialization - Initial values are assigned outside the loop to all counters, conditions, and variables needed within the body of the loop.

input/output - A general term for the equipment used to communicate with a computer, commonly called I/O. Also the data involved in such a communication.

input stream - The sequence of job control statements and data submitted to an operating system on an input unit especially activated for this purpose by the operator. Synonymous with input job stream, and job input stream.

input stream data set - A data set that physically resides in the input stream.

instruction - A request to the computer to perform one of its basic functions.

instruction classes - The different formats of machine instructions used on the computer (RR, RX, RS, SI, and SS).

instruction address counter - A location within the CPU where the address of the beginning of the next instruction to be executed is kept.

instruction format - The allocation of bits or characters of a machine instruction to specific classes of instructions.

instruction repertoire - The list of mnemonic opcodes that an assembler recognizes as valid.

instruction time - The time during which an instruction is fetched from storage of a computer into an instruction register. Abbreviated I-time.

integral boundary - A location in main storage at which a fixed-length field, such as a halfword or doubleword, must be positioned. The address of an integral boundary is a multiple of the length of the field in bytes. See also boundary alignment.

internal sort - A sorting technique that creates sequences of records or keys or elements of a table. All the items that participate in the sort in storage as the sort is being accomplished.

interpreter - An executive routine that translates a program into machine code subroutines and immediately performs the resulting operations prior to the next translating function. This contrasts with compilers that translate complete programs into machine code for execution at a later time.

interrupt - The supervisor seizing control when an error condition has occurred or assistance is needed to provide I/O.

interruption - A break in the normal sequence of instruction execution. It causes an automatic transfer to a preset location (trap) where appropriate action is taken.

interrupt program, I/O - An efficient method of I/O handling that interrupts the processor whenever a peripheral device signals that it's ready for information transfer. The processor first stores the necessary information to enable it to return to the present operating mode, then jumps to the routine appropriate for exercising the requested transfer. Upon completion of the I/O transfer, the processor restores either the previously running task or another task, depending on priorities and available resources.

interrupt vector - Facilitates fast handling of external interrupts by having the hardware supply a value corresponding to the device causing the interrupt. This value then becomes an index into the interrupt vector that contains a pointer to the appropriate interrupt service routine.

I/O - See Input/Output.

iterate - To repeatedly execute a loop or series of steps, for example, a loop in a routine.

I-time - Instruction time.

job - A unit of work to the computer; consists of one or more job steps, and each step involves the execution of a program.

job step - A unit of work associated with one processing program or one cataloged procedure and related data. A job consists of one or more job steps.

jump - See branch

K - 1024 Bytes: used in referring to storage capacity.

keyword - One of the significant and informative words in a title or document that describes the content of that document. A symbol that identifies a parameter. A part of a command operand that consists of a specific character string (such as DSNAME=).

keyword parameter - A parameter that consists of a keyword, followed by one or more values. See also positional parameter.

label - An identification record for a tape or disk file. A name entry in an assembler language statement.

least significant - The digit with the smallest place value in the number. Rightmost.

left-justify - To align on the left hand side of the field.

length attribute - The length, in bytes, associated with a symbol.

length modifier - A subfield which can be specified in the operand of a DC or DS statement. For example: DS CL6. If used on a fixed length definition (F,H or D) the automatic alignment will be overridden.

LIFO - A technique for handling a chained list on a last-in-first-out basis. Often used for priority queues.

link field - The pointer field in data items in a chained list which connects the data items to each other.

linkage conventions - A set of rules for calling routine responsibilities when using subroutines.

linkage editor - A program which processes object modules preparing them for execution. It resolves cross references between separately assembled object modules.

linking loader - A relocatable loader that links various object modules into a single load module, resolving external references in the process. This lets users load their programs into any memory area.

listing - A printout, usually prepared by a language translator, that lists the source language statements and contents of a program.

literal - A literal represents data. It can be used in instruction operands to introduce data. It is a means to avoid defined constants and using the symbolic names of these constants in instructions. Literals will be assembled and are relocatable values; but it is the assembler that defines them in literal pools rather than the programmer.

literal pool - An area of storage into which the values of the literals specified in a source module are assembled.

load - To fetch a fullword from storage and place it in a register. Also to place a load module into real storage.

load module The output of the linkage editor; a program in a format suitable for loading into main storage for execution.

loader - A program that handles the transfer of information from off-line memory to on-line memory.

location counter - A value kept by the assembler to tell it which is the next byte available for allocation as it builds the object module. Its value is displayed to the left of each instruction in the source listing.

logical expression - A conditional assembly expression that is a combination of logical terms, logical operators, and paired parentheses.

logical record - A record from the standpoint of its content, function, and use rather than its physical attributes; that is, one that is defined in terms of the information it contains.

logical relation - A logical term in which two expressions are separated by a relational operator. The relational operators are EQ, GE, GT, LE, LT and NE.

logical shift - The type of shift in which all bits participate equally.

logic error - A case where a program seems to execute correctly but provides incorrect results.

look-up, table - A method of retrieving from a table of function values a specific function value corresponding to an argument.

loop - A programming technique which permits the reuse of a group of instructions a specified number of times or until a particular condition occurs.

loop body - The instructions which are reused.

loop control - The instructions in the loop which determine when the reuse of the instructions should be stopped.

loop counter - A counter used to prevent excessive looping.

low order - Rightmost. For instance, bit 31 in a register.

machine address - See absolute address.

machine language - The lowest-level language of a particular type of computer; a string of binary numbers (1s and 0s).

macro - See macro definition, macro instruction, and macro prototype statement.

macroassembler - An assembler that facilitates definition of macro's for frequently used code segments. Marco's simplify program coding; however, unlike subroutine calls, they generate in-line code for each reference.

macro body - The body is all statements that follow the prototype statement and precede the MEND statement in a macro definition.

macro call - An assembler language statement that causes the assembler to process a predefined set of statements called a macro definition. The statements normally produced from the macro definition replace the macro instruction in the program and they are identified by a plus (+) sign that precedes each statement. Same as macro instruction.

macro definition - A set of statements that defines the name of, format of, and conditions for generating a sequence of assembler language statements. Contains assembler language control and machine instructions.

macro expansion - The sequence of statements that result from a macro generation operation.

macro-generated instruction. A statement that results at pre-assembly time as the macro definition is being handled.

macro generation - An operation in which an assembler produces a sequence of assembler language statements by processing a macro definition called by a macro instruction. Macro generation takes place at preassembly time. Synonymous with macro expansion.

macro instruction - See macro call.

macro instruction operand - An operand that supplies a value to be assigned to the corresponding symoblic parameter of the macro definition called by the macro instruction.

macro library - A library of macro definitions used during macro expansion.

macro prototype statement - A statement used to give a name to a macro definition and to provide a model for the macro instruction, that is, to call the macro definition.

main storage - All program-addressable storage from which instructions may be executed and from which data can be loaded directly into registers. Contrast with auxiliary storage.

mask - A pattern of 4 or more bits used in the testing of alteration of another field. For example, the 4-bit mask in a Branch on Condition instruction or the 8-bit mask in the Test under Mask instruction.

masking - A technique for detecting the presence or absence of specific binary conditions by performing some logical operation (e.g., AND, OR, etc.) between a program variable and a preset mask. Also used for setting or resetting binary conditions in other variables.

match - An equal condition occurring when two items are compared.

media - The material on which data is recorded, such as magnetic tape, or paper.

microsecond - One-millionth of a second.

millisecond - One thousandth of a second.

mnemonic operation code - An easy to remember symbol that represents a machine opcode and helps a human understand the nature of the operation to be performed, the type of data used, and the format of the instruction performing the operation.

model statement - A statement in the body of a macro definition from which an assembler language statement can be generated at preassembly time. Values can be substituted at one or more points in a model statement; one or more identical or different statements can be generated from the same model statement under the control of a conditional assembly loop.

module - A discrete programming unit. For example, source module, object module, and load module.

monitor - A resident debug routine providing real-time breakpoint capabilities and a capability for examining and altering memory locations and system status variables. Also contains the necessary linkage points to allow user programs to call the monitor routines.

most significant - The digit with the largest place value in the number. Leftmost.

multiple-precision notation - A technique whereby two or more computer words represent a single numeric quantity.

multiprocessing system - A computing system employing two or more interconnected processing units to execute programs simultaneously.

multiprogramming system - A system that can process two or more programs concurrently by interleaving their execution.

multitasking - A method of achieving concurrency by separating a program or programs into two or more interrelated tasks that share code, buffers and files while running.

nanosecond - One-thousand-millionth of a second.

nesting - A programming technique involving the embedding of routines within other routines.

next sequential instruction - Physically the next instruction in storage, the next instruction in the program.

normalized form - A form in which a floating-point number is kept with a non-zero high-order digit. A number can be normalized prior to the execution of the operation (prenormalization), or after the execution (postnormalization), or both before and after.

NSI - Abbreviation for next sequential instruction.

null character string - A character string of length zero. A blank is not a null character string because it has a length of one.

null operand - The absence of an operand. Usually used when passing positional parameters in a macro call. For instance, NAME MAC FIRST,, THIRD where the second parameter is not being passed.

numeric punch - A punch in one of the ten rows numbered 0-9 on a standard punched card.

object module - The block of machine code created by the assembler when it translates the source module.

object program - A set of problem solving machine-language instructions obtained through the compilation or assembly of the related source program.

opcode - The most important part of an instruction. It informs the system what operation is to be performed and the type of data to be used.

open subroutine - A subroutine that lies wholly within the main routine. No special instructions are necessary to pass control to an open subroutine.

operand - The data to be used in an operation or the location of that data.

operating system - The software, programs, that aid in the operation of the mechanical devices, the hardware. They aid in I/O operations, error conditions handling, and resource management. For example, DOS, DOS/VS, or OS.

Operation code - See opcode.

operators - Symbols that represent mathematical or logical operations performed on one or more operands. For example, + , -, ÷, etc.

ordinary symbol - A symbol that represents an assembly-time value when used in the name or operand field of an instruction.

output - The results of the operation of a data processing system.

output stream - Diagnostic messages and other output data issued by an operating system or a processing program on output devices especially activated for this purpose by the system operator. Synonymous with job output stream, output job stream.

output stream data set - A data set that resides in the output stream.

overflow - A condition that sets the condition code and at times abends the program. Can occur when the result of an addition or subtraction requires more bits than are available. Also occurs when a left algebraic shift results in shifting into the sign bit position a different value than was there before the operation.

overlapping fields - Fields overlap when at least one byte is common to both fields

packed-decimal format - Each byte in this format contains two digits, except the right most byte in the field which contains the sign of the number in its rightmost 4 bits.

pad - To fill an area with a prescribed character. For example, unfilled area in a character constant is padded with blanks.

page - In virtual storage systems, a fixed-length block of instructions, data, or both, that can be transferred between real storage and external page storage; also the action of transferring instructions, data, or both, between real storage and external page storage.

page fault - In virtual storage systems a program interruption that occurs when a page that is marked "not in real storage" is referred to by an active page. Synonymous with missing page interruption and page translation exception.

page frame - In virtual storage systems, a block of real storage that can contain a page. Synonymous with frame.

page frame table - In virtual storage systems, a table that contains an entry for each frame. Each frame table entry describes how the frame is being used.

page-in - In virtual storage systems, the process of transferring a page from external page storage to real storage.

page-out - In virtual storage systems, the process of transferring a page from real storage to external page storage.

page table - In virtual storage systems, a table that indicates whether a page is in real storage and correlates virtual addresses with real storage addresses.

parameter - See symbolic parameter.

patch - (1) To alter or correct existing software. (2) Inserted code is often referred to as a "patch."

physical record - A record from the standpoint of the form in which it is stored, retrieved, and moved; that is, one that is defined in terms of physical quantities. A physical record may contain one or more logical records.

