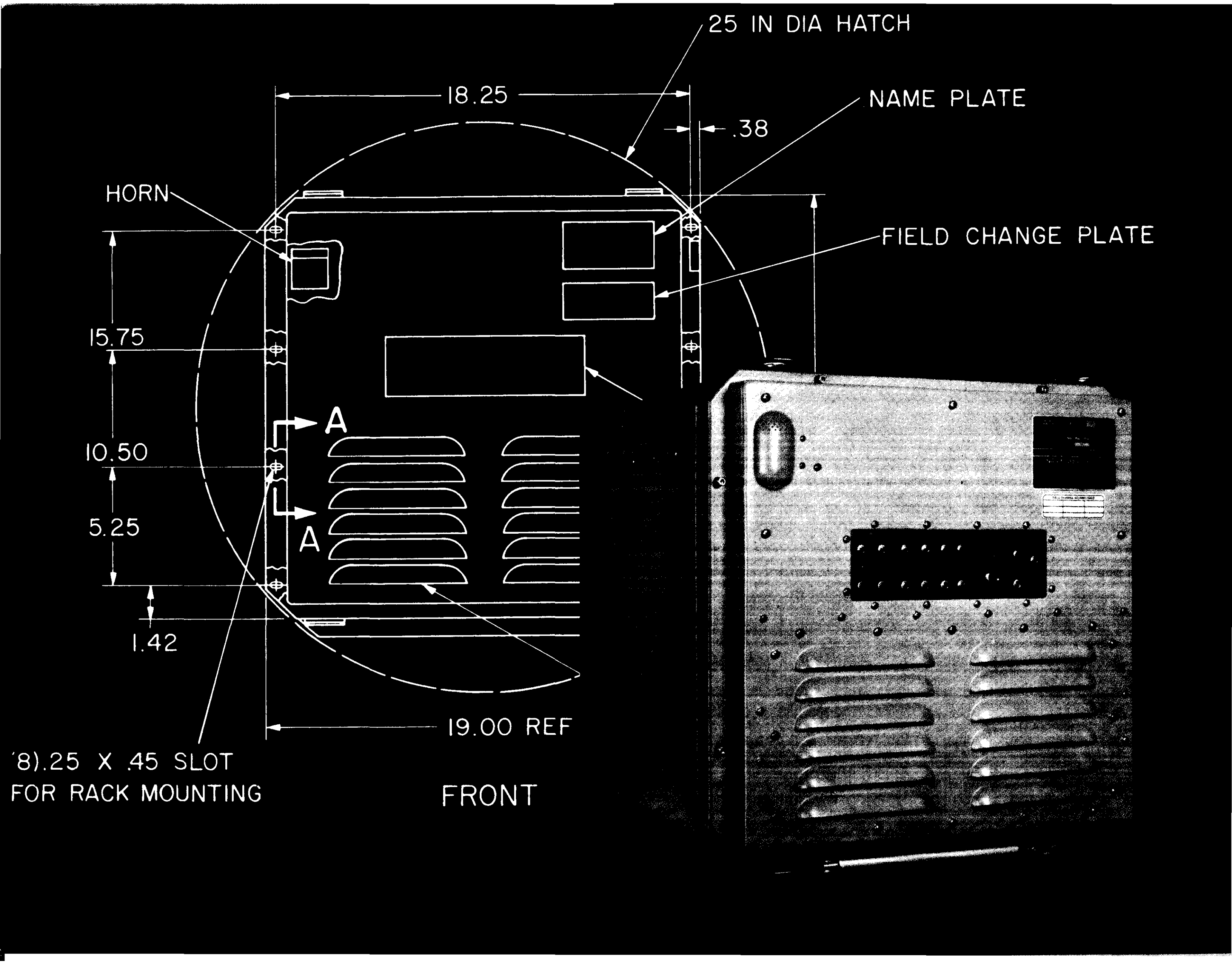
SPERRY UNIVAC

AN/UYK-20



TECHNICAL DESCRIPTION

RESTRICTIVE NOTICE

This manual is intended to inform the reader regarding the general construction, operational characteristics, and capabilities of the AN/UYK-20 computer. It should not, however, be considered as an equipment specification, and Sperry Univac in no way warrants the accuracy or completeness of the manual for procurement purposes. Products and services described herein are available for sale only to the federal government of the United States of America or its designees.

AN/UYK-20

TECHNICAL DESCRIPTION

SPERRY UNIVAC

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT.)

FUNCTIONAL OPERATION (CONT.)

LIST OF ILLUSTRATIONS

LIST OF TABLES

AN/UYK-20X(V)
60 Hz Equipment Identification

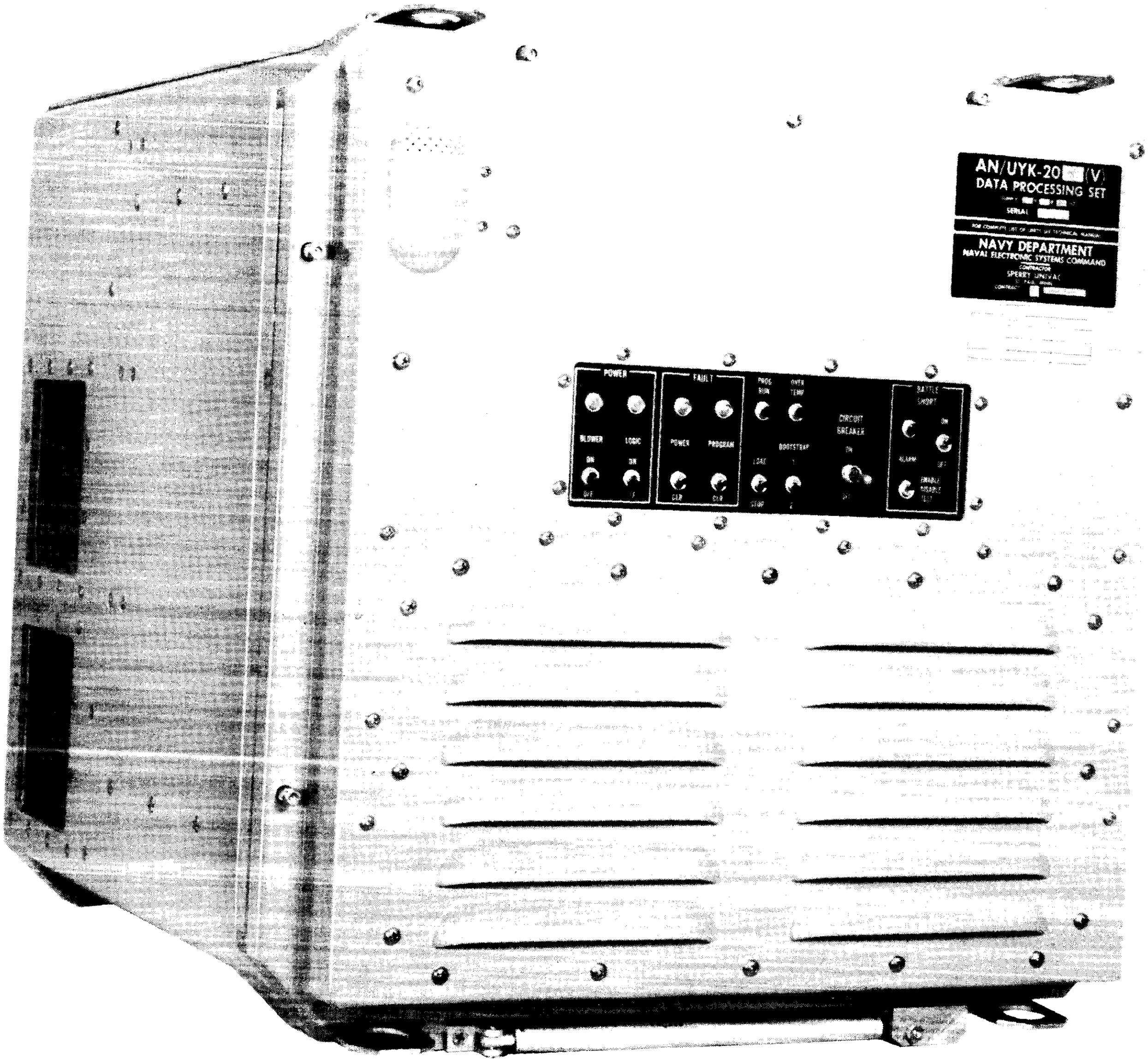
Government Identifier	Description
AN/UYK-20X(V)	Data Processing Set
CY-7446/UYK-20X(V)	Electrical Equipment Cabinet
CP-1189(V)/UYK-20X(V)	Processor-Verifier Unit
C-9675/UYK-20X(V)	Control-Monitor
C-9670(V)/UYK-20X(V)	Core Memory-Control Unit
PP-7109/UYK-20X(V)	Power Supply 3 phase, 115 volt
PP-7110/UYK-20X(V)	Power Supply 3 phase, 208 volt
PP-7111/UYK-20X(V)	Power Supply 1 phase, 115 volt

AN/UYK-20(V)
400 Hz Equipment Identification

AN/UYK-20(V)	Data Processing Set
CY-7445/UYK-20(V)	Electrical Equipment Cabinet
CP-1188(V)/UYK-20(V)	Processor-Verifier Unit
C-9674/UYK-20(V)	Control-Monitor
C-9531(V)/UYK-20(V)	Core Memory-Control Unit
PP-7032/UYK-20(V)	Power Supply 3 phase, 115 volt
PP-7107/UYK-20(V)	Power Supply 3 phase, 208 volt
PP-7108/UYK-20(V)	Power Supply 1 phase, 115 volt

AN/UYK-20(V) and AN/UYK-20X(V)
Equipment Identification

MK-1693/UYK-20(V)	-15V (Slow) Interface Kit
MK-1694/UYK-20(V)	-3V (Fast) Interface Kit
MK-1695/UYK-20(V)	+3.5V (Fast) Interface Kit
MK-1718/UYK-20(V)	EIA-STD-RS232 Synchronous Serial Interface Kit
MK-1719/UYK-20(V)	MIL-STD-188C Synchronous Serial Interface Kit
MK-1720/UYK-20(V)	NTDS Fast Serial Interface Kit
MK-1721(V)/UYK-20(V)	MIL-STD-188C Asynchronous Serial Interface Kit
MK-1722(V)/UYK-20(V)	EIA-STD-RS232 Asynchronous Serial Interface Kit
MK-1723(V)/UYK-20(V)	Micromemory Program Kit
MK-1724/UYK-20(V)	Electronic Equipment Maintenance Kit
MK-1806/UYK-20(V)	VACALES Serial Interface Kit
MU-604-UYK-20(V)	Core Memory Unit


AN/UYK-20 (V)
DATA PROCESSING SET
SERIAL
NAVY DEPARTMENT
NAVAL ELECTRONIC SYSTEMS COMMAND
CONTRACTOR
SPERRY UNIVAC
CONTRACT
POWER
BLOWER
LOGIC
ON
OFF
FAULT
POWER
PROGRAM
CLR
CLR
PROG
RUN
OVER
TEMP
BOOTSTRAP
LOAD
STOP
CIRCUIT
BREAKER
ON
OFF
BATTLE
SHORT
ON
ALARM
OFF
ENABLE
DISABLE
TEST

THE AN/UYK-20(V) COMPUTER

INTRODUCTION

The AN/UYK-20(V) is a general purpose, militarized computer with a wealth of computing power in a small, ruggedized package. It is designed to meet the requirements of small and medium sized applications in shipboard, mobile shelter or other military environments.

A choice of configurations are offered which encompass a variety of applications. A small configuration can grow with optional functions that increase efficiency and versatility. Most other options can be added in the field by simply incorporating printed circuit cards or memory modules to the basic unit. Hence, current and near-future applications can define an initial configuration. Features to enhance processing and input/output capabilities or to meet requirements of system growth may be added in modular form.

Modularity, versatility and serviceability are design features to make it adaptable to the various current and future applications of the armed forces and other Government agencies. Each off-the-shelf unit manufactured by Sperry Univac is wired to accommodate currently offered optional features and also to allow for future enhancements or changes. The computer incorporates a 750 nanosecond core memory and an exceedingly flexible microprogrammable control section. These features provide a very fast computing capability as well as affording a basis for tailoring functional operations to specific or unique applications.

AN EXCELLENT LONG-TERM INVESTMENT

Expansion of functional capability may be accomplished by adding features because any version of the computer contains features that are a subset list of those in a maximum configuration.

A data processing system with a high performance/cost ratio is attainable when the AN/UYK-20 serves as a foundation. Simplicity and compatibility, combined with functional and physical flexibility, characterize the AN/UYK-20 in all of its available configurations. Simplicity, which is accomplished by the power and flexibility of the AN/UYK-20 instructions, provides simple and efficient program generation and implementation. This high quality and maintainable computer, characteristic of Sperry Univac products, will provide the faithful and dependable service expected in a militarized processing system. Reliability and maintainability, two attributes of excellence historically demonstrated in Sperry Univac products, are incorporated in the design and development of the AN/UYK-20. The input/output capabilities offer a wide interface potential that include byte, whole or dual word parallel transfers, serial transfers, internally controlled buffers, peripheral equipment selection and various interface signal levels and transfer speeds.

REAL TIME APPLICATIONS

Functional characteristics of the AN/UYK-20 make it as ideally suited to dedicated real-time applications as to the performance of stand-alone and distributed process systems. A hardware initiated, multilevel interrupt processing capability provides efficient and rapid parameter manipulation and preparation prior to the actual interrupt servicing. Overhead functions normally performed by interrupt processing routines are thereby decreased and faster response time is achieved. The processing efficiency obtainable with the use of the general purpose registers and related instructions provides the capability to meet the high data rate environments encountered in time-critical, real-time systems associated with fire control radar, telemetry or on-line process control applications and real-time systems associated with communications, display controlling or data systems.

Shipboard Defense Systems Applications

Processing all raw data available from a task force and a ship's systems is a huge assignment for a command and control (C&C) system.

The AN/UYK-20 can be utilized very effectively in reducing this burden by absorbing specific data reduction and related overhead tasks in the system.

Functionally a tactical data system coordinates the collection of data from many sources including sonar, radar, IFF and passive detection apparatus communication links. It coordinates all data with ship systems status and navigation information, prepares a clear picture of the tactical situation to aid a decision making process and communicates the decisions to applicable and available action systems and personnel.

The AN/UYK-20 implemented as a pre-processor has the calculating speed and data handling characteristics to reduce large volumes of raw data to usable values and arranging them in a format acceptable to the C&C computer for direct integration into the total system.

Communications Systems

The AN/UYK-20 with the I/O controller and its bit and byte manipulation instruction repertoire is ideally suited to communications applications. The serial I/O channels provide great flexibility for handling both synchronous and asynchronous communications lines in a wide range of rates. Network control, store and forward, and line concentration functions are readily implemented through the incorporation of the AN/UYK-20 in a communications system. Its inherent reliability insures continuous, effective service in these applications.

Signal Processing

A major military application is processing radar, sonar and beacon signals. In this application, the systems provide a continual input of data in a real-time environment. High rate processing and fast reaction time is required to determine targets, direction, distance and other information. This critical time data processing task is handled easily by the AN/UYK-20. Its comprehensive and flexible instruction set executed by the fast central processor section, its programmable real-time clock, and the high-speed, hardware-initiated, interrupt structure provide the capability to perform the complex computations in real-time. Direct access to memory for real-time data input and/or output is accomplished by the very fast, programmable input/output section or externally controlled direct memory access (DMA) option.

Control Systems

In addition to weapons control systems, other control systems normally found include air traffic, radar, electronic countermeasures and navigation. Complex control systems, as with signal processing, require high computational capabilities. While the quantity of input data is lower, input is received from more than one source. Here again the AN/UYK-20 qualifies for this application. The number of input/output channels can be expanded as required by plug-in units. Complex computations required for commanding the system are accomplished with programs that utilize the fast, general registers and the associated single and double-precision arithmetic.

Other AN/UYK-20 Military Applications

- Message Handling – receiving, logging and forwarding
- Fire Control
- Navigation
- Management Information
- Telemetry
- Communication Links
- Radar Processing
- Data Reduction
- Sensor Processing
- Range Tracking
- Logistics

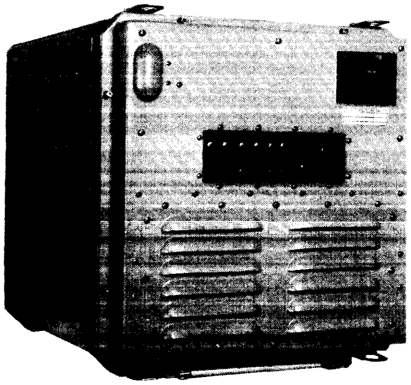


Figure 1. AN/UYK-20 Computer

THE AN/UYK-20(V) COMPUTER

SPECIFICATIONS AND FEATURES

SUMMARY OF STANDARD FEATURES

Militarized Construction; MIL-E-16400
General Purpose, 16-bit digital computer
Physically and functionally modular and expandable
MSI components
Micro program control
Integral blowers and power supplies
19-inch rack mountable
Front access for maintainability
Plug-in options

Central Processor

Microprogrammed controller
Two's complement arithmetic
4-bit, 8-bit, 16-bit and 32-bit operands
16 high speed general purpose registers
2 program status registers
3-level interrupt processing (hardware serviced)
16-bit and 32-bit instructions
Basic instructions - 5 formats
Sample execution time

Shift	1.0 microsecond
Add	.84 microseconds
Multiply	3.6 microseconds
Divide	6.8 microseconds

Direct addressing to 65K words
Relative addressing to 1024 word pages
Indexing via general registers
Cascaded indirect addressing
Relative addressing by page
Power Fault/Auto-restart
Real-time clock and monitor clock
Bootstrap NDRO (read only) memory

Main Storage

Expandable — 8K to 65K words in 8K increments
16-bit words
Read/Restore cycle time — 750 nanoseconds
Asynchronous timing

Input-Output Controller

Up to 32 program initiated input/output chains
I/O instruction repertoire — same format as CP
Full Duplex input/output channels
Control memory for each channel
Up to 16 channels (combination serial and parallel)
Parallel channels:
Expandable in 4 channel groups

Serial channels:
Expandable in 2 or 4 channel groups

PHYSICAL

Temperature Range
Operating: 0^{o}C to 50^{o}C
Storage: -62^{o}C to $+75^{o}$C
Relative Humidity: to 95%
Size (inches); maximum
Height: 20
Width: 19
Depth: 24
Pass through 25 inch hatch
Weight: 220 pounds maximum

Primary Power

115 or 208 volts
1000 watts (maximum configuration)

ENHANCEMENT OPTIONS

Features of the AN/UYK-20 computer provide functional adaptability for many application requirements. Available options increase its capacity, enhance its flexibility, and provide functions required by certain applications. The following options may be selected for the UYK-20 and may be added without wiring or cabinet changes by plugging the required module into the basic computer unit.

Central Processor

Additional 16 general register (maximum of 32 registers).
Additional micromemory - 512 words. (Customer defined)
Customer defined Bootstrap NDRO programs.
Math Pac functions
Square root
Trigonometric and hyperbolic vector and rotate
Floating point arithmetic
Double precision multiply and divide.

Main Memory

Memory Size: 8,192 word increments to 65,536 total
Direct Memory Access (DMA) interface.

Input/Output Controller

A maximum of 16 input/output channels are available in groups as follows:

Parallel Channels in 4 Channel Groups

Types:
-3 Volt NTDS interface (fast)
-15 Volt NTDS interface (slow)
+3.5 Volt ANEW interface

Modes:
8-bit byte, 16-bit word, or 32-bit dual channel transfers.
Dual channel operations on two 4-channel groups (0 and 1) or (2 and 3) having the same interface types.
Normal transfers available on single or dual channels.
Externally specified addressing (ESA) operation on dual channels
Intercomputer operation on single or dual channels
Peripheral input simulation on single channel. (NTDS slow interface)

Serial Channels in 2 Channel Groups

MIL-STD-188C characteristics/EIA-STD-RS232C characteristics
Synchronous — to 9600 bits per second
Asynchronous — 75, 150, 300, 600, 1200 or 2400 bits/second (any four may be obtained by the option) program controlled selection.
Character size:
Synchronous or asynchronous — 5, 6, 7 or 8 level (program controlled selection)
NTDS serial interface characteristics
Intercomputer operation on single or dual channels
Peripheral input simulation on single channel. (NTDS slow)

Serial Channels in 4 Channel Groups

VACALES
Synchronous — to 9600 bits per second
Character size — 1-16 bits under program control

Power Supply Input Power

3 phase 208 volt, 60 Hz or 400 Hz
3 phase 115 volt, 60 Hz or 400 Hz
1 phase 115 volt, 60 Hz or 400 Hz

MODULAR ARCHITECTURE

Functionally, the AN/UYK-20 architecture is organized around a microprogrammed controller and a two-bus data exchange structure. The various functional elements accept bit configurations from the source bus, interpret and manipulate them, and when appropriate, return bit-configured information to the bus for acceptance by another functional element. The second or destination bus provides an additional communication path between the arithmetic and logic unit and the various registers and allows the system to overlap functions. This architectural technique increases processing speed and allows great flexibility in tailoring a system to meet the requirements to special applications. (See Figure 2.)

·AN/UYK-20 is entirely modular. It is in complete compliance with the modularity requirements of MIL-E-16400. The base module of the computer is the cabinet. As shown in Figure 3, the Central Processor/Input-Output Controller (CP/IOC) chassis, memory, power supply, maintenance panel, and operator panel are basic components of the computer.

Sperry Univac uses MSI devices for the AN/UYK-20 logic. The basic replaceable logic unit is a printed circuit card heavily populated with MSI devices. (See Figure 4.) MSI devices combine the flexibility of discrete-component design and the economy, compactness and reliability of large-scale integration. This reduces cost, physical volume and power requirements and also increases circuit speeds. Figure 4 is photo of large and small cards. Plug-in, printed wiring cards are directly inserted into the CP/IOC chassis. Plug-in core matrix boards and printed wiring control boards are directly inserted into the memory module. Careful design of circuits, selection of components and the self contained cooling system assure circuit reliability.

