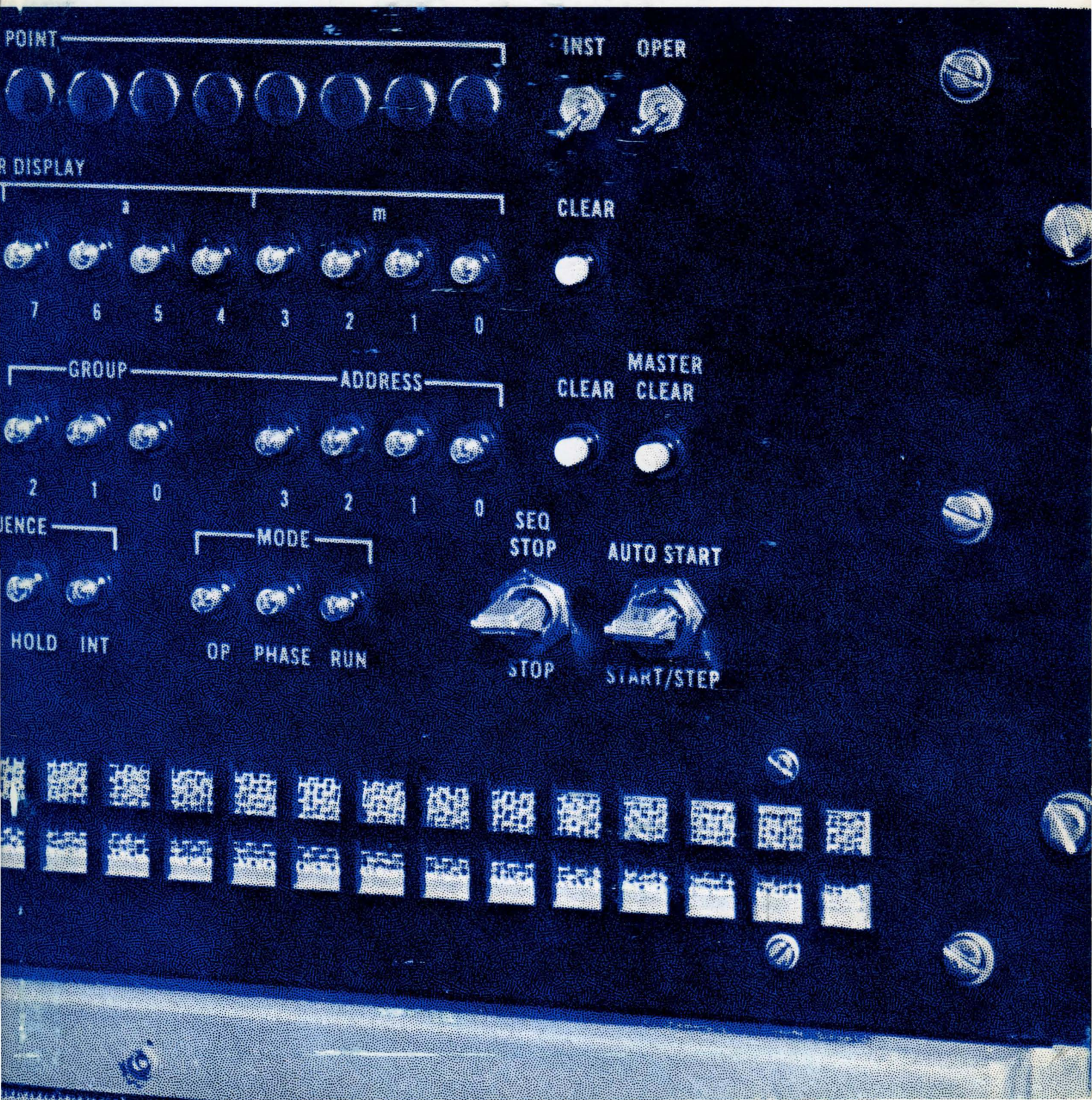
SPERRY UNIVAC

technical description AN/UYK-15



AN/UYK-15
technical description

SPERRY UNIVAC
DEFENSE SYSTEMS

TABLE OF CONTENTS

TABLE OF CONTENTS (CONT.)

The AN/UYK-15

- ***SYSTEM ORIENTED***
- ***RUGGEDIZED***
- ***REAL TIME***
- ***UNIVAC RELIABILITY***
- ***COST EFFECTIVE***

THE AN/UYK-15(V) COMPUTER

INTRODUCTION

The AN/UYK-15 is a general purpose, militarized UNIVAC® computer with a wealth of computing power in a small, ruggedized package. It is designed to meet the requirements of small, medium or large applications in mobile shelters, shipboard or other military environments. Univac offers a choice of configurations that encompass a variety of capabilities. A small version can grow – with optional functions that increase processing efficiency and programming versatility; with hardware expansions that increase processing capacity and speed; and with associated support software to keep pace with growing applications.

The AN/UYK-15 is adaptable to special applications. With its modular design many special needs can be met easily – special operations, simply by adding printed circuit cards and associated programming routines; expansions, by adding more input/output channels or more memory modules.

The basic computer, which has 8,192 words of memory and a single processor-controlled input/output channel, can process all problems that are normally assigned to general-purpose computers. However, features that enhance the computing speed, input/output, computing capacity, programming convenience or some other requirement of your application may be added in modular form. Your current and near future applications define a starting point – an initial configuration. You can start with a system that is used for remote inquiry and local batch processing or you can select a system that will handle a combination of data collection and distribution tasks, inquiry on transaction processing, message switching, business and scientific processing. This system can include multiple-access storage modules, programmable input-output controllers, and an extensive array of local and remote peripheral devices.

AN EXCELLENT LONG-TERM INVESTMENT

The expansion of functional capability by adding features may be accomplished without sacrificing any previously developed application software because any version of the computer contains features that are a subset list of those in a more capable configuration.

A data processing system with a high performance/cost ratio is attainable when the AN/UYK-15 serves as a foundation. Simplicity and compatibility, combined with functional and physical flexibility, characterize the AN/UYK-15 in all of its available configurations. Simplicity, which is accomplished by the power and flexibility of its instructions, provides simpler and more efficient program generation and implementation than any other computer in its class. This high quality and maintainable computer, characteristic of Univac products, will provide the faithful and dependable service expected in a processing system. Reliability and maintainability, two attributes of excellence historically demonstrated in Univac military products, are incorporated in the AN/UYK-15 design. Further savings can be realized by advanced planning and performing feasibility studies with the functional equivalent and lower cost Univac 1616 before implementing the militarized AN/UYK-15.

REAL TIME APPLICATIONS

Functional characteristics of the AN/UYK-15 make it as ideally suited to the implementation of dedicated real-time applications as to the performance of stand-alone and distributed process systems. A new, hardware initiated, 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, less memory space is used 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 rate environments encountered in time-critical, real-time systems associated with communications, radar, telemetry display controlling or on-line process control applications.

Shipboard Defense Systems Applications

Processing all raw data available from an amphibious task force and a ship's systems is a huge assignment for a command and control (C&C) system. The AN/UYK-15 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-15 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.

Signal Processing

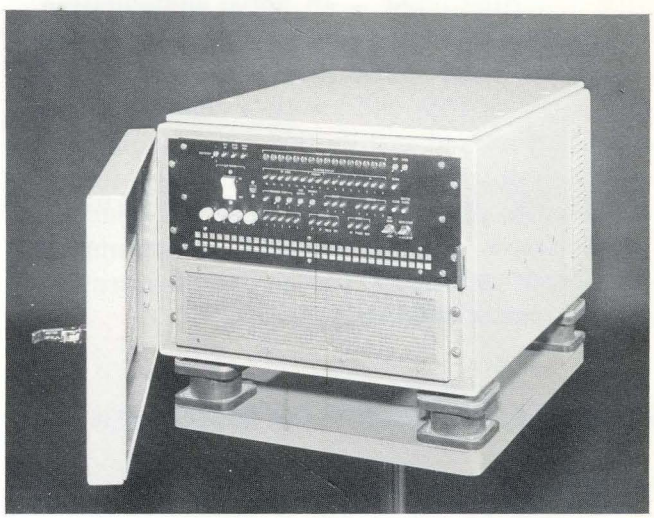
A major shipboard application is processing radar, sonar and beacon signals. In this application the systems provide a continual input of data in a real-time environment. High rate processing and fast reaction time is required to determine targets, direction, distance and other information. This real-time data processing task is handled easily by the AN/UYK-15. 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 which can be expanded by adding input/output controllers.

Control Systems

In addition to weapons control systems, other control systems normally found on board ship include air traffic, radar, electronic countermeasures and navigation. Complex control systems, as with signal processing, required high computational capabilities. While the quantity of input data is lower, input is received from more than one source. Here again the AN/UYK-15 qualifies for this application. The number of input/output channels can be expanded as required. 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 instructions.

Other AN/UYK-15 Military Applications

- Message Handling – receiving, logging and forwarding.
- Array Radar Beam Forming and Steering
- Navigation
- Management Information
- Logistics
- Telemetry
- Ship Instrumentation
- Range Tracking



SPECIFICATIONS

SUMMARY

Militarized Construction
General-purpose, 16-bit digital computer
Real-time capability
Physically and functionally modular and expandable
MSI (medium scale integration) elements
Integral cooling blowers and power supplies

CENTRAL PROCESSOR

Two's complement arithmetic
8-bit byte, 16-bit and 32-bit operands
16 high-speed general purpose registers
Program status register
Single bus functional interface
Direct addressing capability to 65K words or 131K bytes
4-Level interrupt processing (hardware serviced)
16-bit and 32-bit instructions – in any mix
Basic instructions – 4 formats

Add	750 nanoseconds
Multiply	3.75 microseconds
Divide	3.75 microseconds

Indexing via general registers
Load and store multiple registers
Processor – peripheral channel
Up to 16 input/output controllers (multiplexed)

MAIN STORAGE

Expandable – 8K to 65K words in 8K increments
16-bit words
Independently accessible memory banks
Read/restore cycle time – 750 nanoseconds is optimal
Asynchronous timing – request and acknowledge signals

POWER SUPPLY

115V, 1 phase, 60 Hz input (blowers)
115V, 1 phase, 47 Hz to 500 Hz input (logic) and
Regulated dc output to CP, IOC and Memory

INPUT-OUTPUT CONTROLLER (1 to 4 OPTIONAL)

Asynchronous operation
Processor-initiated program chain
10 instructions, format same as for CP
IC buffer control memory (64 words)
4 input and output channel groups (1 to 4 groups)
Parallel 16-bit channel interface
8-bit byte, 16-bit word or 32-bit dual-channel transfer
Interface voltage levels – 4 channel groups
-3.0 volt, 3.5 volt or -15.0 volts
Power supplied by Central Processor
(100 watts maximum)

WORD TRANSFER RATES (Thousand words per second)

Interface & Voltage (Type)		Number of Channels				
		1	2-4	5-8	9-12	13-16
– 15V (NTDS)	IN	41.6	41.6	83.3	124.9	166.6
	OUT	41.6	41.6	83.3	124.9	166.6
+ 3.5 (A NEW) and – 3.0 (NTDS)	IN	190.	250.	500.	750. *	1,000.*
	OUT	190.	250.	500.	750. *	1,000.*
– 3.0 (1108)	IN	667.	1,300.*	1,300.*	1,300.*	1,300.*
	OUT	667.	1,300.*	1,300.*	1,300.*	1,300.*

*Maximum total is 1,300K words per second

I/O Channel operation priority
First level by channel
Second level by function

PHYSICAL

Ship/shore environment: MIL-E-16400F
Temperature Range
Operating: 0°C to 50°C
Storage: -62°C to +75°C
Relative Humidity: to 95%
Size (inches)
Height: 14.4 plus 3.5" shock mounts
Width: 20.75
Depth: 25.75
Weight: Approximately 170 pounds

MODULAR ARCHITECTURE

Functionally, the AN/UYK-15 architecture is organized around a central exchange bus. Transfers and manipulation of data are accomplished through the central bus. The various functional elements accept bit configurations from the bus, interpret or manipulate them, and when appropriate, return bit-configured information to the bus for acceptance by another functional element. This architectural technique allows great flexibility in tailoring a system to meet the requirements of specific applications (see Figure 2).

