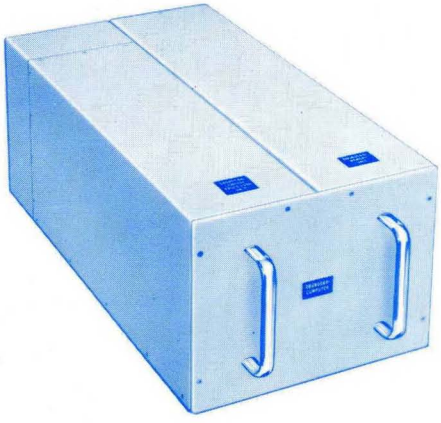
PROGRAMMERS' REFERENCE MANUAL



ALERT

GENERAL PURPOSE DIGITAL COMPUTER

Honeywell *Military Products Group* AERONAUTICAL DIVISION — ST. PETERSBURG, FLORIDA

Honeywell Aero Document R-ED 24290

HONEYWELL ALERT

PROGRAMMERS' REFERENCE MANUAL

Honeywell

AERONAUTICAL DIVISION — ST. PETERSBURG, FLORIDA

THIRD EDITION
First Printing
June 1966

Questions and comments regarding this manual
should be addressed to

Honeywell Aeronautical Division
Marketing Department
13350 U.S. Highway 19
St. Petersburg, Florida

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

Section 1

INTRODUCTION

Honeywell's ALERT computer is a high speed, general purpose digital computer capable of processing large quantities of complex data in a real-time environment. Major features of the computer are:

- Internal high speed storage expandable from 4,096 to 32,768 words.
- 24-bit word length.
- Repertoire of 89 instructions, some of which provide for conditional program branches.
- Single address instructions with address modification capability via six index registers and/or indirect addressing.
- Priority interrupt handling with eight standard levels expandable to 24.
- Character handling.
- Solid-state construction.
- Power failure protection.
- Complies with MIL-E-5400, class 2, and MIL-I-26600.

The ALERT computer, incorporating a binary, parallel, single address organization, is capable of performing logical operations, data moving, and binary arithmetic in both fractional and integer modes.

A variety of ALERT memory unit modules are available for use with the central processor. Either destructible readout (DRO) toroidal core or non-destructible readout (NDRO) microbiax memory units may be used. Memory module size varies according to the number of words and type of element.

Section 2

GENERAL ORGANIZATION

The Honeywell ALERT Computer is a stored program computer designed for versatility in airborne digital computing. ALERT is especially applicable to real-time systems employing missile guidance, navigation, energy management and electronic intelligence (ELINT). The ALERT Computer emphasizes rapid communication and flexibility with external devices and randomly accessed internal storage.

Internal storage consists of a standard 4,096 word memory unit expandable to 32,768 words. Each word may be interpreted as a single 24-bit word, or as four 6-bit characters individually addressed.

The arithmetic and logical operations are performed in two's complement arithmetic. Control registers in the arithmetic section are comprised of six index registers, the accumulator (A-register), and the auxiliary arithmetic B-register. In most cases, the result of an arithmetic operation appears in the 24-bit A-register. However, the B-register is also available for addition and subtraction, when so required.

Computer operation is controlled by a stored program capable of self-modification. Most instructions contain a function code (6 bits), an operand address (15 bits), and an address variant (3 bits). The address variant provides for indexing, character handling, and indirect addressing. The next sequential instruction may be skipped or the program sequence may be altered as determined by the contents of the A-register or the referenced memory location.

A communication path between the ALERT Computer and its associated external equipment is established by a sequence of request and response signals between external equipment and computer. These signals may originate in either the computer or the external device. Once the link has been established, the computer can transmit data with timing under control of the external device. Digital data is transmitted in parallel to and from the computer with two independent data channels. The direct input/output channel transmits or receives data under program control. The direct memory access channel transmits and receives data independent of computer instruction execution.

Associated with the ALERT Computer is a family of modular airborne, miniaturized power supplies. These are consistent with ALERT packaging and they furnish dc power to the computer with various combinations of memory units. Sensing circuitry is provided which detects power failure and causes a sub-sequence. Sufficient energy is retained to store the status of the machine. Power cycling, on and off, is effected in such a way as to protect the contents of memory.

Section 3

OPERATION

The operation of the ALERT computer is outlined under four major topics: arithmetic, addressing, input/output, interrupts.

ARITHMETIC

Arithmetic in the ALERT computer is performed in two's-complement notation (see appendix F) with a complete set of shift, comparison, arithmetic and logical operations. The two arithmetic registers available to the programmer are the main accumulator (A-Register) and the auxillary accumulator (B-Register). The basic function of each is outlined below.

The A-REGISTER contributes one operand to the arithmetic and logical operations in addition to holding the results of these operations when so instructed. The A register holds the high order product of a multiplication or the quotient of a divide. The 24-bit contents may be shifted left or right, in either a circular or arithmetic mode.

The B-REGISTER is a 24-bit auxillary accumulator available for addition and subtraction operations. It holds the low order product at the end of a multiplication and the remainder at the end of a divide. The contents of the B register can be shifted only as the low order 24-bits of a double length quantity in the combined A and B registers. Nevertheless, the double length contents of AB may be shifted left or right in either a circular or arithmetic mode, with or without sign protection in B.

The TRANSFER REGISTER (TR) is a 24-bit register which is loaded from memory and is used as one of the operand registers in the execution of an arithmetic command. During a peripheral output command, this register holds the information that is placed on the output bus. The TR is not available to the programmer.

The Arithmetic Overflow Indicator is turned on if any of the following occur:

Addition or subtraction, resulting in a sum or difference which cannot be contained within the 24-bit register.

Multiplication of Y by Y where Y is the largest negative number that can be represented in a 24-bit word (40000000).

This product cannot be contained within the A and B registers. The arithmetic overflow indicator may be tested by the SKN instruction or cleared by the SKN, STS, or RSS instructions.

The Arithmetic Carry Indicator is set or reset (cleared) at the completion of all add or subtract operations which use the A and B registers and leaves the result in the respective register. The indicator is turned on if any of the following occur:

Addition in A or B forces a carry into the sign bit of the respective register.

Subtraction in A or B forces a borrow from the sign bit of the respective register.

If neither of these conditions occur, the carry indicator is cleared. It may also be cleared by the STS or RSS instructions and tested by the SKN instruction.

The Division Overcapacity Indicator is set when the absolute value of the numerator is equal to or greater than the absolute value of the denominator. When this occurs, all divide operations are inhibited until this indicator is cleared by an SKN, STS or RSS instruction.

ADDRESSING

The ALERT computer is capable of direct addressing, multiple indirect addressing, indexed addressing, or character addressing as indicated by the address variant in the instruction word. All instructions which are not indexable and/or not indirectable are so designated in the discussion of the instruction repertoire. Character addressing applies only to the three character instructions and each character identification is determined not only by a fixed location in memory, but also by the position of the character in that 24-bit word. The basic features of ALERT addressing are outlined as follows:

Direct Addressing pertains to the instruction whose original variant is zero. In this case, the fifteen bit address field contains the address of the operand.

Indirect Addressing pertains to the instruction whose original variant is seven. In this case, the address field is used to reference memory, and the low order fifteen bits of the word thus found is used as the address of the operand. Since in an indirect address, the address variant has the same meaning as in the original instruction word (except for character instructions, see below), indirect addressing may be continued indefinitely and indexing performed, if desired, at the final level of indirect addressing. That is, an address may be indirect any number of times, but it may be indexed only once. All interrupts will be blocked until the effective address is found.

Indexed Addressing pertains to the instruction whose variant is ultimately not zero or seven. These instructions may be divided into two mutually exclusive classes. First, the instructions which are used to manipulate the contents of the index registers are termed "not indexable", and second, the instructions which use the index registers to create an effective address are termed "indexable". The index registers are fifteen bits in length. The contents of these registers are not considered signed quantities; however, in creating a new address, a two's complement number may be used with correct results.

Character Addressing pertains only to the character instructions LCH, CSK, and SCH. These instructions utilize the two least significant bits of the variant, in the original instruction, to specify the character number as 1, 2, 3, or 0 where "0" represents the fourth character. The most significant bit of this variant is used to specify whether the address is direct or indirect. If indirect addressing is specified in the original instruction word, the variant in the indirect address will be interpreted in the normal manner as defined in Indirect Addressing above. As a result, indexing is permissible through indirect addressing with the character instructions.

The principal registers common to the addressing section of the ALERT computer are:

Sequence register

Operand address register

Address variant register

B box adder

Index registers

The SEQUENCE REGISTER (S) holds the instruction address and sequences the execution of the stored program. With its fifteen bits, it is capable of addressing 32, 768 words of memory. The contents of this register can be stored with the Branch and Return (BAR) instruction.

The OPERAND ADDRESS register (LR) is fifteen bits in length and normally holds the fifteen bit address of the operand in memory. It is capable of addressing up to 32, 768 operands in memory. In the case of the AIX instruction, it is used to augment the specified index register. This LR register is not available to the programmer.

The ADDRESS VARIANT register serves to decode the address variant V as the instruction is received from memory. As a result, the operand interpretation is determined as either direct, indexed or indirect.

The B BOX ADDER is fifteen bits in length and is used to increment either the specified index register or the operand address register. It is not available to the programmer.

The INDEX REGISTERS designated X1 through X6, are each fifteen bits in length. These registers store the quantities used for address modification. They may also be used as program counters.

INPUT-OUTPUT

Information into and out of the ALERT Computer may be transmitted along the direct input-output channel and/or the direct memory access channel.

The direct input-output (see Figure 3-1) channel provides an interface between the ALERT Central Processor and external devices. The DIO is under program control for all operations. Eight instructions are used to interrogate, select, control the device, and command data transfer into and out of the central processor. These eight instructions are categorized below and defined more specifically in the instruction repertoire.

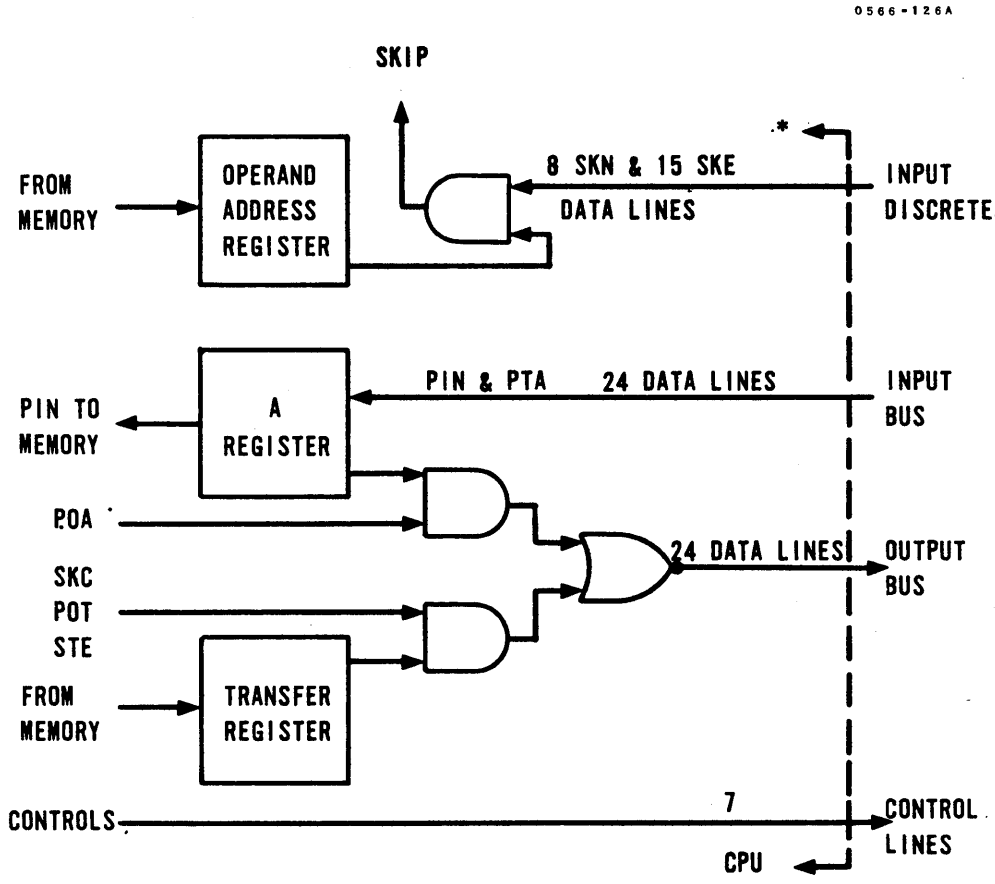


FIGURE 3-1. ALERT DIRECT I/O CHANNEL

Input Discretes

SKN - Skip if internal signal not set

SKE - Skip if external signal not set

Output Discretes

STE - Set external point

Selection, Interrogation and Control

SKC - Control and Skip

Data Transfer

POA - Peripheral output from A and skip

POT - Peripheral output from Memory and skip

PTA - Peripheral input to A and skip

PIN - Peripheral input to Memory and skip.

All input devices share a common input bus. These devices are selected and connected to this bus by the SKC instruction. The immediate fifteen bit address of this instruction may be employed in any combination of control and address to issue inquiries to the device or command it to perform certain functions. For example, the most common combination is to use the nine low order bits for addressing and the six high order bits for control. Upon execution of the SKC instruction, an SKC strobe is generated and used to ask the selected external device if it is ready. The sequence of events is as follows:

Computer connects device

Computer sends inquiry

Peripheral device sends acknowledge

Computer sends response/acknowledge signal.

The device must acknowledge the inquiry within the SKC strobe time (Figure 3-2) to be accepted by the computer. Upon acceptance, the computer generates a response/acknowledge signal which is sent to the selected external device. This two-way communication between the computer and the external equipment insures that both units are ready for information transfer. If the external device is busy, it will not generate a response within the SKC strobe time and next instruction will be skipped (refer to instruction repertoire).

For input data transfers, a PIN or PTA instruction is executed. These commands cause another strobe signal to be generated and the following sequence of events is executed for each word that is transferred.

Computer initiates input inquiry

Peripheral device sends acknowledge that data is ready for transmission

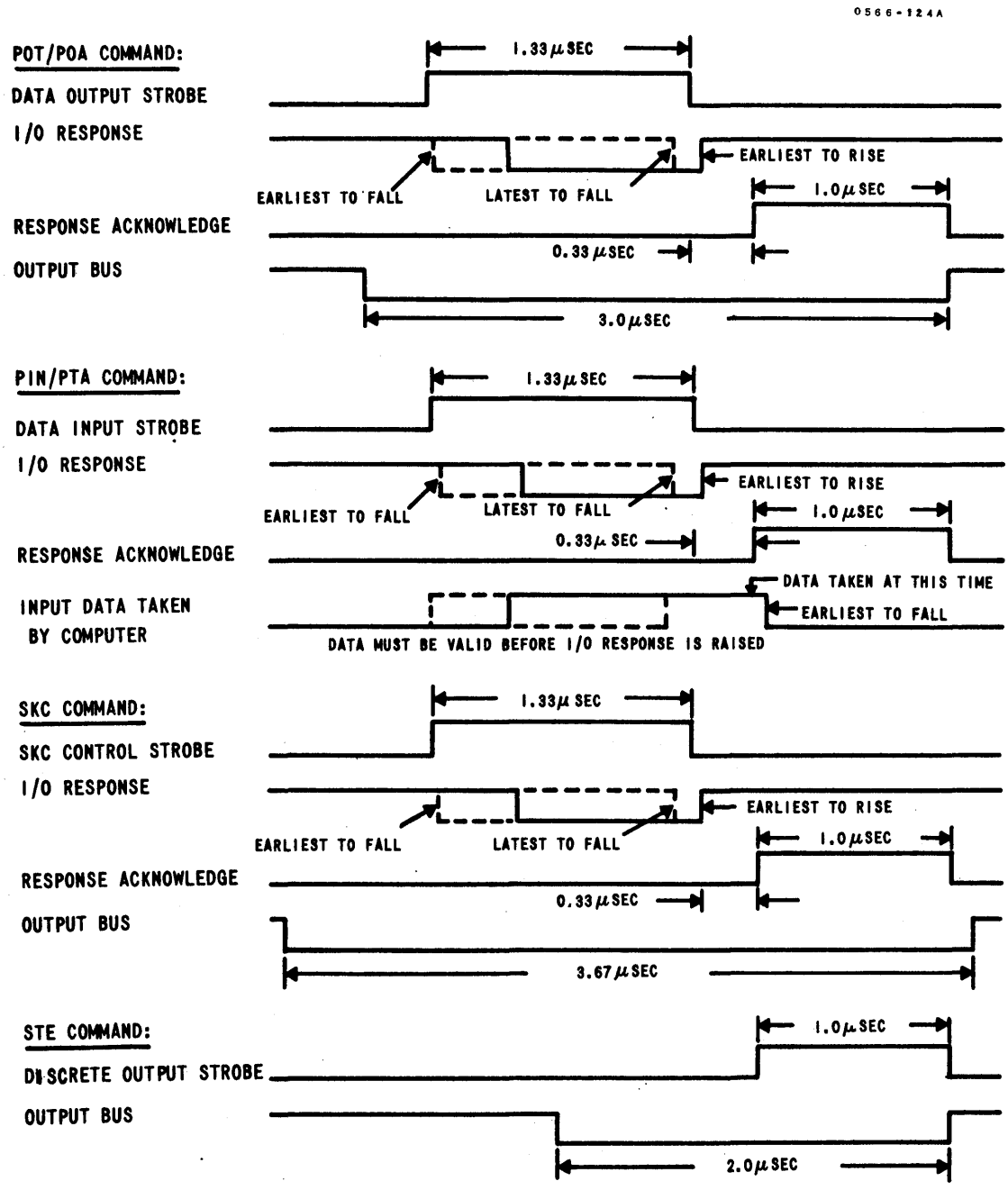


FIGURE 3-2. DIRECT INPUT/OUTPUT TIMING

Computer issues response/acknowledge signal

Peripheral device places data on 24-bit input bus

Computer transfers data to specified location.