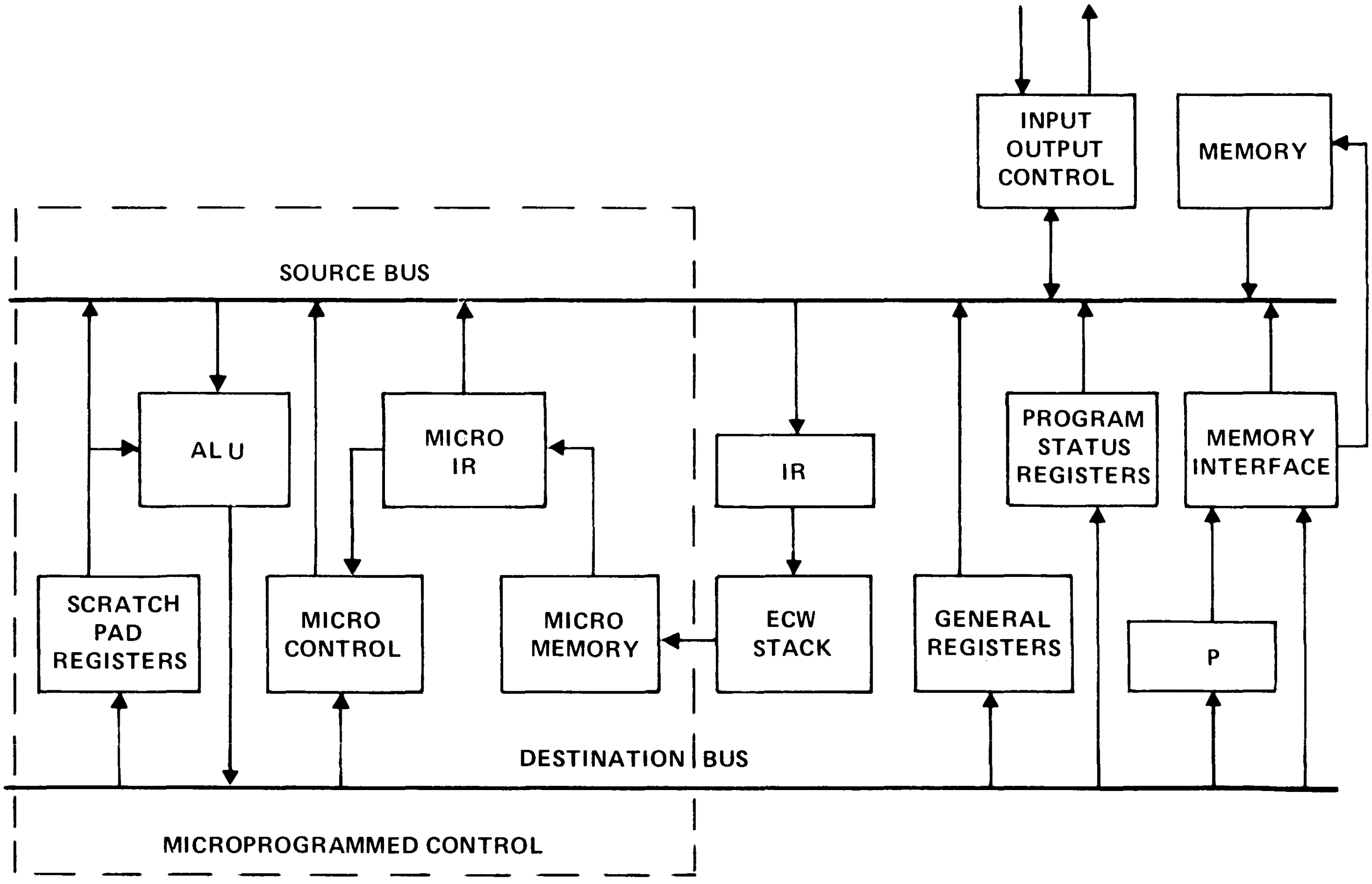


Figure 2. Functional Architecture

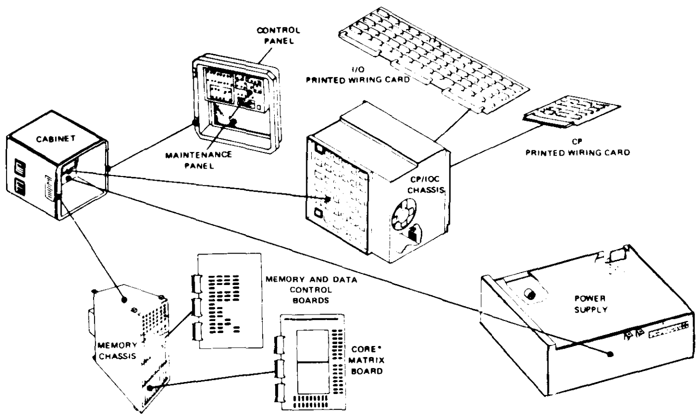


Figure 3. AN/UYK-20 Modular Architecture

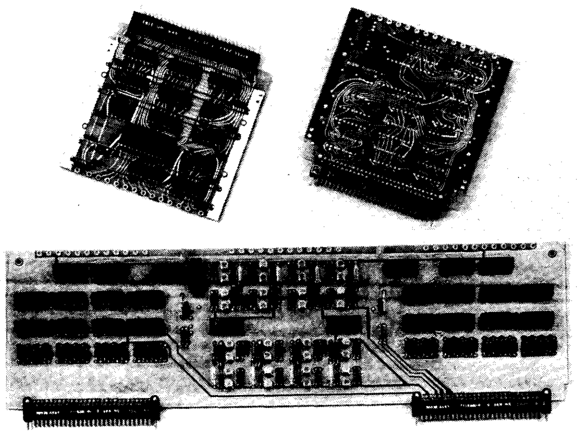


Figure 4. Printed Circuit (PC) Card

CONSTRUCTION

Physically, the functional units are assembled in a cabinet that is constructed from aluminum channel and aluminum sheet and braced to provide structural rigidity. One cooling air inlet is located in the door of the cabinet and the two air exhausts are provided on the left side of the cabinet. Provision has been made in the base for a free-standing mount as well as mounting within a standard 19-inch rack. The front cover incorporates a rugged hinge and latch system that provides a uniform high clamping pressure againsı the cabinet opening. Gasketing around the periphery of the front cover provides for EMI and mositure sealing. A maintenance panel is located on the inside of the front cover and a control panel on the outside.

The rear of the cabinet contains the I/O connector panel, a power connector and its attendant filter assembly, the DMA and external real-time clock jacks.

The cabinet top is a separate panel which is bolted in place and can be replaced with a water cooled heat exchanger if such a feature were to be required.

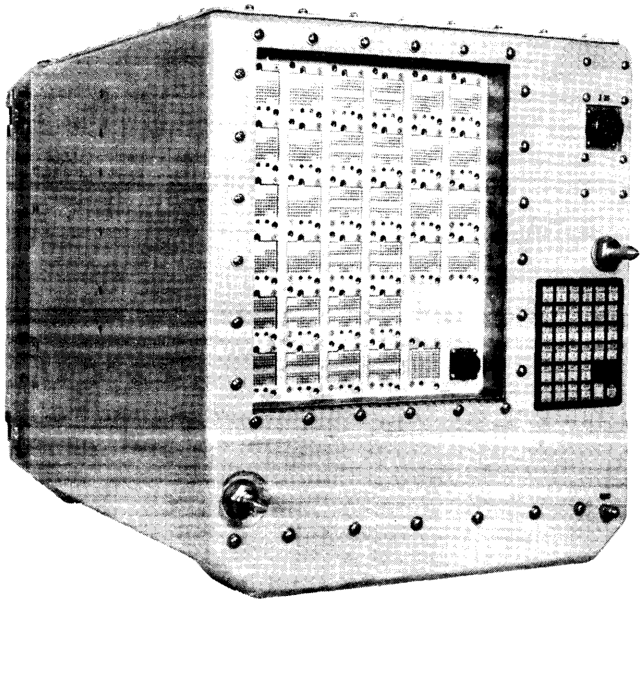


Figure 5. Power, External RTC and I/O Connectors

All assemblies have been designed to be repairable in an emergency situation. The only fixed or chassis mounted components are the cooling fans, power line filters and the control/maintenance panel switches and indicators. The assembled computer is shown in Figure 6.

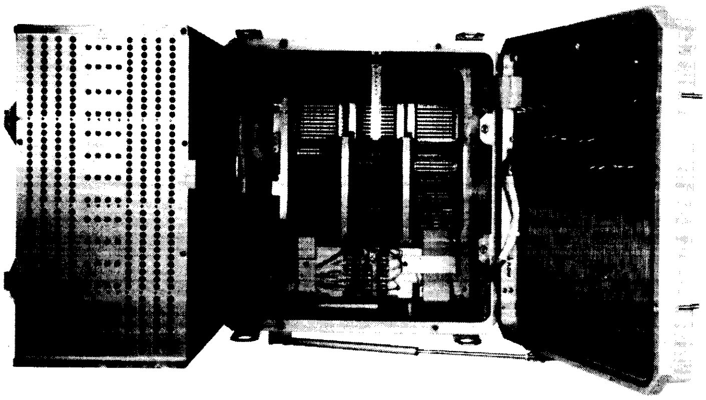


Figure 6. Computer Opened to Show Construction

Optional features offered require only the removal, insertion or substitution of plug-in modules. The AN/UYK-20 is wired for the maximum configuration which includes:

- 65,536 word memory
- 192 word NDRO memory
- 16 input/output channels consisting of any combination of parallel and serial channels, parallel channels in groups of four and serial channels in groups of two or four.
- Direct memory access
- 32 general registers
- Real-time clock and monitor clock
- 512 words of user defined micromemory
- Connectors mounted in accordance with MIL-F-18870.
- MATH PAC

Removal or addition of all options is permitted within the constraints of this maximum configuration.

Design and construction of the AN/UYK-20 is centered around a selection of high quality components and precise manufacturing processes. Sperry Univac experience in producing equipment for defense systems that are used in military environments provides the techniques for building exceptional quality into the manufactured product. Components selected and assembled under Sperry Univac's quality assurance program produces equipment that operates reliably in adverse environments. This same high quality is a characteristic maintained in all modules of the computer, thereby assuring high reliability for any configuration.

MAINTAINABILITY

Accessibility to replaceable items, easy testing and quick malfunction localization are essential to good and efficient maintenance. Sperry Univac design includes these features. The lowest recommended maintenance level is the printed circuit card. When the cabinet front is opened the maintenance panel is exposed and the memory stacks open out on their hinged supports. Test points are exposed. Printed circuit card modules are accessible for removal or insertion. Any card is removable with a simple tool. Each card has its unique, labeled, keyed position and is guided so that it is aligned properly to the female connector when inserted. Circuits in the memory chassis are accessible from the front of the cabinet. Removing the memory chassis air intake grille exposes the memory chassis circuit boards. After circuit malfunctions have been localized, the memory assembly and logic boards can be pulled out of their slide mountings for replacement. Memory stacks and power supplies can be removed and replaced with simple tools.

Maintenance diagnostic routines for quick malfunction localization are available as built-in and program loaded routines. The built-in microcoded diagnostic routine tests the basic micro instructions, control memory, I/O, lower 8K of memory, I/O instructions and the emulate instruction. The program loaded diagnostic routines are more comprehensive and can be loaded from external memory into computer memory as needed.

FUNCTIONAL ARCHITECTURE

Main Memory

Main memory is an assembly of up to eight 8192 sixteen-bit word boards of magnetic core storage with a 750-nanosecond read-write cycle time. One such board, with its reading, writing, and addressing circuits, is the basic increment for memory size selection and expansion.

Memory Interface

A single memory interface handles the transfer of information between the processor and main memory and between the memory and the IOC. The input and output functions that are carried out by the IOC are transferred through this interface. Also, access to the NDRO (bootstrap) memory is made through the interface. All 65,536 words of memory may be directly addressed by both the IOC and processor.

DMA Interface Capability

The computer design permits the addition of a direct memory access (DMA) capability. This option allows a customer-provided external controller to read from and write into main memory. The option provides a second memory interface in addition to the normal processor-to-memory interface which remains unchanged.

Overall processing throughput can be increased by incorporating the DMA feature which provides an additional access port to each of the two 32K memory banks. This option adds up to 65 nanoseconds per memory reference but allows an external device to communicate with one memory bank during the same time period the CP/IOC communicates with the other. Thus, special purpose equipment, that requires a direct access to memory, can be utilized in the UYK-20 system because it need not share memory time with the running program. The two ports in each memory bank operate on a priority basis. Simultaneous requests for memory access by the CP/IOC and the DMA user are separated to give priority service to the CP/IOC port. Any device connected to a DMA port must have its own matching memory interface logic.

Memory Address Allocations

Main memory is used for storage of programs, constants, and data. Relative addressing to 1024 word pages is a built-in feature to aid in the development of relocatable software. All locations are accessible to the programs at random and to all sections of the computer. Some locations are given special assignments which programs must respect and provide for their contents. These assigned addresses may be used for general storage when the feature associated with the assignment is not implemented. Table 1 lists the assigned octal addresses.

TABLE 1. ASSIGNED MEMORY ADDRESSES

Assignment	Addresses (octal)
NDRO Memory	00-77 and 300-477
For Processing	
Class III Interrupts	110-117
Class II Interrupts	120-127
Class I Interrupts	130-137
For IOC Operation	
Command Cells	140-141
External Interrupt Word Storage (IOC)	200-217

Memory Addressing

All locations in main memory up to 65,536 words are directly addressable by the central processor and input/output controller. Both the sequential and random access methods are employed. Addresses are specified relative to 2000_8 word pages. The lower order ten-bits of the relative address specify the address of a word within a 2000_8 word page of main memory. The most significant six bits (index) select one register, from a group of 100_8 page address registers ($00\text{-}77_8$), that contains the base address of a specific 2000_8 word page within main memory. Figure 13 illustrates the final address generated from the two fields of the relative address. Any operation that stores a word in main memory also sets the most significant bit of the page register that was used in generating the memory address.

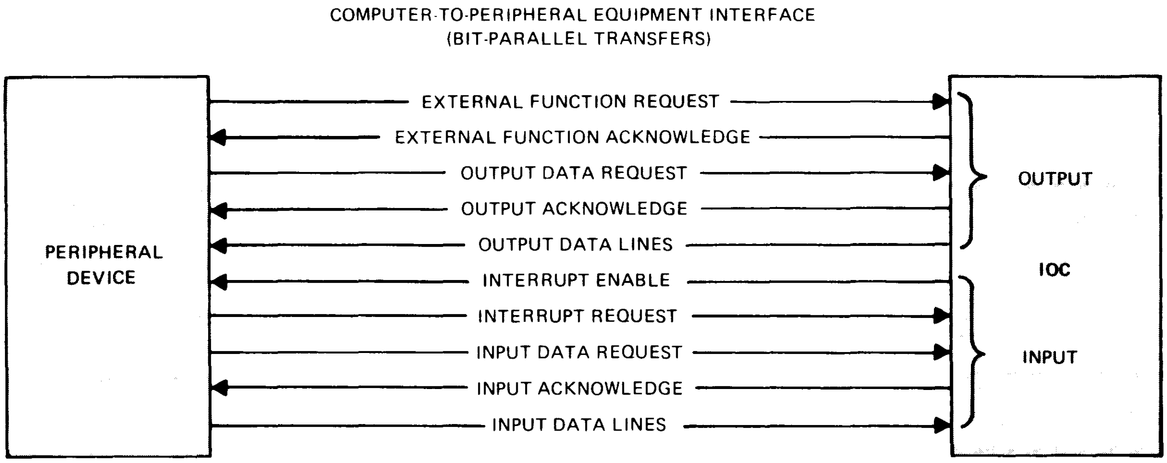


Figure 7. IOC-Peripheral Equipment Interface

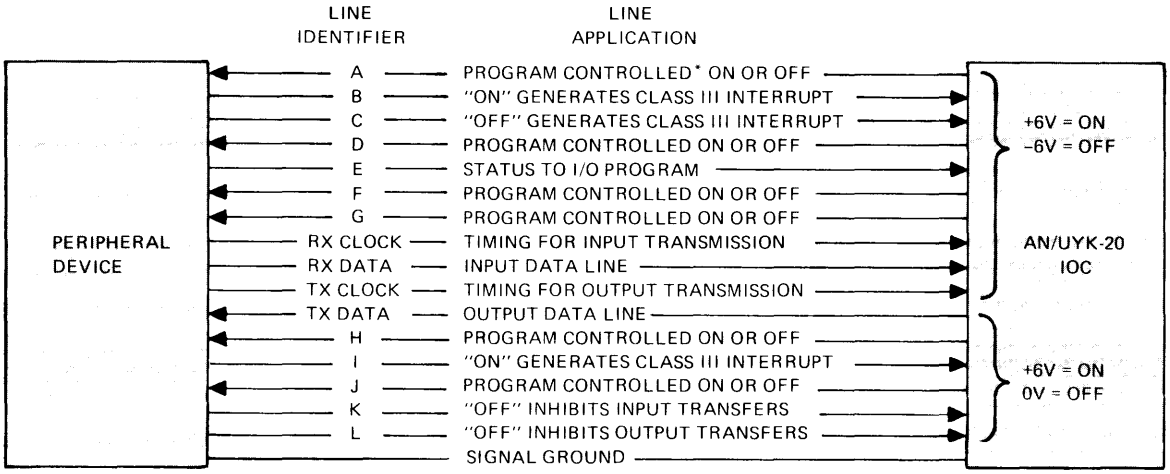
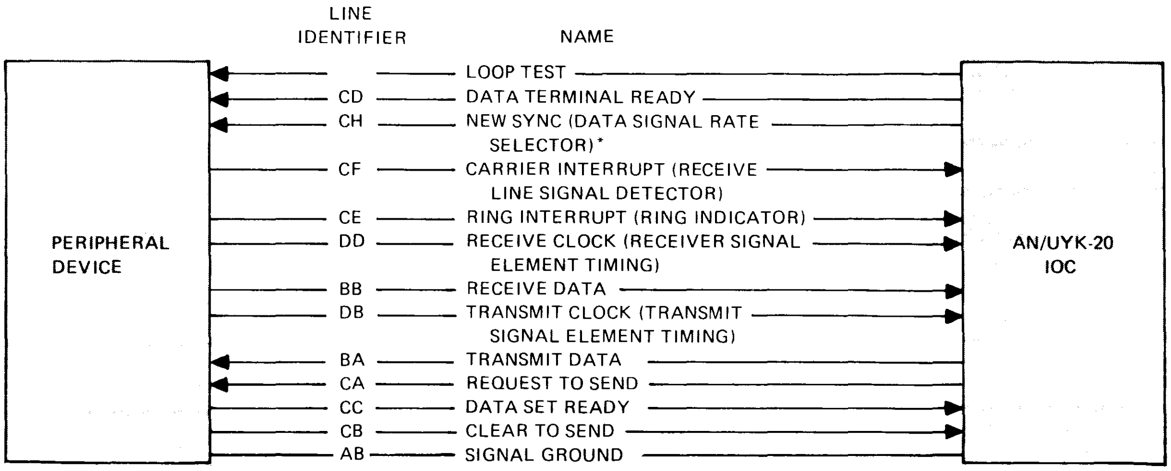

LINE IDENTIFIER
LINE APPLICATION
PERIPHERAL DEVICE
A
PROGRAM CONTROLLED* ON OR OFF
B
"ON" GENERATES CLASS III INTERRUPT
C
"OFF" GENERATES CLASS III INTERRUPT
D
PROGRAM CONTROLLED ON OR OFF
E
STATUS TO I/O PROGRAM
F
PROGRAM CONTROLLED ON OR OFF
G
PROGRAM CONTROLLED ON OR OFF
RX CLOCK
TIMING FOR INPUT TRANSMISSION
RX DATA
INPUT DATA LINE
TX CLOCK
TIMING FOR OUTPUT TRANSMISSION
TX DATA
OUTPUT DATA LINE
H
PROGRAM CONTROLLED ON OR OFF
I
"ON" GENERATES CLASS III INTERRUPT
J
PROGRAM CONTROLLED ON OR OFF
K
"OFF" INHIBITS INPUT TRANSFERS
L
"OFF" INHIBITS OUTPUT TRANSFERS
SIGNAL GROUND
+6V = ON
−6V = OFF
AN/UYK-20
IOC
+6V = ON
0V = OFF

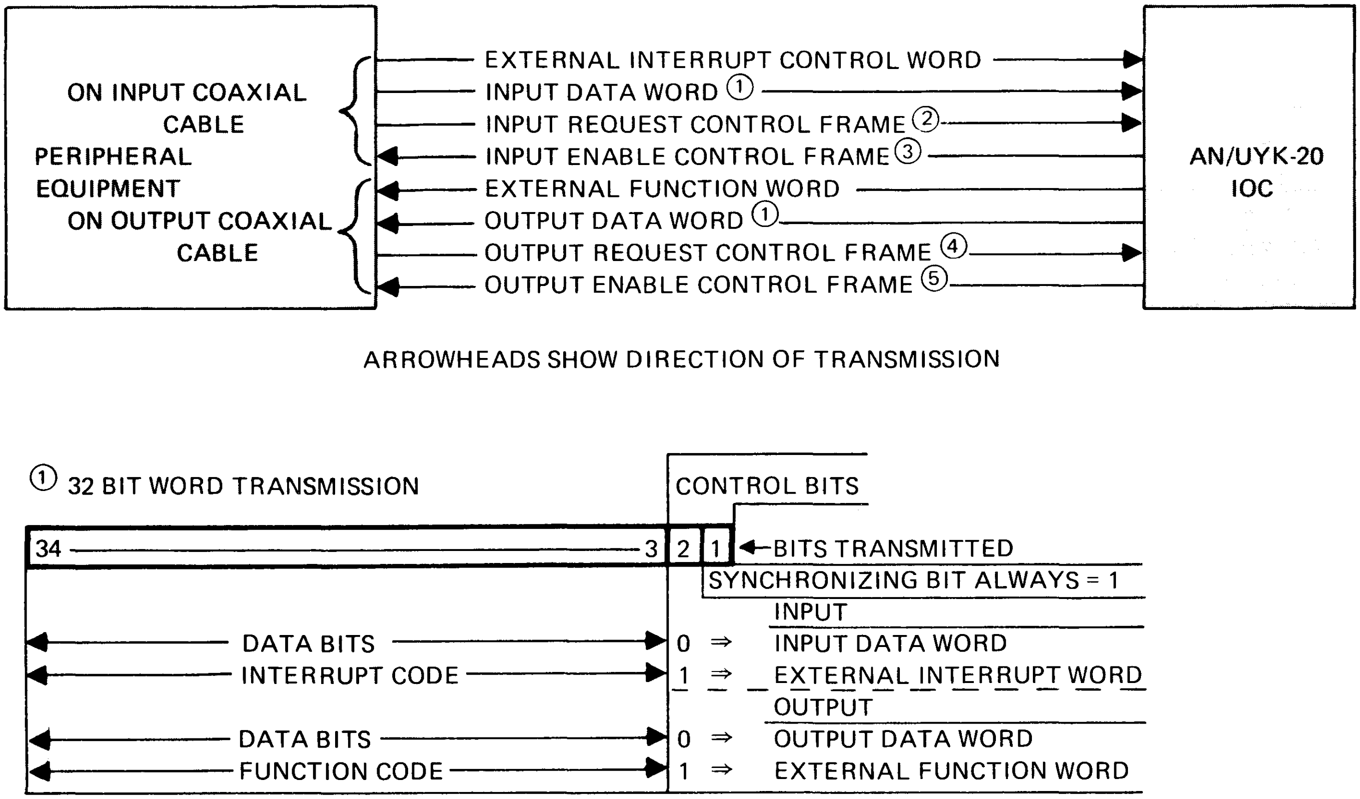
Figure 8. MIL-STD-188C and VACALES Interface


LINE IDENTIFIER
NAME
PERIPHERAL DEVICE
LOOP TEST
CD
DATA TERMINAL READY
CH
NEW SYNC (DATA SIGNAL RATE SELECTOR)*
CF
CARRIER INTERRUPT (RECEIVE LINE SIGNAL DETECTOR)
CE
RING INTERRUPT (RING INDICATOR)
DD
RECEIVE CLOCK (RECEIVER SIGNAL ELEMENT TIMING)
BB
RECEIVE DATA
DB
TRANSMIT CLOCK (TRANSMIT SIGNAL ELEMENT TIMING)
BA
TRANSMIT DATA
CA
REQUEST TO SEND
CC
DATA SET READY
CB
CLEAR TO SEND
AB
SIGNAL GROUND
AN/UYK-20
IOC

*(   ) RS232C TERMINOLOGY

*PROGRAM CONTROLLED LINES ARE ASSIGNED FUNCTIONS ACCORDING TO THE NEED OF THE PARTICULAR DEVICE CONNECTED TO THE CHANNEL.