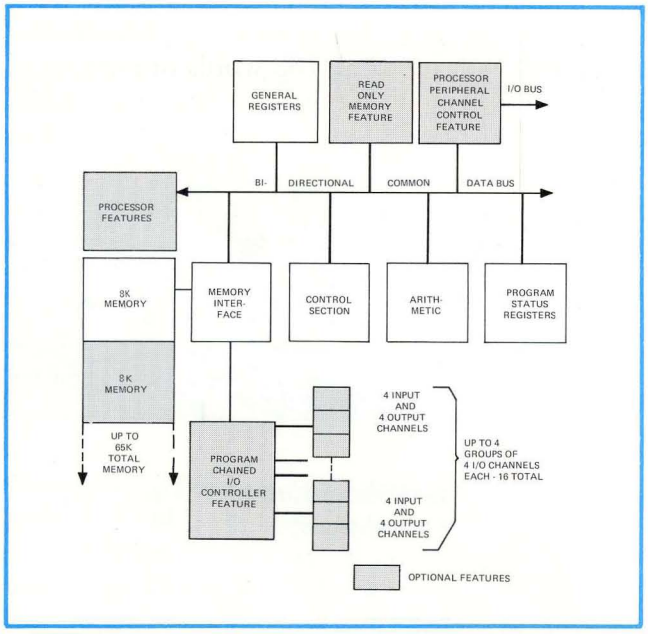


Figure 2. Functional Diagram

Physically, the AN/UYK-15 is assembled on two chassis – the central processor (CP), input/output controller (IOC) chassis and the memory chassis – that are packaged in a tabletop size, aluminum cabinet. Each chassis has its own power supply and cooling blowers. Exhaust air vents are located on the side cabinet panel near the rear. The operator/maintenance panel and air intake grille occupy the front face of the computer.

The basic version can be expanded on site by adding modular features, both in hardware and software, to the existing system. More sophisticated arithmetic and computer control operations are improved by incorporating applicable logic functions. Specialized and/or expanded input and output requirements may be supplied simply by adding an appropriate input/output controller module and the desired peripheral devices. As internal storage requirements increase, the random access memory can be expanded in 8,192 word increments. Software modules that encompass the added features are then integrated to expand the system capability to its fullest (see Figures 3 & 4).

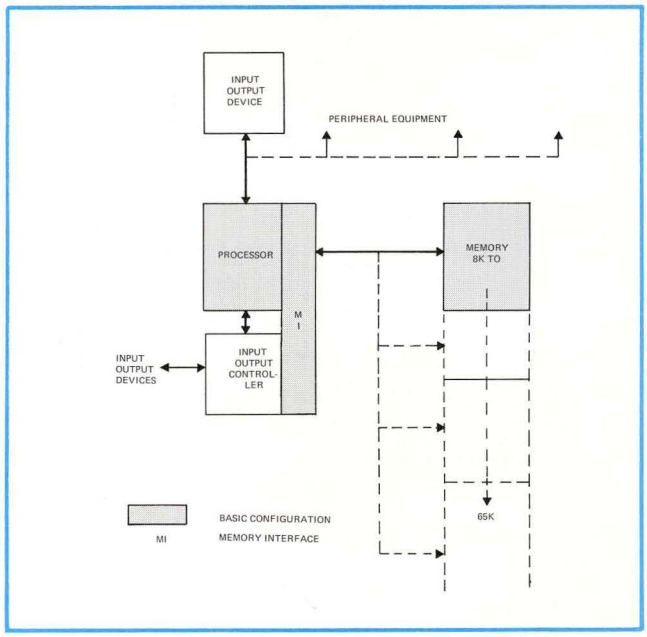


Figure 3. Basic and IOC Configuration

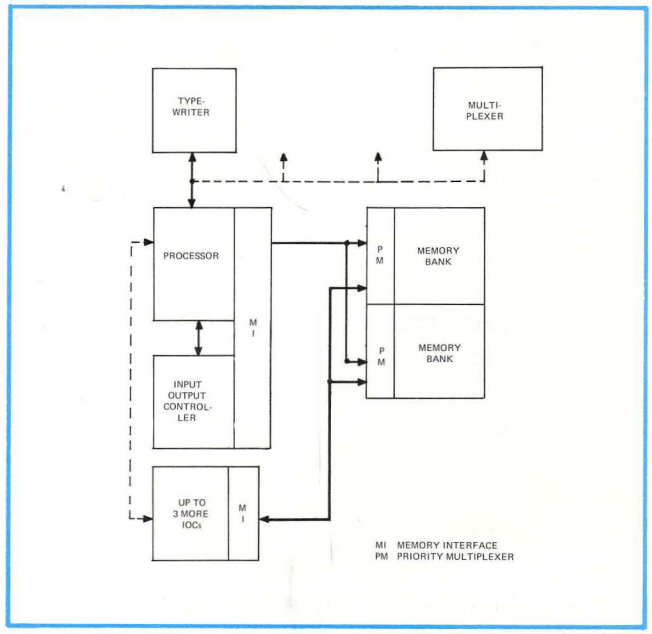


Figure 4. Multi-Port Memory Configuration

New advances in processing speech patterns, digital picture transmissions, sonar and radar signals, have utilized the fast fourier transform (FFT) quite effectively. The hardwired FFT feature for the AN/UYK-15 is a separate module that occupies a single drawer chassis and can be incorporated in a three or four drawer computer cabinet. Functionally the FFT module is programmed and controlled like an IOC feature and requires an independent access to memory (see figure 5).

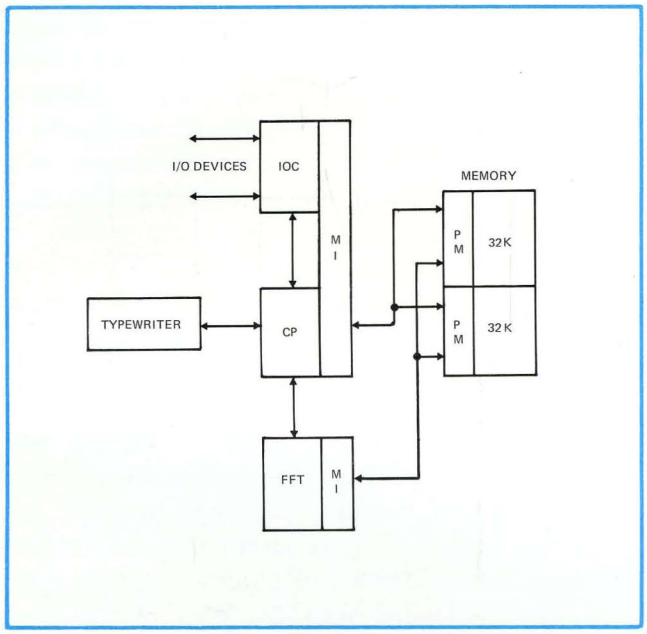


Figure 5. FFT-CP/IOC Configuration

CURRENT TECHNOLOGY – Medium Scale Integration (MSI)

Univac uses MSI devices for the AN/UYK-15 logic. Printed-circuit cards with 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.

CONSTRUCTION

Design and construction of the AN/UYK-15 is centered around a selection of high quality components and precise manufacturing processes. 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 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. The size of a computer with a CP, IOC, up to 32K word memory, power supplies, and cooling system is approximately 14 inches high x 20.75 inches wide x 25.75 inches deep and weighs no more than 200 pounds. The computer with an 8K word memory uses approximately 400 watts.

A building block method of construction is used to assemble a computer in any of its configurations. A computer with up to 32,768 words of memory, 4, 8, 12, or 16 input/output channels under buffer control, and one processor-controlled channel are contained on two chassis in a single tabletop cabinet. Each chassis can be withdrawn from the cabinet by removing the front securing screws and pulling out the "drawer".

The central processor, control panel, input/output controller, power supply, and space for related features are contained in one chassis (CP-IOC chassis). Memory stacks, associated electronics, power supply, and space for optional multiple access ports (priority multiplexer) are contained in a chassis. Interconnecting wiring is provided through connectors on the rear wall of each chassis. These connectors mate with the interchassis wiring harness and the input/output connectors (see Figure 6). The CP-IOC chassis is an assembly of 56-pin, female

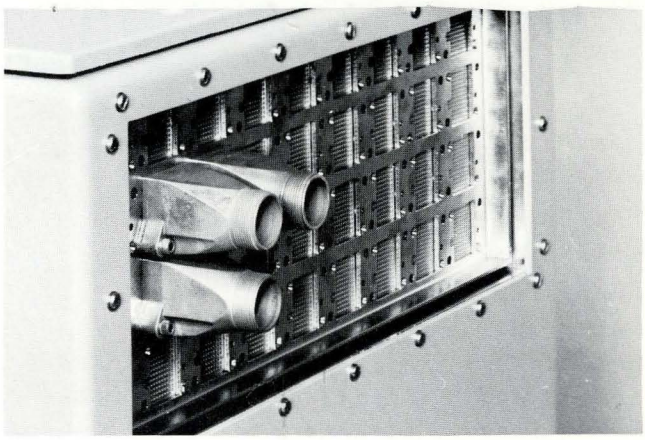


Figure 6. Interchassis and I/O Connectors – CP-IOC Chassis (Rear View)

connectors placed side by side and wire wrapped for all the logic in the central processor, the input/output controller and their related extra features (see Figure 7). The memory chassis has a space allocated for its related extra features. Memory stacks are interfaced to the common data bus via appropriate connectors. All input/output channel connectors are located on the back chassis panel.

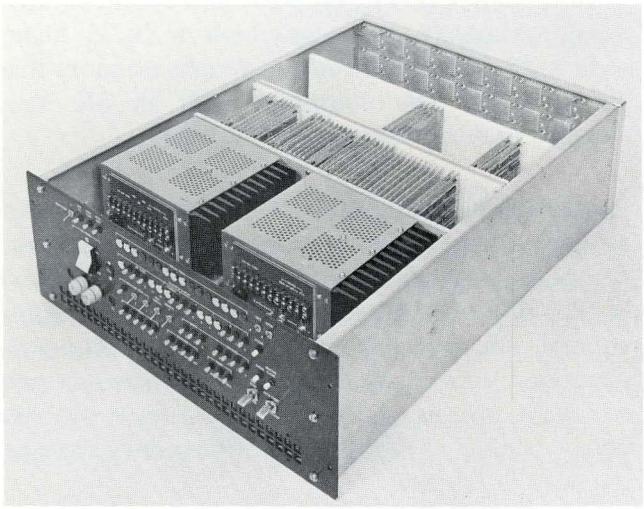


Figure 7. CP-IOC Chassis

Circuit Cards

The basic replaceable logic unit is a single layer, printed circuit card that is heavily populated with MSI devices. Circuits on each card terminate in a 56-pin connector mounted on the bottom of the card (see Figure 8). Test points are brought to the top of each card. Circuits are cooled by circulating air through heat sinks on the power supplies and between logic cards.

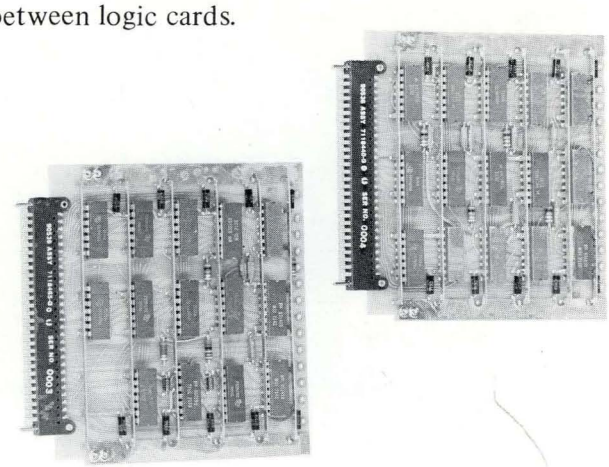


Figure 8. Printed Circuit (PC) Card

MAINTAINABILITY

Accessibility to replaceable items, easy testing, and quick malfunction localization are essential to good and efficient maintenance. Univac design includes these features. The lowest recommended maintenance level is the printed circuit card. When the cabinet top is opened, all central processor and input/output controller logic cards are accessible. Test points are exposed. Any printed circuit card is removable with a simple tool. Each card has its unique, labeled position and is keyed 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 air intake grille exposes the memory chassis test points and the circuit boards. After circuit malfunctions have been localized, the memory assemblies and logic boards can be pulled out of their slide mountings for servicing. Memory stacks and power supplies can be removed and replaced with simple tools.

Maintenance diagnostic routines of varying capabilities are available for quick malfunction localization. Some are included with the optional read only memory. The more comprehensive program packages are available on external storage media and can be loaded into the computer memory when needed.

ADAPTIVE PACKAGING

Univac design has provided for a variety of possible cabinet configurations. Internal assemblies are so designed that they can be adapted to a variety of environmental conditions – packaged in a cabinet that meets rugged industrial applications, limited military or mobile installations, 19-inch rack mounted versions and fully militarized units that may be subjected to high shock and vibration. Actual physical size of each type depends on the amount of memory and on other options that are included. All types maintain the modular architectural philosophy.

Configurations that specify more than one IOC or more than 32,768 words of core storage require additional chassis and larger or additional cabinets.

