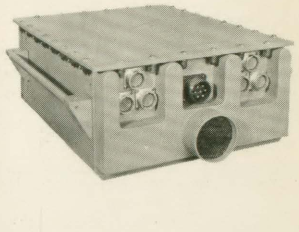
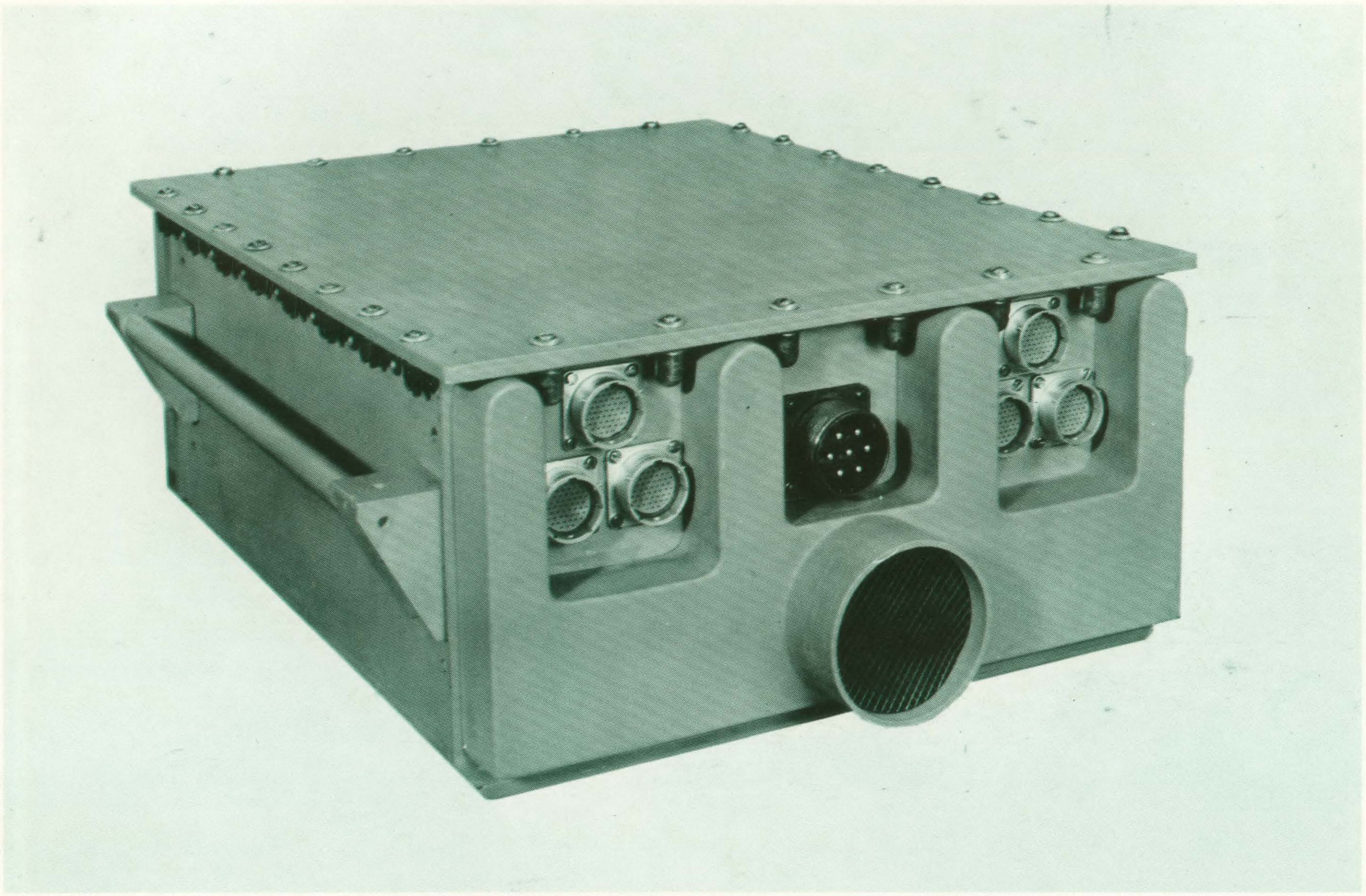
UNIVAC® CP-823/U

MILITARY COMPUTER



TECHNICAL DESCRIPTION

UNIVAC CP-823/U MILITARY COMPUTER



CENTRAL PROCESSOR UNIT

PHYSICAL SIZE AND WEIGHT

Height: 7.25 inches	Depth: 17.50 inches
Width: 14.50 inches	Weight: 50 pounds

UNIVAC CP-823/U

MILITARY COMPUTER

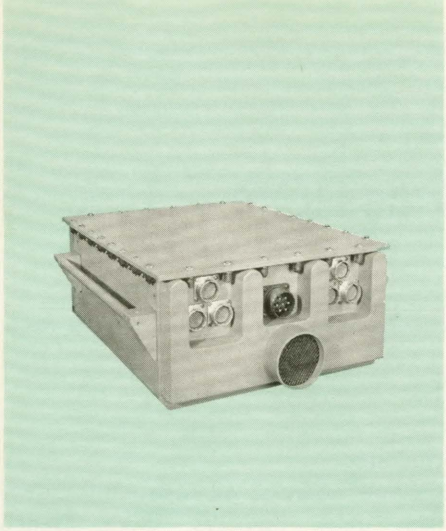
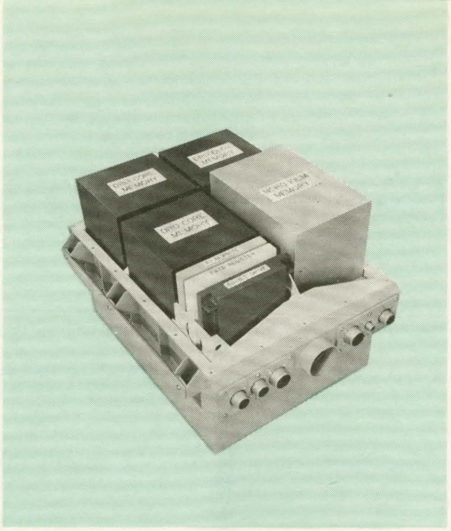
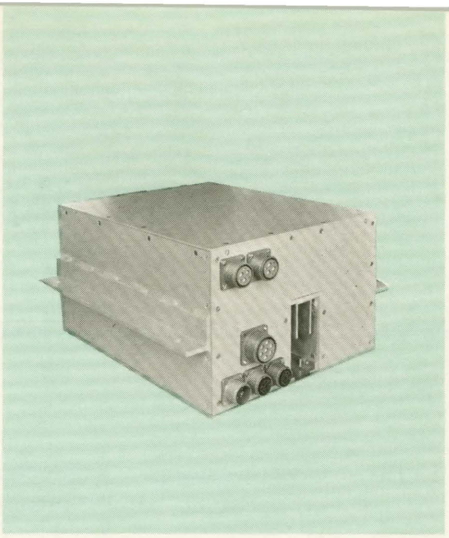
TECHNICAL DESCRIPTION

UNIVAC
DEFENSE SYSTEMS DIVISION
5316 W. IMPERIAL HWY.
LOS ANGELES, CALIF. 90045
678-2531

CONTENTS

ILLUSTRATIONS

TABLES

INTRODUCTION

The UNIVAC CP-823/U* Digital Computer is a miniaturized, real-time, general-purpose, stored-program computer with performance characteristics which exceed those of many large-scale ground machines. Because of the utilization of solid-state microelectronic circuitry, the CP-823/U is an ultra-reliable device well suited for applications which impose severe volume and weight limitations with a minimum power consumption. The CP-823/U computer was constructed in compliance with the exacting standards of MIL-E-5400F.

The concept of modularity, as it is featured in the computer, allows a selection of memory and input/output capacity as required for the application. Form factor flexibility is achieved by separately packaging the central processor unit, memory unit, power supply units, and the input/output units.

The computer memory unit consists of ferrite core modular arrays, each containing 4096 30-bit words. In addition, a 4096 30-bit word magnetic thin-film array is available when nondestructive readout is required. The memory unit may contain from one array to eight arrays to give a memory capacity of 4096 words (minimum) to 32,768 words (maximum). Current memory unit design provides for a memory composed of all ferrite core arrays or one thin-film array and up to seven ferrite core arrays. Cycle times for the core memory and read time for the film memory are four microseconds.

One of the most powerful and most flexible features of the CP-823/U computer is its input/output capability. Input/output to the central processor is provided over four input and four output channels and their associated control circuits. Each channel to the central processor may be further multiplexed into four additional channels by the use of a separately packaged input/output multiplexer unit. Present multiplexer units are designed and packaged to interface four input and four output channels with one of the four input channels and one of the four output channels to the central processor. By interconnecting four input/output units to the central processor, a total of 16 input and 16 output channels can be obtained.

Since the repertoire of instructions is identical with that of other UNIVAC computers, an entire library of programs is readily available. Not the least of the computer's attractions, however, is the direct compatibility with a wide range of existing peripheral equipment.

*CP-823/U is the military nomenclature for the UNIVAC 1830 Avionics Computer.

TECHNICAL CHARACTERISTICS

TYPE

TYPE: General-purpose, large-scale, real-time, microelectronic, parallel-binary.

MEMORY

Core Memory (DRO): Array of 4096-word, 30 bits magnetic core storage

Cycle Time:	4 microseconds
Capacity:	4096, 8192, 12,288, 16,384, 20,480, 24,576, 28,672, 32,768 30-bit words

Magnetic Thin-film Memory (NDRO): Array of 4096-word, thin-film storage

Cycle Time:	4 microseconds

INPUT/OUTPUT

MICROELECTRONIC MULTIPLEXER

Type:	Simplex, 30-bit parallel (Logical "0" is +3.0 volts, +1, —0.5) (Logical "1" is 0.0 volt ±0.5)
Number:	16 channels of input/output maximum, in modules of four
Transfer:	Output or input
Rate:	One output or input channel—62,000 words per second, maximum. Multi-channel—125,000 words per second
Features:	• Full buffer and/or real-time control on each channel • Intercomputer communication with any or all channels • External commands • Internal interrupt capabilities from buffer monitors on each channel • Channel "active" test

CONTROL

Command Structure:
Single address, 62 instructions, seven index registers, seven branch designators, and seven operand interpretation designators

Instruction Execution Time

Add:	8 microseconds
Subtract:	8 microseconds
Multiply:	32-48 microseconds
Divide/Square Root:	48 microseconds
Shift:	8-12 microseconds
Jump:	8 microseconds

Real-Time Clock:
Automatic internal clock controlled by main timing. Provision is made for the use of an external real-time clock source.

CONSTRUCTION

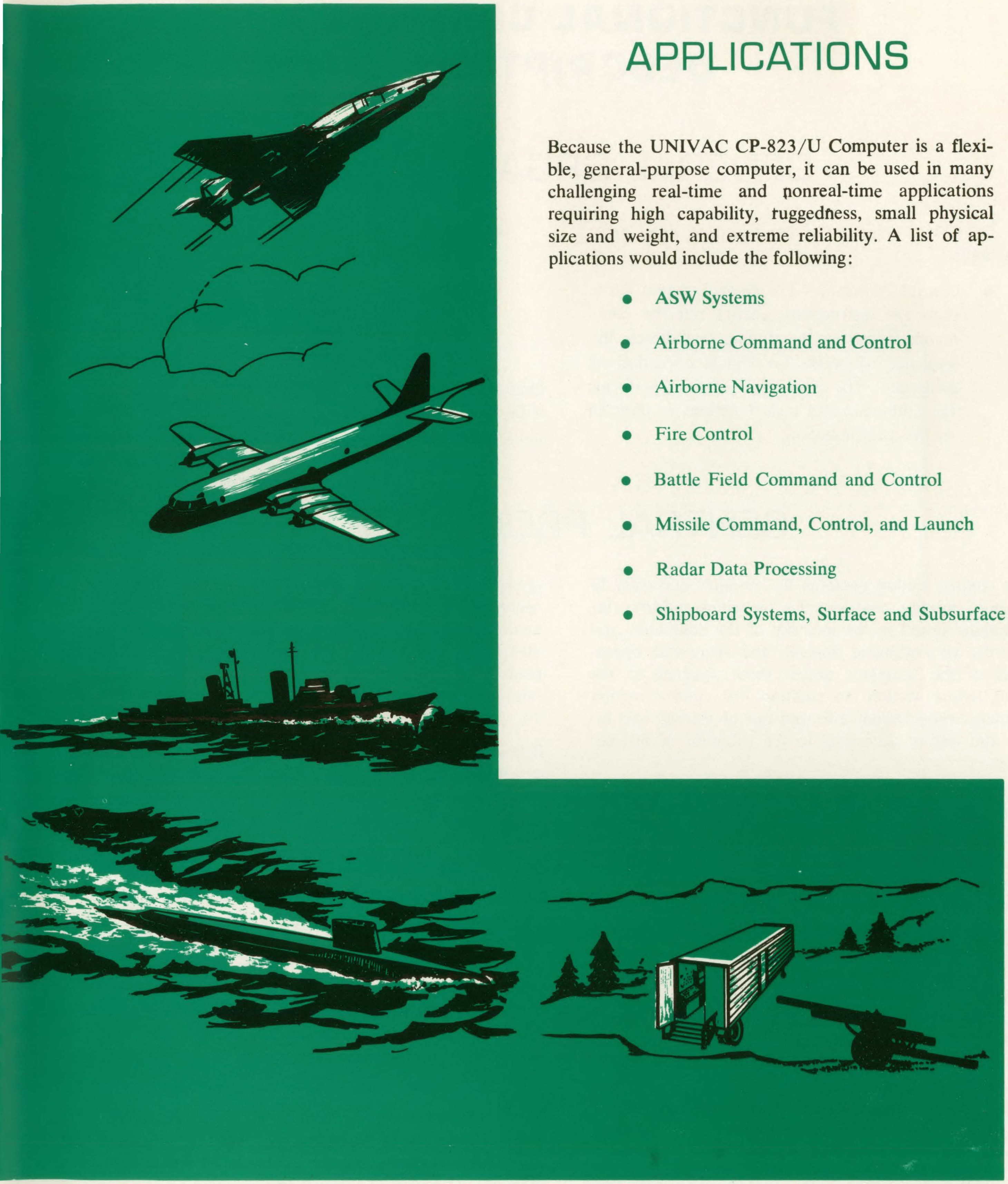
Modular Construction
Semiconductor Microelectronic Integrated Circuits

PRIMARY POWER REQUIREMENTS

Voltage:	200 Volts line to line
Phase:	3
Frequency:	400 cycles
Power:	830 Watts (32,768 words of memory, 16 channels input/output)

APPLICABLE MILITARY SPECIFICATIONS

MIL-E-5400F MIL-STD-704



APPLICATIONS

Because the UNIVAC CP-823/U Computer is a flexible, general-purpose computer, it can be used in many challenging real-time and nonreal-time applications requiring high capability, ruggedness, small physical size and weight, and extreme reliability. A list of applications would include the following:

- ASW Systems
- Airborne Command and Control
- Airborne Navigation
- Fire Control
- Battle Field Command and Control
- Missile Command, Control, and Launch
- Radar Data Processing
- Shipboard Systems, Surface and Subsurface

FUNCTIONAL COMPUTER DESCRIPTION

GENERAL ORGANIZATION

The CP-823/U computer is composed of three principal units:

- Central Processor—The *control* section interprets the instructions, carries out the commands, furnishes the timing, and directs the sequence of events for logical execution of programs. The *arithmetic* section performs the arithmetic and logic functions as directed by the *control* section.
- Memory—Provides random access storage of up to 32,768 30-bit words.
- Power Supply—provides regulated d-c power to operate the computer. Power supply units can be provided to match a variety of input and output power requirements.

Optional input/output units are available to interface with the computer through the four input and four output channels provided in the central processor.

CENTRAL PROCESSOR

The *control* section contains the circuitry necessary to procure, modify, and execute the single address instructions stored in the memory of the computer, and controls all arithmetic, logical, and sequential operations of the computer except those assigned to the *input/output* section. In addition, the *control* section regulates transmission into and out of storage and input and output according to the program of instructions.

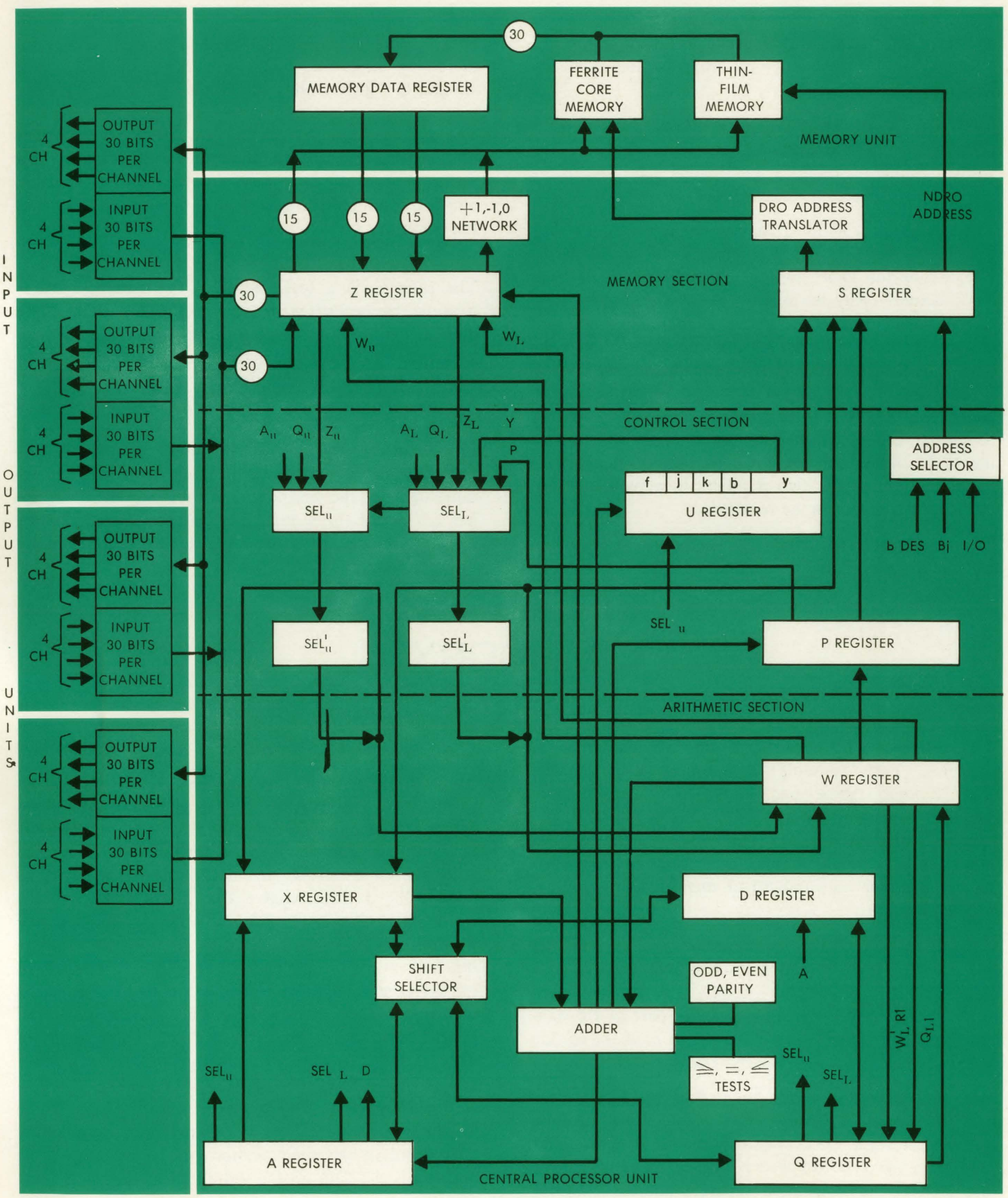
There are 62 basic instructions that may be used: each instruction contains a function code (f-6 bits), an instruction operand designator (y-15 bits), and three execution modifiers (j, k, b - 2, 3, or 4 bits). Execution modifiers provide for address incrementation, operand interpretation, branch-point designation, or input/output channel function. The operand designator may be increased by the amount contained in any one of seven index registers. The operand specified by the execution address may be interpreted as a 30-bit quantity or as a 15-bit upper or lower half-word with or without sign extension. The next sequential program step may be skipped unconditionally or when the branch-point designator places the skip under control of the contents of either the accumulator or the Q-register. (See Appendix A for a complete explanation of the instructions and their modifiers.)