Figure 9. EIA-STD-RS232 Interface


ON INPUT COAXIAL CABLE
PERIPHERAL EQUIPMENT
ON OUTPUT COAXIAL CABLE
EXTERNAL INTERRUPT CONTROL WORD
INPUT DATA WORD ①
INPUT REQUEST CONTROL FRAME ②
INPUT ENABLE CONTROL FRAME ③
EXTERNAL FUNCTION WORD
OUTPUT DATA WORD ①
OUTPUT REQUEST CONTROL FRAME ④
OUTPUT ENABLE CONTROL FRAME ⑤
AN/UYK-20 IOC
ARROWHEADS SHOW DIRECTION OF TRANSMISSION
① 32 BIT WORD TRANSMISSION
CONTROL BITS
34
3
2
1
BITS TRANSMITTED
SYNCHRONIZING BIT ALWAYS = 1
INPUT
DATA BITS
0 ⇒ INPUT DATA WORD
INTERRUPT CODE
1 ⇒ EXTERNAL INTERRUPT WORD
OUTPUT
DATA BITS
0 ⇒ OUTPUT DATA WORD
FUNCTION CODE
1 ⇒ EXTERNAL FUNCTION WORD

3 BIT CONTROL FRAME — **3 2 1 ←BITS TRANSMITTED**

SYNCHRONIZING BIT ALWAYS = 1

② INPUT REQUEST CONTROL FRAME	**0 0**	⇒	**NOT USED**
	0 1	⇒	**INPUT DATA REQUEST (IDR)**
	1 0	⇒	**EXTERNAL INTERRUPT REQUEST (EIR)**
	1 1	⇒	**IDR AND EIR**
③ INPUT ENABLE CONTROL FRAME	**1 1**	⇒	**INPUT DATA ENABLE (IDE) AND EXTERNAL INTERRUPT ENABLE (EIE)**
④ OUTPUT REQUEST CONTROL FRAME	**0 0**	⇒	**NOT READY**
	0 1	⇒	**OUTPUT DATA REQUEST (ODR)**
	1 0	⇒	**EXTERNAL FUNCTION REQUEST (EFR)**
	1 1	⇒	**ODR AND EFR**
⑤ OUTPUT ENABLE CONTROL FRAME	**1 1**	⇒	**OUTPUT DATA ENABLE (ODE) AND EXTERNAL FUNCTION ENABLE (EFE)**

Figure 10. NTDS Serial Channel Transmissions

NDRO Memory Feature

A block of 192 nondestructive readout (NDRO) memory words is provided in the CP. The programs contained in the NDRO memory are fixed at the time of manufacture by the ordering document and cannot be changed by computer read and write operations. The NDRO memory can be changed in the field by a simple and easy card replacement. Addresses assigned to NDRO memory (octal locations 00 through 77 and 300 through 477) parallel similarly numbered relative main memory addresses. A specific bit in status register #1 controls the access to NDRO memory or to corresponding locations in main memory (see Table 1).

NDRO memory is a convenient storage for programs that will always be available to the computer. These might include an initial load routine which loads a program and checks the validity of the program load and an inspect and change routine.

Input/Output Controller

An input/output controller (IOC) relieves the central processor of the computer-peripheral communication burden and permits integrating the AN/UYK-20 into a system that has an input/output equipment complex established. When an input or output related function is required, the main program initiates an I/O chain that performs the input or output operations according to a stored program defined for the specified channel.

The IOC communicates with external units in the system over the IOC-Peripheral equipment interface (see Figures 7, 8, 9 and 10) and with memory on the CP/IOC-memory bus.

Each IOC-peripheral equipment parallel interface consists of one input channel and one output channel connected to the external device by two respective cables. Output channels are used to transmit data and external functions (or commands) to the peripheral device. Input channels are used to receive data or interrupt codes from the external device.

The complete IOC-peripheral equipment parallel interface has 1, 2, 3, or 4 groups of 4 input and 4 output, 16-bit channels. An 8-bit byte, 16-bit word or a 32-bit double word, parallel interface can be utilized for data transfers. The 32-bit parallel transfers use two 16-bit channels (n and n + 4) where n = 0-3, 10-13 in a dual channel communication mode. All input/output activity is asynchronous, and the timing is dependent on the speed of the peripheral device compatible with MIL-STD-1397. Serial interfaces for communication circuits are available in asynchronous and synchronous channels. The IOC performs the necessary serial-to-word and word-to-serial conversions. Serial channels designed to the EIA-STD-RS232C, MIL-STD-188C or NTDS serial in MIL-STD-1397 are available in 2-channel groups while the VACALES interface is available in 4-channel groups.

Interface voltage levels on parallel transfer channels can be supplied in a -15 volt level, a -3 volt level or a +3.5 volt level. See Table 2 for transfer rate.

The memory interface provides the IOC with an access to memory on a time-share basis with the CP. Priority is given to the IOC in case of simultaneous requests.

General Registers

The standard central processor has one set of 16, high speed, 16-bit, general purpose registers designated R_0 through R_{17} (octal) and an instruction set tailored to their manipulation. A second set of 16 general registers is optional. The general registers provide for extrememly rapid processing of parameters or data by decreasing the number of required main memory references. Contents of any number of registers in a set can be changed by one simple instruction which saves program space and 50% of the time to execute the load and store process. With the availability of such registers, programs can be constructed with a greater proportion of single word (RR format) instructions which decreases both program storage space and program execution time.

A general register can be used as 1) an accumulator for arithmetic, shift, and logical functions; 2) an index register for address and operand modification; and/or 3) a temporary storage location for addresses, operands, etc.

TABLE 2. TYPICAL I/O TRANSFER RATES

PARALLEL CHANNELS (words per sec per group)		
Interface & Voltage (Type)	OUTPUT	INPUT
–15 NTDS single channel	31K	27K
–15V NTDS dual channel (32-bit)	30K	26K
+3.5 ANEW and –3.0 NTDS single channel	137K	159K
+3.5 ANEW and –3.0 NTDS dual channel (32-bit)	123K	135K
NTDS SERIAL CHANNEL 125,000 32-bit words/sec.		
EIA-STD-RS 232C, MIL-STD-188C & VACALES SERIAL CHANNELS – (Bits per sec)		
Asynchronous channel	2,400, 1,200, 600, 300, 150 or 75	
Synchronous channels	up to 9600	

The word format and the operation code of an instruction, that requires a general register reference, define the use of the register; one or both register designator fields (a, m) in that instruction select the register or registers in a set.

When the second general register set is included in the processor a program-controlled 1-bit field in a status register selects the set that is used for processor operations. The additional general register set provides greater freedom and increased processing speeds in applications that employ heavy interrupt processing and those that utilize the multi-programming technique. Programs that are called repeatedly for operation, and those in applications requiring rapid task changes (i.e., switching from one to another) can be more independent when a second set is available. An executive program, for example, that is assigned a set for its own use, need not store the contents of general registers used by worker programs every time it assumes control or when it is requested to process an interrupt.

Program Address Register

The program address register, P, holds the address of the next instruction to be executed in a program sequence. Its contents are advanced by one each time a single-length (16-bit) instruction is executed and by two for a double-word instruction. Instructions that cause program transfers (jumps) load the P-register with the entry address of the program that receives control. The variety of ways the P-register contents can be manipulated by instructions provides for efficient program segmentation and for effective use of re-entrant routines.

Real-Time Clock and Monitor Clock Feature

The RTC-MON clock feature provides two, program-controlled interrupts via two high-speed registers; one 32-bit register used as RTC count-up storage and the other a 16-bit register as MON clock count-down storage. This feature and associated controlling instructions are useful for program timing and for synchronizing program segments with real-time events. A 1 Khz RTC oscillator, which has an accuracy of ±2 Hz in 10 seconds, runs continuously and controls the counting speed of both registers. An external clock oscillator with a frequency in the 0 to 50 Khz range may be used instead of the internal oscillator.

Breakpoint Feature

A convenient debugging aid is a breakpoint register that can be loaded and controlled manually by the operator. Breakpoint is a function that stops the computer when it encounters a read or write reference to an address that matches the entry in the breakpoint register. The operator identifies the breakpoint register function as a read operation or as a write operation, or both, by setting two toggle switches on the operator/maintenance panel.

Power Failure Protection Feature

The power fault and automatic recovery feature provides a systematic and safe shut-down and recovery capability in the event that any power to modular sections falls below an operable level. Sensors monitor the power supply voltage and when an "out-of-tolerance" voltage is detected, a voltage out signal is generated. When the CP senses the voltage out signal, it generates a power fault interrupt, suspends the normal program sequence, transfers control to the user's software power fault interrupt routine and allows a minimum of 250 microseconds to arrive at an orderly termination. The power fault interrupt routine should store the contents of all working registers and terminate in a jump instruction (operation code 40 RX format, a=6 and m=0) that jumps to itself as long as the voltage "out of tolerance" signal is present. After the signal is removed (power returns to normal), the instruction allows the routine to continue and to restore the working registers and then return control to the program that was interrupted.

If the power continues to drop below operating limits, a CP master clear results in a shut down. When power is reapplied and the AUTO START is selected, the CP generates an auto start signal that causes execution of the instruction located at address 000 of the NDRO memory.

The automatic shut-down and recovery routine is a user's software responsibility but should normally perform as described and also be compatible with the Class I interrupt codes assigned.

Status Register

Status register #1 is a 16-bit, high-speed register that provides a dynamic picture of certain processing states. Fields in the status word can be examined or changed by programmed instructions when necessary. During a program interruption (interrupt processing), the computer control logic stores the status word and reloads the register before transferring to the interrupt subroutine. When reentering the interrupted program, the computer status existing at the time of interruption is reinstated. This allows the program to continue as though it were not interrupted.

Status register #2 is used to control direct and indirect addressing processes and to hold memory resume interrupt and I/O instruction fault data.

FUNCTIONAL OPERATION

Computer Status Control

The format for status register #1 is divided into fields that indicate computer status, interrupt status, and conditions resulting from Arithmetic section operations (see Figure 11).

Details of field designators are as follows:

a) Bit 0 = 0, disables DMA
   Bit 0 = 1, enables DMA

b) Interrupt enable designator:

   bit 1 = Class III
   bit 2 = Class II
   bit 3 = Class I

   When the bit is set, the respective class interrupt is enabled (can be honored).

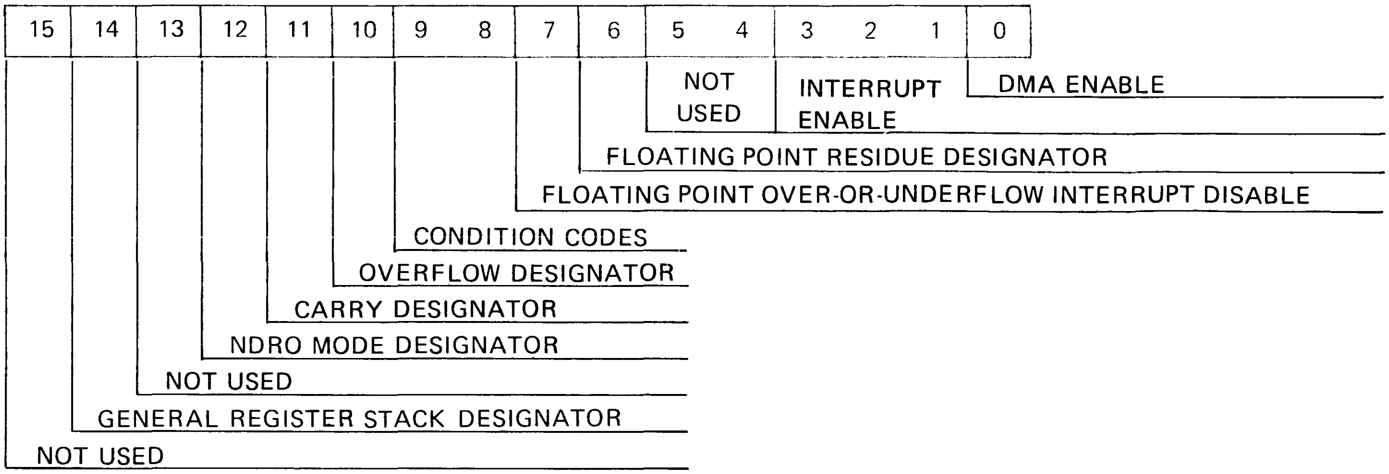


Figure 11. Status Register No. 1 Format

c) Floating-point residue designator bit 6 = 0 specifies that the residue, in a floating-point arithmetic operation, will not be saved. Bit 6 = 1 specifies that the residue will be saved.

d) Control bit 7 = 0 sets the floating-point overflow or underflow interrupt when that result occurs. If bit 7 = 1, no interrupt is generated on floating-point overflow or underflow.

e) The condition code (bits 9 and 8) indicates the results of arithmetic and compare instructions as shown in Table 3.

f) The overflow designator (bit 10) is set when an arithmetic or a shift operation produces a result that requires more bits than provided in a register.

g) The carry designator (bit 11) is set when an arithmetic operation generates a carry beyond the most significant bit in the register.

h) The NDRO Mode (bit 12) directs the CP to select memory as follows for addresses 00 through 77 and 300 through 477:

Bit 12 = 0, Use NDRO memory
Bit 12 = 1, Use main memory

i) The general register stack designator (bit 14) specifies the stack of 16 general registers that will be selected by the a- and m-designators in the instruction word. Bit 14 = 0 selects the standard stack and bit 14 = 1 selects the optional stack.

Status register bit content can be set or cleared by executing the load status register instruction or the load program status word instruction and can be initialized for a class interrupt processing subroutine by loading the "Load Status Register #1" memory location assigned to that particular interrupt class.

Instructions

Instructions defining operations for the AN/UYK-20 are designed to maximize circuit effectiveness in attaining high-speed computer functions. The large set of flexible and comprehensive, single and double-word instructions place the computer far beyond the minicomputer capability.

Table 4 lists the instructions and the execution time for each in the applicable formats. Among the instructions in the total repertoire are many that speed up the capability of application programs and provide greater flexibility for programmers. These include:

The *biased fetch* instruction allows the central processor to check on the performance of tasks it assigns to the input/output chaining program.

A *reverse register* feature is useful in reversing a stream of data that is received from a communication system and must be transmitted to another system in reverse order.

The *scale factor shift* instruction provdes a left-shift function which positions the word for greatest significance and counts the number of digit positions shifted.

Condition Code		Indicated Results of	
Bit 9	Bit 8	Arithmetic Operation	Compare Operations
0	0	Zero	$R_a = R_m$ or Y
0	1	Not Zero and Positive	$R_a > R_m$ or Y
1	0	Not Used	Not Used
1	1	Not Zero and Negative	$R_a < R_m$ or Y

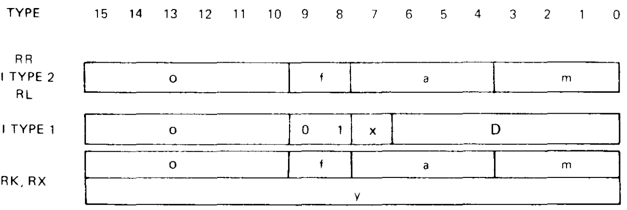
TABLE 3. CONDITION CODE INDICATIONS

Local jump instructions are storage space and time savers in all systems designed around the natural "looping" method of programming. These saving benefits are apparent in both the program generation and job processing phases.

The *jump and link* instructions fill the requirement for linking to re-entrant routines. Because these routines cannot be changed internally, the linking is done externally either through general registers or main memory.

Set bit, clear bit and *test bit* instructions provide a fine grain examination and change capability that is useful in real-time communications. Interacting tasks that communicate by flags and status words benefit highly by this flexible bit-handling feature.

Figure 12 defines the 16-bit instruction word formats and the 32-bit, two-word instruction formats. Single-word formats can be used when operands are manipulated in high-speed general registers. Double word formats are used for operations requiring direct, indirect memory references with or without indexing and those that provide programmers the convenience of listing 16-bit constants in-line with instructions. Programs that can be constructed with a high ratio of one-word instructions to two-word instructions greatly increase the computing speed and also-occupy less memory space.



DEFINITION OF FIELDS

o OPERATION (FUNCTION) CODE
f FORMAT DESIGNATOR
00 ⇒ FORMAT RR, REGISTER TO REGISTER OR RL-1 FORMAT
01 ⇒ FORMAT RI, REGISTER INDIRECT MEMORY OR RL 2 FORMAT
10 ⇒ FORMAT RK, REGISTER-LITERAL CONSTANT OR RL-3 FORMAT
11 ⇒ FORMAT RX, REGISTER-INDEXED ADDRESS, CONSTANT OR RL-4 FORMAT
a GENERAL REGISTER OR SUBFUNCTION DESIGNATOR
m GENERAL REGISTER OR SUBFUNCTION DESIGNATOR
4-BIT UNSIGNED LITERAL CONSTANT IN RL FORMAT
xD SIGNED DEVIATION VALUE (TWO'S COMPLEMENT)
y ADDRESS OR ARITHMETIC CONSTANT

Figure 12. Instruction Word Formats

Math Pac Option

An additional set of arithmetic instructions, that improves programming versatility and increases computational throughput, is provided by the math pac hardware. The following functions are included:

A *square root* instruction is useful in scientific applications.

Hardware trigonometric and hyperbolic functions are provided in the coordinate conversion (CORDIC) feature that uses the general registers for parameter manipulation. Some functions provided by the one instruction include trigonometric and hyperbolic rotate, trigonometric and hyperbolic vector, $\log_e x$, exponential, polar to Cartesian conversion, sin θ and cos θ. Other functions related to these mathematical processes may be obtained by proper choice of input parameters.

Floating point instructions executed by the microprogrammed controller are much faster than ordinary floating point subroutines in the system programs.

Double precision multiply and divide instructions allow for more direct processing of double length operands and for greater precision.

Instruction Addressing

A program address register (P) is an incremental counter in the CP that specifies the address of the next instruction to be executed by the CP. As an instruction is read from memory, the register is advanced in preparation for reading the next sequential instruction address in memory. Executing a single-word instruction advances the register by one and a double-word instruction advances it by two. Any jump instruction executed with its jump condition satisfied changes the address in P and a new program sequence begins at that address. *Local jump* instructions, however, limit the change to the address in P to +177/-200 octal locations.

Input/output operations use chain address pointer locations in control memory as instruction address

