3600

Control Data® 3600 Computer System Reference Manual

3600

Control Data® 3600 Computer System
Reference Manual

Record of Revisions

REVISION	NOTES
D	Manual revised; obsoletes all previous editions

Pub. No. 60021300
July, 1964

Address comments concerning this manual to:

Control Data Corporation
Technical Publications Department
4201 North Lexington Avenue
St. Paul, Minnesota 55112

Or use comment sheet located in the rear of this manual.

CONTENTS

FIGURES

TABLES

PREFACE

This manual provides information for the machine-language use of the 3600 system (computer as well as peripheral equipment). Its intention is to describe the capabilities of the hardware. Options and constraints for programming are noted. Programming examples are given to illustrate how instructions perform.

Other than using COMPASS mnemonics to abbreviate titles of instructions, no software systems are used in describing instructions. A number of programming languages and special purpose programs are now available or are being developed for the 3600 computer. Included are an operating system (SCOPE), an assembly language (COMPASS), FORTRAN, COBOL, ALGOL, and SIMSCRIPT compilers, a SORT program, a linear programming system, and nuclear codes. Brief descriptions of these systems are included in the Appendix section. Reference manuals describing software systems in detail are available for all systems which are in current operation.

Programming information for all peripheral equipments available for use with the 3600 system is available in a separate volume; 3000 Series Computer Systems - Peripheral Equipment Reference Manual, Publication Number 60108800.


3600
3600

CHAPTER I
BASIC SYSTEM DESCRIPTION

The CONTROL DATA* 3600 is a solid-state, stored program, general-purpose digital computing system. With large storage capacity and exceedingly fast data transmission and computation speeds, the 3600 computing system is efficient in large-volume data processing and in solving large-scale scientific problems.

The 3600 system incorporates features of the CONTROL DATA 1604 Computer to provide program compatibility with this machine. To provide greater flexibility and capabilities in systems applications, the 3600 system is constructed in functional modules. With the several available options, a variety of systems configurations is possible.

3600 SYSTEM CHARACTERISTICS

- Stored-program general purpose computer
- Parallel mode of operation
- Single address logic
- 51-bit storage word (48 bits of data, 3 parity bits)
- Six 15-bit index registers
- Indirect addressing
- Magnetic core storage (options available)
  - 32,768 51-bit words in two independent 16,384 word units
- Input/Output
  - Transmission of 48-bit words (12-bit bytes)
  - Four separate bi-directional input/output channels (options available)
- System interrupt
- Flexible repertoire of instructions
  - Fixed point arithmetic (integer and fractional)
  - Floating point arithmetic (single and double precision)
  - Logical and masking operations
  - Variable length data manipulation
  - Block transfers
  - Indexing
  - Storage searching
  - Bit sensing
- Binary arithmetic
  - Modulus $2^{84}-1$ (one's complement) for single precision operations
  - Modulus $2^{84}-1$ (one's complement) double precision floating point operations
- Completely solid-state
  - Diode logic
  - Transistor amplifiers
  - Ready access to circuits
- Console includes:
  - Register contents displayed in octal
  - Electric typewriter
- Inter-computer communication
  - 3604 ⟷ 160/160-A
  - 3604 ⟷ 3604
  - Satellite operations

*Registered Trademark of Control Data Corporation.

BASIC 3600 SYSTEM

The basic 3600 system consists of a central computer, an input/output section, magnetic core storage, and a console (figure 1-1). Over-all system operation depends on the integral operation of these elements. The system is divided into several modules to provide flexibility and other advantages of modularity.

Computation Module

The 3604 computation module (throughout this manual the term "computer" may also be used to refer to the 3604) performs the arithmetic and logical operations required by the instructions of a stored program.

The computation module also generates the commands necessary to initiate input and output operations in the communication module.

Communication Module

Two modules govern input/output operations in the 3600 system. These modules are the 3602 communication module and the 3606 data channels (four 12-bit data channels are standard with the basic 3600 system).

The 3602 acts as an access path between the data channels and storage and the computer. Information from the 3604 to initiate input/output activity is conveyed by the 3602 to the data channels. The 3602 also sequentially examines the data channels for storage requests, increments storage addresses to which storage references will be made, and performs parity operations.

The data channels govern the input/output operations initiated by the computer. All controls and registers necessary for communication between the external equipments and storage or the computer are located in the data channels.

Storage Module

The basic 3600 system provides high-speed, random access magnetic core storage for 32,768 51-bit words (48 bits of data, 3 parity bits). This basic storage module consists of two independent 3609 storage units, each with a capacity of 16,384 words. Two 3609 storage units arranged in a module are termed a 3603. These two units operate together during the execution of a stored program and are independently addressable.

Console

Included in the basic 3600 computing system is a 3601 operator and maintenance console. The console contains all the controls and indicators necessary to operate the 3600 system.

An electric typewriter on the console serves as a direct entry keyboard and an output type-printer.

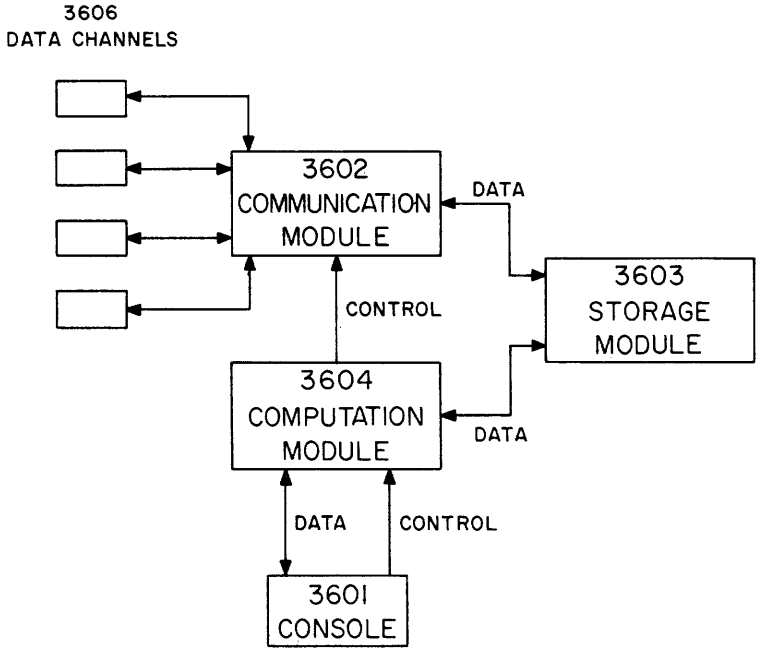


Figure 1 - 1. Basic 3600 Computing System

OPTIONS

For greater systems capability, the 3600 computing system may be expanded with several available options:

1) An additional 3604 computation module may be added to provide greater arithmetic and control capabilities.

2) Additional magnetic core storage may be added in the following configurations:

   a) Up to fourteen 3609 storage modules (each providing 16,384 words of magnetic core storage) or

   b) Up to seven 3603 storage modules.

   The expanded system may thus have a maximum of sixteen 3609 storage modules or eight 3603 storage modules. Expanding the system storage capability to this maximum provides storage for a total of 262,144 51-bit words.

3) To provide additional input/output capability, several options exist:

   a) The communication module supplied with the basic 3600 may be expanded to include four additional data channels. (The associated channel control in the communication module is designed to accommodate additions.) Thus, a communication module may provide a total of eight bi-directional data channels.

   b) Up to three communication modules may be added to the system, providing a total of four modules. Each such module may be supplied with up to eight channels.

If all data channel options are exercised and included in the four communication modules, input/output capability may be expanded to 32 bi-directional data channels.

   c) Three different data channels are available:

      1) Standard 12-bit data channel; handles 12 bits of data at a time and assembles four of these 12-bit bytes to complete a 48-bit word.

      2) Special 24-bit data channel; handles 24 bits of data at a time and assembles two of these 24-bit bytes to complete a 48-bit word (available for special applications only).

      3) Special 48-bit data channel; handles 48 bits of data which are transmitted directly as a 48-bit word (available for special applications only).

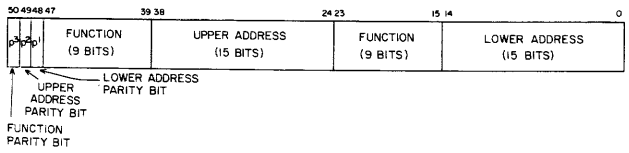
   d) For data transmission to and from the computing system, several external equipments may be attached to the data channels. Examples of these equipments are: magnetic tape control, magnetic disc file, card reader, card punch, and high-speed printer.

CHAPTER II
STORAGE MODULE

The 3609 magnetic core storage module provides high-speed, random access storage for 16,384 words. The 3609 storage module consists of the storage elements themselves, five access channels, a priority section, an address parity checker, and the circuitry for addressing the storage elements. Two 3609 storage modules located within the same cabinet are termed a 3603 storage module.

STORAGE WORD

A storage word may be two 24-bit instructions, a single 48-bit instruction, a 48-bit data word, or half a 96-bit data word. Three parity bits are appended to each 48-bit word; thus a storage word is 51 bits in length. The format of a typical storage word is diagrammed below.

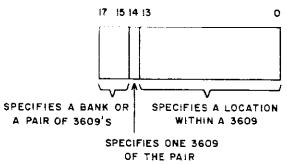


The storage word is divided into three portions: (1) a 15-bit lower address, (2) a 15-bit upper address, and (3) an 18-bit function portion, distributed in the storage word as diagrammed. A parity bit accompanies each of these portions when the word is stored. The parity bit (P1) associated with the lower address portion is placed in bit 48 of the storage word, parity bit P2 (upper address) is placed in bit 49, and parity bit P3 (function) is placed in bit 50.

When part of the word or the entire 51 bits is read from storage, the appropriate parity bit(s) accompanies the word to the 3604 or 3602 where it is checked for parity (refer to chapter VI).

STORAGE ADDRESSING

The location of each word in storage is identified by an assigned number (address). An address consists of 18 bits interpreted as shown.



A pair of 3609 modules (typically, these are in the same cabinet, but need not be) make up a storage bank which can store 32,768 words. The addresses within a bank (e.g., bank 2) are assigned as follows: one 3609 has locations with addresses 2 00000_8 through 2 37777_8, and the other 3609 has locations with addresses 2 40000_8 through 2 77777_8.

STORAGE ACCESS

Each 3609 storage module has five access channels to permit storage requests from five sources. For example, four communication modules and one computation module may have access to a 3609 (figure 2-1).

Each of the five access channels of a 3609 is assigned a designation number by manually setting four binary switches associated with that access channel.

The equipment requesting access to storage transmits a request and an 18-bit address to storage. The upper 4 bits of the 18-bit address select or address a particular 3609. Thus, only the 3609 which has a channel switch designation matching that specified by the upper 4 address bits recognizes the memory request. The lower 14 bits of the 18-bit address specify a particular location within the 3609.

The storage request accompanying the storage address indicates whether a read or write reference is to be performed.

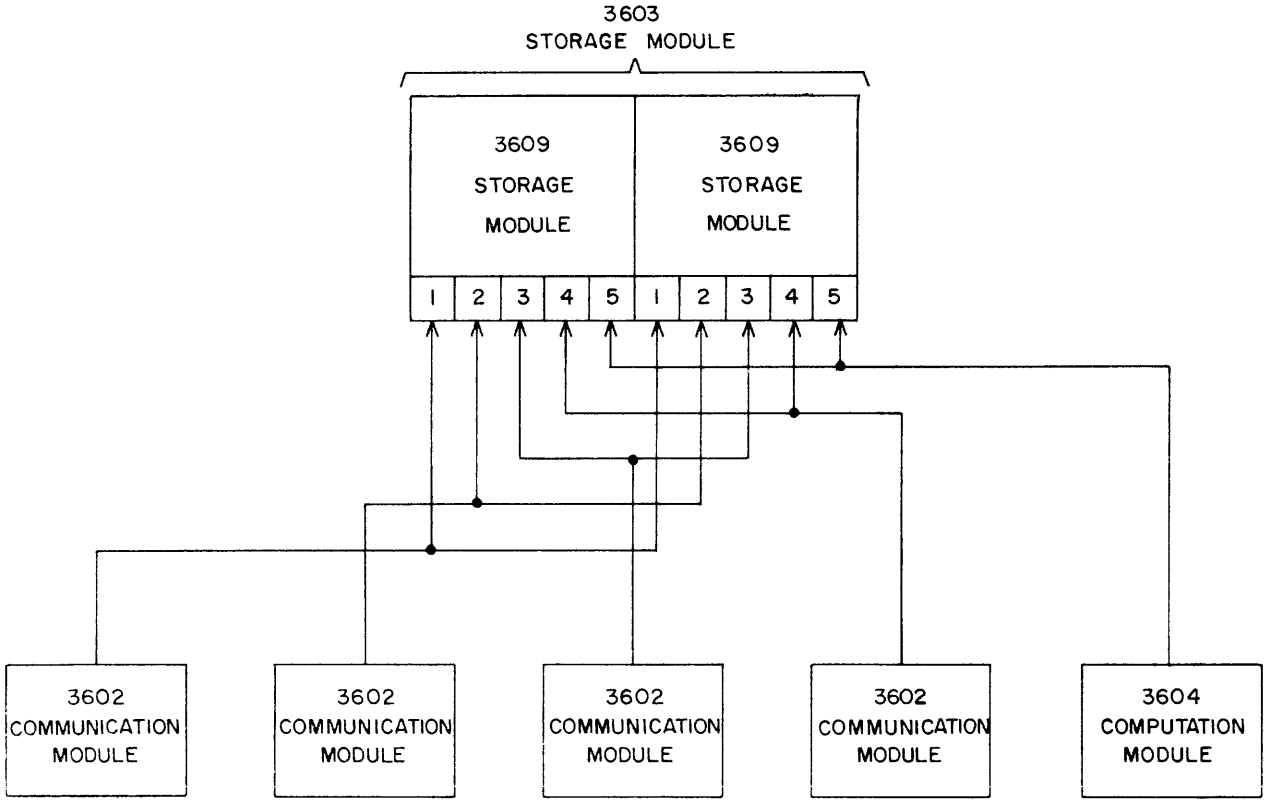


Figure 2-1. Typical Storage Access Channel Hook-up

PRIORITY SECTION

Each 3609 storage module contains a priority section which scans the five access channels for storage requests. This scanner successively examines each channel to insure that each channel has access to storage once per scan cycle, thus preventing simultaneous storage requests. If the channel being examined by the scanner has recognized a memory request, the scanner halts and the memory reference via that channel is initiated. If any other units are requesting storage access via their channels, these units must wait until the current storage request is completed and the scanner advances to their particular access channel.