"play computer" - To manually excute the instructions of a program in sequence just as the computer would to ensure that the program does what it is expected to do. Substitute values for variables and follow logic through flowchart.

pointer - An address or other indication of location.

positional notation - A means of representing a number by specifying the value of each of its digits by a power of the base of the number raised to the power equal to the position of the digit being evaluated in the number.

positional operand - An operand in a marco instruction that assigns a value to the corresponding positional parameter declared in the prototype statement of the called marco definition.

positional parameter - A parameter that must appear in a specified location, relative to other parameters. See also keyword parameter.

postmortem dump - A dump taken when a program has done something in error which causes the supervisor to abend the program.

pre-assembly time - The time at which an assembler processes macro definitions and performs conditional assembly operations.

print control character - See carriage-control character.

printer - A device that writes output data from a system on paper or other media.

privileged instruction - An instruction that can be executed only when the central processing unit is in the supervisor state.

problem program - Any program that is executed when the central processing unit is not in the supervisor state. Any program that does not contain privileged instructions.

problem state - A state during which the CPU cannot execute privileged instructions. Contrast with supervisor state.

procedure - See cataloged procedure.

processing program - A general term for any program that is not a control program.

program - A series of instructions, in a language understood by a computer, which solve a problem. Also the process of creating the series.

program check interruption - An interruption caused by unusual conditions encountered in a program, such as incorrect operands.

program flowchart - See flowchart.

programmer - An individual capable of breaking a problem down into discrete steps and expressing those steps in one of the languages understood by the computer.

programming - A skill that requires that problem solutions be broken down into steps and expressed in a language understood by a computer.

programming language - A language understood by the computer and used by the programmer to say which instructions are to be executed and in what order.

program status word - A doubleword in storage used to control the order in which instructions are executed, and to hold and indicate the status of the computing system in relation to a particular program. Abbreviated PSW.

prototype statement - Same as macro prototype statement

pseudo op - An opcode for an instruction that gives information to the assembler. It does not represent a machine instruction.

PSW - Abbreviation for Program Status Word.

pushdown list - A list of items maintained in a Last-In-First-Out (LIFO) order, where each item is effectively "puched down" by the addition of a new item. (See also stack.)

push operation - Refers to the storing of operand(s) from a general register (s) into the most current top location in a pushdown memory stack. (See also Stack.)

queue - A waiting line or list formed by items in a system waiting for service; for example, messages to be printed. Also, to arrange in, or form, a queue.

queue control word - See header.

radix - A number that is used as the base of a number system.

real storage - In virtual storage systems, the storage of a System/370 computing system from which the central processing unit can directly obtain instructions and data, and to which it can directly return results.

real time - (1) As related to problem solving, a rate that provides solutions within the actual time that the problem must be solved. (2) Performing computations in the actual time needed to control a related physical process.

receiving field - The operand receiving the data that is participating in an operation.

record - A collection of related date items, or fields, which are treated as a unit.

record length - The number of bytes in a logical record.

register - Special areas of storage in the processor. There are 16 and each holds 32 bits or 4 bytes. They are used in certain operations.

relational operator - An operator that can be used in an arithmetic or character relation to indicate the comparison to be performed between the terms in the relation. The relational operators are EQ (equal to), GE (greater than or equal to). GT (greater than), LE (less than or equal to). LT (less than), and NE (not equal to).

relative address - An address specified as a relationship to a relocatable symbol. The symbol is followed by a plus (+) sign and a decimal number. For example, LOC + 6 is an address 6 bytes past the address LOC.

relocatable - The attribute of a set of code whose address constants can be modified to compensate for a change in origin.

relocatable assembler - A program that translates object code from an assembly-language source program with memory locations specified as displacements from a relative origin or as external references. This facilitates the running of programs in any memory area.

relocatable expression - An assembly-time expression whose value is affected by program relocation. A relocatable expression can represent a relocatable address.

relocatable term - A term whose value is affected by program relocation. Its value is assigned by the assembler.

relocation - The modification of address constants to compensate for a change in origin of a module, program, control section, or page.

resident assembler - An assembler that runs on the machine for which it generates code. Eliminates the need for another computer system or time-sharing service, as required by a cross-assembler.

resident compiler - A compiler that runs on the machine for which it generates code. Eliminates the need for another computer system, as required by a cross-compiler.

resource - Any facility of the computing system required by a job or task, including storage, input/output devices, the central processing unit, data sets, and control of processing programs.

return code - The return code is a flag (expressed as a decimal number) that is passed to a calling routine (such as the Command Processor or some other program/procedure) to indicate the results of program execution. By convention, a return code of zero indicates normal completion; non-zero return codes indicate error conditions. Typically, the higher the return code, the more severe the error. The return code must be defined in both the calling routine and the routine called, so that proper action may be taken by the calling routine for non-zero return codes. In the 2200VS, the return code is always stored in General Register zero.

right-justify - To align on the right-hand side of the field.

routine - See subroutine.

routine, diagnostic - Any program designed to aid in the detection of hardware or software malfunctions.

RR-type instruction - An instruction in which both operands are contained in registers.

RS-type instruction - An instruction in which the first (and third, if present) operand is a register and the second a storage address.

RX-type instruction - An instruction in which the first operand is a register and the second a storage address which may be indexed.

save area - An 18-word area used to store the calling routine's register contents when control is received by a subroutine.

scan - The assembler's examination of the syntax of a source statement from the left to right across the statement.

search - A systematic check for a particular value or values.

search argument - The value that is used to locate a match, if possible, with an argument in a search reference table.

search reference table - A table in which each element has two parts, an argument and a function.

search key - Same as search argument.

secondary storage - Same as auxiliary storage.

segment - In vitual storage systems, a contiguous area of virtual storage that is allocated to a job or system task.

segmentation - The process of dividing a program up into pieces to allow the possibility for part of the program to be in storage and execute without having to have the entire program in storage.

segment table - In virtual storage systems, a table used in dynamic address translation to control user access to virtual storage segments. Each entry indicates the length, location, and availability of a corresponding page table.

self-defining term - An absolute term whose value is implicit in the specification of the term itself.

semantics - The relationship between symbols and their meanings.

sequence symbol - A symbol used as a branching label for conditional assembly instructions. It consists of a period, followed by one to seven alphameric characters, the first of which must be alphabetic.

sequential access method - Storing and retrieving logical records in a continuous stream. To read the third record, the first and second records must be read first.

sequential data set - A data set whose records are organized on the basis of their successive physical positions, such as a magnetic tape file or a deck of punched cards.

sequential operation - The execution of instructions one after another in the sequence in which they appear in the program. See NSI.

set breakpoint - A user debug command that causes the setting of a breakpoint in a specified memory location. At program execution, encountering this breakpoint causes temporary program suspension and a transfer of control to the system debug routine. (See breakpoint and monitor.)

SET symbol - A variable symbol used to communicate values during conditional assembly processing.

severity code - A code assigned to an error detected in a source module.

shift - A set of eight instructions which move bits left or right in registers.

SI-type instruction - Instructions with a storage address in the first operand and a byte of immediate data in the second operand.

SS-type instruction - Instructions with storage addresses in both operands. In some, one length is specified with the first operand, and in others, lengths are specified with both operands.

sign bit - Bit 0 in a fixed-point binary field; 0 indicates a positive value and 1 a negative value.

significant digit - A digit whose value is greater than zero.

simulator program - A program that causes one computer to imitate the logical operation of another computer for purposes of measurement and evaluation. Primarily used to exercise program logic independent of hardware environment. Extremely useful for debugging logic prior to committing it to ROM.

single threaded - A type of chained list or queue in which each block of data contains a single pointer to the block ahead of it in the chain and the last block contains a zero. See also double threaded.

slot - In OS/VS, a continuous area on a paging device in which a page can be stored.

SNAP dump - A dynamic dump on an OS-based system.

software - The programs which aid the problem program in its execution.

sort - A programming routine that orders data.

source field - The operand that provides the data that is to participate in an operation. For example, the data that is moved by an MVC.

source module - The source statements that constitute the input to a language translator for a particular translation.

source program - A set of user-written instructions designed to solve a problem after compilation or assembly into machine-language object code.

source statement - A statement written in symbols of a programming language.

special character - A graphic character that is not an A-Z, 0-9, @, #, or $.

stack - A reserved storage area for holding temporary data.

stacked job - See batched job.

stacked job processing - A technique that permits multiple job definitions to be grouped (stacked) for presentation to the system, which automatically recognizes the jobs, one after the other.

stack, interrupt - A reserved memory area that automatically stores important registers whenever a program is interrupted. By accessing the stack from one end on a Last-In-First-Out (LIFO) basis, return from interrupts proceeds in exactly the reverse order that they occurred. Stack architecture needs fewer registers for temporary storage, provides easy handling of multiple-level interrupts and permits almost unlimited subroutine nesting.

standardization - Input media for cumputers is standardized by set codes on cards, tape, disk, etc. The media themselves are standardized in size, shape, thickness, etc., as they can be handled by machines.

static relocation - Really a case of no relocation at all. All programs are loaded at the same address.

storage - Part of the computer system into which data is entered and stored or from which data is retrieved.

storage fragmentation - See fragmentation

storage protection - A means of preventing a program from writing or storing in areas of storage that don't belong to it. In some cases a program is even prevented from access of such an area.

store - The process of placing data in storage or an auxiliary device.

stored program computer - A computer that is capable of holding not only the data to be iperated upon, but the instructions which make up the program that handles the data.

subroutine - A block which implements a section of the logic for solution of a problem. May be part of or separate from the rest of the routine.

subroutine call - The process of passing control to a subroutine.

subroutine, package, floating point - A subroutine that achieves floating-point arithmetic functions without additional hardware. Usually consists of routines for fixed to floating point conversion and vice versa, conversion from decimal to floating point and vice versa, and floating-point move, as well as such floating-point arithmetic functions as addition, subtraction, multiplication and division.

supervisor - The part of a control program that coordinates the use of resources and maintains the flow of CPU operations.

supervisor call - An instruction that interrupts the program being executed and passes control to the supervisor so that it can perform a specific service indicated by the instruction. Abbreviated SVC.

supervisor state - A state during which the central processing unit can execute input/output and other privileged instructions. Contrast with problem state.