TABLE 4. REPERTOIRE OF INSTRUCTIONS

OCTAL CODE			DESCRIPTION	EXECUTION TIME *(MICROSECONDS)				NOTE REF.
o	a	m		RR	RI	RK	RX	
00			Diagnostic Return	.7	–	–	–	
00			Byte Load	–	–	–	2.25	
01			Load	.75	1.5	1.5	2.25	
02		0	Make Positive	1	–	–	–	
		1	Make Negative	1	–	–	–	
		2	Round R_a	1	–	–	–	
		4	Two's Complement Single	1	–	–	–	
		5	Two's Complement Double	1	–	–	–	
		6	One's Complement Single	1	–	–	–	
		10	Increase R_a by 1	1	–	–	–	
		11	Decrease R_a by 1	1		–	–	
		12	Increase R_a by 2	1	–	–	–	
		13	Decrease R_a by 2	1	–	–	–	
02			Load Double	–	2.25	–	3.0	
03		0	Executive Return	8.0	–	–	–	
		1	Store Status Register 1	.9	–	–	–	
		2	Store Status Register 2	.9	–	–	–	
		3	Store RTC Lower	.9	–	–	–	
		4	Load P	1.2	–	–	–	
		5	Load Status Register 1	1.65	–	–	–	
		6	Load Status Register 2	.9	–	–	–	
		7	Load RTC Lower	.9	–	–	–	
		10	Enable RTC	1.2	–	–	–	
		11	Disable RTC	1.2	–	–	–	
		12	Load and Enable MON. Clock	1.35	–	–	–	
		13	Disable MON. Clock	.9	–	–	–	
		14	Load RTC Double	1.5	–	–	–	
		15	Store RTC Double	1.5	–	–	–	
		16	Enable RTC Interrupt	.9	–	–	–	
		17	Disable RTC Interrupt	.9	–	–	–	
03			Load Multiple	–	–	–	1.5	1
04		0	Square Root	9.5	–	–	–	
		1	Reverse Register	6.3	–	–	–	
		2	Count Ones	7.2	–	–	–	
		3	Scale Factor Shift	3.3	–	–	–	
04			Byte Load and Index by 1	–	–	–	2.25	
05			Set Bit	1.5	–	–	–	
05			Load and Index by 1	–	1.5	–	2.25	
06			Clear Bit (Zero Bit)	1.5	–	–	–	
06			Load Double and Index by 2	–	2.55	–	3.3	
07			Test Bit	1.8	–	–	–	
07			Load PSW	–	3.0	–	3.75	
10			Logical Right Single Shift	1	–	1.7	–	
10			Byte Store	–	–	–	2.4	
11			Algebraic Right Single Shift	1	–	1.7	–	
11			Store	–	1.7	–	2.4	
12			Logical Right Double Shift	2.6	–	3.2	–	
12			Store Double	–	2.4 *	–	3.2	
13			Algebraic Right Double Shift	2.6	–	3.2	–	
13			Store Multiple	–	–	–	1.4	2
14			Algebraic Left Single Shift	1	–	1.7	–	
14			Byte Store and Index by 1	–	–	–	2.4	
15			Circular Left Single Shift	1	–	1.7	–	
15			Store and Index by 1	–	1.7	–	2.4	
16			Algebraic Left Double Shift	2.7	–	3.3	–	
16			Store Double and Index by 2	–	2.6	–	3.3	
17			Circular Left Double Shift	2.4	–	3.0	–	
17			Store Zeros	–	1.7	–	2.4	
20			Subtract	.75	1.5	1.5	2.25	
21			Subtract Double	1.7	2.25	–	3.0	
22			Add	.75	1.5	1.5	2.25	
23			Add Double	1.5	2.25	–	3.0	
24			Compare	.9	1.5	1.7	2.25	
25			Compare Double	1.7	2.25		3.	
26			Multiply	3.8	4.	4.4	4.6	
27			Divide	6.8	7.	7.4	7.5	
30			AND	.75	1.5	1.5	2.25	
31			OR	.75	1.5	1.5	2.25	
32			Exclusive OR	.75	1.5	1.5	2.25	
33			Masked Substitute	1.4	1.5	2.0	2.25	
34			Compare Masked	1.5	1.7	2.1	2.4	
35			I/O Command	4.	–	–	–	3
35			Biased Fetch	–	2.25	–	3.0	
35			Execute Remote	–	–	1.5	–	4
37		0	Vector – Trig. Mode	11.8	–	–	–	
		1	Rotate – Trig. Mode	11.8	–	–	–	
		2	Vector – Trig. Mode with Prescale	15.3	–	–	–	
		3	Rotate – Trig. Mode with Prescale	15.3	–	–	–	
		4	Vector – Hyperb. Mode	11.8	–	–	–	
		5	Rotate – Hyperb. Mode	11.8	–		–	
		6	Vector – Hyperb. Mode with Postscale	15.3	–	–	–	
		7	Rotate – Hyperb. Mode with Postscale	15.3	–	–	–	
40	0		Jump CC Zero/Equal	1.1		1.7	2.4	
	1		Jump CC Not Zero/Not Equal	1.1	–	1.7	2.4	
	2		Jump CC Pos/Greater Than or Equal	1.1	–	1.7	2.4	
	3		Jump CC Neg/Less Than	1.1	–	1.7	2.4	
	4		Jump on Overflow	1.1	–	1.7	2.4	
	5		Jump on Carry	1.1	–	1.7	2.4	
	6		Jump Power Out	1.1	–	1.7	2.4	
	7		Jump Bootstrap 2	1.1	–	1.7	2.4	
	10		Jump	1.1	–	1.7	2.4	
	11		Jump Stop	1.1	–	1.7	2.4	
	12		Jump Stop Key 1	1.1	–	1.7	2.4	
	13		Jump Stop Key 2	1.1	–	1.7	2.4	
40			Local Jump	–	#1.2	–	–	
41			Index Jump	1.4	–	2.1	2.25	
41			Local Jump Indirect	–	#2.0	–	–	
42			Jump and Link Register	1.2	–	1.2	2.25	
43			Local Jump and Link Memory	–	#2	–	–	
43			Jump and Link Memory	–	–	2.9	3.2	
44			Jump Register Zero	1.4	–	2.1	2.25	
44			Local Jump Equal	–	#1.2	–	–	
45			Jump Register Not Zero	1.4	–	2.1	2.25	
45			Local Jump Not Equal	–	#1.2	–	–	
46			Jump Register Positive	1.4	–	2.1	2.25	
46			Local Jump Greater Than or Equal	–	#1.2	–	–	
47			Jump Register Negative	1.4	–	2.1	2.25	
47			Local Jump Less Than	–	#1.2	–	–	
50			Floating Point Subtract	7.7–17.4	7.7–17.4	–	7.7–17.4	6
51			Floating Point Add	7.7–17.4	7.7–17.4	–	7.7–17.4	
52			Floating Point Multiply	15.2–18.9	15.2–18.9	–	15.2–18.9	
53			Floating Point Divide	7.7–17.7	7.7–17.7	–	7.7–17.7	
54			Load Address Register	1.8	2.6	–	–	
54			Load Address Register Multiple	–	–	–	3.0	1
55			Store Address Register	1.8	2.6	–	–	
55			Store Address Register Multiple	–	–	–	3.0	2
56			Double Multiply	5.5–15.3	5.5–15.3	–	5.5–15.3	
57			Double Divide	17.6 21.0	17.6–21.0	–	17.6–21.0	
60			Logical Right Single Shift	RL-1		1.3		
			Algebraic Right Single Shift	RL-2		1.3		
			Logical Right Double Shift	RL-3		2.8		
			Algebraic Right Double Shift	RL-4		2.8		
61			Algebraic Left Single Shift	RL-1		1.3		
			Circular Left Single Shift	RL-2		1.3		
			Algebraic Left Double Shift	RL-3		2.8		
			Circular Left Double Shift	RL-4		2.8		
62			Subtract	RL-1		0.9		
			Subtract Double	RL-2		1.8		
			Add	RL-3		0.9		
			Add Double	RL-4		1.8		
63			Load	RL-1		0.9		
			Compare	RL-2		1.2		
			Multiply	RL-3		4.2		
			Divide	RL-4		7.4		
				RR	RI	RK	RX	
64			Byte Subtract	–	–	–	2.25	
65			Byte Add	–	–	–	2.25	
66			Byte Compare	–	–	–	2.25	
67			Reserved	–	–	–	–	
67			Byte Compare and Index by 1	–	–	–	2.25	
70		0	Master Clear	30.0	–	–	–	
		4	Enable All External Interrupts	30.0	–	–	–	
		5	Disable All External Interrupts	30.0	–	–	–	
		6	Enable All External Monitors	30.0	–	–	–	
		7	Disable All External Monitors	30.0	–	–	–	
		10	Master Clear Chan a	2.0/2.25	–	–	–	5
		14	Enable Chan a Ext. Int.	2.0/2.25	–	–	–	
		15	Disable Chan a Ext. Int.	2.0/2.25	–	–	–	
		16	Enable Chan a Ext. Mon.	2.0/2.25	–	–	–	
		17	Disable Chan a Ext. Mon.	2.0/2.25	–	–	–	
70			Initiate I/O Transfer (Chain)	–	–	–	4.5	
71		2	Initiate Input Chain (Comm)	–	–	2.25	–	
		6	Initiate Output Chain (Comm)	–	–	2.25	–	
71			Load Control Memory (Chain)	–	–	2.25	3.0	
71			Write (Load) Control Memory (Comm)	–	–	–	3.0	
72			Read (Store) Control Memory (Comm)	–	–	–	2.0	
72			Store Control Memory (Chain)	–	–	–	3.0	
73	0		Halt (Chain)	1.5	–	–	–	
73	1		Interrupt (Chain)	1.5	–	–	–	
73	0		Clear Flag (Chain)	–	–	–	3.0	
73	1		Set Flag (Chain)	–	–	–	3.0	
74			Conditional Jump (Chain)	–	–	2.25	–	
75			Search for Sync/Set Mon/ Set Supp (Chain)	1.5	–	–	–	
76			Set/Clear Discretes (Comm)	1.5	–	–	–	
76			Set/Clear Discretes (Chain)	1.5	–	–	–	
76			Store Status (Comm)	–	–	–	3.0	
76			Store Status (Chain)	–	–	–	3.0	

Footnotes:

# RI, Type 1 Format
1 Add .75 times number of registers
2 Plus 1.1 times the number of registers
3 Plus I/O instruction
4 Plus remote instruction time
5 Command/chaining
6 Variation dependent on data
*Add 65 nanoseconds for DMA

counters. Each input and each output channel is assigned an address pointer that advances like a P-register as it executes its instructions in its program chain. If a jump is desired in the program, the chain address pointer is changed either by executing a load control memory instruction or the conditional jump instruction.

The a-, m-, and y-designator fields in the instructions define a variety of functional operations and parameters. Collectively, this variety offers a programmer much flexibility. For a computing system that interprets simple instruction formats with the variety of field assignments, speed is the reward. The general application for the a-, m-, and y-designator fields is described in the paragraphs following. Special assignments and uses are defined in the individual instruction description.

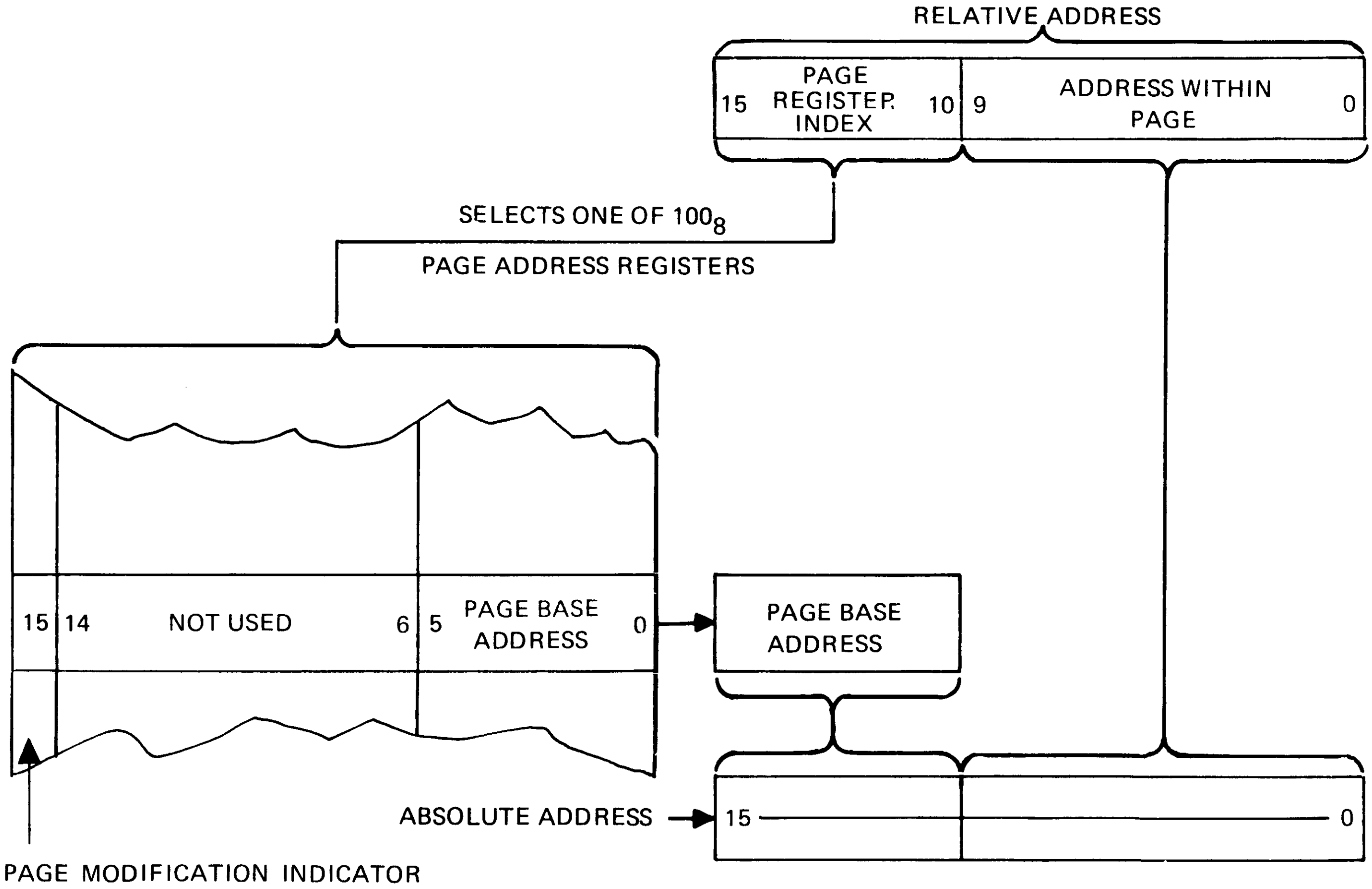
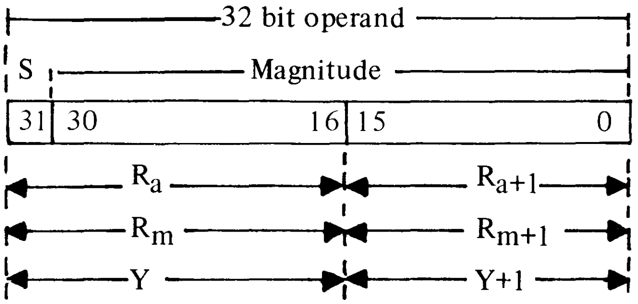


Figure 13. Memory Address Generation

Double Length Operands

Instructions that perform operations with 32-bit words (double-length) use two adjacent registers and memory locations for each operand. The word in R_a, R_m or in memory address Y, when selected by a "double length" instruction, is the most significant half of the operand and contains the sign bit for both words. The word in R_a+1, R_{m+1} or in memory address Y+1, respectively, is the least significant half of the operand.



Instruction Word Formats

RR Format instructions perform operations involving general registers; no main memory references are made for operands. The a- and m-designators select the general registers designated R_a and R_m, respectively, that are used in the operation.

RL Format instructions perform operations involving one or two general registers. The a-designator selects the general register designated R_a or two general registers designated R_a and R_{a+1}. The m-designator is an unsigned literal that is used in the operation.

RI Format, Type 1 instructions are local jump operations that either increase or decrease the contents of P by the value D in the instruction. The effective jump address Y = (P) + xD, where xD is the two's complement deviation value.

RI Format, Type 2 instructions perform operations that involve general registers and a main memory reference. The a- and m-designators select general registers designated R_a and R_m respectively. R_m, however, contains an address Y that is used for the main memory reference.

RK Format instructions are double-word instructions that are stored in two numerically adjacent memory locations. The first word contains the operation code and designator fields. The second word is a value y that may be used as a constant operand or address or as a modified constant or address. The a-designator selects a general register designated R_a. When m = 0, the operand or address Y equals y, no R_m is selected. m $\neq$ 0 selects a general register R_m; the operand Y equals y plus the contents of R_m – i.e., y is indexed by the contents of R_m. (Operand Y is used as an address in RK Format jump instructions and in the remote execute instructions).

RX Format instructions are two-word instructions that are stored in two numerically adjacent memory locations. The first word includes the a- and m-designators and the second word contains the y-value. RX format instructions perform byte (8-bit), whole word (16-bit) and double-word (32-bit) operations with general registers and memory references. The a-designator selects a general register, designated R_a, for all three types of operands.

The operand addressing process provides much programming flexibility. Direct addressing, indirect addressing or cascaded indirect addressing of operands may be selected with the RX format instructions. When the instruction m-designator equals zero, direct addressing without indexing is selected (address Y = y). Direct addressing with indexing is specified when the m-values 1 through 7, 11, 13, 15 or 17 (octal) are used (address Y = y + (R_m)). The general registers specified by these m-values contain the indexing modifier. When the m-value 10, 12, 14 or 16 is specified, corresponding indirect control fields in Status Register #2 are interpreted to generate the address of the operand or a pair of indirect words (IW1 & IW2) as illustrated in Figure 14.

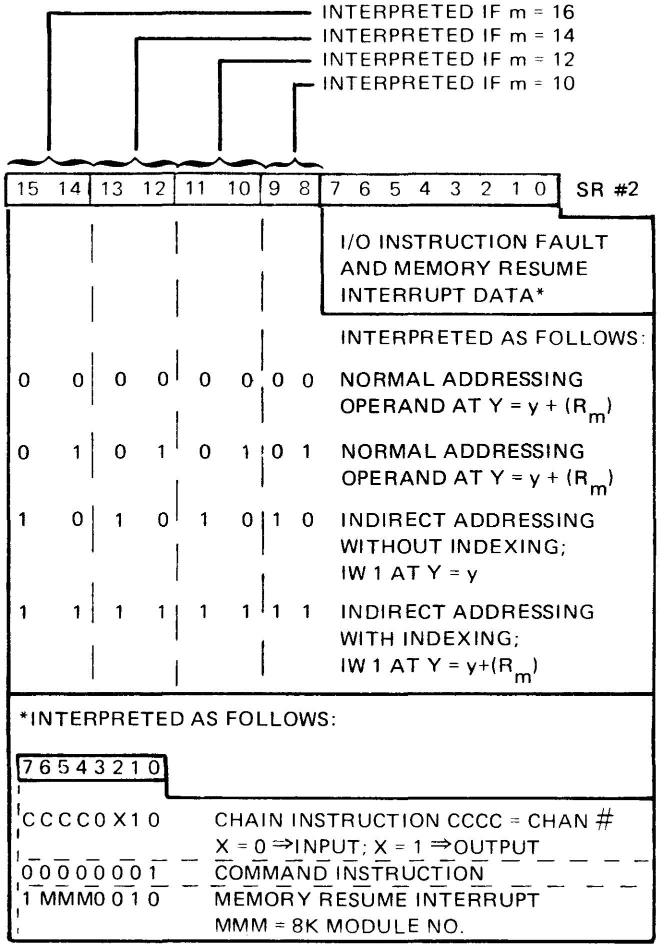
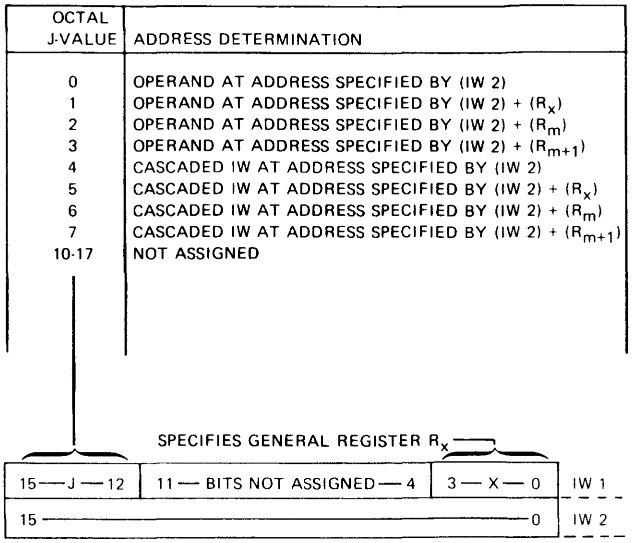

INTERPRETED IF m = 16
INTERPRETED IF m = 14
INTERPRETED IF m = 12
INTERPRETED IF m = 10
15 14 | 13 12 | 11 10 | 9 8 | 7 6 5 4 3 2 1 0
SR #2
I/O INSTRUCTION FAULT AND MEMORY RESUME INTERRUPT DATA*
INTERPRETED AS FOLLOWS:
0 0 0 0 0 0 0 0 NORMAL ADDRESSING OPERAND AT Y = y + (R_m)
0 1 0 1 0 1 0 1 NORMAL ADDRESSING OPERAND AT Y = y + (R_m)
1 0 1 0 1 0 1 0 INDIRECT ADDRESSING WITHOUT INDEXING; IW 1 AT Y = y
1 1 1 1 1 1 1 1 INDIRECT ADDRESSING WITH INDEXING; IW 1 AT Y = y+(R_m)
*INTERPRETED AS FOLLOWS:
7 6 5 4 3 2 1 0
C C C C 0 X 1 0 CHAIN INSTRUCTION CCCC = CHAN # X = 0 ⇒INPUT; X = 1 ⇒OUTPUT
0 0 0 0 0 0 0 1 COMMAND INSTRUCTION
1 MMM0 0 1 0 MEMORY RESUME INTERRUPT MMM = 8K MODULE NO.

Figure 14. Status Register No. 2 Format

If control bits in the field interpreted equal 00 or 01 (binary), direct addressing results as though m = 1 through 7, 11, 13, 15 or 17 (octal) – i.e., Y = y+(R_m). But when m = 10, 12, 14 or 16 and the field interpreted equals 10 or 11 (binary), the addresses Y and Y + 1 contain a pair of indirect words (IW) that are interpreted according to Figure 15.

Double-length operations are assigned even-numbered addresses or registers for operand references. The first, or most significant, half of an operand is in the even numbered location. From the even-numbered address or register specified by the instruction (i.e., R_a R_m and Y as applicable), the computer logic selects the next sequentially numbered address or register for the second half of the operand. Memory addresses for the first half of the double-length operand are formed like those for whole-word operands.



OCTAL J-VALUE	ADDRESS DETERMINATION
0	OPERAND AT ADDRESS SPECIFIED BY (IW 2)
1	OPERAND AT ADDRESS SPECIFIED BY (IW 2) + (R_x)
2	OPERAND AT ADDRESS SPECIFIED BY (IW 2) + (R_m)
3	OPERAND AT ADDRESS SPECIFIED BY (IW 2) + (R_{m+1})
4	CASCADED IW AT ADDRESS SPECIFIED BY (IW 2)
5	CASCADED IW AT ADDRESS SPECIFIED BY (IW 2) + (R_x)
6	CASCADED IW AT ADDRESS SPECIFIED BY (IW 2) + (R_m)
7	CASCADED IW AT ADDRESS SPECIFIED BY (IW 2) + (R_{m+1})
10-17	NOT ASSIGNED

Figure 15. Indirect Word Interpretation

Byte (8-bit, half word) operand addressing requires a byte identifier (B), i.e., the upper byte or the lower byte in memory.

B = 0 designates the most significant half word in address Y as the operand byte.

B = 1 designates the least significant half word in address Y as the operand byte.

The least significant (LSB) in the indexing register is used as the byte identifier and the value in the remaining bits is used as the index to generate the effective address as follows:

m = 0, address Y = y and B = 0

m = 1-7, 11, 13, 15 or 17 (octal)

$Y = y + \frac{(R_m)}{2}$ and B = LSB of (R_m)

When indirect addressing is used and

if j = 0, y = (IW2) and B = 0

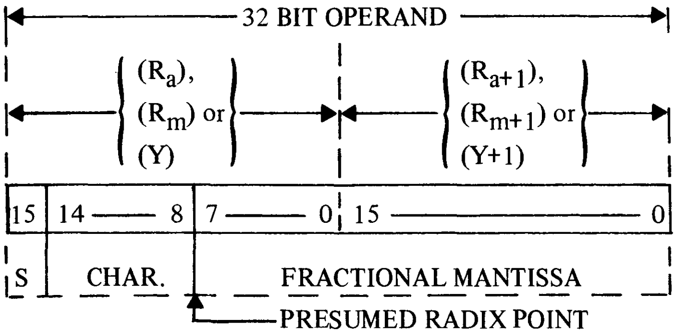
if j = 1, Y = (IW2) + $\frac{(R_x)}{2}$ and B = LSB of (R_x)

if j = 2, Y = (IW2) + $\frac{R_m)}{2}$ and B = LSB of (R_m)

if j = 3, Y = (IW2) + $\frac{(R_{m+1})}{2}$ and B = LSB of (R_{m+1})

Floating-Point Operands

Floating-point addition, subtraction, multiplication and division may be performed with a normalized result with or without a residue. The process uses a two-word operand in the format shown below. the format shown below.



a, m and address Y are even numbers

Word 1 of the operand, stored in R_a, R_m or memory address Y contains the algebraic sign (S) of the fractional mantissa, a biased characteristic in the range $0 \leqslant C \leqslant 177$ (octal) and the two most significant hexadecimal digits of the fraction. Word 2 of the operand, stored in R_{a+1}, R_{m+1}, or memory address Y+1, contains the four least significant hexadecimal digits of the fractional mantissa. A normalized floating-point number has a nonzero hexadecimal digit in the most significant four bits of the fractional mantissa. When a residue is requested by the program, the computer stores the result in R_a and R_{a+1} and stores the unused lower order digits (residue) in general registers R_{a+2} and R_{a+3} in floating-point data format.