ENHANCEMENTS

Features of the AN/UYK-15 computer provide functional adaptability for many application requirements. Some of the available features increase its capacity, some enhance its flexibility, and others provide functions required by certain applications. The computer is designed to accommodate the following features:

Central Processor Features

Status register #2
Breakpoint
Additional general register sets
Real-time clock and interrupt clock
Power protection and recovery
    Automatic master clear
    Automatic restart
NDRO memory (read only memory)
Processor-peripheral input/output channel
One to four input/output controllers
Unary instructions
    Square root
    Reverse register
    Count ones
    Scale Factor shift
CORDIC (coordinate rotation digital computer) arithmetic
Processor-memory parity checking

Main Memory Features

Multi-port priority multiplexer
Parity

Input/Output Controller Features

Parallel channels
Serial channels per MIL-STD-188C
Serial channels per EIA-STD-RS232C
Intercomputer channels
Peripheral input channel
Independent memory interface

Power Supply Features

60-Hz, 1-phase 115V or
400-Hz, 1-phase 115V input power
Power fault sensors (for power protection and recovery)

TABLE 1. ASSIGNED MEMORY ADDRESSES

Assignment	Addresses (octal)			
NDRO Memory*	00-77 and 300-477			
For Processing	CP-0	CP-1*	CP-2*	CP-3*
Class IV Interrupts	100-107	500-507	600-607	700-707
Class III Interrupts	110-117	510-517	610-617	710-717
Class II Interrupts	120-127	520-527	620-627	720-727
Class I Interrupts	130-137	530-537	630-637	730-737
For IOC Operation	IOC-0*	IOC-1*	IOC-2*	IOC-3*
Command Cells	140-143	144-147	150-153	154-157
Reserved	160-177	----	----	----
External Interrupt Word Storage	200-217	220-237	240-257	260-277
Buffer Control Words for Processor-Peripheral Channels	1000-1777			

*Addresses are assigned as indicated when these features are included in the system. Otherwise the locations are used for general storage.

FUNCTIONAL ARCHITECTURE

Main Memory

Main memory is an assembly of 8192 sixteen-bit word stacks of magnetic core storage with a 750-nanosecond read-write cycle time. One such stack, with its reading, writing, and addressing circuits, is the basic increment for memory size selection and expansion. Two or more modules (within the 65K maximum memory size) may be united to form an independently addressable "bank" that has one interfacing, time-shared access port and an integral timing clock. The bank interface operates in an asynchronous, request-acknowledge mode.

Optional memory interfaces are available in the computer to match the requirements of optional memory systems (e.g., core, plated wire, modular, future technological advances). Computers that must operate in a military environment can be supplied only by using memory systems that meet military specifications.

A Central Processor-Memory Parity feature provides automatic checking of data and instruction transfers between memory and central processor on an 8-bit byte basis.

Multi-port Priority Multiplexer

Overall processing throughput can be increased considerably by incorporating the priority multiplexer feature which provides four access ports, each with a different priority level, for a memory bank. Two memory banks with this feature would allow an IOC to communicate with one bank of memory during the same time period the CP/IOC communicates with another. Three such memory configurations would provide a triple overlapping capability (i.e., two IOCs and one CP/IOC). Each user – CP/IOC or IOC – however, must have its own memory interface logic to optimize the priority multiplexer.

A priority network in the multiplexer gives prior service to the user connected to the higher priority port in case of two or three simultaneous requests. Memory time is, therefore, shared when these simultaneous requests occur.

Memory Address Allocations

Main memory is used for storage of programs, constants, and data. 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.

Input/Output Controller Feature

One to four optional Input/Output Controllers (IOC) may be attached to a central processor when applications require:

Large capacity input/output capability
(An IOC can transfer data at memory speeds on an 8-, 16-, or 32-bit interface)
Direct access to memory for input/output devices
Independent input/output control
Different channel interfaces or operating modes.

An IOC relieves the CP of the computer-peripheral communication burden and also increases the overall input/output capacity. The IOC permits integrating an AN/UYK-15 into a system that has an input/output equipment complex established. When an input or output related function is required, the CP merely directs the IOC to a stored program and then returns to its own processing tasks. The IOC takes its instructions from the stored program and performs accordingly. A controller communicates with other units in the system over three different interfaces.

- IOC – CP interface (see Figure 9)
- IOC – Peripheral equipment interface (see Figure 10)
- IOC – Memory interface

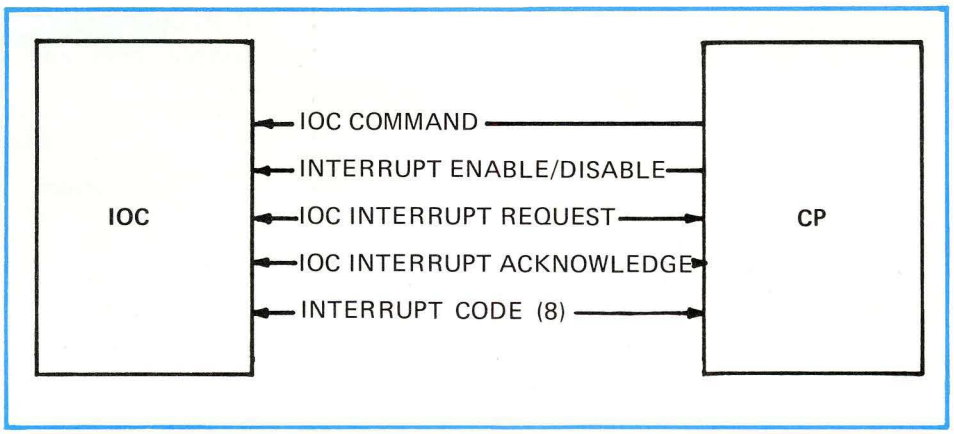


Figure 9. IOC to CP Interface

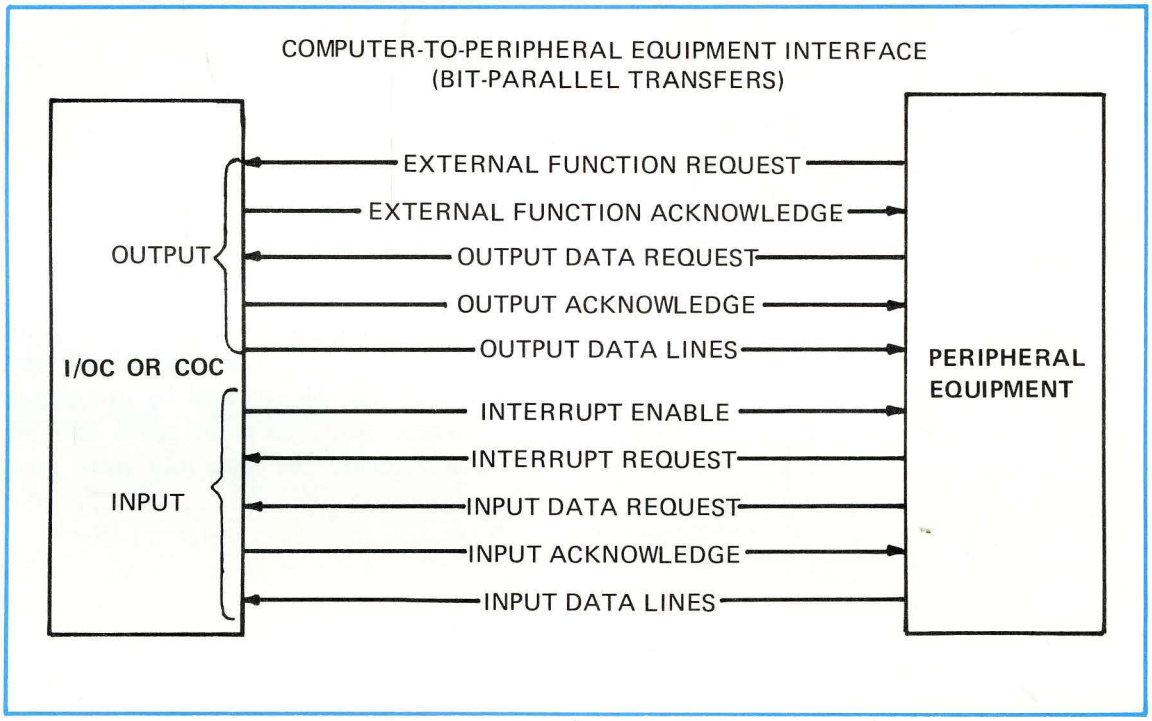


Figure 10. IOC – Peripheral Equipment Interface

The complete IOC-peripheral equipment interface has 1, 2, 3, or 4 groups of 4 input and 4 output, 16-bit parallel 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) in a dual channel communication mode. All input/output activity is asynchronous, and the timing is dependent on the speed of the peripheral device. If a serial interface is required for communications circuits, that interface can be provided in 2 -channel groups. In this case the IOC performs the necessary serial-to-word and word-to-serial conversion. Control signals and interface lines meet the electrical signal characteristics specified in EIA Standard RS-232 C or MIL-STD-188 C.

Interface voltage levels on parallel transfer channels can be supplied in a -15 volt level, a -3 volt level, a +3.5 volt level or the 1108A compatible -3 volt level. The serial transfer voltage level is a bipolar nominal ± 3 volt level.

Two IOC-Memory interfacing features are available. One method provides the IOC with an access to memory via the CP-memory interface on a time-shared basis with the CP. Priority is given to the IOC in case of simultaneous requests. The

other interface provides an IOC or an FFT module an independent, direct access to memory. In this configuration a separate memory interface is included in both the CP/IOC and the IOC; and the priority multiplexer (multiport memory interface) is included with each memory bank.

The IOC-CP interface provides the communication control lines and interrupt data paths. When the CP executes the I/O command instruction (Code 35, RR Format), the IOC Command Request line signals the IOC to execute the instruction in its command cell. If the CP program can honor Class III interrupts, it does so on the Interrupt Enable/ Disable line via the status register. The enabled line allows the IOC to interrupt the CP program by identifying the type of interrupt on the two Interrupt Type lines and setting the IOC Interrupt Request line. If a channel is involved, that channel number is set on four I/O Channel Designator lines. The CP responds on the IOC Interrupt Acknowledge line and disables Class III and Class IV interrupts when it honors the request.

Each two-way, IOC-peripheral equipment 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.

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). As soon as the output register is available, the IOC transfers the command word to the device. 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.

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.

General Registers

Each central processor has at least 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. They provide for extremely rapid processing of parameters or data by decreasing the number of required main memory references. The contents of any number of registers 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:

an accumulator for arithmetic, shift, and logical functions

an index register for address and operand modification

a temporary storage location for addresses, operands, etc.

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.

UNARY Feature

The UNARY feature provides extra instructions that manipulate the contents of general registers on a bit basis. These include such functions as set, clear and test bit, count ones, reverse register, scale factor shift and square root.

Additional General Register Feature

When one, two or three additional general register sets are included in the processor as an option, a program-controlled 2-bit field in the status register selects the set that is used for processor operations.

Additional general register sets provide 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 more sets are 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) clear the P-register and load it 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 Interrupt Clock Feature

The RTC-INT clock feature provides two program-controlled interrupts via two high-speed registers; one used as RTC count-up storage and the other as INT clock count-down storage. This feature and associated controlling instructions are useful for program timing and for synchronizing program segments with real-time events. The 16-bit registers can be loaded, read, enabled, or disabled by programmed instructions. An RTC oscillator, which has an accuracy of ± 2 Hz in 10 seconds, runs continuously and controls the counting speed of both registers. Its frequency is optional in the range $f \times 2^n$
where f = 1,000 Hz, 1,024 Hz, 10,000 Hz or 10,240 Hz and n = 0, 1, 2, . . . 7.

When enabled by the appropriate instruction (code 03 RR Format; m = 10), the RTC register counts up at the rate of the RTC oscillator. As the register overflows (changes 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 INT clock register count-down function is enabled by executing the Load and Enable Interrupt Clock instruction (code 03 RR Format, m = 12) which also loads the register with a starting point. When the contents of the register change from one to zero, an INT clock interrupt (class II, priority 6) is generated, the count-down function is disabled, and control is transferred to the appropriate processing routine.

Breakpoint Feature

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

Power Protection Feature

The Power Protection and 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 VOUT (voltage out) signal is generated. When the CP senses the VOUT signal, it generates a power fault interrupt, suspends the normal program sequence, and transfers control to the respective interrupt entrance register. The entrance register instruction transfers to a routine that stores the contents of all working registers and terminates in an instruction (operation code 40 RX format, a = 6 and m = 0) that is addressed to jump to itself as long as the VOUT signal is held. After the signal is removed (power returns to normal), the instruction allows the routine to go on, restore the working registers and return control to the program that was interrupted at the point of interruption.

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 (automatic start) is selected on the control panel, the CP generates an auto-start interrupt that causes execution of the instruction immediately following the 40 RX format instruction, mentioned above.

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

NDRO Memory Feature

A block of 192 non-destructive read-out (NDRO) memory words can be provided in the CP. The programs contained in the NDRO memory are fixed at the time of manufacture and cannot be changed by computer read and write operations. Addresses assigned to NDRO memory (octal locations 00 through 77 and 300 through 477) parallel similarly numbered 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 should always be available to the computer. These usually include initial load routines (Bootstrap) and some hardware diagnostic routines.

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 re-entering 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.

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) Interrupt lockout designator:

bit 0 = Class IV
bit 1 = Class III
bit 2 = Class II
bit 3 = Class I

When a lockout bit is cleared, the respective class interrupt is locked out (cannot be honored).

b) The condition code (bits 9 and 8) indicates the results of arithmetic and compare instructions as shown in Table 2.

c) 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.