The *arithmetic* section of the computer performs the operations of addition, subtraction, multiplication, division, shifting, comparison, and parity check. Arithmetic and logical operations are performed in the parallel binary mode. For most operations, the result appears in a 30-bit accumulator register. Arithmetic is one's complement subtractive, with a modulus (2^{30}-1).

Below is a description of the *control* section registers which are used for computer control and the *arithmetic* section registers used for numerical and logical calculations. (See Figure 1, Block Diagram of the Computer.)

The U-register or program control register is a 30-bit register that holds the instruction word during the execution of an operation. The function code and the various instruction designators are translated from the appropriate sections of this register. If an address modification is required before execution, the contents of the appropriate B-register are added to the lower order 15-bits of the U-register. The lower order 15-bits of the U-register may also be used directly as data, modulus 2^{15}-1.

The 15-bit registers, B_1 through B_7, store the quantities used for U_L modification. These B-registers, also called index registers, occupy the lower 15 bits of core memory addresses 161 through 167.



NOTE: Units enclosed within double lines are separately packaged.

Figure 1. CP-823/U Computer — Block Diagram with Full 16 Channels Input and 16 Channels Output

The P-register (15 bits) holds the memory address of a computer instruction word—that of the next instruction to be executed.

The K-registers (K_1, K_2, K_3) function as a shift counter for all arithmetic operations that involve shifts. Some instructions employing the K-registers are multiply, divide, and square root.

The A-register (30 bits) may be thought of, for programming purposes, as a conventional accumulator. Because of the logic employed, however, the A-register is actually only the main rank of the accumulator; the W-register serves as a second rank. The add operation is typical of the relationship between the A- and W-registers: the augend and addend are initially contained in the A- and W-registers respectively. Before the addition is performed, the contents of the A-register are transmitted to the X-register. The values of X and W are combined by the add network to form the sum of the two numbers in a parallel manner and then placed in the A-register.

The Q-register (30-bits) is used principally during multiply and divide operations. The contents of both A and Q may be shifted left or right, either individually or as one double-length, 60-bit word. The A- and Q-registers are addressable arithmetic registers.

The X-, D-, and W-registers are 30-bit, nonaddressable registers. These registers are used primarily for the exchange of data within the *arithmetic* section and for communicating with the remaining sections of the computer.

The selector is a logic network that controls transmissions to the *arithmetic* section and is used to place data in the various registers during control sequences. When data is read from memory or written into memory by execution of an instruction, the selector will translate the data read or stored based upon the k designator (see Appendix A for k designator usage). For arithmetic operations, the selector loads the W-register with data or its complement to satisfy the requirements of the subtractive arithmetic logic. The upper 15 bits of the U-register (U_u) are loaded directly from the selector as is the S-register for certain input/output operations.

MEMORY

The *memory* unit consists of up to eight memory modules, each module containing 4096 words. Two module types—one a nondestructive readout (NDRO) memory module and one a ferrite core destructive readout (DRO) module—may be used in combinations, which offer unusual flexibility when selecting the type and amount of memory for a specific application.

This discussion of the *memory* unit will include the Z-register, S-register and translator, data control circuits, and timing control circuits. The Z-register, S-register, and translator are physically part of the central processor, but are discussed in the *memory* section because they are most closely associated with it.

Each memory unit allows high-speed random access to 4096 words of storage with a cycle time of four microseconds for DRO memory and a 4-microsecond read time for NDRO. Every storage location (DRO and NDRO) is assigned a separate address (00000_8 through 77777_8), and each word in storage consists of 30 bits that can be divided into two 15-bit words: the upper 15 bits (M_U) and the lower 15 bits (M_L).

By means of programming and the use of the k designator, each 15-bit word can be handled separately. When reading an address, the whole 30-bit word is read into the Z-register (for core memory, Z is then written back to replace the data) and the selector, as defined by k, will control the transmission of the whole word (Z upper* [Z_u] and Z lower* [Z_L]) or half word (Z_u or Z_L) transmission to X or W. When storing half words into memory, only the half word of the DRO storage address that is not to be changed is read into Z from memory. The half word to be stored is transferred from W to Z and the whole word of Z is restored in core. (Example: Store A_L in M_U: $A_L \rightarrow SEL_L$, $SEL_L \rightarrow SEL_u$, $SEL_u \rightarrow W_u$, $M_L \rightarrow Z_L$, and $W_u \rightarrow Z_u$; $Z \rightarrow$ storage address). If a whole word is to be stored, computer control inhibits retaining the output of the read portion of the memory cycle and stores the data on the write half of the cycle.

When a specific storage location in memory is referenced, the S-register contains a 15-bit address word

*Z_u is used to denote the upper 15 bits, and Z_L is used to denote the lower 15 bits.

that specifies one of the 32,768 storage locations. The *control* and *input/output* sections of the computer have independent access to the storage registers through the use of the S-register and translator, the Z-register, and the memory timing sequence.

The Z-register holds the information to be read out or written into the *memory* unit. Data is normally transferred from the *input/output* section, and the W-register to the Z-register for storage in *memory*. Information read from memory is outputted from the Z-register to the *input/output* section, and to the various computer registers through the selector.

The address selector performs the task of translating designator I/O buffer requests and interrupt requests into the proper control word memory address location. After translation, the memory address is placed in S to allow the control word to be read from memory.

INPUT/OUTPUT

The input/output capabilities of the UNIVAC CP-823/U Computer will allow a variety of input/output units to be utilized. The microelectronic multiplexers have a maximum of 16 input and 16 output channels, each comprised of 30 bits, which provides for a maximum data transfer rate of 125,000 30-bit words per second. Each channel is equipped with four control lines to allow maximum reliable data transfer without monitoring or intervention from the central computer. Packaged modularly, each input/output unit contains four input and four output channels with associated control lines. Table 1 illustrates the control signals associated with each input/output channel, and the technical discussion that follows is based upon its characteristics.

TABLE 1. INPUT/OUTPUT CHANNEL CONTROL SIGNALS

FUNCTION	SIGNAL ORIGIN	
INTERRUPT ENABLE	COMPUTER	COMPUTER INPUT CHANNEL CONTROL LINES
INTERRUPT	PERIPHERAL	
INPUT DATA REQUEST	PERIPHERAL	
INPUT ACKNOWLEDGE	COMPUTER	
EXTERNAL FUNCTION REQUEST	PERIPHERAL	COMPUTER OUTPUT CHANNEL CONTROL LINES
EXTERNAL FUNCTION	COMPUTER	
OUTPUT DATA REQUEST	PERIPHERAL	
OUTPUT ACKNOWLEDGE	COMPUTER	

Through the use of an input/output buffer mode of operation, the CP-823/U has the capability of communicating with peripheral equipment on any and all input/output channels concurrently with the operating program. Buffering operation, once initiated by a programmed instruction, proceed to termination asynchronously with the program.

The CP-642 compatible input/output unit enables direct interfacing with existing peripheral equipments. A list of compatible equipment includes the following:

- Data Links
- Magnetic Tape Units
- Paper Tape (Punch and Flex) Units
- High-Speed Printers
- Video Processors
- Keyset Systems
- Displays
- Teletypewriting Equipment
- Magnetic Drums
- Disc Files
- Punch Card Equipment
- Plotter
- Analog-to-Digital and Digital-to-Analog Converters.

Other input/output units can be provided for special analog or digital applications.

SEQUENCE OF EVENTS

Normal output sequence for data transfer from computer to peripheral equipment (buffer mode) is as follows:

a. Computer initiates output buffer for given channel;
b. Peripheral equipment sets the Output Data Request line indicating that it is in a condition to accept data;
c. Computer detects Output Data Request;
d. Computer, at its convenience, places data on the 30 data lines;
e. Computer sets the Output Acknowledge line, indicating that the data is ready for sampling;
f. Peripheral equipment detects the Output Acknowledge;
g. Peripheral equipment may drop the Output Data Request any time after detecting the Output Acknowledge;
h. Peripheral equipment samples the 30 data lines;
i. Computer drops the Output Acknowledge and data lines.

Steps b through i of this sequence are repeated for every data word until the number of words specified in the output buffer have been transferred. Figure 2 shows the sequence for data transfer from computer to peripheral equipment.

The sequence for transmitting an External Function code from computer to peripheral equipment is as follows:

a. Peripheral equipment sets the External Function Request line when it is ready to accept an External Function word;
b. Computer detects the External Function Request;
c. Computer places the External Function word on the data lines (if its External Function buffer is active);
d. Computer sets the External Function line to indicate that the External Function word is ready for sampling;
e. The External Function clears the External Function Request at the peripheral equipment;
f. Peripheral equipment samples the External Function word;
g. Computer drops the External Function line and the External Function word.

This sequence is repeated when the peripheral equipment is ready to accept another External Function word (see Figure 3).

It is possible for the computer to send External Function commands to peripheral equipment without receiving an External Function Request signal. This operation is called External Function with force generation and is shown in Figure 4.

The computer places the External Function code on the 30 output data lines, and a minimum of 1.0 microsecond later energizes the External Function line. The External Function signal indicates to the peripheral equipment that an External Function word, which should now be sampled, is present on the data lines.

The peripheral equipment has no control over the rate at which External Functions with Force are sent. If the rate is too fast so that the peripheral equipments cannot accept External Functions, restrictions must

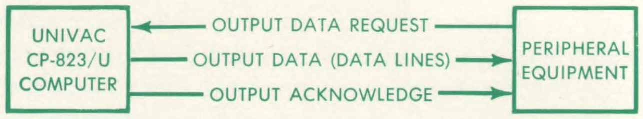


Figure 2. Data Transfer (Buffer Mode)

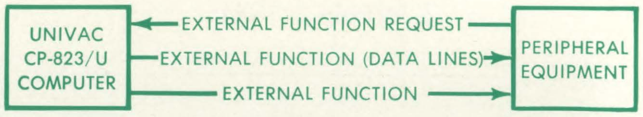


Figure 3. External Function Code Transmission

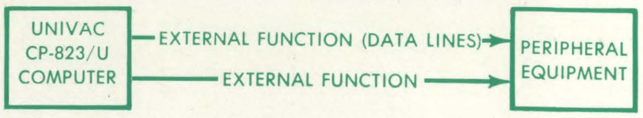


Figure 4. External Function Code Transmission with Force

be made in the programming of External Function instructions to the equipment. It should be noted that if the peripheral equipments have no provision for sending an External Function Request signal, all External Functions to existing peripheral equipment must be sent as External Function with Force.

Normal input sequence for data transfer to computer from peripheral equipment (buffer mode) is as follows:

a. Computer initiates input buffer for given channel;
b. Peripheral equipment places data on the 30 data lines;
c. Peripheral equipment sets the Input Data Request line to indicate that it has data ready for transmission;

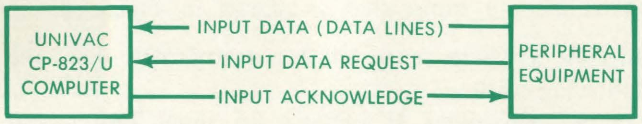


Figure 5. Input Data Transfer (Buffer Mode)

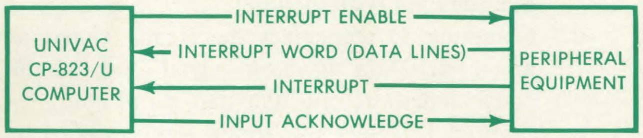


Figure 6. Interrupt Transmission

TABLE 2. INPUT/OUTPUT CHANNEL CONTROL SIGNALS FOR INTERCOMPUTER COMMUNICATION

FUNCTION	LINES
INTERRUPT ENABLE INTERRUPT INPUT DATA REQUEST INPUT ACKNOWLEDGE	COMPUTER INPUT CHANNEL CONTROL LINES
EXTERNAL FUNCTION REQUEST EXTERNAL FUNCTION READY RESUME	COMPUTER OUTPUT CHANNEL CONTROL LINES

d. Computer detects the Input Data Request;
e. Computer samples the 30 data lines, at its convenience;
f. Computer sets the Input Acknowledge line, indicating that it has sampled the data;
g. Peripheral equipment senses the Input Acknowledge line;
h. Peripheral equipment drops the data lines and the Input Data Request line.

Steps b through h of this sequence are repeated for every data word until the number of words specified in the input buffer have been transferred. Figure 5 shows the input data transfer to computer from peripheral equipment.

The sequence for transmitting an Interrupt from peripheral equipment to the computer is as follows:

a. Computer sets the Interrupt Enable when it is ready to accept an External Interrupt;
b. Peripheral equipment detects the Interrupt Enable;
c. Peripheral equipment places the Interrupt word on the 30 data lines;
d. Peripheral equipment sets the Interrupt line to indicate that the External Interrupt word is on the data lines;
e. Computer detects the Interrupt signal, and at its convenience, accepts the Interrupt word;
f. Computer drops the Interrupt Enable and sets the Input Acknowledge line;
g. Peripheral equipment detects the drop of the Interrupt Enable and clears the Interrupt line and data lines.

The Input Acknowledge of an interrupt will be initiated simultaneously with the clearing of the Interrupt Enable. The simultaneous occurrence of these conditions is used by peripheral equipment to differentiate between an Interrupt Acknowledge and a Data Acknowledge. An interrupt transfer is illustrated in Figure 6.

COMPUTER-TO-COMPUTER INTERFACE

It is possible for the CP-823/U computer to communicate with up to 16 other computers by converting any or all of the input/output channels. The control signals for intercomputer communication are shown in Table 2. Two cases of intercomputer communication are discussed. In both cases, computer A is transmitting to computer B.

In case 1 where computer A is transmitting a command word, the following occurs:

a. Computer B sets the Interrupt Enable when it is ready to accept a command word from computer A;
b. Computer A recognizes the Interrupt Enable as an External Function Request and places the External Function word on the data lines;
c. Computer A sets the External Function to indicate that the External Function word is on the data lines;

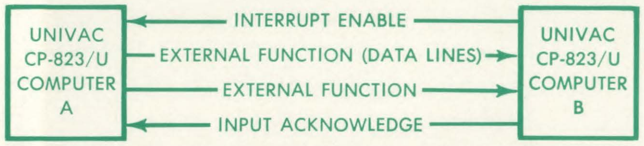

UNIVAC CP-823/U COMPUTER A
INTERRUPT ENABLE
EXTERNAL FUNCTION (DATA LINES)
EXTERNAL FUNCTION
INPUT ACKNOWLEDGE
UNIVAC CP-823/U COMPUTER B

Figure 7. Transmitting Command Word (Case 1)

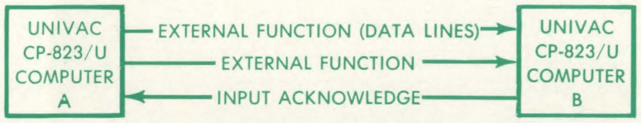

UNIVAC CP-823/U COMPUTER A
EXTERNAL FUNCTION (DATA LINES)
EXTERNAL FUNCTION
INPUT ACKNOWLEDGE
UNIVAC CP-823/U COMPUTER B

Figure 8. Transmitting Command Word by Force (Case 1)

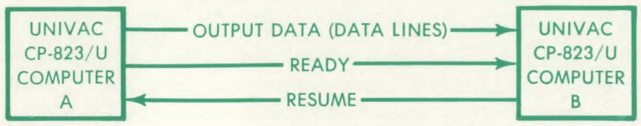


Figure 9. Transfer of Data (Buffer Mode) —Case 2

d. Computer B recognizes the External Function as an Interrupt and accepts the command word;
e. Computer B clears the Interrupt Enable line and sets the Input Acknowledge line;
f. Computer A recognizes the Input Acknowledge as a Resume and clears the External Function line.

In the event computer A sets the External Function line while the Interrupt Enable line is cleared (this is possible when an External Function with Force instruction is used), all communications on the associated group of output channels in A are suspended until computer B acknowledges receipt of the External Interrupt or until an intercomputer time-out interrupt in A permits A to resolve the problem. (See Figure 7.)

If either computer A or B is not designed for External Function Requests or Interrupt Enables, then all External Functions are transferred by force, and interrupts are transferred as shown in Figure 8.