All output devices share a common output bus and are selected and connected to this bus by the SKC command. The execution of this instruction is identical to the sequence used for input devices. Data is transferred to a device by executing the POA or POT commands. The basic sequence of events for an output transfer is similar to that specified for input transfers. An SKC is issued to select the device, recognition action takes place, the POT or POA command is given and its strobe causes the external device to acknowledge that a transfer has occurred.

The external device selected by the SKC command remains connected to the input (or output) bus until the next input (or output) SKC instruction is executed. Program branch decisions are based on external and internal conditions. The discrete external signals are tested by the SKE command and the discrete internal signals are tested by the SKN command. Six of the internal discretes (sense switches) are located at the operator's console panel.

Discrete outputs may be generated by executing an STE command. A particular external point is selected by bit position in the address field of the STE command and a one microsecond pulse is sent to the external point (see Figure 3-2). Fifteen external points can be set without external decoding. With decoding, as many as 32, 767 points may be serviced.

The direct memory access (see Figure 3-3) channel is a separate input-output channel with direct communication to or from memory entirely under external control. The computer relinquishes the memory for the next memory cycle upon an associated (read or write) request by the input-output devices connected to the DMA channel. Upon initiation, the DMA may hold its read or write requests for as many memory cycles as the I/O requires, and of course, the I/O may intermix the read or write requests in any combination necessary to satisfy system requirements. During the DMA memory cycle the computer "pauses" in the instruction it was executing at the time the DMA requested memory. Once the DMA removes its read or write request, the computer continues its instruction execution in a normal manner.

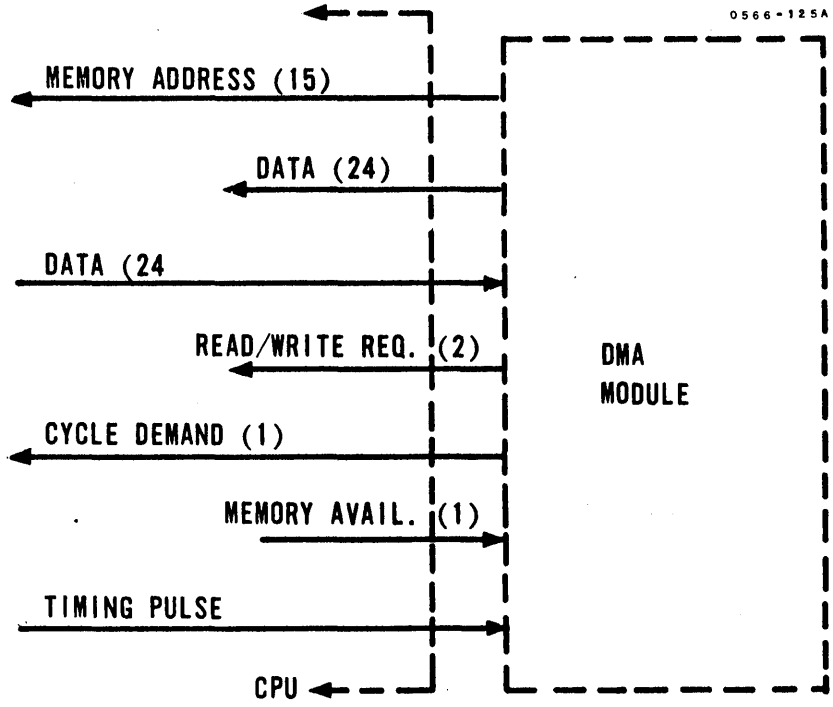


FIGURE 3-3. DIRECT MEMORY ACCESS CHANNEL

The DMA channel provides one 24-bit input bus and one 24-bit output bus for transmission of data to or from the computer at the associated request of the external device. The fifteen lines of the address bus provide the memory address of where the data is to be written or read. These allow full random addressing of up to 32, 768 words of memory.

BOOTSTRAP INPUT

The standard computer control unit provides a bootstrap capability to load paper tapes and manual console entries into sequential memory locations. In the bootstrap mode, the computer memory is addressed by the sequence register (S) instead of the operand address register (LR). Any information placed on the direct input bus is loaded by the PIN instruction (520XXXXX) into successive locations beginning with one plus the initial contents of the sequence register.

Beginning with the first start character on paper tape, every fourth non-skip character generates a sequence to pack the four characters into a 24-bit word (from left to right) and transfer the word to the memory location defined by the sequence register. The "stop" character (Figure 3-4)

stops the bootstrap after the character has been read. This should be the fourth character in the last word; otherwise, the last one, two, or three characters including the stop character will not be transferred.

Bootstrap tapes must be prepared in the format of Figure 3-4.

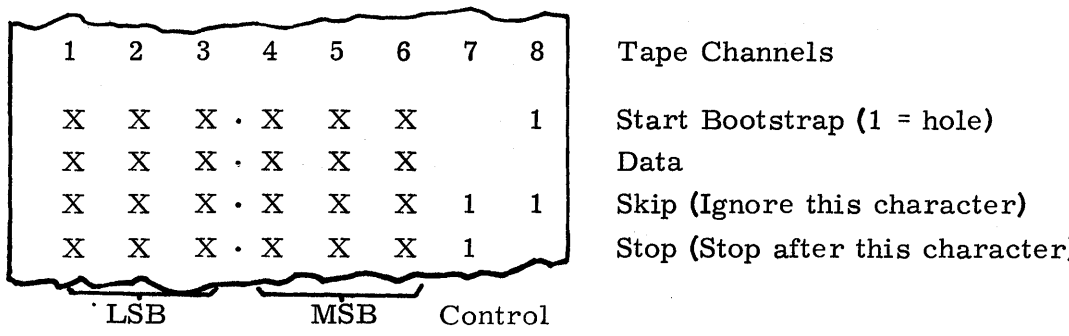


FIGURE 3-4. BOOTSTRAP TAPE FORMAT

INTERRUPTS

Interrupt conditions within and without the computer will cause the next instruction to be taken out of sequence from one of a group of fixed locations. These locations are termed "interrupt entrance addresses" as defined in Table 3-1.

The interrupts are arranged in priority sequence (see Table 3-1). In case of multiple interrupts occurring simultaneously, the highest priority (lowest priority number) interrupt is serviced. Of those not serviced, the internal interrupts are reset, but the external interrupts are preserved in the interrupt register. The basic interrupt channel configuration is illustrated in Figure 3-5.

TABLE 3-1. INTERRUPT DESIGNATIONS

Interrupt	Priority	Entrance Address	Is the Interrupt Blockable?
Power Failure	1	00037	No
Power Recovery	2	00036_8	No
Control Error	3	00035	No
Trapped Order T34	4	00034	No
Trapped Order T33	5	00033	No
Trapped Order T32	6	00032	No
Trapped Order T31	7	00031	No
Trapped Order T30	8	00030	No
Standard Ext. Line No. 23	9	00000	Yes
Standard Ext. Line No. 22	10	00001	Yes
Standard Ext. Line No. 21	11	00002	Yes
Standard Ext. Line No. 20	12	00003	Yes
Standard Ext. Line No. 19	13	00004	Yes
Standard Ext. Line No. 18	14	00005	Yes
Standard Ext. Line No. 17	15	00006	Yes
Standard Ext. Line No. 16	16	00007	Yes
Optional Ext. Nos. (15-8)	17-24	00010-17	Yes
Optional Ext. Nos. (7-0)	25-32	00020-27	Yes

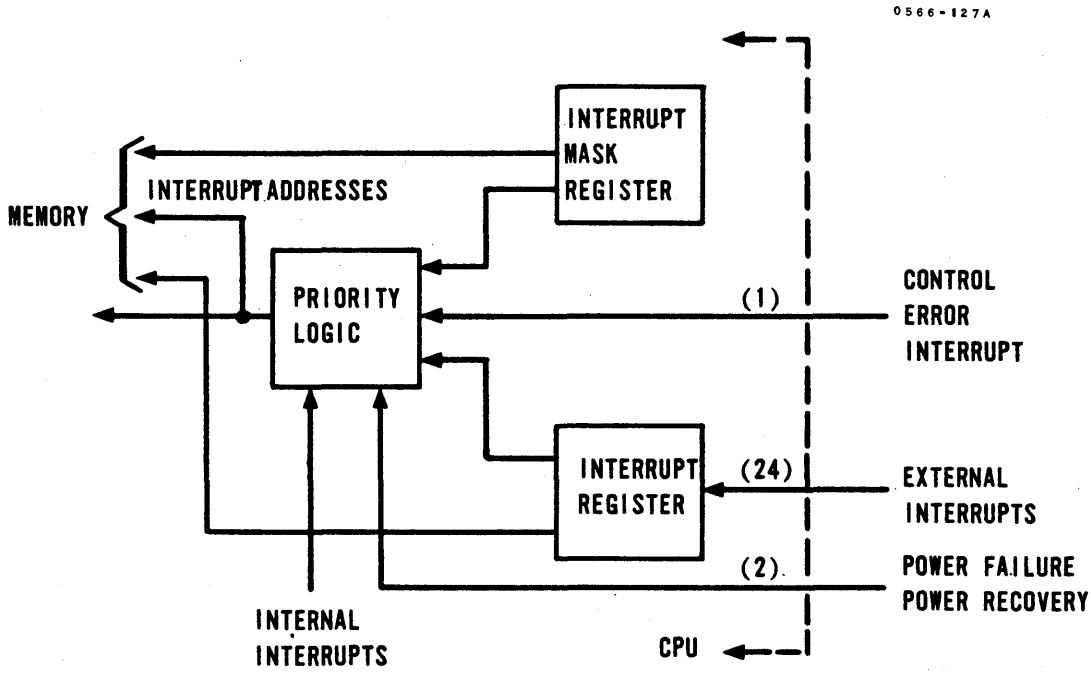


FIGURE 3-5. ALERT INTERRUPT CHANNEL

At the moment of interrupt subsequence, an interrupt block is automatically set to lock out all external interrupts. This block remains set until cleared by the program with an SRB or RSS instruction. The former status of this interrupt block is preserved in the interrupt block history register.

When an interrupt occurs, the instruction in execution is completed, and the sequence register at the moment of subsequence contains the location of the "next normal instruction". Consider the following cases:

If the instruction just completed did not jump, the "next normal instruction" is the next instruction in the current sequence.

If the instruction just completed did jump, the "next normal instruction" is the location to which the jump was made.

If the instruction, being executed, is a trapped order, only the most significant 6 bits of the instruction are interpreted to cause the subsequence. The "next normal instruction" is the location of the trapped order, plus one.

In all cases, the sequence register at the moment of subsequence contains the address that will be stored by a BAR instruction located at the interrupt entrance address.

Internal interrupts have absolute priority and cannot be blocked. The four types included in the ALERT computer are power failure, power recovery, control error*, and trapped orders.

Power Failure - Following the power failure interrupt, the computer is given three milliseconds in which to store (at the programmer's discretion) the status of principal registers and counters. Hence, the power failure interrupt is a warning signal from the power unit that primary power is failing. The instruction currently under execution is completed, and a subsequence is made to location 00037 (octal). The sequence register, at the time of interrupt, will be set to the address of the next normal instruction.

Power Recovery - Upon receiving an initializing pulse from the power supply, the computer automatically subsequences to location 00036 (octal). The instruction at this entrance address should transfer control to the power recovery program where principal registers and counters are restored and the latest instruction sequence is resumed. A minimum of three milliseconds is guaranteed between successive power failure and power recovery interrupts.

Control Error* - This unblockable interrupt is activated only from a source external to the computer. Its employment will therefore depend upon the external emergency conditions common to the particular system configuration.

Trapped Orders - The trapped order causes a subsequence to a fixed memory location associated with the particular command. This location should contain an instruction which stores the program counter (sequence register) and initiates a jump to an associated subroutine. By using trapped orders, operations such as floating point addition and subtraction may be performed with considerably less hardware. The sequence register at the time of subsequence will be set to the address of the trapped order, plus one.

---

* Although control errors are activated externally, they are classified as internal interrupts because they are unblockable and not part of interrupt register.

The external interrupt system provides eight lines from the outside world, each of which when activated, causes a subsequence to its own fixed location (Table 3-1), from whence the program may take action on that particular interrupt. The usage of these lines may be very generalized, however. This interrupt system, which is expandable to 24 has automatic assignment of separate "entrance addresses" for each external line, according to a fixed priority, with the ability to selectively block, or mask, individual lines. The external lines are collectively called the interrupt register, with priority according to bit position (bit 23 highest). The interrupt mask register has a bit for bit correspondence with the interrupt register. Only the lines in the interrupt register for which there is a corresponding "1" bit in the Mask Register can cause a subsequence. The others are disabled (blocked). The contents of the Interrupt Mask Register may be set or exchanged with the contents of a memory location under program control through the use of the respective LIM and XML instructions. Operation shall be as follows:

> If, at the end of an instruction, one or more interrupt lines are active and their corresponding mask bits are ones, a particular address corresponding to the line of highest priority is selected and a subsequence is made to that address. Any other interrupts are stored in the interrupt register until later. When the subsequence takes place, the interrupt block is automatically set to lock out all further external interrupts, until it is cleared by the SRB or RSS instruction. Other external interrupts will then be serviced as soon as the block is cleared. This interrupt block may also be set by the SRB or RSS instruction whenever an external interrupt block is required.

The interrupt logic is designed to accept signal levels from external devices. This is consistent with the type of signal furnished by the peripheral units which may be attached to the computer via the direct input-output bus. These units possess their own local interrupt storages, which they present to the computer in the form of steady signal levels. These signals will cause subsequences as described above. The signals are not cleared automatically when they are honored with a subsequence. However, they may be cleared by program execution of an SKC instruction which tests for interrupt status and, at the same time, clears the interrupt being tested. One extra microsecond is required for each interrupt subsequence. If the trapped order is used, this extra time is absorbed in the stated execution time.

To increase coverage of interrupt lines, more than one external device may be serviced by the same line by external buffering. A scanning subroutine using the SKC instruction may be used to determine interrupt priority when more than one external device is sharing the same interrupt line.

Section 4

COMPUTER CONSOLE CONTROL

Both the standard computer control panel (Figure 4-1) and the optional operator's control panel (Figure 4-2) provide a visible report of the internal status of the computer. During the debugging or operational run of the object program, several modes of computer utilization are selectable at these units. It is not necessary, however, to monitor the consoles during normal operation. Basic operation of the standard computer control panel is outlined below.

To initiate a program:

a. Place RUN-STOP switch at STOP.

b. Place MODE switch at INST. LOCK.

c. Load Instruction Word Switches with 650XXXXX where XXXXX is the starting address of the program.

d. Depress CCU CLEAR switch.

e. Depress CPU CLEAR switch.

f. Depress INSTR. ADV switch.

g. Place MODE switch at PROG. RUN position.

h. Depress CCU CLEAR switch.

i. Depress CPU CLEAR switch.

j. Depress INSTR ADV for single instruction advance or place the RUN-STOP switch at the RUN position for high speed operation.

To step by instruction:

a. Place RUN-STOP switch at STOP position.

b. Depress INSTR ADV for each instruction execution.