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

TABLE 2. CONDITION CODE INDICATIONS

Condition Code		Indicated Results of	
Bit No.	Value	Arithmetic Operation	Compare Operation
8	0	Zero	Equal
8	1	Not Zero	Not Equal
9	0	Positive	$R_a \geqslant R_m$ or Y
9	1	Negative	$R_a < R_m$ or Y
Combined Value			
Bit 9	Bit 8		
0	0	Zero	$R_a \geqslant R_m$ 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

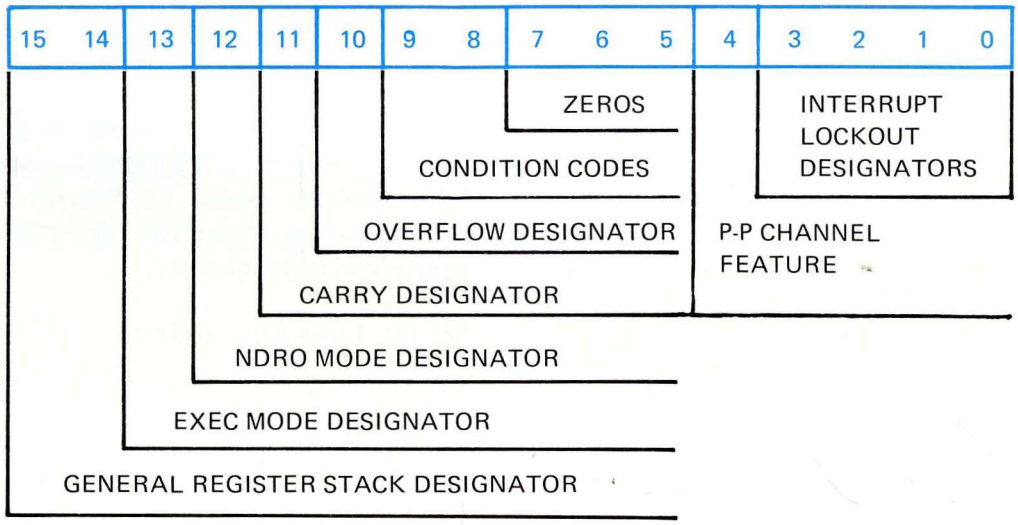


Figure 11. Status Register #1 Format

e) 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

f) The EXEC Mode (bit 13) is set when the CP is operating in the executive mode.

g) The general register stack designator (bits 15 and 14) selects the stack of 16 general registers that will be selected by the a- and m-designators in the instruction word. Binary values, in bits 15 and 14, of 00, 01, 10, and 11 select register stack number 1, 2, 3, and 4 respectively. Stacks numbered 2, 3, and 4 are added features in the AN/UYK-15 central processor.

Status register bit content can be set or cleared by executing the Load Status Register 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.

Status register #2 is a feature that is used in conjunction with processing interrupts from the processor/peripheral channel and the memory lock and key feature. The 16-bit register has the following format and field assignment:

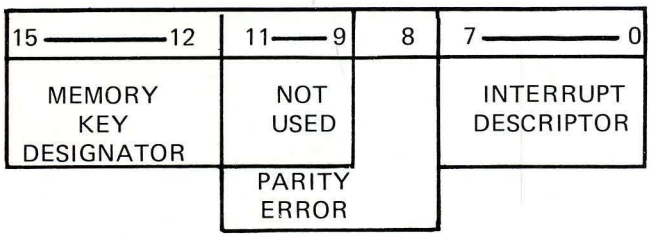


Figure 12. Status Register #2 Format

Interrupt descriptor field receives the contents of BUS IN during the processing of a Class IV interrupt. Bit 8 is set if the BUS IN has bad parity. The memory key designator is used to specify a memory area for lockout protection when the lock and key feature is used. Unassigned fields may be used for future functional options.

FUNCTIONAL OPERATION

Instructions

Instructions defining operations for the AN/UYK-15 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 AN/UYK-15 far beyond the mini-computer capability; it is truly a medium scale processor. Table 3 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 an input/output controller.

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 provides a left shift function which positions the word for greatest significance and counts the number of digit positions shifted. It is used effectively in floating point arithmetic routines that convert fixed point numbers to floating point format.

A Square Root instruction is useful in scientific applications.

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, computer word 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 13 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 memory references, indexing type operations and those that provide programmers the convenience of listing 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 AN/UYK-15 computing speed and also occupy less memory space.

TABLE 2. REPERTOIRE OF INSTRUCTIONS

OCTAL CODE f	a	m	DESCRIPTION	Exec Time Memory Cycles RR	RI	RK	RX	NOTE REF.
00			Byte Load	–	–	–	3	
01			Load	1	2	2	3	
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 by2	1	–	–	–	
		13	Decrease R_a by 2	1	–	–	–	
02			Load Double	–	3	–	4	
03		0	Executive Return	1	–	–	–	1
		1	Store Status Register 1	1	–	–	–	1
		2	*Store Status Register 2	1	–	–	–	1
		3	*Store RTC	1	–	–	–	1
		4	Load P	1	–	–	–	1
		5	Load Status Register 1	1	–	–	–	1
		6	*Load Status Register 2	1	–	–	–	1
		7	*Load RTC	1	–	–	–	1
		10	*Enable RTC	1	–	–	–	1
		11	*Disable RTC	1	–	–	–	1
		12	*Load and Enable INT. Clock	1	–	–	–	1
03			Load Multiple	–	–	–	2	2
04		0	*Square Root	1	–	–	–	3
		1	*Reverse Register	1	–	–	–	4
		2	*Count Ones	1	–	–	–	4
		3	*Scale Factor Shift	1	–	–	–	5
04			Byte Load and Index by 1	–	–	–	3	
05			*Set Bit	1	–	–	–	
05			Load and Index by 1	–	2	–	3	
06			*Clear bit (Zero Bit)	1	–	–	–	
06			Load Double and Index by 2	–	3	–	4	
07			*Test Bit	1	–	–	–	
07			Load PSW	–	3	–	4	
10			Logical Right Single Shift	1	–	2	–	6
10			Byte Store	–	–	–	3	
11			Algebraic Right Single Shift	1	–	2	–	6
11			Store	–	2	–	3	
12			Logical Right Double Shift	1	–	2	–	6
12			Store Double	–	3	–	4	
13			Algebraic Right Double Shift	1	–	2	–	6
13			Store Multiple	–	–	–	2	
14			Algebraic Left Single Shift	1	–	2	–	6
14			Byte Store and Index by 1	–	–	–	3	
15			Circular Left Single Shift	1	–	2	–	6
15			Store and Index by 1	–	2	–	3	
16			Algebraic Left Double Shift	1	–	2	–	6
16			Store Double and Index by 2	–	3	–	4	
17			Circular Left Double Shift	1	–	2	–	6
17			Store Zeros	–	2	–	3	
20			Subtract	1	2	2	3	
21			Subtract Double	1	3	–	4	1
22			Add	1	2	2	3	
23			Add Double	1	3	–	4	1
24			Compare	1	2	2	3	
25			Compare Double	1	3	–	4	1
26			Multiply	1	2	2	3	7
27			Divide	1	2	2	3	7
30			AND	1	2	2	3	
31			OR	1	2	2	3	
32			Exclusive OR	1	2	2	3	
33			Masked Substitute	1	2	2	3	
34			Compare Masked	1	2	2	3	
35			I/O Command	1	–	–	–	
35			Biased Fetch	–	2	–	3	
35			Execute Remote	–	–	2	–	8

OCTAL CODE f	a	m	DESCRIPTION	Exec Time Memory Cycles RR	RI	RK	RX	NOTE REF.
36			Processor-Peripheral Command	–	–	–	3	
37			*CORDIC	–	–	–	–	
40	0		Jump CC Zero/Equal	1	–	2	3	9
	1		Jump CC Not Zero/Not Equal	1	–	2	3	9
	2		Jump CC Pos/Greater Than or Equal	1	–	2	3	9
	3		Jump CC Neg/Less Than	1	–	2	3	9
	4		Jump on Overflow	1	–	2	3	9
	5		Jump on Carry	1	–	2	3	9
	6		Jump Power Out	1	–	2	3	9
	7		Jump Bootstrap 2	1	–	2	3	9
	10		Jump	1	–	2	3	9
	11		Jump Stop	1	–	2	3	9
	12		Jump Stop Key 1	1	–	2	3	9
	13		Jump Stop Key 2	1	–	2	3	9
	16		Jump P-P Channel Active	1	–	2	3	9
40			Local Jump	–	#1	–	–	10
41			Index Jump	1	–	2	3	9
42			Jump and Link Register	1	–	2	3	9
43			Local Jump and Link Memory	–	#2	–	–	11
43			Jump and Link Memory	–	–	2	3	12
44			Jump Register Zero	1	–	2	3	9
44			Local Jump Equal	–	#1	–	–	10
45			Jump Register Not Zero	1	–	2	3	9
45			Local Jump Not Equal	–	#1	–	–	10
46			Jump Register Positive	1	–	2	3	9
46			Local Jump Greater Than or Equal	–	#1	–	–	10
47			Jump Register Negative	1	–	2	3	9
47			Local Jump Less Than	–	#1	–	–	10
70		0	Master Clear	1	–	–	–	
		2	Enable All Chains	1	–	–	–	
		3	Disable All Chains	1	–	–	–	
		4	Enable All External Interrupts	1	–	–	–	
		5	Disable All External Interrupts	1	–	–	–	
		6	Enable All External Monitors	1	–	–	–	
		7	Disable All External Monitors	1	–	–	–	
		10	Master Clear	1	–	–	–	
		12	Enable Chan. a Chain	1	–	–	–	
		13	Disable Chan. a Chain	1	–	–	–	
		14	Enable Chan. a Ext. Int.	1	–	–	–	
		15	Disable Chan. a Ext. Int.	1	–	–	–	
		16	Enable Chan. a Ext. Int. Monitor	1	–	–	–	
		17	Disable Chan. a Ext. Int. Monitor	1	–	–	–	
70			Initiate IO Transfer	–	–	–	4	
71			Initiate Chain (Comm)	–	–	2	–	
71			Load CM (Chain)	–	–	2	3	
71			Write (Load) CM (Comm)	–	–	–	3	
72			Read (Store) CM (Comm)	–	–	–	3	
72			Store CM (Chain)	–	–	–	3	
73	0		Halt	1	–	–	–	
73	1		Interrupt	1	–	–	–	
73	1		Set Flag	–	–	–	3	
73	0		Clear Flag	–	–	–	3	

Footnotes:

* Optional instruction
\# RI, Type 1 Format
1 Add 750 nanoseconds
2 Plus the number of registers
3 Add 5.55 microseconds
4 Add 3.15 microseconds
5 Plus 750 nanoseconds and 150 nanoseconds times number of places shifted.
6 Plus 600 nanoseconds and 150 nanoseconds times number of places shifted.
7 RR and RK Format add 3.0 microseconds; RI and RX Format add 2.5 microseconds
8 Plus remote instruction time
9 RR Format add 450 nanoseconds; RK and RX Formats add 300 nanoseconds – if no jump, execution time is 1 cycle plus 300 nanoseconds.
10 Add 300 nanoseconds
11 Add 150 nanoseconds.
12 Add 150 nanoseconds – if no jump, execution time is 1 cycle plus 300 nanoseconds.

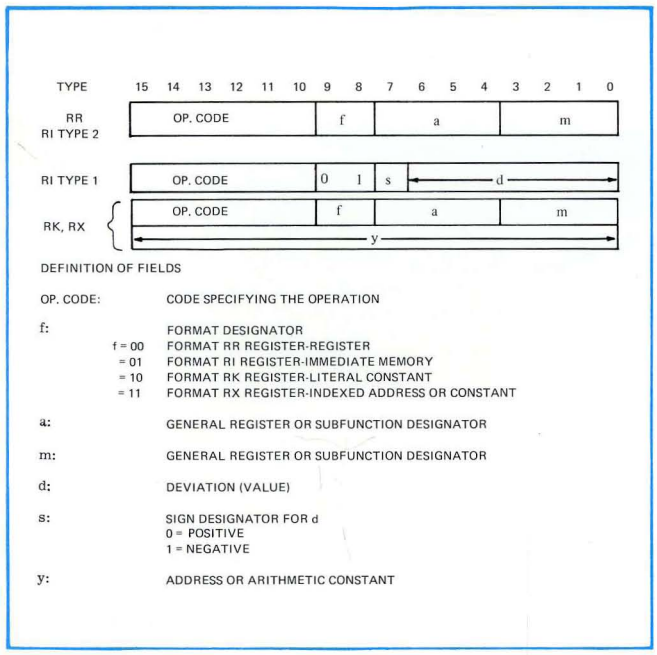


Figure 13. Instruction Word Formats

Memory Addressing

All locations in main memory up to 65,536 words are directly addressable by the CP and IOC. Both the sequential and random access methods are employed.

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 octal locations.