In Case 2 where computer A is transferring data, the following occurs:

a. Computer B initiates an input buffer and computer A initiates an output buffer for the required channel;
b. Computer A places data on the data lines;
c. Computer A sets the Ready line to indicate that the data is on the line;
d. Computer B recognizes the Ready signal as an Input Data Request signal and, at its convenience, accepts the data word;
e. Computer B sets the Input Acknowledge;
f. Computer A recognizes the Input Acknowledge as a Resume signal and clears the Ready (Input Data Request) line and the data lines;
g. Repeat steps b through f for each word specified in the buffer. (See Figure 9.)

TIMING

Two types of input/output modules are available for the CP-823/U computer. One type matches the computer with existing peripheral equipments and utilizes a —15-volt interface. The other type of input/output unit uses a +4-volt interface.

Because both units operate in a similar manner, the following discussion covers only the microelectronic input/output unit.

INPUT TIMING CONSIDERATIONS

Data Input

The Input Data Request signal indicates to the computer that data has been placed on the 30 Input Data lines. To ensure that the data will be accepted, the Input Data Request must be maintained on the lines until an answering Input Acknowledge is received. As shown in Figure 10, there is an 8-microsecond minimum delay between the setting of the Input Data Request and its answering Input Acknowledge. There is no maximum limit stated for the delay since its value for any particular cycle is determined by the interaction with the computer program and the other input/output channels. The data lines must remain stable as long as the Input Data Request is set.

The Input Acknowledge indicates to peripheral equipment that the computer has sampled the 30 data lines. The Input Acknowledge signal is set for a fixed time interval. Peripheral equipment must be capable of detecting, as an Input Acknowledge, a signal which may exist in the stable "1" state for as little as 2.1 microseconds. The Input Data Request line and data lines may be dropped to the "O" state any time after detecting the Input Acknowledge. The Input Data Request cannot be reset immediately to indicate readiness of a second data word because the computer will not recognize the second Input Data Request unless a minimum time delay of 1.8 microseconds is allowed between the start of the dropping of the first Input Data Request and the start of the setting of the second Input Data Request. Timing would allow a peripheral equipment requiring a maximum data transmission rate to set the Input Data Request legitimately to the "1" state for the second time before the first Input Acknowledge has dropped to the "O" state. This will not affect operation of the cycle, however, since an 8-microsecond minimum delay will again be required between the setting of the second Input Data Request and the setting of the second Input Acknowledge.

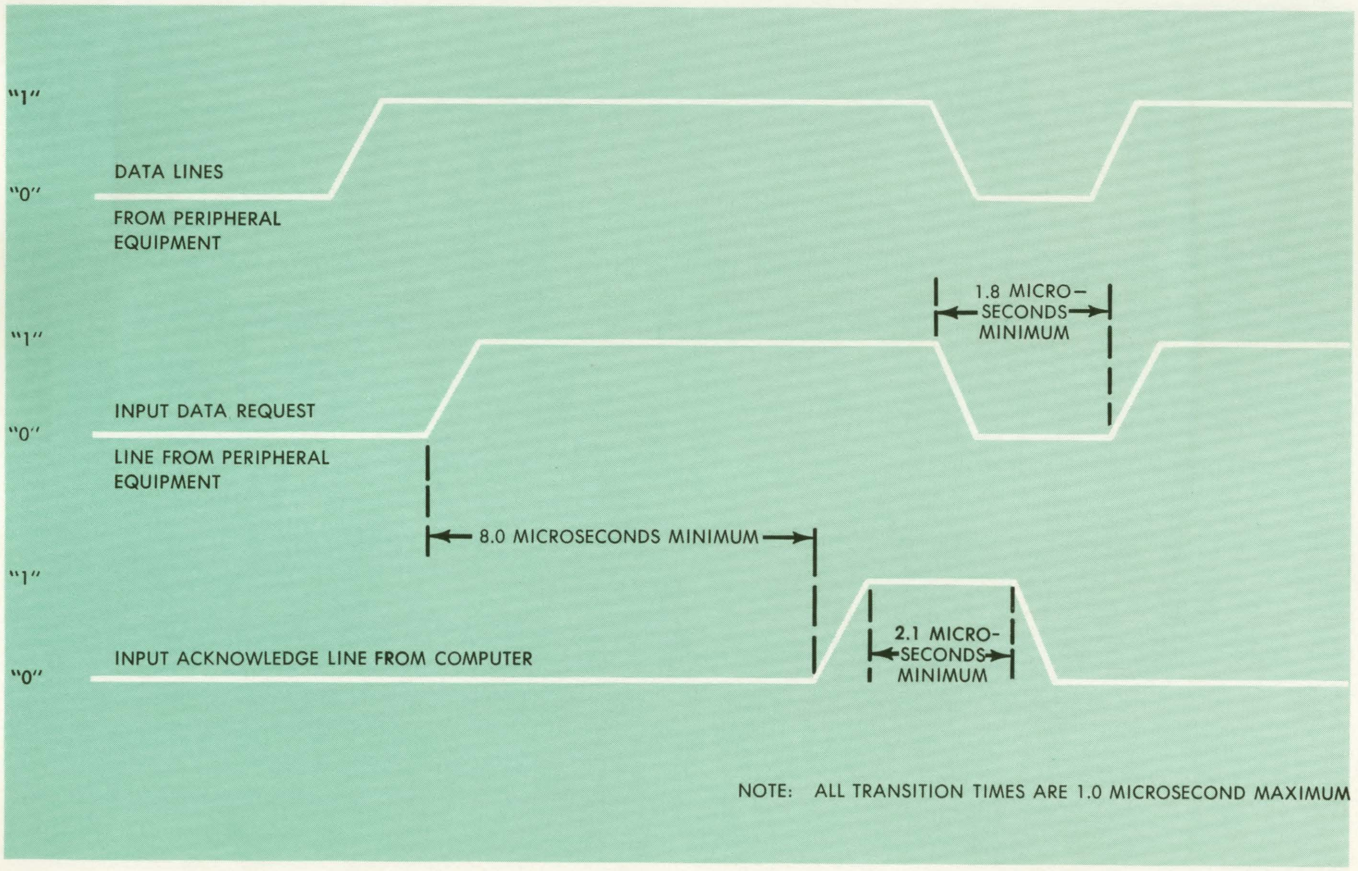

"1"
"0"
DATA LINES
FROM PERIPHERAL EQUIPMENT
1.8 MICRO-SECONDS MINIMUM
"1"
"0"
INPUT DATA REQUEST
LINE FROM PERIPHERAL EQUIPMENT
8.0 MICROSECONDS MINIMUM
"1"
"0"
INPUT ACKNOWLEDGE LINE FROM COMPUTER
2.1 MICRO-SECONDS MINIMUM
NOTE: ALL TRANSITION TIMES ARE 1.0 MICROSECOND MAXIMUM

Figure 10. Computer Data Inputs from Peripheral Equipment

Interrupt Inputs to the Computer
The Interrupt line indicates to the computer that the peripheral equipment has placed data on the 30 data lines. The computer then accepts the Interrupt word at its convenience. Simultaneously, the computer drops the Interrupt Request and sets the Input Acknowledge. Peripheral equipment can now drop the Interrupt signal and the Interrupt word. Data lines must remain stable as long as the Interrupt line is in the "set" condition. Figure 11 shows Interrupts to the computer.

OUTPUT TIMING CONSIDERATIONS

Data Output and External Functions
Peripheral equipment must first set the Output Request line or External Function Request line indicating that it is in a condition to accept a data or External Function word from the computer. Data lines will not necessarily be cleared to the "O" state before being reset to the "1" state. Minimum time interval between the Output or External Function Request signal and the placement of answering data on the lines is 8 microseconds. Maximum time interval depends upon the computer program, the priority of the particular channel, and the data request rates of the other peripheral equipment.

The Output Acknowledge or External Function signal indicates to the peripheral equipment that the re-

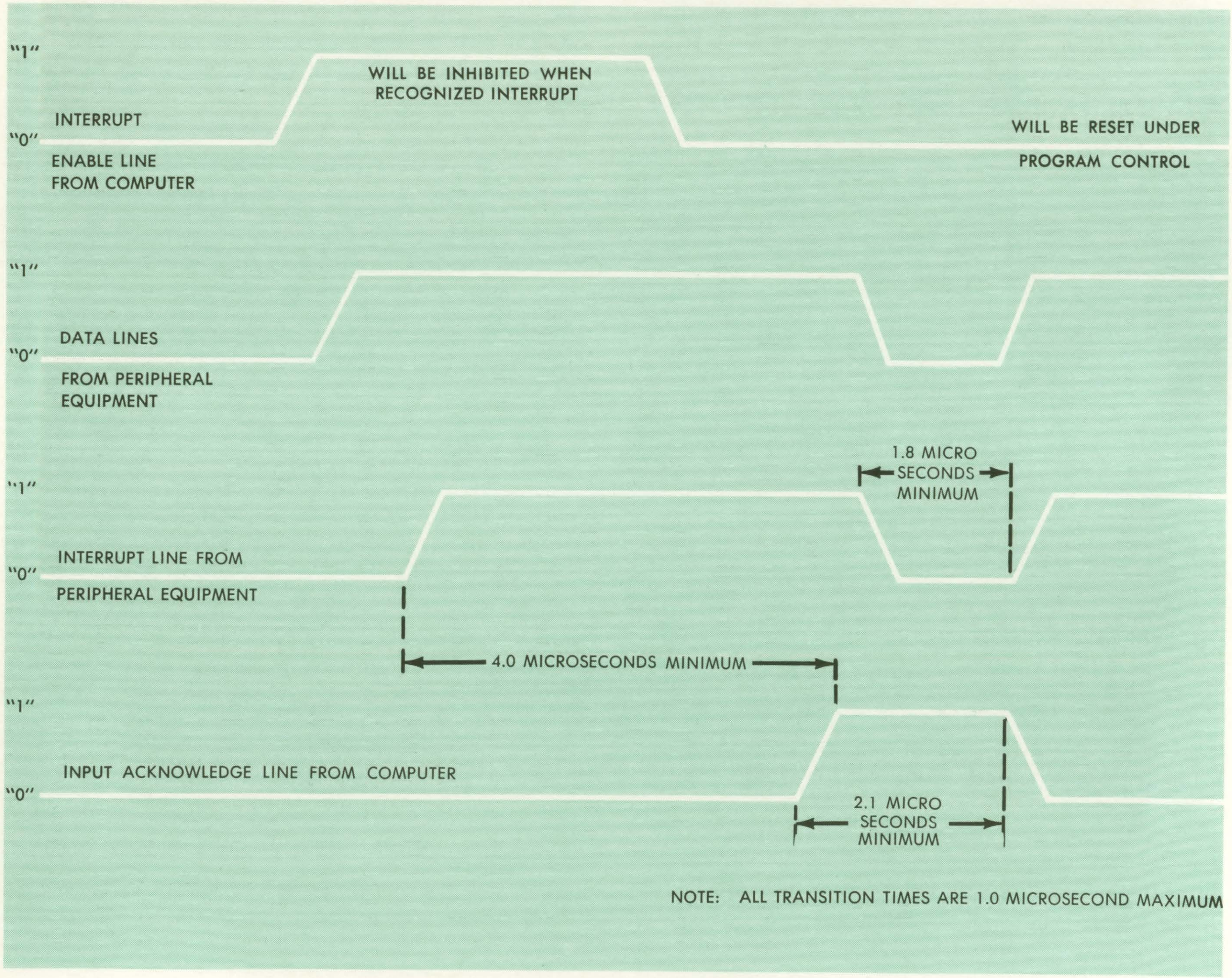

"1"
WILL BE INHIBITED WHEN RECOGNIZED INTERRUPT
INTERRUPT ENABLE LINE FROM COMPUTER
"0"
WILL BE RESET UNDER PROGRAM CONTROL
"1"
DATA LINES FROM PERIPHERAL EQUIPMENT
"0"
1.8 MICRO SECONDS MINIMUM
"1"
INTERRUPT LINE FROM PERIPHERAL EQUIPMENT
"0"
4.0 MICROSECONDS MINIMUM
"1"
INPUT ACKNOWLEDGE LINE FROM COMPUTER
"0"
2.1 MICRO SECONDS MINIMUM
NOTE: ALL TRANSITION TIMES ARE 1.0 MICROSECOND MAXIMUM

Figure 11. Interrupt Inputs to the Computer

quested word is now present on the data lines and that the lines should now be sampled. As shown in Figure 12, the Output Acknowledge or External Function signal will be sent a minimum of 1 microsecond after the data has been placed on the lines. The peripheral equipment must be capable of recognizing, as an Output Acknowledge or an External Function, a signal which may exist in the stable "1" state for as short a time as 2.1 microseconds.

The Output or External Function Request may be dropped to the "O" state any time after detecting the Output Acknowledge or the External Function. The Output or External Function Request cannot be reset immediately to indicate readiness of the peripheral equipment to accept a second word because the computer will not recognize the second Output or External Function Request unless a minimum time delay of 1.8 microseconds is allowed between the start of the dropping of the first Output or External Function Request and the start of the setting of the second Output or External Function Request. The timing would allow any peripheral equipment that wishes to receive data from the computer at a maximum rate to set legitimately the Output or External Function Request to the "1" state for the second time before the first Output Acknowledge or External Function has dropped to the "O" state. However, this will not affect operation of the cycle since an 8-microsecond minimum delay will again exist between the setting of the second Output or External Function Request and the availability of the data on the data lines.

External Functions with Force

External Functions with Force are unique since no request is sent by the peripheral equipment. The computer places the External Function code on the 30 output data lines and a minimum of 1.0 microsecond later energizes the External Function line. The External Function signal indicates to the peripheral equipment that an External Function word is present on the data lines. The External Function line will remain in the stable "1" state for an interval which may be as short as 2.1 microseconds. The External Function word will remain on the data lines for a minimum of 1.0 microsecond after the External Function signal begins to drop. Figure 13 is an External Function with Force timing diagram.

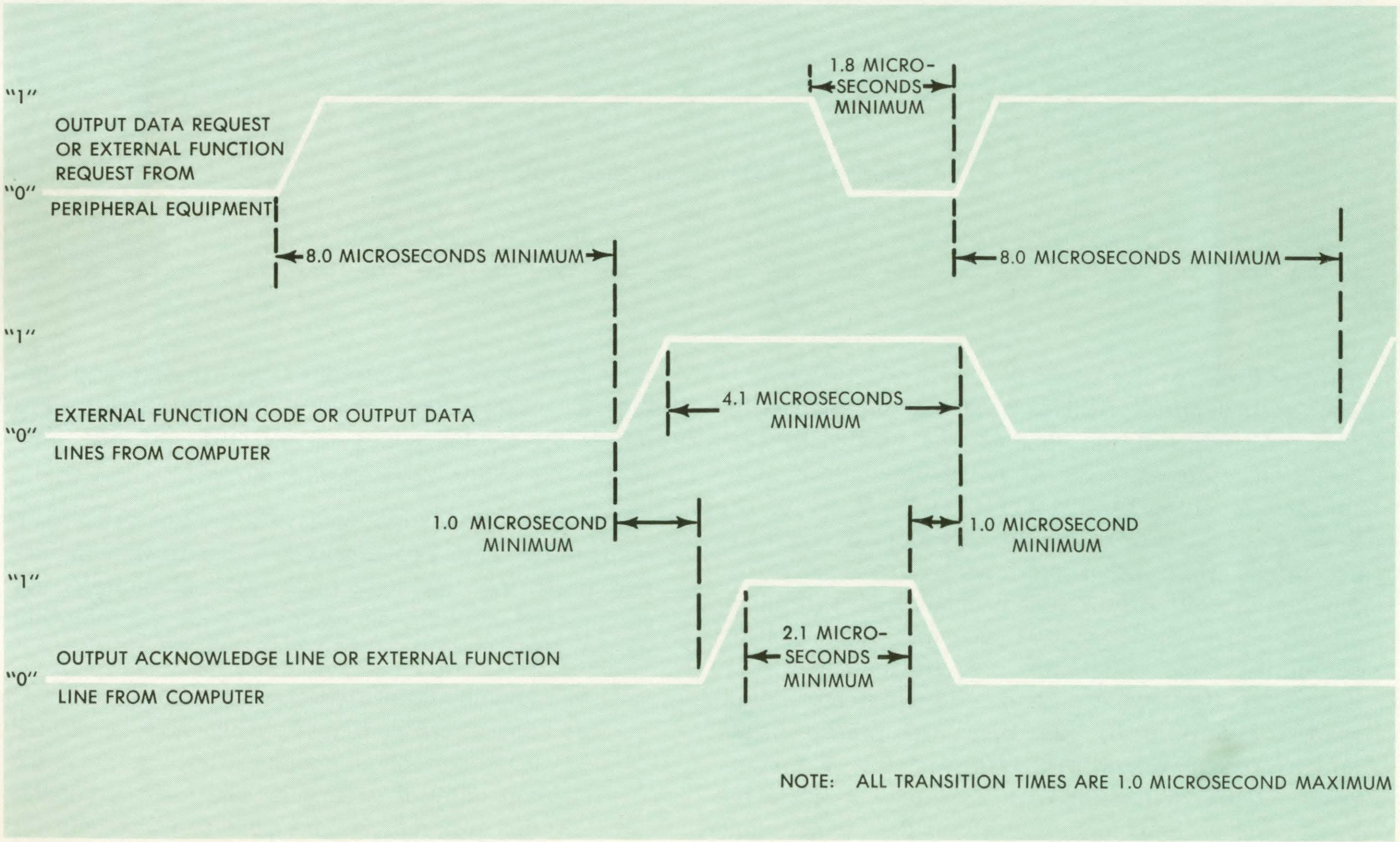

"1"
OUTPUT DATA REQUEST OR EXTERNAL FUNCTION REQUEST FROM PERIPHERAL EQUIPMENT
"0"
1.8 MICRO-SECONDS MINIMUM
8.0 MICROSECONDS MINIMUM
8.0 MICROSECONDS MINIMUM
"1"
EXTERNAL FUNCTION CODE OR OUTPUT DATA LINES FROM COMPUTER
"0"
4.1 MICROSECONDS MINIMUM
1.0 MICROSECOND MINIMUM
1.0 MICROSECOND MINIMUM
"1"
OUTPUT ACKNOWLEDGE LINE OR EXTERNAL FUNCTION LINE FROM COMPUTER
"0"
2.1 MICRO-SECONDS MINIMUM
NOTE: ALL TRANSITION TIMES ARE 1.0 MICROSECOND MAXIMUM