A change of "one" in the characteristic represents one hexadecimal digit position shift (4 bits). Therefore, the magnitude (M) of a floating-point number is approximately $5.4 \times 10^{-79} \leqslant M \leqslant 7.2 \times 10^{75}$. A zero quantity is represented by a positive sign (0), a zero characteristic and a zero fractional mantissa.

Interrupts

The central processor can be interrupted in its execution of programs. Some interrupts are generated by events within the CP, some within the IOC, and some as interrupt requests by peripheral input or output devices. AN/UYK-20 system interrupts are classified in three priority levels. Interrupts within a class are assigned a priority rank within that class and an identifying code that is used by CP logic to select the appropriate processing routine from memory. Table 5 lists the interrupts, their classification and assigned identity codes. Higher priority is given to the class and the interrupt within the class that has the lower number. As each interrupt is honored, its class and all classes of a lower priority can be locked out by the reloaded status register until released by the processing subroutine. Thus an event in a higher priority class can interrupt a routine that is processing a lower priority class interrupt. The interrupt routine is held until the higher level is processed and then is allowed to continue.

RTC and MON clock registers can be loaded, read, enabled, or disabled under program control. When enabled by the appropriate instruction (code 03 RR Format, m = 10), the RTC register counts up at the rate of the RTC oscillator or external RTC input. As the register lower order 16 bits overflow (change from all ones to all zeros), the CP generates the RTC overflow interrupt (class II priority 5) and control is transferred to the appropriate processing routine. The RTC register continues to count-up until disabled by the *disable RTC* instruction (code 03 RR Format, m = 11). The RTC and MON clock do not advance when the machine is stopped.

Bit 8 of the RTC register has the added function of timing data or external function outputs on intercomputer channels. If the AN/UYK-20 holds a word in its output register longer than the time required to toggle bit 8 twice (receiving computer did not acknowledge), an intercomputer time-out interrupt is generated.

The MON clock register count-down function is enabled by executing the *load and enable monitor clock* instruction (code 03 RR Format, m = 12) which also loads the register with a starting point.

TABLE 5. INTERRUPT PRIORITY

Class	Priority Within Class	Interrupt	Binary Interrupt Code Generated
Class I, Hardware Errors	1	Power Fault	0000
	2	Memory Resume	0010
Class II, Software Interrupts	1	CP Instruction Fault†	0000
	2	I/O Instruction Fault†	0010
	3	Floating Point Overflow or Underflow	0100
	4	Executive Return Instruction	0110
	5	RTC Overflow	1000
	6	Monitor Clock	1010
Class III, IOC Interrupts	1	Intercomputer Time-Out	110
	2	External Interrupt or Discrete Interrupt*	000
	3	Output Chain Interrupt	100
	4	Input Chain Interrupt	010

*Serial MIL-STD-188C, VACALES or EIA-STD-RS 232C Channels

†Cannot be locked out by status register 1

When the contents of the register change from one to zero, a MON clock interrupt (class II, priority 6) is generated, the count-down function is disabled, and control is transferred to the appropriate processing routine. The count-down function can be disabled by programming a *disable monitor clock* instruction (code 03 RR Format, m = 13).

A memory resume interrupt is generated when a memory module (8K) fails to acknowledge a request within 12 microseconds. If class I interrupts are not enabled when the event happens, the interrupt is lost.

"Instruction fault" interrupts are generated when the computer attempts to execute an instruction that is "not assigned" (not used). The "not assigned" group and those with octal code 70 through 77 when addressed by the P-register, will generate a CP instruction fault interrupt. Likewise, the "not assigned" group and those with octal code other than 70 through 77 when addressed by an I/O chain address pointer or CP command will generate an IOC instruction fault interrupt.

Peripheral devices may attempt to interrupt the computer by setting a coded message and an external interrupt request on the input cable. When enabled by the program the IOC stores the code in an assigned memory location (see Table 1) for that channel, responds with an "input acknowledge" and disables further external interrupts on that channel. If the program has also enabled the external interrupt monitor, a Class III, priority 2 interrupt suspends the CP program and transfers control to an appropriate subroutine for proper action. The program can disable or enable either or both of these functions. A disabled monitor prevents the interrupt generation but allows the storage of external interrupt data.

Interrupt Processing

When an interrupt is honored the CP hardware enters the following interrupt processing sequence:

a. Terminates the current program sequence and locks out all interrupts.

b. Stores the contents of P, SR#1, SR#2, and RTC register in assigned main memory as shown below.

Function	Address Assignment to Class		
	III	II	I
Stores the contents of P at address	110	120	130
Stores the contents of SR #1 at address	111	121	131
Stores the contents of SR #2 at address	112	122	132
Stores the contents of RTC lower at address	113	123	133
Stores the contents of RTC upper at address	117	127	137

c. Reloads the P, SR#1, and SR#2 from assigned memory locations as shown below. Interrupt lockouts and their release are controlled by program via the *load status register* instruction (code 03 RR, m = 1) or load PSW instruction (code 07).

		Address Assignment to Class		
	Function	III	II	I
c.	Reloads P with index[1] plus the contents of address	114	124	134
	Reloads SR #1[2] from address	115	125	135
	Reloads SR #2 from address	116	126	136

[1] See Figure 16 for index values.

[2] SR #1 bits 3-1 control interrupt lockout or release

d. Enables honoring interrupts not locked out by new contents of SR#1.

e. Executes the instruction at address in P and continues the program sequence from that point.

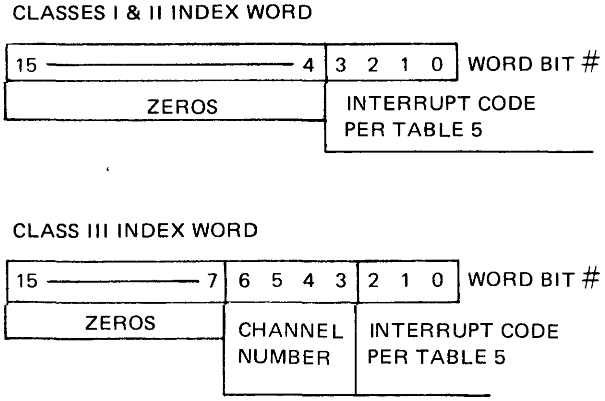


Figure 16. Interrupt Entrance Address Index

IOC Instruction Execution

Input/output instructions in the repertoire are divided into two types: 1) Chaining instructions are executed under control of an active channel chain; 2) Command instructions are executed under direction of the main program.

Parallel Input Interface Communication

When a device is ready to transmit data or an interrupt code, it places the information on the input data lines and raises the input data request line or the interrupt request line, respectively. The IOC, at its convenience, stores the word in memory and answers either request on the input acknowledge line.

Parallel Output Interface Communication

When an external device is ready to accept a command, it raises the external function request line to the IOC. At its convenience, the IOC places a command code on the output data lines and sets the external function acknowledge line. In another method, the IOC can "force" command words to external devices; in which case, the external function request line need not be set (refer to instruction code 70, RX Format, a = 3). The IOC places the command code on the data lines and sets the external function acknowledge. The external device reads the code and performs as commanded. When the device is ready to receive data, it raises the output data request line and the IOC responds at its convenience by placing a data word on the output lines and sets the output acknowledge line. External functions may be performed from either input or output chains.

Intercomputer Communication

Any parallel input/output channel may be used for communication with another computer having a compatible interface. The logic of the intercomputer output channel provides a time-out interrupt if the receiving computer does not accept an output word or is too slow at responding. This limit is determined by the time it takes to toggle bit 8 of the RTC register twice.

Peripheral Input Channel

A peripheral input channel is useful in communicating with another computer that does not have an available intercomputer channel. It allows a computer, equipped with a peripheral input channel, to perform like an ordinary output device of another computer. This channel is designed to capture input data "when presented" rather than "at its convenience", thus assuring the transmitting computer of successful transfers. The peripheral input channel has characteristics of the -15 volt NTDS (slow) interface and is the lowest numbered channel of a -15 volt NTDS (slow) 4-channel group.

MIL-STD-188C, VACALES and EIA-STD-RS232C Serial Channels

A MIL-STD-188C, VACALES or EIA-STD-RS232C serial channel communicates over a serial interface that transfers data and control information in both directions. Discrete lines are turned "ON" or "OFF" by command and chaining instructions to establish and hold a data link with external equipment. The external equipment turns discrete lines "ON" or "OFF" to interrupt the computer program, to furnish responses to computer controlled discretes and to inform the computer of "ability to transmit" (status). Asynchronous channels outline each character transmitted with START and STOP signals. Synchronous data transmissions, however, are timed by the external equipment via the transmit and receive clock lines. The MIL-STD-188C, VACALES and EIA-STD-RS232C interface lines and the direction of signal transmission are shown in Figure 8 and Figure 9 respectively.

Serial Channel Interrupts

The processor can be informed of the status of serial channels by a Class III external interrupt that is generated when any of the applicable events occur. Figure 17 defines the interrupt word format and the related bit interpretation for the MIL-STD-188C, VACALES and EIA-STD-RS232C serial channels. When the interrupt is generated the word is stored at the assigned external interrupt word for that channel.

BITS	MIL-STD-188	RS-232	VACALES
0 - 7	ALWAYS ONES	ALWAYS ONES	ALWAYS ONES
8	1 ⇒ B DISCRETE TURNED ON	1 ⇒ RING INDICATOR ON	1 ⇒ B DISCRETE TURNED ON
9	1 ⇒ C DISCRETE TURNED OFF	1 ⇒ RECEIVED LINE SIGNAL DETECTOR OFF	1 ⇒ CARRIER DETECT TURNED OFF
10	1 ⇒ I DISCRETE TURNED ON	ALWAYS ONE	1 ⇒ ALARM INDICATE TURNED ON
11	ALWAYS ONE	ALWAYS ONE	1 ⇒ SYNC ERROR TURNED ON
12	ALWAYS ONE	ALWAYS ONE	1 ⇒ TRANSMIT FULL ON TURNED OFF
13 - 15	ALWAYS ONES	ALWAYS ONES	ALWAYS ONES

Figure 17. Serial Channel Interrupt Word Format

NTDS Serial Channel Communication

NTDS serial channels transfer output data and external functions over a single coaxial line and transfer input data and external interrupt codes over another coaxial line. Words are identified by a single bit that follows the synchronizing bit. Transmissions are initiated by the transfer of appropriate control frames between the computer and the peripheral device. Figure 10 illustrates the direction of messages on each coaxial line and lists the codes used for control frames. An interchange of compatible control frames is required for each word that will be transmitted over the interface with the exception of a "forced" external function transfer. In this case the computer transmits the external function word even though the output request control frame does not specify the external function request.

Command Instruction

The computer is assigned a command cell in main memory from which it reads a command instruction when requested by the main program. The command cell consists of two addresses (see Table 1) that are used as follows:

1st location (address 140)	Storage for 1st word of command instruction
2nd location (address 141)	Storage for 2nd word of double-length command instruction (storage for y)

When the IOC reads and executes an instruction from the command cell, it clears the two most significant bits of the first command cell location to indicate that the instruction was executed as requested (see Table 6 for a list of Command instructions). The program has the option of checking the first command cell before loading the command cells with a new instruction. Channel activity is established by the instruction in the

TABLE 6. IOC INSTRUCTION LIST

Operation Code and Format	Instruction and Type		Execution Time Microseconds	
	Command	Chaining	Command	Chaining
70 RR	Channel Control	Channel Control	*30.0/2.0	2.25
70 RX	N/A	Initiate Transfer	–	4.5
71 RK	Initiate Chain	Load Control Memory	2.25	2.25
71 RX	Load Control Memory	Load Control Memory	3.0	3.0
72 RX	Store Control Memory	Store Control Memory	3.0	3.0
73 RR	N/A	Halt/Interrupt	–	1.5
73 RX	N/A	Set/Clear Flag	–	3.0
74 RK	N/A	Conditional Jump	–	2.25
75 RR	N/A	Search for Sync/Set	–	1.5
		Monitor/Set Suppress	–	1.5
76 RR	Set/Clear Discretes	Set/Clear Discretes	1.5	1.5
76 RX	Store Status	Store Status	3.0	3.0

*m = 0-7/m = 10-17

command cell therefore, the program can reload the command cell for an activity related to another channel with a different peripheral or set of peripherals.

Program Chaining

Instructions that control the input and output activity on all peripheral channels are executed from an active chain that is associated with each input and each output channel. Three I/O control memory locations are used for operations over each input channel, three for each output channel and three additional locations for each MIL-STD-188C VACALES and EIA-STD-RS232C serial I/O channel (see Figure 18 for the format and application of each word).

The transfer mode (TM) field specifies the type of I/O transfer as follows:

TM = 00: abort the transfer (input only)

TM = 01: transfer 8-bit bytes

TM = 10: transfer 16-bit words

TM = 11: transfer 32-bit (double) words

Bit 13: Not used

Byte pointer (B) is used, when performing 8-bit transfers, to specify the most or least significant byte in memory location for the next transfer. As each byte is transferred, the B-bit changes state.

B = 0 specifies the most significant 8-bits

B = 1 specifies the least significant 8-bits

Buffer transfer count specifies the number of bytes, single-length words, or double-length words to be transferred during the selected input data, output data, or external function buffer operation. As each byte or word is transferred, the buffer transfer count is decreased by one. When the count changes from 1 to 0, the buffer terminates. A beginning count of zero specifies the maximum number of transfers (4096).

Buffer address pointer specifies the memory address for the next transfer. The contents of this location are increased by one each time the B-bit changes from 1 to 0 for byte operations and each time a single-length word is transferred. For double-length word operations, the contents are increased by two for each transfer.

Whenever the computer executes the *initiate chain* instruction from the command cell, the chain address pointer (word 2) is loaded in the control memory for the specified channel and that chain is activated. A chain address pointer specifies an address in main memory where the next chaining instruction is located. As the pointer address is used, its value is advanced by one if a single-word instruction is read, and by two if a double-length instruction is read. Any time an executed instruction activates a buffer on a channel, the associated chain is deactivated until that buffer has terminated. Then the channel chain can proceed to the next instruction.

Monitor and suppress registers are used in serial mode (see instruction 75). The monitor register can be used to look for a specific character such as end of data in serial transmission while the suppress register can be used to suppress specific characters in serial transmission.

15 14	13	12	11 ——— 0	BIT #	
TM	0	B	BUFFER TRANSFER COUNT	WORD 0	INPUT
BUFFER ADDRESS POINTER				WORD 1	
CHAIN ADDRESS POINTER				WORD 2	
NOT USED				WORD 3	
TM	0	B	BUFFER TRANSFER COUNT	WORD 4	OUTPUT
BUFFER ADDRESS POINTER				WORD 5	
CHAIN ADDRESS POINTER				WORD 6	
NOT USED				WORD 7	
MONITOR REGISTER				WORD 10	SERIAL
SUPPRESS REGISTER				WORD 11	
SERIAL MODE INFORMATION				WORD 12	
NOT USED					

Figure 18. I/O Channel Control Memory

The serial mode information controls the mode, character size and parity for serial transmission.

Externally Specified Addressing (ESA)

The ESA feature provides peripheral devices with a means of specifying an absolute memory location for storage or retrieval of data. An active parallel dual-channel mode of operation is required for computer response to this function. If input is desired, the external device presents an input data request with the address and data. The address is presented on the lower order 16 data lines of the input pair and the data on the higher order lines. The computer stores all 32-bits from the channel – data at the specified address and the specified address at the next sequential memory location. If output is desired, an output data request is presented on the output channel with the address on the lower order input data lines. The computer loads the contents of the specified address on the higher order output data lines of the channel, the contents of the next sequential memory address on the lower order output data lines and raises the output acknowledge line.

External function and external interrupts on ESA Channels operates as normal dual channel.

Master Clear

A hardware initiated or operator initiated master clear extinguishes the FAULT indicators on the control and maintenance panels and places the computer in an initial condition which includes:

P – register cleared
Status register #1 cleared
Status register #2 cleared
RTC count-up and MON clock count-down functions disabled
Bits 0-5 of each page register set to its own register address and bit 15 cleared
All I/O channels cleared
- all data buffers deactivated
- all interrupt data store disabled
- Class III interrupts disabled
- RS 232 serial channel discretes cleared.
- 188C serial channel lines 5 & 6 set and other discrete lines cleared.

NORMAL DSPL set (maintenance panel).

Control and Maintenance Panels

A complete set of controls and indicators are provided for both operating and maintenance personnel. The front of the cabinet swing-out door contains the control panel (Figure 19) that includes those items necessary for turning on the computer, loading and running programs and for observing its operation. A more comprehensive maintenance panel (Figure 20) of controls and indicators is located on the inside of the door. This panel provides an effective aid to general servicing. Tables 7 and 8 list the respective operator's and maintenance panel controls and indicators and their functions.

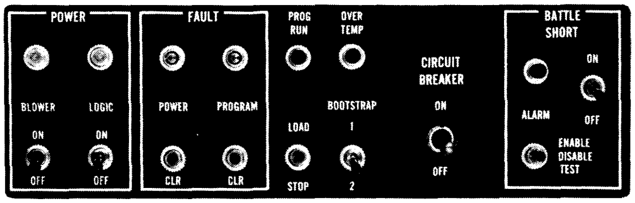


Figure 19. Control Panel

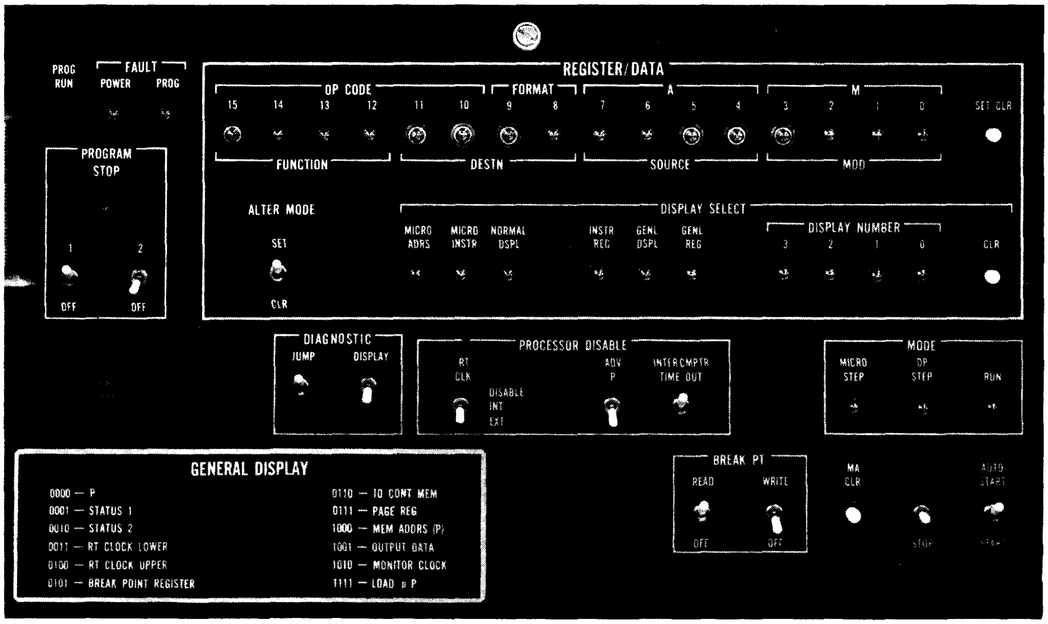


Figure 20. Maintenance Panel

TABLE 7. CONTROL PANEL SWITCHES AND INDICATORS

Indicator/Switch	Function
CIRCUIT BREAKER ON/OFF switch (two-position)	ON position enables primary power to the computer.
BLOWER POWER ON/OFF switch (two-position)	ON position applies power to the computer cooling fans and enables LOGIC POWER ON/OFF switch function.
BLOWER POWER indicator light	Lights when power is applied to blower.
LOGIC POWER ON/OFF switch (two-position)	ON position applies power to the computer logic.
LOGIC POWER indicator light	Lights when power is applied to logic.
POWER FAULT indicator light	Lights when power fault interrupt occurs.
POWER FAULT CLR switch (two-position return-to-neutral)	Momentary CLR position clears the POWER FAULT indicators on both the control panel and the maintenance panel.
PROGRAM FAULT indicator light	Lights when computer attempts to execute an illegal CP or I/O instruction.
PROGRAM FAULT CLR switch (two-position return-to-neutral)	Momentary CLR position, clears the PROGRAM FAULT indicator, on the control panel and the PROG FAULT indicator on the maintenance panel.
PROG RUN indicator light	Lights when computer is executing instructions in "run" mode.
OVER TEMP warning light	Lights when computer internal cabinet air temperature is within 25°F. of the maximum recommended operating temperature.
ALARM (audible)	Emits an audible sound when cabinet internal air temperature is in OVER TEMP range.

TABLE 7. CONTROL PANEL SWITCHES AND INDICATORS (CONT.)

Indicator/Switch	Function
ALARM ENABLE/ DISABLE/TEST switch (three-position)	ENABLE position allows the audible ALARM to sound. DISABLE position prevents the audible ALARM function. TEST position causes the audible ALARM to sound and the OVER TEMP indicator to light.
BATTLE SHORT ON/OFF switch (two-position)	ON position disables computer over-temperature shutdown function. OFF position enables computer over-temperature shutdown function.
BATTLE SHORT indicator light	Lights when BATTLE SHORT switch is ON
BOOTSTRAP 1-2 switch (two-position)	Position 1 enables execution of the first bootstrap program in NDRO memory. Position 2 enables execution of the second bootstrap program in NDRO memory.
LOAD/STOP switch (three-position return-to-neutral)	Momentary LOAD position causes the computer to execute a master clear, then begins executing the bootstrap program starting at P = 2. Momentary STOP position causes the computer to stop executing instructions if the computer is in the "run" mode.

TABLE 8. MAINTENANCE PANEL SWITCHES AND INDICATORS

Indicator/Switch	Function
PROG RUN indicator-switch	Indicator lights when the computer is executing instructions in the "run" mode. Depressing the switch selects the run condition in micro-step mode.
POWER FAULT indicator-switch	Indicator lights when power fault interrupt has occurred. Depressing the switch clears indicators on maintenance and control panel.
PROG FAULT indicator-switch	Indicator lights when the computer attempted to execute an illegal instruction. Depressing the switch clears indicators on maintenance and control panel.
PROGRAM STOP indicator light	Lights when a program "stop" condition has been satisfied (execution of a conditional jump with a = 11, 12 or 13).
PROGRAM STOP 1/OFF switch (two-position)	Switch in position 1 causes a program stop when the computer executes a conditional jump instruction with a = 12.
PROGRAM STOP 2/OFF switch (two-position)	Switch in position 2 causes a program stop when the computer executes a condition jump instruction with a = 13.
Time meter (0000 to 9999)	Indicates time, in hours, that power has been applied to computer logic.
DIAGNOSTIC JUMP switch (two-position)	JUMP position causes the microprogram to execute the micro-diagnostic from the master clear state or modify the execution of the code OO RR instruction.
DISPLAY switch (two-position)	In the DISPLAY position while the computer is in the Micro-step mode, a. With MICRO ADRS set, REGISTER/DATA displays the address of the next micro-instruction to be executed. b. With MICRO INSTR set, REGISTER/DATA displays the micro-instruction currently being executed. c. With NORMAL DSPL set, REGISTER/DATA displays the data on the source bus.