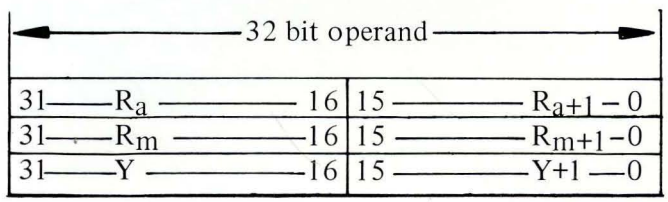
The IOC uses chain address pointer locations in its control memory as its instruction address counter. As in the CP, executing a single-word instruction advances the pointer by one and a double-word instruction advances it by two. If a jump effect is desired in the IOC programs, the chain address pointer is changed by executing a load control memory instruction.

Operand Addressing

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.

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 register designated R_a, R_m or the word in memory address Y, when selected by a "Double" instruction, is the most significant half of the operand. The word in the register designated R_{a+1}, R_{m+1} or the word in memory address Y+1, respectively, is the least significant half of the operand.



RR Format instructions perform operations involving general registers; no main memory references are made. The a- and m-designators select the general registers designated R_a and R_m, respectively, that are 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. When the s-designator bit is zero, the effective jump address Y = (P) + d; when the s-bit is one, the effective jump address Y = (P) - d.

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. When the instruction specifies a double-length operation, the computer logic selects the second general register designated R_{a+1}.

Single-length and double-length operand addressing is performed as follows:

> When m = 0, the operand address Y is equal to y. $m \neq 0$ selects a general register designated R_m and the operand address Y is equal to the sum of y and the contents R_m – i.e., y is indexed by the contents of R_m.

For double-length operations the address Y must result in an even number. The computer logic addresses the next sequential memory location, Y+1, for the second half (the least significant half) of the 32-bit operand. Register R_a and memory address Y contain the most significant 16-bits.

Byte (8-bit, half word) operand addressing requires a byte identifier – i.e., the upper byte or the lower byte in memory. The least significant (LSB) in general register R_m 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 is not used – byte position is not determined.
>
> $m \neq 0$ selects a general register designated R_m.
>
> The operand address Y is the sum of y and the contents of R_m shifted right 1 bit position – i.e. $Y = y + \frac{(Rm)}{2}$
>
> LSB of R_m = 0 designates the most significant half word in address Y as the operand byte.
>
> LSB of R_m = 1 designates the least significant half word in address Y as the operand byte.

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. All AN/UYK-15 system interrupts are classified in four 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 an appropriate processing routine from memory. Table 4 lists the interrupts, their classification and assigned 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 are locked out until released by the processing subroutine. 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.

Interrupt Processing

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

a. Terminates the current program sequence.

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

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).

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

	Function	Address Assignment to Class IV	III	II	I
b.	Stores the contents of P at address	100	110	120	130
	Stores the contents of SR #1 at address	101	111	121	131
	Stores the contents of SR #2 at address	102	112	122	132
	Stores the contents of RTC at address	103	113	123	133
c.	Reloads P with index[1] plus the contents of address	104	114	124	134
	Reloads SR #1[2] from address	105	115	125	135
	Reloads SR #2 from address	106	116	126	136
	Locations not used	107	117	127	137

1 See Figure 14 for index values.

2 SR #1 bits 3-0 control interrupt lockout or release

TABLE 4. INTERRUPT PRIORITY

Class	Priority Within Class	Interrupt	Binary Interrupt Code Generated
Class I Hardware Errors	1	Power Fault	000
	2	CP Memory Resume Error	001
	3	CP Parity Error*	010
	4	Auto Start*	011
Class II, Software Interrupts	1	CP Instruction Fault	000
	2	Privileged Instruction Error	001
	3	Memory Lock and Key Error*	010
	4	Executive Call	011
	5	RTC Overflow*	100
	6	Interrupt Clock*	101
	7	P-P Channel No Response*	110
	8	P-P Channel Address Failure*	111
Class III, I/OC Interrupts	1	External Interrupt	000
	2	Chain	010
	3	IOC Instruction Fault	100
Class IV, Proc./Periph. Interrupts	1	SERVICE IN Request	001
	2	STATUS IN Request	000

*Optional

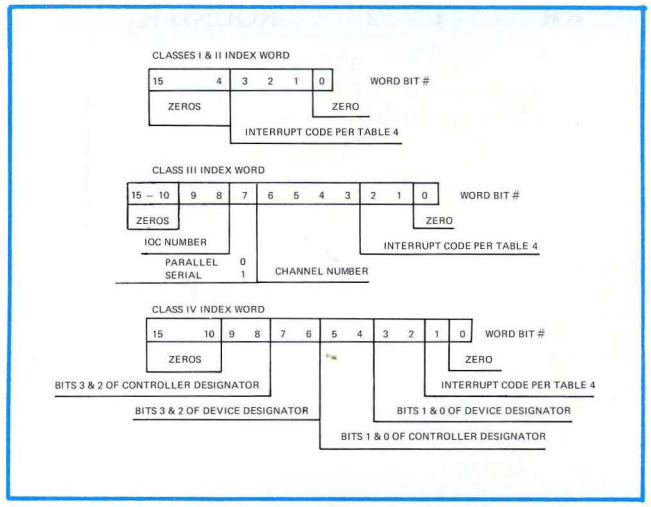


Figure 14. Interrupt Entrance Address Index

IOC Instruction Execution

Instructions in the IOC 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 a CP command.

Command Instruction

Each IOC is assigned a command cell in main memory from which it reads a Command instruc-

tion when requested by the C.P. The command cell consists of four addresses (see Table 1) that are assigned as follows:

1st location	Storage for 1st word of Command instruction
2nd location	Storage for 2nd word of double-length Command instruction (storage for y)
3rd location	Not assigned
4th location	Not assigned

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 notify the CP that the instruction was executed as requested (see Table 5 for a list of Command instructions). Because the channel activity is established by the chaining instructions and the CP can test for IOC execution of an instruction in the command cell, the CP 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 IOC – peripheral channels are executed under an active chain that is associated with each channel. Four integrated circuit memory locations are assigned to each channel (see Figure 15 for the format and application of each word).

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

B = 0 specifies the most significant bit

B = 1 specifies the least significant bit

Bit 13: Not used

TM: The Transfer Mode field specifies the type of I/O transfer as follows:

TM = 00: abort the transfer

TM = 01: transfer 8-bit bytes

TM = 10: transfer 16-bit words

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

Buffer Word 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 word count is decreased by one. When the count changes from 1 to 0, the buffer terminates.

TABLE 5. IOC INSTRUCTION LIST

Operation Code and Format	Instruction and Type		Execution Time (Memory Cycles)
	Command	Chaining	
70 RR	Channel Control	Channel Control	1
70 RX	N/A	Initiate Transfer	4
71 RR	Initiate Chain	Load Control Memory	2
71 RX	Load Control Memory	Load Control Memory	3
72 RX	Store Control Memory	Store Control Memory	3
73 RR	N/A	Halt/Interrupt	1
73 RX	N/A	Set/Clear Flag	3

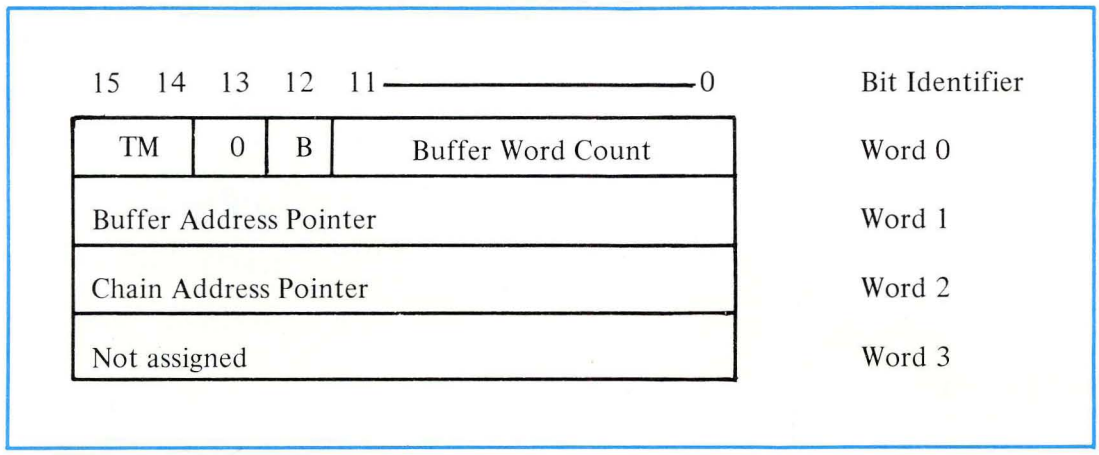


Figure 15. Control Memory Locations for One Channel

A beginning count of zero specifies the maximum number of transfers (4096).

Buffer Address Pointer specifies the memory address for the next input data, output data, or external function 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 IOC executes the Initiate Chain instruction from the command cell, the chain address pointer (word 2) is loaded in the control memory for a specific channel and that chain is activated. A chain address pointer specifies an address in main memory where the next 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 IOC can proceed to the next instruction in the chain.

IOC Buffer Operation

Memory area assignments for chained programs and buffer areas for each channel are pre-arranged by programmers for compatibility in:

Input and output data buffers

External functions and function buffers

Channel chains and associated chain address pointers.

Each time the executive program is called upon to initiate an input or output buffer it must command the IOC to execute a chain of instructions that

1. Command the external equipment, on the desired channel, to perform an input or an output.

2. Activate the channel and define the buffer area in main memory.

3. Halt the chain and/or interrupt the processor on completion of the buffer (buffer monitor) and halt the chain (terminate chaining action).

For example, to illustrate the interaction between the central processor, controller and memory during an I/O operation, assume that an output data buffer is to be initiated on the controller's channel 2. Refer to Figure 16 during the following discussion.

Prior to initiating any transfer action, the processor's executive program must place the current commands, chain instructions, and data in the proper locations in memory. The command cell addresses hold the first instruction which the IOC will execute.

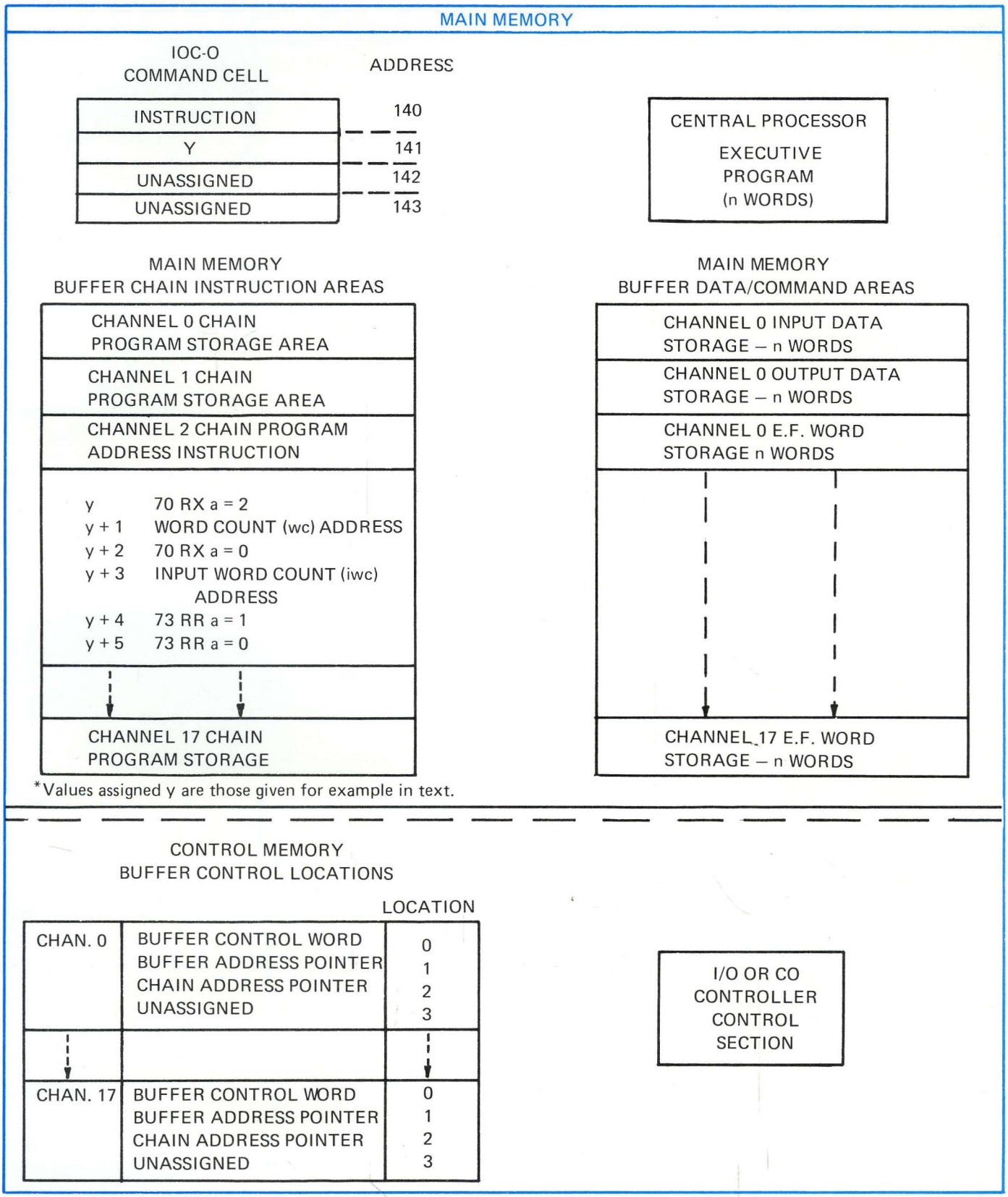


Figure 16. Main Memory and Control Memory Storage Areas

As shown in Figure 16, the memory addresses of the command cell for IOC-0 in this example are 140 through 143. The executive program loads cell locations 140 and 141 with the double-length initiate chain instruction (code 71 RK, a = 2, m = 0). The a-designator of the first word (location 140) specifies channel 2; y (in location 141) specifies the address of the first instruction in (main memory) the program chain for channel 2.