A6604-166

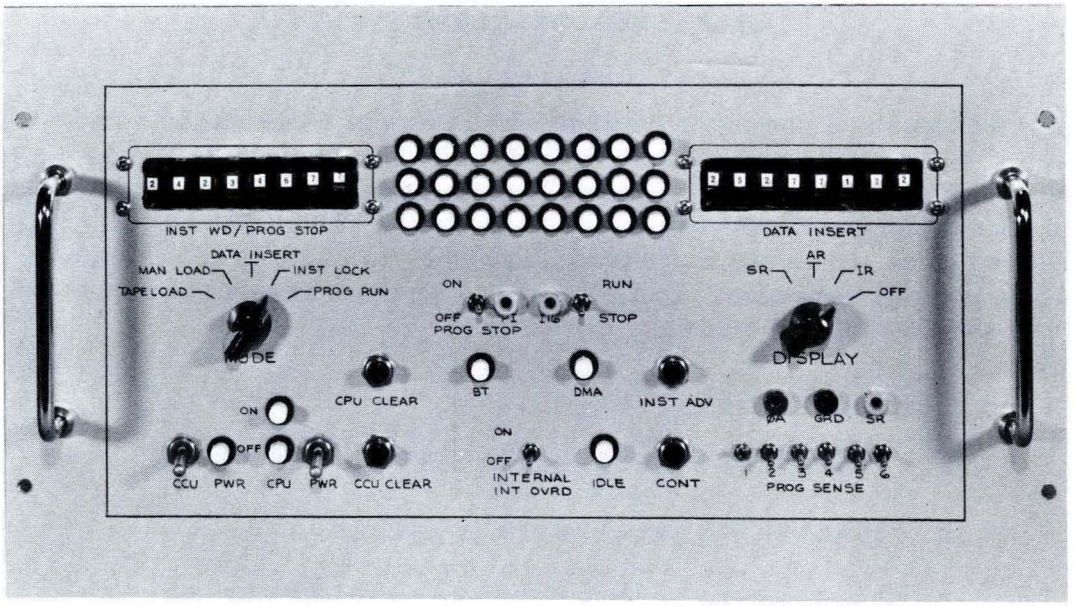


FIGURE 4-1. STANDARD COMPUTER CONTROL PANEL

A6601-175

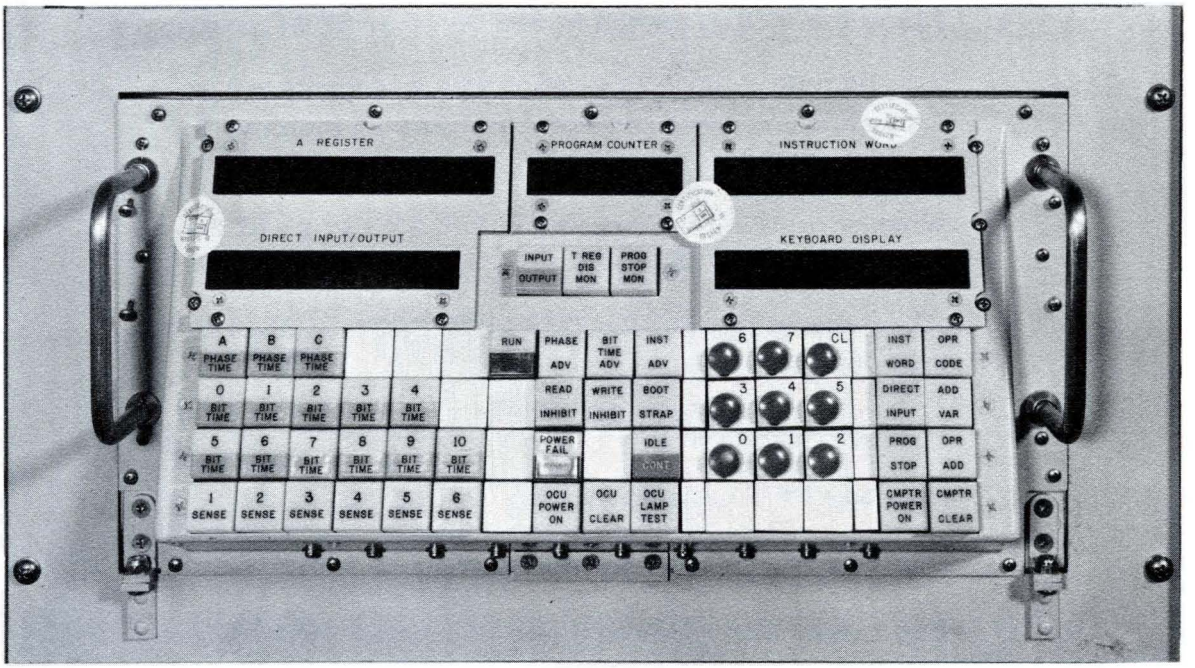


FIGURE 4-2. OPTIONAL OPERATOR'S CONTROL PANEL

To inspect or change one memory location only:

a. Place the RUN-STOP switch at STOP position.

b. Place the MODE switch at the DATA INSERT position.

c. Set INST WD switches to 400XXXXX/520XXXXX where XXXXX represents the octal address of memory to inspect/change.

d. When inspecting: Place the DISPLAY switch at the A-register (AR) position.

e. When changing: Set the DATA INSERT switches to the new contents.

f. Depress CCU CLEAR switch.

g. Depress CPU CLEAR switch.

h. Depress INSTR ADV switch.

i. For inspection: The contents of location XXXXX are displayed in the register display lamps.

To inspect or manually load a program into sequential memory locations:

a. Place the RUN-STOP switch in STOP position.

b. Place the MODE switch in MAN LOAD position.

c. Set 650XXXXX into the INST WD switches, where XXXXX is one less than the first location to be loaded or inspected.

d. Depress CCU CLEAR switch.

e. Depress CPU CLEAR switch.

f. Depress INSTR ADV switch.

g. To inspect: Set DISPLAY switch to AR.

Set 400XXXXX into the INST WD switch.

Depress INSTR ADV and observe the contents of XXXXX in the register display lamps. Repeat this step for each location to be inspected.

h. To load: Set 520XXXXX into the INST WD switches.

Set data into DATA INSERT switches and depress INSTR ADV. Repeat this step for each new data word to be loaded.

To load a constant throughout memory:

a. Place RUN-STOP switch in the STOP position.

b. Place MODE switch at MAN LOAD.

c. Depress CCU CLEAR switch.

d. Depress CPU CLEAR switch.

e. Set 52077777 into INST WD switches.

f. Set constant data into DATA INSERT switches.

b. Place RUN-STOP switch in the RUN position.

To set the PROG STOP address:

a. Set five-digit octal address into the INST WD/PROG STOP switches.

b. Place PROG STOP switch in the ON position.

To read paper tape via bootstrap control:

a. Load paper tape reader.

b. Place RUN-STOP switch to STOP position.

c. Place MODE switch to INST LOCK.

d. Set 650XXXXX* into INST WD switches.

---

* XXXXX is one less than the first location to be loaded.

e. Depress CCU CLEAR switch.

f. Depress CPU CLEAR switch.

g. Depress INSTR ADV switch.

h. Place MODE switch to TAPE LOAD position.

i. Set 520XXXXX into INST WD switches.

j. Depress CCU CLEAR switch.

k. Depress CPU CLEAR switch.

l. Place RUN-STOP switch in RUN position.

Auxiliary Monitor Functions and Controls:

a. The IDLE indicator monitors an Idle Halt condition.

b. The Continue (CONT) switch advances the program out of an idle halt condition.

c. The program stop (PROG STOP) switch allows the program to stop at a predetermined location. This location is set in the lower five switches of the INST WD/PROG STOP register.

d. The DMA indicator monitors the status of the Direct Memory Access channel. When the light is on, the DMA channel is active.

e. The Bit Time (BT) indicator monitors bit time status. The light will be turned on if no bit times are present.

f. The INTERNAL INT ORVD (Internal Interrupt Override) switch inhibits all internal interrupts when in the ON position.

g. Program sense switches are discrete switches for program use in conjunction with the SKN order.

Section 5

INSTRUCTION REPERTOIRE

This section lists the repertoire of instructions used with the ALERT Computer. Instructions are grouped by functional categories. Lists of instructions in functional, alphabetical, and numerical order are given in Appendixes A, B, and C respectively.

The following statements apply to the instruction descriptions:

Octal notation is used throughout.

Parentheses are used to denote "contents of". For example, (B) denotes the contents of the B register.

All instruction times (given in microseconds) apply to the Microbiax type memory and include accessing of the instruction. Alternate instruction times are tabulated in Appendixes B and C for one microsecond core and two microsecond core respectively.

Indexing and indirect addressing apply to all instructions unless otherwise stated. One additional memory (read) cycle is required for each level of indirect addressing and indexing used to form the effective address.

The letter "Y" denotes the "effective address" which applies to the instruction after all modifications (indexing and indirect addressing) have been completed.

The letter "S" denotes the sequence register which contains the location of the next instruction to be executed. If an interrupt occurs, the sequence register is altered only by the instruction in the interrupt entrance register.

The letters "A" and "B" are used to denote the A and B registers respectively. Furthermore, the 48-bit combination is represented by "AB". The instruction address is denoted by X (or XXXXX) and the address variant by V.

The interrupt system can interrupt the program at the end of any instruction except as noted.

Common usage of these instructions is also included where clarification is necessary; however, no attempt is made to indicate more sophisticated use.

CONTROL INSTRUCTIONS

BAR · BRANCH AND RETURN (04 V XXXXX) TIME: 6

The contents of S are stored in memory location Y, and control is transferred to memory location Y + 1. Example:

Sequence Counter	Instruction	(3000)
(Before Execution)		650 XXXXX
2000	BAR 3000	000 02001
3001	etc. etc.	

NOTE

This instruction stores the sequence counter into Y and then clears the remaining nine bits of (Y) to zero.

EXC · EXECUTE (01 V XXXXX) TIME: 1

The instruction at memory location Y is transferred to the internal instruction register and executed without changing the contents of the sequence register. The next instruction in the current sequence will be executed in a normal manner unless location Y contains a jump or skip instruction which would otherwise alter the program sequence.

HLT · HALT (IDLE) (00 7 XXXXX) TIME: 2+Δ

This instruction halts the program sequence.

NOTES

1. Peripheral transfers in progress through the direct memory access channel are not halted, but proceed to completion.

2. When the machine is halted, interrupts will still be honored, and the machine restarted by the subsequence. The sequence counter at the moment of interrupt is set at the address of the halt instruction.

3. The machine may be restarted at the next instruction in sequence by manually depressing the CONTINUE button at the operator' s console.

4. The halt order should not be executed unless the computer is connected to the computer test unit or some other device capable of restarting the computer as per Note 3.

5. Δ denotes idle time.

The address is not interpreted by the computer and may be used at the programmer's discretion for instruction identification or program data. Indirect and indexed addressing do not apply to this instruction.

PAS · NO OPERATION (00 0 XXXXX) TIME: 2

Proceed to the next instruction in the current sequence. Indirect and indexed addressing do not apply to this instruction. The address is not interpreted by the computer and may be used at the programmer' s discretion for program data.

JAN/JAP · JUMP ON (A) NEGATIVE/POSITIVE (05 V XXXXX)/ (15 V XXXXX) TIME: 2

This instruction enters a new program address in S, if the content of A is negative/positive. If the jump condition is satisfied, location Y becomes the address of the next instruction and the beginning of a new program sequence. Otherwise, the next instruction in the current sequence is executed in a normal manner.

JAZ/JNZ · JUMP ON (A) ZERO/NOT ZERO (14 V XXXXX)/
(21 V XXXXX) TIME: 2

This instruction enters a new program address in S, if the content of A is zero/not zero. If the jump condition is satisfied, location Y becomes the address of the next instruction and the beginning of a new program sequence. Otherwise, the next instruction in the current sequence is executed in a normal manner.

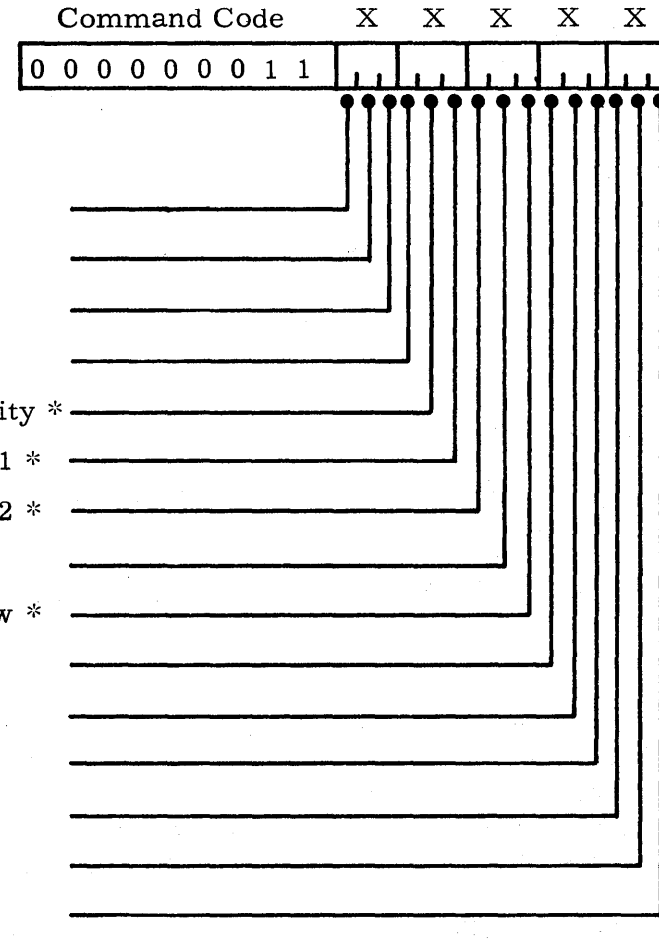
JMP · JUMP UNCONDITIONALLY (65 V XXXXX) TIME: 2

This instruction enters a new program address in S unconditionally. The effective address Y becomes the address of the next instruction and the beginning of a new program sequence.

SKN · SKIP IF INDICATOR NOT SET (00 3 XXXXX) TIME: 3

This instruction tests the condition of the specific indicators designated by the immediate bit configuration of X. Each one-bit of X uniquely defines an indicator to be tested. If all designated points are not set, the next instruction in the current sequence is skipped. Otherwise (if any designated point is set), the next instruction is executed in a normal manner. Indexing and indirect addressing do not apply to this instruction.

The SKN test indicators are defined as follows:



SMZ · SKIP IF MEMORY IS ZERO (03 V XXXXX) TIME: 3

This instruction compares the content of Y with zero. The content of Y is not altered; however, if it is equal to zero, the next instruction in the current sequence is skipped. Otherwise the next instruction is executed in a normal manner.

---

* Cleared when tested by the SKN instruction with $X_{14} = 1$. Program indicators 1 and 2 may be used as flags at the programmer's discretion since they can be set by the RSS instruction and tested and/or reset by the SKN instruction.

** Bits 12 and 13 of the SKN instruction may be used to supplement the basic 15 external test points common to the SKE instruction.

SMN · SKIP IF MEMORY NOT ZERO (73 V XXXXX) TIME: 3

This instruction compares the content of Y with zero. The content of Y is not altered; however, if it is not equal to zero, the next instruction in the current sequence is skipped. If the content of Y is zero, the next instruction is executed in a normal manner.

SKM · SKIP IF (A) = (Y) (02 V XXXXX) TIME: 4

This instruction compares the content of location Y with the content of the A register. It does not alter either (A) or (Y). If the contents of A and Y are equal, the next instruction in the current sequence is skipped. Otherwise, the next instruction is executed in a normal manner.

SAN · SKIP IF (A) $\neq$ (Y) (72 V XXXXX) TIME: 4

This instruction compares the content of location Y with the content of the A register. It does not alter either (A) or (Y). If the contents of A and Y are not equal, the next instruction in the current sequence is skipped. Otherwise, the next instruction is executed in a normal manner.

SML · SKIP IF (Y) $\leqq$ (A) (22 V XXXXX) TIME: 4

This instruction compares the content of location Y with the content of the A register. It does not alter either (A) or (Y). If the content of Y is algebraically less than or equal to the content of the A register, the next instruction in the current sequence is skipped. Otherwise, the next instruction is executed in a normal manner.

SMG · SKIP IF (Y) > (A) (23 V XXXXX) TIME: 4

This instruction compares the content of location Y with the content of the A register. It does not alter either (A) or (Y). If the content of Y is algebraically greater than the content of A, the next instruction in the current sequence is skipped. Otherwise, the next instruction is executed in a normal manner.