TABLE 8. MAINTENANCE PANEL SWITCHES AND INDICATORS (CONT.)

Indicator/Switch	Function
PROCESSOR DISABLES RT CLK DISABLE/INT/EXT switch (three-position)	DISABLE position inhibits counting up the real-time clock register and counting down the monitor clock register. INT position connects the internal 1 Khz clock to the real-time clock and monitor clock functions. EXT position connects the external clock source to the real-time clock and monitor clock functions.
AUTO START/START switch (three-position return-to-neutral)	The AUTO START position, after power is applied, or restored after a power fault interrupt, causes the computer to execute the instruction at NDRO address 000000. The momentary START position, starts normal computer operation in the selected mode.
STOP switch (two-position return-to-neutral)	Momentary operation to the STOP position, while the computer is executing instructions in the "run" mode, stops the computer.
MA CLR pushbutton switch	Depressing the switch when the computer is in the "run" condition, clears the FAULT indicators on the control and maintenance panels. Depressing the switch when the computer is not in the "run" condition clears the FAULT indicators.and places the computer in an initial state.
BREAK PT READ/OFF switch (two-position)	READ position stops the computer after reading data from the memory address entered in the breakpoint register.
BREAK PT WRITE/OFF switch (two-position)	WRITE position stops the computer after writing data in the memory address entered in the breakpoint register.
PROCESSOR DISABLES ADV P switch (two-position)	Up position disables advancing the P-register.
PROCESSOR DISABLES INTERCMPTR TIME OUT switch (two-position)	Up position inhibits the Class III, priority 1, intercomputer timeout interrupt.
MODE MICRO STEP indicator-switch	Indicator lights when computer is in "micro-step" mode. Depressing the switch places computer in "micro-step" mode.
MODE OR STEP indicator-switch	Indicator lights when computer is in "operation step" mode. Depressing the switch clears "run" mode and places computer in "op step" mode.

TABLE 8. MAINTENANCE PANEL SWITCHES AND INDICATORS (CONT.)

Indicator/Switch	Function
MODE RUN indicator-switch	Indicator lights when computer is in "run" mode. Depressing the switch clears "op step" mode and places computer in "run" mode.
DISPLAY SELECT CLR pushbutton switch	Depressing the switch clears DISPLAY NUMBER 0 through 3 and clears the "micro-step" mode.
ALTER MODE SET/CLEAR switch (two-position)	The SET position: a. enables each REGISTER/DATA indicator-switch to set when operated. b. causes all REGISTER/DATA indicator-switches to "clear" when the REGISTER/DATA SET/CLR switch is operated. In the CLEAR position: a. causes each REGISTER/DATA indicator-switch to clear when operated. b. causes all REGISTER/DATA indicator-switches to "set" when the REGISTER/DATA SET/CLR switch is operated.
REGISTER/DATA SET/CLR pushbutton switch	When operated, sets or clears (according to ALTER MODE SET/CLR position) all REGISTER/DATA indicator-switches.
REGISTER/DATA 0 through 15 indicator-switches	Indicators display the contents of a selected register. Depressing a switch modifies that particular bit of the selected register.
DISPLAY SELECT indicator-switches	Indicators identify the switch functions and the register being displayed by REGISTER/DATA indicators.
MICRO ADRS	Depressing the switch when computer is in "micro step" mode clears MICRO INSTR, NORMAL DSPL and selects micro-P register for display.
MICRO INSTR	Depressing the switch when computer is in the "micro step" mode clears MICRO ADRS, NORMAL DSPL and selects micro-I register for display.

TABLE 8. MAINTENANCE PANEL SWITCHES AND INDICATORS (CONT.)

<table>
<tr><th>Indicator/Switch</th><th>Function</th></tr>
<tr><td>NORMAL DSPL</td><td>Depressing the switch clears MICRO ADRS, MICRO INSTR and enables switch functions INSTR REG, GEN DSPL and GENL REG.</td></tr>
<tr><td>INSTR REG
indicator-switch</td><td>Depressing the switch, when enabled by NORMAL DSPL, clears GENL DSPL and GENL REG functions and causes REGISTER/DATA to display the contents of the instruction register.</td></tr>
<tr><td>GENL DSPL
indicator-switch</td><td>Depressing the switch, when enabled by NORMAL DSPL, clears the INSTR REG and GENL REG switch functions and causes REGISTER/DATA to display the contents of register selected by DISPLAY NUMBER.</td></tr>
<tr><td>GENL REG
indicator-switch</td><td>Depressing the switch, when enabled by NORMAL DSPL, clears the INSTR REG and GENL DSPL switch functions and causes REGISTER/DATA to display the contents of the general register selected by DISPLAY NUMBER in the set designated by bit 14 of Status Register #1.</td></tr>
<tr><td>DISPLAY NUMBER
3, 2, 1, 0
indicator-switches</td><td>Select and indicate register to be displayed by REGISTER/DATA as follows:
<table>
<tr><th>Bits
3,2,1,0</th><th>GENL DSPL
Selection</th><th>GENL REG
Selection</th></tr>
<tr><td>0 0 0 0</td><td>P-Register</td><td>General Register 0</td></tr>
<tr><td>0 0 0 1</td><td>SR #1</td><td>General Register 1</td></tr>
<tr><td>0 0 1 0</td><td>SR #2</td><td>General Register 2</td></tr>
<tr><td>0 0 1 1</td><td>RTC Lower</td><td>General Register 3</td></tr>
<tr><td>0 1 0 0</td><td>RTC Upper</td><td>General Register 4</td></tr>
<tr><td>0 1 0 1</td><td>Breakpoint</td><td>General Register 5</td></tr>
<tr><td>0 1 1 0</td><td>I/O Control Memory</td><td>General Register 6</td></tr>
<tr><td>0 1 1 1</td><td>Page Register</td><td>General Register 7</td></tr>
<tr><td>1 0 0 0</td><td>Memory Address</td><td>General Register 10</td></tr>
<tr><td>1 0 0 1</td><td>Output Data</td><td>General Register 11</td></tr>
<tr><td>1 0 1 0</td><td>Monitor Clock</td><td>General Register 12</td></tr>
<tr><td>1 0 1 1</td><td>not assigned</td><td>General Register 13</td></tr>
<tr><td>1 1 0 0</td><td>not assigned</td><td>General Register 14</td></tr>
<tr><td>1 1 0 1</td><td>not assigned</td><td>General Register 15</td></tr>
<tr><td>1 1 1 0</td><td>not assigned</td><td>General Register 16</td></tr>
<tr><td>1 1 1 1</td><td>Load Micro P-Register</td><td>General Register 17</td></tr>
</table>
</td></tr>
</table>

APPENDIX
REPERTOIRE OF INSTRUCTIONS

Instructions defined in this list include the basic instruction set and those required for optional features in the computer. Users of computer configurations that do not include certain optional instructions must place those respective instructions in the "Not assigned" category and assemble programs accordingly. "Not assigned" codes generate an instruction fault interrupt when executed.

The instructions are described in the following format:

(Operation Code)

(ULTRA symbol) (instruction format) (instruction name)

(Detailed descriptive text that includes special designator interpretations when applicable)

When the a- or m-designator is used as a sub-function code, the information is presented in table form.

When the instruction also sets the Condition Code as a result of its function the symbol "(CC)" appears after the description.

Symbols Used In Instructions

Symbol	Description
a	The a-designator from instruction words (a-values are expressed in octal).
xD	The two's complement deviation value in a local jump instruction.
R_a	The register designated by a.
m	The m-designator from instruction words (m-values are expressed in octal).
R_m	The register designated by m.
Y	The operand or operand address generated in the execution of an instruction.
y	The contents of the second word of an RK or RX instruction.
P	The Program Address register.
( )	The contents of the location specified within the parenthesis.
CC	The Condition Code

Operation Code 00

– RR Format – If the diagnostic jump switch is in the up position load the μP register with the contents of general register 17; otherwise a "not assigned" instruction.

– RI Format – Not assigned

– RK Format – Not assigned

BL RX Format – BYTE LOAD
Load the selected byte from address Y in bits 7 through 0 of R_a and clear bits 15 through 8. (CC)

Operation Code 01

LR RR Format – LOAD
Load (R_m) in R_a. (CC)

LI RI Format Type 2 – LOAD
Load the contents of memory address Y in R_a. (CC)

LK RK Format – LOAD
Load the Operand Y in R_a.(CC)

L RX Format – LOAD
Load the contents of memory address Y in R_a. (CC)

Operation Code 02

① RR Format – UNARY ARITHMETIC
Perform the operation specified for the m-value in Table I and then set the Condition Code according to the quantity resulting in R_a.

LDI RI Format, Type 2 – LOAD DOUBLE
Load the contents of addresses Y and Y+1 in R_a and R_{a+1} respectively. (CC)

– RK Format – Not assigned

LD RX Format – LOAD DOUBLE
This instruction shall load the contents of memory addresses Y and Y + 1 in R_a and R_{a+1} respectively. (CC)

Operation Code 03

② RR Format – UNARY-CONTROL
Perform the operation specified in Table II for the m-value. Set the condition code for m=0-3 and 15.

– RI Format – Not assigned

– RK Format – Not assigned

LM RX Format – LOAD MULTIPLE
Load the contents of sequential memory addresses beginning at Y, in sequential registers beginning at R_a and ending at R_m. If a is greater than m, load registers in the order R_a, R_{a+1}, . . . , R_{17}, R_0 . . . R_m. Address Y is equal to y; no indexing or indirect addressing is performed.

① See Table I
② Sec Table II

TABLE I. UNARY-ARITHMETIC INSTRUCTION m-VALUES

ULTRA Symbol	m Value	Operation	Description
PR	0	MAKE POSITIVE	If (R_a) are negative, perform the two's complement of (R_a) and store the result in R_a. When the maximum negative number* is complemented, set the overflow designator. If (R_a) are positive, do not change (R_a)
NR	1	MAKE NEGATIVE	If (R_a) are positive and not zero, perform the two's complement of (R_a) and store the result in R_a. If (R_a) are negative or zero, do not change (R_a)
RR	2	ROUND R_a	Add bit 15 of R_{a+1} to (R_a) and store the result in R_a. R_a must be even.
--	3	-------	Not assigned
TCR	4	TWO'S COMPLEMENT, SINGLE	Perform the two's complement of (R_a) and store the result in R_a. When the maximum negative number is complemented, set the overflow designator.
TCDR	5	TWO'S COMPLEMENT, DOUBLE	Perform the two's complement of double length (R_a, R_{a+1}) and store the result in R_a, R_{a+1}. When the maximum negative number is complemented, set the overflow designator.
OCR	6	ONE'S COMPLEMENT, SINGLE	Perform the one's complement of (R_a) and store the result in R_a.
--	7	-------	Not assigned
IROR	10	INCREASE R_a BY 1	Increase (R_a) by 1 and store the result in R_a.
DROR	11	DECREASE R_a BY 1	Decrease (R_a) by 1 and store the result in R_a.
IRTR	12	INCREASE R_a BY 2	Increase (R_a) by 2 and store the result in R_a.
DRTR	13	DECREASE R_a BY 2	Decrease (R_a) by 2 and store the result in R_a.
--	14-17	-------	Not assigned

*(1,000,000,000,000,000) binary

TABLE II. UNARY-CONTROL INSTRUCTION m-VALUES

ULTRA Symbol	m Value	Operation	Description
ER	0	EXECUTIVE RETURN	Generate Class II priority 4 Interrupt, set the Executive mode designator in the Status Register and store (P)+1 in R_a; No-op if Class II is not enabled. (CC)
SSOR	1	STORE STATUS REGISTER #1	Store the contents of Status Register #1 in R_a. (CC)
SSTR	2	STORE STATUS REGISTER #2	Store the contents of Status Register #2 in R_a. (CC)
SCR	3	STORE RTC LOWER	Store the contents of the Real Time Clock Register, lower order half, in R_a. (CC)
LPR	4	LOAD P	Load (R_a) in P.
LSOR	5	LOAD STATUS REGISTER #1	Load (R_a) in Status Register #1.
LSTR	6	LOAD STATUS REGISTER #2	Load (R_a) in Status Register #2.
LCR	7	LOAD RTC LOWER	Load (R_a) in the lower order half of the Real Time Clock Register.
ECR	10	ENABLE RTC	Enable the Real Time Clock Register to increase by one for each cycle of the RTC oscillator. Generate an RTC Interrupt when the lower half of the register overflows.
DCR	11	DISABLE RTC	Disable the Real Time Clock Register from advancing. The RTC oscillator continues to operate. Disable further RTC interrupts but process any current queued RTC interrupt.
LEM	12	LOAD AND ENABLE MONITOR CLOCK	Load (R_a) in the Monitor Clock Register and enable the register to decrease by one for each cycle of the RTC oscillator. Generate a monitor clock interrupt when the register contents equal zero.
			Disable the Monitor Clock Register from counting down.
			Load (R_a, R_a+1) in the Real Time Clock Register and enable it to advance by one for each cycle of clock.
			Store the contents of the Real Time Clock Register in R_a and R_a+1. (CC)
ECIR	16	ENABLE RTC INTERRUPT	Enable the generation of an RTC Interrupt when the lower half of the register overflows.
DCIR	17	DISABLE RTC INTERRUPT	Disable the generation of the RTC Interrupt.

Operation Code 04

① RR Format – UNARY-SHIFT
Perform the operation specified in Table III for the m-value.

– RI Format – Not assigned

– RK Format – Not assigned

BLX RX Format – BYTE LOAD AND INDEX BY 1
Generate memory address Y, increase (R_m) by 1, load the selected byte from memory address Y in bits 7 through 0 of R_a, clear bits 8 through 15. (CC)

Operation Code 05

SBR RR Format – SET BIT
Set the bit in R_a corresponding to the value of m. (CC)

LXI RI Format, Type 2 – LOAD AND INDEX BY 1
Generate memory address Y, increase (R_m) by 1 and then load the contents of memory address Y in Ra. (CC)

– RK Format – Not assigned

LX RX Format – LOAD AND INDEX BY 1
Generate memory address Y, increase (R_m) by 1 and then load the contents of memory address Y in R_a. (CC)

Operation Code 06

ZBR RR Format – ZERO BIT (Clear Bit)
Clear the bit in R_a corresponding to the value of m. (CC)

LDXI RI Format, Type 2 – LOAD DOUBLE AND INDEX BY 2
Generate memory address Y, increase (R_m) by 1, load the contents of memory addresses Y + 1 in R_a + 1. Increase (R_m) by 1, load the contents of memory address Y in R_a. (CC)

– RK Format – Not assigned

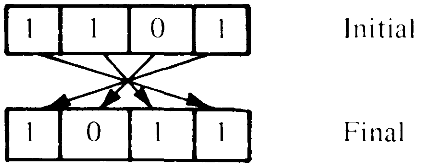
LDX RX Format – LOAD DOUBLE AND INDEX BY 2
Generate memory address Y, increase (R_m) by 1, load the contents of memory address Y + 1 in R_a + 1. Increase (R_m) by 1, load the contents of memory address Y into R_a. (CC)

Operation Code 07

CBR RR Format – COMPARE BIT
Compare the bit in R_a corresponding to the m-value with zero. (CC)

① See Table III

TABLE III. UNARY-SHIFT INSTRUCTION m-VALUE

ULTRA Symbol	M Value	Operation	Description
SQR	0	SQUARE ROOT*	Perform the square root of the double length (R_a, R_{a+1}) and store the result in R_{a+1} with the remainder in R_a. Set the condition code.
RVR	1	REVERSE REGISTER	Change (R_a) to the reverse order according to the 4-bit example and set the condition code. 
CNT	2	COUNT ONES	Count the number of one bits in (R_a), and store the count in R_{a+1}.
SFR	3	SCALE FACTOR	Shift the double length (R_a, R_{a+1}) to the left with zeros extended to fill, until bits 15 and 14 of R_a are not equal and store the shift count in R_{a+2}. (If the registers contain all zeros or all ones, the shift count is 31.)
---	4-17	-------	Not assigned.

*Optional Math Pac instruction. The square root of a number larger than 7777777777_8 or of a negative number sets the overflow designator bit 10, status register 1.

LPI RI Format, Type 2 – LOAD PSW
Load the contents of memory addresses Y, Y + 1 and Y + 2 in Program Address Register, Status Register #1 and Status Register #2, respectively. Y = (R_m).

– RK Format – Not assigned

LP RX Format – LOAD PSW
Load the contents of memory addresses Y, Y + 1 and Y + 2 in Program Address Register, Status Register #1 and Status Register #2, respectively.

Operation Code 10

LRSR RR Format – LOGICAL RIGHT SINGLE SHIFT
Shift (R_a) to the right n-places with zeros extended to fill. n is the value in bits 5-0 of R_m. (CC)

– RI Format – Not assigned

LRS RK Format – LOGICAL RIGHT SINGLE SHIFT
Shift (R_a) to the right n places with zeros extended to fill. n is the value in bits 5-0 of operand Y. (CC)

BS RX Format – BYTE STORE
Store bits 7-0 of (R_a) in the selected byte of memory address Y.

Operation Code 11

ARSR RR Format – ALGEBRAIC RIGHT SINGLE SHIFT
Shift (R_a) to the right n places with sign extended to fill. n is the value in bits 5-0 of R_m. (CC)

SI RI Format, Type 2 – STORE
Store (R_a) at memory address Y.

ARS Format RK – ALGEBRAIC RIGHT SINGLE SHIFT
Shift (R_a) to the right n places with sign extended to fill. n is the value in bits 5-0 of operand Y. (CC)

S RX Format – STORE
Store (R_a) at memory address Y.

Operation Code 12

LRDR RR Format – LOGICAL RIGHT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) to the right n-places with zeros extended to fill. n is the value in bits 5-0 of R_m. (CC)

SDI RI Format, Type 2 – STORE DOUBLE
Store (R_a) and (R_{a+1}) at memory addresses Y and Y + 1 respectively.

LRD RK Format – LOGICAL RIGHT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) to the right n places with zeros extended to fill. n is the value in bits 5-0 of operand Y. (CC)

SD RX Format – STORE DOUBLE
Store (R_a) and (R_{a+1}) at memory addresses Y and Y + 1 respectively.

Operation Code 13

ARDR RR Format – ALGEBRAIC RIGHT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) to the right n places with the sign extended to fill. n is the value in bits 5-0 of R_m.

– RI Format – Not assigned

ARD RK Format – ALGEBRAIC RIGHT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) to the right n places with the R_a sign extended to fill. n is the value in bits 5-0 of operand Y. (CC)

SM RX Format – STORE MULTIPLE
Store in sequential memory addresses beginning at Y, the contents of sequential registers beginning at R_a and ending at R_m. If a is greater than m store registers in the order R_a, R_{a+1}, . . . , R_{17}, R_o, . . . , R_m. Y equals y; no indexing or indirect addressing is performed.

Operation Code 14

ALSR RR Format – ALGEBRAIC LEFT SINGLE SHIFT
Shift (R_a) to the left n places with zeros extended to fill. n is the value in bits 5-0 of R_m. (CC)

– RI Format – Not assigned

ALS RK Format – ALGEBRAIC LEFT SINGLE SHIFT
Shift (R_a) to the left n places with zeros extended to fill. n is the value in bits 5-0 of operand Y. (CC)

BSX RX Format – BYTE STORE AND INDEX BY 1
Store bits 7-0 in R_a in the selected byte at memory address Y; and then increase (R_m) by 1.

Operation Code 15

CLSR RR Format – CIRCULAR LEFT SINGLE SHIFT
Shift (R_a) circularly to the left n places. n is the value in bits 5-0 of R_m. (CC)

SXI RI Format, Type 2 – STORE AND INDEX BY 1
Store (R_a) at memory address Y; and then increase (R_m) by 1. Y = (R_m).

CLS RK Format – CIRCULAR LEFT SINGLE SHIFT
Shift (R_a) circularly to the left n places. n is the value of bits 5-0 of operand Y. (CC)

SX RX Format – STORE AND INDEX BY 1
Store (R_a) at memory address Y; and then increase (R_m) by 1. Y = (R_m)

Operation Code 16

ALDR RR Format – ALGEBRAIC LEFT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) to the left n places with zeros extended to fill. n is the value in bits 5-0 of R_m. (CC)

SDXI RI Format, Type 2 – STORE DOUBLE AND INDEX BY 2
Generate memory address Y. Store (R_{a+1}) in memory address Y + 1, increase (R_m) by 1. Store (R_a) in memory address Y, increase (R_m) by 1.

ALD RK Format – ALGEBRAIC LEFT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) to the left n places with zeros extended to fill. n is the value in bits 5-0 of operand Y. (CC)

SDX RX Format – STORE DOUBLE AND INDEX BY 2
Generate memory address Y. Store (R_{a+1}) in memory address Y + 1, increase (R_m) by 1. Store (R_a) in memory address Y, increase (R_m) by 1.

Operation Code 17

CLDR RR Format – CIRCULAR LEFT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) circularly to the left n places with bit 15 of R_a transferred to bit o of R_{a+1} in each shift. n is the value in bits 5-0 of R_m. (CC)

SZI RI Format, Type 2 – STORE ZEROS
Store all zeros at memory address Y. Y = (R_m).

CLD RK Format – CIRCULAR LEFT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) circularly to the left n places with bit 15 of R_a transferred to bit 0 of R_{a+1} in each shift. n is the value in bits 5-0 of Y. (CC)

SZ RX Format – STORE ZEROS
Store all zeros at memory address Y.