For this example, assume that the program chain contains the following instructions:

y: 70 RX a = 2 (External Function)

y+1: Word count (wc) address

y+2: 70 RX a = 0 (Input data)

y+3: Input word count (iwc) address

y+4: 73 RR a = 1 (Interrupt)

y+5: 73 RR a = 0 (Terminate chain)

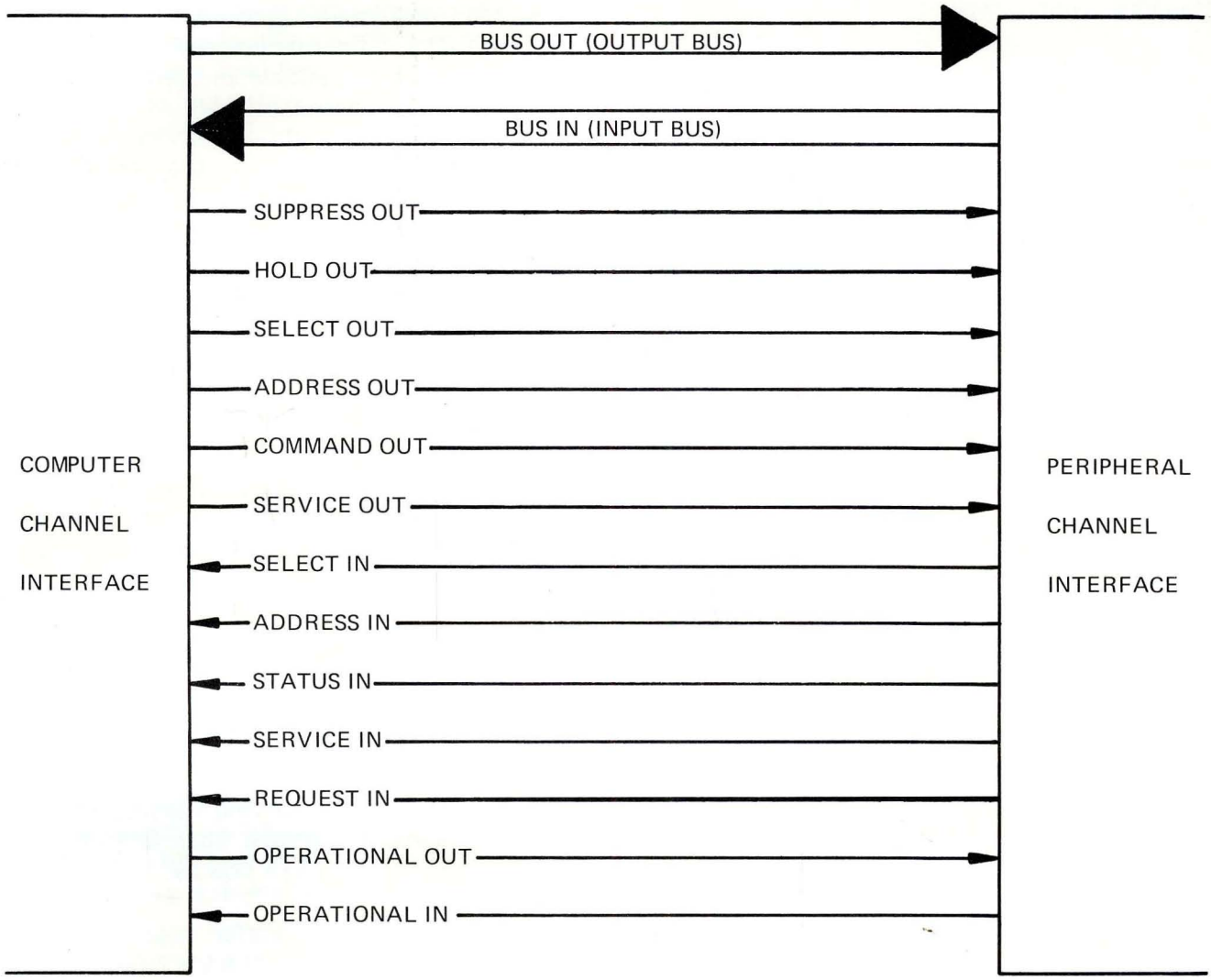
To initiate the IOC operation, the processor's executive program executes a code 36 RR format instruction which specifies controller 0 and sets its command request line.

- IOC-0 senses command signal, reads command cell 140 (wired address), clears bits 14 and 15 of location 140, transfers contents of location 141, "y", to channel 2 control memory channel address pointer and activates chain for channel 2.
- IOC reads the instruction from location y of the channel 2 chain.
- The 70 RX instruction executed, reads the wc address from location y+1, transfers the contents of addresses wc and wc+1 to buffer control words 0 and 1, respectively, and activates channel 2 external function buffer (a = 2).
- The buffer transfers one 16-bit external function word from EF word address to channel 2 and sets External Function line.
- When the word is transferred, the chain pointer being advanced to y+2, the IOC reads the 70 RX instruction from that location, transfers the contents of locations iwc and iwc+1 to buffer control words 0 and 1, respectively, and activates the data input buffer (a = 0) on channel 2.
- The chain stops until all input words are transferred. Because byte transfers were indicated, the B-bit in location 0 is toggled (changes state) and the word count is decreased by one each time a byte is transferred. When the word count reaches zero, the buffer terminates.
- When the buffer terminates, the channel 2 chain is activated and the IOC reads the instruction from y+4. Instruction 73 RR, a = 1 interrupts the CP with channel 2 identified on CP-IOC interface lines.
- IOC reads the instruction 73 RR, a = 0 from location y+5 which terminates the chain.

Processor/Peripheral Channel Feature

The AN/UYK-15 communicates with low transfer rate peripheral devices on the BYTE-oriented processor/peripheral channel. Figure 17 shows the interface lines that provide the information transfer paths and the control lines which identify the type of information transferred (i.e., data, commands, address or status), probe the channel devices and enable communication (interlocks).

There are no wires unique to any single control unit. The interface lines are shared by all control units and the information transmitted thereon is timed by signal/response sequences on the time-shared control lines. These sequences are interlocked in such a manner that the interface is insensitive to circuit delays except as they affect maximum data rates.



Purpose and interpretation of interface lines are:

BUS OUT — 9 lines: P, 7, 6, 5, 4, 3, 2, 1, 0
Bit #: P, 0, 1, 2, 3, 4, 5, 6, 7
Bit 0 is LSB; P is odd parity bit.

The channel transmits output data, I/O commands, and device selection codes to the control units via the BUS OUT. Timing as well as the nature of the information on the BUS OUT is signaled to the control unit(s) by activating one of the outbound control lines.

BUS IN — 9 lines: P, 7, 6, 5, 4, 3, 2, 1, 0
Bit #: P, 0, 1, 2, 3, 4, 5, 6, 7
Bit 0 is LSB; P is odd parity bit.

The control units transmit input data, device numbers, and status information to the channel via the BUS IN. Signals that identify and time the information on the BUS IN to the channel are transmitted on the inbound control lines.

Signals on control lines have the following meanings:

SELECT OUT (from computer) probes all control units serially from highest to lowest priority order. If none respond the last control unit propogates a signal back on the SELECT IN line.

SELECT IN line (from peripherals) indicates that no control unit responded to the ADDRESS OUT/ SELECT OUT. No unit recognized the address on BUS OUT.

ADDRESS OUT (from computer) indicates a device address on output bus (BUS OUT) – held until control activates ADDRESS IN for initial selection sequence.

ADDRESS IN (from peripheral) indicates that the address of the "currently selected" device is on the input bus (BUS IN).

SERVICE IN (from peripheral) indicates that the device control unit is "in service" and is ready to receive data or command or that it is transmitting data on the input bus.

STATUS IN (from peripheral) indicates that a status byte was placed on the input bus (BUS IN).

COMMAND OUT (from Computer);

1) in response to ADDRESS IN, indicates that a command word is on the output bus or, if no command word, to "proceed";

2) in response to SERVICE IN, during a data transfer sequence indicates "terminate";

3) in response to STATUS IN, indicates that the computer cannot accept transfers at this time – stack the status and try again when priority allows it.

OPERATIONAL IN (from peripheral) indicates that a device has been "selected" (line stays active for the duration of selection).

OPERATIONAL OUT enables interpretation of all channel outbound signals (Interlock); when the OPERATIONAL OUT signal drops all incoming signals must drop.

SERVICE OUT (from computer) indicates receipt of a byte (data or status) or that a data byte is on the output bus (BUS OUT).

The following functions may or may not be included in all processor-peripheral channels.

REQUEST IN indicates that one or more of the attached control units requires attention. A control unit with logic for this line activates the line to indicate that it will initiate an operation whenever it can capture the interface.

HOLD OUT provides for blocking SELECT OUT at all control units simultaneously rather than waiting for the SELECT IN response. The SELECT OUT signal is gated in each control unit with the HOLD OUT signal from the channel. When HOLD OUT drops, all control units must drop SELECT OUT (must not pass the signal on to the next control unit).

SUPPRESS OUT activated alone or in conjunction with other outgoing signals provides for

- Suppressing REQUEST IN from I/O devices
- Suppressing STATUS from control units holding "stacked" status or other suppressable status (dependent on control unit types)
- Indicating command chaining to control units, if activated when SERVICE OUT responds to STATUS IN.
- Indicating "buffer segment end" if activated when SERVICE OUT responds to SERVICE IN.

A channel initiates, directs, and monitors the flow of information between I/O devices and main storage and performs the required addressing functions. The channel also synchronizes I/O sequences to processor and memory timing.

An I/O control unit adapts the control signals supplied by the channel to the form required by a particular type of I/O device. The control unit performs any data conversions required to adapt the I/O device to the channel.

Up to 256 devices may be addressed via one interface, assuming that these devices may be controlled by no more than sixteen control units.

In the Multiplex Mode of operation, the channel

services several concurrently operating devices by assigning the interface to a device only long enough to transfer one (or a few) byte(s) of information and then services the other devices in a similar manner, if necessary, before servicing the same device again.

The interface allows multiplexed Burst Mode operation. A Multiplexer channel may be forced into Burst Mode by appropriate responses from control units.

During a Burst Mode operation, one control unit retains control of the interface during the execution of an entire I/O command (such as the transfer of complete blocks of data). Burst Mode allows a higher rate of data transfer than Multiplex Mode. Control units may be forced into Burst Mode by channels that operate only in Burst Mode.

Command Code Formats

Both the channel and the control units must be in agreement on some aspects of the command byte, and all control units should recognize the same codes for some basic commands. The basic format for command bytes transmitted during the Initial Selection Sequence is shown below:

COMMAND	BIT POSITION
	P 0 1 2 3 4 5 6 7
TEST	P 0 0 0 0 0 0 0 0
SENSE	P D D D D 0 1 0 0
WRITE (OUTPUT)	P D D D D D D 0 1
READ (INPUT)	P D D D D D D 1 0
READ BACKWARD	P D D D D 1 1 0 0
CONTROL	P D D D D D D 1 1
RESERVED FOR CHANNEL CONTROL	P X X X X 1 0 0 0

P = Odd parity bit
X is ignored by control units
D = Command detail bits (ignored by the channel)

Depending on the type of control unit, with data transfer functions, the detail bits specify such details as code translations, tape density, search controls, etc. or they may precondition the control unit for subsequent operation(s) in cases where the 8-bit command code cannot supply sufficient detail or where a mode of operation is to carry through several operations. (The latter should be avoided if possible.) The D bits are not interpreted by the channel.

Status Byte Format

STATUS BYTE FORMAT 0 1 2 3 4 5 6 7 P

Bit Position	Interpretation
P	Odd Parity for STATUS byte
0	Attention
1	STATUS Modifier
2	Control Unit End
3	Busy
4	Channel End
5	Device End
6	Unit Check
7	Unit Exception

The STATUS byte is transmitted to the channel in the following situations:

1) During the Initial Selection Sequence.

2) To present Channel End at the end of data transfer in some cases.

3) To present Device End and any associated conditions to the channel.