ARITHMETIC INSTRUCTIONS

ADD · ADD TO (A) (34 V XXXXX) TIME: 2

The contents of location Y are added to the contents of the A register and the result appears in A. The carry indicator is set if the addition forces a carry into the sign bit on the A register. Otherwise the carry indicator is reset to zero. Overflow occurs and the overflow indicator is set if both numbers are of the same sign, and the sign of the result is opposite. In this case, the signed result is incorrect and the overflow indicator will remain set until cleared by an SKN or STS instruction. Example:

	Location		Contents		
	2000		20000000		
Location	Instruction	A	B	Carry	Overflow
(Before Execution)		20000000	00000000	0	0
2200	ADD 2000	40000000	00000000	1	1
2201	ADD 2000	60000000	00000000	0	1
2202	ADD 2000	00000000	00000000	1	1

ABD · ADD TO (B) (64 V XXXXX) TIME: 3

The contents of location Y are added to the contents of the B register and the result appears in B. The carry indicator is set if the addition forces a carry into the sign bit of the B register. Otherwise, the carry indicator is reset to zero. Overflow occurs and the overflow indicator is set if both numbers are of the same sign, and the sign of the result is opposite. In this case, the signed result is incorrect and the overflow indicator will remain set until cleared by an SKN or STS instruction. Example:

	Location		Contents		
	2000		20000000		
Location	Instruction	A	B	Carry	Overflow
(Before Execution)		00000000	20000000	0	0
2300	ABD 2000	00000000	40000000	1	1
2301	ABD 2000	00000000	60000000	0	1
2302	ABD 2000	00000000	00000000	1	1

ADC · ADD TO (A) WITH CARRY (54 V XXXXX) TIME: 2

This instruction is used to perform double-precision addition. The lower half of the numbers are added first in the B register with the ABD instruction. The carry is automatically retained in the carry indicator as a one or a zero. The two upper halves are then added using this instruction which executes the same as the ADD instruction except that the content of the carry indicator is also added. Carry and overflow indicators are set accordingly at the end of the operation. Example:

Location	Contents
3000	12345670
3001	20200000

Location	Instruction	A	B	Carry	Overflow
(Before Execution)		00000000	20000000	0	0
2400	ABD 3001	00000000	40200000	1	1
2401	ADC 3000	12345671	40200000	0	1

SUB · SUBTRACT FROM (A) (35 V XXXXX) TIME: 2

The contents of location Y are subtracted from the contents of the A register and the result appears in A. The carry (borrow) indicator is set if the subtraction requires a borrow from the sign bit of the A register. On the other hand, the carry indicator is reset to zero if a borrow is not required. Overflow occurs and the overflow indicator is set if both numbers are of different signs and the sign of the result does not agree with the original sign of A. In this case, the signed difference is incorrect and the overflow indicator will remain set until cleared by an SKN or STS instruction. Example:

Location	Contents
2000	60000000

Location	Instruction	A	B	Carry	Overflow
(Before Execution)		20000000	00000000	0	0
3000	SUB 2000	40000000	00000000	0	1
3001	SUB 2000	60000000	00000000	1	1

SBB · SUBTRACT FROM (B) (66 V XXXXX) TIME: 3

The contents of location Y are subtracted from the contents of the B register and the result appears in B. The carry (borrow) indicator is set if the subtraction requires a borrow from the sign bit of the B register. On the other hand, the carry indicator is reset to zero if such a borrow is not required. Overflow occurs and the overflow indicator is set if both numbers are of different signs and the sign of the result does not agree with the original sign of B. In this case, the signed difference is incorrect and the overflow indicator will remain set until cleared by an SKN or STS instruction. Example:

Location	Contents
2000	60000000

Location	Instruction	A	B	Carry	Overflow
(Before Execution)		00000000	20000000	0	0
2100	SBB 2000	00000000	40000000	0	1
2101	SBB 2000	00000000	60000000	1	1

SUC · SUBTRACT FROM (A) WITH CARRY (55 V XXXXX) TIME: 2

This instruction is used to perform double-precision subtraction. The lower half of the numbers are subtracted first in the B register with the SBB instruction. The "borrow" is automatically retained in the carry indicator as a one or zero. The two upper halves are then subtracted using this instruction which executes the same as the SUB instruction except that the content of the carry indicator is also subtracted. Carry and overflow indicators are set accordingly at the end of the operation. Example:

Location	Contents
3000	00000000
3001	20000000

Location	Instruction	A	B	Carry	Overflow
(Before Execution)		01010101	00000000	0	0
2100	SBB 3001	01010101	60000000	1	0
2101	SUC 3000	01010100	60000000	0	0

ADM · ADD (A) TO MEMORY (24 V XXXXX) TIME: 7

This instruction adds the contents of the A register to the previous contents of location Y. The result appears in location Y and the contents of A remain undisturbed. Carry and overflow indicators are not altered by this instruction.

TLY · TALLY (25 V XXXXX) TIME: 7

The contents of location Y are increased by one and the result is restored in Y. The (A) and the carry and overflow indicators are not altered by this instruction.

MPY · MULTIPLY (76 V XXXXX) TIME: 12

The contents of the A register are multiplied by the contents of Y. The product is located in the A and B registers; the most significant portion is in A. The sign of the product is retained in the sign bit of both A and B; hence, the least significant portion of the product is in the remaining bits of B to the right of its sign bit. The original contents of B do not affect the operation and are destroyed. If the contents of both the multiplier and multiplicand have the value 40000000, overflow will occur and the overflow indicator will be set. The carry indicator is not affected by this instruction. Example:

Location	Instruction	A	B	(3000)
(Before Execution)		00000003	--------	00000003
1000	MPY 3000	00000000	00000011	00000003
(Before Execution)		20000000	--------	20000000
2000	MPY 3000	10000000	00000000	20000000
(Before Execution)		00000003	--------	20000000
2100	MPY 3000	00000001	20000000	20000000

DIV · DIVIDE (77 V XXXXX) TIME: 30

Treating the combined contents of the A and B registers as a double length number and excluding the sign bit of B, the contents of A and B are divided by the contents of location Y. The quotient appears in the A register, the remainder in B. The sign of the remainder is identical to the sign of the original content of A unless the remainder is zero and the original content of A was negative.

Overflow will occur and the division overcapacity indicator will be set if:

$$1 \leq \frac{(AB)}{D} < -1,$$

where D is the double length divisor whose most significant half is equal to the contents of location Y and whose least significant half is zero. In this case, the contents of A and B will remain undisturbed with one exception - the sign of B will be set equal to the sign of A.

NOTE

The divide operation is always inhibited if the division overcapacity indicator is set. It is therefore the programmer's responsibility to test and clear this indicator with an SKN instruction if there has been any possibility of division overcapacity prior to this instruction.

Example:

Location	Instruction	A	B	(2000)	Div Overcap
(Before Execution)		00000000	00000003	00000002	0
3000	DIV 2000	00000001	00000001	00000002	0
(Before Execution)		00000004	00000000	00000002	0
3100	DIV 2000	00000004	00000000	00000002	1

ABS · SET (A) POSITIVE (00 2 00747) TIME: 2

This instruction replaces the current contents of A with its absolute value. Addressing is not permitted with this instruction. The contents of B are not altered. Carry and overflow indicators are not altered by this instruction.

CMP · COMPLEMENT (A) (00 2 00727) TIME: 2

This instruction replaces the current contents of A with its two's complement. Addressing is not permitted with this instruction. The contents of B are not altered. Carry and overflow indicators are not altered by this instruction.

CAB · CLEAR A AND B (00 2 01727) TIME: 2

This instruction clears the contents of both A and B to zero. Addressing is not permitted.

CLA · CLEAR A (00 2 00700) TIME: 2

This instruction sets the A register to zero. Addressing is not permitted.

INDEXING INSTRUCTIONS

The function of each indexing instruction, unless otherwise indicated, is not defined when the effective V is equal to zero.

AIX · AUGMENT INDEX IMMEDIATE (37 V XXXXX) TIME: 3

Augment the contents of index register V by the immediate value of X. The address variant V is used to select the index register for this instruction and augment its contents by the immediate value of X. The address variant V is used to specify the index register for this instruction and is not available for indexing or indirect addressing.

AUX · AUGMENT INDEX BY (Y) (36 V XXXXX) TIME: 3

Augment the contents of the effective index register by the contents of location Y. Indexing does not apply to this instruction, since V is used to specify the selected index register.

IJX · INCREMENT AND JUMP (20 V XXXXX) TIME: 2

Compare the contents of the selected index register with zero. If the content is nonzero, execute a jump in program sequence to location Y. Increase the content of the index register by one.

If the content of the index register is zero, proceed to the next instruction in the current sequence. Do not alter the contents of the index register.

Indexing does not apply to this instruction since V is used to specify the selected index register.

DJX · DECREMENT AND JUMP (07 V XXXXX) TIME: 2

Compare the content of the selected index register with zero. If the content is nonzero, execute a jump in program sequence to location Y. Decrease the content of the index register by one.

If the content of the index register is zero, proceed to the next instruction in the current sequence. Do not alter the contents of the index register.

Indexing does not apply to this instruction since V is used to specify the selected index register.

JIX · JUMP ON INDEX NONZERO (17 V XXXXX) TIME: 2

If the contents of the effective index register is nonzero, execute a jump in program sequence to location Y. If the content of this index register is zero, proceed to the next instruction in a normal manner. The contents of the index register are not altered by this instruction. Indexing does not apply to this instruction since V is used to specify the selected index register.

SXI · SKIP IF INDEX EQUALS Y (06 V XXXXX) TIME: 3

If the content of the effective index register is equal to the immediate value of Y, the next instruction in the current sequence is skipped. Otherwise, the next instruction is executed in a normal manner.

Indexing does not apply to this instruction since V is used to specify the selected index register.

SKX · SKIP IF INDEX EQUALS (Y) (16 V XXXXX) TIME: 4

If the content of the effective index register is equal to the content of location Y, the next instruction in the current sequence is skipped. Otherwise, the next instruction is executed in a normal manner. Indexing is not permitted with this instruction since V is used to specify the selected index register.

LDX · LOAD INDEX (26 V XXXXX) TIME: 2

Transmit the lower-order 15 bits of location Y to the effective index register. The higher-order 9 bits of Y are ignored by this instruction. Indexing is not permitted since V is used to specify the selected index register.

STX · STORE INDEX (27 V XXXXX) TIME: 6

Store a 24-bit quantity, whose lower 15 bits correspond to the content of the effective index register and whose higher order 9 bits are zero, at location Y. Indexing is not permitted since V is used to specify the selected index register. If the effective V is zero, this instruction will store zeros into location Y. (See STZ.)

TAX · TRANSFER (A) TO INDEX (00 2 00V07) TIME: 2

Transmit the lower-order 15 bits of A to the index register specified by V. Indexing and indirect addressing are not permitted. The contents of A are not altered by this instruction.

TXA · TRANSFER INDEX TO A (00 2 0070V) TIME: 2

Transmit a 24-bit quantity, whose lower order 15 bits correspond to the content of the index register specified in V and whose higher order 9 bits are zero, to the A register. The contents of the index register are not altered and A will be cleared if V is zero. Indexing and indirect addressing do not apply to this instruction.

INPUT/OUTPUT INSTRUCTIONS

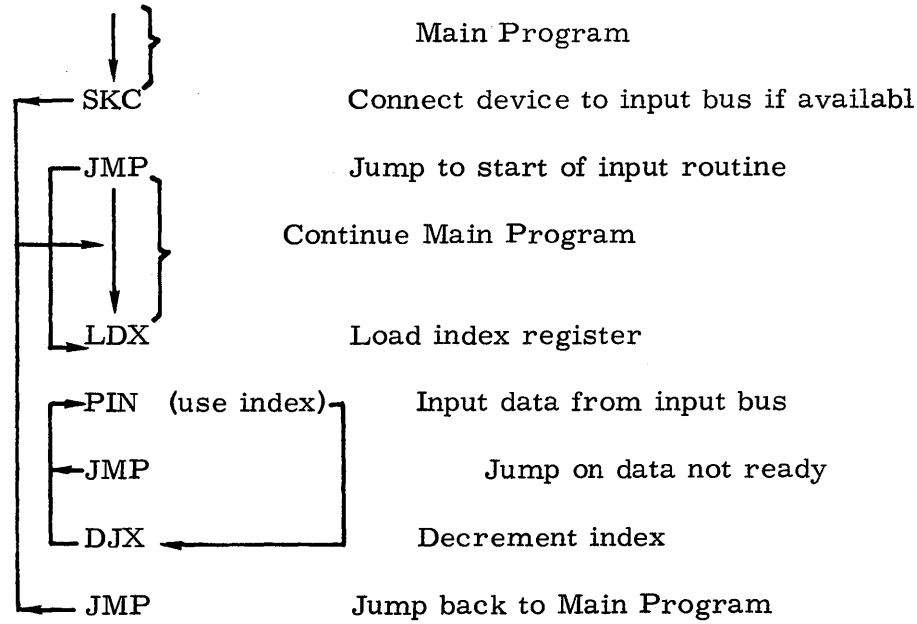
PTA · PERIPHERAL INTO A (13 V XXXXX) TIME: 4

This instruction tests the condition of the input bus. If the input bus is ready to transmit data, the 24-bit word on the input bus replaces the previous contents of the A register and the next instruction in the current sequence is skipped. If the input bus is not ready, the contents of A will be destroyed and the next instruction in the current sequence is executed in a normal manner.

PIN · PERIPHERAL INTO Y (52 V XXXXX) TIME: 8/4

This instruction tests the condition of the input bus. If the input bus is ready to transmit data, the 24-bit word on the input bus replaces the previous content of location Y and the next instruction in the current sequence is skipped. If the input bus is not ready, no data is transmitted and the next instruction in the current sequence is executed in a normal manner. The contents of A are not altered by this instruction.

Execution time is eight microseconds if the input bus is ready to transmit data. Otherwise, the execution time is four microseconds. Example:



Maximum Transfer Rate

9 μsec PIN
JMP
2 μsec DJX
11 μsec

Maximum Search Rate

5 μsec PIN
2 μsec JMP
DJX
7 μsec

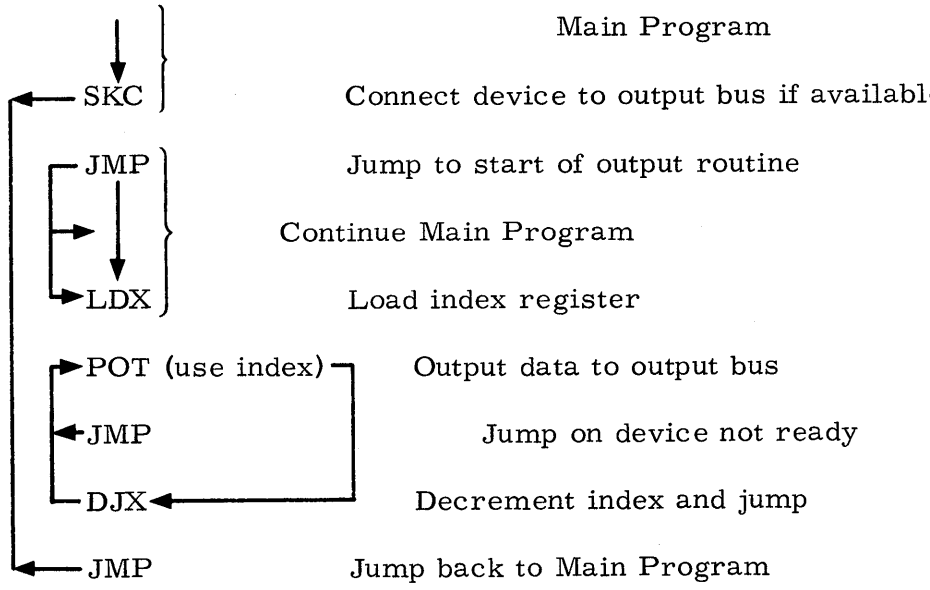
(No writing)

POA · PERIPHERAL OUT OF A (12 V XXXXX) TIME: 4

This instruction tests the condition of the output bus. If the output bus is ready to receive data, the 24-bit contents of the A register are placed on the output bus and the next instruction in the current sequence is skipped. If the output bus is not ready, no data is transmitted and the next instruction in the current sequence is executed in a normal manner. The contents of the A register are not altered by this instruction.