Figure 12. Data Outputs and External Functions from the Computer

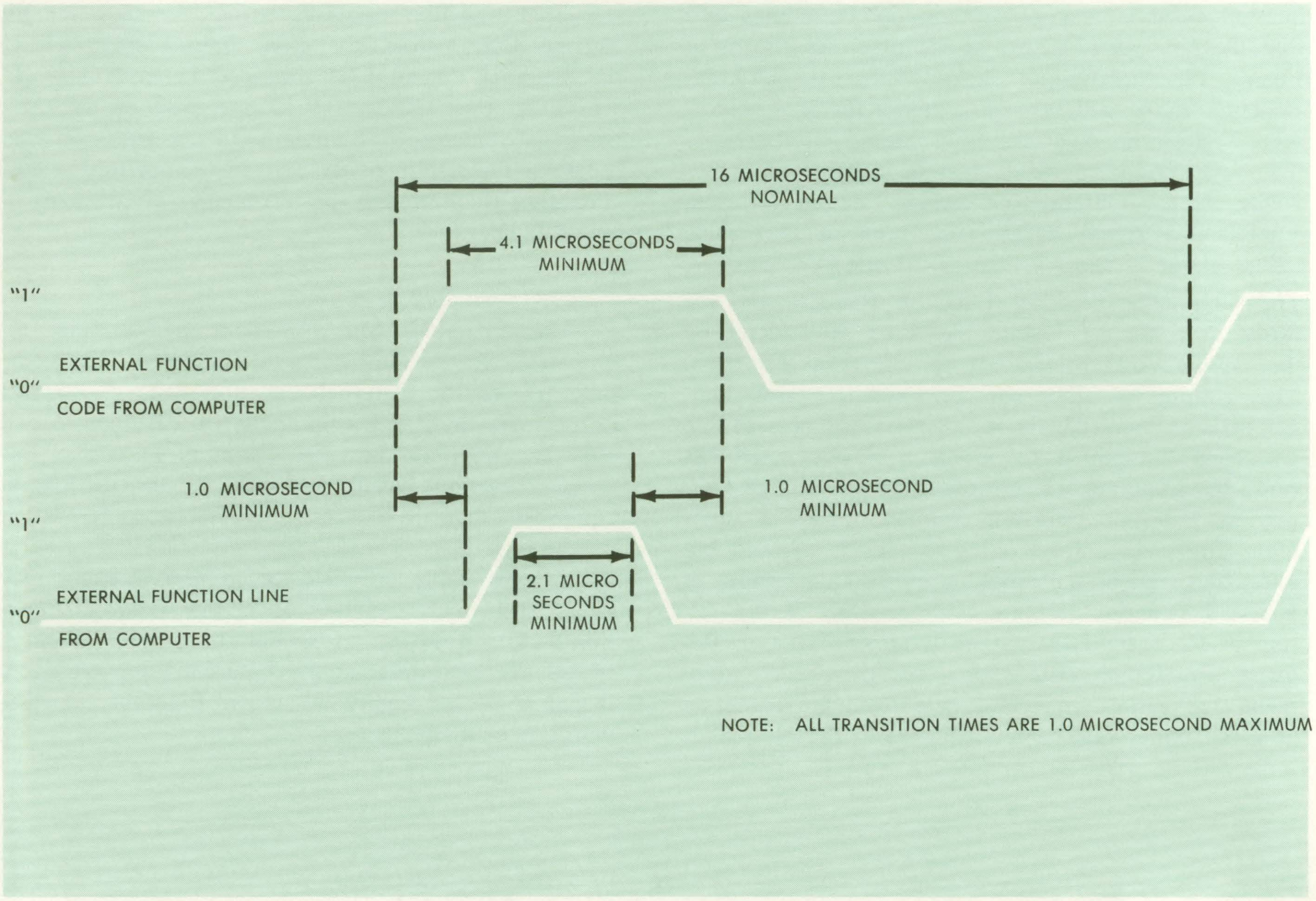


Figure 13. External Function with Force

Channel Priority

The channel priority will be such that the higher the numerical value of the channel, the greater its priority. Therefore, in order of descending priority, the channels are (15, 14, 13, 12) (11, 10, 9, 8) (7, 6, 5, 4) (3, 2, 1, and 0). Parentheses indicate the channel groups. Each group interfaces with a separate input/output unit. However, if a channel in a group is already active when a request occurs on another channel in the same group, this requesting channel may not be serviced next if a channel in a different group requests control at the same time. This priority method provides maximum I/O transfer rates. The computer requires 8 microseconds to service a single request from an input/output unit, but the input/output unit requires additional time to complete the data transfer. Therefore, rather than waiting for the input/output unit to accomplish the transfer and taking the next highest priority channel request in a group, the computer may service lower priority channels in other units as shown in Figure 14.

The previous discussion dealt with the microelectronic input/output unit. The larger CP-642 compatible input/output unit, designed to be used with existing peripheral equipments, is similar in operation with the following exceptions:

- Timing considerations are generally slower;
- The Interrupt Request signal may be deleted if the peripheral equipment does not have the capability of sensing it. The interrupt operation proceeds as in Figure 6, using only the Interrupt line, Input Acknowledge line, and Data lines.

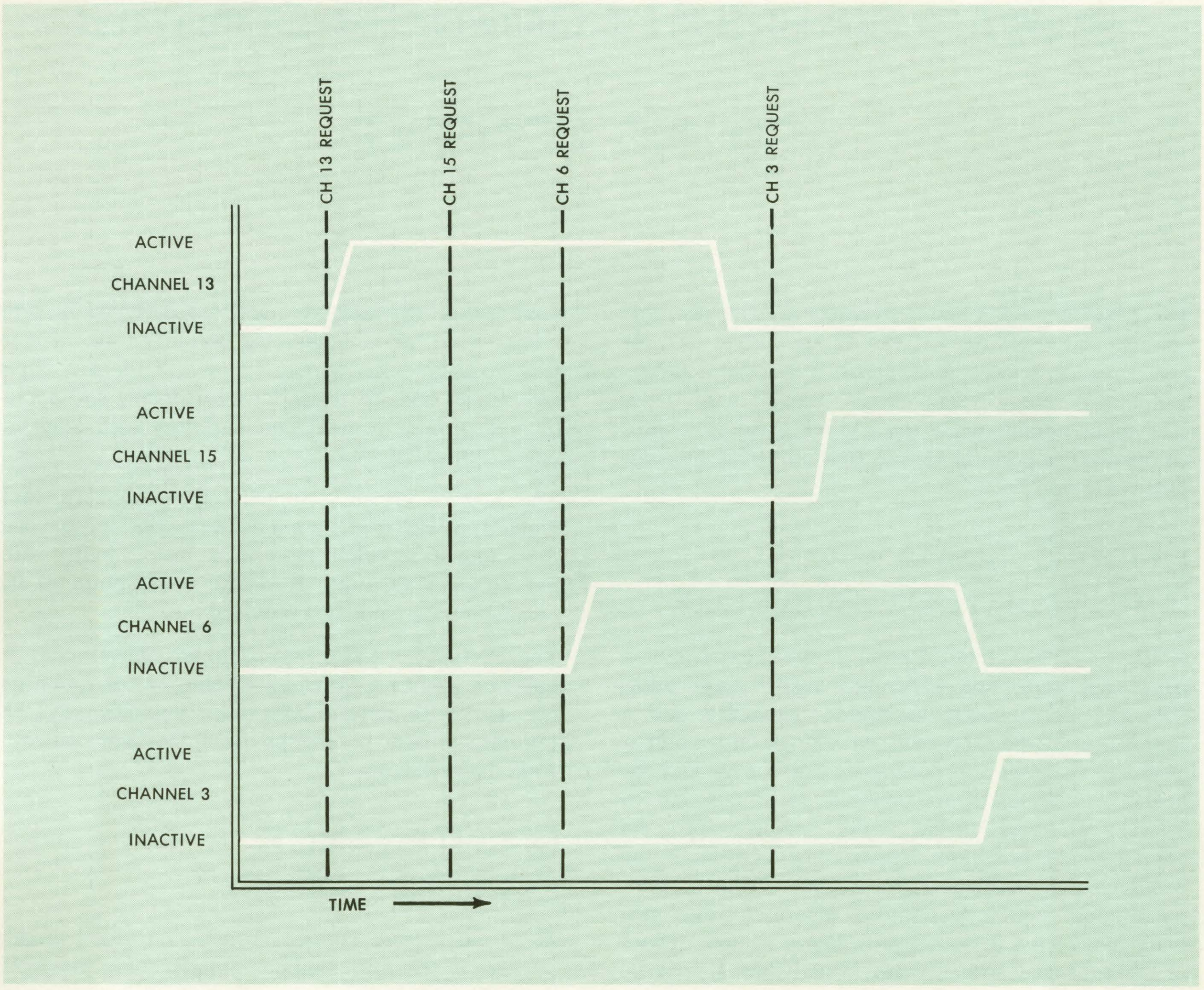


Figure 14. Channel Priority

POWER SUPPLY

Two power supply units are furnished with the CP-823/U computer: One furnishes power to the Central Processor and Memory Units, while the second furnishes power to from one to four input/output units.

Input power requirements vary depending upon computer configuration, but both power supplies operate from a 400-CPS, 3-phase, 200-volt, line-to-line source or a 400-CPS, 3-phase, 115-volt, line-to-line source. Both power supplies have been designed for d-c output voltages, at $\pm$ 5 percent, when the input power source conforms to MIL Standard 704. Power supply units can also be provided which operate on 60-cycle or 28 volt d-c service. The over-all efficiency of each power supply is greater than 75 percent at maximum input and maximum load conditions.

UNIVAC CP-823/U COMPUTER CONTROL GROUND CONSOLE

The UNIVAC CP-823/U Computer Ground Console contains the UNIVAC 1232A Input/Output Console and a maintenance/control panel. The 1232A Input/Output Console will provide the capability of paper tape input/output, while the control panel will allow an operator to perform various functions with the computer (Figure 15A).

UNIVAC 1232A INPUT/OUTPUT CONSOLE

UNIVAC 1232A Input/Output Console consists of a paper tape perforator, paper tape reader, page printer*, and alphanumeric keyboard* assembled into a compact unit which operates on a single input/output channel. Programs or program modification may be loaded by the reader on punched paper tape (5- to 8-level) prepared off-line under computer program control by the perforator. Alphanumeric entries to the computer may be made at the keyboard with or without printout. The page printer is also useful as a program monitoring device in providing a running record of real-time and normal program activities. It may be used with all UNIVAC general-purpose military computers.

FUNCTIONAL DESCRIPTION

The UNIVAC 1232A Input/Output Console operates under the control of the computer program. It accepts instructions from the computer program for any function or combination of functions assigned to it. The control circuitry translates the function word to initiate the proper mechanical and electronic response and executes the operations designated. Data are transferred in 6-bit frames from the alphanumeric keyboard to the computer and are accepted in 6-bit frames from the computer for output on the printer. Five to 8-bit frames may be used with the paper tape reader and the paper tape perforator, and operations may be performed simultaneously with other input or output functions.

The control circuitry supplies the necessary communication signals required between the console and the computer to complete an input/output buffer without program attention. Keyboard interruption of the computer program is also provided.

The UNIVAC 1232A Input/Output Console may be operated off-line to print information, to perforate paper tape, or to perform both tasks simultaneously. The information may be entered from the tape reader or from the keyboard.

MAINTENANCE CONTROL PANEL

The maintenance/control panel, on the ground console, is the man-computer interface for the UNIVAC CP-823/U Computer. The maintenance/control panel contains all controls and displays necessary to operate and maintain the CP-823/U computer. Figure 15B shows the current panel layout and all the controls and indicators on the panel (refer to section entitled "Central Processor.")

The indicators are neon lights that contain a set button to allow an operator to select manually a particular situation for the computer. During times of low-speed operation, or whenever the computer is stopped, the neon indicators give the operator a visual summary of the existing conditions of the computer program.

*The Keyboard and Printer are optional items.

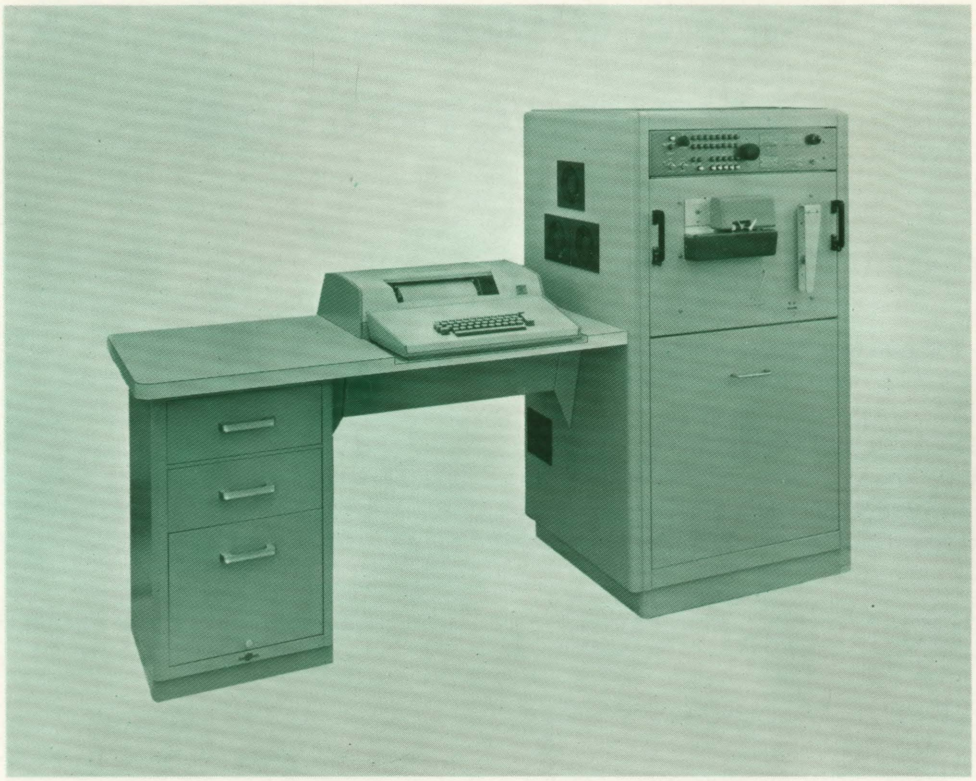
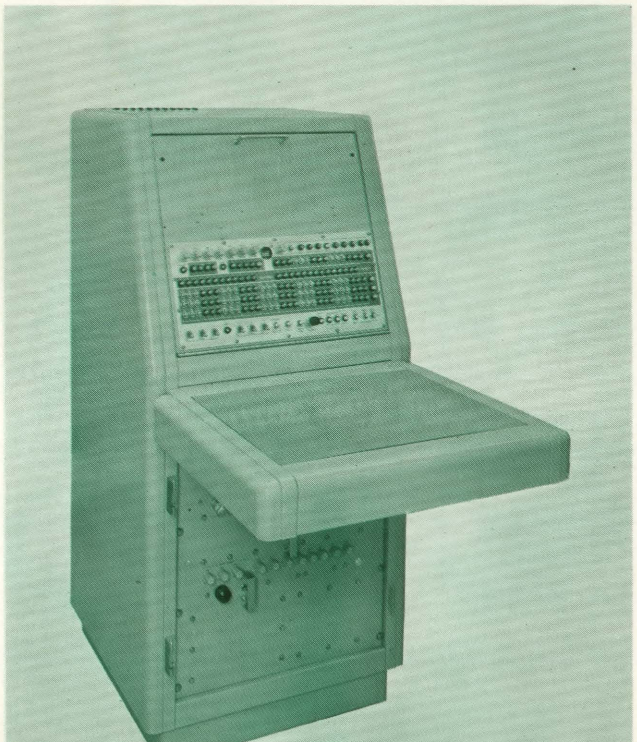


Figure 15A. UNIVAC 1232A Input/Output Console



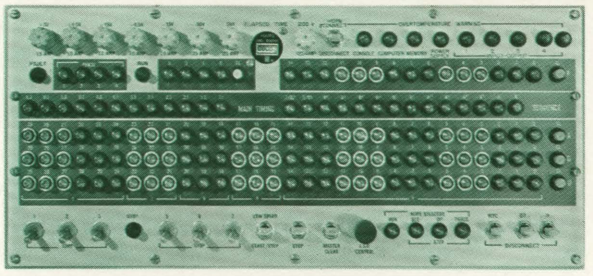

Maintenance/Control Panel

Figure 15B.
Maintenance/Control
Console

PHYSICAL CHARACTERISTICS AND ELECTRONICS

GENERAL

The CP-823/U computer operates over a temperature range of $-54°C$ to $+55°C$ and under extreme shock and vibrational stress as specified by the specification MIL-E-5400F. It is constructed of modularly discrete units (i.e., power supply, central processor, memory, and input/output) to allow the user to match the hardware to the application.

MICROELECTRONICS

The UNIVAC CP-823/U Computer is constructed primarily of integrated, monolithic semiconductor elements. Resistors, diodes, and transistors are contained within a single chip of silicon and interconnected by deposited aluminum conductors in monolithic semiconductor elements to form a NAND logic element (See Figure 16, logic element, and Figure 17, circuit schematic). Using integrated circuits obtains maximum benefits from the state-of-the-art techniques developed by the semiconductor industry to provide high reliability, small size, light weight, and low cost. The use of integrated semiconductor circuits eliminates most of the packaging of discrete elements now required in welded cordwood or soldered printed card circuits; however, discrete components are not completely eliminated. The number of steps required in manufacturing is reduced to about one-tenth of those required for the same circuit function using conventional components. Complex circuits are being mass produced through the use of automated manufacturing techniques which previously had been restricted exclusively to the manufacture of better transistors.