4) To present externally initiated STATUS to the channel.

Control and Maintenance Panel

The control/maintenance panel on the front of the CP-IOC chassis provides the necessary switches, indicators, and controls to operate the AN/UYK-15 and to provide maintenance personnel with an effective aid to general servicing. Table 6 lists the panel components and describes the function assigned to each.

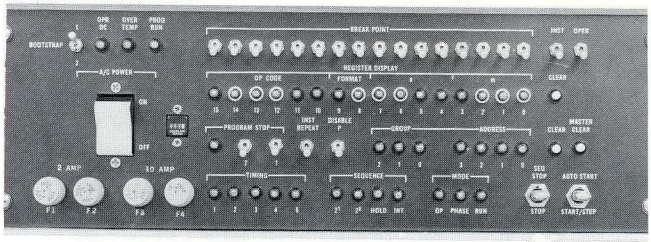


Figure 18. Operator/Maintenance Panel

TABLE 6. CONTROL/MAINTENANCE PANEL

Indicator/Switch	Function
A/C POWER ON-OFF switch	ON position applies the ac input power to the power supplies and blowers.
F1 and F2 Fuses	Fuses for blower input power protection.
F3 and F4 Fuses	Fuses for dc power supply input power protection.
Time Meter	Records the accumulated time that ac power is distributed to the CP, IOC, and memory power supplies.
PROG RUN indicator light	Lights when the computer is executing instructions in the Run Mode.
OVER TEMP indicator light	Lights when the temperature in any part of the cabinet is approaching unsafe temperature operating conditions.
OPR DC indicator light	Lights when all dc voltages are within operating tolerances.
BOOTSTRAP 1-2 switch	Used in conjunction with the operation code 40 (conditional jump) instruction to select the corresponding Bootstrap program.
BREAK POINT* 16 switches	Each switch sets either a one or a zero in the corresponding bit position of the Break Point Register.
INST switch (2-Position)	Enables or disables the instruction address "compare" used in conjunction with the Break Point Register.
OPER switch (2-Position)	Enables or disables the operand address "compare" used in conjunction with the Break Point Register.
REGISTER DISPLAY 0-15 indicator-switches	Set or display the contents of the register selected by the GROUP and ADDRESS indicator-switches, whenever the computer is not executing instructions in the Run Mode.

*Optional Feature

TABLE 6. CONTROL/MAINTENANCE PANEL (CONTINUED)

Indicator/Switch	Function
CLEAR switch	Pressing the switch clears the register selected by the GROUP and ADDRESS indicator-switches, whenever the computer is not executing instructions in the Run Mode.
PROG STOP 1 and 2 switches and indicator	Used in conjunction with the operation code 40 (Conditional Jump) instruction to stop the computer. The single indicator lights when either stop condition is satisfied.
INST REPEAT switch	The down position disables changing the contents of the U-register (current instruction).
DISABLE P switch	The down position disables advancing the P-register.
GROUP 0-2 and ADDRESS 0-3 indicator-switches	Selects according to Table 7, the register to be connected to the REGISTER DISPLAY indicator-switches. Pressing one GROUP switch sets that bit and clears any other GROUP bit that was set. Pressing an ADDRESS switch sets that particular bit without changing the others.
CLEAR switch	Pressing the switch clears the ADDRESS bits 0-3.
TIMING 1-5 indicators	Indicates the timing circuit that is enabled.
SEQUENCE 2^0, 2^1, HOLD, INT indicators	Indicates the sequence that is enabled. 2^0 and 2^1 indicate one of four sequences as follows (indicator lights on 1): 2^0 2^1 Sequence 0 0 Instruction 1 (I1) 0 1 Instruction 2 (I2) 1 0 Operand (OP) 1 1 Return Jump (RJP)
MODE switch-indicators	Select and indicate the computer mode of operation. Operating one MODE switch (when the computer is not executing instructions in the RUN Mode) selects that mode and clears any other mode that was selected.
OP Mode	Enables the computer to execute one instruction or one sequence according to the position of the SEQ STOP-STOP switch.
PHASE Mode	Enables the computer to execute both an early and a normal clock phase.
RUN Mode	Enables the computer to execute instructions at normal operating speed.

TABLE 6. CONTROL/MAINTENANCE PANEL (CONTINUED)

Indicator/Switch	Function
SEQ STOP-STOP 3-position switch	When the computer is executing instructions in the RUN mode, the STOP position causes the computer to stop at the completion of the current instruction. The SEQ STOP position enables the computer to execute one sequence when the OP mode is enabled. The inactive (center) position, enables the computer to operate in the selected mode.
AUTO START-START/STEP	START/STEP position initiates the execution of a phase, a sequence, an instruction or a program, according to the mode selected and the position of the SEQ STOP-STOP switch. The AUTO START position generates a Class I, priority 4 interrupt when power is applied to the computer.
MASTER CLEAR pushbutton	MASTER CLEAR, depressed when the computer is not executing instructions in the RUN mode, clears the computer logic circuits to an initial state.

TABLE 7. REGISTER SELECT ADDRESSING

GROUP Indicator Switches 2 1 0	ADDRESS Indicator Switches 3 2 1 0	Selected Register
0 0 1	Octal Value 00 – 17	General Register 0 through 17 respectively
	Binary Value	
1 0 0	X X 0 0	P-register
1 0 0	X X 0 1	Status register #1
1 0 0	X X 1 0*	Status register #2
1 0 0	X X 1 1*	Real-Time Clock
0 1 0	X X X X	U-Register

*Significant only when the computer contains this feature.

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.

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.

Symbols Used In Instructions

Symbol	Description
a	The a-designator from instruction words.
d	The deviation value in a local jump instruction.
R_a	The register designated by a.
m	The m-designator from instruction words.
R_m	The register designated by m.
Y	The operand or memory 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.

Operation Code 00

– RR Format – INSTRUCTION FAULT
Generate the Instruction Fault Interrupt.

– 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, leaving bits 15 through 8 unchanged. Address $Y = \frac{(R_m)}{2} + y$; bit 0 of R_m is the byte identifier.

Operation Code 01

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

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

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

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

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.

– 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.

Operation Code 03

(2) RR Format – UNARY-CONTROL
Perform the operation specified in Table II for the m-value.

– 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, $R_{a+1}, \ldots, R_{17}, R_0 \ldots R_m$. Address Y is equal to y.

(1) See Table I
(2) See 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	If (R_a) are positive, add bit 15 of R_{a+1} to (R_a) and store the result in R_a.
			
			If (R_a) are negative, subtract the complement of bit 15 of R_{a+1} from (R_a) and store the result in R_a.

--	3	-------	Not assigned
TCR	4	TWO'S COMPLEMENT, SINGLE	Perform the two's complement of (R_a) and store the result in R_a.
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
I ROR	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.
I RTR	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 an Executive Interrupt, set the Executive mode designator in the Status Register and store (P)+1 in R_a.
SSOR	1	STORE STATUS REGISTER #1	Store the contents of Status Register #1 in R_a.
SSTR	2	*STORE STATUS REGISTER #2	Store the contents of Status Register #2 in R_a.
SCR	3	*STORE RTC	Store the contents of the Real Time Clock Register in R_a.
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	Load (R_a) in 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 a RTC Interrupt when the contents of the Real Time Clock Register changes from all ones to all zeros.
DCR	11	*DISABLE RTC	Disable the Real Time Clock Register from advancing. The RTC Oscillator continues to operate.
LEM	12	*LOAD AND ENABLE MONITOR CLOCK	Load (Ra) in the Interrupt 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 equals zero.
---	13-17	--------	Not assigned

*Optional Instruction Operations.

Operation Code 04

(1) RR Format – UNARY-SHIFT (Optional Feature)
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
Load the selected byte from memory address Y in bits 7 through 0 of R_a, leaving bits 8 through 15 unchanged; and then increase (R_m) by 1.
Address $Y = \frac{(R_m)'}{2} + y$; bit 0 of R_m is the byte identifier.

Operation Code 05

SBR RR Format – SET BIT (Optional)
Set the bit in R_a specified by the m-value.

LXI RI Format, Type 2, – LOAD AND INDEX BY 1
Load the contents of memory address Y in R_a; and then increase (R_m) by 1.

– RK Format – Not assigned

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

Operation Code 06

ZBR RR Format – ZERO BIT (Optional) Clear Bit
Clear the bit in R_a specified by the m-value.

LDXI RI Format, Type 2 – LOAD DOUBLE AND INDEX BY 2
Load the contents of memory addresses Y and Y +1 in R_a and R_{a+1} respectively; and then increase (R_m) by 2.

– RK Format – Not assigned

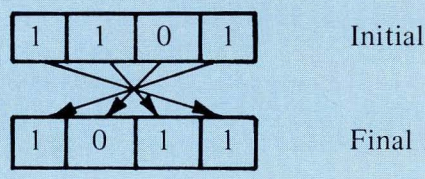
LDX RX Format – LOAD DOUBLE AND INDEX BY 2.
Load the contents of memory addresses Y and Y + 1 in R_a and R_{a+1} respectively; and then increase (R_m) by 2.

Operation Code 07

CBR RR Format – COMPARE BIT (Optional) Test Bit
Test the bit in R_a specified by the m-value and set the Condition Code.

(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 with the remainder in R_{a+1}.
RVR	1	REVERSE REGISTER	Change (R_a) to the reverse order according to the 4-bit example: 
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+1} are not equal and store the shift count in R_{a+2}.
---	4-17	-------	Not assigned.

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

– — RK Format – Not assigned

LP — RX Format – LOAD PSW
Load the contents of memory addresses Y and Y + 1 in the Program Address Register and Status Register #1, respectively. Y = (R_m) + y.

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.

– — 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.

BS — RX Format – BYTE STORE
Store in the selected byte of memory address Y bits 7-0 of (R_a). $Y = \frac{(R_m)}{2} + y$. Bit 0 of (R_m) is byte identifier.

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.

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.

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.

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.

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 0-5 of operand Y.

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, $R_{a+1}, \ldots, R_{17}, R_o, \ldots, R_m$. Y equals y.

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.

– 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.

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. Y = $\frac{(R_m)}{2}$ + y; bit 0 of (R_m) is byte identifier.

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.

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.

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

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.

SDXI RI Format, Type 2 – STORE DOUBLE AND INDEX BY 2
Store (R_a) and (R_{a+1}) at memory addresses Y and Y + 1 respectively; then increase (R_m) by 2.

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.

SDX RX Format – STORE DOUBLE AND INDEX BY 2
Store (R_a) and R_{a+1}) at memory addresses Y and Y + 1 respectively; and then increase (R_m) by 2.

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. n is the value in bits 5-0 of R_m.

SZI RI Format, Type 2 – STORE ZEROS
Clear 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. n is the value in bits 5-0 of Y.

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

Operation Code 20

SUR RR Format – SUBTRACT
Subtract (R_m) from (R_a) and store the result in R_a; then set the Condition Code.

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

SUK RK Format – SUBTRACT
Subtract operand Y from (R_a) and store the result in R_a; then set the Condition Code.

SU RX Format – SUBTRACT
Subtract the contents of memory address Y from (R_a) and store the result in R_a; then set the Condition Code.

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}; then set the Condition Code.

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}; then set the Condition Code.

– 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}; then set the Condition Code.

Operation Code 22

AR RR Format – ADD
Add (R_m) to (R_a) and store the result in R_a; then set the Condition Code.

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

AK RK Format – ADD
Add operand Y to (R_a) and store the result in R_a; and then set the Condition Code.

A RX Format – ADD
Add the contents of memory address Y to (R_a) and store the result in R_a; and then set the Condition Code.

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}; then set the Condition Code.

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}; then set the Condition Code.

– 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}; then set the Condition Code.

Operation Code 24

CR — RR Format – COMPARE
Arithmetically compare (R_a) to (R_m), and set the Condition Code.

CI — RI Format, Type 2 – COMPARE
Arithmetically compare (R_a) to the contents of memory address Y, and set the Condition Code.

CK — RK Format – COMPARE
Arithmetically compare (R_a) to operand Y, and set the Condition Code.

C — RX Format – COMPARE
Arithmetically compare (R_a) to the contents of memory address Y, and set the Condition Code.

Operation Code 25

CDR — RR Format – COMPARE DOUBLE
Arithmetically compare the double length (R_a, R_{a+1}) to the double length (R_m, R_{m+1}) and set the Condition Code.

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 and set the Condition Code.

– — 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 and set the Condition Code.

Operation Code 26

MR — RR Format – MULTIPLY
Multiply (R_m) by (R_a) and store the double length result in R_a, R_{a+1}; and then set the Condition Code.

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

MK — RK Format – MULTIPLY
Multiply operand Y by (R_a) and store the double length result in R_a, R_{a+1}; and then set the Condition Code.

M — RK Format – MULTIPLY
Multiply the contents of memory address Y by (R_a) and store the double length result in R_a, R_{a+1}; then set the Condition Code.

Operation Code 27

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; then set the Condition Code.

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; then set the Condition Code.

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; then set the Condition Code.