ADDRESS PARITY CHECKING

The unit requesting access to storage transmits a parity bit along with the storage address. This parity bit and address are checked in the 3609. If a transmission error has occurred, the 3609 returns a Parity Error signal to the equipment requesting access. (For a comprehensive explanation of parity, refer to chapter VI.)

CHAPTER III
COMPUTATION MODULE

The 3604 computation module performs calculations and processes data in a parallel binary mode through the step-by-step execution of individual instructions. The instructions and data are stored in the 3609 storage module(s).

LOGICAL DESCRIPTION

Functionally, the 3604 may be divided into an arithmetic section and a control section.

Arithmetic Section

The arithmetic section performs the arithmetic and logical operations necessary for executing instructions. It consists primarily of several operational registers. The operational registers are described below. Table 3-1 lists the arithmetic properties of the registers.

A Register

Nearly all arithmetic and logical operations use the 48-bit A (Arithmetic) register. The contents of this register may be shifted right or left, separately or in conjunction with the Q register. In certain conditional instructions, the A register holds control quantities which govern operations. In some arithmetic operations, the A register operates as a 49-bit register. Only 48 bits are necessary, however, to display the final result.

Q Register

The 48-bit Q (Auxiliary Arithmetic) register assists the A register in performing arithmetic and logical operations. The contents of the Q register may be shifted right or left, separately or in conjunction with the A register. Q may also be used with the A register to form a double length register, AQ or QA. In addition to assisting the A register, certain instructions reference the Q register directly. In some arithmetic operations, the Q register operates as a 49-bit register. Only 48 bits are necessary, however, to display the final result.

P Register

The 15-bit P register functions as a program address counter. The P register holds the address of each program step. After executing the instruction (or instructions) contained in the program step, the antity in P is advanced to the address of the next instruction. The amount by which P is advanced is determined by the type of exit the instruction takes:

1) Normal exit - the P register is advanced by one.
2) Skip exit - the P register is advanced by two.
3) Jump exit - the P register is set to the quantity specified by the execution address of the jump instruction. If the instruction is a return jump, the contents of P are stored before executing the jump, permitting a return to the program sequence after the jump is made.

Since the P register is a two's complement additive register, it can generate storage addresses in sequence from 00000 to 77777_8. When a count of 77777_8 is reached, the next count in P reduces its value to 00000. (Note that in generating storage addresses by adding the contents of an index register to a base quantity, address 77777_8 cannot be reached. Refer to the section on Address Modification Modes.)

Table 3-1. Arithmetic Properties Of Registers

Register	No. of Stages	Modulus	Complement Notation	Arithmetic	Result
A Register	48	$2^{48}-1$	one's	subtractive	signed*
Q Register	48	$2^{48}-1$	one's	----------	signed
P Register	15	2^{15}	two's	additive	unsigned
Index Registers	15	$2^{15}-1$	one's**	----------	--------
Time Register	27	2^{27}	two's†	additive	unsigned

D Register

The 48-bit D (Flag) register is an auxiliary register which may be:

1) Specifically referenced in the D Register Jump, Inter-Register, Register Jump and Bit Sensing instructions.

2) Used to provide temporary storage for a quantity while operations proceed in other registers. This eliminates an additional storage reference.

U Register

The 48-bit U (Program Control) register holds the program step while it is being executed. All operations necessary to execute an instruction are governed by the contents of this register.

B^1 - B^6 (Index Registers)

Six 15-bit index registers may:

1) Hold quantities used as address modifiers.

2) Hold control quantities for certain instructions (e.g., Search, Locate List Element, etc.).

The index registers may also be explicitly referenced by certain instructions (refer to Repertoire of Instructions section).

Bounds Register

The 37-bit Bounds register is a memory and jump lock-out. The lower 18 bits of the Bounds register hold an 18-bit lower bound address (15-bit storage address plus 3-bit bank address). The upper 19 bits hold a 15-bit storage address, a 3-bit bank address and a single upper bit which, together, define the upper bound address.

Bounds checking on jump and memory addresses occurs only if the interrupt system is active and if the Bounds bit in the Interrupt Mask register is set to "1". If these conditions are present, the following operations are not permitted, and will cause an interrupt if attempted:

1) A jump to an address outside the bounds defined by the upper and lower bound addresses. (If a jump is attempted out of bounds, it will not be effected, and interrupt will occur.†† Upon return to the main program after processing the interrupt, the next instruction will be executed.)

2) Writing in an address out of bounds.

3) Reading an instruction from out of bounds.

The only permissible use of an address out of bounds is to read its contents as an operand except when:

* The result of an arithmetic operation in A satisfies $A \leq 2^{47} - 1$ since A is always treated as a signed quantity. When the result in A is zero, it is always represented as 000 . . . 000 except when 111 . . . 111 is added to 111 . . . 111 or when 000 . . . 000 is subtracted from 111 . . . 111. In these cases, the result is 111 . . . 111 (negative zero).

** Though the index registers have no arithmetic capabilities themselves, address modification using the index registers is performed modulus $2^{15} - 1$ (one's complement). If two's complement mode is selected, address modification is performed modulus 2^{15} (two's complement).

† In selecting Real Time Clock interrupt, the addition of a time value (in milliseconds) to the count held in the Time register (to be placed in the Time Limit register) is performed in one's complement notation. In the add operation, the operands are treated as 48-bit operands with the upper 21 bits zeros. Therefore (even though the addition is performed in one's complement arithmetic) the next count after reaching the capacity of the Time Limit register ($2^{27} - 1$) becomes all zeros.

†† The following actions occur on jump instructions before Bounds interrupt occurs:
1) No register contents are changed in the 22, 23, 63.0, 63.1, 75, and 76 instructions; the only action is the stop during the 76 instruction.
2) All functions of the 55, 62, 63.4, 63.6, 77.4, 77.5, and 77.6 instructions are performed except the actual jump (e.g., the contents of B^b are decremented in the 55 instruction.).

a.) the Bounds Fault bit in the Interrupt Mask register is set,

b.) the 1604 Mode bit in the Interrupt Mask register is set, and

c.) the Interrupt system is active.

When the above three conditions are met, any read reference out of bounds will result in an interrupt.

Addresses which may be referenced are such that $B_L \leq S$ and $B_U > S$ (where S is the storage address). Setting the upper bit of the upper bound address effectively removes the upper bound. This permits writing into address 777777.

When the contents of the Bounds register are transmitted into a 48-bit register via the Inter-Register instruction, its contents are distributed as follows in that register:

a.) lower bound address placed in bits 0-17.
b.) upper bound address placed in bits 24-42.

Instruction Bank Register

The 3-bit Instruction Bank register holds the designation of the storage bank in which instructions are located. This storage bank remains selected until explicitly changed.

Operand Bank Register

The 3-bit Operand Bank register holds the designation of the storage bank in which operands are located. This storage bank remains selected until explicitly changed.

Interrupt Register

Each interruptible condition in the system is connected to a particular bit position of the Interrupt register. The lower bit positions (16) detect internal interrupt conditions such as overflow, divide fault, exponent fault, etc. The upper bit positions (32) are interrupt lines coming from each of the 32 possible communication channels.

Interrupt Mask Register

This register enables testing of external interrupt lines and internal conditions. The bit positions of this register match the Interrupt register. The interrupt lines are always tested because the upper Mask bits are forced to "1's". Interrupt occurs on an internal condition if the matching bit of the lower Interrupt Mask register is set to "1". The Inter-Register or Bit Sensing instructions may be used to set the Interrupt Mask register bit.

Product Register

The Product register contains the bit-by-bit logical product of the Interrupt register and the Interrupt Mask register (refer to the Interrupt section).

Shift Count Register

The 7-bit Shift Count register (SCR) holds the shift count whenever a shift operation is performed. On a right shift operation, the count held in the SCR is the true right shift count. On a left shift operation, however, the count held in the SCR is 140_8 minus the true left shift count (since the shift network can only shift right). The right shift performed is thus equivalent to left-shifting by a true left shift count.

In addition to the Shift instructions, several other instructions use shifting operations (and thus use the SCR), or in some way tamper with its contents in their execution. These instructions are listed in table 4-2, page 4-9.

A typical use of the SCR (in instructions other than Shifts) is to govern normalize operations in the floating point instructions. At the completion of a floating point instruction, the contents of the SCR depend on whether normalized or un-normalized arithmetic was selected before the operation.

If normalized arithmetic was selected (either by augmenting a floating point instruction or by executing the instruction in the normal manner), the SCR holds the following:

1) If a right shift was necessary to normalize the number, a "1" is held in the register. In single precision operations requiring a right shift, the lowest bit of the A register is shifted into the uppermost bit of the Q register.

2) If a left shift was necessary to normalize the number, the SCR holds 140_8 minus the true left shift (i.e., true left shift is the number of places the coefficient must be shifted left to normalize the number.).

If un-normalized arithmetic was selected (by augmenting a Floating Point instruction), the Shift Count register holds the following:

1) Even though un-normalized arithmetic was selected, if a right shift was necessary to normalize the number, a "1" is held in this register, and the number is normalized. In this case, normalizing is necessary to pack the exponent into the floating point word.

2) Otherwise, the SCR holds zeros at the end of the operation, and the number is not normalized.

If the Shift Count register is to be examined, the Inter-Register, Register Jump, or Bit Sensing instructions may specifically address this register. The Shift Count register must, however, be examined by one of these instructions before executing an instruction which may destroy its contents. Refer to table 4-2, page 4-9.

The Inter-Register and Register Jump instructions treat the Shift Count register as a 15-bit register, with the upper bits zeros.

Miscellaneous Mode Selections Register

The 15-bit Miscellaneous Mode Selections register holds a binary indication of the status of several switches and internal operating conditions. The condition associated with each of the register's bit positions is listed in the table below.

The Miscellaneous Mode Selections register may be examined by the Inter-Register, Register Jump, or Bit Sensing instructions. Using these instructions, the programmer may determine a current operating mode or use the Sense switches to control program flow.

Bit	Switch or Mode	Condition if Bit = "1"
00	Console Sense Switch 1	The programmer may use these switches to flag internal conditions. The switches may then be checked by machine instructions and used to control program flow.
01	Console Sense Switch 2	
02	Console Sense Switch 3	
03	Console Sense Switch 4	
04	Console Sense Switch 5	
05	Console Sense Switch 6	
06	Two's Complement Mode	All index arithmetic is performed in two's complement arithmetic.
07	I/O Illegal Instruction	If this bit = "1", the I/O Illegal Instruction FF is set. If the Illegal Instruction bit in the Interrupt Mask register is set, interrupt will occur when an Input/Output instruction (74.0-74.6) is to be executed (refer to the Internal Function instruction).
08	Interrupt Exit	Upon return to the main program after processing an interrupt condition, the upper instruction will be executed if this bit is a "0"; the lower instruction will be executed if this bit is a "1".
09	Interrupt Active	Interrupt system is active. Interrupt will occur if the condition arises and the appropriate Interrupt Mask register bit is set.
10	Selective Stop Switch 1	Stop switch 1 is set.
11	Selective Stop Switch 2	Stop switch 2 is set.
12	Selective Stop Switch 3	Stop switch 3 is set.
13	Card Input Mode/Normal Switches at Console	Reader is selected for operation if this bit is a "1"; i.e., Card Input Mode switch has been pressed. If a "0", the console typewriter is selected (Normal switch has been pressed).
14	Negate BCD Conversion	Negate BCD Conversion FF is set, inhibiting BCD conversion.