SVC - Abbreviation for supervisor call.

switch - A programming device used to remember a condition.

symbol - Any group of eight or less alphameric and national characters that begins with an alphabetic or national (#,@,$) character, Same as symbolic name.

symbolic address - The specification of an address by using symbols which the assembler resolves into a base register and displacement.

symbolic language - A programming language which permits the programmer to use symbolic names, or mnemonics, to specify opcodes and the data for the operation. This makes the programmer's job much easier.

symbolic linkage - Symbols defined in one csect which can be referred to from another csect. They permit transfer of control in subroutines.

symbolic name - See symbol.

symbolic name space - The block of space occupied, or defined, by a source program.

symbolic parameter - A variable symbol declared in the prototype statement of a macro definition. A symbolic parameter is usually assigned a value from the corresponding operand in the macro instruction that calls the macro definition. See also keyword parameter, and positional parameter.

symbolic unit name - See logical unit name.

symbol table - See cross-reference table.

syntactically valid - The instruction follows all the rules that govern the structure of the assembler language.

syntax error - A specification in an instruction which does not follow the rules that govern the structure of assembler language. For example, an index register specified in the operand of an MVC instruction

system input device - A device specified as a source of an input stream.

system output device - A device assigned to record output data for a series of jobs.

system programmer - A programmer who plans, generates, maintains, extends, and controls the use of an operating system with the aim of improving the overall productivity of an installation. Also, a programmer who designs programming systems.

systems software - Generally, supervisory and support modules, as opposed to application programs. May include such programs as an operating system, an assembler, compilers, debug routines, text editors, library maintenance, utilities, I/O drivers and a linking loader.

table - A collection of related data items that are contained in elements which reside in continuous areas of storage.

table argument - See argument.

table function - See function.

table look up - The process of comparing a search argument to each argument portion in a table to locate a possible match.

target instruction - The instruction that is executed as the result of an execute (EX) instruction.

task queue - A queue that contains control information for all tasks in a system at any given time.

telecommunications - Data transmission between a computing system and remotely located devices via a unit that performs the necessary format conversion and controls the rate of transmission.

teleprocessing - The processing of data that is received from or sent to remote locations by way of telecommunication lines.

temporary data set - A data set that is created and deleted in the same job.

term - The smallest part of an expression that can be assigned a value.

throughput - The total volume of work performed by a computing system over a given period of time.

time sharing - A method of using a computing system that allows a number of users to execute programs concurrently and to interact with the programs during execution.

time slicing - A feature that can be used to prevent a job from monopolizing the central processing unit and thereby delaying the assignment of CPU time to other jobs. In systems with time sharing, the allocation of time slices to terminal jobs.

trace - A debugging tool that prints or displays a specific set of registers and/or memory locations as they are encountered throughout the execution of a program. Program execution is not interrupted, but a trace of the contents of key variables and registers is provided for later problem analysis.

trailer statement - The statement (MEND) that marks the end of a macro definition.

transfer - See branch.

translators - See assemblers and compilers.

traps - Halts inserted in object code that, when encountered during execution, cause a branch to a debug program. (See also breakpoint.)

trouble shoot - See debug.

truncate - Chopped off or ignored. For example, if 87.657 is truncated to 4 digits, the result is 87.65 , the 7 is simply ignored.

two's complement notation - Representation of negative binary numbers. Created by subtracting each digit of the number from the value of one and then adding one to the least significant digit.

type attribute - The type associated with a symbol. For example, F for fullword, H for halfword. Can be tested in conditional assembly instructions.

type subfield - A portion of a DC or DS statement that informs the assembler which type of constant is to be defined.

unary operator - An arithmetic operator having only one term. They can be used in absolute, relocatable, and arithmetic expressions. They are positive (+) and negative (-).

unconditional branch - An instruction which causes a branch to be taken each and every time it is executed.

USASCII. - Same as ASCII.

utilities - Standard routines of often-used functions, usually supplied as part of system software.

utility program - A problem program designed to perform an everyday task, such as transcribing data from one storage device to another.

validity check - A check that a code group is actually a character of the particular code in use. For example, a check to see that the combination of holes punched into a column of a card is a valid combination.

value subfield - The portion of the operand of a DC statement that specifies the constant to be assembled. If specified in the operand of a DS statement it is only used to establish the length of the constant.

variable - A symbolic location that can contain a variety of values.

variable-length data - Data which consists of a string of bytes of no fixed length and located on no specific integral boundary.

variable symbol - A symbol used in macro and conditional assembly processing that can assume any of a given set of values.

virtual address - In virtual storage systems, an address that refers to virtual storage and must, therefore, be translated into a real storage address when it is to be used.

virtual address space - In virtual storage systems, the virtual storage assigned to a job.

virtual storage - Addressable space that appears to the user to be real storage, from which instructions and data are mapped into real storage locations. The size of virtual storage is limited by the addressing scheme of the computing system (or virtual machine) and by the amount of auxilliary storage available, rather than by the actual number of real storage locations.

volume - A recording medium that is mounted and demounted as a unit, for example, a reel of tape or a disk pack.

volume serial number - A number in a volume label that is assigned when a volume is prepared for use in the system.

volume table of contents - A table on a direct access volume, that describes each dataset on the volume. Abbreviated VTOC.

VTOC - Abbreviation for volume table of contents.

word - A contiguous series of 32 bits, or four bytes, in storage which can be addressed as a unit. The address of the first byte of the word is evenly divisible by four.

work area - An area used to reference an input record or build an output record. Its name is specified in the GET or PUT instruction.

zoned-decimal format - A format in which each character occupies one byte with the first four bits being the zone protion and the second four bits the digit portion. The zone portion of the low-order byte is the sign of the number (A, C, E and F are positive signs, and B and D are negative).

APPENDIX B

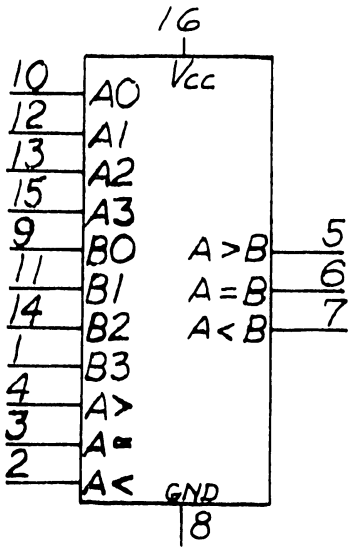
2200 VS COMMON IC'S

APPENDIX B

This section contains descriptions of integrated circuits that are commonly used throughout the 2200VS.

7485

4-BIT MAGNITUDE COMPARATOR



DESCRIPTION

This device checks the following functions between the two four-bit words. A=B, A B, A B.

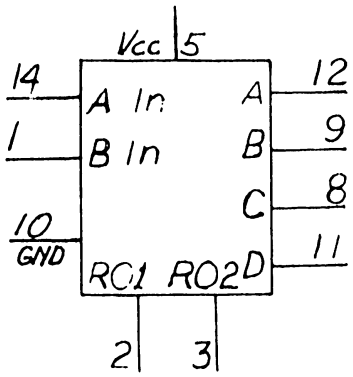
TRUTH TABLE

COMPARING INPUTS				CASCADING INPUTS			OUTPUTS		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H

NOTE: H = high level, L = low level, X = irrelevant

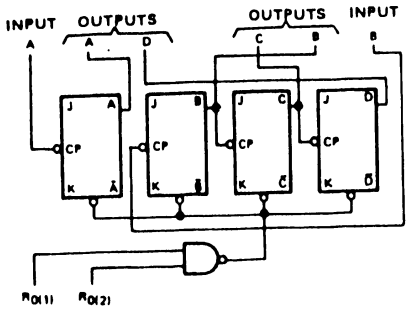
7493

4-BIT BINARY COUNTER



DESCRIPTION

For this device, a high level on both R0 inputs will cause the counter to reset to "0". A low level on R0 will cause a count on the next pulse. Without resetting, this device will count from 0-15 with the A output being the least significant bit.



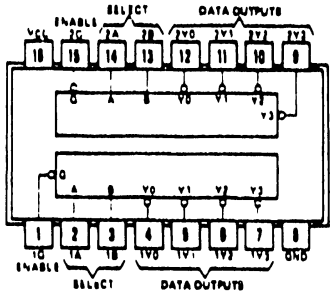
TRUTH TABLE (See Notes 1 and 2)

COUNT	OUTPUT			
	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

NOTES:
1. Output A connected to input B.
2. To reset all outputs to logical 0 both $R_{0(1)}$ and $R_{0(2)}$ inputs must be at a logical 1.
3. Either (or both) reset inputs $R_{0(1)}$ and $R_{0(2)}$ must be at a logical 0 to count.

74139

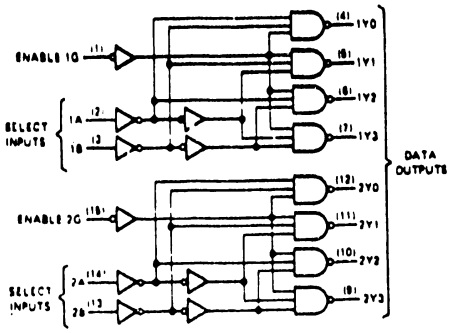
DUAL 2 TO 4 DECODERS



DESCRIPTION

With this device, a low level on the G input will cause the combination of A and B inputs to be decoded at the Y output.

'LS139, 'S139



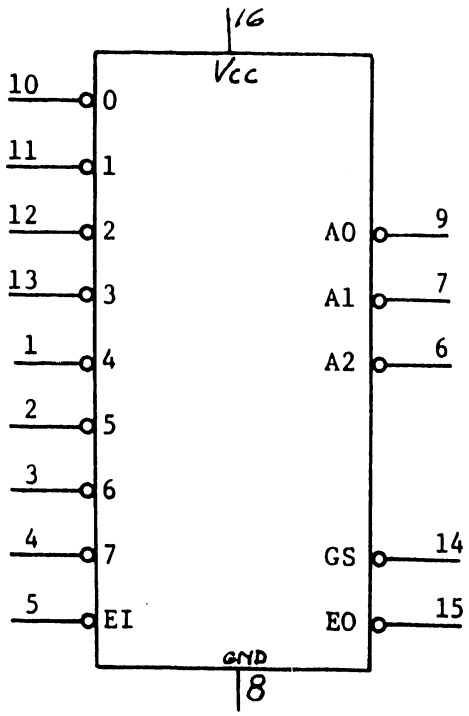
'LS139, 'S139
(EACH DECODER/DEMULTIPLEXER)
FUNCTION TABLE

INPUTS			OUTPUTS			
ENABLE	SELECT					
G	B	A	Y0	Y1	Y2	Y3
H	X	X	H	H	H	H
L	L	L	L	H	H	H
L	L	H	H	L	H	H
L	H	L	H	H	L	H
L	H	H	H	H	H	L

H = high level, L = low level, X = irrelevant

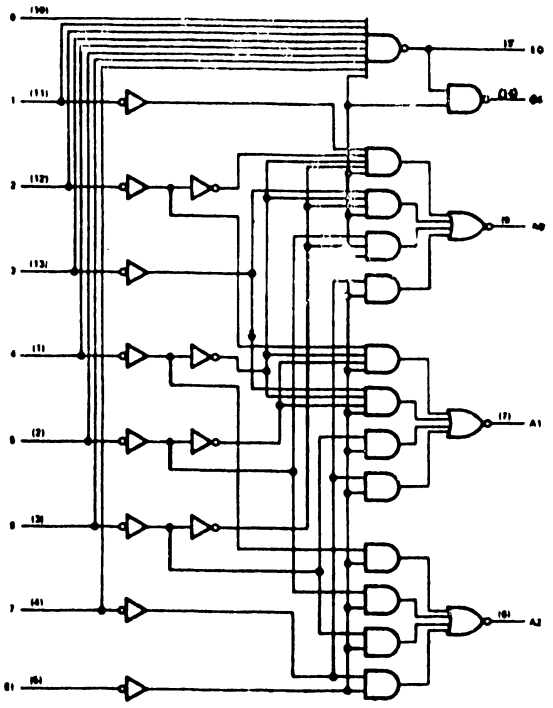
74148

8 TO 3 LINE PRIORITY ENCODER



DESCRIPTION

This device insures that only the highest-order data line is decoded. The El input enables the 8 inputs to be encoded in a BCD count.