INTEGRATED CIRCUIT CONSTRUCTION

Microelectronic integrated computer circuits are subminiature in sizes, formed by selectively oxidizing and photo-etching 0.010-inch thick slices of silicon. The circuits are made on silicon chips small enough to be

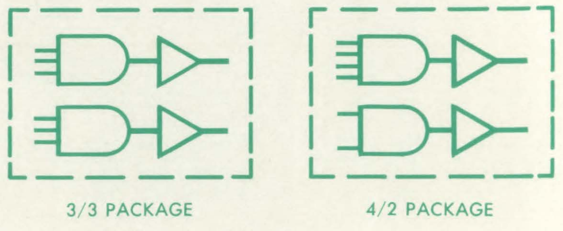


Figure 16. Basic Logic Circuit Packages

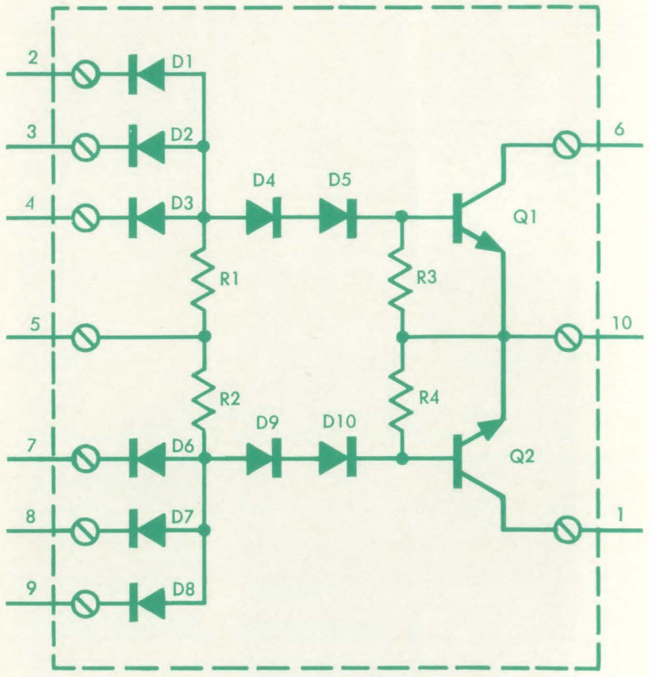


Figure 17. Schematic — Integrated Logic Element

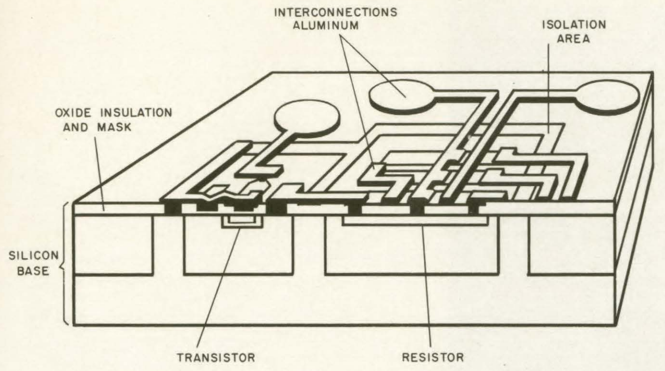


Figure 18. Integrated Circuit Cutaway

mounted inside 0.25 by 0.25-inch flat packages. Figure 18 depicts an integrated circuit cutaway showing the areas into which P or N type impurities have been diffused to dope the silicon crystal. A silicon-oxide coating is placed over the silicon chip, forming a glass resist to stop the diffusion of any impurities. The impurities can be selectively diffused through windows photo-etched in the oxide coating. First, the crystal is divided into discrete components by a diffusion through slots on the top surface, as shown in Figure 18. By growing additional oxide layers, etching new patterns through the oxide, and diffusing impurities through the etched windows, either a resistor, diode, or transistor component is diffused into each isolated area. This component diffusion process is of about the same complexity as the manufacturing process used for single planar transistors. When all components have been diffused into the silicon, an additional oxide layer is placed over the silicon slice, and windows are etched through the oxide where the intercomponent printed circuitry is required. An aluminum layer is then vacuum deposited onto the crystal, and the intercomponent printed circuitry is obtained by photo-etching the aluminum layer. Finally, external connections are made by thermocompression bonding gold wires from the component connection pads to the external leads. The diffused components, intercomponent printed circuitry, and external wire connections for the microelectronic circuit chip designed for a 0.25 by 0.25-inch package are shown in an enlarged view in Figures 19 and 20.

The approach to integrated circuit construction described previously features photo-etching techniques to obtain minimum size, lower costs, and higher reliability.

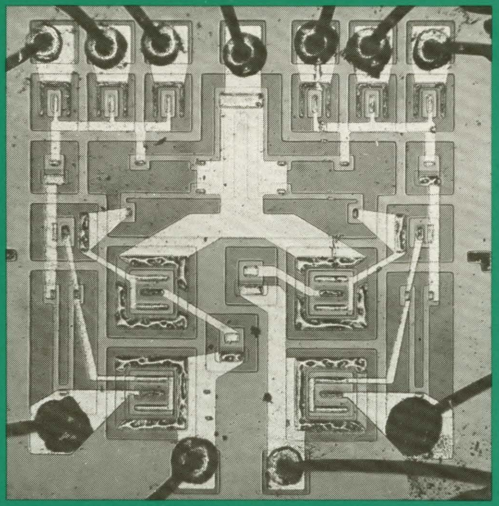


Figure19.

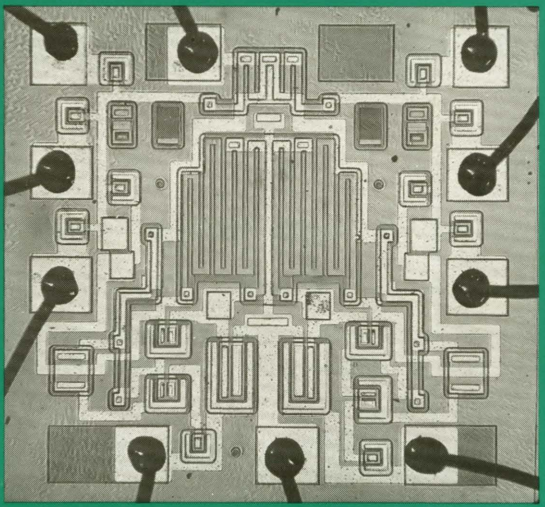


Figure 20.

Integrated Circuits — Enlarged View

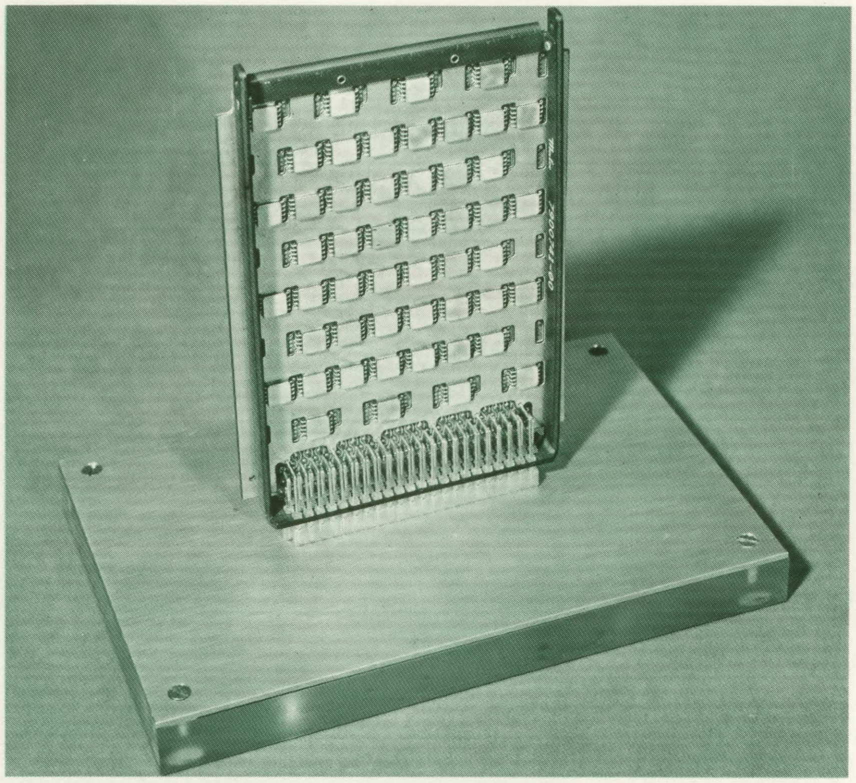


Figure 21.
Computer Logic Module

LOGIC MODULE CONSTRUCTION

The computer logic module (Figure 21) is a pluggable unit designed to mount and interconnect a maximum of 64 integrated circuit flat packs. It is comprised of a frame-connector assembly, two printed circuit boards, and an isolation board. Outside dimensions of the module measure 0.475 x 3.67 x 5.37 inches.

An integral part of the module frame, the 90-pin male connector provides for intermodule and power connections.

There are two double-sided printed circuit boards used on the module. One, of epoxy glass, is the interconnection board which is printed with horizontal circuit interconnection lines on one side and vertical lines on the other. The two sides are interconnected by plated through holes at the desired interconnections to provide a signal path grid work between individual circuits and between circuits and the connector. Circuit element leads and connector pins are also connected to the signal path grid work by means of plated through holes.

The second board provides for a ground, heat sink, and for voltage bussing. One side of the board is used as a ground plane for the circuit elements and is also used for a heat sink for conductive heat removal from the integrated circuit flat packs, which are mounted flat against it. The other side of the ground voltage board is used for bussing of the two voltages to the various circuits from the connector voltage pins.

These two printed circuit boards are mounted on both sides of a thin piece of insulating board. Flat packs are mounted on the ground side of the ground voltage board with their signal leads bent at 90° and passing through a clearance hole in the ground plane to contact an appropriate point on the signal interconnection board. Ground and voltage leads are terminated on the other board.

Although there are 64 spaces for integrated circuit flat packs on this module, several of the spaces are utilized by a combination resistor-capacitor flat pack, containing eight resistors, and one capacitor. The resistors are used as a collector load, where outputs of the integrated circuit chips are tied together into an "OR" circuit. The capacitor is used as a filter for the supply voltage.

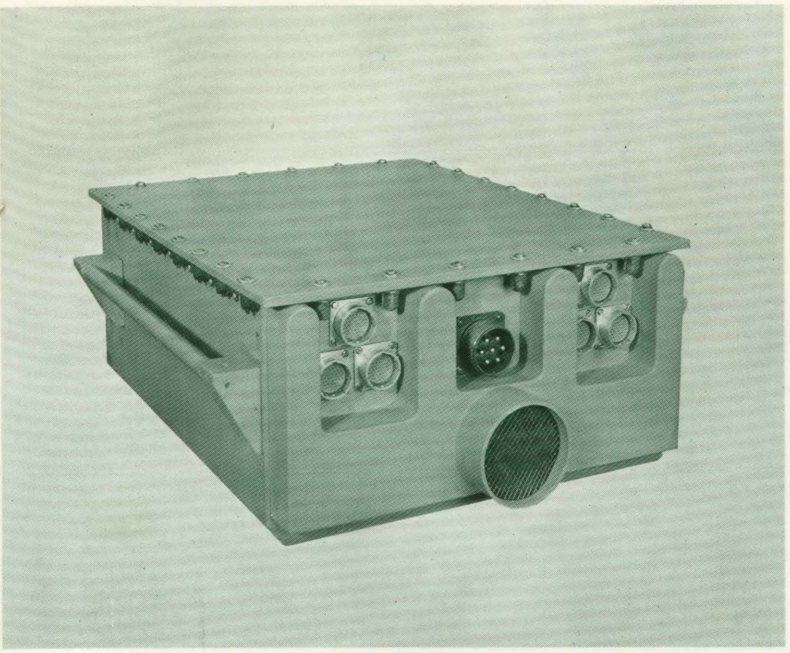


Figure 22. Central Processor

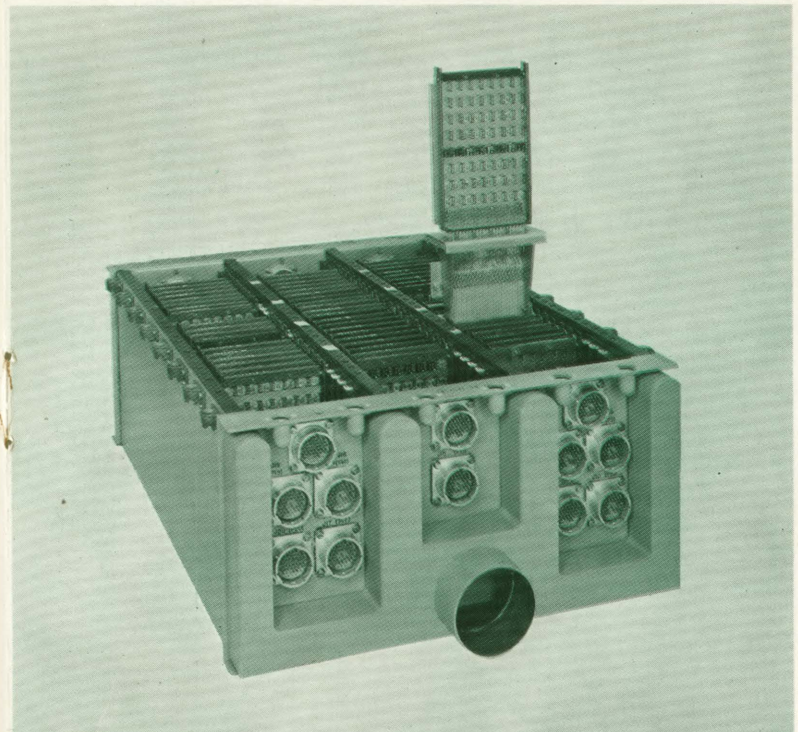


Figure 23. Central Processor with Module Extended—Inside View

CENTRAL PROCESSOR CONSTRUCTION

The central processor is contained within a case approximately 7 1/4 inches x 17 1/2 inches x 14 1/2 inches (see Figure 22). Within the case are spaces and connectors for up to 72 logic modules, heat exchangers and cooling ducts, and intralogic module chasis wiring (see Figure 23). The CP-823/U Central Processor utilizes 71 removable logic modules (see Figure 21). Each logic module fits into a heat conducting slot provided in the central processor and plugs into the associated connector. The back panel connector into which the modules are inserted is wired by means of a programmed, automatic wire-wrap machine. Since many logic modules of the same type are used within the central processor, the different logic functions depend upon connector wiring. When the logic modules are in place, two sides of the ground plane (heat sink), on the logic module, contact heat exchanger-support ducts to provide a physical path for heat conduction. The heat exchanger ducts have been designed to allow sufficient air flow to maintain proper circuit temperature. All circuits are cooled by conduction, and no air flows directly over the modules and circuits.

The top cover of the unit is secured to the frame with quick-disconnect type fasteners to allow easy access to the replaceable modules.

Connectors are located at each end of the unit to accommodate cables to the memory, input/output, and power supply units.

INPUT/OUTPUT CONSTRUCTION

Each 4-channel microelectronic input/output unit is similar to the central processor, except it is of smaller size. Case dimensions are approximately 7 1/4 x 13 x 14 1/2 inches. This unit has space for 42 plug-in modules and is, therefore, shorter over-all than the central processor. All other details of construction are similar to the central processor.

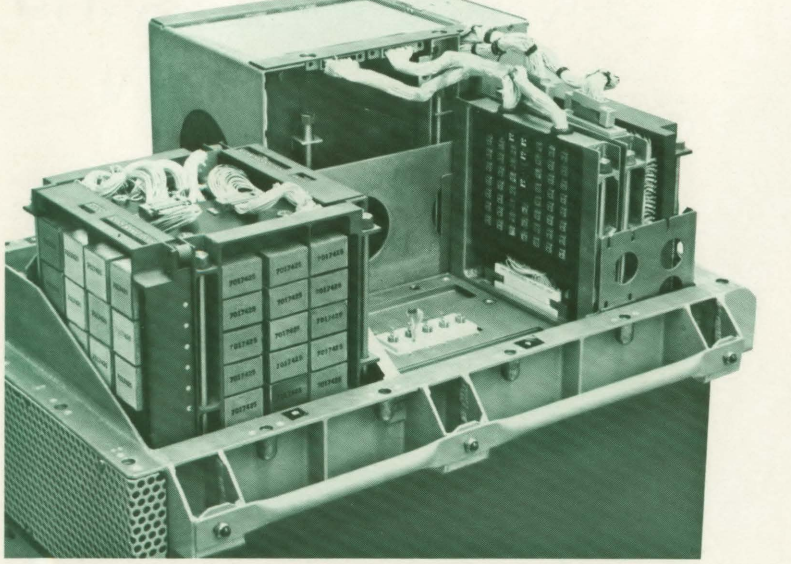


Figure 24. Memory Unit Showing 4K Thin-Film Stack and 4K Core Stack with Service Modules

MEMORY UNIT CONSTRUCTION

The memory unit (Figure 24) consists of removable 4096 word, 30-bit memory modules. Memory unit design provides for eight memory modules with one position designed to accept a 4096-word ferrite core (DRO) module or a 4096-word thin-film (NDRO) module. The entire memory unit is packaged in a case approximately 17.5 inches x 14 inches x 14 inches over-all. The interconnection panel for all the pluggable circuitry is in the center of the unit, with the plug-in memory modules inserted from each side. Modules are accessible by removing a cover on either side of the unit. The same type of air cooling is used as on the other units, with integral heat exchangers.

Memory electronics, which are common to all memory stack modules, data register, voltage regulator, film write, current regulator, stack select, and x-y address select, are contained on six pluggable service modules. Throw-away circuit blocks, similar to those discussed in the core memory section, contain most of the electronic components; some integrated circuit packages and discrete components are mounted directly on the service modules' printed circuit boards.

CORE MEMORY

Each core memory module consists of a 4096 30-bit word core stack and associated electronics packaged as a pluggable, modular unit. A core stack is composed of 30 double-window planes, with each plane containing 4096 cores. The stack-associated electronics, consisting of sense amplifiers, inhibit switch circuits, read and write transformers, and flat pack strobe gates are mounted on seven printed circuit boards. The read and write transformers and diodes are on boards which form the two ends of the core stack. The other five printed circuit boards are mounted in heat conducting frames for heat transfer to the air heat exchangers. These frame assemblies are mounted to the periphery of the core stack and wired to it, to form the complete 4096-stack module, pluggable to the back panel of the memory unit.

The sense amplifier and inhibit switch circuits are packaged in potted, individually pluggable circuit blocks of approximately 0.5 cubic inch each. The potted blocks are constructed of discrete components, mounted in cordwood fashion, and soldered to a flexible printed circuit board. This assembly is mounted to a connector and potted as a throw-away package. One circuit of each type (sense amplifier, inhibit switch, etc.) is contained within one potted block.

THIN-FILM MEMORY

Nondestructive readout memories are used when it is necessary to eliminate chance alteration of stored information or critical constants and to achieve higher operating speeds by eliminating the necessity of rewriting after each readout.

Basic to the NDRO memory is the Bicore* element which is used as the NDRO memory storage element. This memory element consists of a high-coercivity storage film and a low-coercivity readout film deposited in a multilayer fashion that physically appears to be a single film. The Bicore elements are deposited on a 3-mil glass substrate measuring 1.7 by 2.4 inches. The Bicore elements are 30 mils square and have a total thickness of 6000 angstroms. Each substrate has 960 elements deposited in a 30 by 32 array.