Time Register

The 27-bit Time register holds the Real Time Clock count. Its contents are incremented by one each millisecond. This register is addressable as an operand (i.e., its contents may be read, but writing into this register is not permitted) via the Inter-Register, Bit Sensing, or Register Jump instructions (refer to section on Real Time Clock).

Time Limit Register

The 27-bit Time Limit register may be set to hold the Real Time Clock count plus a given time in milliseconds. Setting the Time Limit register may be accomplished by the Inter-Register, Bit Sensing, or Register Jump instructions (refer to section on Real Time Clock).

Control Section

The control section of the 3604 directs the operations required to execute instructions, and establishes the timing relationships needed to perform these operations in the proper sequence. It also sends to the communication module the preliminary commands necessary to begin the processing of input/output data.

The control section acquires an instruction from storage, interprets it, and sends the necessary commands to other sections. A program step may be a single 48-bit instruction or a pair of 24-bit instructions which together occupy a single storage location as a 48-bit word.

The program address counter, P, is a two's complement additive register. It provides program continuity by generating in sequence the storage addresses which contain the individual program steps. Usually, at the completion of each program step, the count in P is advanced by one to specify the address of the next program step.

The Program Control register, U, holds a program step while it is being executed. If the program step is a pair of 24-bit instructions, the upper instruction is executed first followed by the lower instruction.

After executing an instruction, a half exit, full exit, skip exit, or jump exit is performed. A half exit always allows the lower instruction of a program step to be executed. A full exit advances the count in P by one and executes the upper instruction of the new program step at the address specified by the contents of P. A skip exit advances the count in P by two, skipping the next sequential program step. A jump exit allows a new sequence of instructions to be executed; the storage location of the new instruction is specified by the execution address of the jump instruction. In this case, the execution address is entered into P and specifies the starting location of a new sequence of program steps.

Storage Bank Selection

The 3600 system may be expanded to include up to eight 3603 storage modules (refer to Options section).

Expanding the system to include several storage modules requires the provision for addressing any specific module.

The storage modules are numbered 0, 1, 2, 3, ... 7. Two 3-bit bank registers, the Operand Bank register and the Instruction Bank register, may be manually manipulated from the console and may also be addressed as operands in the Inter-Register, Register Jump and Bit Sensing instructions. Also, a 3-bit bank address designator contained in the instruction itself specifies the bank to which the storage reference is to be made. Thus, a complete storage address is an 18-bit composite address (bank address plus execution address).

Once an instruction bank selection is made, all instructions will be read from that storage bank until one of the following instructions is executed:

1) An Unconditional Bank Jump (63.0 s=0)
2) An Unconditional Bank Jump to Lower (63.1)
3) A 48-bit Return Jump (63.0 s=1)

Once an operand bank selection is made, all operands will be read from that storage bank until one of the following instructions is executed:

1) A 48-bit Return Jump
2) Any Bank Jump
3) Execute
4) Search Order
5) Transmit
6) Locate List
7) Augment

INTERNAL OPERATING MODES

Trace Mode

The 3604 computer may be operated in a special mode called Trace. The Trace mode is selected by setting the Trace Mode bit in the Interrupt Mask register. In Trace mode, all jump instructions (in which the jump condition is met) are trapped in interrupt before executing the jump (assuming the interrupt system is active.)*

The contents of the Program Address register are automatically stored when the interrupt subroutine is entered to permit return to the main program. After interpretively executing the jump instruction in the interrupt subroutine, the main program resumes with the instruction which was about to be executed when interrupt occurred unless the interrupt routine provides for a return elsewhere.

1604 Mode

Most programs written for the 1604 computer can be executed on the 3600 system through the 1604 compatibility program package. Except for five cases, operation codes in the 1604 and 3604 designate identical instructions. Codes 00, 62, 63, 74, and 77 designate different instructions in the two computers. To aid in achieving compatibility of 1604 programs with the 3600 system, the latter may be placed in 1604 mode, which detects the occurrence of any of these codes and causes an interrupt. The interrupt routine then can interpretively execute the instruction.

The 1604 mode of operation is selected by setting the 1604 Mode bit in the Interrupt Mask register and activating the interrupt system. Setting this bit sends a Negate BCD Conversion signal to external equipments, indicating that 1604 mode is in operation.

Some external equipments associated with the 3600 system continue normal operation whether or not 3600 (normal) or 1604 mode is in operation. Others perform the following operations in transmitting data into or out of the 3600 system (cases are defined for normal 3600 mode and for 1604 mode):

1) 3600 (Normal) Mode with 362X:

   a) In reading information from magnetic tape, the 362X magnetic tape controller converts IBM 704 external BCD to internal BCD (i.e., if the 5th level is "1", complement the 6th level - for example, on tape, the code for the character "A" (61) is converted to 21).

   b) In writing information on magnetic tape, internal BCD is converted to external BCD.

2) 1604 Mode with 362X:
   No conversion is performed while in 1604 mode; the 3600 performs like the 1604 computer. The program should provide for reading and writing in external BCD (even parity).

3) 3600 (Normal) Mode with High-Speed Printer:
   Data transmitted to the printer in 3600 (normal) mode is converted from internal to external BCD.

4) 1604 Mode with High-Speed Printer:
   In 1604 mode, the program should provide for transmitting data in external BCD format.

In transmitting data into or out of the 3600 system, two ways exist for inhibiting BCD conversion:

1) Setting the 1604 Mode bit in the Interrupt Mask register sends a Negate BCD Conversion signal to the external equipments.

2) Executing an Internal Function instruction (77.0 00031 - Select Negate BCD Conversion) transmits a Negate BCD Conversion signal to the external equipments.

To permit BCD conversion:

1) The 1604 Mode bit in the Interrupt Mask register must be clear, and

2) An Internal Function instruction (77.0 00032) Release Selection of Negate BCD Conversion) must be executed.

The Negate BCD Conversion FF, set or cleared by the Internal Function codes listed above, may be examined by sampling the Miscellaneous Mode Selections register. Bit 14 of this register is the Negate BCD Conversion FF. A "1" in this bit indicates that this FF is set, inhibiting BCD conversion.

An internal master clear also clears the Negate BCD Conversion FF and the Interrupt Mask register. Thus, BCD conversion will occur in any input/output operations performed following an internal master clear.

* The following actions occur on jump instructions before Trace Mode interrupt occurs:
1.) No register contents are changed in the 22, 23, 63.0, 63.1, 75, and 76 instructions; the only action is the stop during the 76 instruction.
2.) All functions of the 55, 62, 63.4, 63.6, 77.4, 77.5, and 77.6 instructions are performed except the actual jump (e.g., the contents of B^b are decremented in the 55 instruction.).

Two's Complement Mode

In the normal mode of operation, one's complement arithmetic is used in address modification and all indexing instructions (except ISK and IJP). When Two's Complement mode is selected, two's complement arithmetic is used in all address modification and in the execution of the Increase Index instruction.

Selecting Two's Complement mode permits using storage location 77777_8 as an addressable location through address modification.

Once Two's Complement mode is selected, it remains selected until it is:

1) Cleared by the Internal Function instruction,

2) Cleared by a manual internal master clear, or

3) Temporarily suppressed* by one of the following occurrences:

   a) Exponent arithmetic

   b) One of the scale instructions

   c) One of the product register jump instructions

   d) One of the Search instructions

When Two's Complement mode is cleared via the Internal Function instruction, one's complement arithmetic is again used in address modification and all indexing instructions except Index Skip and Index Jump. These instructions use two's complement arithmetic regardless of whether or not Two's Complement mode is selected.

REAL TIME CLOCK

The Real Time Clock in the 3604 is a free-running, two's complement,** 27-bit counter that advances each millisecond. This counter runs continually as long as power is on.

The clock count (held in the 27-bit Time register which is specified by code 31) may be examined by the Inter-Register, Bit Sensing or Register Jump instructions.

A 27-bit Time Limit register, specified by code 32, may be set or examined by the above three instructions.

The program may select interrupt to occur when the Time register (Clock) and the Time Limit register become equal. The necessary steps to select a Real Time Clock interrupt are outlined below (assume the program wants an interrupt indication after 25 milliseconds have elapsed).

1) Examine clock count in Time register (using one of the three instructions listed above).

2) Set Time Limit register to clock count plus 25.

3) Select interrupt by setting the Real Time Clock bit in the Interrupt Mask register to "1".

4) Activate the interrupt system.

After 25 milliseconds have elapsed, the Time register and Time Limit register become equal. The Real Time Clock bit in the Interrupt register is set when this time occurs.

Interrupt occurs immediately after an advance clock pulse if the Time register and Time Limit register are equal. This precludes setting the two registers equal to force a Real Time Clock interrupt.

---

* Temporarily suppressing Two's Complement mode as in case 3 above does not clear the selection of the mode.

** When the clock reaches its maximum count of all ones, the next count reverts the clock count to all zeros, and the count continues in sequence.

DESCRIPTION OF INSTRUCTIONS

Word Format

A computer word consists of 48 bits and may be interpreted as one 48-bit data word, half a 96-bit data word, a 48-bit instruction, or two 24-bit instructions.

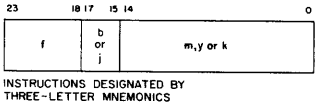
Most instructions designated by three-letter mnemonic codes are 24-bit instructions common to the 1604 computer. These instructions are arranged in a 48-bit word; the higher order 24 bits are called the upper instruction and the lower order 24 bits are called the lower instruction.

Instructions which are not common to the 1604 computer and designated by mnemonic codes of three or four letters, differ in format and in word length (some are 24 bits; others are 48 bits).

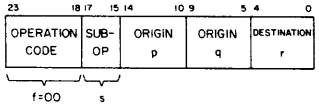
Instruction formats are arranged in four major classes, according to differences in word length and the position of the function code within the format. A typical format from each class is outlined below. Designators used within these formats are explained at the end of this section. For a comprehensive description of instructions, refer to the Repertoire of Instructions section.

Class I

Class I instruction formats are 24 bits in length and have 6-bit function codes, 'f'. All instructions common to the 1604 computer and designated by three-letter mnemonic codes are included in this category. The Inter-Register instruction, not a 1604 instruction, but indicated by a three-letter mnemonic code, is also included.

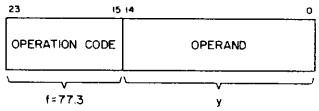


INSTRUCTIONS DESIGNATED BY THREE-LETTER MNEMONICS



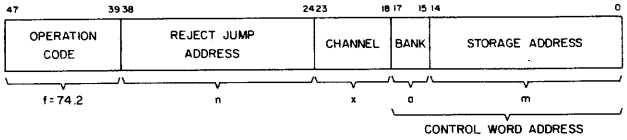
Class II

Class II instruction formats are 24 bits in length and have 9-bit function codes. All instructions in this category are designated by mnemonic codes of three or four letters.



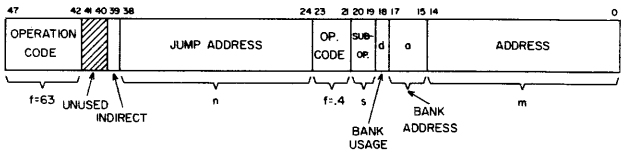
Class III

Class III instruction formats are 48 bits in length and have 9-bit function codes. All instructions in this category are designated by mnemonic codes of four letters.

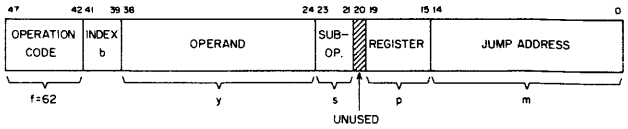


Class IV

Class IV instruction formats are 48 bits in length, and except for the Register Jump instruction, have 9-bit function codes. These formats differ from Class III formats in that the bits of the function code are non-contiguous. Six bits of the function code are in bit positions 42 - 47, and the remaining three bits are in bit positions 21 - 23.



The Register Jump instruction differs from other 48-bit Instructions in Class IV in that a 6-bit function code is used.



DESCRIPTION OF DESIGNATORS

Designators used throughout the Description of Instructions section and in instruction formats are explained below. For specific interpretations of designators, refer to the individual instructions.