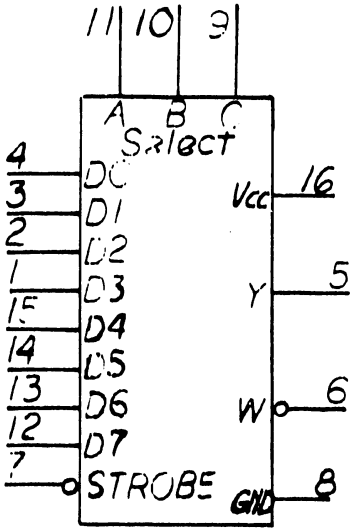


FUNCTION TABLE

INPUTS									OUTPUTS				
EI	0	1	2	3	4	5	6	7	A2	A1	A0	GS	EO
H	X	X	X	X	X	X	X	X	H	H	H	H	H
L	H	H	H	H	H	H	H	H	H	H	H	H	L
L	X	X	X	X	X	X	X	L	L	L	L	L	H
L	X	X	X	X	X	X	L	H	L	L	H	L	H
L	X	X	X	X	X	L	H	H	L	H	L	L	H
L	X	X	X	X	L	H	H	H	L	H	H	L	H
L	X	X	X	L	H	H	H	H	H	L	L	L	H
L	X	X	L	H	H	H	H	H	H	L	H	L	H
L	X	L	H	H	H	H	H	H	H	H	L	L	H
L	L	H	H	H	H	H	H	H	H	H	H	L	H

74151

8-BIT DATA SELECTOR/MULTIPLEXER



DESCRIPTION

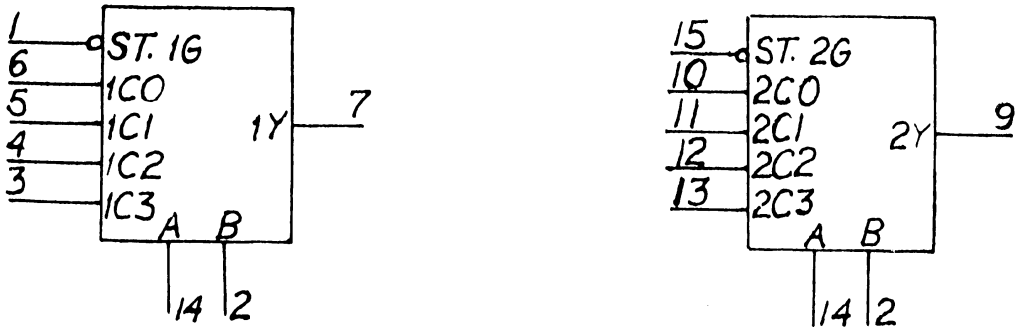
When this device is strobed low, the Y output will yield a selected D input . The W output is the inverted Y output.

INPUTS												OUTPUTS	
C	B	A	STROBE(1)	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	Y(1)	W
X	X	X	1	X	X	X	X	X	X	X	X	0	1
0	0	0	0	0	X	X	X	X	X	X	X	0	1
0	0	0	0	1	X	X	X	X	X	X	X	1	0
0	0	1	0	X	0	X	X	X	X	X	X	0	1
0	0	1	0	X	1	X	X	X	X	X	X	1	0
0	1	0	0	X	X	0	X	X	X	X	X	0	1
0	1	0	0	X	X	1	X	X	X	X	X	1	0
0	1	1	0	X	X	X	0	X	X	X	X	0	1
0	1	1	0	X	X	X	1	X	X	X	X	1	0
1	0	0	0	X	X	X	X	0	X	X	X	0	1
1	0	0	0	X	X	X	X	1	X	X	X	1	0
1	0	1	0	X	X	X	X	X	0	X	X	0	1
1	0	1	0	X	X	X	X	X	1	X	X	1	0
1	1	0	0	X	X	X	X	X	X	0	X	0	1
1	1	0	0	X	X	X	X	X	X	1	X	1	0
1	1	1	0	X	X	X	X	X	X	X	0	0	1
1	1	1	0	X	X	X	X	X	X	X	1	1	0

X = Irrelevant.

74153

DUAL 4-LINE-TO-1-LINE DATA SELECTOR



DESCRIPTION

When this device is strobed low, it will decode the A & B inputs, and put the corresponding C input on the Y output.

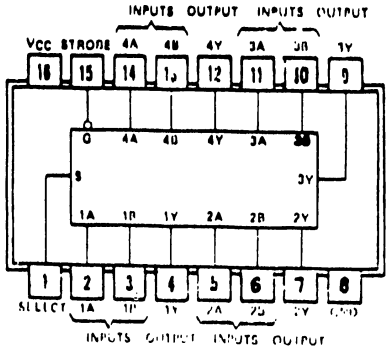
TRUTH TABLE

ADDRESS INPUTS		DATA INPUTS				STROBE	OUTPUT
B	A	C0	C1	C2	C3	G	Y
X	X	X	X	X	X	1	0
0	0	0	X	X	X	0	0
0	0	1	X	X	X	0	1
0	1	X	0	X	X	0	0
0	1	X	1	X	X	0	1
1	0	X	X	0	X	0	0
1	0	X	X	1	X	0	1
1	1	X	X	X	0	0	0
1	1	X	X	X	1	0	1

Address inputs A and B are common to both sections.
X = irrelevant.

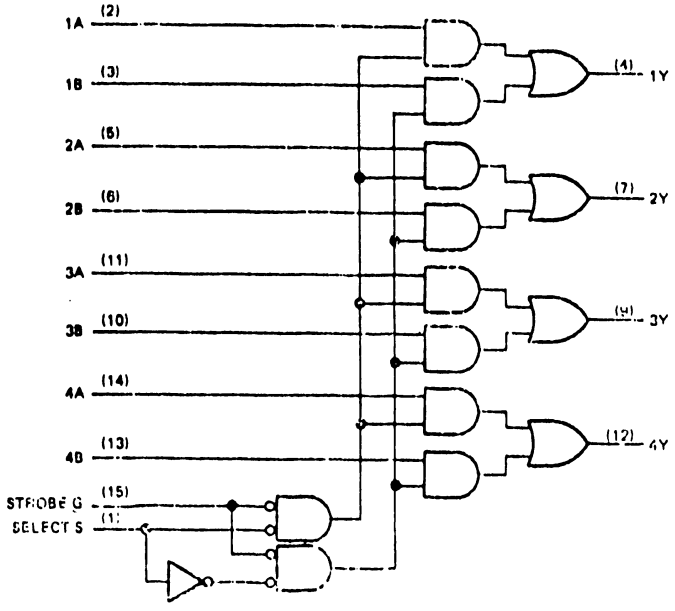
74157

QUAD 2-TO-1 LINE DATA SELECTOR



DESCRIPTION

If the select line of this device is low when the device is strobed, the A inputs will appear on the Y outputs, or if the select line is high when the device is strobed, the B inputs will appear on the Y outputs.



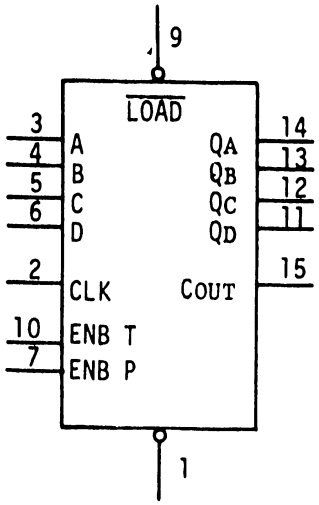
FUNCTION TABLE

INPUTS				OUTPUT Y	
STROBE	SELECT	A	B	'157, 'L157, 'LS157, 'S157	'LS158 'S158
H	X	X	X	L	H
L	L	L	X	L	H
L	L	H	X	H	L
L	H	X	L	L	H
L	H	X	H	H	L

H = high level, L = low level, X = irrelevant

74161

SYNCHRONOUS 4-BIT COUNTER



DESCRIPTION

This device is fully programmable; the outputs may be set to either state. If Load goes low, the A, B, C, D inputs will appear at the QA, QB, QC, QD outputs on the next clock pulse. Both enable inputs must be high to clock the counter.

74163

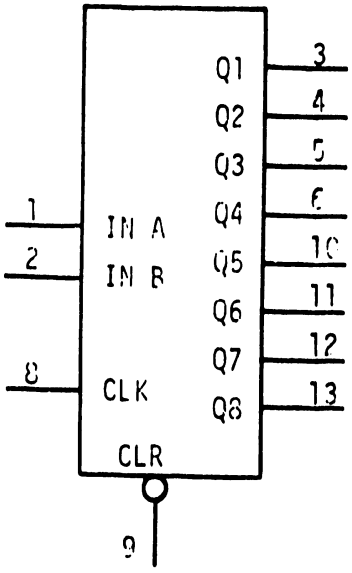
SYNCHRONOUS 4-BIT COUNTER

DESCRIPTION

Same as 74161.

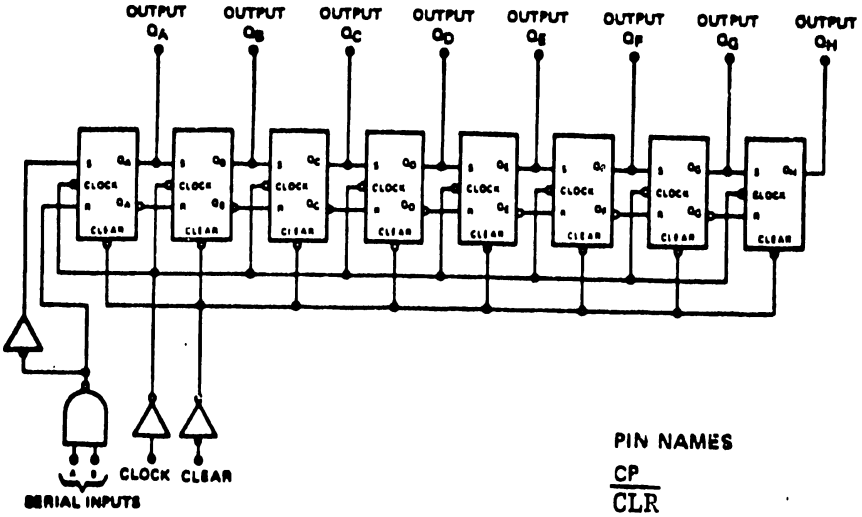
74164

8-BIT SERIAL TO PARALLEL CONVERTER



DESCRIPTION

This device "ANDs" the A & B inputs, and shifts this through the register, from Q1 to Q8, with each clock pulse. A low level on the clear line resets the entire register.