Operation Code 20

SUR RR Format – SUBTRACT
Subtract (R_m) from (R_a) and store the result in R_a. (CC)

SUI RI Format, Type 2 – SUBTRACT
Subtract the contents of memory address Y from (R_a) and store the result in R_a. (CC) Y = (R_m)

SUK RK Format – SUBTRACT
Subtract operand Y from (R_a) and store the result in R_a. (CC)

SU RX Format – SUBTRACT
Subtract the contents of memory address Y from (R_a) and store the result in R_a. (CC)

Operation Code 21

SUDR RR Format – SUBTRACT DOUBLE
Subtract the double length (R_m, R_{m+1}) from the double length (R_a, R_{a+1}) and store the result in R_a and R_{a+1}. (CC)

SUDI RI Format, Type 2 – SUBTRACT DOUBLE
Subtract the double length contents of memory addresses Y, Y + 1 from the double length (R_a, R_{a+1}) and store the result in R_a and R_{a+1}. (CC)

– RK Format – Not assigned

SUD RX Format – SUBTRACT DOUBLE
Subtract the double length contents of memory addresses Y, Y + 1 from the double length (R_a, R_{a+1}) and store the result in R_a and R_{a+1}. (CC)

Operation Code 22

AR PR Format – ADD
ADD (R_m) to (R_a) and store the result in R_a. (CC)

AI RI Format, Type 2 – ADD
Add the contents of memory address Y to (R_a) and store the result in R_a. (CC)

AK RK Format – ADD
Add operand Y to (R_a) and store the result in R_a. (CC)

A RX Format – ADD
Add the contents of memory address Y to (R_a) and store the result in R_a. (CC)

Operation Code 23

ADR RR Format – ADD DOUBLE
Add the double length (R_m, R_{m+1}) to the double length (R_a, R_{a+1}) and store the result in R_a and R_{a+1}. (CC)

ADI RI Format, Type 2 – ADD DOUBLE
Add the double length contents of memory addresses Y, Y + 1 to the double length (R_a, R_{a+1}) and store the result in R_a and R_{a+1}. (CC)

– RK Format – Not assigned

AD RX Format – ADD DOUBLE
Add the double length contents of memory address Y, Y + 1 to the double length (R_a, R_{a+1}) and store the result in R_a and R_{a+1}. (CC)

Operation Code 24

CR RR Format – COMPARE
Arithmetically compare (R_a) to (R_m). (CC)

CI RI Format, Type 2 – COMPARE
Arithmetically compare (R_a) to the contents of memory address Y. (CC)

CK RK Format – COMPARE
Arithmetically compare (R_a) to operand Y. (CC)

C RX Format – COMPARE
Arithmetically compare (R_a) to the contents of memory address Y. (CC)

Operation Code 25

CDR PR Format – COMPARE DOUBLE
Arithmetically compare the double length (R_a, R_{a+1}) to the double length (R_m, R_{m+1}). (CC)

CDI RI Format, Type 2 – COMPARE DOUBLE
Arithmetically compare the double length (R_a, R_{a+1}) to the double length contents of memory addresses Y, Y + 1. (CC)

– RK Format – Not assigned

CD RX Format – COMPARE DOUBLE
Arithmetically compare the double length (R_a, R_{a+1}) to the double length contents of memory address Y, Y +1. (CC)

Operation Code 26

MR RR Format – MULTIPLY
Multiply (R_m) by (R_{a+1}) and store the double length result in R_a, R_{a+1}. (CC)

MI RI Format, Type 2 – MULTIPLY
Multiply the contents of memory address Y by (R_{a+1}) and store the double length result in R_a, R_{a+1}. (CC)

MK RK Format – MULTIPLY
Multiply operand Y by (R_{a+1}) and store the double length result in R_a, R_{a+1}. (CC)

M RX Format – MULTIPLY
Multiply the contents of memory address Y by (R_{a+1}) and store the double length result in R_a, R_{a+1}. (CC)

Operation Code 27

Note: For all divide operations, the remainder has the same sign as the dividend and the absolute value of the remainder is less than the absolute value of the divisor. (R_a, R_{a+1}) may not be the maximum negative number.

DR RR Format – DIVIDE
Divide the double length (R_a, R_{a+1}) by (R_m), store the quotient in R_{a+1} and the remainder in R_a. (CC)

DI RI Format, Type 2 – DIVIDE
Divide the double length (R_a, R_{a+1}) by the contents of memory address Y, store the quotient in R_{a+1} and the remainder in R_a. (CC)

DK RK Format – DIVIDE
Divide the double length (R_a, R_{a+1}) by operand Y, store the quotient in R_{a+1} and the remainder in R_a. (CC)

D RX Format – DIVIDE
Divide the double length (R_a, R_{a+1}) by the contents of memory address Y, store the quotient in R_{a+1} and the remainder in R_a. (CC)

Operation Code 30

ANDR RR Format – AND
Perform the logical AND of (R_a) and (R_m), and store the result in R_a. (CC)

ANDI RI Format, Type 2 – AND
Perform the logical AND of (R_a) and the contents of memory address Y and store the result in R_a. (CC)

ANDK RK Format – AND
Perform the logical AND of (R_a) and operand Y, and store the result in R_a. (CC)

AND RX Format – AND
Perform the logical AND of (R_a) and the contents of memory address Y, and store the result in R_a. (CC)

Operation Code 31

ORR RR Format – OR
Perform the logical OR of (R_a) and (R_m), and store the result in R_a. (CC)

ORI RI Format, Type 2 – OR
Perform the logical OR of (R_a) and the contents of memory address Y, and store the result in R_a. (CC)

ORK RK Format – OR
Perform the logical OR of (R_a) and operand Y, and store the result in R_a. (CC)

OR RX Format – OR
Perform the logical OR of (R_a) and the contents of memory address Y, and store the result in R_a. (CC)

Operation Code 32

XORR RR Format – EXCLUSIVE OR
Perform the exclusive OR of (R_a) and (R_m) and store the result in R_a. (CC)

XORI RI Format, Type 2 – EXCLUSIVE OR
Perform the exclusive OR of (R_a) and the contents of memory address Y, and store the result in R_a. (CC)

XORK RK Format – EXCLUSIVE OR
Perform the exclusive OR of (R_a) and operand Y, and store the result in R_a. (CC)

XOR RX Format – EXCLUSIVE OR
Perform the exclusive OR of (R_a) and the contents of memory address Y, and store the result in R_a. (CC)

Operation Code 33

MSR RR Format – MASKED SUBSTITUTE
For each bit set in (R_{a+1}) transfer the corresponding bit of (R_m) to the corresponding bit in R_a and leave the remaining bits in R_a unchanged. (CC)

MSI RI Format, Type 2 – MASKED SUBSTITUTE
For each bit set in (R_{a+1}) transfer the corresponding bit of the contents of memory address Y to the corresponding bit in R_a and leave the remaining bits in R_a unchanged. (CC)

MSK RK Format – MASKED SUBSTITUTE
For each bit set in (R_{a+1}) transfer the corresponding bit in operand Y to the corresponding bit in R_a and leave the remaining bits of R_a unchanged. (CC)

MS RX Format – MASKED SUBSTITUTE
For each bit set in (R_{a+1}) transfer the corresponding bit in the contents of memory address Y to the corresponding bit in R_a and leave the remaining bits of R_a unchanged. (CC)

Operation Code 34

CC=00 Note: This instruction with a positive mask will give results per Table 3; with a negative mask a resulting $CC=00_2$ indicates equality and a $CC\neq00_2$ indicates inquality.

CMR RR Format – COMPARE MASKED
Compare (bit by bit) the result of the logical AND of (R_a) and (R_{a+1}) to the result of the logical AND of (R_m) and (R_{a+1}). (CC)

CMI RI Format, Type 2 – COMPARE MASKED
Compare (bit by bit) the logical AND of (R_a) and (R_{a+1}) to the logical AND of contents of memory address Y and (R_{a+1}). (CC)

CMK RK Format – COMPARE MASKED
Compare (bit by bit) the logical AND of (R_a) and (R_{a+1}) to the logical AND of operand Y and (R_{a+1}). (CC)

CM RX Format – COMPARE MASKED
Compare (bit by bit) the logical AND of (R_a) and (R_{a+1}) to the logical AND of contents of memory address Y and (R_{a+1}). (CC)

Operation Code 35

IOCR RR Format – I/O COMMAND
Execute the I/O command instruction stored in main memory address 000140 and clear bits 15 and 14 of that address.

BFI RI Format, Type 2 – BIASED FETCH
Set the Condition Code on the contents of memory address Y and then set the two most significant bits at that memory location leaving the remaining bits unchanged.

REX RK Format – EXECUTIVE REMOTE
Execute the instruction as specified by the contents of memory address Y; do not change (P) when reading this instruction. Then continue with the next sequential instruction unless the remote instruction changes (P).

BF RX Format – BIASED FETCH
Set the Condition Code on the contents of memory address Y and then set the two most significant bits at that memory location leaving the remaining bits unchanged.

Operation Code 36 – Not assigned

Operation Code 37

RR Format -- CORDIC (optional feature)
Perform the arithmetic function specified by the m-designator on the initial contents of three general registers specified by the a-designator and leave the results in the same respective general registers. See Table IV for input parameters and output results.

a-designator specifies R_a, R_{a+1} and R_{a+2} ; m-designator specifies function as follows:

	m-value	Function
VF	0	Vector function trigonometric mode
RF	1	Rotate function trigonometric mode
VFP	2	Vector function trigonometric mode with prescale
RFP	3	Rotate function trigonometric mode with prescale
VH	4	Vector function hyperbolic mode
RH	5	Rotate function hyperbolic mode
VHP	6	Vector function hyperbolic mode with postscale
RHP	7	Rotate function hyperbolic mode with postscale
	10-17	Not assigned

TABLE IV. TRIGONOMETRIC AND HYPERBOLIC FUNCTIONS
(Operation Code 37)

37 RR m	Name of Function	INPUT PARAMETERS			OUTPUT RESULTS		
		R_a	R_{a+1}	R_{a+2}	R_a (Y)	R_{a+1} (X)	R_{a+2} (W)
0	Trigonometric vector without prescale	y	x	0	0	$X = \frac{R}{K} = \frac{\sqrt{x^2+y^2}}{K}$	$W = \theta = \tan^{-1}\frac{y}{x}$
1	Trigonometric rotate without prescale	y	x	θ	$Y = \frac{y \cos \theta + x \sin \theta}{K}$	$X = \frac{x \cos \theta - y \sin \theta}{K}$	0
2	Trigonometric vector with prescale	y	x	0	0	$X = R = \sqrt{x^2+y^2}$	$W = \theta = \tan^{-1}\frac{y}{x}$
3	Trigonometric rotate with prescale	y	x	θ	$Y = y \cos \theta + x \sin \theta$	$X = x \cos \theta - y \sin \theta$	0
4	Hyperbolic vector without postscale	y	x	0	0	$X = \frac{R}{K_1} = \frac{\sqrt{x^2-y^2}}{K_1}$	$W = v = \tanh^{-1}\frac{y}{x}$
5	Hyperbolic rotate without postscale	y	x	v	$Y = \frac{x \sinh v + y \cosh v}{K_1}$	$X = \frac{x \cosh v + y \sinh v}{K_1}$	0
6	Hyperbolic vector with postscale	y	x	0	0	$X = \sqrt{x^2-y^2}$	$W = v = \tanh^{-1}\frac{y}{x}$
7	Hyperbolic rotate with postscale	y	x	v	$Y = x \sinh v + y \cosh v$	$X = x \cosh v + y \sinh v$	0
1	Sin θ cos θ without prescale	0	0.46672_8	θ	$Y = \sin \theta$	$X = \cos \theta$	0
6	Log$_e$ x	x-1	x+1	0	0	2x	$W = \frac{1}{2}\log_e x$ $= \tanh^{-1}\frac{x+1}{x-1}$
7	Exponential	1	1	v positive	$Y = e^v = \sinh v + \cosh v$	$X = e^v = \cosh v + \sinh v$	0
1	Polar to Cartesian without prescale	0	R	θ	$Y = \frac{R \sin \theta}{K}$	$X = \frac{R \cos \theta}{K}$	0
3	Polar to Cartesian with prescale	0	R	θ	$Y = R \sin \theta$	$X = R \cos \theta$	0
1	Sin θ; cos θ	0	1	θ	$Y = \frac{\sin \theta}{K}$	$X = \frac{\cos \theta}{K}$	0

NOTES

x & y	Cartesian coordinates
θ	Angle of rotation Trigonometric mode
v	Angle of rotation Hyperbolic mode
K	0.46672_8
K_1	1.15217_8

Bit 15 of all input parameters indicates sign: 0 = positive, 1 = negative

Two's Complement notation is used for negative values

The radix point for Registers R_a and R_{a+1} must be the same

The radix point for W = Constant in hyperbolic mode is between bit 2^{15} and bit 2^{14}

The maximum value for positive trigonometric coordinates x and y is 33366_8 for m = 0, 1 and 55202_8 for m = 2, 3

The maximum value for positive hyperbolic coordinates x and y is 32700_8 for m = 5 and 26574_8 for m = 7

Angle θ is represented in Binary Angular Measurement (BAMS), Bit 2^{14} represents 90°. Each successive bit equal to one represents an angle one-half as large as its adjoining higher order bit. Least significant bit = .0054931° = 19.7"

$y/x \leqq .75$ for m = 4, 6 and $x \leqq 75646_8$ for m = 6

Operation Code 40

① RR Format – CONDITIONAL JUMP
Test for the condition specified in Table V for the a-value and perform one of the following:

(1) If the specified condition is met, jump to the instruction located at the address specified by (R_m). If the condition is not met execute the next instruction.

(2) If a specified Stop, or a Stop Key condition is met stop the computer. When the computer is started after a stop or the condition is not met, load (R_m) in P and execute the instruction at that address – (unconditional jump).

LJ RI Format, Type 1 – LOCAL JUMP
Jump to the instruction located at memory address Y. Y = (P) + xD.

① RK Format – CONDITIONAL JUMP
Test for the condition specified in Table V for the a value and perform one of the following:

(1) If the specified condition is met, jump to the instruction located at the address specified by operand Y. If the condition is not met execute the next instruction.

(2) If a specified Stop, or a Stop Key condition is met, stop the computer. When the computer is started after a stop or the condition is not met, load the operand Y in P and execute the instruction at that address (unconditional jump).

① RX Format – CONDITIONAL JUMP
Test for the condition specified in Table V for the a-value and perform one of the following:

(1) If the specified condition is met, jump to the instruction located at the address specified by the contents of memory address Y. If the condition is not met execute the next instruction.

(2) If a specified Stop, or a Stop Key condition is met, stop the computer. When the computer is started after a stop or the condition is not met, load (Y) in P and execute the instruction at that address (unconditional jump).

Operation Code 41

XJR RR Format – INDEX JUMP
Test (R_a) and perform one of the following:

(1) If (R_a) does not equal zero, decrease (R_a) by 1, jump to the instruction located at the address stored in R_m.

(2) If (R_a) equals zero, execute the next instruction.

LJI RI Format, Type 1 – LOCAL JUMP INDIRECT
Jump unconditionally to the address specified by the contents of memory address Y.

① See Table V

TABLE V. CONDITIONS FOR a-VALUE IN JUMP INSTRUCTIONS

ULTRA Symbol for Format			a-Value	Jump Condition	
RR	RK	RX		Condition code for Arithmetic Operation Indicates	Condition code for Compare Operation Indicates
JER	JE	JE	0	Zero	Equal
JNER	JNE	JNE	1	Not Zero	Not Equal
JGER	JGE	JGE	2	Positive	Greater Than or Equal
JLSR	JLS	JLS	3	Negative	Less Than
JOR	JO	JO	4	Overflow designator is set	
JCR	JC	JC	5	Carry Designator is set	
JPTR	JPT	JPT	6	Power is out of tolerance	
JBR	JB	JB	7	Bootstrap 2 is selected	
JR	J	J	10	Unconditional jump	
JSR	JS	JS	11	Unconditional Stop; jump on restart.	
JKSR	JKS	JKS	12	Stop if program stop key 1 is selected, then jump on restart; otherwise, unconditional jump	
JKSR	JKS	JKS	13	Stop if program stop key 2 is selected, then jump on restart; otherwise, unconditional jump.	
—	—	—	14-17	Unconditional jump	

XJ RK Format – INDEX JUMP
Test (R_a) and perform one of the following:

(1) If (R_a) does not equal zero, decrease (R_a) by 1 and jump to the instruction located at address Y.

(2) If (R_a) equals zero, execute the next instruction.

XJ RX Format – INDEX JUMP
Test (R_a) and perform one of the following:

(1) If (R_a) does not equal zero, decrease (R_a) by 1 and jump to the instruction located at the address specified by the contents of memory address Y.

(2) If (R_a) equals zero, execute the next instruction.

Operation Code 42

JLRR RR Format – JUMP AND LINK REGISTERS
Store (P)+1 in R_a, and jump to the instruction located at the address stored in R_m.

– RI Format – Not assigned

JLR RK Format – JUMP AND LINK REGISTER
Store (P)+2 in R_a, and jump to the instruction located at the address specified by operand Y.

JLR RX Format – JUMP AND LINK REGISTER
Store (P)+2 in R_a, and jump to the instruction located at the address specified by the contents of address Y.

Operation Code 43

– RR Format – Not assigned

LJLM RI Format, Type 1 – LOCAL JUMP AND LINK MEMORY
Store (P)+1 at memory address Y, and jump to the instruction located at memory address Y+1. Y = (P)+xD.

JLM RK Format – JUMP AND LINK MEMORY
Store (P)+2 at memory address Y, and jump to the instruction located at memory address Y+1.

JLM RX Format – JUMP AND LINK MEMORY
Store (P)+2 at the address specified by the contents of address Y, and jump to the instruction located at the address specified by (Y)+1.

Operation Code 44

JXR RR Format – JUMP REGISTER = 0
Test (R_a) and perform one of the following:

(1) If (R_a) equals zero, jump to the instruction located at the address stored in R_m.

(2) If (R_a) does not equal zero, execute the next instruction.

LJE RI Format, Type 1 – LOCAL JUMP EQUAL
Test the Condition Code in the Status Register and perform one of the following:

(1) If bit 8 of the Condition Code is "zero", jump to the instruction located at memory address Y. Y = (P)+xD.

(2) If bit 8 of the Condition Code is "one", execute the next instruction.

JZ RK Format – JUMP REGISTER = 0

Test (R_a) and perform one of the following:

(1) If (R_a) equals zero, jump to the instruction located at the address specified by operand Y.

(2) If (R_a) does not equal zero, execute the next instruction.

JZ RX Format – JUMP REGISTER = 0

Test (R_a) and perform one of the following:

(1) If (R_a) equals zero, jump to the instruction located at address specified by the contents of memory address Y.

(2) If (R_a) does not equal zero, execute the next instruction.

Operation Code 45

JNZR RR Format – JUMP REGISTER $\neq$ 0

Test (R_a) and perform one of the following:

(1) If (R_a) does not equal zero, jump to the instruction located at the address specified by R_m.

(2) If (R_a) equals zero, execute the next instruction.

LJNE RI Format, Type 1 – LOCAL JUMP NOT EQUAL

Test the Condition Code and perform one of the following:

(1) If bit 8 of the Condition Code is "one", jump to the instruction located at memory address Y. Y = (P)+xD.

(2) If bit 8 of the Condition Code is "zero", execute the next instruction.

JNZ RK Format – JUMP REGISTER $\neq$ 0

Test (R_a) and perform one of the following:

(1) If (R_a) does not equal zero, jump to the instruction located at the address specified by operand Y.

(2) If (R_a) equals zero, execute the next instruction.

JNZ RX Format – JUMP REGISTER $\neq$ 0

Test (R_a) and perform one of the following:

(1) If (R_a) does not equal zero, jump to the instruction located at the address specified by the contents of memory address y.

(2) If (R_a) equals zero, execute the next instruction.

Operation Code 46

JPR RR Format – JUMP REGISTER POSITIVE
Test (R_a) and perform one of the following:

(1) If (R_a) is equal to or greater than zero, jump to the instruction located at the address specified by R_m.

(2) If (R_a) is less than zero, execute the next instruction.

LJGE RI Format, Type 1 – LOCAL JUMP GREATER THAN OR EQUAL
Test the Condition Code and perform one of the following:

(1) If bit 9 of the Condition Code is "zero", jump to the instruction located at memory address Y. Y = (P)+xD.

(2) If bit 9 of the Condition Code is "one", execute the next instruction.

JP RK Format – JUMP REGISTER POSITIVE
Test (R_a) and perform one of the following:

(1) If (R_a) is equal to or greater than zero, jump to the instruction located at the address specified by operand y.

(2) If (R_a) is less than zero, execute the next instruction.

JP RX Format – JUMP REGISTER POSITIVE
Test (R_a) and perform one of the following:

(1) If (R_a) is equal to or greater than zero, jump to the instruction located at address specified by the contents of memory address Y.

(2) If (R_a) is less than zero, execute the next instruction.

Operation Code 47

JNR RR Format – JUMP REGISTER NEGATIVE
Test (R_a) and perform one of the following:

(1) If (R_a) is less than zero, jump to the instruction located at the address specified by R_m.

(2) If (R_a) is equal to or greater than zero, execute the next instruction.

LJLS RI Format, Type 1 – LOCAL JUMP LESS THAN
Test the Condition Code and perform one of the following:

(1) If bit 9 of the Condition Code is "one", jump to the instruction located at memory address Y. Y = (P)+xD.

(2) If bit 9 of the Condition Code is "zero", execute the next instruction.

JN RK Format – JUMP REGISTER NEGATIVE
Test (R_a) and perform one of the following:

(1) If (R_a) is less than zero, jump to the instruction located at the address specified by operand Y.

(2) If (R_a) is equal to or greater than zero, execute the next instruction.

JN RX Format – JUMP REGISTER NEGATIVE
Test (R_a) and perform one of the following:

(1) If (R_a) is less than zero, jump to the instruction located at the address specified by the contents of memory address Y.

(2) If (R_a) is equal to or greater than zero, execute the next instruction.

Operation Code 50

FSUR RR Format – FLOATING POINT SUBTRACT
Subtract the floating point number (R_m, R_{m+1}) from the floating point number (R_a, R_{a+1}); store the normalized floating point difference in R_a, R_{a+1} and then set the Condition Code. If residue is specified, store the residue in R_{a+2} and R_{a+3} in floating point format.

FSUI RI Format, Type 2 – FLOATING POINT SUBTRACT
Subtract the floating point number at memory addresses Y, Y+1 from the floating point number (R_a, R_{a+1}); store the normalized floating point difference in R_a, R_{a+1} and then set the Condition Code. If residue is specified, store the residue in R_{a+2} and R_{a+3} in floating point format.

– RK Format – Not assigned

FSU RX Format – FLOATING POINT SUBTRACT
Subtract the floating point number at memory addresses Y, Y+1 from the floating point number (R_a, R_{a+1}); store the normalized floating point difference in R_a, R_{a+1} and then set the Condition Code. If residue is specified, store the residue in R_{a+2} and R_{a+3} in floating point format.

Operation Code 51

FAR RR Format – FLOATING POINT ADD
Add the floating point number (R_m, R_{m+1}), to the floating point number (R_a, R_{a+1}); store the normalized floating point sum in R_a, R_{a+1} and then set the Condition Code. If residue is specified, store the residue in R_{a+2} and R_{a+3} in floating point format.

FAI RI Format, Type 2 – FLOATING POINT ADD
Add the floating point number at memory addresses Y, Y+1 to the floating point number (R_a, R_{a+1}); store the normalized floating point sum in R_a, R_{a+1} and then set the Condition Code. If rounding is not specified, store the residue in R_{a+2} and R_{a+3} in floating point format.

– RK Format – Not assigned

FA RX Format – FLOATING POINT ADD
Add the floating point number at memory addresses Y, Y+1 to the floating point number (R_a, R_{a+1}); store the normalized floating point sum in R_a, R_{a+1} and then set the Condition Code. If residue is specified, store the residue in R_{a+2} and R_{a+3} in floating point format.