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; then set the Condition Code.

Operation Code 30

ANDR — RR Format – AND
Perform the logical AND of (R_a) and (R_m), and store the result in R_a (clear bits in R_a corresponding to zeros in R_m).

ANDI — RI Format, Type 2 – AND
Form the logical AND of (R_a) and the contents of memory address Y, and store the contents of memory address Y, and store the result in R_a (clear bits in R_a corresponding to zeros in the contents of address Y).

ANDK — RK Format – AND
Form the logical AND of (R_a) and operand Y, and store the result in R_a (clear bits in R_a corresponding to zeros in operand Y).

AND — RX Format – AND
Form the logical AND of (R_a) and the contents of memory address Y, and store the result in R_a (clear bits in R_a corresponding to zeros in the contents of address Y).

Operation Code 31

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

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

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

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

Operation Code 32

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

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

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

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

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.

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.

MSK RK Format – MASKED SUBSTITUTE
For each bit set in (R_{a+1}) the value of the corresponding bit in operand Y shall be transferred to the corresponding bit in R_a. For each bit not set in (R_{a+1}) the corresponding bit in R_a is unaltered.

MS RX Format – MASKED SUBSTITUTE
For each bit set in (R_{a+1}) the value of the corresponding bit in the contents of memory address Y shall be transferred to the corresponding bit in R_a. For each bit not set in (R_{a+1}) the corresponding bit in R_a is unaltered.

Operation Code 34

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}) and set the Condition Code.

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}) and set the Condition Code.

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}) and set the Condition Code.

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}) and set the Condition Code.

Operation Code 35

IOCR RR Format – I/O COMMAND
Enable the Command Request line to the IOC specified by a.

BFI RI Format, Type 2 – BIASED FETCH
Transfer bit 15 of the contents of memory address Y to the Condition Code and then set the two most significant bits at that memory location leaving the remaining bits unchanged.

REX RK Format – EXECUTE REMOTE
Execute the instruction stored at memory address Y; do not change (P) when reading this instruction. Then continue with the next sequential instruction.

BF RX Format – BIASED FETCH
Transfer bit 15 of the contents of memory address Y to the Condition Code and then set the two most significant bits at that memory location leaving the remaining bits unchanged.

Operation Code 36

– RR Format – Not assigned

– RR Format – Not assigned

– RK Format – Not assigned

PTC RX Format – PROCESSOR/PERIPHERAL COMMAND
Perform the I/O function on the processor/peripheral channel as specified for the a-designator bit configuration in Table IV.

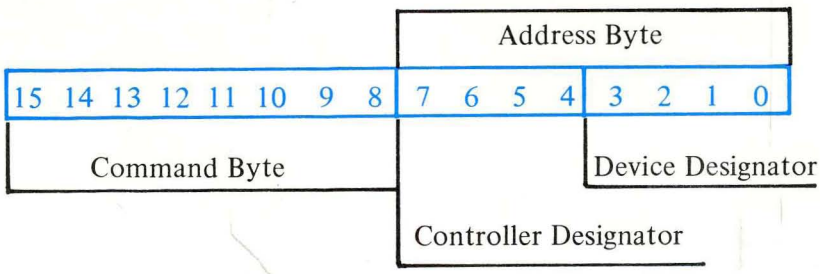


Figure I. Format of Y for Command Out Function

TABLE IV. PROCESSOR/PERIPHERAL COMMAND INSTRUCTIONS, a-DESIGNATOR INTERPRETATION

a-Designator Bits				Description
7	6	5	4	
0	1	1	1	PERFORM SELECTIVE RESET
1	0	1	1	PERFORM INTERFACE DISCONNECT
1	1	0	1	INITIATE SELECTOR MODE
1	1	1	0	INITIATE MULTIPLEXER MODE
1	1	1	1	CLEAR COMMAND ACTIVE
0	0	0	0	TRANSFER IN – Read the information on the data bus, store the information in the selected byte at memory address Y, increase (R_m) by 1, and perform one of the following: (1) If the information is status, enable SERVICE OUT. *(2) If the information is data and the condition code indicates "not zero", enable SERVICE OUT. *(3) If the information is data and the condition code indicates "zero" (end of transfer), enable COMMAND OUT and clear Selector mode.
0	1	0	0	TRANSFER OUT – Load the selected byte from memory address Y on the data bus, increase (R_m) by 1, read the information, and perform one of the following: (1) If the information is status, enable SERVICE OUT. *(2) If the information is data and the condition code indicates "not zero", enable SERVICE OUT. *(3) If the information is data and the condition code indicates "zero" (end of transfer), enable a COMMAND OUT and clear Selector mode.
1	0	0	0	COMMAND OUT, Multiplexer – Load the address byte (bits 7-0) of Y as specified in Figure I on the data bus, and enable ADDRESS OUT for the processor-peripheral channel specified by bits 4 and 5 of the a-designator. Then in response to ADDRESS IN, load the command byte (bits 15-8) of Y on the data bus, and enable COMMAND OUT. Leave the channel in the Multiplexer mode.
1	1	0	0	COMMAND OUT, Selector – Load the address byte (bits 7-0) of Y as specified in Figure I on the data bus, and enable ADDRESS OUT for the processor-peripheral channel specified by bits 4 and 5 of the a-designators. Then in response to ADDRESS IN, load the command byte (bits 15-8) of Y on the data bus, and enable COMMAND OUT. Leave the channel in the Selector mode.

*The condition code value is a result of the last arithmetic operation performed.

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.

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

(R_a) = X coordinate; radix point between 2^{15} and 2^{14} bit positions

(R_{a+1}) = Y coordinate; radix point between 2^{15} and 2^{14} bit positions

(R_{a+2}) = W; Bit 2^{14} = 1, represents 90^o in trigonometric mode. Each successive bit equal to one represents an angle one-half as large as its adjoining higher order bit.

W in hyperbolic mode is a constant with radix point between 2^{14} and 2^{13} bit positions.

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, load (R_m) in P, jump to the instruction located at the address specified in (R_m). If a specified Stop, or a Stop Key condition is met, load (R_m) in P and stop the computer.

(2) If the specified condition is not met, execute the next instruction.

LJ RI Format, Type 1 – LOCAL JUMP
Load Y in P (jump to the instruction located at memory address Y).Y = $(P)_i \pm d$

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

(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	Stop	
JKSR	JKS	JKS	12	Stop if program stop key 1 is selected	
JKSR	JKS	JKS	13	Stop if program stop key 2 is selected	
---	---	---	14-15	Not assigned	
JCAR	JCA	JCA	16	Processor-Peripheral Channel active	
---	---	---	17	Not assigned	

(1) If the specified condition is met, load Y in P (jump to the instruction located at the address specified by operand Y). If a specified Stop, or a Stop Key condition is met, load the operand Y in P and stop the computer.

(2) If the specified condition is not met, execute the next instruction.

(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, load (Y) in P (jump to the instruction located at the address specified by the contents of memory address Y). If a specified Stop, or a Stop Key condition is met, load (Y) in P and stop the computer.

(2) If the specified condition is not met, execute the next instruction.

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 and load (R_m) in P (jump to the instruction located at the address stored in R_m).

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

– RI Format – Not assigned

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 load operand Y in P (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 load (Y) in P (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 load (R_m) in P (jump to the instruction located at the address stored in R_m).

(1) See Table V

– RI Format – Not assigned

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

JLR RX Format – JUMP AND LINK REGISTER
Store (P)+2 in R_a, and load (Y) in P (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 load Y+1 in P (jump to the instruction located at memory address Y+1. Y = (P)±d.

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

Operation Code 44

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

(1) If (R_a) equals zero, load (R_m) in P (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 the Condition Code is zero, load Y in P (jump to the instruction located at memory address Y). Y = (P)±d.

(2) If the Condition Code is not zero, 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, load operand Y in P (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, load (Y) in P (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 ≠ 0
Test (R_a) and perform one of the following:

(1) If (R_a) does not equal zero, load (R_m) in P (jump to the instruction located at the address stored in 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 the Condition Code is a binary 01 or 11, load Y in P (jump to the instruction located at memory address Y). Y = (P)±d.

(2) If the Condition Code is not a binary 01 or 11, execute the next instruction.

JNZ RK Format – JUMP REGISTER ≠ 0
Test (R_a) and perform one of the following:

(1) If (R_a) does not equal zero, load Y in P (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 ≠ 0
Test (R_a) and perform one of the following:

(1) If (R_a) does not equal zero, load (Y) in P (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, load (R_m) in P (jump to the instruction located at the address stored in 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 the Condition Code is a binary 00 or 01, load Y in P (jump to the instruction located at memory address Y). Y = (P)±d.

(2) If the Condition Code is not a binary 00 or 01, 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, load Y in P (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, load (Y) in P (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, load (R_m) in P (jump to the instruction located at the address stored in 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 the Condition Code is a binary 11, load Y in P (jump to the instruction located at memory address Y). Y = (P)±d.

(2) If the Condition Code is not a binary 11, 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, load Y in P (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, load (Y) in P (jump to the instruction located at 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 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)
ACR	0	*Master Clear – deactivate all chains and data buffers —disable all external interrupts and buffer monitors
	1	Not assigned
	2	*Enable all chains
	3	*Disable all chains
	4	*Enable all external interrupts
	5	*Disable all external interrupts
	6	*Enable all external interrupt monitors (accept external interrupt data without interrupting the CP)
	7	*Disable all external interrupt monitors
		Operations effecting only the channel specified by the a-designator (chaining)
CCR	10	Master clear
	11	Not assigned
	12	Enable the channel chain
	13	Disable the channel chain
	14	Enable external interrupt
	15	Disable external interrupt
	16	Enable external interrupt monitor
	17	Disable external interrupt monitor

*The a-designator must be zero.

IO RX Format – INITIATE TRANSFER (Chaining)
Load the contents of memory addresses Y and Y+1 in control memory locations 0 and 1 respectively, and activate the transfer mode specified for the a-value in Table VII. Disable the chain on the channel until the buffer terminates and then enable it. (The m-designator is not used and Y must be an even number.)

TABLE VII. INITIATE TRANSFER INSTRUCTION a-VALUE

a-Value	Transfer Mode
0	Input data
1	Output data
2	External function
3	External function with force
4-17	Not assigned

Operation Code 71

ICK RK Format – INITIATE CHAIN (Command)
Transfer Y to the control memory chain Address Pointer for the channel specified by the a-designator and enable the chain for that channel. (The m-designator must equal 2.)

LCMK RK Format – LOAD CONTROL MEMORY (Chaining)
Load operand Y in the control memory location specified by the m-designator. (m-values 0-3 select locations 0-3 respectively; m-values 4-17 and all 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 by the combined am-designator as defined in Figure II.

LCM RX Format – LOAD CONTROL MEMORY (Chaining)
Load the contents of memory address Y in the control memory location specified by the m-designator (m-values 0-3 select locations 0-3 respectively; m-values 4-17 and all a-designator values are not interpreted.

Operation Code 72

RCM RX Format – STORE CONTROL MEMORY (Command)
Store in memory address Y the contents of control memory location specified by the combined am-designator, as defined in Figure II.

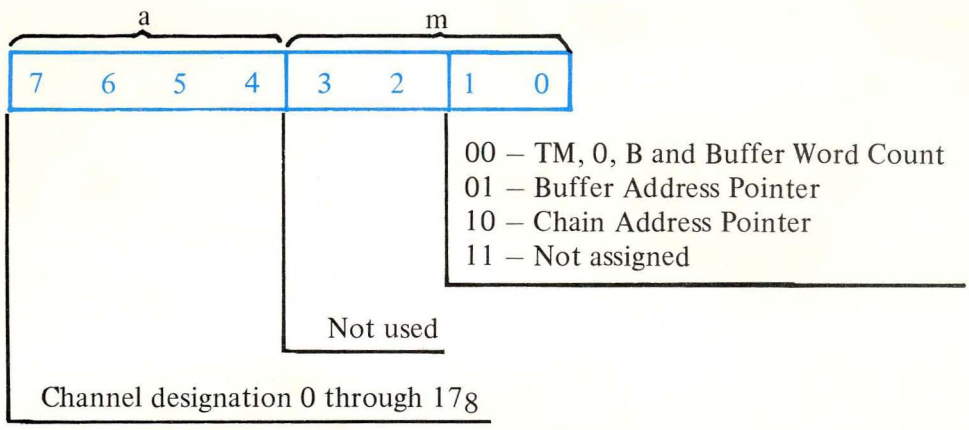


Figure II. Control Memory Addresses (Command)

SCM RX Format – STORE CONTROL MEMORY (Chaining)
Store in memory address Y the contents of control memory location specified by the m-designator (m-values 0-3 select control memory locations 0-3 respectively; m-values 4-17 and all a-designator values are not interpreted).

Operation Code 73

RR Format — HALT/INTERRUPT (Chaining)
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 2-17$_8$ are not interpreted.

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

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

The m-designator and a-values 2-17$_8$ are not interpreted.

SPERRY UNIVAC
DEFENSE SYSTEMS

PX 7917
10.72