Designator		Use
a	First Bank Address	With first storage address, 'm', specifies an 18-bit composite storage location.
b	First Index	Specifies index register (B) used, or whose contents are used in the operation.
c	Connect and Function	Specifies codes used in Connect and Function instructions.
d	Bank Address Usage	Indicates whether or not bank address designators ('a' or 'i') will be interpreted in the operation (if d = 0, not used; d = 1, used).
e	Byte Size	Length of the byte (number of bits of the word) used in operation.
f	Function Code	A 6 or 9-bit code (depending on the operation) which specifies the operation to be performed.
g	Bit Address	Specifies which bit of 48; i.e., address of any bit from 0 - 47.
i	Second Bank Address	With second storage address, 'n', specifies an 18-bit composite storage location.
j	Condition	Conditions operations in jumps and stops.
k	Unmodified Shift Count	Number of shifts to be executed.
K	Modified Shift Count	$[K = k + (B^b)]$; when shift instruction is augmented, becomes $[K = k + (B^b) + (V^v)]$.
m	Unmodified Execution Address (Address One)	Address of operand.
M	Modified Execution Address (Address One)	$[M = m + (B^b)]$; when a 24-bit instruction is augmented or when the execution address is modified in the Execute instruction, becomes $[M = m + (B^b) + (V^v)]$.
n	Address Two	Usually used as a jump address.
o	Offset	Specifies the A or Q register address of leftmost bit receiving the byte.
p	Operand One	Register which holds first operand in Inter-Register and Register Jump instructions.
q	Operand Two	Register which holds second operand in Inter-Register instruction.

Designator		Use
r	Destination	Register to which result is sent after specified operation is complete.
s	Suboperation Code	Specifies one or more suboperations to be performed. (See individual instructions for interpretations of 's'.)
t	Augment Operation (t^0 - t^7)	Specifies one or more of eight specific operations (see Augment instructions).
v	Second Index	Specifies second index register (V) used, or whose contents are used in the operation.
w	Word Count	A 15-bit quantity which specifies the number of words to be processed in a transfer operation.
x	Channel Number	Specifies data channel; also used to specify channel whose status will be read or sensed.
y	Unmodified Operand	Used in execution address portion of instruction; specifies this address will be used as the operand. Specifies a 15-bit comparison quantity in Register Jump instruction. Designates the quantity used as an addend in Add to Exponent instruction.
Y	Modified Operand	[$Y = y + (B^b)$].
z	Instruction Bank	Storage bank in which instructions are located.

ADDRESS MODIFICATION

The portion of the instruction word designated by 'm', 'y', or 'k' is often termed the base execution address. The base execution address may be used as (1) a shift count, 'k', (2) an operand, 'y', (3) an address of an operand, 'm', in storage. The execution address may be modified or left unmodified depending on the index designator. The execution address is modified by adding the contents of the designated index register to the execution address. If left unmodified, the lower-case symbols 'k', 'y', or 'm' are used. If the address is modified, the symbols are capitalized.

The modified shift count is represented by:

1) $K = k + (B^b)$ where:

K = modified shift count
k = unmodified shift count (execution address)
(B^b) = contents of index register b

If the index designator = 0, then K = k.

The modified operand is represented by:

2) $Y = y + (B^b)$ where:

Y = modified operand
y = unmodified operand (execution address)
(B^b) = contents of index register b

If the index designator = 0, then Y = y.

The modified operand address is represented by:

3) $M = m + (B^b)$ where:

M = modified address of operand
m = unmodified address of operand (execution address)
(B^b) = contents of index register b

If the index designator = 0, then M = m.

In some instructions, the contents of a second index register are also used to modify the execution address. These are double indexed instructions. The modified operand address is then represented by:

4) $M = m + (B^b) + (V^v)$ where:

M = modified address of operand
m = unmodified address of operand (execution address)
(B^b) = contents of index register b
(V^v) = contents of index register v

If the index designators are both = 0, then M = m.

The operand, 'y', or the shift count, 'k', may also be modified by the double indexing operation. This operation proceeds exactly as in the above example.

Address Modification Modes

Three possible modes of address modification, as determined by the index designator, are:

1) b = 0 No address modification. The execution address is directly interpreted.

2) b = 1-6 Relative address modification. The execution address is modified as outlined above. One's complement arithmetic is used in determining the modified execution address.

NOTE

Since one's complement arithmetic is used in determining the relative address, address 77776_8 is the highest address which can be generated in the normal manner. For example, modifying execution address 77776 by adding, in one's complement arithmetic, an index value of 1, results in 00000. During address modification, the modified address will equal 77777_8 only if: (1) the unmodified execution address equals 77777_8 and b = 0, or (2) the unmodified execution address equals 77777_8 and $(B^b) = 77777_8$.

3) b = 7 Indirect addressing. A storage reference is made to the location specified by the execution address. The lower order 18 bits of the word at this storage location are interpreted as the index designator 'b' (3 bits) and execution address (15 bits) of the present instruction. The new index designator may refer to any one of the three modes.

NOTE

Indirect addressing occurs in the internal sequence before the instruction is executed. Therefore, the indirect address is obtained from the storage bank currently selected. If, in the execution of the instruction, a change in storage bank is specified, the bank is changed and the operand will be obtained from the new bank.

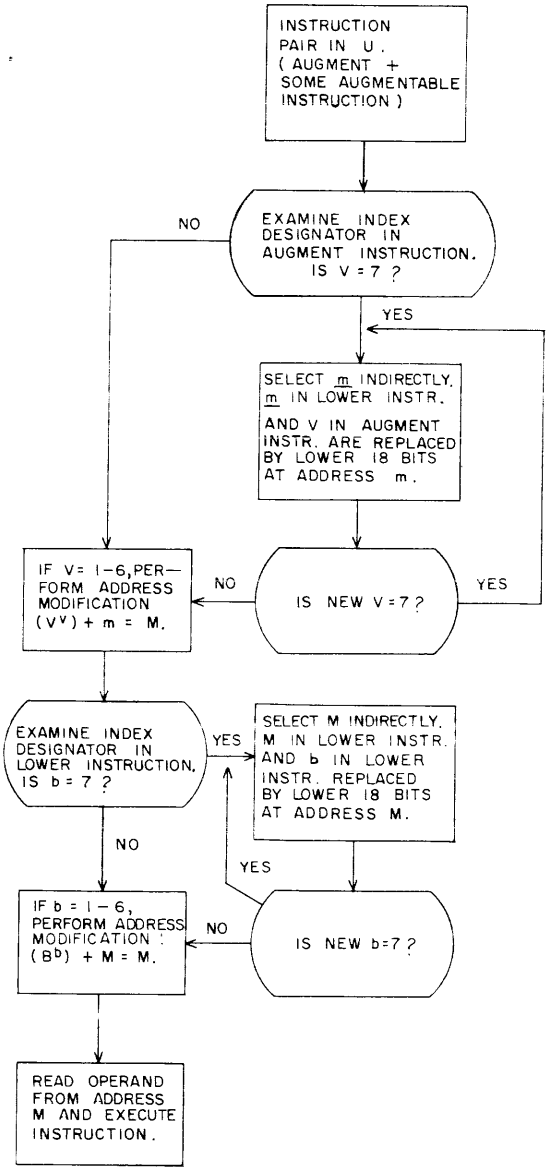


Figure 3-1. Indirect Addressing

When instructions use more than one index designator, indirect addressing may be specified by one or both designators. Since both index designators specifying indirect addressing (b = 7, v = 7) constitutes a special case, this operation is outlined in detail. This double indirect addressing applies only to instructions being augmented by the Augment instructions (see figure 3-1).

Example:

Storage Address	Contents of Storage Location					
	f	v		f	b	m
00005	Augment	7	———	LDA	7	00010
.						
.					b	m
00010	———	———	———	———	7	00011
					b	m
00011	———	———	———	———	3	54321
.						
.						
.						
.					b	m
54322	———	———	———	———	3	60000
.						
.						
60001	———	———	———	———	———	———
$(B^3) = 00001$						

The contents of storage location 00005 are read from storage and placed in U. The second index designator, 'v', is examined first. Since v = 7, indirect addressing is designated. The lower order 18 bits of the storage word at address 00010 are read and interpreted as the new index designator and execution address. Since the new index designator also specifies indirect addressing, the lower order 18 bits of the storage word at address 00011 are read and interpreted. The new index designator (3) indicates the contents of index register 3 are to be added to the execution address (54321_8 + 00001_8 = 54322_8).* The first step is now complete; the lower instruction at address 00005 reads LDA 7 54322 (the index designator of 7 has been preserved during the first step). The first index designator, 'b', is examined. The indirect addressing mode is again specified and the lower order 18 bits of the storage word at address 54322 are read and examined. The contents of the new index register, designated by 3, are added to the execution address, 60000. Upon completing this address modification, the instruction is executed, loading the A register with the contents of storage location 60001.

---

* If the Augment instruction at address 00005 were to specify a change in bank as well as indirect addressing, the sequence of events would be as follows:

1) Obtain relative address indirectly.
2) Modify relative address (if new 'v' = 1-6).
3) Change bank.
4) Obtain relative address in LDA instruction indirectly (in new bank).
5) Proceed

EXECUTION OF INSTRUCTIONS

A program example and a step by step explanation of its execution are outlined below. Instructions from Classes I and IV are used in the example to help explain the use of various designators.

Example:

Storage Address	Contents of Address									
	f		b	m		f		b	m	
00300	LDA		0	00200		ADD		1	00210	
	f	b	v		f		d	a	m	
00301	63	1	2	____	7	____	1	3	03000	

.
.
. $(B^1) = 00100_8$
. $(V^2) = 02000_8$

The storage reference is initiated at address 00300 (the address held in the P register). The 48-bit word is read from address 00300 and entered into U. Computer operation is now dependent upon the interpretation of the 24-bit instruction in the upper half of U.

The operation code, LDA, and the index designator, 0, are translated. The function of the LDA instruction is to load the A register with the contents of the designated storage location. Because the index designator is 0, the execution address is not modified. The translation of the operation code initiates the sequence of commands which executes the instruction, and the operand in address 00200 is loaded into A.

The lower instruction in U is translated. The ADD instruction causes the quantity in storage location M to be added to the contents of the A register. Since the index designator is not 0 or 7, the contents of the index register are added to the execution address to form M ($M = m + (B^b) = 00210_8 + 00100_8 = 00310_8$). The contents of storage address 00310 are added to the contents of the A register, completing the instruction. The contents of the P register are increased by one and the next program step at address 00301 is read from storage, entered into U, and translated.

Address 00301 contains a 48-bit instruction (Execute). The function of the Execute instruction is to execute the instruction (if 48-bit) or instructions (if a pair of 24-bit instructions) at M. M specifies the storage location designated by the execution address plus the contents of index register 1 and the contents of index register 2 (double additive indexing). $M = m + (B^1) + (V^2) = 03000_8 + 00100_8 + 02000_8 = 05100_8$.

After address modification, two designators are examined before the "jump" to address 05100. The 'a' designator, whose value is 3, specifies the "jump" will be to address 05100 in storage bank 3. Designator 'd' indicates (by a "1") that this bank designator will be considered (if 'd' is a "0", 'a' will be disregarded and the "jump" will be within the storage bank currently in use).

The instruction or instruction pair at M is placed in U and execution proceeds in the normal manner. Upon completion of program step 00301, the contents of P are increased by one and the program proceeds.

SYMBOLS

The following symbols are used in the order of Instructions section. For definitions of terms used in this section, refer to the Glossary of Terms.

A	The A register
A_n	The binary digit in position 'n' of the A register
B^b	Designated first index register
D	Auxiliary register; also referred to as the Flag register
Exit (full)	Proceed to upper instruction of next program step
Exit (half)	Proceed to lower instruction of same program step
Exit (skip)	Proceed to upper instruction of next program step plus one
Exit (jump)	Proceed to the address specified by the execution address
LA	Lower address; execution address portion of lower instruction of a program step
Q	Auxiliary Arithmetic register
UA	Upper address
V^v	Designated second index register
()	Contents of a register or storage location
()'	One's complement contents of a register or storage location
()f	Final contents of a register or storage location
()i	Initial contents of a register or storage location
#	A flag to denote the instruction must be located in the upper instruction position of an instruction word
$\vee$	The logical inclusive OR function
$\not\vee$	The logical exclusive OR function
$\wedge$	The logical AND function
$\supset$	The logical implication function
$\equiv$	The logical equivalence function

NOTE

Except for the Double Precision and Truncated Divide instructions (these instructions are selected via one of the augment instructions), the Order of Instructions table which follows outlines unaugmented instructions.