POT · PERIPHERAL OUT OF Y (53 V XXXXX) TIME: 4

This instruction tests the condition of the output bus. If the output bus is ready to receive data, the 24-bit contents of location Y are placed on the output bus and the next instruction in the current sequence is skipped. If the output bus is not ready, no data is transmitted and the next instruction in the current sequence is executed in a normal manner. The contents of the A register are not altered by this instruction. Example:

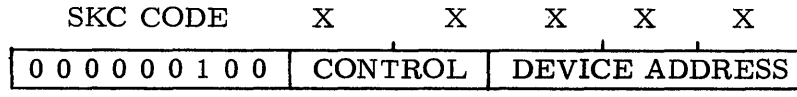


Maximum Transfer Rate	Maximum Search Rate
7 μsec	7 μsec

SKC · CONTROL AND SKIP (00 4 XXXXX) TIME: 4

This instruction transmits questions or commands to devices attached to the direct input/output bus. If a positive response or acknowledgement is received from the addressed device, the next instruction in the current sequence is executed in a normal manner. However, if the requested response is not received from the addressed device, the next instruction is skipped. This instruction is designed to be used with a peripheral input or output instruction for the transfer of data along the direct input/output channel.

The immediate bit configuration of X is interpreted as follows:



The address selects one of a possible 512 external devices, where bits 7 and 8 are defined as follows:

a) 00 ALERT Standard Peripherals
b) 01 ALERT System Peripherals
c) 10 DMA Device
d) 11 DMA Channel Number

The remaining bits of (a), (b), and (c) may be used to specify the address of the respective device; however, the remaining bits of (d) will specify the DMA channel number to be connected to the DMA input/output bus.

The control portion transmits questions or commands to the addressed device as follows:

01 Start this device

02 Stop this device

04 Did this device interrupt? If so, drop the interrupt

10 Has this device been started?

etc. etc.

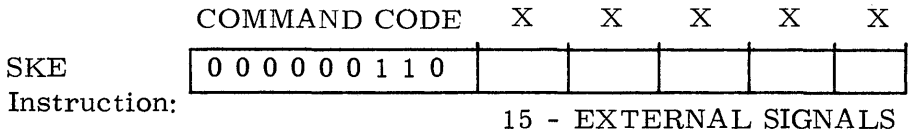
Indexing and indirect addressing do not apply to this instruction. Use of the SKC instruction automatically blocks both internal and external interrupts until the next instruction has been executed. Example:

Location	Instruction
2000	SKC 01575
2001	POT 2100
2002	JMP
2003	etc.

The instruction at location 2000 in the above example sends a start (01) command to the DMA device whose address is 175. If an acknowledge is not received by the computer confirming this action, the next instruction in the current sequence is skipped and the jump command at location 2002 is executed. However, if the acknowledge is received, the next instruction will be executed in the normal manner.

SKE · SKIP IF EXTERNAL SIGNALS NOT SET (00 6 XXXXX) TIME: 3

This instruction tests the condition of the specific external signals designated by the immediate bit configuration of X. Each one-bit of X uniquely defines a signal to be tested. If all designated signals are not set, the next instruction in the current sequence is skipped. If any designated point is set, the next instruction is executed in a normal manner. Indexing and indirect addressing do not apply to this instruction.



Example:

Location	Instruction
2000	SKE 01001
2001	JMP
2002	etc

The instruction at location 2000 tests the condition of signals 0 and 9. If both signals are not set, the next instruction (2001) will be skipped. However, if either 0 or 9 is set, the next instruction will be honored and the JMP will be executed in a normal manner.

STE · SET EXTERNAL POINT (00 2 XXXXX) TIME: 2

This instruction causes a pulse to be transmitted along the specific external activating lines designated by the immediate bit configuration of X. Each one-bit of X uniquely defines an external line to be activated. Any combination of 15 lines may be defined by the 15 bits of X. Indexing and indirect addressing do not apply to this instruction. Example:

Location	Instruction
3000	STE 40201

The instruction at location 3000 causes a pulse to be sent along external lines numbered 0, 7, and 14 as specified in the address field of the STE instruction.

INTERRUPT INSTRUCTIONS

LIM · LOAD INTERRUPT MASK (42 V XXXXX) TIME: 2

This instruction transfers the contents of location Y to the interrupt mask register. Each bit of the interrupt mask register provides selectivity of up to 24 unique external interrupts set in the interrupt register. Example:

Location	Instruction	Interrupt Mask	Interrupt Register	(3100)
(Before Execution)		00000004	00000007	00000005
3000	LIM 3100	00000005	00000007	00000005

Before execution, the interrupt mask register was set to select only interrupts on line number two. After execution, the interrupt mask register is set to select interrupts on lines 0 and 2, as the lines are numbered 0 to 23 from right to left.

SRB · SET/RESET INTERRUPT BLOCK (00 1 0000X) TIME: 2

This instruction is used to change the state of the interrupt block as designated by the immediate bit configuration of X. If X = 1, the interrupt block will be set, thus blocking all external interrupts. If X = 0, the interrupt block is reset (released), enabling all external interrupts; in this case, the next instruction in sequence will be in execution before the interrupt block is actually removed. Indexing and indirect addressing do not apply to this instruction.

STI · STORE INTERRUPT REGISTER (46 V XXXXX) TIME: 6

Store the contents of the interrupt register at location Y. The contents of the interrupt register are not altered by this instruction.

XML · EXCHANGE INTERRUPT MASK (43 V XXXXX) TIME: 6

Exchange the contents of the interrupt mask register with the contents of location Y.

LOGIC INSTRUCTIONS

EXT · EXTRACT (30 V XXXXX) TIME: 2

This instruction forms the logical (bit by bit) product of (A) and (Y). The result replaces the contents of A. Example:

Location	Instruction	A	(4000)
(Before Execution)		00077700	12345670
2000	EXT 4000	00045600	12345670

HAD · HALF ADD (31 V XXXXX) TIME: 2

This instruction performs a binary addition of the 24-bit contents of A and Y, discarding all carries, and retaining the result in the A register. This is the "logical exclusive OR" function. If the corresponding bit positions of (A) and (Y) have the same value, the result shall contain a zero in this position. In all other cases, the result shall contain a one. Example:

Location	Instruction	A	(5000)
(Before Execution)		10101010	77777777
3000	HAD 5000	67676767	77777777

SEX · SKIP ON EXTRACT ZERO (70 V XXXXX) TIME: 4

This instruction forms the logical (bit by bit) product of (A) and (Y) and compares the result with zero. If the result is zero, the next instruction in the current sequence is skipped. Otherwise, the next instruction is executed in the normal manner. The contents of A are not disturbed.

SMP · SUPERIMPOSE (33 V XXXXX) TIME: 2

This instruction sets the individual bits of A to one corresponding to ones in (Y), leaving the remaining bits of A unaltered. This is the logical "inclusive OR" function. Example:

Location	Instruction	A	(4000)
(Before Execution)		12345670	00700700
2000	SMP 4000	12745770	00700700

SST · SUBSTITUTE IN Y (32 V XXXXX) TIME: 7

Set the individual bits of Y with bits of A corresponding to ones in B, leaving the remaining bits of Y unaltered. The contents of A and B are not altered by this instruction. Example:

Location	Instruction	A	B	(3000)
(Before Execution)		12343210	00770077	77777777
2000	SST 3000	12343210	00770077	77347710

SSA · SUBSTITUTE IN A (11 V XXXXX) TIME: 2

Set the individual bits of A with bits of Y corresponding to zeros in B, leaving the remaining bits of A unaltered. The contents of Y and B are not altered by this instruction. Example:

Location	Instruction	A	B	(3000)
(Before Execution)		12343210	00770077	77777777
2000	SSA 3000	77347710	00770077	77777777

SHIFT INSTRUCTIONS

All shift instructions are identified by the basic operation code 10, and the address variant V available for indexing and indirect addressing. Bits 11 through 14 of the effective address are used to specify the method of shifting. Bits 0 through 5 of the effective address are used to specify the number of bit positions to be shifted (00 through 77). Indexing is performed in a normal manner on the complete 15-bit address. It is therefore possible to index the number of shifts with or without affecting the method of shifting. Indirect addressing may be used when bits 11 through 14 of the effective address are set to specify the method of shifting. In this case, the variant V is equal to seven and its associated address is used to reference a particular location in memory.

ALC · SHIFT (A) LEFT CYCLIC (10 V 000NN) TIME: $2 + \frac{NN}{3}$

This instruction shifts (A) circularly to the left NN bit positions. The lower-order bits are replaced with the higher-order bits as the word is shifted.

Example:

Location	Instruction	A
(Before Execution)		12345670
2000	ALC 3	23456701

ARC · SHIFT (A) RIGHT CYCLIC (10 V 100NN) TIME: $2 + \frac{NN}{3}$

This instruction shifts (A) circularly to the right NN bit positions. The higher order bits are replaced with the lower order bits as the word is shifted. Example:

Location	Instruction	A
(Before Execution)		12345670
2000	ARC 3	01234567

ALS · SHIFT (A) LEFT ARITHMETIC (10 V 200NN) TIME: $2 + \frac{NN}{3}$

This instruction shifts (A) arithmetically to the left NN bit positions. The higher-order bits are shifted off the end and lost and the lower-order bits are replaced with zeros as the word is shifted. Example:

Location	Instruction	A
(Before Execution)		76543211
2000	ALS 6	54321100
2001	ALS 1	30642200

ARS · SHIFT (A) RIGHT ARITHMETIC (10 V 300NN) TIME: $2 + \frac{NN}{3}$

This instruction shifts (A) arithmetically to the right NN bit positions. The lower-order bits are shifted off the end and lost. The higher-order bits are replaced with the original sign bit as the word is shifted. Example:

Location	Instruction	A
(Before Execution)		40000010
2000	ARS 6	77400000

LLC · SHIFT LONG LEFT CYCLIC (10 V 400NN) TIME: $2 + \frac{NN}{3}$

This instruction shifts (AB) circularly to the left NN bit positions. The lower order bits of B are replaced by the higher order bits of A as the word is shifted. Example:

Location	Instruction	A	B
(Before Execution)		12344321	00400000
2000	LLC 6	34432100	40000012

LRC · SHIFT LONG RIGHT CYCLIC (10 V 500NN) TIME: $2 + \frac{NN}{3}$

This instruction shifts (AB) circularly to the right NN bit positions. The higher order bits of A are replaced with the lower order bits of B as the word is shifted. Example:

Location	Instruction	A	B
(Before Execution)		54321234	00007000
2000	LRC 14	70005432	12340000

LLS · SHIFT LONG LEFT ARITHMETIC (10 V 600NN) TIME: $2 + \frac{NN}{3}$

This instruction shifts (AB) arithmetically to the left NN bit positions. The higher order bits of A are shifted off the end and lost and the lower order bits of B are replaced with zeros as the word is shifted. See examples below.

LRS · SHIFT LONG RIGHT ARITHMETIC (10 V 700NN) TIME: $2 + \frac{NN}{3}$

This instruction shifts (AB) arithmetically to the right NN bit positions. The lower order bits of B are shifted off the end and lost. The higher order bits of A are replaced with the original sign bit of A, as the word is shifted. See examples as follows,

LLP · SHIFT LONG LEFT AND PROTECT (10 V 640NN) TIME: $2 + \frac{NN}{3}$

This instruction shifts (AB) arithmetically to the left NN bit positions. The sign bit of B is skipped and thus protected as the word is shifted. As the higher order bits of A are shifted off the end and lost, the lower order bits of B are replaced with zeros. See examples below.

LRP · SHIFT LONG RIGHT AND PROTECT (10 V 740NN) TIME: $2 + \frac{NN}{3}$

This instruction shifts (AB) arithmetically to the right NN bits positions. The sign bit of B is skipped and thus protected as the word is shifted. As the low order bits of B are shifted off the end and lost, the high order bits of A are replaced with the original sign bit of A. Examples:

Location	Instruction	A	B
(Before Execution)		40302010	01020304
2000	LLS 14	20100102	03040000
2001	LRS 6	00201001	02030400
2002	LLP 6	20100104	03040000
2003	LRP 6	00201001	02030400

WORD TRANSMISSION

LDA · LOAD A (40 V XXXXX) TIME: 2

Transmit the contents of location Y to A.

LDB · LOAD B (41 V XXXXX) TIME: 2

Transmit the contents of location Y to B.

DLD · DOUBLE PRECISION LOAD (50 V XXXXX) TIME: 3

Transmit the contents of locations Y and Y + 1 to A and B respectively.

TAB · TRANSMIT (A) TO B (00 2 01707) TIME: 2

Transmit the contents of A to B. Indexing and indirect addressing do not apply.

TBA · TRANSMIT (B) TO A (00 2 00717) TIME: 2

Transmit the contents of B to A. Indexing and indirect addressing do not apply.

XAB · EXCHANGE (A) AND (B) (00 2 11707) TIME: 2

Exchange the contents of A with the contents of B. Indexing and indirect addressing do not apply.

XMA · EXCHANGE (Y) AND (A) (74 V XXXXX) TIME: 6

Exchange the contents of memory location Y with the contents of the A register.

STS · STORE STATUS (57 V XXXXX) TIME: 7

Transmit the status of the following indicators to their respective bit positions of location Y. The high order 12 bits of Y will be protected and the low order 6 bits will be set to zero.

Indicator	Y Bit Position
Overflow	6
Carry	7
Program Indicator 1	8
Program Indicator 2	9
Division Overcapacity	10
Interrupt Block History	11

All of these functions will be reset to zero, by this instruction, after the transfer is complete. Sense switch settings are not affected by this instruction.

RSS · RESTORE STATUS (56 V XXXXX) TIME: 2

Set the following indicators equal to the immediate bits of Y as follows:

Indicator	Y Bit Position
Overflow	6
Carry	7
Program Indicator 1	8
Program Indicator 2	9
Division Overcapacity	10
Interrupt Block	11

The action on the interrupt block is delayed until the next instruction is fetched. Sense switch settings are not affected by this instruction. Example:

Location	Instruction	Function Status
(Before Execution)		6400
2000	STS 2001	4000
2001	RSS 2400	6400
2002	JMP 3000	2400

NOTE: The current status of the interrupt block (not the block history) is displayed in the most significant bit under "Function Status". In this case, the block history has an assumed state of zero.

STA · STORE A (44 V XXXXX) TIME: 6

Transmit (A) to memory location Y.

STB · STORE B (45 V XXXXX) TIME: 6

Transmit (B) to memory location Y.

STZ · STORE ZERO (27 0 XXXXX) TIME: 6

Transmit a word of all zeros to memory location Y. This instruction is a special case of STX where the variant V = 0. Indexing does not apply to this instruction.

DST · DOUBLE PRECISION STORE (51 V XXXXX) TIME: 10

Transmit the contents of A and B to memory locations Y and Y + 1 respectively.

CHARACTER INSTRUCTIONS

LCH · LOAD CHARACTER (60 V XXXXX) TIME: 2

Select one of four 6-bit characters from the contents of location Y and transfer the character to its corresponding position in A. The remaining bits of A are protected during this operation. The address variant V has special meaning as follows:

V	Interpretation
1	Character No. 1 (Direct)
2	Character No. 2 (Direct)
3	Character No. 3 (Direct)
0	Character No. 4 (Direct)
5	Character No. 1 (Indirect)
6	Character No. 2 (Indirect)
7	Character No. 3 (Indirect)
4	Character No. 4 (Indirect)

NOTE: If indirect addressing is used, the address variant of the indirect address word has the normal meaning. In this case, indexed addressing would be permissible. Example:

Location	Instruction	A	(3000)
(Before Execution)		15151515	21222324
2000	LCH 3000, 2	15221515	21222324
2001	LCH 3000, 0	15221524	21222324

CSK · CHARACTER SKIP (62 V XXXXX) TIME: 5

This instruction compares a selected character in A with its corresponding character in location Y. If these selected characters are not equal, the next instruction in the current sequence is skipped. Otherwise, the next instruction is executed in a normal manner. The address variant V has the same interpretation as for the LCH instruction. The contents of A and Y are not altered by this instruction.

SCH · STORE CHARACTER (61 V XXXXX) TIME: 7

Store the specified 6-bit character from (A) to its corresponding character position in Y, while protecting the remaining characters of Y. The address variant V has the same interpretation as for the LCH instruction above.

TRAPPED INSTRUCTIONS

Trapped instructions cause an internal interrupt and a subsequence to an associated interrupt address. These special instructions are defined as follows:

T30 · TRAP TO LOCATION 00030 (octal) (75 V XXXXX) TIME: 3

T31 · TRAP TO LOCATION 00031 (octal) (71 V XXXXX) TIME: 3

T32 · TRAP TO LOCATION 00032 (octal) (67 V XXXXX) TIME: 3

T33 · TRAP TO LOCATION 00033 (octal) (63 V XXXXX) TIME: 3

T34 · TRAP TO LOCATION 00034 (octal) (47 V XXXXX) TIME: 3

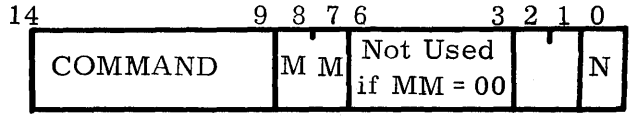
Trapped instructions may contain any bit configuration in the address and variant fields. However, these fields are ignored by the computer and will have no effect on the destination of the trapping operation.