The NDRO memory plane is made up of two substrates holding 64 30-bit words, placed in a wiring array which contains the drive and sense conductors. The conductors are the same width as the film and are etched from 2.8-mil copper laminated to 1-mil Mylar film. Two conductor layers are used in the NDRO array.

A component board, which is the final level of address translation, completes the plane assembly. It consists of Permalloy** switch cores, microdiodes, and interconnection wiring. Memory associated electronics are packaged on circuit boards within the thin-film module.

INSTALLATION REQUIREMENTS

The computer has been designed to be cooled by air, with an operating temperature range of —54°C to +55°C. Cooling requirements may be found in Table 3. The operating temperature of the CP-642 compatible input/output unit is 0°C to +55°C.

Signal interconnection of the units comprising a CP-823/U computer system is accomplished through interconnecting cables terminated in connectors, which are plugged into mating receptacles at each unit.

When using the microelectronic input/output unit, the interface with peripheral equipment may be accomplished with up to 100 feet of shielded cable. The cable is composed of 42 twisted pair, terminated in a 92-pin connector. The CP-642 compatible input/output unit will interconnect with up to 300 feet of cable. Interconnecting cable for the CP-642 compatible input/output unit may be the same as that used for interconnecting the CP-642/USQ-20 with its peripheral equipment.

TABLE 3. SUMMARY OF EQUIPMENT CHARACTERISTICS

EQUIPMENT	WEIGHT LBS.	VOLUME CU. FT.	POWER REQUIREMENTS	LBS. OF AIR** PER HR.	COOLING REQUIREMENTS (CFM SEA LEVEL)
CENTRAL PROCESSOR	50	1.1	90	59	13
MEMORY 32 K CORE	90	2.1	125	81	18
MICROELECTRONIC INPUT/OUTPUT	35	0.80	105	69	17
CP-642 COMPATIBLE INPUT/OUTPUT*	60	1.65	200	130	30
CENTRAL POWER SUPPLY	50	.93	65	42	10
INPUT/OUTPUT POWER SUPPLY	50	.93	110	71	17

*Contains its own power supply.

**Based on the requirement of 65 pounds of air per hour per 100 watts power dissipation.

*Registered trademark of Sperry Rand Corporation.

**Registered trademark of Western Electric Company.

SOFTWARE

Experience with a multitude of digital systems has definitely demonstrated the need for support software that will expedite delivery of the initial program and provide for the rapid and orderly incorporation of programming changes as the system develops. Without such support software the initial programming and subsequent incorporation of program changes become the primary constraint in terms of initial system delivery and subsequent product improvement. As the operational programs required for the various systems become larger in terms of number of computer instructions, the need for support software becomes even more critical.

This consideration was instrumental in determining that the CP-823/U computer would be compatible with the CP-642A/USQ-20(V) and CP-642B/USQ-20(V) computers. Compatible in this case means that the three computers have similar instruction repertoire and input/output. Thus, without modification any program for the USQ-20 computers could be run on the UNIVAC CP-823/U Computer. The CP-823/U computer is also a software subset of the CP-667 in that any program written for it can be run on the CP-667 computer.

The advantages of this software compatibility are considerable: all software previously developed for the CP-642A/USQ-20(V) and CP-642B/USQ-20(V) is available for the CP-823/U computer. Further, any subsequent software development for any of these computers will be available to the CP-823/U. Such available software would not only cost several million dollars to develop but would require a lead time of several years.

A second advantage of selecting a compatible design is that a large pool of programmers and systems people exist both in the Navy and at UNIVAC who are familiar with the CP-823/U predecessors. Thus, it is not necessary to retrain personnel in the utilization and programming of the computer. The extensive system and software experience gained with the other computers is directly applicable to the utilization of the CP-823/U computer.

Another obvious advantage is that existing computers can be used for data reduction, analysis, program development, etc., pertaining to the CP-823/U computer. For example, CP-823/U programs could be directly checked out on the CP-642B/USQ-20(V) computer. It is not necessary to first develop an instruction simulation, which not only requires development time and money but would normally be very inefficient in its use of computer time and memory.

In addition to the obvious software advantages, the CP-823/U instruction and I/O design were chosen because of their proven excellence in solving a multitude of real-time and command and control problems.

With the selection of the compatible instruction repertoire and input/output, assemblers, compilers, debugging routines, utility packages, and subroutine libaries are available for the CP-823/U computer at no additional development cost.

ASSEMBLERS AND COMPILERS

AS-1 ASSEMBLER

The AS-1 language defines a problem in machine-oriented terminology. A thorough knowledge of the computer is necessary to use this language. There are four basic types of statements that can be used with this assembler:

- *Mono-operations,* which mnemonically express machine instructions;
- *Poly-operations,* which mnemonically express functions that give rise to more than one machine instruction;
- *Declarative operations,* which state control information to the assembler;
- *Debugging operations,* which help the programmer debug his program.

Inputs and outputs of the AS-1 assembler are limited to punched paper tape and on-line typewriter.

AS-1X ASSEMBLER

The AS-1 assembler has somewhat limited capabilities in processing large programs. AS-1X is an expanded version of AS-1 that will allow assembly of larger programs more quickly through the use of two magnetic tape transports and a high-speed printer.

The added equipment provides much greater versatility of outputs.

COMPILERS

CS-1 COMPILER

The CS-1 compiler was jointly developed by UNIVAC and the Navy, primarily for use in real-time command and control applications. It has been used for several years and is constantly being improved.

The CS-1 compiler is a program used as an aid in the preparation of other computer programs. It accepts information coded in an English-language, algebraic-type shorthand and produces a running computer program which is efficient from the standpoint of both time and memory space. Various debugging aids are an integral part of the compiler and significantly reduce the time normally required to debug the program. Program maintenance is simplified by a "librarian" which catalogues subroutines and data designs for subsequent retrieval.

Some typical features included in the CS-1 compiler are:

a. A common mnemonic source language. This language uses alphanumeric terms and subdivides into computer-oriented and problem-oriented operations;

b. An expandable source language. The compiler design provides the capability of making additions to the source language. More powerful operations may then be continuously devised to increase the usefulness of the system;

c. Both punched paper tape and punched card input are used with a uniform set of coding symbols;

d. Flexible memory assigning ability, for both object program positioning and data positioning in the computer;

e. A method of correcting the source programs;

f. Format error detection with printouts;

g. An integrated librarian. This librarian stores subroutines at the source language level. The subroutines are available for program inclusion when constructing a programming system;

h. Preparation of edited printouts of the program at various stages and memory address allocations;

i. Object program media option of paper tape, magnetic tape, or cards;

j. The ability to produce object programs for more than one computer version;

k. The ability to use AS-1 and AS-1X assembler coding as input.

COBAL

COBAL is primarily a business - oriented compiler designed to operate with a Real-Time Executive Routine (REX). REX controls, sequences, and provides for the efficient use of facilities for user programs.

DEBUGGING AIDS

CS-I DEBUGGING PACKAGE

CS-1 debugging aids provide a list of changes in the contents of computer words within a designated area of core storage during execution of the program being debugged. In addition, they make a non-zero dump of the designated area, or areas, at any point of the program during its execution. The areas to be checked and the place in the program where these are to be checked are designated by the debugging operations during the program compiling. The CS-1 debugging packages are then operated concurrently with the program to provide the desired debugged information. Flex code, field data code, or magnetic tape outputs can be selected from the debugging package.

UTILITY DEBUGGING PACKAGE

The Utility Debugging Package consists of a set of programs designed to assist the programmer to debug and check out his program. The programs are called up via the operator console and consist of the following routines:

- Module loader
- Paper tape handler
- Inspect and change memory addresses
- Store (Q) in memory
- Memory search
- Magnetic tape handler
- Magnetic tape editor/duplicator
- CS-1 magnetic tape output load
- Core dump on printer
- Keyboard entry
- Core dump on magnetic tape in printer format
- Teletype* handler

*Registered trademark of the Teletype Corporation.

TRACE DIAGNOSTIC

Trace is a diagnostic routine used for tracing the course of a program being executed. For each instruction traced, the contents of all appropriate registers and the instruction itself are recorded on magnetic tape for subsequent listing on a high-speed printer.

TALK

Besides the debugging operations available in CS-1, a real-time, data-extraction system has been developed to assist in debugging programs in terms of the high-level CS-1 language in which they were written. This language is TALK (Take a Look). Basically, this technique permits programmers to interrupt a program at various points to extract data which they have specified by TALK requests. Extensive editing is performed on the extracted data as a separate function to allow both real-time extractions and a flexible edited format for reading.

Some of the characteristics and capabilities of TALK are as follows:

a. The users program is written and compiled normally with no additional instructions generated;

b. The data to be extracted is specified by the same high-level language names as those used in the original source program;

c. TALK requests, which specify the data units to be extracted during execution of the program, are prepared and processed through a separate translator as a post compiling process. TALK requests may be updated or changed as the debugging progresses;

d. Real-time capabilities exist since data is merely extracted for later editing;

e. The edited output is an easily interpreted listing of data associated with its high-level source language identification. For instance, total tables can be listed by their high-level definitions. Decimal or octal may be specified;

f. Equipment flexibility exists for most environments so that on-site debugging can be achieved;

g. The TALK process does not necessarily have to be used as a debugging aid; it could be used as a data-extraction procedure with the user writing his own data-editing program.

CORNELL UTILITY PACKAGE

The Cornell Utility Package is a paper tape and magnetic tape handler. Paper tape operations handle bioctal, Flex code, and Teletype (TWX) code formats. The following programs are available as part of the Cornell Utility Package:

1. Paper Tape Operations
   - Bioctal, Flex, and TWX dumps with/without zero suppression
   - Bioctal, Flex, TWX and relative bioctal loads
   - Bioctal check-read with either Flex or TWX code punch tape output of discrepancies
   - Duplicate paper tape of any format
2. Magnetic Tape Operations
   - Write UNIVAC/dual format with/without check sum
   - Read UNIVAC/dual format with/without check sum
   - Position/rewind tapes
   - Read CS-1 machine language outputs
   - Dump area for off-line high-speed printer listing
3. Miscellaneous Operations
   - Fault condition display
   - External interrupt control
   - Inspect and change
   - Store Q
   - Memory search with either Flex or TWX code punch tape output of addresses of the found information
   - List area on on-line Teletype with/without zero suppression

SUBROUTINE LIBRARY

A vast subroutine library is available with the CP-823/U computer. The following is a representative list ot those subroutines. It should be noted that the list is not complete in that many more subroutines are available than those depicted.

FIXED POINT MATHEMATICAL ROUTINES

a. Sine
b. Cosine
c. Arcsine
d. Arccosine
e. Tangent
f. Arctangent
g. Arccotangent
h. Natural logarithm
i. Exponential
j. Stable Platform to Ships Deck Coordinate Conversion
k. Ships Deck to Stable Platform Coordinate Conversion

FLOATING POINT ARITHMETIC PACKAGE

a. Enter floating point accumulator
b. Floating point add
c. Floating point subtract
d. Floating point multiply
e. Floating point divide
f. Floating point store
g. Floating point compare
h. Fixed point to floating point conversion
i. Floating point to fixed point conversion

FLOATING POINT POWERS PACKAGE

a. Logarithm
b. Powers of e
c. Powers of N

FLOATING POINT TRIGONOMETRIC PACKAGE

a. Sine
b. Cosine
c. Tangent
d. Cotangent
e. Arcsine
f. Arccosine
g. Arctangent
h. Arccotangent

CONVERSION PROGRAMS

a. Field data decimal to octal
b. Octal to field data decimal
c. XS-3 to octal
d. Octal to XS-3
e. Field data to Flex/Flex to field data

SORT ROUTINES

a. Sort any number of 1-word items
b. Modified shell sort (sorting in small units and then merging) with bit capabilities.

DOUBLE PRECISION ARITHMETIC

a. Double precision addition
b. Double precision subtraction
c. Algebraic double precision addition
d. Algebraic double precision subtraction
e. Double precision multiplication
f. Double precision division

FIELD DATA ARITHMETIC

a. Field data addition
b. Field data subtraction

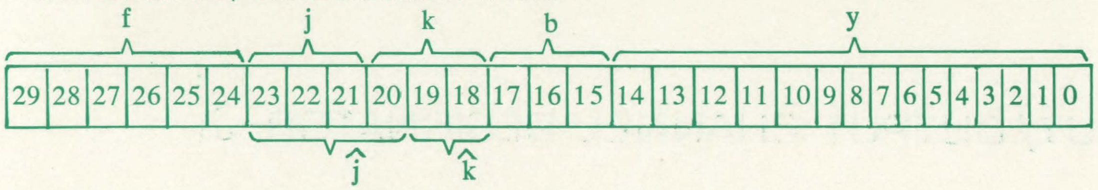
APPENDIX A

REPERTOIRE OF INSTRUCTIONS

The UNIVAC CP-823/U is a self-modifying, stored-program, one-address computer. Although this means that one reference is provided for the execution of an instruction, this reference can be modified automatically during a programmed sequence. Instructions are in the form of 30-bit instruction words stored sequentially in memory; the computer executes the instructions sequentially unless an instruction alters the sequence.

By means of five designators, a computer instruction word completely defines a computer operation. The highest order six bits constitute the function code designator, f. The balance of the upper half of the instruction word contains the designators j, k, and b. The lower half of the word contains the operand designator, y. The $\hat{j}$ and $\hat{k}$ are designators that pertain to input/output instructions.

UNIVAC CP-823/U INSTRUCTION WORD



FUNCTION CODE DESIGNATOR–F

The f designator appears in bit positions 29 through 24 of the U-register, or an instruction, designating the function to be performed by that instruction. Values 00_8 and 77_8 are fault conditions and if executed will cause a fault interrupt and a jump to the fault entrance address.

BRANCH CONDITION DESIGNATOR–J

The j designator appears in bit positions 23, 22, and 21 of the U-register, or an instruction; it is used in a majority of the instructions for jump and skip determination, for B-register specification, and for repeat status interpretation. Appropriate interpretations of the j designator are listed below and under the descriptions of the individual instructions.

For those instructions in which the j designator has no special interpretation, it specifies the condition under which the next sequential instruction in the program will be skipped. This permits branching from a sequence without executing a jump instruction, as would normally occur if a skip condition were not satisfied.

Skip of the next sequential instruction is determined by the following rules in all instructions except 04, 12, 13, 16, 17, 26, 27, 60 through 67, and 70 through 76:

j=0:	Execute the next instruction.
j=1:	Skip the next instruction.
j=2:	Skip the next instruction if (Q) is positive.
j=3:	Skip the next instruction if (Q) is negative.
j=4:	Skip the next instruction if (A) is zero (positive zero).
j=5:	Skip the next instruction if (A) is non-zero.
j=6.	Skip the next instruction if (A) is positive.
j=7:	Skip the next instruction if (A) is negative.

When the branch condition involves the sign of the quantity of A or Q, the evaluation examines the sign bit of these quantities: hence, positive zero is considered a positive quantity, and negative zero is considered a negative quantity. A negative zero, however, that is generated as a result of an arithmetic operation is normally forced to a positive zero.

INPUT/OUTPUT CHANNEL DESIGNATOR-$\hat{j}$

The $\hat{j}$ designator appears in bit positions 23, 22, 21, and 20 of the U-register, or an input/output instruction, specifying the C-channel for the instruction. Thus the $\hat{j}$ provides accessibility to the 16 (decimal) input/output channels numbered $0\text{-}17_8$. Instructions 13, 17, 62, 66, 67, 73, 74, 75, and 76 use the $\hat{j}$ designator configuration.

OPERAND INTERPRETATION DESIGNATOR-K OR $\hat{K}$

A k designator (3 bits) appears in bit positions 20, 19, and 18 of the U-register, or an instruction; a $\hat{k}$ designator appears only in bit positions 19 or 18 since bit 20 is a portion of the $\hat{j}$ designator. Instructions 13, 17, 62, 63, 66, 67, and 73 through 76 use the $\hat{k}$ designator configuration since they perform input/output activities and require $\hat{j}$ designator for channel specification. The k designator controls both operand and instruction interpretation. The $\hat{k}$ designator is explained under the descriptions of the input/output related instructions.

The k designator controls operand interpretation. Those instructions that read an operand, but do not replace it after the arithmetic is performed are designated read instructions. Those instructions that do not read an operand but store it are designated store instructions. Instructions which both read and store operands are classified as replace instructions.

The various values of k affect the operands (except where noted otherwise under individual instruction description) as follows:

(a) Read Instructions (01-12, 20-23, 26, 27, 30, 31, 40-43, 50-53, 60-65, 70-72):

k=0: Y_u=0's; Y_L=Y

k=1: Y_u=0's; Y_L=$(Y)_L$

k=2: Y_u=O's; Y_L=$(Y)_u$

k=3: Y=(Y)

k=4: Y_u=All bits same as Y_{14}; Y_L=Y

k=5: Y_u=All bits same as Y_{14}; Y_L=$(Y)_L$

k=6: Y_u=Same bits as Y_{29}; Y_L=$(Y)_u$

k=7: Y=(A)

For instructions 22, 52, and 53, k=7 is not used.

(b) Store instructions (14-16, 32, 33, 47):

k=0: Store operand in Q*.

k=1: Store operand_L in Y_L, leaving $(Y)_u$ undisturbed.

k=2: Store operand_L in Y_u, leaving $(Y)_L$ undisturbed.

k=3: Store operand in Y.

k=4: Store operand in the A-register**.

k=5: Store complement of operand_L in Y_L, leaving $(Y)_u$ undisturbed.

k=6: Store complement of operand_L in Y_u, leaving $(Y)_L$ undisturbed.

k=7: Store the complement of operand in Y (storing the complement of B^j is the same as storing a 30-bit register complemented.)

NOTE: When storing B^j, the B-register is tested as a 30-bit register, with the upper 15 bits set to 0. With a k=7, store B^j will complement as a 30-bit register ($(Y)_u$=77777).

(c) Replace instructions (24, 25, 34-37, 44-46, 54-57):

k=0: Not used.

k=1: Read portion—Y_u=0's; Y_L=$(Y)_L$.

Store portion—Store operand_L in Y_L leaving $(Y)_u$ undisturbed.

k=2: Read portion—Y_u=0's; Y_L=$(Y)_u$.