ORDER OF INSTRUCTIONS

Octal Code	Mnemonic Code	Name	Indirect Addressing	Storage* References	Address Modification	Number of Instruction Bits
		Inter-Register Transmission				
00	ROP	Inter-Register	No	0		24
		Full-Word Transmission				
12	LDA	Load A	Yes	1	Yes	24
16	LDQ	Load Q	Yes	1	Yes	24
20	STA	Store A	Yes	1	Yes	24
21	STQ	Store Q	Yes	1	Yes	24
13	LAC	Load A, Complement	Yes	1	Yes	24
17	LQC	Load Q, Complement	Yes	1	Yes	24
63.2	XMIT	Transmit	No	2	No	48
63.2	____	Transmit Augment	No	2r	Yes	48
		Address Transmission				
53	LIL	Load Index (lower)	Yes	1	No	24
52	LIU	Load Index (upper)	Yes	1	No	24
57	SIL	Store Index (lower)	Yes	1	No	24
56	SIU	Store Index (upper)	Yes	1	No	24
61	SAL	Substitute Address (lower)	Yes	1	Yes	24
60	SAU	Substitute Address (upper)	Yes	1	Yes	24
50	ENI	Enter Index	Yes	0	No	24
04	ENQ	Enter Q	Yes	0	Yes	24
10	ENA	Enter A	Yes	0	Yes	24
		Instruction Augment				
77.1	____	Single Precision Augment	Yes	0	Yes†	24
77.2	____	Double Precision Augment	Yes	0	Yes†	24
		Fixed Point Arithmetic				
14	ADD	Add	Yes	1	Yes	24
15	SUB	Subtract	Yes	1	Yes	24
24	MUI	Multiply Integer	Yes	1	Yes	24
26	MUF	Multiply Fractional	Yes	1	Yes	24
25	DVI	Divide Integer	Yes	1	Yes	24
27	DVF	Divide Fractional	Yes	1	Yes	24
77.1-27	____	Truncated Divide	Yes	1	Yes	48

* If indirect addressing is designated, at least one additional storage reference is required.

r Number of repetitions of the operation. † On lower address.

ORDER OF INSTRUCTIONS (Cont'd)

Octal Code	Mnemonic Code	Name	Indirect Addressing	Storage References	Address Modification	Number of Instruction Bits
	Single Precision Floating Point Arithmetic					
30	FAD	Floating Add	Yes	1	Yes	24
31	FSB	Floating Subtract	Yes	1	Yes	24
32	FMU	Floating Multiply	Yes	1	Yes	24
33	FDV	Floating Divide	Yes	1	Yes	24
77.3	ADX	Add to Exponent	No	0	No	24
	Double Precision Floating Point Arithmetic					
77.2-30	DFAD	Floating Add	Yes	2	Yes	48
77.2-31	DFSB	Floating Subtract	Yes	2	Yes	48
77.2-32	DFMU	Floating Multiply	Yes	2	Yes	48
77.2-33	DFDV	Floating Divide	Yes	2	Yes	48
	Address Arithmetic					
11	INA	Increase A	Yes	0	Yes	24
51	INI	Increase Index	Yes	0	No	24
54	ISK	Index Skip	Yes	0	No	24
	Logical					
40	SST	Selective Set	Yes	1	Yes	24
41	SCL	Selective Clear	Yes	1	Yes	24
42	SCM	Selective Complement	Yes	1	Yes	24
43	SSU	Selective Substitute	Yes	1	Yes	24
44	LDL	Load Logical	Yes	1	Yes	24
45	ADL	Add Logical	Yes	1	Yes	24
46	SBL	Subtract Logical	Yes	1	Yes	24
47	STL	Store Logical	Yes	1	Yes	24
	Shifting					
01	ARS	A Right Shift	Yes	0	Yes	24
02	QRS	Q Right Shift	Yes	0	Yes	24
03	LRS	Long Right Shift (AQ)	Yes	0	Yes	24
05	ALS	A Left Shift	Yes	0	Yes	24
06	QLS	Q Left Shift	Yes	0	Yes	24
07	LLS	Long Left Shift (AQ)	Yes	0	Yes	24
34	SCA	Scale A	Yes	0	No	24
35	SCQ	Scale AQ	Yes	0	No	24

ORDER OF INSTRUCTIONS (Cont'd)

Octal Code	Mnemonic Code	Name	Indirect Addressing	Storage References	Address Modification	Number of Instruction Bits
	Replace					
70	RAD	Replace Add	Yes	2	Yes	24
71	RSB	Replace Subtract	Yes	2	Yes	24
72	RAO	Replace Add One	Yes	2	Yes	24
73	RSO	Replace Subtract One	Yes	2	Yes	24
	Storage Test					
36	SSK	Storage Skip	Yes	1	Yes	24
37	SSH	Storage Shift	Yes	2	Yes	24
	Search					
64	EQS	Equality Search	Yes	n	Yes†	24
65	THS	Threshold Search	Yes	n	Yes†	24
66	MEQ	Masked Equality Search	Yes	n	Yes†	24
67	MTH	Masked Threshold Search	Yes	n	Yes†	24
63.4	SEQU	Equality Search	Yes	n	Yes	48
63.4	SMEQ	Masked Equality Search	Yes	n	Yes	48
63.4	SEWL	Search Within Limits	Yes	n	Yes	48
63.4	SMWL	Search Magnitude Within Limits	Yes	n	Yes	48
63.3	LSTU LSTL	Locate List Element	Yes	n	Yes	48
	Jumps and Stops					
22	AJP	A Jump	No	1*	No	24
23	QJP	Q Jump	No	1*	No	24
55	IJP	Index Jump	Yes	0	No	24
75	SLJ	Selective Jump	No	1*	No	24
76	SLS	Selective Stop	No	1*	No	24
63.7	EXEC	Execute	Yes	0 + x	Yes	48
62	RGJP	Register Jump	Yes	0	Yes	48
77.6	DRJ	D Register Jump	No	0	No	24
63.6	NBJP ZBJP	Bit Sensing	Yes	0	Yes	48

† If b = 0, only the word at 'm' is searched; if (B^b) = 0, no search is made.

n Number of words searched.

* Return jump only.

x Number of storage references required by the executed instruction(s).

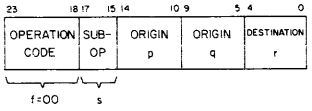
ORDER OF INSTRUCTIONS (Cont'd)

Octal Code	Mnemonic Code	Name	Indirect Addressing	Storage References	Address Modification	Number of Instruction Bits
	Bank Jumps					
63.0	UBJP	Unconditional Jump	Yes	0	Yes	48
63.0	BRTJ	Unconditional Return Jump	Yes	1	Yes	48
63.1	BJPL	Unconditional Jump Lower	Yes	0	Yes	48
63.0	BJSX	Jump and Set Index	Yes	1	No	48
	Variable Data Field					
63.5	LBYT	Load Byte	Yes	1	Yes	48
63.5	SBYT	Store Byte	Yes	2	Yes	48
63.5	SCAN	Scan Byte	Yes	s	Yes	48
	Interrupt					
77.4	MPJ	Main Product Register Jump	No	0	No	24
77.5	CPJ	Channel Product Register Jump	No	0	No	24
	Input/Output					
74.0	CONN	Connect	No	0	No	48
74.1	EXTF	Function	No	0	No	48
74.2	BEGR	Read	No	0	No	48
74.3	BEGW	Write	No	0	No	48
74.4	COPY	Copy Status	No	0	No	48
74.5	CLCH	Channel Clear	No	0	No	48
74.6	IPA	Input to A	No	0	No	24
74.7	ALG	Perform Algorithm	No	0	No	24
77.0	INF	Internal Function	No	0	No	24
77.7	____	Fault	No	0	No	24

s Number of words scanned.

INSTRUCTION REPERTOIRE

Inter-Register Transmission (ROP)(00)



The 24-bit Inter-Register Transmission instruction performs an operation, 's', upon operands 'p' and 'q' and places the result in 'r'. Origin (operand) or destination registers are indicated below according to their octal values of 'p', 'q', and 'r'.

01 - B^1
02 - B^2
03 - B^3
04 - B^4
05 - B^5
06 - B^6
07 - A Lower Address
10 - A Upper Address
11 - Q Lower Address
12 - Q Upper Address
13 - A Full
14 - Q Full
15 - D Full
16 - Bounds Register
17 - Interrupt Mask Register
25 - Operand Bank Register
32 - Time Limit Register

The registers indicated below must be used for operands only.

* {
20 - Interrupt Register
21 - All "0's"
22 - +1
23 - All "1's"
24 - Instruction Bank Register
26 - Shift Count Register
27 - Miscellaneous Mode Selections
30 - P Register
31 - Time Register

The operations which may be performed are listed below according to their octal values of 's'.

s = 0 Register Inclusive OR $p \vee q \rightarrow r$

Forms the logical inclusive OR of operands 'p' and 'q' and places the result in 'r'.

s = 1 Register Exclusive OR $p \veebar q \rightarrow r$

Forms the logical exclusive OR of operands 'p' and 'q' and places the result in 'r'.

s = 2 Register AND $p \wedge q \rightarrow r$

Forms the logical AND of operands 'p' and 'q' and places the result in 'r'.

s = 3 Register Implication $p \supset q \rightarrow r$

Forms the logical implication of operands 'p' and 'q' and places the result in 'r'.

s = 4 Register Equivalence $p \equiv q \rightarrow r$

Forms the logical equivalence of operands 'p' and 'q' and places the result in 'r'.

s = 5 Register Sum $p + q \rightarrow r$

Adds the contents of 'p' to the contents of 'q' and places the result in 'r'. If 'p' and 'q' are negative zero, 'r' is negative zero. Arithmetic overflow fault conditions apply.

s = 6 Register Difference $p - q \rightarrow r$

Subtracts the operand 'q' from 'p' and places the result in 'r'. If 'q' is positive zero and 'p' is negative zero, 'r' is negative zero. Arithmetic overflow fault conditions apply.

s = 7 Register Transmit/Swap (RXT/RSW)

Uses register designators 'q' and 'r' only. The unused 'p' portion of the Inter-Register instruction format becomes a function modifier with the following designator:

Values for the 't' designator are:

0 Swap (q) and (r); do not clear q; do not clear r
1 Swap (q) and (r); do not clear q; clear r
2 Swap (q) and (r); clear q; do not clear r
3 Swap (q) and (r); clear q; clear r
4 Transmit (q) to r; do not clear q; do not clear r
5 Transmit (q) to r; do not clear q; clear r
6 Transmit (q) to r; clear q; do not clear r
7 Transmit (q) to r; clear q; clear r

* All "0's", + 1, and all "1's" are not registers, but are forced operands which may be referenced in an operation.

The clear operation in the Transmit/Swap order functions as follows:

Swap Operations

1) When the selected registers are not A_L, A_U, Q_L, or Q_U, the registers are automatically cleared while the contents of the registers are being transmitted to their respective destinations.

2) When one or both of the selected registers is A_L, A_U, Q_L, or Q_U, clearing is not automatic, but depends on the function specified by 't'. If a clear is specified, that register of which A_L or A_U is a part (the A_{full} register) or that register of which Q_L or Q_U is a part (the Q_{full} register) is cleared before the new quantity is entered. If no clear operation is specified, a replace operation is effected.

Transmit Operations

1) When the selected registers are not A_L, A_U, Q_L, or Q_U, the destination register is automatically cleared before the new quantity is entered. The contents of the source register remain unchanged unless a clear function is specified for the source register.

2) When one or both of the selected registers is A_L, A_U, Q_L, or Q_U, clearing the destination register is not automatic, but depends on the function specified by 't'. If a clear operation is specified on the:

   a) Destination register, the full register is cleared before the new quantity is entered. If no clear is specified on the destination register, a replace operation is effected.

   b) Source register, the specified register or partial register is cleared after transmitting its contents to the destination register. If no clear is specified on the source register, its contents remain unchanged.

Performing the logical operations listed above (s = 0-6) with any of the four possible combinations yields the results indicated in the following table.

Logical Combinations

p	q	AND $p \wedge q$	OR $p \vee q$	Excl. OR $p \veebar q$	Impl. $p \supset q$	Equiv. $p \equiv q$
1	1	1	1	0	1	1
1	0	0	1	1	0	0
0	1	0	1	1	1	0
0	0	0	0	0	1	1

Operations performed on designated register(s) contents follow the rules outlined below:

1) If the destination register is a full register and one or both source operands designate 15-bit quantities, the sign bits are extended when arithmetic operations are performed. No sign extension occurs when logical operations are performed on 15-bit quantities.

2) If the destination register is 15 bits and one or both source operands designate 48-bit quantities, the operation is performed on 48 bits, and the result is taken from the lower 15 bits.

3) In the swap or transfer operations, the sign bit of the source operand is extended if the destination register is 48 bits. In the contrary situation, the lower 15 bits are taken as the source operand.

4) When A_L, A_U, Q_L, or Q_U are used as the destination registers, a replace operation is effected.

5) In the swap or transfer operations, when a clear function is not designated, statements 1 and 2 above do not hold true (i.e., a replace operation is effected).

6) When the Operand or Instruction Bank registers or the Shift Count register are used as operands, the upper bits are always "0's".

If, in executing an Inter-Register instruction, an operation attempts to alter the contents of a register designated by codes 20-31 (except 25), the following occurs:

1) That operation which could alter the contents of registers 20-31 (except 25) is not performed, and

2) The instruction continues to completion. The next instruction is then executed.

This operation does not constitute a fault or an interruptible condition.

Executing the Inter-Register instruction (f = 00, s = xx) with q = 0 or r = 0 results in an interruptible fault condition. If the Illegal Instruction bit in the Interrupt Mask register is set, and the interrupt system is active, interrupt occurs. If this Interrupt Mask register bit is not set, this instruction becomes a pass (do nothing) instruction. The next instruction is then executed.

Full-Word Transmission

1) In Full-Word Transmission instructions, a 48-bit operand or data word is used in executing the instruction.