Section 6

PERIPHERAL EQUIPMENT

The following paragraphs contain brief functional descriptions of the standard peripheral equipment for the Honeywell ALERT Computer. The basic units provide paper tape input and output, keyboard input and typewriter output and data input through data insert switches at the computer control unit. Figure 6-1 illustrates additional equipment available to any Honeywell ALERT/H-200 Laboratory System.

The general format of the SKC address, used to reference the standard ALERT peripherals, is a six-bit command field and nine-bit address field. The complete fifteen bit address of the SKC instruction is illustrated below:



If MM $\neq$ 00, the device address must conform to the specifications defined under SKC in the instruction repertoire. However, if MM = 00, only bits 0, 1, and 2 of the remaining eight bits are decoded to determine whether the address is for keyboard, printer, control panel, paper tape reader, or paper tape punch. If N = 0, the standard device being addressed is an input device. If N = 1, the device is used for output. The letter "Q" will represent the octal configuration of bits 0, 1, and 2, for the particular device.

The command codes for the standard set of peripherals are defined according to the respective device.

PAPER TAPE READER (Q = 6)

The paper tape reader is a bidirectional, computer-controlled unit, capable of three unique modes of tape processing:

Character by character (0-200 ch/sec).

Manual slewing (100-500 ch/sec, adjustable).

Wind/search (1,000 ch/sec).

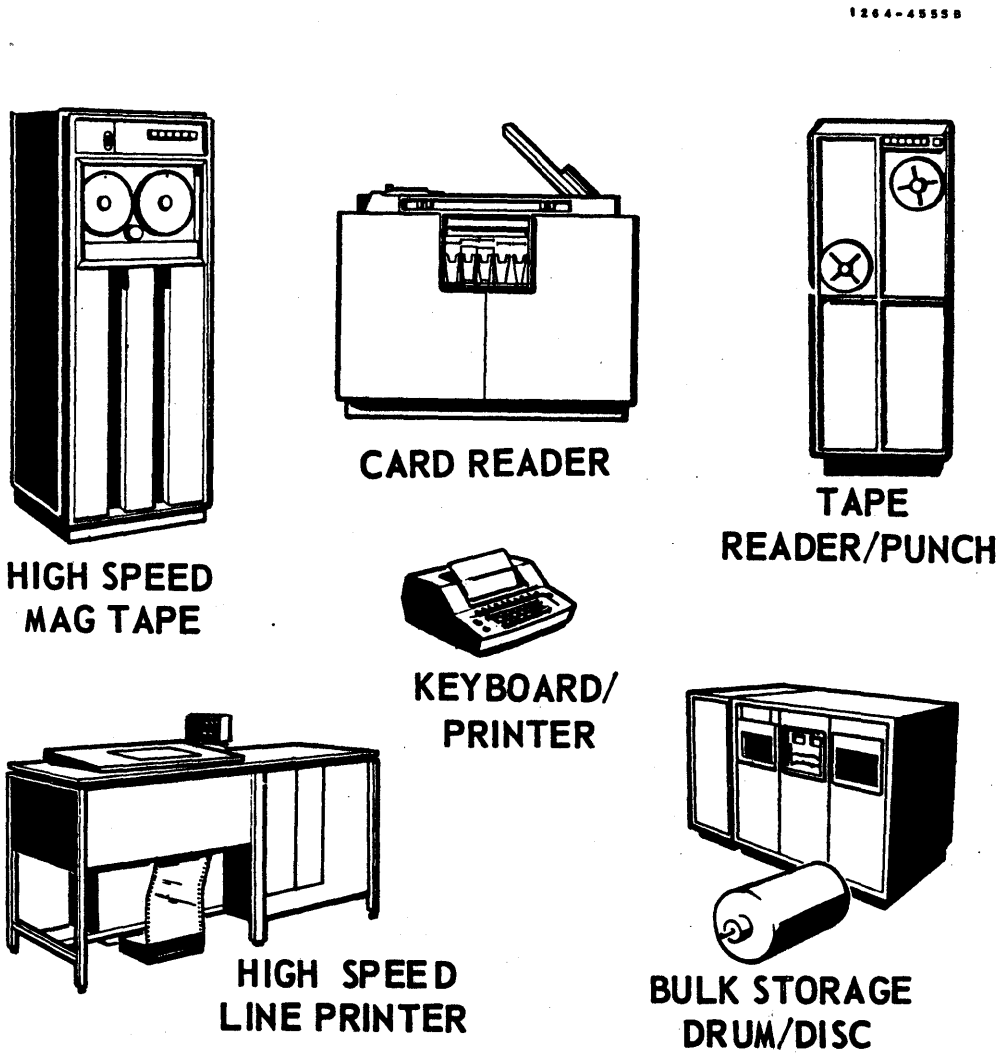


FIGURE 6-1. HONEYWELL COMMERCIAL PERIPHERALS

The six-bit command codes for the paper tape reader are:

Start forward character by character	(01)
Start backward character by character	(21)
Start forward wind/search	(41)
Start backward wind/search	(61)
Stop	(02)
Interrupt Inquiry	(04)
Ready-busy (respond if ready)	(10)

PAPER TAPE PUNCH (Q = 7)

The paper tape punch unit provides the capability of retrieving information stored in the computer memory The punch is capable of perforating eight level tape at any speed from zero to 150 ±10 characters per second.

The six-bit command codes for the paper tape punch are:

Start	(01)
Stop	(02)
Interrupt Inquiry	(04)
Ready-busy (respond it ready)	(10)

KEYBOARD (Q = 4)

The keyboard is a general purpose manual data entry device. It permits data transfer to the ALERT Computer during an operational program without stopping the program. Typical applications that exploit this feature include:

Insert and update data

Command an alternate operational mode

Control operation of communication links.

Manually override computer decisions under abnormal conditions.

The six-bit command coders for the standard keyboard are the same as those defined for the paper tape punch.

TYPEWRITER (Q = 5)

Upon appropriate commands from the computer, the standard typewriter unit can prepare a printed copy of any information stored in the computer. It accepts parallel six-bit characters in the Modified Baudot Code and responds (prints) at an average rate of 40 characters per second. The six-bit command codes for the standard printer are the same as defined for the paper tape punch. The character codes are tabulated in Appendix D.

COMPUTER CONTROL UNIT (Q = 2)

The data insert switches on the standard computer control panel may be connected to the direct input bus and referenced by the peripheral input (PIN or PTA) instructions. Typical functions that exploit this feature include:

Data entry at computer request.

System simulation.

NOTE

a. If an SKC command receives no response when connecting these standard peripherals, the associated device will not be connected and the status of the input-output bus will not be disturbed.

b. Each of these standard peripherals will contain the buffering and control circuits necessary to satisfy interface requirements of the ALERT computer.

Appendixes

A THROUGH H

Three lists of ALERT instructions are given for easy reference with execution times for Microbiax, One-Microsecond Core, and Two-Microsecond Core memory units. The following notations are used throughout:

( )	Contents of location(s) indicated within the parenthesis.
A	Main Accumulator (A-Register)
B	Auxiliary Accumulator (B-Register)
X	Address portion of Operand
V	Variant portion of Operand
Y	Effective address defined by the X and V combination
IMR	Interrupt Mask Register

Appendix A

FUNCTIONAL INSTRUCTION LIST WITH MICROBIAX EXECUTION TIMES

Control Instructions				Execution Time (μsec)
BAR	-	BRANCH AND RETURN	(04 V XXXXX)	6
EXC	-	EXECUTE	(01 V XXXXX)	1
HLT	-	HALT (IDLE)	(00 7 XXXXX)	2 + Δ
PAS	-	NO OPERATION	(00 0 XXXXX)	2
JAN	-	JUMP ON (A) NEGATIVE	(05 V XXXXX)	2
JAP	-	JUMP ON (A) POSITIVE	(15 V XXXXX)	2
JAZ	-	JUMP ON (A) ZERO	(14 V XXXXX)	2
JNZ	-	JUMP ON (A) NOT ZERO	(21 V XXXXX)	2
JMP	-	JUMP UNCONDITIONALLY	(65 V XXXXX)	2
SKN	-	SKIP INDICATOR NOT SET	(00 3 XXXXX)	3
SMZ	-	SKIP ON MEMORY ZERO	(03 V XXXXX)	3
SMN	-	SKIP ON MEMORY NOT ZERO	(73 V XXXXX)	3
SKM	-	SKIP IF (A) = (Y)	(02 V XXXXX)	4
SAN	-	SKIP IF (Y) $\neq$ (A)	(72 V XXXXX)	4
SML	-	SKIP IF (Y) $\leq$ (A)	(22 V XXXXX)	4
SMG	-	SKIP IF (Y) > (A)	(23 V XXXXX)	4
Arithmetic Instructions				
ADD	-	ADD TO (A)	(34 V XXXXX)	2
ABD	-	ADD TO (B)	(64 V XXXXX)	3
ADC	-	ADD TO (A) WITH CARRY	(54 V XXXXX)	2
SUB	-	SUBTRACT FROM (A)	(35 V XXXXX)	2
SBB	-	SUBTRACT FROM (B)	(66 V XXXXX)	2

Arithmetic Instructions (Continued)				Execution Time (μsec)
SUC	-	SUBTRACT FROM (A) WITH CARRY	(55 V XXXXX)	2
ADM	-	ADD (A) TO MEMORY	(24 V XXXXX)	7
TLY	-	TALLY	(25 V XXXXX)	12
MPY	-	MULTIPLY	(76 V XXXXX)	12
DIV	-	DIVIDE	(77 V XXXXX)	30
ABS	-	SET (A) POSITIVE	(00 200747)	2
CMP	-	COMPLEMENT (A)	(00 200727)	2
CLA	-	CLEAR A	(00 200700)	2
CAB	-	CLEAR A AND B	(00 201727)	2
Indexing Instructions				
AIX	-	AUGMENT INDEX IMMEDIATE	(37 V XXXXX)	2
AUX	-	AUGMENT INDEX BY (Y)	(36 V XXXXX)	3
IJX	-	INCREMENT AND JUMP	(20 V XXXXX)	2
DJX	-	DECREMENT AND JUMP	(07 V XXXXX)	2
JIX	-	JUMP ON INDEX NOT ZERO	(17 V XXXXX)	2
SXI	-	SKIP IF INDEX EQUALS Y	(06 V XXXXX)	3
SKX	-	SKIP IF INDEX EQUALS (Y)	(16 V XXXXX)	4
LDX	-	LOAD INDEX	(26 V XXXXX)	2
STX	-	STORE INDEX	(27 V XXXXX)	6
TAX	-	TRANSFER (A) TO INDEX	(00 200V07)	2
TXA	-	TRANSFER INDEX TO A	(00 20070V)	2

Input/Output Instructions				Execution Time (μsec)
PTA	-	PERIPHERAL INTO A	(13 V XXXXX)	4
PIN	-	PERIPHERAL INTO Y	(52 V XXXXX)	8/4
POA	-	PERIPHERAL OUT OF A	(12 V XXXXX)	4
POT	-	PERIPHERAL OUT OF Y	(53 V XXXXX)	4
SKC	-	CONTROL AND SKIP	(00 4 XXXXX)	4
SKE	-	SKIP IF EXTERNAL SIGNALS NOT SET	(00 6 XXXXX)	3
STE	-	SET EXTERNAL POINT	(00 2 XXXXX)	2
Interrupt Instructions				
LIM	-	LOAD INTERRUPT MASK	(42 V XXXXX)	2
SRB	-	SET/RESET INTERRUPT BLOCK	(00 1 0000X)	2
STI	-	STORE INTERRUPT REGISTER	(46 V XXXXX)	6
XML	-	EXCHANGE INTERRUPT MASK	(43 V XXXXX)	6
Logic Instructions				
EXT	-	EXTRACT	(30 V XXXXX)	2
HAD	-	HALF ADD	(31 V XXXXX)	2
SEX	-	SKIP ON EXTRACT ZERO	(70 V XXXXX)	4
SMP	-	SUPERIMPOSE	(33 V XXXXX)	2
SST	-	SUBSTITUTE IN Y	(32 V XXXXX)	7
SSA	-	SUBSTITUTE IN A	(11 V XXXXX)	2
Shift Instructions				
ALC	-	SHIFT (A) LEFT CYCLIC	(10 V 000NN)	$2 + \frac{NN}{3}$
ARC	-	SHIFT (A) RIGHT CYCLIC	(10 V 100NN)	$2 + \frac{NN}{3}$

Shift Instructions (Continued)				Execution Time (μsec)
ALS	-	SHIFT (A) LEFT ARITHMETIC	(10 V 200NN)	$2 + \frac{NN}{3}$
ARS	-	SHIFT (A) RIGHT ARITHMETIC	(10 V 300NN)	$2 + \frac{NN}{3}$
LLC	-	SHIFT LONG LEFT CYCLIC	(10 V 400NN)	$2 + \frac{NN}{3}$
LRC	-	SHIFT LONG RIGHT CYCLIC	(10 V 500NN)	$2 + \frac{NN}{3}$
LLS	-	SHIFT LONG LEFT ARITHMETIC	(10 V 600NN)	$2 + \frac{NN}{3}$
LRS	-	SHIFT LONG RIGHT ARITHMETIC	(10 V 700NN)	$2 + \frac{NN}{3}$
LLP	-	SHIFT LONG LEFT AND PROTECT SIGN OF B	(10 V 640NN)	$2 + \frac{NN}{3}$
LRP	-	SHIFT LONG RIGHT AND PROTECT SIGN OF B	(10 V 740NN)	$2 + \frac{NN}{3}$

Word Transmission				
LDA	-	LOAD A	(40 V XXXXX)	2
LDB	-	LOAD B	(41 V XXXXX)	2
DLD	-	DOUBLE PRECISION LOAD	(50 V XXXXX)	3
TAB	-	TRANSMIT (A) TO B	(00 2 01707)	2
TBA	-	TRANSMIT (B) TO A	(00 2 00717)	2
XAB	-	EXCHANGE (A) AND (B)	(00 2 11707)	2
XMA	-	EXCHANGE (Y) AND (A)	(74 V XXXXX)	6
STS	-	STORE STATUS	(57 V XXXXX)	7
RSS	-	RESTORE STATUS	(56 V XXXXX)	2
STA	-	STORE A	(44 V XXXXX)	6
STB	-	STORE B	(45 V XXXXX)	6
STZ	-	STORE ZERO	(27 V XXXXX)	6
DST	-	DOUBLE PRECISION STORE	(51 V XXXXX)	10

Character Instructions				Execution Time (μsec)
LCH	-	LOAD CHARACTER	(60 V XXXXX)	2
CSK	-	CHARACTER SKIP	(62 V XXXXX)	5
SCH	-	STORE CHARACTER	(61 V XXXXX)	7
Trapped Instructions				
T30	-	TRAP TO LOCATION 00030	(75 V XXXXX)	3
T31	-	TRAP TO LOCATION 00031	(71 V XXXXX)	3
T32	-	TRAP TO LOCATION 00032	(67 V XXXXX)	3
T33	-	TRAP TO LOCATION 00033	(63 V XXXXX)	3
T34	-	TRAP TO LOCATION 00034	(47 V XXXXX)	3

Note: The address and variant of trapped instructions has no effect on the destination of the trapping operation.