Store portion—Store operand_L in Y_u leaving $(Y)_L$ undisturbed.

k=3: Read portion—Y=(Y)

*A 1400000000 instruction complements (Q)

**A 1504000000 instruction complements (A)

Store portion—Stores operand in Y.

k=4: Not used.

k=5: Read portion—Y_U=All bits same as Y_{14}; $Y_L=(Y)_u$

Store portion—Stores operand_L in Y_L leaving $(Y)_u$ undisturbed.

k=6: Read portion—Y_u=all bits same as Y_{29}; $Y_L=(Y)_u$

Store portion—Stores operand_L in Y_u leaving $(Y)_L$ undisturbed.

k=7: Not used.

NOTE: Repeat instructions require special interpretation when followed by a replace instruction.

INDEX DESIGNATOR–B

The B designator appears in bit positions 17, 16, 15 of the U-register, or an instruction and specifies which of the B-index registers, if any, will be used to modify the operand address designator, y, to form $Y=y+(B^b)$. This operation employs an additive accumulator; hence, a quantity consisting of all zeros cannot result unless the bits of both the operand designator, y, and (B^b) are all zeros. Use of a B-index register to modify the operand designator, y, to form $Y=y+(B^b)$ will cause the higher order 15 bits of the B-register memory address to be made zero.

The effect of the various values of the b designator is a 15-bit one's complement sum as follows:

b=0:	Do not modify y.
b=1:	Add (B^1) to y.
b=2:	Add (B^2) to y.
b=3:	Add (B^3) to y.
b=4:	Add (B^4) to y.
b=5:	Add (B^5) to y.
*b=6:	Add (B^6) to y.
b=7:	Add (B^7) to y.

OPERAND DESIGNATOR–Y

The y designator appears in bit positions 14 through 0 of an instruction. The operand or address of an operand, Y, is related to y through the expression $Y=y+(B^b)$.

*A repeat instruction makes special use of B-register 6 when followed by a replace instruction.

LEGEND

Symbol	*Represents*
a	A-register (A, Q, B^n), a memory location Y, or a constant
(a)	Content of a
$(a)_i$	Initial content of a
$(a)_f$	Final content of a
a_n	The n^{th} bit of a
$(a)_n$	The n^{th} bit of the content of a
f	Function code designator
j	Branch condition designator
$\hat{j}$	Input/output channel designator
k	Operand interpretation designator
$\hat{k}$	Operand interpretation designator
b	Index designator
y	Operand designator
$\not{Y}$	Operand (regardless of source)
$\underline{Y}$	$y+(B^b)$ The operand or address for the Read portion of an instruction. The destination address for the Store portion of an instruction.
$(\underline{Y})$	Content of memory address $\underline{Y}$
L $\not{Y}(Q)$	Bit-by-bit multiplication, logical multiply of $\not{Y}_n$, and $(Q)_n$
A	A-register or accumulator (30-bit arithmetic register)
B	Seven B-registers (15 bits each). B-registers are address-modifying registers generally used for indexing loops in a program; in addition, B^7 serves as a repeat counter. (The address modification does not alter the instructions as stored in memory.) The j or the b designator specifies the B-register used. The B-registers are located in seven DRO memory locations.
Q	Q-register (30-bit arithmetic register)
U	U-register (30 bits). The U-register holds the instruction word during execution of an operation. If address modification is required before execution, the appropriate B-register content is added to the lower order 15 bits of the U-register before execution.
P	P-register (15 bits). The P-register is the program address register. This register holds the address of the next instruction throughout the program except for jump instructions where the P-register is cleared and the new program address is entered.
C	The 16 input/output channels (30 data lines each). Channels consist of transmission lines; therefore, they cannot be considered registers. The designator $\hat{j}$ specifies (in octal) the channel used.

TABLE A-1. REPERTOIRE OF INSTRUCTIONS

CODE (OCTAL)	INSTRUCTION	NORMAL EXECUTION TIME μs
00	(Fault Interrupt)	
01	Right Shift Q	8-12
02	Right Shift A	8-12
03	Right Shift AQ	8-12
04	Compare	8
05	Left Shift Q	8-12
06	Left Shift A	8-12
07	Left Shift AQ	8-12
10	Enter Q	8
11	Enter A	8
12	Enter B^n	12
13	External Function on C^n	8
14	Store Q	8
15	Store A	8
16	Store B^n	12
17	Store C^n or Test EFB	4-12
20	Add A	8
21	Subtract A	8
22	Multiply	32-48
23	Divide/Square Root	48
24	Replace A+Y	12
25	Replace A−Y	12
26	Add Q	8
27	Subtract Q	8
30	Enter Y+Q	8
31	Enter Y−Q	8
32	Store A+Q	12
33	Store A−Q	12
34	Replace Y+Q	12
35	Replace Y−Q	12
36	Replace Y+1	12
37	Replace Y−1	12
40	Enter Logical Product	8
41	Add Logical Product	8
42	Subtract Logical Product	8
43	Compare Mask	8
44	Replace Logical Product	12
45	Replace A+Logical Product	12
46	Replace A−Logical Product	12
47	Store Logical Product	8
50	Selective Set	8
51	Selective Complement	8
52	Selective Clear	8
53	Selective Substitute	8
54	Replace Selective Set	12
55	Replace Selective Complement	12
56	Replace Selective Clear	12
57	Replace Selective Substitute	12
60	Jump (Arithmetic)	8
61	Jump (Manual)	8
62	Jump on C^n Active Input Buffer	8
63	Jump on C^n Active Output Buffer	8
64	Return Jump (Arithmetic)	12
65	Return Jump (Manual)	12
66	Terminate C^n Input Buffer or Enable, Disable Interrupts	4
67	Terminate C^n Output Buffer, all Buffers, or Terminate C^n External Function Buffer	4
70	Repeat	12
71	B Skip on B^n	12
72	B Jump on B^n	12
73	Input Buffer on C^n (without monitor mode)	12
74	Output Buffer on C^n (without monitor mode)	12
75	Input Buffer on C^n (with monitor mode)	12
76	Output Buffer on C^n (with monitor mode)	12
77	(Fault Interrupt)	

TABLE A-2. MEMORY ALLOCATIONS

TYPE OF MEMORY	OCTAL ADDRESS RANGE	USE
Core Memory	00000	Unassigned
	00001-00017	Unassigned
	00020-00037	External interrupt entrance address (one per input channel)
	00040-00057	Input monitor interrupt entrance address (one per input channel)
	00060-00077	Output monitor interrupt entrance address (one per output channel)
	00100-00117	Input buffer control words (one per input channel)
	00120-00137	Output buffer control words (one per output channel)
	00140-00157	External function buffer control word (one per output channel)
	00160	Real-time clock
	00161-00167	B-boxes
	00170-00477	Unassigned
	00500-00517	External function buffer monitor interrupt entrance address (one per output channel)
	00520-00537	Interrupt word storage address (one per input channel)
	00540-00577	Unassigned
	00600-00617	Intercomputer time out interrupt entrance address (one per output channel)
	00620-67777	Unassigned
Film Memory	70000	Bootstrap entrance address
	70001	Power restart entrance address
	70002-77777	Unassigned

LIST OF INSTRUCTIONS

01 RIGHT SHIFT Q

This instruction causes an open-ended shift of (Q) to the right Ɏ bit positions. The higher order bits are replaced with the original sign bit as the word is shifted. Only the lower order six bits of Ɏ are recognized for this instruction.

02 RIGHT SHIFT A

This instruction causes an open-ended shift of (A) to the right Ɏ bit positions. The higher order bits are replaced with the original sign bit as the word is shifted. Only the lower order der six bits of Y are recognized for this instruction.

03 RIGHT SHIFT AQ

This instruction causes an open-ended shift of (A) and (Q) as one 60-bit register to the right Ɏ bit positions. The lower order bits of A are shifted into the higher order bit positions of Q. The higher order bits of A are replaced with the original sign bit as the word is shifted. Only the lower order six bits of Ɏ are recognized for this instruction.

04 COMPARE

This instruction compares the signed value of Ɏ with the signed value of (A) and/or (Q). It does not alter either (A) or (Q). The branch condition designator, j, is interpreted in a special way for this instruction as listed below:

j=0:	Execute the next instruction.
j=1:	Skip the next instruction.
j=2:	Skip the next instruction if Ɏ is less than, or equal to, (Q).
j=3:	Skip the next instruction if Y is greater than (Q).
j=4:	Skip the next instruction if (Q) is greater than, or equal to Y, and Y is greater than (A).
j=5:	Skip the next instruction if Y is greater than (Q), or if Y is less than, or equal to (A).
j=6:	Skip the next instruction if Y is less than, or equal to (A).
j=7:	Skip the next instruction if Y is greater than (A).

05 LEFT SHIFT Q

This instruction shifts (Q) circularly to the left Ɏ bit positions. The lower order bits are replaced with the higher order bits as the word is shifted. Only the lower order six bits of Ɏ are recognized for this instruction. Maximum shift count permitted is 59 places.

06 LEFT SHIFT A

This instruction shifts (A) circularly to the left Ɏ bit positions. The lower order bits are replaced with the higher order bits as the word is shifted. Only the lower order six bits of Ɏ are recognized for this instruction. Maximum shift count permitted is 59 places.

07 LEFT SHIFT AQ

This instruction shifts (A) and (Q) as one 60-bit register. The shift is circular to the left Ɏ bit positions. The lower order bits of A are replaced with the higher order bits of Q, and the lower order bits of Q are replaced with the higher order bits of A. Only the lower order six bits of Ɏ are recognized by this instruction. Maximum shift count permitted is 59 places.

10 ENTER Q

Clear the Q-register. Then transmit Ɏ to Q.

11 ENTER A

Clear the A-register. Then transmit Ɏ to A.

12 ENTER B^n

Clear B-register j. Then transmit Ɏ to B-register j. The branch condition designator, j, is used to specify the selected B-register for this instruction and is not available for its normal function.

13 EXTERNAL FUNCTION ON C^n

This instruction establishes a 1-word external function buffer via the output buffer channel $\hat{j}$ from the storage address $\underline{Y}$. The buffered word address is maintained in the lower order 15 bits of storage address 00140 plus $\hat{j}$.

This instruction is implemented as follows: For all $\hat{k}$ values, $\underline{Y}$ is stored in the upper and lower order half of storage location 00140 plus $\hat{j}$.

If $\hat{k}=3$, a 1-word external function buffer with force is established. (With force, the instruction ignores the external function request from the external device.) The program will hold until the external function word is transmitted.

If $\hat{k}=2$, initiate a 1-word external function buffer and proceed to the next instruction. Subsequent to this instruction the transfer is accomplished when requested by the external device.

If $\hat{k}=1$, a 1-word external function buffer with monitor and with force is established. A monitor interrupt follows the completion of the buffering operation.

If $\hat{k}=0$, a 1-word external function buffer with monitor is established. Proceed to the next instruction. A monitor interrupt follows the completion of the buffering operations. Subsequent to this instruction, the individual transfer is accomplished when requested by the external device.

14 STORE Q

Store (Q) at storage address $\underline{Y}$ as directed by the operand interpretation designator, k. If k=0, complement (Q). If k=4, store in A.

15 STORE A

Store (A) at storage address $\underline{Y}$ as directed by the operand interpretation designator, k. If k=4, complement (A). If k=0, store in Q.

16 STORE B^n

Store a 30-bit quantity, whose lower order 15 bits correspond to the content of B-register j and whose higher order 15 bits are zero, at storage address $\underline{Y}$ as directed by the operand interpretation designator k. The branch condition designator, j, is used to specify the selected B-register for this instruction and is not available for its normal function.

17 STORE C^n OR TEST EFB

If $\hat{k}=3$, transfer the interrupt word from storage address 00520 plus $\hat{j}$ to storage address $\underline{Y}$. The external interrupt request line is reset on channel C^n.

If $\hat{k}=2$, transfer the contents of the C channel specified by $\hat{j}$ to storage address $\underline{Y}$. An Input Data Acknowledge signal is then sent on the C channel. The program will hold until the word is read.

If $\hat{k}=1$ or 0, this instruction tests the status of the EF buffer on the designated channel. If the buffer is active, a jump condition is satisfied.

Then, if $\hat{k}=1$, $(Y)_L$ or, if $\hat{k}=0$, $\underline{Y}$ becomes the address of the next instruction. If the buffer is inactive, the jump condition is not satisfied and the next sequential instruction in the current sequence is executed in the normal manner.

20 ADD A

Add Ɏ to the previous content of the accumulator.

21 SUBTRACT A

Subtract Ɏ from the previous content of the accumulator.

22 MULTIPLY

Multiply (Q) times Ɏ, leaving the double-length product in AQ. If the factors are considered as integers, the product is an integer in AQ.

23 DIVIDE/SQUARE ROOT

If $k \neq 7$, divide (AQ) by Ɏ leaving the quotient in the Q-register and the remainder in the A-register. If $k=7$, take the square root of the (Q) leaving the root in the Q-register and the remainder in the A-register. The remainder $a=n-(Q \times Q)$, where n=the original number.

NOTE: If a divide overflow condition exists, no maintenance console indication is given. However, by coding each divide instruction with $j=3$, a program test for the divide overflow is automatic. With this selection of j, a skip of the next instruction occurs if a divide overflow exists. The skip should be made to a jump instruction which provides a remedial means of noting the error or of correcting it. Therefore, the instruction which follows the divide instruction should have its $j=1$ in order to preclude the jump instruction whenever the divide sequence culminates in a correct answer.

A divide overflow can be detected also if the divide instruction is executed with $j=2$. In this case, a correct answer is indicated when a skip occurs.

24 REPLACE $A+\underline{Y}$

Add $(\underline{Y})$ to the previous content of A. Store (A) at storage address $\underline{Y}$.

25 REPLACE A—$\underline{Y}$
Subtract ($\underline{Y}$) from the previous content of A. Then store (A) at storage address $\underline{Y}$.

26 ADD Q
Interchange (A) and (Q). Then add ¥ to (A). Interchange (A) and (Q). The content of A is undisturbed by this instruction. The branch condition designator, j, has special meaning in this instruction as listed below:

27 SUBTRACT Q
Interchange (A) and (Q). Then subtract ¥ from (A). Interchange (A) and (Q). The content of A is undisturbed by this instruction. The branch condition designator, j, has special meaning in this instruction as listed below.

In instructions 26 and 27 the branch condition designator, j, has the following meaning:

j=0:	Execute the next instruction.
j=1:	Skip the next instruction.
j=2:	Skip the next instruction if (A) is positive.
j=3:	Skip the next instruction if (A) is negative.
j=4:	Skip the next instruction if (Q) is zero.
j=5:	Skip the next instruction if (Q) is non-zero.
j=6:	Skip the next instruction if (Q) is positive.
j=7:	Skip the next instruction if (Q) is negative.

30 ENTER ¥+Q
Clear A. Then transmit (Q) to A. Then add ¥ to (A).

31 ENTER ¥—Q
Clear A. Transmit (Q) to A. Then subtract ¥ from (A). Finally, complement (A).

32 STORE A+Q
Add (Q) to the previous content of A. Then store (A) at storage address $\underline{Y}$ as directed by the operand interpretation designator, k.

33 STORE A—Q
Subtract (Q) from the previous content of A. Then store (A) at storage address $\underline{Y}$ as directed by the operand interpretation designator, k.

34 REPLACE $\underline{Y}$+Q
Clear A. Transmit (Q) to A. Then add ($\underline{Y}$) to (A). Store (A) at storage address $\underline{Y}$.

35 REPLACE $\underline{Y}$—Q
Clear A. Transmit (Q) to A. Subtract ($\underline{Y}$) from (A). Then complement (A) and store at storage address $\underline{Y}$.

36 REPLACE $\underline{Y}+1$

Clear A. Set $(A)=1$. Then add $(\underline{Y})$ to (A). Store (A) at storage address $\underline{Y}$.

37 REPLACE $\underline{Y}-1$

Clear A. Set $(A)=1$. Then subtract $(\underline{Y})$ from (A). Complement (A) and store at storage address $\underline{Y}$.

40 ENTER LOGICAL PRODUCT

Enter the bit-by-bit product of Ɏ and (Q) in A.

The j designator is interpreted in a special way for this instruction for the values $j=2$ or 3.

If $j=2$, skip if the parity of $(A)_f$ is even.

If $j=3$, skip if the parity of $(A)_f$ is odd.

NOTE: Even parity means an even number of "ones" in the A-register. Odd parity means an odd number of "ones" in the A-register.

41 ADD LOGICAL PRODUCT

Add to (A) the bit-by-bit product of Ɏ and (Q).

42 SUBTRACT LOGICAL PRODUCT

Subtract from (A) the bit-by-bit product of Ɏ and (Q).

43 COMPARE MASK

Subtract from (A) the bit-by-bit product of Ɏ and (Q), and perform the branch point evaluation for skip of next sequential instruction as directed by the branch condition designator, j.

This instruction results in no net change in the content of any operational register. It provides, through the branch condition designator, j, a comparison of a portion of Ɏ with (A).

44 REPLACE LOGICAL PRODUCT

Enter in A the bit-by-bit product of $(\underline{Y})$ and (Q). Then store (A) at storage address $\underline{Y}$. The j designator is interpreted in a special way for this instruction for the values $j=2$ or 3. If $j=2$, skip if the parity of $(A)_f$ is even. If $j=3$, skip if the parity of $(A)_f$ is odd.

NOTE: Even parity—an even number of "ones" in the A-register. Odd parity—an odd number of "ones" in the A-register.

45 REPLACE A+LOGICAL PRODUCT

Add to (A) the bit-by-bit product of $(\underline{Y})$ and (Q). Then store (A) at storage address $\underline{Y}$.

46 REPLACE A—LOGICAL PRODUCT

Subtract from (A) the bit-by-bit product of $(\underline{Y})$ and (Q). Then store (A) at storage address $\underline{Y}$.

47 STORE LOGICAL PRODUCT

Store in address $\underline{Y}$ the bit-by-bit product of (A) and (Q) as directed by the operand interpretation designator, k.

50 SELECTIVE SET

Set the individual bits of A to "one" corresponding to "ones" in ¥, leaving the remaining bits of A unaltered.

51 SELECTIVE COMPLEMENT

Complement the individual bits of A to "one" corresponding to "ones" in ¥, leaving the remaining bits of A unaltered.

52 SELECTIVE CLEAR

Clear the individual bits of A corresponding to "ones" in ¥, leaving the remaining bits of A unaltered. In this instruction, k=7 should not be used.