2) Index registers used when augmenting the Transmit commands are fixed. For example, index register 1 (designated by B^1) must be used to hold the word count, as specified.

LDA bm (12) Load A

Replaces the contents of A with a 48-bit operand contained in the storage location specified by M. Negative zero is formed in A if the operand at M is equal to negative zero.

LAC bm (13) Load A Complement

Replaces the contents of A with the complement of a 48-bit operand contained in the storage location specified by M. Negative zero is formed in A if the operand at M is equal to positive zero.

LDQ bm (16) Load Q

Replaces the contents of Q with a 48-bit operand contained in the storage location specified by M. Negative zero is formed in Q if the operand at M is equal to negative zero.

LQC bm (17) Load Q Complement

Replaces the contents of Q with the complement of a 48-bit operand contained in the storage location specified by M. Negative zero is formed in Q if the operand at M is equal to positive zero.

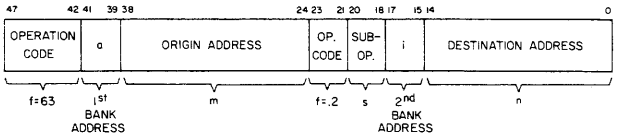
STA bm (20) Store A

Replaces the contents of the designated storage location, M, with the contents of A.

STQ bm (21) Store Q

Replaces the contents of the designated storage location, M, with the contents of Q.

Transmit Commands and Transmit Augment (*XMIT*)



The Transmit commands read an operand from the storage location designated by the first storage address 'a m', perform the specified operations, and place the results in storage address 'i n'. At the end of the operation, the Operand Bank register is set to 'i'. Interpretations of the 3-bit suboperation designator 's' are given below (the lower order two bits of 's' specify the operation):

s = X00 Transmit

The contents of storage location 'a m' are transmitted to storage location 'i n'. The contents of 'a m' remain unchanged.

s = X01 Transmit Complement

The complement of the contents of storage location 'a m' are placed in storage location 'i n'. The contents of 'a m' remain unchanged.

s = X10 Transmit Masked

The logical product of the contents of storage location 'a m' and Q is placed in storage location 'i n'. The contents of 'a m' remain unchanged.

s = X11 Transmit + Constant

The contents of storage location 'a m' are added to a constant (constant is in A) and transmitted to storage location 'i n'. The contents of 'a m' remain unchanged.

Values for the upper order bit of 's' are:

s = 0XX Execute the specified Transmit command without augmentation.

s = 1XX Augment the specified Transmit command.

The augment portion of this instruction may be used to increase the capability of the designated command. Commands may be executed without using the augment capability; in this case, just one word is transmitted.

With augmentation selected, the specified Transmit operation may be repeated a given number of times. Storage addresses may be increased by an increment quantity for each repetition of the operation. Five index registers are assigned to hold the necessary control quantities for an augmented Transmit operation. Index register assignment is as follows:

B^1 - Holds word count; i.e., the number of words to be transmitted. Its contents are reduced by one after each operation. When $(B^1) = 0$, operation is complete.

B^2 - Holds new origin address modifier (increment in B^3 has been added to old origin address). Normally, this index register is clear at start.

B^3 - Holds an increment quantity which will be added to the origin address modifier in B^2. Note: If $(B^3) = 0$, the same storage word will be transmitted the number of times specified by (B^1).

B^4 - Holds new destination address modifier (increment in B^5 has been added to old destination address). Normally this index register is clear at start.

B^5 - Holds an increment quantity which will be added to the destination address modifier in B^4. Note: If $(B^5) = 0$, all storage words transmitted will be placed in the same storage location.

Figure 3-2 details the operation of a sample augmented Transmit operation (s = 100).

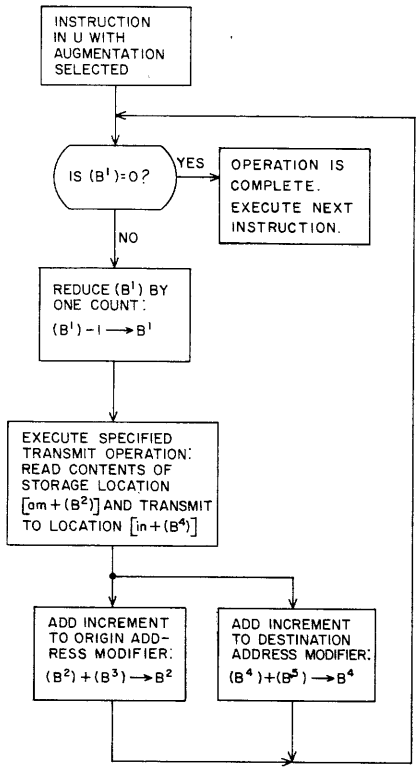


Figure 3-2. Augmented Transmit Operation.

Address Transmission

1) In the Address Transmission instructions, only the lower 15 bits of a 24-bit half-word instruction or data word are used.

2) In the LIU and LIL instructions, an index designation of "0" has no meaning and should not be used. If used, these instructions become pass instructions, but use some time in storage reference. The next instruction is then executed. Using "0" as an index designation does not constitute a fault.

LIU bm (52) Load Index Upper

Replaces the contents of the designated index register with the upper address of storage location 'm'. If b = 0, this instruction becomes a pass (do nothing) instruction.

LIL bm (53) Load Index Lower

Replaces the contents of the designated index register with the lower address of storage location 'm'. If b = 0, this instruction becomes a pass (do nothing) instruction.

SIU b m (56) Store Index Upper

Replaces the upper address portion of storage location 'm' with the contents of the designated index register. The remaining bits of the word in storage remain unchanged. If b = 0, (m_{ua}) is cleared.

SIL b m (57) Store Index Lower

Replaces the lower address portion of storage location 'm' with the contents of the designated index register. The remaining bits of the word in storage remain unchanged. If b = 0, (m_{la}) is cleared.

SAU b m (60) Substitute Address Upper

Replaces the upper address portion of M with the lower order 15 bits of A. Remaining bits of M are not modified and the initial contents of A are unchanged.

SAL b m (61) Substitute Address Lower

Replaces the lower address portion of M with the lower order 15 bits of A. Remaining bits of M are not modified and the initial contents of A are unchanged.

ENQ b y (04) Enter Q

The 15-bit operand, Y, is entered into Q and its highest order bit (sign bit) is extended in the remaining 33 bits. The largest positive 15-bit operand that can be entered into Q is 37777_8 $(2^{14}-1)$ and its "0" sign bit will be duplicated in each of the remaining 33 bits of Q. Negative zero will be formed in Q if:

*1) $(B^b) = 77777_8$ and $y = 77777_8$ or

*2) b = 0 and $y = 77777_8$.

ENA b y (10) Enter A

The 15-bit operand, Y, is entered into the A register and its highest order bit (sign bit) is extended in the remaining 33 bits. The largest positive 15-bit operand that can be entered into A is 37777_8 $(2^{14}-1)$ and the "0" sign bit will be duplicated in each of the remaining 33 bits. Negative zero will be formed in A if:

*1) $(B^b) = 77777_8$ and $y = 77777_8$ or

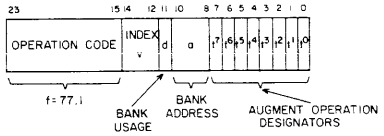
*2) b = 0 and $y = 77777_8$.

* One's complement mode.

ENI b y (50) Enter Index

Replaces (B^b) with the operand y. If b = 0, this instruction becomes a pass (do nothing) instruction.

Single Precision Augment



The 24-bit Single Precision Augment command may be used to perform one or more of the following operations:

1) Increase the capabilities of certain instructions by specifying additional operations to be performed.

2) Change the value of the Operand Bank register.

3) Provide additional modification of the address portion of the lower instruction.

When this command is used in the lower instruction position of a program step (case 2 above), the following operations occur:

1) The value of the Operand Bank register is set to 'a' (if 'd' is a "1").

2) All other designators perform no meaningful operations and have no effect on subsequent instructions.

When this command is used in the upper instruction position of a program step (case 1 above), the following operations occur:

1) Operations using the index designator 'v' are peformed (refer to table 3-3).

2) The value of the Operand Bank register is set to 'a' (if 'd' is a "1").

3) The augment operation designators 't' are stored to condition the operation of the lower instruction being augmented. If the lower instruction is not augmentable (i.e., not listed in either table 3-4 or 3-5), the "t" designators have no effect on these instructions. Note also that these instructions can only be augmented by using certain "t" designators listed. Any other "t" bits set have no effect on the instruction being augmented.

Instructions which may be augmented using 'v' and the augment designators 't' are listed in tables 3-4 and 3-5. Designators which may be used when augmenting a given instruction are checked opposite that instruction. Augment operation designators are listed in table 3-2.

When the Augment command is in the upper instruction position of a program step, the index designator 'v' may be used as shown in table 3-3 to augment the lower instruction.

Table 3-2. Augment Operation Designators

Designator	Value	
	If a "0"	If a "1"
d	Bank address not used	Bank address is used; the value of the bank address designator 'a' is always placed in the Operand Bank register.
t^0	Rounded arithmetic	Un-rounded arithmetic*
t^1	Normalized arithmetic	Un-normalized arithmetic
t^2	Use signed operand	Use magnitude of operand (positive value)
t^3	Leave source alone	Clear source (source is always a register)
t^4	Do not complement operand	Complement operand
t^5	Do normal operation	Do replace operation
t^6	Direction of shift determined by lower instruction operation code (normal).	Direction of shift determined by lower instruction operation code as modified by operand sign value (i.e., if $k + (B^b) + (V^v)$ is a negative value, reverse the direction of the shift being augmented).
t^7	Shift being augmented is end around (left shift) or end-off and sign extended (right shift).	Shift being augmented is end-off (left or right) or sign not extended (right shift).

Table 3-3. Augment Operation With 'v'

Value	Operation
v = 0	If v = 0, this designator has no significance in the operation.
v = 1-6	If v = 1-6, address modification rules apply. The contents of the index register specified by 'v' are added to the address portion (bits 00-14) of the lower instruction to form m, y, or k, whichever the case.
v = 7	If v = 7, indirect addressing rules apply. The quantity held in the address portion (bits 00-14) of the lower instruction is treated as a storage address (whether 'm', 'y', or 'k'). The lower 18 bits at this storage address are read from storage and: a) The upper 3 bits (new 'v') are placed in the 'v' designator position of the augment instruction. b) The lower 15 bits are placed in the address portion of the lower instruction. If new v = 7, indirect addressing continues until completed; if new v = 1-6, address modification is performed.

* If augmenting Divide Fractional instruction, execute Truncated Divide.

Upon completing the operations specified by the upper (Augment) instruction, the address portion of the lower instruction now contains a modified value (if 'v' specified indirect addressing or address modification). When the instruction being augmented (the lower instruction) is executed, its index designator is interpreted in the normal manner. Indirect addressing or address modification is performed on the address modified by the Augment operation.

Instructions which may be augmented are listed in table 3-4. In addition to the instructions listed, there are several instructions which may be augmented using designators t^2 and t^4. Even though augmenting

Table 3-4. Augmentable Instructions

Instruction to be Augmented	t^7	t^6	t^5	t^4	t^3	t^2	t^1	t^0	v**
ARS	X	X							
QRS	X	X							
LRS	X	X							
ALS	X	X							
QLS	X	X							
LLS	X	X							
LDA				X		X			
LDQ				X		X			
STA				X	X	X			
STQ				X	X	X			
LAC				X					
LQC				X					
ENA				X					
INA				X					
ENQ				X					
ADL			X						
SBL			X						
ADD				X		X			
SUB				X		X			
MUI				X		X			
MUF				X		X			
DVI				X		X			
DVF				X		X		X*	
FAD			X	X		X	X	X	
FSB			X	X		X	X	X	
FMU				X		X	X	X	
FDV				X		X	X	X	

* Refer to Truncated Divide instruction description.

** May be meaningfully used to augment most 24-bit instructions.

these instructions is a redundant operation (another instruction without augmentation usually can accomplish the same function), they may legitimately be augmented using designators t^2 and t^4. These instructions are listed in table 3-5.

Table 3-5. Instructions Which May Be Augmented Using t^2 and t^4

Mnemonic Code	Instruction
LAC	Load A Complement
LQC	Load Q Complement
SST	Selective Set
SCL	Selective Clear
SCM	Selective Complement
SSU	Selective Substitute
LIU	Load Index (Upper)
LIL	Load Index (Lower)
RAD	Replace Add
RSB	Replace Subtract
RAO	Replace Add One
RSO	Replace Subtract One

NOTE

If an instruction is augmented with designators t^2 and t^4 (t^2 = 1, t^4 = 1), the operation specified by designator t^4 takes precedence over the operation specified by t^2; i.e., t^2 is ignored.

Fixed Point Arithmetic

1) The Single Precision Augment instruction may be used with all instructions in this category (see table 3-4 for uses).