Appendix B

ALPHABETICAL INSTRUCTION LIST WITH ONE-MICROSECOND CORE EXECUTION TIMES

Instruction Mnemonic	Machine Oper. Code	Execution Time (μsec)	Interpretation
ABS	00 2 00747	2	Set (A) Positive
ABD	64	3	Add to (B)
ADC	54	2	Add with Carry
ADD	34	2	Add to (A)
ADM	24	4	Add (A) to Memory
AIX	37	2	Augment Index Immediate
ALC	10 V 000NN	2 + N/3	Shift (A) Left Cyclic
ALS	10 V 200NN	2 + N/3	Shift (A) Left Arithmetic
ARC	10 V 100NN	2 + N/3	Shift (A) Right Cyclic
ARS	10 V 300NN	2 + N/3	Shift (A) Right Arithmetic
AUX	36	3	Augment by (Y)
BAR	04	3	Branch and Return
CAB	00 2 01727	2	Clear A and B
CLA	00 2 00700	2	Clear A
CMP	00 2 00727	2	Complement A
CSK	62	5	Character Skip

Instruction Mnemonic	Machine Oper. Code	Execution Time (μsec)	Interpretation
DIV	77	30	Divide
DJX	07	2	Decrement and Jump
DLD	50	3	Double Precision Load
DST	51	4	Double Precision Store
EXC	01	1	Execute
EXT	30	2	Extract
HAD	31	2	Half Add
HLT	00 7 XXXXX	2 + Δ	Halt (Idle)
IJX	20	2	Increment and Jump
JAN	05	2	Jump on a Negative
JAP	15	2	Jump on a Positive
JAZ	14	2	Jump on a Zero
JIX	17	2	Jump on Index Not Zero
JMP	65	2	Jump Unconditionally
JNZ	21	2	Jump on a Not Zero
LCH	60	2	Load Character
LDA	40	2	Load A
LDB	41	2	Load B
LDX	26	2	Load Index
LIM	42	2	Load Interrupt Mask

Instruction Mnemonic	Machine Oper. Code	Execution Time (μsec)	Interpretation
LLC	10 V 400NN	2 + N/3	Shift Long Left Cyclic
LLP	10 V 640NN	2 + N/3	Shift Long Left and Protect
LLS	10 V 600NN	2 + N/3	Shift Long Left Arithmetic
LRC	10 V 500NN	2 + N/3	Shift Long Right Cyclic
LRS	10 V 700NN	2 + N/3	Shift Long Right Arithmetic
LRP	10 V 740NN	2 + N/3	Shift Long Right and Protect
MPY	76	12	Multiply
PAS	00 0 XXXXX	2	No Operation
PTA	13	4	Peripheral Input to A
PIN	52	5/4	Peripheral Input to Y
POA	12	4	Peripheral Out of A
POT	53	4	Peripheral Out of Y
RSS	56	2	Restore Status
SAN	72	4	Skip if (A) $\neq$ (Y)
SBB	66	3	Subtract from (B)
SCH	61	4	Store Character
SEX	70	4	Skip on Extract Zero
SKC	00 4 XXXXX	4	Control and Skip
SKE	00 6 XXXXX	3	Skip if all Ext. Not Set
SKM	02	4	Skip if (A) = (Y)

Instruction Mnemonic	Machine Oper. Code	Execution Time (μsec)	Interpretation
SKN	00 3 XXXXX	3	Skip if all Internals Not Set
SKX	16	4	Skip if Index = (Y)
SMG	23	4	Skip if (Y) > (A)
SML	22	4	Skip if (Y) $\leq$ (A)
SMN	73	3	Skip if (Y) is not Zero
SMP	33	2	Superimpose
SMZ	03	3	Skip on Memory Zero
SRB	00 1 0000X	2	Set/Reset Interrupt Block
SSA	11	2	Substitute in A
SST	32	4	Substitute in Y
STA	44	3	Store A
STB	45	3	Store B
STE	00 5 XXXXX	2	Set External Points
STI	46	3	Store Interrupt Register
STS	57	4	Store Status
STX	27	3	Store Index
STZ	27 0 XXXXX	3	Store Zero in X
SUB	35	2	Subtract from Z
SUC	55	2	Subtract with Carry
SXI	06	3	Skip on Index = Y

Instruction Mnemonic	Machine Oper. Code	Execution Time (μsec)	Interpretation
TAB	00 2 01707	2	Transfer (A) to B
TAX	00 2 00X07	2	Transfer (A) to Index X
TBA	00 2 00717	2	Transfer (B) to A
TLY	25	4	Tally
TXA	00 2 0070X	2	Transfer Index to A
XAB	00 2 11707	2	Exchange (A) and (B)
XMA	74	3	Exchange (Y) and (A)
XML	43	3	Exchange (Y) and (IMR)
Trapped Instructions			
T30	75	3	Trap to Loc. 00030_8
T31	71	3	Trap to Loc. 00031_8
T32	67	3	Trap to Loc. 00032_8
T33	63	3	Trap to Loc. 00033_8
T34	47	3	Trap to Loc. 00034_8

Appendix C

NUMERICAL INSTRUCTION LIST WITH TWO-MICROSECOND CORE EXECUTION TIMES

Instruction Mnemonic	Machine Oper. Code	Execution Time (μsec)	Interpretation
PAS	00 0 XXXXX	3	No Operation
SRB	00 1 0000X	3	Set/Reset Interrupt Block
CLA	00 2 00700	3	Clear A
TAX	00 2 00X07	3	Transfer (A) to Index X
TXA	00 2 0070X	3	Transfer Index X to A
TBA	00 2 00717	3	Transfer (B) to A
CMP	00 2 00727	3	Complement A
ABS	00 2 00747	3	Set (A) Positive
TAB	00 2 01707	3	Transfer (A) to B
CAB	00 2 01727	3	Clear A and B
XAB	00 2 11707	3	Exchange (A) and (B)
SKN	00 3 XXXXX	4	Skip if all Internals Not Set
SKC	00 4 XXXXX	5	Control and Skip
STE	00 5 XXXXX	3	Set External Point
SKE	00 6 XXXXX	4	Skip if all Externals Not Set
HLT	00 7 XXXXX	2 + Δ	Halt (Idle)

Instruction Mnemonic	Machine Oper. Code	Execution Time (μsec)	Interpretation
EXC	01	2	Execute
SKM	02	6	Skip if (Y) (A)
SMZ	03	5	Skip if (Y) is Zero
BAR	04	6	Branch and Return
JAN	05	4	Jump on (A) Negative
SXI	06	5	Skip if Index Equals Y
DJX	07	4	Decrement and Jump
ALC	10 V 000NN	4 + N/3	Shift (A) Left Cyclic
ARC	10 V 100NN	4 + N/3	Shift (A) Right Cyclic
ALS	10 V 200NN	4 + N/3	Shift (A) Left Arithmetic
ARS	10 V 300NN	4 + N/3	Shift (A) Right Arithmetic
LLC	10 V 400NN	4 + N/3	Shift (AB) Left Cyclic
LRC	10 V 500NN	4 + N/3	Shift (AB) Right Cyclic
LLS	10 V 600NN	4 + N/3	Shift (AB) Left Arithmetic
LLP	10 V 640NN	4 + N/3	Shift (AB) Left and Protect
LRS	10 V 700NN	4 + N/3	Shift (AB) Right Arithmetic
LRP	10 V 740NN	4 + N/3	Shift (AB) Right and Protect
SSA	11	4	Substitute INA A
POA	12	6	Peripheral Output From A
PTA	13	6	Peripheral Input To A

Instruction Mnemonic	Machine Oper. Code	Execution Time (μsec)	Interpretation
JAZ	14	4	Jump on A Zero
JAP	15	4	Jump on A Positive
SKX	16	6	Skip if Index Equals (Y)
JIX	17	4	Jump on Index Not Zero
IJX	20	4	Increment and Jump
JNZ	21	4	Jump if (A) is not Zero
SML	22	6	Skip if (Y) $\leq$ (A)
SMG	23	6	Skip if (Y) > (A)
ADM	24	7	Add (A) to Memory
TLY	25	7	Tally
LDX	26	4	Load Index
STX	27	6	Store Index
STZ	27 0 XXXXX	6	Store Zero in Y
EXT	30	4	Extract
HAD	31	4	Half Add
SST	32	7	Substitute in Y
SMP	33	4	Superimpose
ADD	34	4	Add to (A)
SUB	35	4	Subtract from (A)
AUX	36	5	Augment Index by (Y)

Instruction Mnemonic	Machine Oper. Code	Execution Time (μsec)	Interpretation
AIX	37	4	Augment Index by Y
LDA	40	4	Load A
LDB	41	4	Load B
LIM	42	4	Load Interrupt Mask
XML	43	6	Exchange (Y) and (Interrupt Mask Reg.)
STA	44	6	Store A
STB	45	6	Store B
STI	46	6	Store Interrupt Register
T34	47	5	Trap to Loc. 00034_8
DLD	50	6	Double Precision Load
DST	51	8	Double Precision Store
PIN	52	8/6	Peripheral Input to Y
POT	53	6	Peripheral Output from Y
ADC	54	4	Add With Carry
SUC	55	4	Subtract With Carry
RSS	56	4	Restore Status
STS	57	7	Store Status
LCH	60	4	Load Character
SCH	61	7	Store Character
CSK	62	7	Character Skip

Instruction Mnemonic	Machine Oper. Code	Execution Time (μsec)	Interpretation
T33	63	5	Trap to Loc. 00033_8
ABD	64	5	Add to (B)
JMP	65	4	Jump Unconditionally
SBB	66	5	Subtract From (B)
T32	67	5	Trap to Loc. 00032_8
SEX	70	6	Skip on Extract Zero
T31	71	5	Trap to Loc. 00031_8
SAN	72	6	Skip on (A) = (Y)
SMN	73	5	Skip on (Y) Not Zero
XMA	74	6	Exchange (Y) and (A)
T30	75	5	Trap to Loc. 00030_8
MPY	76	14	Multiply
DIV	77	32	Divide (AB) By (Y)

Appendix D

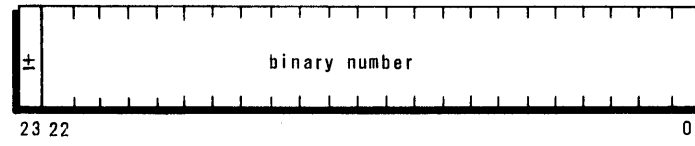
KEYBOARD/PRINTER CHARACTER CODES

Character	Key Punch Code	Standard Code	Keyboard* Printer Code	Character	Key Punch Code	Standard Code	Keyboard* Printer Code
0	0	000000	66	-	X	100000	43
1	1	000001	67	J	X, 1	100001	13
2	2	000010	63	K	X, 2	100010	17
3	3	000011	41	L	X, 3	100011	22
4	4	000100	52	M	X, 4	100100	34
5	5	000101	60	N	X, 5	100101	14
6	6	000110	65	O	X, 6	100110	30
7	7	000111	47	P	X, 7	100111	26
8	8	001000	46	Q	X, 8	101000	27
9	9	001001	70	R	X, 9	101001	12
	8, 2	001010			X, 8, 2	101010	
#	8, 3	001011		$	X, 8, 3	101011	51
:	8, 4	001100	56	*	X, 8, 4	101100	76
Space	Blank	001101	04		X, 8, 5	101101	
C/R	8, 6	001110	10		X, 8, 6	101110	
L. F.	8, 7	001111	02		X, 0	101111	
&	R	010000	72		8, 5	110000	
A	R, 1	010001	03	/	0, 1	110001	75
B	R, 2	010010	31	S	0, 2	110010	05
C	R, 3	010011	16	T	0, 3	110011	20
D	R, 4	010100	11	U	0, 4	110100	07
E	R, 5	010101	01	V	0, 5	110101	36
F	R, 6	010110	15	W	0, 6	110110	23
G	R, 7	010111	32	X	0, 7	110111	35
H	R, 8	011000	24	Y	0, 8	111000	25
I	R, 9	011001	06	Z	0, 9	111001	21
'	R, 8, 2	011010	53	?	0, 8, 2	111010	71
.	R, 8, 3	011011	74	,	0, 8, 3	111011	54
)	R, 8, 4	011100	62	(	0, 8, 4	111100	57
	R, 8, 5	011101			0, 8, 5	111101	
"	R, 8, 6	011110	61		0, 8, 6	111110	
!	R, 0	011111	55		0, 8, 7	111111	

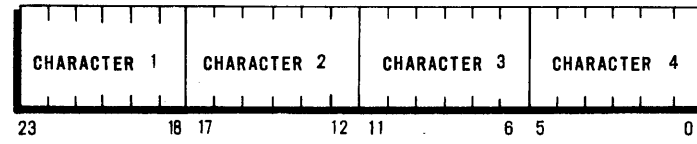
* Kleinschmidt Keyboard/Page Printer

Appendix E
TYPES OF MACHINE WORDS

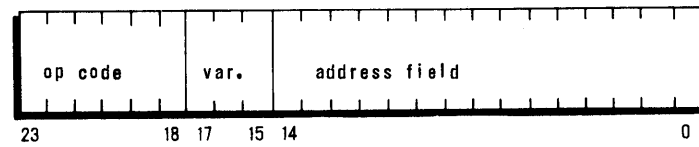
Fixed-Point Word



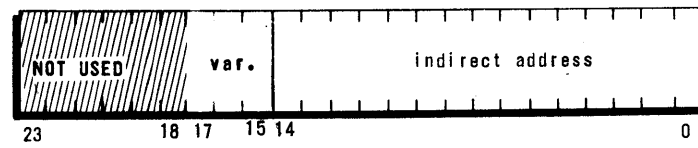
Alphanumeric Word



Instruction Word



Indirect-Address Word



Appendix F
TWOS-COMPLEMENT ARITHMETIC

COMPLEMENTATION

Two types of complements are commonly used for a number of given radix.[1] These are the radix-minus-1 complement and the true complement. It is the purpose of this appendix to demonstrate the distinction between the two and the use which the *ALERT* makes of the true complement.

TO FIND THE RADIX-MINUS-1 COMPLEMENT, SUBTRACT EACH DIGIT FROM THE RADIX MINUS 1. In the decimal system, the radix minus 1 is 9. To find the radix-minus-1 complement of decimal 44, subtract 44 from 99, giving 55. A convenient property of the binary number system is that the radix-minus-1 complement may be generated by changing all 0's to 1's and all 1's to 0's, as illustrated below.

(radix-minus-1)'s:	9 9	n_2:	100
subtract n_{10}:	4 4	invert:	⇩
9's complement:	5 5	1's complement:	011

Specifically, the radix-minus-1 complement of a binary number is called the ones complement, the radix-minus-1 complement of an octal number is called the sevens complement, and the radix-minus-1 complement of a decimal number is called the nines complement.

TO FIND THE TRUE COMPLEMENT OF A NUMBER, ADD 1 TO ITS RADIX-MINUS-1 COMPLEMENT. As demonstrated above, the radix-minus-1 complement of decimal 44 is 55; adding 1 by the rules of decimal addition gives 56, which is the true complement of decimal 44. Likewise, the true complement of binary 100 is 011 plus 1, which, by the rules of binary addition, is binary 100.

Specifically, the true complement of a binary number is called the twos complement, the true complement of an octal number is called the eights complement, and the true complement of a decimal number is called the tens complement.

TWOS-COMPLEMENT NOTATION

To clarify some essential properties of twos-complement numbers and twos-complement

[1] The radix of a number system is the quantity of permissible symbols in the system.

arithmetic, a four-bit word is used below as a frame of reference. The properties of this word that are discussed apply as well to the 24-bit Honeywell *ALERT* word. The numbers that can be represented by a four-bit word are listed in Table F-1 below.

Table F-1. Twos-Complement Notation

Decimal Number	Binary Number	Decimal Number	(2's Complement) Binary Number
		- 8	1 0 0 0
+ 7	0 1 1 1	- 7	1 0 0 1
+ 6	0 1 1 0	- 6	1 0 1 0
+ 5	0 1 0 1	- 5	1 0 1 1
+ 4	0 1 0 0	- 4	1 1 0 0
+ 3	0 0 1 1	- 3	1 1 0 1
+ 2	0 0 1 0	- 2	1 1 1 0
+ 1	0 0 0 1	- 1	1 1 1 1
+ 0	0 0 0 0		

The table shows that positive numbers are represented in main memory in pure binary, as distinguished from twos-complement binary. The high-order bit is always zero and can be interpreted as a plus sign. The rest of the word can be interpreted directly from binary to decimal using the rules of positional notation. For example, 0111 may be interpreted as plus 7_{10}.

There is only one zero in twos-complement notation. Therefore zero ambiguity is impossible.

Negative numbers are represented in twos-complement form. The high-order bit is always 1 and can be interpreted as a minus sign. Such numbers, including the sign bit, must be recomplemented if they are to be interpreted directly from binary to decimal using the rules of positional notation. Any of the following equivalent methods may be used:

1. RECOMPLEMENT THE BINARY NUMBER (take the true complement). That is, invert all 1's to 0's and all 0's to 1's. Then add 1 to the low-order bit.
2. UNCOMPLEMENT THE BINARY NUMBER. That is, subtract 1 from the low-order bit using the rules of binary subtraction. Then invert all 1's to 0's and all 0's to 1's.
3. BINARY TRUE COMPLEMENTATION. Beginning with the low-order bit position, proceed to the high-order end of the number as follows. If the bit is 0 or the first-encountered 1, put it in the corresponding position of the result; else invert it and put it in the corresponding position of the result.
4. OCTAL TRUE COMPLEMENTATION. Beginning with the low-order octal digit, proceed to the high-order end of the number as follows. If the digit is 0, put it in the corresponding position of the result; else take the 8's complement of it, and take the 7's complement of the remaining (higher-order) digits.
5. GENERAL TRUE COMPLEMENTATION. Beginning with the low-order digit, proceed to the high-order end of the number as follows. If the digit is 0, put it in the corresponding position of the result; else take the true complement of it, and take the radix-minus-1 complement of the remaining (higher order) digits.