TRUTH TABLE
SERIAL INPUTS A AND B

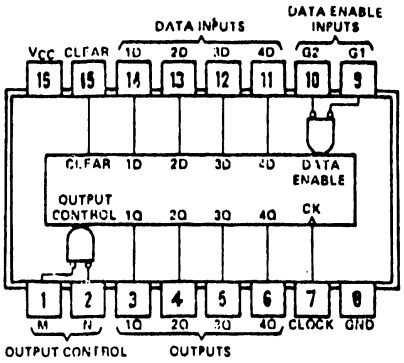
INPUTS AT t_n		OUTPUT AT t_{n+1}
A	B	Q_A
H	H	H
L	H	L
H	L	L
L	L	L

PIN NAMES

$\overline{CP}$	Clock Pulse Input
$\overline{CLR}$	Clear Input
A, B	Serial Inputs
Q_A to Q_H	Parallel Outputs

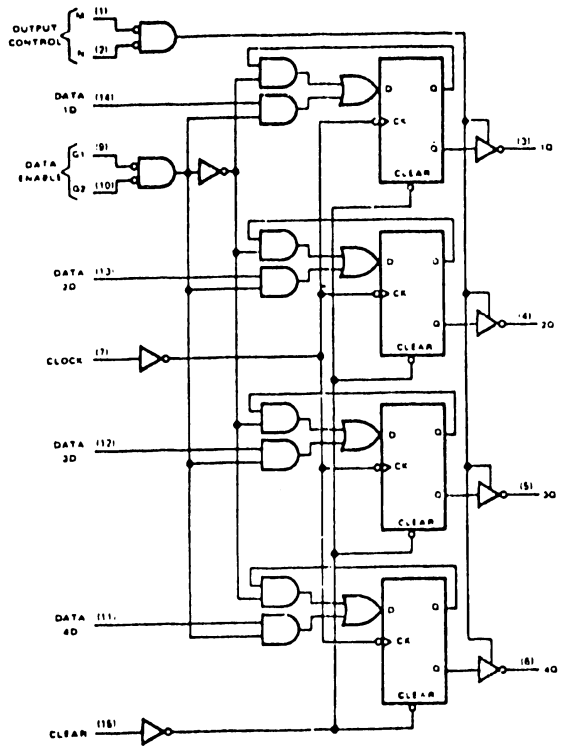
74173

4-BIT D-TYPE REGISTERS WITH TRI-STATE OUTPUTS



DESCRIPTION

This device allows the data inputs to be clocked into the device, if both data enables are at a low level. The data will be present on the outputs if the M and N lines are held low. If the M and N lines are at a high level, the outputs go to the high impedance state, but do not interfere with any data being clocked in. A high level on the clear pin will clear the entire register.



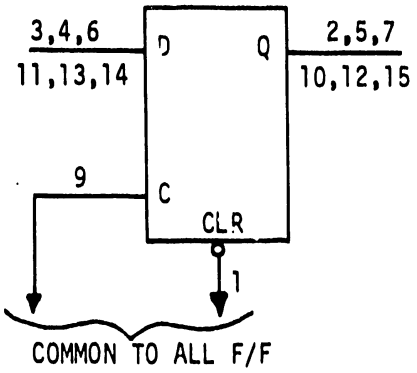
FUNCTION TABLE

INPUTS					OUTPUT
CLEAR	CLOCK	DATA ENABLE		DATA	Q
		G1	G2	D	
H	X	X	X	X	L
L	L	X	X	X	Q_0
L	↑	H	X	X	Q_0
L	↑	X	H	X	Q_0
L	↑	L	L	L	L
L	↑	L	L	H	H

When either M or N (or both) is (are) high the output is disabled to the high-impedance state; however sequential operation of the flip-flops is not affected.

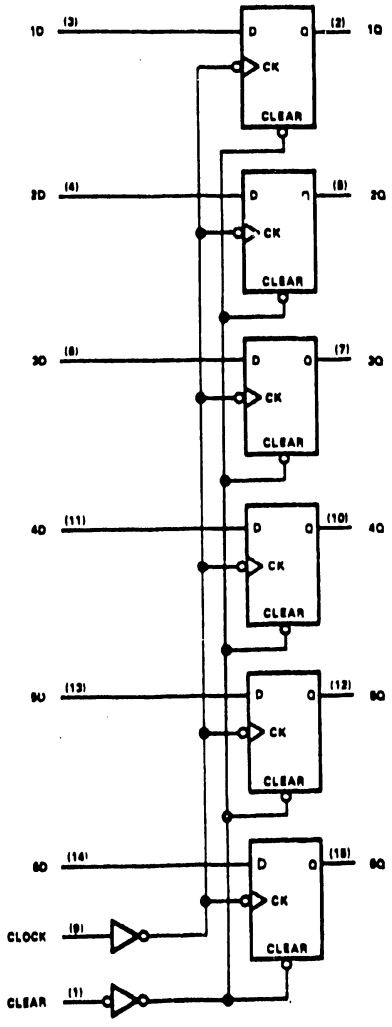
H = high level (steady state)
L = low level (steady state)
↑ = low-to-high-level transition
X = irrelevant (any input including transitions)
Q_0 = the level of Q before the indicated steady state input conditions were established.

HEX D-TYPE FLIP-FLOPS



DESCRIPTION

This device transfers the data from the D input to the Q output with each clock. A low level on the clear will clear all six flip-flops.

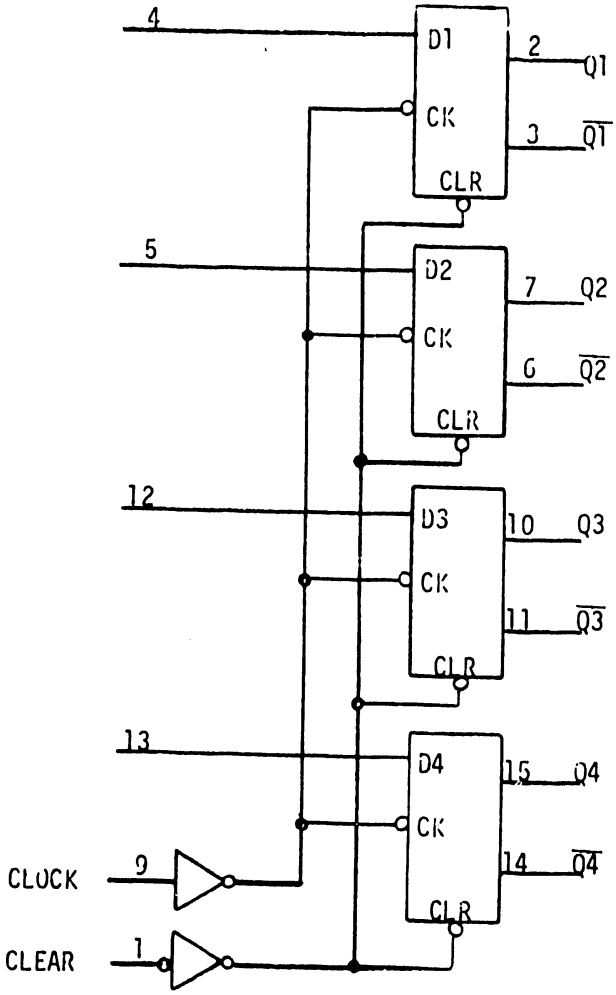


FUNCTION TABLE
(EACH FLIP-FLOP)

INPUTS			OUTPUTS
CLEAR	CLOCK	D	Q
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q_0

H = high level (steady state)
L = low level (steady state)
X = irrelevant
↑ = transition from low to high level
Q_0 = the level of Q before the indicated steady-state input conditions were established.

QUAD D-TYPE FLIP-FLOPS



FUNCTION TABLE
(EACH FLIP-FLOP)

INPUTS			OUTPUTS	
CLEAR	CLOCK	D	Q	$\bar{Q}$↑
L	X	X	L	H
H	↑	H	H	L
H	↑	L	L	H
H	L	X	Q_0	$\bar{Q}_0$

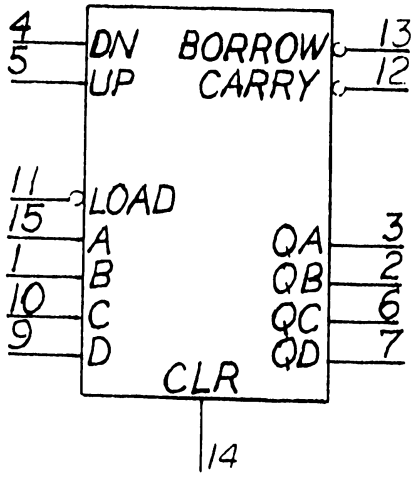
H = high level (steady state)
L = low level (steady state)
X = irrelevant
↑ = transition from low to high level
Q_0 = the level of Q before the indicated steady-state input conditions were established.

DESCRIPTION

These monolithic, positive-edge-triggered flip-flops utilize TTL circuits to implement the D-type flip-flop logic. Information at input D is transferred to the Q output on the positive-going edge of the clock pulse. Clock triggering occurs at a voltage level of the clock pulse and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the high or low level, the D-input signal has no effect.