2) If the capacity of the A register, $\pm(2^{47}-1)$, is exceeded during the execution of the Fixed Point Arithmetic instructions (ADD and SUB), an arithmetic overflow fault is produced. When executing the DVI or DVF instructions, if the result exceeds the capacity of the A register, $\pm(2^{47}-1)$, a divide fault is produced (refer to appendix).

3) The Multiply Integer instruction (MUI) uses the double register configuration QA. The least significant bit of the product is left in bit position A_{00}. The most significant may be in either A or Q, depending on the magnitude of the product.

ADD b m (14) Add

Adds a 48-bit operand obtained from storage location M to contents of A. A negative zero may be produced by this instruction if (A) and (M) are initially negative zero.

SUB b m (15) Subtract

Obtains a 48-bit operand from storage location M and subtracts it from the initial contents of A. A negative zero will be produced if the initial contents of A are negative zero and that of storage location M are positive zero.

MUI b m (24) Multiply Integer

Forms a 96-bit product from two 48-bit operands. The multiplier must be loaded into A prior to execution of the instruction. The execution address specifies the storage location of the multiplicand. The product is contained in QA as a 96-bit quantity. The operands are considered as integers and the binary point is assumed to be at the lower order (right-hand) end of the A register.

DVI b m (25) Divide Integer

Divides a 96-bit integer dividend by a 48-bit integer divisor. The 96-bit dividend must be formed in the QA register prior to executing the instruction. If a 48-bit dividend is loaded into A, the sign of Q must be set (the sign of the dividend in A must be extended throughout Q). The 48-bit divisor is read from the storage location specified by the execution address. The quotient is formed in A and the remainder is left in Q at the end of the operation. Dividend and remainder have the same sign.

MUF b m (26) Multiply Fractional

Forms a 96-bit product from two 48-bit operands. The operands are treated as fractions with the binary point immediately to the right of the sign bit. The multiplier must be loaded into A prior to executing the instruction. The multiplicand is read from the storage location specified by M. The 96-bit product is contained in AQ.

DVF b m (27) Divide Fractional

Divides a 96-bit quantity by a 48-bit divisor (divisor must be > dividend). All operands are treated as fractions with the binary point immediately to the right of the sign bit. The 96-bit dividend must be

loaded into AQ prior to executing this instruction. If a 48-bit dividend is loaded into Q, the sign of Q must be extended throughout A. At the end of this operation the quotient is left in A and the remainder in Q. Remainder and dividend have the same sign.

77.1–27 Truncated Divide

If the Divide Fractional instruction is being augmented by the Single Precision Augment instruction, and the value of bit t^0 is a "1", a Truncated Divide is executed.

With this operation selected, a 96-bit integer is divided by a 48-bit integer. The Truncated Divide operation differs from the normal Divide Integer operation in the following respects:

1) The 96-bit dividend is formed in the AQ register prior to executing the instruction.

2) In the iterative sequence of the divide operation, six iterations are performed, rather than 48.

3) The quotient (most significant 6 bits) is left in the lower order bit positions of Q at the end of the operation.

4) The partial remainder is left in the remaining portion of AQ (the position of the remainder is its position after the shift operations for the six iterations).

Single Precision Floating Point Arithmetic

1) Refer to appendix for a discussion of floating point format and notes on floating point operations.

2) The Single Precision Augment instruction may be used with all instructions in this category.

3) Rapid FMU and FDV arithmetic by powers of two may be accomplished by using the Add to Exponent instruction.

4) Floating Point range faults (overflow/underflow) occur if the exponent exceeds $\pm(2^{10}-1)$. (Refer to the appendix.) Note that AQ is cleared if an exponent underflow fault occurs during any of these instructions.

FAD b m (30) Floating Add

Forms the sum of two operands packed in floating point format. A floating point operand is read from storage location M and added to the floating point word in A. The result is rounded, normalized, and retained in A at the end of the operation. Q contains only the residue of the rounding operation at the end of the sequence.

FSB b m (31) Floating Subtract

Forms the difference of two 48-bit operands in floating point format. The subtrahend is acquired from storage address M and is subtracted from the minuend in A. The result is rounded and normalized if necessary and retained in A. The residue from the rounding operation is left in Q at the end of the sequence.

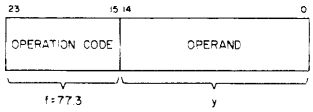
FMU b m (32) Floating Multiply

Forms the product of an operand in floating point format with the previous contents of A also in floating point format. The operand is read from storage location M. The product is rounded and normalized if necessary and retained in A. The residue from the rounding operation i left in Q at the end of the sequence.

FDV b m (33) Floating Divide

Forms the quotient of two 48-bit operands in floating point format. The dividend must be loaded into A prior to executing this instruction. The divisor is read from the storage location specified by M. The quotient is rounded and normalized if necessary and retained in A at the end of the operation. The residue from the rounding operation is left in Q at the end of the operation.

ADX y (77.3) Add to Exponent



This instruction adds a signed value field 'y' to the signed exponent in the A register. The operation is performed only if A ≠ 0. If A = 0, this instruction is a pass (or do nothing) instruction. If A ≠ 0 and is negative, (A) will be complemented before the add operation and recomplemented after the operation. Overflow and underflow conditions apply.

Double Precision Floating Point Arithmetic

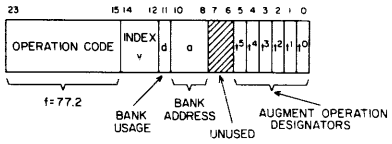
1) Refer to appendix for double precision floating point format and notes on floating point operations.

2) This category of instructions, selected by using the Double Precision Augment instruction, enables the use of 96-bit operands in executing the floating point operations (Add, Subtract, Multiply, and Divide). Refer to Double Precision Augment instruction.

3) The first 96-bit operand is in AQ by a previous instruction. The second 96-bit operand is read from consecutive storage locations M and M +1.

4) Floating point range faults (overflow/underflow) occur if the exponent exceeds $\pm(2^{10}-1)$. (Refer to the appendix.) Note that AQ is cleared if an exponent underflow fault occurs during any of the floating point instructions.

Double Precision Augment



The 24-bit Double Precision Augment command is used in the same manner as the Single Precision Augment command. Instructions which may be augmented by this command are listed in table 3-6. Operations in this instruction category are performed on 96-bit operands. Designators which may be used when augmenting a given instruction are checked opposite that instruction. Augment operation designators are listed in table 3-7.

Table 3-6. Augmentable Instructions

Instruction to be Augmented	t^5	t^4	t^3	t^2	t^1	t^0
LDA		X		X		
STA		X	X	X		
FAD	X	X		X	X	X
FSB	X	X		X	X	X
FMU		X		X	X	X
FDV		X		X	X	X

Table 3-7. Augment Operation Designators

Designator	Value	
	If a "0"	If a "1"
t^0	Rounded arithmetic	Un-rounded arithmetic
t^1	Normalized arithmetic	Un-normalized arithmetic
t^2	Use signed operand	Use magnitude of operand
t^3	Leave source alone	Clear source
t^4	Do not complement operand	Complement operand
t^5	Do normal operation	Do replace operation
d	Bank address not used	Bank address is used; the value of the bank address designator 'a' is always placed in the Operand Bank register.

DLDA (77.2-12) Load A Double Precision

Loads the double-length register AQ with a 96-bit operand contained in storage locations M and M + 1. The contents of storage location M are loaded into A; the contents of M + 1 are loaded into Q.

DSTA (77.2-20) Store A Double Precision

Stores the contents of the double-length register AQ in storage locations M and M + 1. The contents of A are stored at address M; the contents of Q are stored at M + 1.

DFAD (77.2-30) Double Precision Floating Add

Forms the sum of two 96-bit operands packed in floating point format. A floating point operand is read from storage locations M and M + 1 and added to the floating point word in AQ. The result is rounded, normalized, (if specified by appropriate augment designators), and retained in AQ at the end of the operation.

DFSB (77.2-31) Double Precision Floating Subtract

Forms the difference of two 96-bit operands in floating point format. The subtrahend is acquired from storage locations M and M + 1 and is subtracted from the minuend in AQ. The result is rounded and normalized if necessary (and if specified) and retained in AQ at the end of the operation.

DFMU (77.2-32) Double Precision Floating Multiply

Forms the product of a 96-bit operand in floating point format with the previous contents of AQ also in floating point format. The 96-bit operand is read from storage locations M and M + 1. The product is rounded and normalized if necessary (and if specified) and retained in AQ at the end of the operation.

DFDV (77.2-33) Double Precision Floating Divide

Forms the quotient of two 96-bit operands in floating point format. The dividend must be loaded into AQ prior to executing this instruction. The divisor is read from the storage locations M and M + 1. The quotient is rounded and normalized if necessary (and if specified) and retained in AQ at the end of the operation.

Address Arithmetic

1) In the Address Arithmetic instructions, only the lower 15 bits of the operand or data words are used.

INA b y (11) Increase A

Adds Y to A. The 15-bit operand Y, with its highest order bit (sign bit) extended, is added to A.

INI b y (51) Increase Index

Increases (B^b) by the operand 'y'. If the 'b' designator is zero, this instruction becomes a pass (do nothing) instruction.

ISK b y #(54) Index Skip

Compares (B^b) with 'y'. If the two quantities are equal, B^b is cleared and a full exit is performed. If the quantities are unequal, (B^b) is increased one count and a half exit is performed. (Counting in this instruction is performed in two's complement notation, regardless of whether or not the computer is in two's complement mode.) If b = 0, a full exit is taken. ISK is usually restricted to the upper instruction. If used as a lower instruction, it will half exit upon itself until the full exit condition is satisfied.

Logical

1) The LDL, ADL, SBL and STL instructions achieve their result by forming a logical product. A logical product is a bit by bit multiplication of two binary numbers:

$$0 \times 0 = 0 \qquad 1 \times 0 = 0$$
$$0 \times 1 = 0 \qquad 1 \times 1 = 1$$

2) A logical product is used, in many cases, to select specific portions of an operand for entry into another operation. For example, if only a specific portion of an operand in storage is to be added to (A), the operand is subjected to a mask composed of a predetermined pattern of "0's" and "1's". Forming the logical product of the operand and the mask causes the operand to retain its original contents only in those stages which have corresponding "1's" in the mask. When only the selected bits remain, the instruction proceeds to conclusion.

3) Capabilities of the Inter-Register instruction in logical operations should be noted.

SST b m (40) Selective Set

Sets the individual bits of A to "1" where there are corresponding "1's" in the word at storage location M; "0" bits in the storage word do not modify the corresponding bits in A. In a bit by bit comparison of (A) and (M) there are four possible combinations of bits.

1)	2)	3)	4)
$(A)_i = 1$	$(A)_i = 1$	$(A)_i = 0$	$(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(A)_f = 1$	$(A)_f = 1$	$(A)_f = 1$	$(A)_f = 0$
$(M)_f = 1$	$(M)_f = 0$	$(M)_f = 1$	$(M)_f = 0$

SCM b m (42) Selective Complement

Individual bits of A are complemented where there are corresponding "1's" in the word at storage location M. If the corresponding bits at M are "0's", the associated bits of A remain unchanged.

1)	2)	3)	4)
$(A)_i = 1$	$(A)_i = 1$	$(A)_i = 0$	$(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(A)_f = 0$	$(A)_f = 1$	$(A)_f = 1$	$(A)_f = 0$
$(M)_f = 1$	$(M)_f = 0$	$(M)_f = 1$	$(M)_f = 0$

SCL b m (41) Selective Clear

Clears individual bits of A where there are corresponding "1's" in the word at storage location M. If the corresponding bits at M are "0's" the associated bits of A remain unchanged.

In a bit by bit comparison of (A) and (M) there are four possible combinations of bits.

1)	2)	3)	4)
$(A)_i = 1$	$(A)_i = 1$	$(A)_i = 0$	$(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(A)_f = 0$	$(A)_f = 1$	$(A)_f = 0$	$(A)_f = 0$
$(M)_f = 1$	$(M)_f = 0$	$(M)_f = 1$	$(M)_f = 0$

SSU b m (43) Selective Substitute

Substitutes selected portions of an operand at storage address M into the A register where there are corresponding "1's" in the Q register (mask). The portions of A not masked by "1's" in Q are left unmodified.

In a bit by bit comparison of (A) and (M), there are four possible combinations of bits. The mask in Q may have one of two values. The result in A thus may have the final values for the various combinations as indicated below.

1)	2)	3)	4)
$(A)_i = 1$	$(A)_i = 1$	$(A)_i = 0$	$(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(Q)_i = 1$	$(Q)_i = 1$	$(Q)_i = 1$	$(Q)_i = 1$
$(A)_f = 1$	$(A)_f = 0$	$(A)_f = 1$	$(A)_f = 0$

1)	2)	3)	4)
$(A)_i = 1$	$(A)_i = 1$	$(A)_i = 0$	$(A)_i = 0$
$(M)_i = 1$	$(M)_i = 0$	$(M)_i = 1$	$(M)_i = 0$
$(Q)_i = 0$	$(Q)_i = 0$	$(Q)_i = 0$	$(Q)_i = 0$
$(A)_f = 1$	$(A)_f = 1$	$(A)_f = 0$	$(A)_f = 0$

LDL b m (44) Load Logical

Loads A with the logical product of Q and the designated storage location, M. The operand can be in either Q or M.