TWOS-COMPLEMENT ARITHMETIC

One of the advantages of twos-complement notation is that negative numbers stored in memory as twos-complement operands can be added together as full-word, unsigned integers during arithmetic operations. Yet the high-order bit of the result can be interpreted as the sign of the result, with the qualification, stated above, that a negative result must be recomplemented if it is to be interpreted directly by the rules of positional notation.

The first example in Figure F-1 illustrates subtraction as it is normally performed using decimal arithmetic. The second example shows that the same problem can be solved by adding the true complement of the subtrahend instead of by subtracting the subtrahend itself. And the third example demonstrates that the same principle applies to binary numbers. Note that, when subtraction is performed by adding the true complement, all operands must have the same number of digits. The need for this can be understood by observing that the complement of, for example, 100_2 is not identical to the complement of 0100_2.

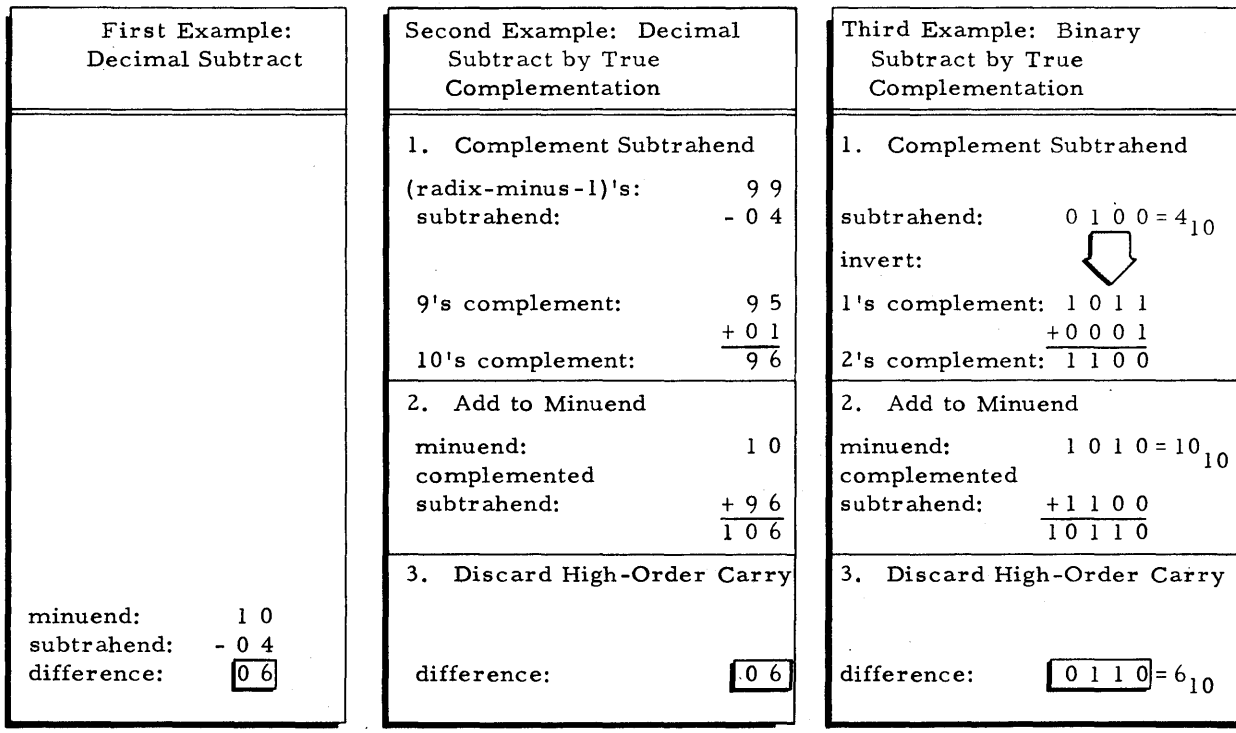


Figure F-1. Twos-Complement Arithmetic

The following examples illustrate the way in which the *ALERT* performs arithmetic operations. Again, a four-bit word is used for brevity.

Example 1: 7 + (-8) = -1

Assume that -8 has been loaded into the accumulator (A) and that +7 has been stored at address x. Then, using the ADD (Add to Accumulator) instruction, (A) plus (x) replace (A) as follows:

(A):	1 0 0 0 ("-8")
(x):	0 1 1 1 (+7)
new (A):	1 1 1 1 ("-1")

The result, 1111_2, is minus one according to the twos-complement collating sequence (Table F-1). The quotation marks merely indicate that the number is in its twos-complement form.

Example 2: 2 - (+3) = -1

Assume that the minuend, +2, is in the accumulator (A) and that the subtrahend, +3, has been stored at address x. Then, using the SUB (Subtract from Accumulator) instruction, the contents of x are twos-complemented:

(x):	0 0 1 1 (+3)
complemented (x):	1 1 0 1 ("-3")

The complemented subtrahend is then added to the contents of the accumulator. The sum replaces the contents of the accumulator.

(A):	0 0 1 0 (+2)
complemented (x):	1 1 0 1 ("-3")
new (A):	1 1 1 1 ("-1")

Appendix G

ALERT SOFTWARE* SUMMARY

Two standard software systems are supplied with each Honeywell ALERT computer, the Honeywell ALERT Software and the Honeywell MASCOT Support Software.

The Honeywell ALERT software system is a group of basic programs designed for operation on the basic computer. This group includes the SNAP-ALERT symbolic assembler, the STEP-ALERT paper tape edit program, a set of computer confidence tests, and a large number of utility subroutines necessary for effective program development.

The Honeywell MASCOT Support System is operated on the Honeywell 800 and 1800 computers and is capable of being implemented on any major data processing system of adequate capacity. The system includes a symbolic program processor for assembling and editing, a relocatable loader, an address search program and an ALERT simulator. The primary objective of the MASCOT systems is to support the complete development, filing, and documentation of programs for the ALERT computer.

* Refer to Honeywell ALERT Software Systems General Description, Third Edition, R-ED 24292

Appendix H

POWERS OF TWO

2^n	n	2^{-n}
1	0	1.0
2	1	0.5
4	2	0.25
8	3	0.125
16	4	0.062 5
32	5	0.031 25
64	6	0.015 625
128	7	0.007 812 5
256	8	0.003 906 25
512	9	0.001 953 125
1 024	10	0.000 976 562 5
2 048	11	0.000 488 281 25
4 096	12	0.000 244 140 625
8 192	13	0.000 122 070 312 5
16 384	14	0.000 061 035 156 25
32 768	15	0.000 030 517 578 125
65 536	16	0.000 015 258 789 062 5
131 072	17	0.000 007 629 394 531 25
262 144	18	0.000 003 814 697 265 625
524 288	19	0.000 001 907 348 632 812 5
1 048 576	20	0.000 000 953 674 316 406 25
2 097 152	21	0.000 000 476 837 158 203 125
4 194 304	22	0.000 000 238 418 579 101 562 5
8 388 608	23	0.000 000 119 209 289 550 781 25
16 777 216	24	0.000 000 059 604 644 775 390 625
33 554 432	25	0.000 000 029 802 322 387 695 312 5
67 108 864	26	0.000 000 014 901 161 193 847 656 25
134 217 728	27	0.000 000 007 450 580 596 923 828 125
268 435 456	28	0.000 000 003 725 290 298 461 914 062 5
536 870 912	29	0.000 000 001 862 645 149 230 957 031 25
1 073 741 824	30	0.000 000 000 931 322 574 615 478 515 625
2 147 483 648	31	0.000 000 000 465 661 287 307 739 257 812 5
4 294 967 296	32	0.000 000 000 232 830 643 653 869 628 906 25
8 589 934 592	33	0.000 000 000 116 415 321 826 934 814 453 125
17 179 869 184	34	0.000 000 000 058 207 660 913 467 407 226 562 5
34 359 738 368	35	0.000 000 000 029 103 830 456 733 703 613 281 25
68 719 476 736	36	0.000 000 000 014 551 915 228 366 851 806 640 625
137 438 953 472	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
274 877 906 944	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
549 755 813 888	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125
1 099 511 627 776	40	0.000 000 000 000 909 494 701 772 928 237 915 039 062 5
2 199 023 255 552	41	0.000 000 000 000 454 747 350 886 464 118 957 519 531 25
4 398 046 511 104	42	0.000 000 000 000 227 373 675 443 232 059 478 759 765 625
8 796 093 022 208	43	0.000 000 000 000 113 686 837 721 616 029 739 379 882 812 5
17 592 186 044 416	44	0.000 000 000 000 056 843 418 860 808 014 869 689 941 406 25
35 184 372 088 832	45	0.000 000 000 000 028 421 709 430 404 007 434 844 970 703 125
70 368 744 177 664	46	0.000 000 000 000 014 210 854 715 202 003 717 422 485 351 562 5
140 737 488 355 328	47	0.000 000 000 000 007 105 427 357 601 001 858 711 242 675 781 25
281 474 976 710 656	48	0.000 000 000 000 003 552 713 678 800 500 929 355 621 337 890 625
562 949 953 421 312	49	0.000 000 000 000 001 776 356 839 400 250 464 677 810 668 945 312 5
1 125 899 906 842 624	50	0.000 000 000 000 000 888 178 419 700 125 232 338 905 334 472 656 25
2 251 799 813 685 248	51	0.000 000 000 000 000 444 089 209 850 062 616 169 452 667 236 328 125
4 503 599 627 370 496	52	0.000 000 000 000 000 222 044 604 925 031 308 084 726 333 618 164 062 5
9 007 199 254 740 992	53	0.000 000 000 000 000 111 022 302 462 515 654 042 363 166 809 082 031 25
18 014 398 509 481 984	54	0.000 000 000 000 000 055 511 151 231 257 827 021 181 583 404 541 015 625
36 028 797 018 963 968	55	0.000 000 000 000 000 027 755 575 615 628 913 510 590 791 702 270 507 812 5
72 057 594 037 927 936	56	0.000 000 000 000 000 013 877 787 807 814 456 755 295 395 851 135 253 906 25
144 115 188 075 855 872	57	0.000 000 000 000 000 006 938 893 903 907 228 377 647 697 925 567 626 953 125
288 230 376 151 711 744	58	0.000 000 000 000 000 003 469 446 951 953 614 188 823 848 962 783 813 476 562 5
576 460 752 303 423 488	59	0.000 000 000 000 000 001 734 723 475 976 807 094 411 924 481 391 906 738 281 25
1 152 921 504 606 846 976	60	0.000 000 000 000 000 000 867 361 737 988 403 547 205 962 240 695 953 369 140 625
2 305 843 009 213 693 952	61	0.000 000 000 000 000 000 433 680 868 994 201 773 602 981 120 347 976 684 570 312 5
4 611 686 018 427 387 904	62	0.000 000 000 000 000 000 216 840 434 497 100 886 801 490 560 173 988 342 285 156 25
9 223 372 036 854 775 808	63	0.000 000 000 000 000 000 108 420 217 248 550 443 400 745 280 086 994 171 142 578 125
18 446 744 073 709 551 616	64	0.000 000 000 000 000 000 054 210 108 624 275 221 700 372 640 043 497 085 571 289 062 5
36 893 488 147 419 103 232	65	0.000 000 000 000 000 000 027 105 054 312 137 610 850 186 320 021 748 542 785 644 531 25
73 786 976 294 838 206 464	66	0.000 000 000 000 000 000 013 552 527 156 068 805 425 093 160 010 874 271 392 822 265 625
147 573 952 589 676 412 928	67	0.000 000 000 000 000 000 006 776 263 578 034 402 712 546 580 005 437 135 696 411 132 812 5
295 147 905 179 352 825 856	68	0.000 000 000 000 000 000 003 388 131 789 017 201 356 273 290 002 718 567 848 205 566 406 25
590 295 810 358 705 651 712	69	0.000 000 000 000 000 000 001 694 065 894 508 600 678 136 645 001 359 283 924 102 783 203 125
1 180 591 620 717 411 303 424	70	0.000 000 000 000 000 000 000 847 032 947 254 300 339 068 322 500 679 641 962 051 391 601 562 5
2 361 183 241 434 822 606 848	71	0.000 000 000 000 000 000 000 423 516 473 627 150 169 534 161 250 339 820 981 025 695 800 781 25
4 722 366 482 869 645 213 696	72	0.000 000 000 000 000 000 000 211 758 236 813 575 084 767 080 625 169 910 490 512 847 900 390 625

INDEX

Note: The page number(s) of the main reference(s) is underscored.

INDEX (cont)

INDEX (cont)

INDEX (cont)

INDEX (cont)

INDEX (cont)

For more detailed information...

CONTACT YOUR NEAREST HONEYWELL REPRESENTATIVE

or write directly to: Honeywell Inc., Marketing Department,
13350 U. S. Highway 19, St. Petersburg, Florida
Phone 525-1121

Honeywell

MILITARY PRODUCTS GROUP SALES OFFICES

ALBUQUERQUE, New Mexico 87110. . . . . 3301 Carlisle Blvd. N.E., 505/345-1656
ARLINGTON, Virginia 22209. . . . . 1801 North Moore Street, 703/524-8200
ATLANTA, Georgia 30324. . . . . 500 Plaster Ave. N.E., 404/875-9561
BOSTON, Massachusetts 02135 . . . . . 1230 Soldiers Field Road, 617/254-8730
DALLAS, Texas 75206. . . . . P.O. Box 4776, 6000 North Central Expressway, 214/363-5441
DAYTON, Ohio 45404 . . . . . 1164 Stanley Avenue, 513/461-4480
DETROIT, Michigan 48227 . . . . . P.O. Box 3901, Strathmoor Station, 313/834-6020
EL SEGUNDO, California 90245 . . . . Aerospace Center, 650 N. Sepulveda Blvd. 213/322-6957
HOUSTON, Texas 77006. . . . . Suite 213, 1730 F.M. 528, 713/488-0534
HUNTSVILLE, Alabama 35801 . . . . . Suite 73 Holiday Office Center
3322 Memorial Parkway So., 205/881-2711
LONG ISLAND CITY, New York 11101. . . . . 24-30 Skillman Avenue, 212/392-4300
ORLANDO, Florida 32802. . . . . P.O. Box 2369, 2545 Diversified Way, 305/422-5134
PHILADELPHIA, Pennsylvania 19132. . . . . 3345 West Hunting Park Avenue, 215/226-2400
ROME, New York 13446. . . . . 1333 East Dominick Street, 315/336-4770
ST. LOUIS, Missouri 63110 . . . . . 2146 Hampton Avenue, 314/647-6100
RIALTO, California 92406. . . . . 959 E. Mariposa Dr., 714/875-1879
SAN DIEGO, California 92110. . . . . 1515 Morena Boulevard, 714/276-4665
SANTA CLARA, California 95050 . . . . . 1041 DiGiulio Avenue, 408/241-5650
SEATTLE, Washington 98109. . . . . 401 Pontius Avenue N., 206/682-5610
SHALIMAR, Florida 32579. . . . . P.O. Box 757, 305/651-2191
UNION, New Jersey 07083 . . . . . P.O. Box 161, 201/688-9000
WASHINGTON, D.C. 20006 . . . . . Military and Space Sciences Department
1701 Pennsylvania Ave., N.W., 202/298-6040
FRANKFURT/MAIN, 6, Germany Theodor-Heuss-Allee-112, 770661
LINKOPING, Sweden. . . . . Swedish Project Office, Honeywell, St. Larsgatan 30, 013/36700
LONDON, England. . . . . Honeywell Controls, Ltd., Honeywell House
Great West Road, Brentford, Middlesex, Atlas 9191
PARIS, 2e, France. . . . . 30 rue Notre Dame des Victoires, Room 5C, CEntral 39.92
STOCKHOLM 42, Sweden. . . . . Honeywell AB, Elektravagen 5, 18.01.00
TOKYO, Japan. . . . . Yamatake-Honeywell Keiki 2-6 Marunouchi Chiyoda-ku, 28.67.51
TORONTO, Ontario, Canada Honeywell Controls Ltd., Vanderhoof Ave., Leaside, 416/421-8400