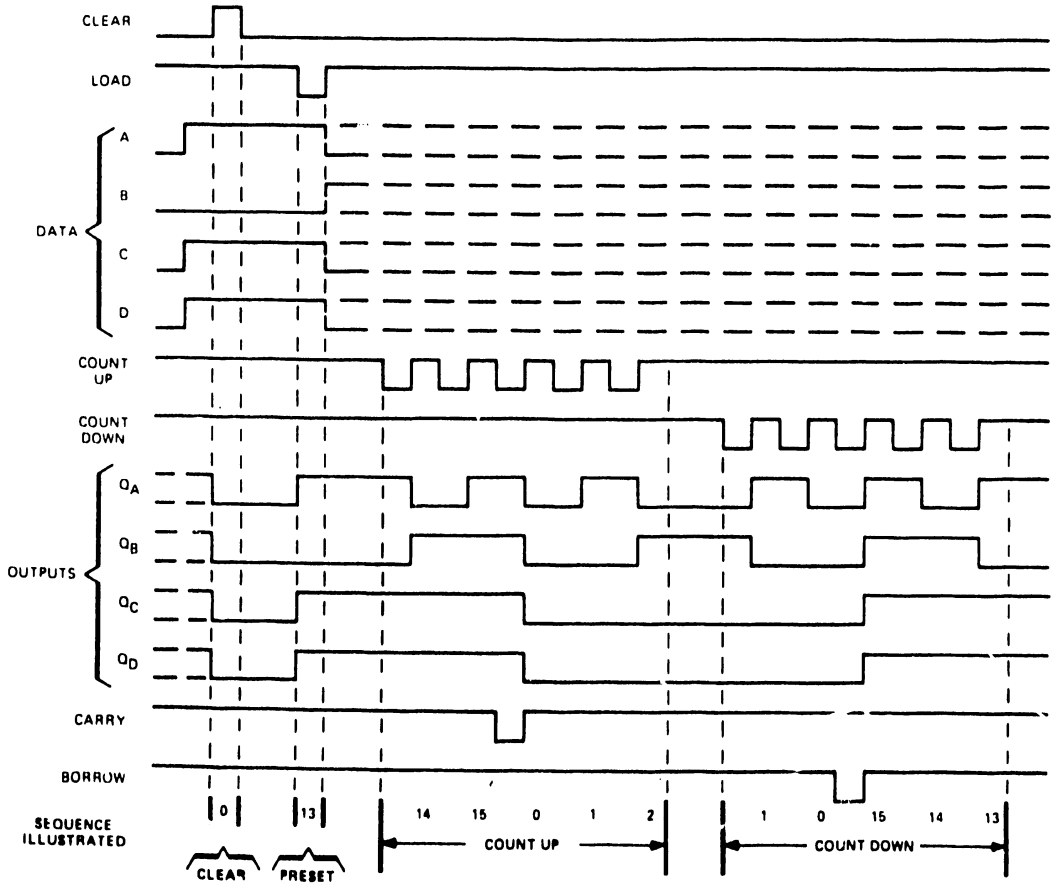
74193

SYNCHRONOUS 4-BIT UP/DOWN BINARY COUNTER



Illustrated below is the following sequence:

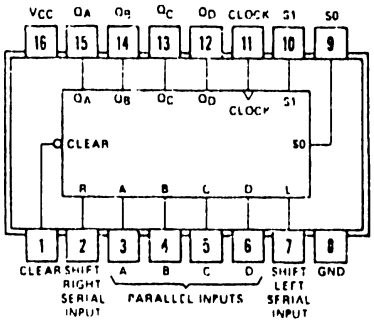
1. Clear outputs to zero.
2. Load (preset) to BCD thirteen.
3. Count up to fourteen, fifteen, carry, zero, one, and two.
4. Count down to one, zero, borrow, fifteen, fourteen, and thirteen.



NOTES: A. Clear overrides load, data, and count inputs.
B. When counting up, count-down input must be high; when counting down, count up input must be high.

74194

4 BIT BI-DIRECTIONAL SHIFT REGISTER



description

These bidirectional shift registers are designed to incorporate virtually all of the features a system designer may want in a shift register. The circuit contains 46 equivalent gates and features parallel inputs, parallel outputs, right-shift and left-shift serial inputs, operating-mode-control inputs, and a direct overriding clear line. The register has four distinct modes of operation, namely:

Parallel (Broadside) Load
Shift Right (In the direction Q_A toward Q_D)
Shift Left (In the direction Q_D toward Q_A)
Inhibit Clock (Do nothing)

Synchronous parallel loading is accomplished by applying the four bits of data and taking both mode control inputs, S_0 and S_1, high. The data is loaded into the associated flip-flop and appears at the outputs after the positive transition of the clock input. During loading, serial data flow is inhibited.

Shift right is accomplished synchronously with the rising edge of the clock pulse when S_0 is high and S_1 is low. Serial data for this mode is entered at the shift-right data input. When S_0 is low and S_1 is high, data shifts left synchronously and new data is entered at the shift-left serial input.

Clocking of the flip-flop is inhibited when both mode control inputs are low. The mode controls of the SN54194/SN74194 should be changed only while the clock input is high.

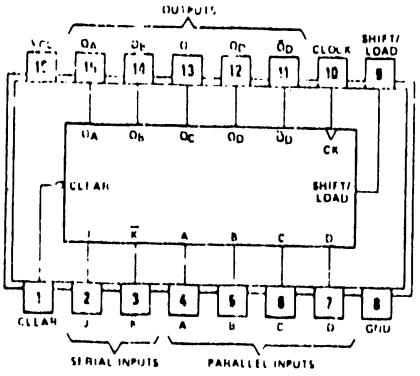
FUNCTION TABLE

INPUTS									OUTPUTS			
CLEAR	MODE		CLOCK	SERIAL		PARALLEL						
	S_1	S_0		LEFT	RIGHT	A	B	C	D	Q_A	Q_B	Q_C	Q_D
L	X	X	X	X	X	X	X	X	X	L	L	L	L
H	X	X	L	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}
H	H	H	↑	X	X	a	b	c	d	a	b	c	d
H	L	H	↑	X	H	X	X	X	X	H	Q_{An}	Q_{Bn}	Q_{Cn}
H	L	H	↑	X	L	X	X	X	X	L	Q_{An}	Q_{Bn}	Q_{Cn}
H	H	L	↑	H	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	H
H	H	L	↑	L	X	X	X	X	X	Q_{Bn}	Q_{Cn}	Q_{Dn}	L
H	L	L	X	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}

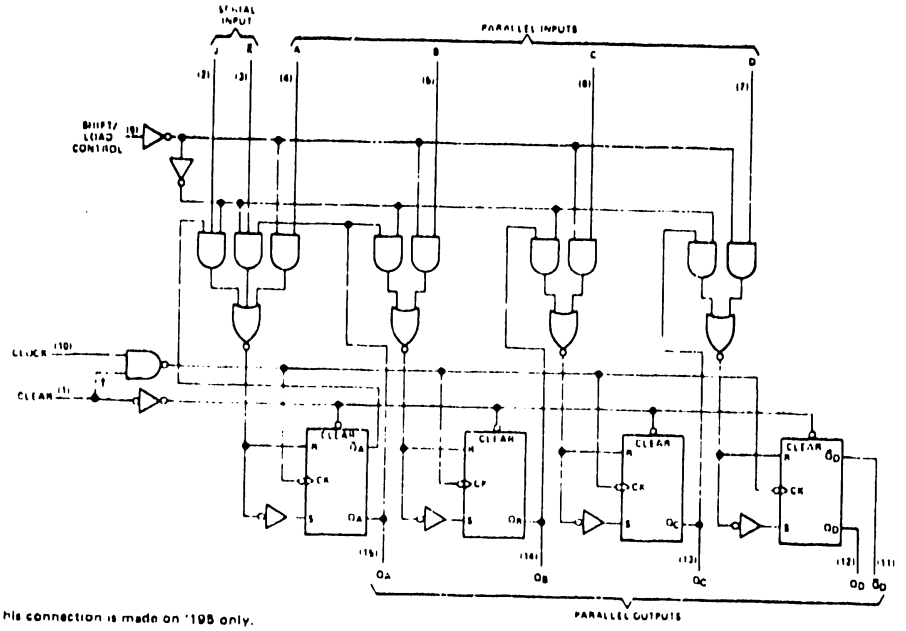
H = high level (steady state)
L = low level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level
a, b, c, d = the level of steady-state input at inputs A, B, C, or D, respectively
Q_{A0}, Q_{B0}, Q_{C0}, Q_{D0} = the level of Q_A, Q_B, Q_C, or Q_D, respectively, before the indicated steady-state input conditions were established
Q_{An}, Q_{Bn}, Q_{Cn}, Q_{Dn} = the level of Q_A, Q_B, Q_C, Q_D, respectively, before the most-recent ↑ transition of the clock

74195

4-BIT SHIFT REGISTER



functional block diagram



† This connection is made on '195 only.

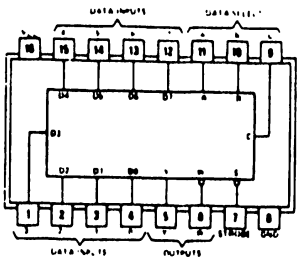
FUNCTION TABLE

INPUTS									OUTPUTS				
CLEAR	SHIFT/ LOAD	CLOCK	SERIAL		PARALLEL				Q_A	Q_B	Q_C	Q_D	$\bar{Q}_D$
			J	$\bar{K}$	A	B	C	D					
L	X	X	X	X	X	X	X	X	L	L	L	L	H
H	L	↑	X	X	a	b	c	d	a	b	c	d	$\bar{d}$
H	H	L	X	X	X	X	X	X	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}	$\bar{Q}_{D0}$
H	H	↑	L	H	X	X	X	X	Q_{An}	Q_{An}	Q_{Bn}	Q_{Cn}	$\bar{Q}_{Cn}$
H	H	↑	L	L	X	X	X	X	L	Q_{An}	Q_{Bn}	Q_{Cn}	$\bar{Q}_{Cn}$
H	H	↑	H	H	X	X	X	X	H	Q_{An}	Q_{Bn}	Q_{Cn}	$\bar{Q}_{Cn}$
H	H	↑	H	L	X	X	X	X	$\bar{Q}_{An}$	Q_{An}	Q_{Bn}	Q_{Cn}	$\bar{Q}_{Cn}$

H = high level (steady state)
L = low level (steady state)
X = irrelevant (any input, including transitions)
↑ = transition from low to high level
a, b, c, d = the level of steady state input at A, B, C, or D, respectively
Q_{A0}, Q_{B0}, Q_{C0}, Q_{D0} = the level of Q_A, Q_B, Q_C, or Q_D, respectively, before the indicated steady state input conditions were established
Q_{An}, Q_{Bn}, Q_{Cn} = the level of Q_A, Q_B, or Q_C, respectively, before the most recent transition of the clock

74251

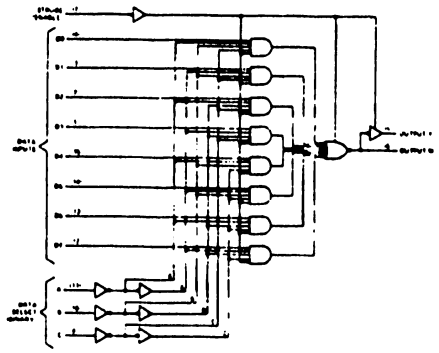
DATA SELECTORS/MULTIPLEXERS WITH TRI-STATE OUTPUTS



DESCRIPTION

With this device, the decoded 'data-selects' will cause the corresponding data input to be output. This can only happen with a low level strobe. The Y output is the true output, while the W output is inverted.

functional block diagram



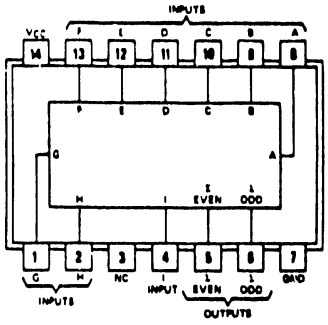
FUNCTION TABLE

INPUTS				OUTPUTS	
SELECT			STROBE		
C	B	A	S	Y	W
X	X	X	H	Z	Z
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

H = high logic level, L = low logic level
X = irrelevant, Z = high impedance (off)
D0, D1 . . . D7 = the level of the respective D input

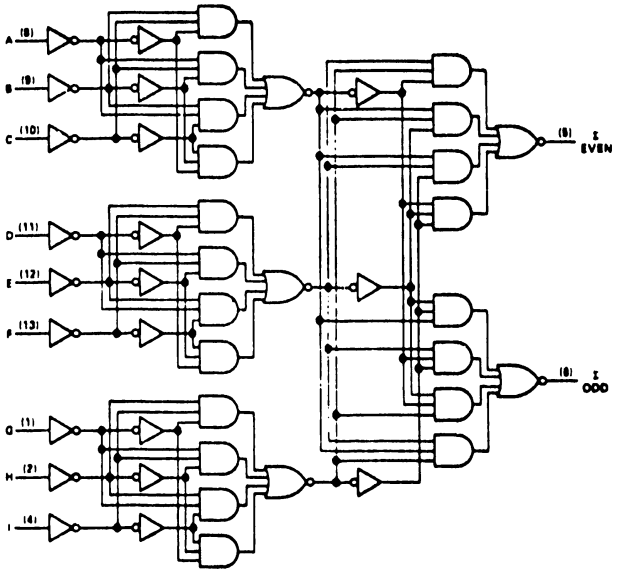
74280

9-BIT ODD/EVEN PARITY GENERATOR/CHECKER



DESCRIPTION

With this device, if 0, 2, 4, 6, or 8 total inputs are high level then, even output will be high level and odd output will be low level. If 1, 3, 5, 7, or 9 total inputs are high level then, even output will be low level and odd output will be high level.