53 SELECTIVE SET

Set the individual bits of A with bits of ¥ corresponding to "ones" in Q, leaving the remaining bits of A unaltered. In this instruction, k=7 should not be used.

54 REPLACE SELECTIVE SET

Set the individual bits of A to "one" corresponding to "ones" in $(\underline{Y})$, leaving the remaining bits of A unaltered. Then store (A) at storage address $\underline{Y}$.

55 REPLACE SELECTIVE COMPLEMENT

Complement the individual bits of A corresponding to "ones" in $(\underline{Y})$, leaving the remaining bits of A unaltered. Then store (A) at storage address $\underline{Y}$.

56 REPLACE SELECTIVE CLEAR

Clear individual bits of A corresponding to "ones" in $(\underline{Y})$, leaving the remaining bits of A unaltered. Then store (A) at storage address $\underline{Y}$.

57 REPLACE SELECTIVE SUBSTITUTE

Clear individual bits of A corresponding to "ones" in Q, leaving the remaining bits of A unaltered. Then form the bit-by-bit product of $(\underline{Y})$ and (Q), and set "ones" of this product in corresponding bits of A, leaving the remaining bits of A unaltered. Then store (A) at storage address $\underline{Y}$.

60 JUMP (Arithmetic)

This instruction clears the program address register, P, and enters a new program address in P for certain conditions of either the A or Q-register content. The branch condition designator, j, is interpreted in a special way for this instruction and thus determines the conditions under which a jump in program address occurs. If the jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the jump condition is satisfied, as listed below, then ¥ defines the address of the next instruction and the beginning of a new program sequence.

j=0: No jump. Set interrupt enable to remove interrupt lockout, thus clearing bootstrap and interrupt modes. Continue with current program sequence.

j=1: Execute jump. Set interrupt enable to remove interrupt lockout, thus clearing bootstrap and interrupt modes.

j=2: Execute jump if (Q) is positive.

j=3: Execute jump if (Q) is negative.

j=4: Execute jump if (A) is zero.

j=5: Execute jump if (A) is non-zero.

j=6: Execute jump if (A) is positive.

j=7: Execute jump if (A) is negative.

61 JUMP (Manual)

This instruction clears the program address register, P, and enters a new program address in P for certain conditions of manual JUMP key selections. The branch condition designator, j, is interpreted in a special way for this instruction and thus determines the conditions under which a jump in program address occurs. If the jump condition is not satisfied, the next sequential instruction of the current sequence is executed in a normal manner. If the jump condition is satisfied, as listed below, then ¥ defines the address of the next instruction and the beginning of a new program sequence.

Program execution may be stopped by certain STOP selections on execution of this instruction. The branch condition designator, j, specifies which key selections are effective.

j=0: Execute jump regardless of key selections.

j=1: Execute jump if JUMP 1 is selected.

j=2: Execute jump if JUMP 2 is selected.

j=3: Execute jump if JUMP 3 is selected.

j=4: Execute jump. Stop computation.

j=5: Execute jump. Stop computation if STOP 5 is selected.

j=6: Execute jump. Stop computation if STOP 6 is selected.

j=7: Execute jump. Stop computation if STOP 7 is selected.

62 JUMP ON C^n ACTIVE INPUT BUFFER

This instruction clears the program address register, P, and enters a new program address in P for certain input buffer conditions on the channel designated by $\hat{j}$. If the buffer is active, the jump condition is satisfied; then ¥ defines the address of the next instruction. If the buffer is inactive, the jump condition is not satisfied. The next sequential instruction in the current sequence is executed in the normal manner. $\hat{k}$=0, 1, 2, or 3 permitted.

63 JUMP ON C^n ACTIVE OUTPUT BUFFER

This instruction clears the program address register, P, and enters a new address in P for certain output conditions on the channel designated by $\hat{j}$. If the buffer is active, the jump condition is satisfied, then ¥ defines the address of the next instruction. If the buffer is inactive, the jump condition is not satisfied. The next sequential instruction in the current sequence is executed in the normal manner. $\hat{k}$=0, 1, 2 or 3 permitted.

64 RETURN JUMP (Arithmetic)

This instruction executes a return jump sequence for certain conditions of either the A or

Q-register content. The branch condition designator, j, is interpreted in a special way for this instruction and determines the conditions under which the return jump sequence is executed. If the return jump condition is not satisfied, then the next sequential instruction in the current sequence is executed in a normal manner. If the return jump condition is satisfied, as listed below, then the following sequence is performed.

Store (P) in the lower half of memory address defined by ¥. Then jump to ¥+1.

j=0:	No jump. Set interrupt enable to remove interrupt lockout, thus clearing bootstrap and interrupt modes. Continue with current program sequence.
j=1:	Execute jump. Set interrupt enable to remove interrupt lockout, thus clearing bootstrap and interrupt modes.
j=2:	Execute return jump if (Q) is positive.
j=3:	Execute return jump if (Q) is negative.
j=4:	Execute return jump if (A) is zero.
j=5:	Execute return jump if (A) is non-zero.
j=6:	Execute return jump if (A) is positive.
j=7:	Execute return jump if (A) is negative.

65 RETURN JUMP (Manual)

This instruction executes a return jump sequence for certain conditions of manual key selections. The branch condition designator, j, is interpreted in a special way for this instruction and determines the conditions under which the return jump sequence is executed. If the return jump condition is not satisfied, the next sequential instruction in the current sequence is executed in a normal manner. If the return jump condition is satisfied, as listed below, then the following sequence is performed.

Store (P) in the lower half of memory address defined by ¥. Then jump to ¥+1.

j=0:	Execute return jump regardless of key selections.
j=1:	Execute return jump if JUMP 1 is selected.
j=2:	Execute return jump if JUMP 2 is selected.
j=3:	Execute return jump if JUMP 3 is selected.
j=4:	Execute return jump. Then stop computation.
j=5:	Execute return jump. Stop computation if STOP 5 is selected.
j=6:	Execute return jump. Stop computation if STOP 6 is selected.
j=7:	Execute return jump. Stop computation if STOP 7 is selected.

66 TERMINATE C^n INPUT BUFFER OR ENABLE, DISABLE INTERRUPTS

If $\hat{k}=0$, terminate the input buffer on channel $\hat{j}$. No input buffer monitor interrupt will occur. If $\hat{k}=1$, b=0, enable all interrupts. If $\hat{k}=1$, $b\neq0$, disable all interrupts. If $\hat{k}=2$, b=0, enable all external interrupts. If $\hat{k}=2$, $b\neq0$, disable all external interrupts. The external interrupt request signal is removed from all channels. If $\hat{k}=3$, b=0, enable the external interrupt on the channel specified by $\hat{j}$. If $\hat{k}=3$, $b\neq0$, disable the external interrupt on the channel specified by $\hat{j}$. The External Interrupt Request signal is removed from the channel specified by $\hat{j}$. The operand address designator, y, is not translated for this instruction.

67 TERMINATE C^n OUTPUT BUFFER, ALL BUFFERS, OR TERMINATE C^n EXTERNAL FUNCTION BUFFER

If $\hat{k}=0$, terminate the output data buffer on channel $\hat{j}$. If $\hat{k}=1$, terminate the External Function Buffer on channel $\hat{j}$. If the channel specified by $\hat{j}$ is involved with the use of an output register of an intercomputer group, a resume signal is simulated and sent to the register. If $\hat{k}=2$, terminate all buffers. If an output register is being used for intercomputer communications, a resume signal is simulated and sent to this register. For all values of $\hat{k}$, no output buffer monitor interrupt will occur. The index designator, b, and the operand address designator, y, is not translated for this instruction.

70 REPEAT

Clear B^7 and transmit $\yen$ to B^7. If $\yen_L$ is non-zero, transmit (j) to r (designator register), thereby initiating the repeat mode. If $\yen_L$ is zero, skip the next instruction.

The repeat mode executes the instruction immediately following the Repeat instruction $\yen_L$ times. B^7 contains the number of executions remaining throughout the repeat mode.

If no skip condition is met for the repeated instruction, the repeat mode terminates and the instruction following the repeated instruction is executed. If the skip condition for the repeated instruction is met, the repeat mode terminates, and the instruction following the repeated instruction is skipped. Following the repeat mode termination, the count remains in B^7.

In no way does the repeat mode alter a repeated instruction as stored in memory.

The three low-order bits of the r-designator (from j of instruction 70) affects the operand indexing as follows:

$r=0$: Do not modify the operand address of the repeated instruction after each individual execution.

$r=1$: Increase the operand address of the repeated instruction by one after each execution of the repeated instruction.

$r=2$: Decrease the operand address of the repeated instruction by one after execution of the repeated instruction.

$r=3$: Repeat the initial B-register modification of the repeated instruction before each execution.

$r=4$: Do not modify the operand address of the repeated instruction after each individual execution. If the repeated instruction is a replace instruction, the operand address is modified by (B^6) for the store portion of the replace instruction.

$r=5$: Increase the operand address of the repeated instruction by one each execution of the repeated instruction. If the repeated instruction is a replace instruction, the incremented operand address is modified by (B^6) for the store portion of the replace instruction.

r=6: Decrease the operand address of the repeated instruction by one after each execution of the repeated instruction. If the repeated instruction is a replace instruction, the decremented operand address is modified by (B^6) for the store portion of the replace instruction.

r=7: Repeat the initial B-register modification of the repeated instruction before each execution. If the repeated instruction is a replace instruction, the modified operand address is further modified by (B^6) for the store portion of the replace instruction.

NOTE: Instruction 70 j designator establishes the repeat mode r designator since j is transmitting to r.

71. SKIP ON B^n

If the content of B-register j is equal to ¥, skip the next instruction in the current sequence and proceed to the following instruction. Clear B-register j.

If the content of B-register j is not equal to ¥, proceed to the next instruction in the sequence in a normal manner. Increase the content of B-register j by one.

The branch condition designator, j, is used to designate the selected B-register in this instruction and is not available for its normal function. Only the lower order 15 bits of ¥ are used in the comparison described in the preceding paragraph.

72 J JUMP ON B^n

If the content of B-register j is non-zero, execute a jump in program address as defined by ¥. Reduce the content of B-register j by one.

If the content of B-register j is zero, proceed to the next instruction in a normal manner. Do not alter the content of B-register j.

The branch condition designator, j, is used to designate the selected B-register in this instruction and is not available for its normal function. If the jump condition is satisfied, then the lower order 15 bits of ¥ define the address of the next instruction and the beginning of a new program sequence. The highest order 15 bits defined by ¥ are not used in this instruction.

73 INPUT BUFFER ON C^n (Without Monitor Mode)

This instruction establishes an input buffer via input channel $\hat{j}$ utilizing an area of memory defined by $(\underline{Y})$. The memory address limits determined from $(\underline{Y})$ are transferred to the buffer control register associated with the channel. Subsequent to this instruction, individual transfers will be executed at a rate determined by the external device. The storage address initially established by this instruction will be advanced by one preceding each individual transfer. The next current address will be maintained throughout the buffer process in the lower order 15 bits of control register 00100 plus $\hat{j}$. This mode will continue until it is superseded by a subsequent initiation or termination of an input buffer via the same input channel or until the higher order half and the lower order half of control register 00100 plus j contain equal quantities, whichever occurs first. It should be noted that since the storage required for a buffering operation may be of any size utilizing any area of computer memory, it is a program responsibility to ensure that correct values for the buffer limits have been established for ¥ previous to the execution of this instruction.

This instruction is implemented as follows: If $\hat{k}=3$, store $(\underline{Y})$ in storage location 00100 plus $\hat{j}$. If $\hat{k}=1$, store the lower order 15 bits of $(\underline{Y})$ in the lower order half of storage location 00100 plus $\hat{j}$, leaving the higher order half undisturbed. If $\hat{k}=0$, store $(\underline{Y})$ in the lower order half of storage location 00100 plus $\hat{j}$ leaving the higher order half undisturbed. Proceed to the next instruction. $\hat{k}=2$ is not permitted.

74 OUTPUT BUFFER ON C^n (Without Monitor Mode)

This instruction establishes an output buffer (if $\hat{k}\neq 2$, output data buffer; if $\hat{k}=2$ External Function buffer) via output channel $\hat{j}$ utilizes an area of memory defined by $(\underline{Y})$. The memory address limits determined from $(\underline{Y})$ are transferred to the buffer control register associated with the channel. Subsequent to this instruction, the individual transfers will be executed at a rate determined by the external device. The storage address initially established by this instruction will be advanced by one preceding each individual transfer. The next current address will be maintained throughout the buffer process in the lower order 15 bits of the control register if $\hat{k}\neq 2$, 00120 plus $\hat{j}$; if $\hat{k}=2$, 00140 plus $\hat{j}$. This mode will continue until it is superseded by a subsequent initiation or termination of an output buffer via the same output channel or until the higher order half and the lower order half of the control register (if $\hat{k}\neq 2$, 00120 plus $\hat{j}$; if $\hat{k}=2$, 00140 plus $\hat{j}$) contain equal quantities, whichever occurs first. It should be noted that since the storage required for a buffering operation may be of any size, utilizing any area of computer memory, it is a program responsibility to ensure that correct values for the buffer limits have been established for ~~Y~~ previous to the execution of this instruction.

This instruction is implemented as follows: If $\hat{k}=3$, store $(\underline{Y})$ in storage location 00120 plus $\hat{j}$. If $\hat{k}=1$, store the lower order 15 bits of $(\underline{Y})$ in the lower order half of storage location 00120 plus $\hat{j}$ leaving the higher order half undisturbed. If $\hat{k}=0$, store $\underline{Y}$ in the lower half of storage location 00120 plus $\hat{j}$, leaving the higher order half undisturbed. If $\hat{k}=2$, store $(\underline{Y})$ in storage location 00140 plus $\hat{j}$. Proceed to the next instruction.

75 INPUT BUFFER ON C^n (With Monitor Mode)

This instruction establishes an input buffer via input channel $\hat{j}$ utilizing an area of memory defined by $(\underline{Y})$. The memory address limits determined from $(\underline{Y})$ are transferred to the buffer control register associated with the channel. Subsequent to this instruction, the individual transfers will be executed at a rate determined by the external device. The storage address initially established by this instruction will be advanced by one preceding each individual transfer. The next current address will be maintained throughout the buffer process in the lower order 15 bits of control register 00100 plus $\hat{j}$. This mode will continue until it is superseded by a subsequent initiation or termination of an input buffer via the same input channel or until the higher order half and the lower order half of the control register contain equal quantities, whichever occurs first. The initiation of this input buffer selects the input channel $\hat{j}$ and establishes a buffer monitor on input channel $\hat{j}$. A monitor interrupt to address $00040+\hat{j}$ follows completion of the buffering operation. It should be noted that since the storage required for a buffering operation may be of any size utilizing any area of computer memory, it is a program responsibility to ensure that correct values for the buffer limits have been established for ~~Y~~ previous to the execution of this instruction.

This instruction is implemented as follows: If $\hat{k}=3$, store $(\underline{Y})$ in storage location 00100 plus $\hat{j}$. If $\hat{k}=1$, store the lower order 15 bits of $(\underline{Y})$ in the lower order half of storage location 00100 plus $\hat{j}$ leaving the higher order half undisturbed. If $\hat{k}=0$, store $\underline{Y}$ in the lower order half of storage location 00100 plus $\hat{j}$. Proceed to the next instruction. $\hat{k}=2$ is not permitted.

76 OUTPUT BUFFER ON C^n (With Monitor Mode)

This instruction establishes an output buffer (if $\hat{k}\neq 2$, output data buffer; if $\hat{k}=2$, external function buffer) via output channel $\hat{j}$ utilizing an area of memory defined by $(\underline{Y})$. The memory address limits determined from $(\underline{Y})$ are transferred to the buffer control register associated with the channel. Subsequent to this instruction, the individual transfers will be executed at a rate determined by the external device. The storage address initially established by this instruction will be advanced by one preceding each individual transfer. The next current address will be maintained throughout the buffer process in the lower order 15 bits of control register (if $\hat{k}\neq 2$, 00120 plus $\hat{j}$; if $\hat{k}=2$, 00140 plus $\hat{j}$). This mode will continue until it is superseded by a subsequent initiation or termination of an output buffer via the same output channel or until the higher order half and the lower order half of the control register contain equal quantities, whichever occurs first. The initiation of this output buffer selects the output channel $\hat{j}$ and establishes a buffer monitor on output channel $\hat{j}$. A monitor interrupt (if $\hat{k}\neq 2$, to address $00060+\hat{j}$; if $\hat{k}=2$ to address $00500+\hat{j}$) follows the completion of the buffering operation. It should be noted that since the storage required for a buffering operation may be of any size utilizing any area of computer memory, it is a program responsibility to ensure that correct values for the buffer limits have been established for $\underline{Y}$, previous to the execution of this instruction.

This instruction is implemented as follows: If $\hat{k}=3$, store $(\underline{Y})$ in storage location 00120 plus $\hat{j}$. If $\hat{k}=1$, store the lower order 15 bits of $(\underline{Y})$ in the lower order half of storage location 00120 plus $\hat{j}$ leaving the higher order half undisturbed. If $\hat{k}=0$, store $\underline{Y}$ in the lower order half of storage location 00120 plus $\hat{j}$ leaving the higher order half undisturbed. If $\hat{k}=2$, store $(\underline{Y})$ in storage location 00140 plus $\hat{j}$. Proceed to the next instruction.

UNIVAC
DIVISION OF SPERRY RAND CORPORATION
DEFENSE MARKETING
UNIVAC PARK, ST. PAUL, MINN. 55116
AREA CODE 612, 698-2451

UNIVAC Regional Offices

PX 3414

WASHINGTON, D. C., 20007, *2121 Wisconsin Avenue, 338-8510* ★ WALTHAM, MASS., 02154, *69 Hickory Drive, 899-4110* ★ COCOA BEACH, FLORIDA, 32931, *Suite 176, Holiday Office Center, 1325 No. Atlantic Avenue, 783-8461* ★ HOUSTON, TEXAS, 77058, *Suite 122, Alpha Building, 16811 El Camino Real, HU 8-2240* ★ LOS ANGELES, CALIFORNIA, 90045, *Suite 220, 5316 West Imperial Highway, 678-2531* ★ SAN BERNARDINO, CALIFORNIA, 92410, *Suite 219, 808 East Mill Street, 889-1096* ★ SAN DIEGO, CALIFORNIA, 92110, *3045 Rosecrans, 224-3333*