Operation Code 52

FMR RR Format – FLOATING POINT MULTIPLY
Multiply the floating point number (R_m, R_{m+1}) by the floating point number (R_a, R_{a+1}); store the normalized floating point product in R_a, R_{a+1} and then set the Condition Code. (R_a, R_{a+1}) is a floating point number representing the most significant digits of the product. If residue is specified, R_{a+2} and R_{a+3} contain a floating point number representing the least significant portion of the product.

FMI RI Format, Type 2 – FLOATING POINT MULTIPLY
Multiply the floating point number at memory addresses Y, Y+1 by the floating point number (R_a, R_{a+1}); store the normalized floating point product in R_a, R_{a+1} and then set the Condition Code. (R_a, R_{a+1}) is a floating point number representing the most significant digits of the product. If residue is specified, R_{a+2} and R_{a+3} contain a floating point number representing the least significant portion of the product.

– RK Format – Not assigned

FM RX Format – FLOATING POINT MULTIPLY
Multiply the floating point number at memory addresses Y, Y+1 by the floating point number (R_a, R_{a+1}); store the normalized floating point product in R_a, R_{a+1} and then set the Condition Code. (R_a, R_{a+1}) is a floating point number representing the most significant digits of the product. If residue is specified, R_{a+2} and R_{a+3} contain a floating point number representing the least significant portion of the product.

Operation Code 53

FDR RR Format – FLOATING POINT DIVIDE
Divide the floating point number (R_a, R_{a+1}) by the floating point number (R_m, R_{m+1}); store the normalized floating point quotient in R_a, R_{a+1} and then set the Condition Code. If residue is specified, R_{a+2} and R_{a+3} contain the remainder in floating point format.

FDI RI Format, Type 2 – FLOATING POINT DIVIDE
Divide the floating point number (R_a, R_{a+1}) by the floating point number at memory addresses Y, Y+1; store the normalized floating point quotient in R_a, R_{a+1} and then set the Condition Code. If residue is specified, R_{a+2} and R_{a+3} contain the remainder in floating point format.

– RK Format – Not assigned

FD RX Format – FLOATING POINT DIVIDE
Divide the floating point number (R_a, R_{a+1}) by the floating point number at memory addresses Y, Y+1; store the normalized floating point quotient in R_a, R_{a+1} and then set the Condition Code. If residue is specified, R_{a+2} and R_{a+3} contain the remainder in floating point format.

Operation Code 54

LARR RR Format – LOAD PAGE ADDRESS REGISTER
Load (R_m) in the page register specified by (R_a) 5-0.

LARI RI Format, Type 2 – LOAD PAGE ADDRESS REGISTER
Load page register as specified by (R_a) with the contents of the memory address specified by (R_m).

– RK Format – Not assigned

LARM RX Format – LOAD PAGE ADDRESS REGISTER MULTIPLE
Load the contents of sequential memory addresses beginning at Y, in sequential page registers beginning at the address defined by (R_a) 5-0 until the number of executions equals one plus the count in (R_a) 13-8. A count of "zero" loads one page register.

Operation Code 55

SARR RR Format – STORE PAGE ADDRESS REGISTER
Store the page register specified by (R_a) 5-0 in R_m.

SARI RI Format, Type 2 – STORE PAGE ADDRESS REGISTER
Store the page register specified by (R_a) 5-0 memory address specified by (R_m).

– RK Format – Not assigned

SARM RX Format – STORE PAGE ADDRESS REGISTER MULTIPLE
Store the contents of sequential page registers beginning at the register number defined by (R_a) 5-0 in sequential memory addresses beginning at Y until the number of executions equals one plus the count in (R_a) 13-8. A count of "zero" stores one page register.

Operation Code 56

MDR RR Format – MULTIPLY DOUBLE
Multiply the double length (R_m, R_{m+1}) by the double length (R_a, R_{a+1}), store the result in R_a*, R_{a+1}, R_{a+2}, R_{a+3}. (CC)

MDI RI Format, Type 2 – MULTIPLY DOUBLE
Multiply the double length contents of memory addresses Y, Y+1 by the double length (R_a, R_{a+1}), store the results in R_a*, R_{a+1}, R_{a+2}, R_{a+3}. (CC)

– RK Format – Not assigned

*Bit 15 of (R_a) is the sign bit for the 64-bit operand.

MD RX Format – MULTIPLY DOUBLE
Multiply the double length contents of memory addresses Y, Y+1 by the double length (R_a, R_{a+1}), store the results in R_a*, R_{a+1}, R_{a+2}, R_{a+3}. (CC)

Operation Code 57

See Note under Operation Code 27

DDR RR Format – DIVIDE DOUBLE
Divide the number (R_a*, R_{a+1}, R_{a+2}, R_{a+3}) by the double length number (R_m, R_{m+1}), store the double length quotient in R_{a+2}, R_{a+3} and the double length remainder in R_a, R_{a+1}. (CC)

DDI RI Format, Type 2 – DIVIDE DOUBLE
Divide the number (R_a*, R_{a+1}, R_{a+2}, R_{a+3}) by the double length contents of memory address Y, Y+1, store the double length quotient in R_{a+2}, R_{a+3} and the double length remainder in R_a, R_{a+1}. (CC)

– RK Format – Not assigned

DD RX Format – DIVIDE DOUBLE
Divide the number (R_a*, R_{a+1}, R_{a+2}, R_{a+3}) by the double length contents of memory address Y, Y+1, store the double length quotient in R_{a+2}, R_{a+3} and the double length remainder in R_a, R_{a+1}. (CC)

Operation Code 60

RR Format – Not assigned
RI Format – Not assigned
RK Format – Not assigned
RX Format – Not assigned

LLRS RL-1 Format – LOGICAL RIGHT SINGLE SHIFT
Shift (R_a) right n places with zeros extended to fill, and set the Condition Code; n is the value of the m-designator.

LARS RL-2 Format – ALGEBRAIC RIGHT SINGLE SHIFT
Shift (R_a) right n places with the sign extended to fill, and set the Condition Code; n is the value of the m-designator.

LLRD RL-3 Format – LOGICAL RIGHT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) right n places with zeros extended to fill, and set the Condition Code; n is the value of the m-designator.

LARD RL-4 Format – ALGEBRAIC RIGHT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) right n places with sign extended to fill, and set the Condition Code; n is the value of the m-designator.

*Bit 15 of (R_a) is the sign bit for the 64-bit operand. The dividend cannot be the maximum negative number.

Operation Code 61

RR Format – Not assigned
RI Format – Not assigned
RK Format – Not assigned
RX Format – Not assigned

LALS RL-1 Format – ALGEBRAIC LEFT SINGLE SHIFT
Shift (R_a) left n places with zeros extended to fill, and set the Condition Code; n is the value of the m-designator.

LCLS RL-2 Format – CIRCULAR LEFT SINGLE SHIFT
Shift (R_a) left circular n places, and set the Condition Code; n is the value of the m-designator.

LALD RL-3 Format – ALGEBRAIC LEFT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) left n places with zeros extended to fill, and set the Condition Code; n is the value of the m-designator.

LCLD RL-4 Format – CIRCULAR LEFT DOUBLE SHIFT
Shift the double length (R_a, R_{a+1}) left circular n places, and set the Condition Code; n is the value of the m-designator.

Operation Code 62

RR Format – Not assigned

RI Format – Not assigned

RK Format – Not assigned

RX Format – Not assigned

LSU RL-1 Format – SUBTRACT
Subtract the 4-bit m-value from (R_a), store the result in R_a. (CC)

LSUD RL-2 Format – SUBTRACT DOUBLE
Subtract the 4-bit m-value from the double length (R_a, R_{a+1}), store the result in R_a, R_{a+1}. (CC)

LA RL-3 Format – ADD
Add the 4-bit m-value to (R_a), store the result in R_a. (CC)

LAD RL-4 Format – ADD DOUBLE
Add the 4-bit m-value to the double length (R_a, R_{a+1}), store the result in R_a, R_{a+1}. (CC)

Operation Code 63

RR Format – Not assigned

RI Format – Not assigned

RK Format – Not assigned

RX Format – Not assigned

LL RL-1 Format – LOAD
Load the 4-bit m-value into R_a. (CC)

LC RL-2 Format – COMPARE
Arithmetically compare the 4-bit m-value with (R_a). (CC)

LMUL RL-3 Format – MULTIPLY
Multiply the 4-bit m-value by (R_{a+1}), store the double length result in R_a, R_{a+1}. (CC)

LDIV RL-4 Format – DIVIDE
Divide the double length (R_a, R_{a+1}) by the 4-bit m-value, store the quotient in R_{a+1}, the remainder in R_a. (CC)

Operation Code 64

– RR Format – Not assigned

– RI Format – Not assigned

– RK Format – Not assigned

BSU RX Format – BYTE SUBTRACT
Subtract the selected byte of the contents of memory address Y from (R_a), store the result in R_a. (CC)

Operation Code 65

– RR Format – Not assigned

– RI Format – Not assigned

– RK Format – Not assigned

BA RX Format – BYTE ADD
Add the selected byte from the contents of memory address Y to (R_a), store the sum in R_a. (CC)

Operation Code 66

– RR Format – Not assigned

– RI Format – Not assigned

– RK Format – Not assigned

BC RX Format – BYTE COMPARE
Arithmetically compare (R_a) to the selected byte of contents of memory address Y and set the Condition Code.

Operation Code 67

– RR Format – Reserved for "user" defined micro program instructions; otherwise not assigned.
– RI Format – (Same as RR)

– RK Format – (Same as RR)

BCX RX Format – BYTE COMPARE AND INDEX BY 1
Arithmetically compare (R_a) to the selected byte of contents of memory address Y, set the Condition Code and increase (R_m) by 1.

Operation Code 70

① RR Format – CHANNEL CONTROL (Command/Chaining)
Perform the operation specified for the m-value in Table VI

TABLE VI. CHANNEL CONTROL INSTRUCTION m-DESIGNATOR

ULTRA Symbol	m-Value	Operations effecting all channels collectively (command or chaining)
ACR	0	*Master Clear – deactivate all chains and data buffers – disable all external interrupt data storage and associated monitors, clear EIE lines – clear monitor and suppress flags.
	1	Not assigned
	2	Not assigned
	3	Not assigned
	4	*Enable all external interrupt data store ; set EIE lines.
	5	*Disable all external interrupt data store ; clear EIE lines.
	6	*Enable all external interrupt monitors to allow Class III, priority 2, 3 and 4 interrupt generation. If external interrupt data were stored while monitors were disabled, generate the Class III, priority 2 interrupt.
	7	*Disable all priority 2, 3 and 4 interrupt generation.
		Operations effecting only the channel specified by the a-designator (command) or the associated channel (chaining)
CCR	10	Master Clear the channel (See m = 0 above)
	11	Not assigned
	12	Not assigned
	13	Not assigned
	14	Enable the channel external interrupt data store ; set EIE lines
	15	Disable the channel external interrupt data store ; clear EIE lines
	16	Enable the channel Class III priority 2, 3 and 4 interrupt generation (See m = 6 above)
	17	Disable the channel Class III priority 2, 3 and 4 interrupt generation

*The a-designator must be zero.

RI Format – Not assigned

RK Format – Not assigned

① See Table VI.

IO RX Format – INITIATE TRANSFER (Chaining)
Load the contents of memory addresses Y and Y+1 in control memory BCW and BAP locations, respectively, and enable input/output transfers on the channel corresponding to the chain executing the instruction. Disable the chain until the buffer terminates and then enable the chain corresponding to buffer terminated (the m-designator is not used and Y must be an even number).

TABLE VII. INITIATE TRANSFER INSTRUCTION a-VALUE

a-Value	Transfer Mode
XX00	Input data
XX01	Output data
XX10	External function
XX11	External function with force
X can be 0 or 1	

Operation Code 71

RR Format - Not assigned

RI Format – Not assigned

RK Format – INITIATE CHAIN (Command)
Transfer Y to the control memory input or output Chain Address Pointer as specified by the m-designator for the channel specified by the a-designator and enable the chain for that channel.

ICK m = 2 ⇒ Input Chain
OCK m = 6 ⇒ Output Chain
Other m-values load operand Y in the control memory location specified by the m designator.

LCMK RK Format – LOAD CONTROL MEMORY (Chaining)
Load operand Y in the control memory location specified in Table VIII for the m-designator (a-designator values are not interpreted.)

WCM RX Format – LOAD CONTROL MEMORY (Command)
Load the contents of memory address Y in the control memory location specified in Table VIII for the combined am-designator.

LCM RX Format – LOAD CONTROL MEMORY (Chaining)
Load the contents of memory address Y in the control memory location specified in Table VIII for the m-designator (m-values 3, 7 and 13-17 and all a-designator values are not interpreted.)

TABLE VIII. CONTROL MEMORY ADDRESS SELECTION

a- Value	m- Value	Register selected	
	0	TM, O, B and Buffer word count	Input
	1	Buffer Address Pointer	
	2	Chain Address Pointer	
	3	Not used	
	4	TM, O, B and buffer word count	Output
	5	Buffer Address Pointer	
	6	Chain Address Pointer	
	7	Not used	
	10	Monitor register	Serial*
	11	Suppress register	
	12**	Serial mode information register	
	13-17	Not used	
0-17	Channel designator for command instructions, not used for chaining instructions		

*MIL-STD-188, VACALES and EIA-STD-RS232 serial channels.
**See Figure I for interpretation of serial mode information.

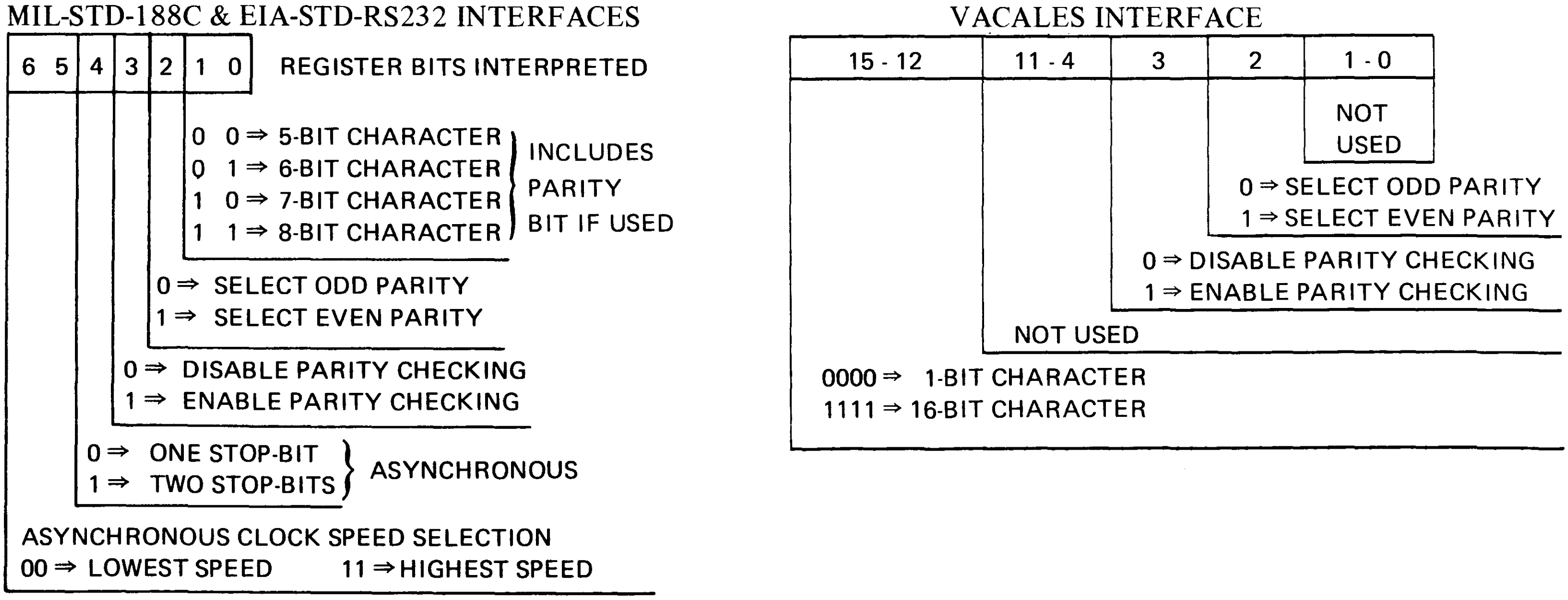


Figure I. Serial Mode Information Interpretation

Operation Code 72

– RR Format – not assigned

– RI Format – Not assigned

– RK Format – Not assigned

RCM RX Format – STORE CONTROL MEMORY (Command)
Store in memory address Y the contents of control memory location specified in Table VIII for the combined am-designator.

SCM RX Format – STORE CONTROL MEMORY (Chaining)
Store in memory address Y the contents of control memory location specified in Table VIII for the m-designator (a-designator values are not interpreted).

Operation Code 73

RR Format – HALT/INTERRUPT
Perform the operation specified as follows:

HCR If a = 0, halt the chaining action
IPR If a = 1, generate the chain interrupt to the central processor.

The m-designator and a-values $2\text{-}17_8$ are not interpreted.

RI Format – Not assigned

RK Format – Not assigned

RX Format – SET/CLEAR FLAG (Chaining)
Set or clear the most significant two bits (flag) in memory location specified by Y as follows:

SF If a = 1, set the flag
ZF If a = 0, clear the flag

The m-designator is not used.

Operation Code 74

RR Format – Not assigned

RI Format – Not assigned

SJC RK Format – CONDITIONAL JUMP (Chaining)
If the condition specified for the a-value in Table IX is satisfied, load the chain address pointer location with Y; otherwise execute the next sequential instruction in the chain. Clear the designator tested in either case.

RX Format – Not assigned

Operation Code 75

SFSC RR Format – SEARCH FOR SYNC/SET MONITOR/SET SUPPRESS (Chaining)
Condition the IOC for the next activated input buffer to perform the operation specified by bits 3, 2, 1 and 0 of the m-designator as follows:

2^0-bit set (Set synchronous serial channel active)
Set the synchronous serial channel active and enable the chain for the channel. If 2^0-bit is clear, disable synchronization.

2^1-bit set (Set Suppress on synchronous and asynchronous channel)
Compare each character of the input stream to the character in the suppress register and store it in memory until equality is detected, then suppress that character.

2^2-bit set (Set Monitor on synchronous or asynchronous channel)
Compare each character of the input stream to the character in the monitor register and store it in memory. When equality is detected set the monitor designator, store the character and enable the channel chain (terminate the buffer). If 2^2-bit is cleared, disable set monitor.

2^3-bit and 2^0-bit set (Search for Sync on synchronous channels)
At each bit-time of the incoming data stream, compare the value of a character length word to the character in the suppress register. When equality is detected, compare the next character to the suppress register and enable the channel chain; if equality is detected again set the suppress designator. If 2^3-bit is clear, disable the search for sync function.

2^3-bit set 2^2-bit clear (Search for Sync Bit by Bit on VACALES Channels)
At each bit time of incoming data stream, compare the value of a character length to the contents of the suppress register. When a match occurs start assembling the next input data and enable the next chain instruction. If 2^3-bit is cleared, disable search for sync.

2^3-bit and 2^2-bit set (Search for Sync Character on VACALES Channels)
Compare each character of the input stream to the contents of the suppress register. When a compare results in a match, set the suppress flag and enable the next chain instruction. If compare does not result in a match, the suppress flag is not set and the next chain instruction is enabled.

RK Format – Not assigned
RI Format – Not assigned
RX Format – Not assigned

Operation Code 76

RR Format – SET/CLEAR DISCRETE (Command and chaining)
Set or clear the discrete associated with the MIL-STD-188C, VACALES or EIA-STD-RS232 serial interface according to the m-value in Table X.

SICR Command instruction – "a" specifies the channel
CSIR Chaining instruction – "a" is not used.

RI Format – Not assigned
RK Format – Not assigned

RX Format – STORE STATUS (Command and chaining)
Store the channel status word at memory address Y (bit-interpretation is shown in Table XI.

SST Command instruction – "a" specifies the channel.
CSST Chaining instruction – "a" is not used.

Operation Code 77 — Not assigned.

TABLE IX. a-DESIGNATOR JUMP CONDITIONS

a-Value	Jump Condition
0	Unconditional jump
1	Jump if "suppress" designator is not set
2	Jump if "monitor" designator is set
3-17	Not used

TABLE X. DISCRETE SET/CLEAR FUNCTIONS

		MIL-STD-188C/VACALES			EIA-STD-RS232	
Octal m-Value	Function	Discrete	Line Designator MIL-STD-188C	Line Designator VACALES	Discrete	Line Designator
0	Set	Loop test* (internal)	–	–	Loop test* (internal)	–
1	Clear	Loop test* (internal)	–	–	Loop test* (internal)	–
2	Clear	Not used	Not used	Not used	Spare	–
3	Set	Not used	Not used	Not used	Spare	–
4	Clear	Outbound Control Line 6	J (off)	J1	Spare	–
5	Set	Outbound Control Line 6	J (on)	J1	Spare	–
6	Clear	Outbound Control Line 5	H (off)	Transmitter Prep.	Enable Ring Indicator*	CE
7	Set	Outbound Control Line 5	H (on)	Transmitter Prep.	Enable Ring Indicator*	CE
10	Clear	Outbound Control Line 4	G (off)	G1	Request to Send	CA
11	Set	Outbound Control Line 4	G (on)	G1	Request to Send	CA
12	Clear	Outbound Control Line 3	F (off)	F1	New Sync	CH
13	Set	Outbound Control Line 3	F (on)	F1	New Sync	CH
14	Clear	Outbound Control Line 2	D (off)	D1	Data Terminal Ready	CD
15	Set	Outbound Control Line 2	D (on)	D1	Data Terminal Ready	CD
16	Clear	Outbound Control Line 1	A (off)	Loop Back (Modem)	Loop Test (external)	–
17	Set	Outbound Control Line 1	A (on)	Loop Back (Modem)	Loop Test (external).	–

*Internal function – no interface line affected.

TABLE XI. STATUS WORD INTERPRETATION

Word Bit #	MIL-STD-188 Function	EIA-STD-RS232 Function	MIL-STD-188 and EIA-STD-RS232 Description
2^0	Parity Error	Parity Error	Serial channel detects a parity error on an input word.
2^1	Overrun	Overrun	Serial channel does not store an input word before another is transmitted.
2^2	Break	Break	Serial channel does not detect a STOP-bit. (Used in asynchronous mode only)
2^3	E Active	Clear to Send	Line is set "active" by an external device.

Bit	VACALES Function	VACALES Description
2^1	Overrun	The serial I/O did not transfer to memory before another I/O word was received.
2^2	Parity Error	The serial I/O detected a parity error on an input data word.
2^3	Sync Error	The inbound discrete control line, Sync Error, was set by an external device.

RESTRICTIVE NOTICE

This manual is intended to inform the reader regarding the general construction, operational characteristics, and capabilities of the AN/UYK-20 computer. It should not, however, be considered as an equipment specification, and Sperry Univac in no way warrants the accuracy or completeness of the manual for procurement purposes. Products and services described herein are available for sale only to the federal government of the United States of America or its designees.

SPERRY UNIVAC
DEFENSE SYSTEMS