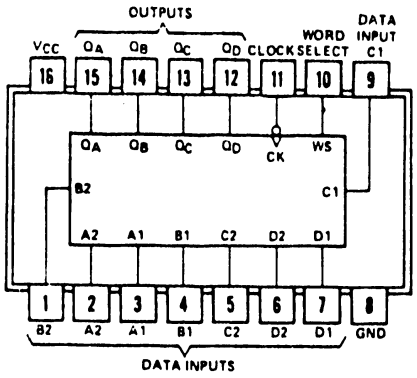


FUNCTION TABLE

NUMBER OF INPUTS A THRU I THAT ARE HIGH	OUTPUTS	
	Σ EVEN	Σ ODD
0, 2, 4, 6, 8	H	L
1, 3, 5, 7, 9	L	H

H = high level, L = low level

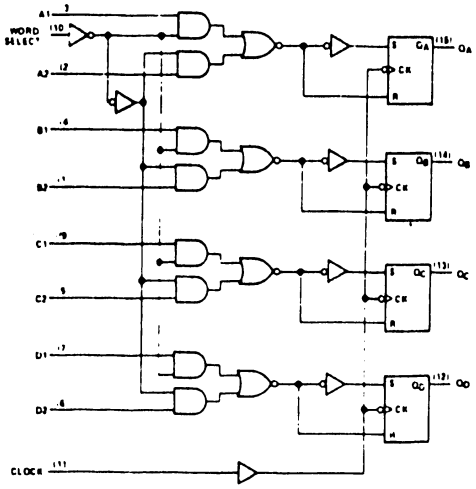
QUAD 2-INPUT MULTIPLEXERS W/STORAGE



DESCRIPTION

When the word select input is low, word 1 is applied to the flip-flops. A high input to word select will cause word 2 to be selected. The selected word must then be clocked into the flip-flops.

functional block diagram



⊣◇ . . . Dynamic input activated by a transition from a high level to a low level

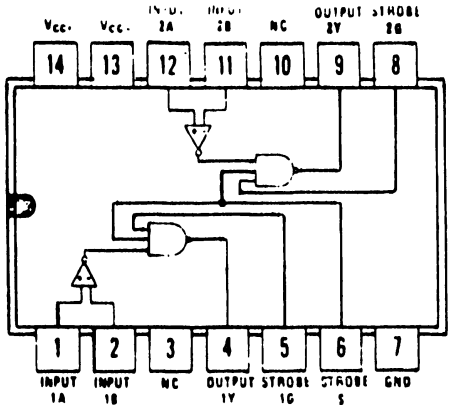
FUNCTION TABLE

INPUTS		OUTPUTS			
WORD SELECT	CLOCK	Q_A	Q_B	Q_C	Q_D
L	↓	a1	b1	c1	d1
H	↓	a2	b2	c2	d2
X	H	Q_{A0}	Q_{B0}	Q_{C0}	Q_{D0}

H = high level (steady state)
L = low level (steady state)
X = irrelevant (any input, including transitions)
↓ = transition from high to low level
a1, a2, etc. = the level of steady-state input at A1, A2, etc.
Q_{A0}, Q_{B0}, etc. = the level of Q_A, Q_B, etc. entered on the most-recent ↓ transition of the clock input.

75107

DUAL LINE RECEIVER



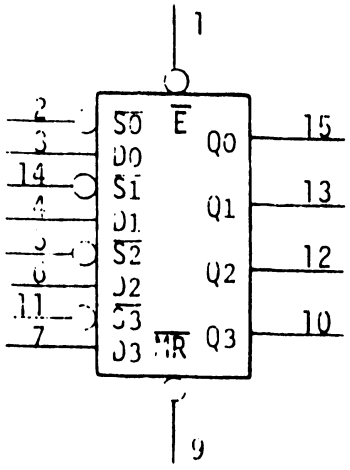
The 75107A features independent channels of common voltage supply and ground terminals, a TTL-compatible active pull-up (totempole) output, high input impedance and low input currents which induce very little loading on the transmission line, and individual strobe inputs for each channel and a strobe input common to both channels for logic versatility.

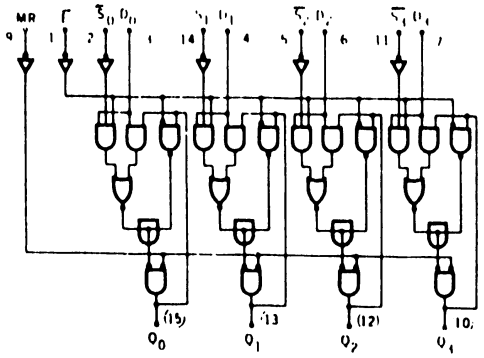
TRUTH TABLE

DIFFERENTIAL INPUTS A-B	STROBES G	STROBES S	OUTPUT Y
$V_{ID} \geq 25$ mV	L or H	L or H	H
-25 mV $< V_{ID} < 25$ mV	L or H	L	H
	L	L or H	H
	H	H	INDETERMINATE
$V_{ID} \leq -25$ mV	L or H	L	H
	L	L or H	H
	H	H	L

9314

QUAD LATCH





PIN NAMES

$\overline{E}$	(Active LOW) Enable Input
D_0, D_1, D_2, D_3	Data Inputs
$\overline{S}_0, \overline{S}_1, \overline{S}_2, \overline{S}_3$	Set (Active LOW) Inputs
$\overline{MR}$	Master Reset (Active LOW) Input
Q_0, Q_1, Q_2, Q_3	Latch Outputs

○ PIN NUMBERS

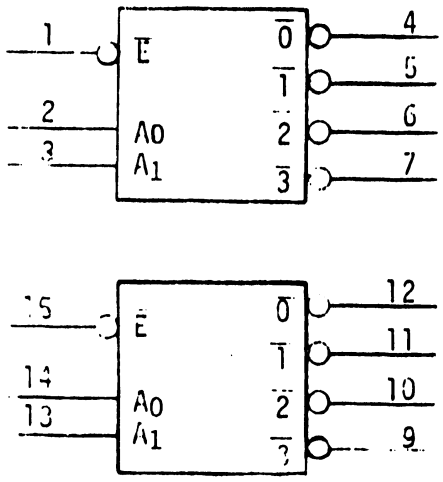
TRUTH TABLE

$\overline{MR}$	$\overline{E}$	D	$\overline{S}$	Q_N	OPERATION
H	L	L	L	L	D MODE
H	L	H	L	H	
H	H	X	X	Q_{N-1}	
H	L	L	L	L	R/S MODE
H	L	H	L	H	
H	L	L	H	L	
H	L	H	H	Q_{N-1}	
H	H	X	X	Q_{N-1}	
L	X	X	X	L	RESET

X = Don't Care
L = LOW Voltage Level
H = HIGH Voltage Level
Q_{N-1} = Previous Output State
Q_N = Present Output State

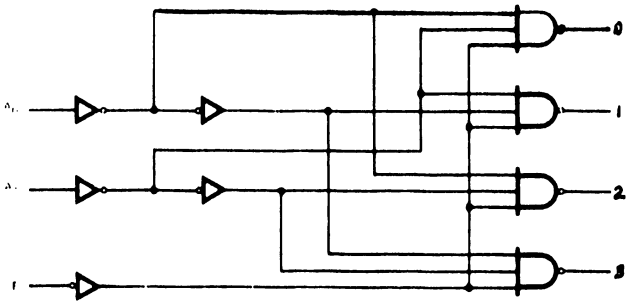
9321

DUAL ONE-OF-FOUR DECODER



DESCRIPTION

With this device, when the E input is a low level, the A inputs will be decoded, appearing as low truth outputs.



TRUTH TABLE
DECODER 1 & 2

$\overline{E}$	A_0	A_1	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$
L	L	L	L	H	H	H
L	H	L	H	L	H	H
L	L	H	H	H	L	H
L	H	H	H	H	H	L
H	X	X	H	H	H	H

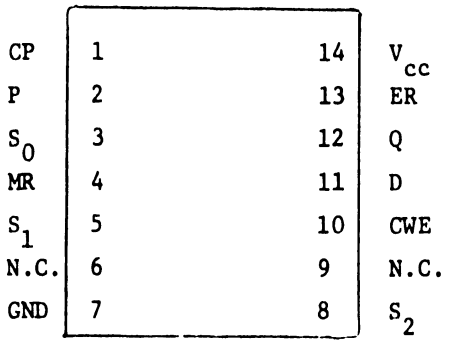
H = HIGH Voltage Level
L = LOW Voltage Level
X = Level Does Not Affect Output

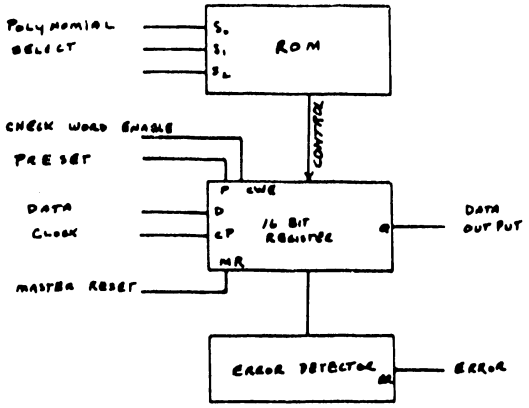
PIN NAMES

Decoder 1 and 2	
$\overline{E}$	Enable (Active LOW) Input
A_0, A_1	Address Inputs
$\overline{0}$, $\overline{1}$, $\overline{2}$, $\overline{3}$	(Active LOW) Outputs

9401

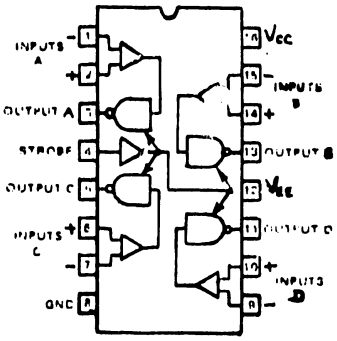
CRC GENERATOR CHECKER





MC 3450

QUAD LINE RECEIVER



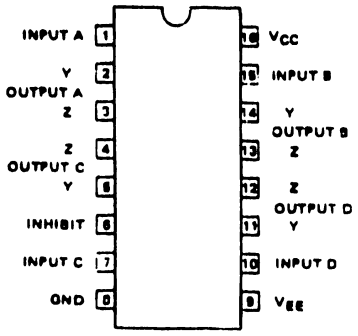
TRUTH TABLE

INPUT	STROBE	OUTPUT MC3450
$V_{ID} \geq$ +25 mV	L	H
	H	Open
-25 mV $\leq V_{ID} \leq$ +25 mV	L	L
	H	Open
$V_{ID} \leq$ -25 mV	L	L
	H	Open

L = Low Logic State
H = High Logic State
Open = High Impedance State
I = Indeterminate State

MC 3453

QUAD LINE DRIVER



TRUTH TABLE
(positive logic)

LOGIC INPUT	INHIBIT INPUT	OUTPUT CURRENT Z	OUTPUT CURRENT Y
H	H	On	Off
L	H	Off	On
H	L	Off	Off
L	L	Off	Off

L = Low Logic Level
H = High Logic Level

APPENDIX C

UPGRADING THE WCS/60

APPENDIX C

UPGRADING THE WCS/60

The user who wishes to upgrade his WCS/60 CPU can expand it to a WCS/80 CPU. The WCS/80 can support up to 512K bytes of main memory, up to 23 Workstations, and 600 megabytes of on-line disk storage.

Additional options for the WCS/60 include:

Memory upgrade - increments of 64K (256K maximum)
Additional Language - COBOL or BASIC
2246P Workstations - up to 15 additional workstations
IOP's (up to four additonal)

22V01	Printer Workstation IOP
22V02	10 megabyte fixed/removable disk IOP
22V03	75 megabyte removable disk IOP (one per system)
22V05	9-track, 1600 bpi, 120 kilobyte/sec (75 ips) magnetic tape IOP

Disk Drives

2260V	10 megabyte fixed/removable disk drive
2265V-1	75 megabyte removable disk drive

Tape Drive

2209V	9-track, 1600 bpi, 120 kilobyte/sec (75 ips) magnetic tape drive

Printers

2281	Daisy Output Writer (50 cps) Pin feed forms feeder for 2281V
2231W-1	Wang Line Printer (112 col./120 cps)
2231W-2	Wang Line Printer (132 col./120 cps)
2221W	Wang Line Printer (132 col./200 cps)
2261W	Wang Printer (240 lpm)
2263V-1	Line Printer (400 lpm)
2263V-2	Line Printer (600 lpm)

APPENDIX D

UPGRADING THE WCS/80

UPGRADING THE WCS/80

Additional options for the WCS/80 include:

Memory upgrade - increments of 64K (512K maximum memory)

Additional language - COBOL or BASIC

2246P Workstations - up to 22 additional Workstations

Additional IOP's (system supports a total of eight)

22V01	Printer/Workstation IOP
22V04	80MB removable disk IOP (up to two per system)
22V05	9 track, 1600 bpi, 120 kilobyte/sec (75 ips) magnetic tape IOP (controls up to four magnetic tapes)

Disk Drives

2265V-1	75MB removable disk drive

Tape Drives

2209V	9 Track, 1600bpi, 120K kilobyte/sec (75 ips) magnetic tape drive

Printers

2281	Daisy output writer (40 cps)
	pin feed forms feeder for 2281V
2231W-1	Wang line printer (112 col/120cps)
2231W-2	Wang line printer (132 col/120cps)
2221W	Wang line printer (132 col/200cps)
2263V-1	Line Printer (400 lpm)
2263V-2	Line Printer (600 lpm)

APPENDIX E CONFIG-URATIONS

APPENDIX E
CONFIGURATIONS

WCS/60 -

	MINIMUM	MAXIMUM
MEMORY	64KB	256KB
I/O PROCESSORS	2	6
WORKSTATIONS	1	16
TAPE DRIVE	-	
DISKETTE DRIVES	1	1
DISK DRIVES -		
10 MEG F/R	1	
80 MEG R	1	2
PRINTERS (ANY TYPE)	1	

WCS/80 -

	MINIMUM	MAXIMUM
MEMORY	256KB	512KB
I/O PROCESSORS	3	8
WORKSTATIONS	1	24
TAPE DRIVES	-	
DISKETTE DRIVES	1	1
DISK DRIVES -		
10 MEG F/R	-	
80 MEG R	2	8
PRINTERS (ANY TYPE)	1	

MEMORY IS IN 64KB INCREMENTS

WCS/60

SMALL

WCS/60-2, 64K, 1 WORKSTATION, 10-MEG DISK, DISKETTE, 240-LPM PRINTER

MEDIUM

WCS/60-6, 192K, 6 WORKSTATIONS, 2 10-MEG DISKS, 1 240-LPM PRINTER, DISKETTE

LARGE

WCS/60-8, 256K, 10 WORKSTATIONS, 2 80-MEG DISK DRIVES, 1 600-LPM PRINTER, DISKETTE

WCS/80

SMALL

WCS/80-8, 256K, 2 80-MEG DISK DRIVES, 1 WORKSTATION, 600-LPM PRINTER, DISKETTE

MEDIUM

WCS/80-12, 384K, 10 WORKSTATIONS, 600-LPM PRINTER, 3 8-MEG DISK DRIVES, DISKETTE

LARGE

WCS/80-16, 512K, 18 WORKSTATIONS, 4 80-MEG DISK DRIVES, 1 9-TRACK TAPE DRIVE, 2 600-LPM PRINTERS, DISKETTE

APPENDIX F SPECIFI-CATIONS

APPENDIX F

SPECIFICATIONS

2200VS CENTRAL PROCESSING UNIT

Memory Size: 64K, 128K, 1924K, 256K (WCS-60)

Memory Size: 256K, 320K, 384K, 448K, 512K (WCS-80)

Size

Height	41 in. (104 cm)
Depth	32 in. (81 cm)
Width	36 in. (9 com)
Weight	
Cable	

Power Requirements

115 VAC (± 10%)

60 HZ (± 1 HZ)

12 Amp.

Independent Power Line Recommended

Operating Environment

50 F to 90 F (10 C to 32 C)

20% to 80% Relative Humidity

Recommended Relative Humidity

35% to 65%

2200VS GENERAL SPECIFICATIONS

Memory Cycle Time

660 nanoseconds per two bytes

Word Length

32 bits (4 bytes)

Registers

Sixteen 32-bit General-Purpose Registers
Four 64-bit Floating-Point Registers
Eight 32-bit Control Registers

Types of Arithmetic

Binary
Packed Decimal
Floating Point

APPENDIX G

2260V FIXED/ REMOVABLE DISK

APPENDIX G

2260V FIXED/REMOVABLE DISK

Storage Capacity

Tracks/Cylinder	4
Cylinders	408
Sector Size	256 bytes
Sectors per Track	24
Total Storage (in Million Bytes)	10.03

Access Time

Average	35ms
Maximum	103ms
Minimum	9ms
Full Rotation Time	25ms
Data Transfer Rate	312kb/sec.

Power Requirements

115 VAC ($\pm$ 10%)
60 HZ ($\pm$ 1 HZ)
5.2 Amp.

APPENDIX
H
2265V-1
REMOVABLE
DISK PACK

APPENDIX H

2265V-1 REMOVABLE DISK PACK

Storage Capacity

Tracks/Cylinder	45
Cylinders	823
Sector size	2048 bytes
Sectors per Track	9
Total Storage (in Million Bytes)	75.85

Access Time

Average	30ms
Maximum	55ms
Minimum	6ms
Full Rotation Time	16.66ms
Data Transfer Rate	1.2mb/sec.

Power Requirements

115 VAC (+ 10%)
60 HZ (+ $\overline{1}$ HZ)
8.2 Amp. (operating), 1.5 Amp. (standby)
Starting current 10 seconds at 40 Amp.

WCS-60/80 DISKETTE

Storage Capacity

Cylinders	77
Sector Size	256 bytes
Sectors per Track	16
Total Storage (in Million Bytes)	.3154

Access Time

Average	424ms
Maximum	847ms
Minimum	11ms
Full Rotation Time	167ms
Data Transfer Rate	31kb/sec.

APPENDIX I

WORK-STATION

APPENDIX I

WORK STATION

CRT

Display Size.	12 in. diagonal (30.4cm)
Capacity.	24 lines, 80 characters/line
Character Size	
height	0.16 in (0.4064cm)
width	0.09 in (0.2286cm)

Power Requirements

115 or 230 VAC ± 10%
50 or 60 Hz ± 1/2 Hz
125 Watts

Fuses

2.0A@ SB 115V
1.0A@ SB 130V

Operating Environment

50 F to 90 F (10 C to 32 C)
20% to 80% Relative humidity allowable
35% to 65% Relative humidity recommended

Cable

One 8ft (2.4m) cord to power source
One length of 25ft (7.6m) direct connection cable is provided with each 2246P workstation; extension cables in increments for distances up to 500ft (152m) are available optionally. Each cable is equipped with a 36-pin amphenol connector.

APPENDIX J

226IV PRINTER

2261V PRINTER

Printer Size:

height	36 inc. (91cm)
depth	26 in. (66 cm)
width	27 in. (68.6cm)
weight	210 lb (94.5kg)
speed	240 lines/min

Character Configurations

11x8 and 9x8 dot matrix (dots not in adjacent columns of same row.)
10 char/in. (4 char/cm) or 11.76 char/in. (4.6 char/cm), selectable 6 lines/in (2.4 lines/com) or 8 lines/in. (3.1 lines/cm), selectable

Character Set:

full ASCII, 96 characters, both upper and lowercase

Line Width

136 characters, maximum with 10 pitch
(68 characters, expanded)
160 characters, maximum with 12 pitch
(80 characters, expanded)

Ribbon:

Nylon, double spool, reversable
1 1/2 in. (3.8 cm) wide
64 yd. (58.5 m) long

Switches/lamps:

ON/OFF, SELECT, PITCH, LINE/IN., LINE FEED, TOP OF FORM, CLEAR, FORMS OVERRIDE, paper out alarm and lamp, power on lamp, select lamp, and alarm tone.

Vertical Format Control:

3-channel, std - 1 inch (2.54 cm) tape. Vertical Tab, Top of Form, Page Eject.

Paper Size:

Maximum width	14.9 in. (37.8cm)
Minimum width	5.0 in. (12.7cm)
Paper width settings adjustable	
Maximum form length	11. in. (27.9cm)

Up to four co s plus original can be printed.

Cable:

6ft (1.8m) to power source
12ft (3.7m) to controller

Power Requirements:

115 or 230 VAC±10%
50 or 60 HZ±1 Hz
460 watts

Fuses:

5A (SB) for 115 VAC
2.5A (SB) for 230 VAC

Operating Environment:

50 to 90 F (10 to 32 C)

40% to 80% relative humidity, non-condensing, allowable

40% to 65% recommended

END