ADL b m (45) Add Logical

Adds to A the logical product of Q and the quantity in location M; the mask may be in Q or storage. Once the logical product is formed, addition follows normal rules (see appendix).

SBL b m (46) Subtract Logical

Subtracts from A the logical product of the Q register and the quantity in storage location M. The mask may be in Q or storage. When the logical product is formed, the subtraction proceeds in the normal manner (see appendix).

STL b m (47) Store Logical

Replaces the bits in location M with the logical product of Q and A. Neither (A) nor (Q) is modified. The mask may be located in A or Q.

Shifting

1) The largest practical shift count for a 48-bit register is 48_{10}; for a 96-bit register, 96_{10}. If a shift greater than the practical shift count is attempted, the Shift Fault indicator will be set and the shift will not be performed.

2) The Single Precision Augment instruction may be used with these instructions to provide greater flexibility.

3) When augmenting a shift instruction, double additive indexing is used to modify the shift count $[K = k + (B^b) + (V^v)]$.

4) Shifts are constant speed; e.g., performing a shift of 46_{10} places takes no longer than a shift of 1 place.

ARS b k (01) A Right Shift

Shifts contents of A to the right K places. The sign is extended and the lower bits are discarded. The largest practical shift count is 47_{10} since the register is now an extension of the sign bit.

QRS b k (02) Q Right Shift

Shifts contents of Q to the right K places. The sign is extended and the lower bits are discarded. The largest practical shift count is 47_{10} since the register is now an extension of the sign bit.

LRS b k (03) Long Right Shift

Shifts contents of AQ to the right K places as one 96-bit register. The A register is considered as the leftmost 48 bits and the Q register as the rightmost 48 bits. The sign of A is extended. The lower order bits of A replace the higher order bits of Q and the lower order bits of Q are discarded. The largest practical shift count is 95_{10} since AQ is now an extension of the sign of A.

ALS b k (05) A Left Shift

Shifts contents of A to the left K places, left circular. The higher order bits of A replace the lower order bits. The largest practical shift count, 48_{10}, returns the register to its original state.

QLS b k (06) Q Left Shift

Shifts contents of Q to the left K places, left circular. The higher order bits of Q replace the lower order bits. The largest practical shift count, 48_{10}, returns the register to its original state.

LLS b k (07) Long Left Shift

Shifts contents of AQ to the left K places, left circular, as one 96-bit register. The higher order bits of A replace the lower order bits of Q and the higher order bits of Q replace the lower order bits of A. The largest practical shift count, 96_{10}, returns AQ to its original state.

Scale

1) Address modification does not apply; the index register is used to preserve the scale factor.

2) If b = 0, scaling is executed but the scale factor is lost.

3) If b = 7, indirect addressing is used and at least one storage reference is made.

4) If $(A)_i$ is already scaled or equal to positive or negative zero, $k \rightarrow B^b$ and scaling is not executed.

5) If the execution address (k) is initially equal to 0, B^b is cleared.

6) The Shift Fault indicator is not affected by these instructions.

SCA b k (34) Scale A

Shifts A left circularly until the most significant digit is to the right of the sign bit or until k = 0. Shift count 'k' is reduced by one for each shift and terminates when k = 0 or the most significant digit is to the right of the sign bit. Upon termination, the count (scale factor) is entered in the designated index register.

SCQ b k (35) Scale AQ

Shifts AQ left circularly until the most significant digit is to the right of the sign bit. Shift count 'k' is reduced by one for each shift. Operation terminates when k = 0 or the most significant digit is to the right of the sign bit. Upon termination, the count (scale factor) is entered in the designated index register.

Replace

1) If the capacity of the A register, $\pm(2^{47}-1)$, is exceeded during the execution of the Replace instructions, an arithmetic overflow fault is produced (refer to appendix).

2) The Single Precision Augment and the Double Precision Augment instructions (with the appropriate designator set) also effect a replace operation when used to augment a FAD or FSB operation.

RAD b m (70) Replace Add

Obtains a 48-bit operand from storage location M and adds it to the initial contents of A. The sum is left in A and is also transmitted to location M.

RSB bm (71) Replace Subtract

Subtracts A from M and places the result in both the A register and location M.

RAO b m (72) Replace Add One

Replaces the operand in storage location M with its original value plus one. The result is also placed in A.

RSO b m (73) Replace Subtract One

Replaces the operand in storage location M with its original contents minus one. The difference is also left in A; the original contents of A and M are destroyed.

Storage Test

SSK b m #(36) Storage Skip

Senses the sign bit of the operand in storage location M. If the sign is negative, a full exit is taken. If the sign is positive, a half exit is taken. The contents of the operational registers are left unmodified. SSK is usually restricted to an upper instruction. If used as a lower instruction and the sign of (M) is negative, a full exit will be executed. If the sign is positive, it will half exit upon itself and never execute a full exit.

SSH b m #(37) Storage Shift

Senses the sign bit of the quantity in storage location M. If the sign bit is negative, a full exit is taken. If the quantity is positive, a half exit is taken. In either case the quantity is shifted left circular one bit before the exit. This instruction is usually restricted to the upper position. If used as a lower instruction and the sign of (M) is positive, the instruction will half exit upon itself until a negative sign bit is found. The contents of the operational registers are left unmodified.

Storage Search

1) If b = 0 in the following instructions, only the word at storage location 'm' will be searched.

2) If b = 7, indirect addressing is used to obtain the execution address and 'b' designator.

3) If $(B^b) = 0$, no search is made, and a half exit is taken.

4) The operands searched by these instructions may be in either fixed or floating point format.

EQS b m #(64) Equality Search

Searches a list of operands to find one that is equal to A. The number of items to be searched is specified by B^b. These items are in sequential addresses beginning at the location specified by 'm'. The search begins with the last address, $m + B^b - 1$. B^b is reduced one count for each word that is searched until an operand is found that equals A or until B^b equals zero. If the search is terminated by finding an operand that equals A, a full exit is made. The address of the operand satisfying this condition is given by the sum of 'm' and the final contents of B^b. If no operand is found that equals A, a half exit is taken. Positive zero and minus zero are recognized as unequal quantities. When EQS is used as a lower instruction, a full exit is always taken when $(B^b) = 0$, or when the condition is met.

THS b m #(65) Threshold Search

Searches a list of operands to find one that is greater than A. The number of items to be searched is specified by B^b. These items are located in sequential addresses beginning at the location specified by 'm'. The search begins with the last address, $m + B^b - 1$. The contents of the index register are reduced by one for each operand examined. The search continues until an operand is reached that is greater than A or until B^b is reduced to zero. If the search is terminated by finding an operand greater than the value in A, a full exit is performed. The address of the operand satisfying the condition is given by the sum of 'm' and the final contents of B^b. If no operand in the list is greater than the value in A, a half exit is performed. If THS is used as a lower instruction, the next instruction will be executed when search terminates. In the comparison made here, positive zero is considered as greater than minus zero.

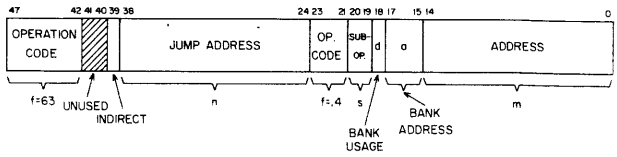
MEQ b m #(66) Masked Equality Search

Searches a list of operands to find one such that the logical product of (Q) and (M) is equal to (A). This instruction, except for the mask in Q, operates in the same manner as an equality search.

MTH b m #(67) Masked Threshold Search

Searches a list of operands to find one such that the logical product of (Q) and (M) is greater than (A). Except for the mask in Q, this instruction operates in the same manner as the threshold search.

Search Order



Four search operations are conditioned by the designator 's'.

s = 0 Equality Search (SEQU)

s = 1 Masked Equality Search (SMEQ)

s = 2 Search Within Limits (SEWL)

s = 3 Search Magnitude Within Limits (SMWL)

Each of these operations searches a list of operands to find one that satisfies the specific criterion. These items may be in sequential or incremented (other than 1) addresses. The first item is in the location specified by starting address 'a m' + B^2.

In the Search instruction (63.4), depending on the value of bit 39, storage is referenced at address [m+ (B^2)] to obtain the operand or the address of the operand. This storage reference is to bank "a" if "d" = 1; ("a" is placed in the operand Bank register) if "d" = 0, the reference is within the bank currently selected as the operand bank.

If bit 39 = 0, the contents of address [m+ (B^2)] is the operand. If bit 39 = 1, the lower 18 bits are read from this address. These lower 18 bits designate the storage address (bank and address) from which the operand will be obtained. Note that the upper 3 bits of this 18 - bit address are not placed in the Operand Bank register. (B^1) is reduced one count for each word that is searched until an operand is found that satisfies the criterion or until B^1 equals zero. If the search is terminated by finding an operand which meets the criterion, an exit is performed to the next instruction. The address of the operand that meets the criterion is m + (B^2) - (B^3). If no operand in the list is found that meets the criterion, a jump is executed to the location specified by the jump address 'n' (jump is effected within the same storage bank).

Three index registers used in the search operations are assigned as follows:

B^1 - Holds the word count; i.e., the number of words to be searched.

B^2 - Holds the new operand address modifier (the increment has been added to the old address modifier).

B^3 - Holds the increment quantity; i.e., the quantity which will be added to the operand address modifier to specify a new operand address.

Index register values are set by program prior to executing the Search instruction

The search operations are:

s = 0 Equality Search (M) = (A)

Searches a list of operands to find one such that (M) is equal to (A).

s = 1 Masked Equality Search L(Q)(M) = (A)

Searches a list of operands to find one such that the logical product of (M) and (Q) is equal to (A).

s = 2 Search Within Limits (A) $\geq$ (M) > (Q)

Searches a list of operands to find one whose value lies between (A) and (Q).

s = 3 Search Magnitude Within Limits (A) $\geq |(M)| >$ (Q)

Searches a list of operands to find one whose absolute magnitude lies between (A) and (Q). (Magnitude refers to the magnitude of signed operands.)

Figure 3-3 outlines the sequence of events during a typical search operation.

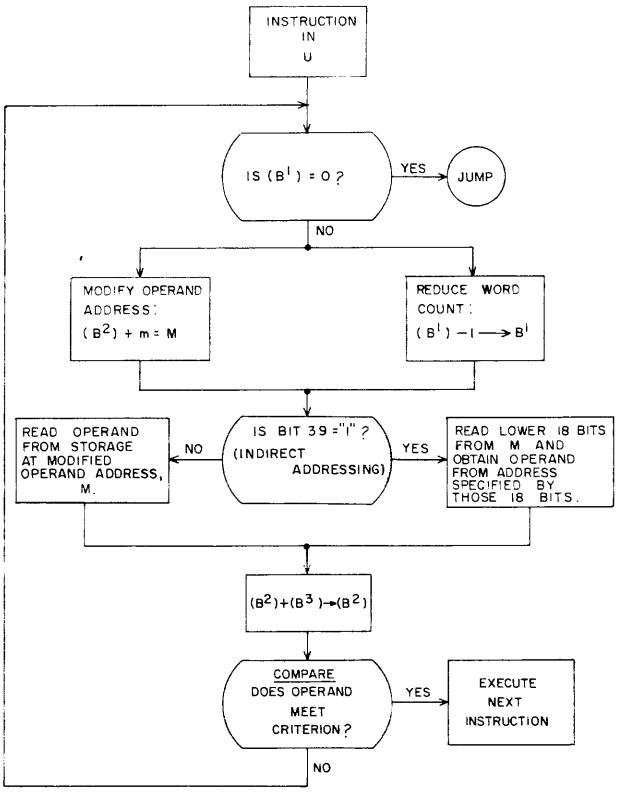
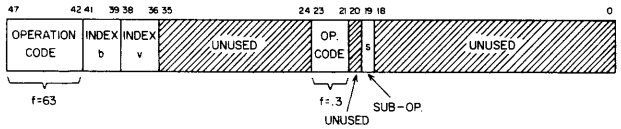


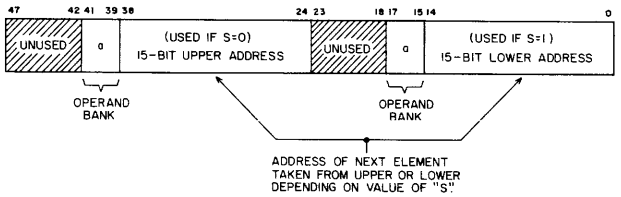
Figure 3-3. Sequencing for 63.4 Search Operations

Locate List Element (*LSTU, LSTL*) (63.3)



This instruction may be used to locate elements of a list when the elements are scattered throughout a storage bank or throughout several storage banks. An element of the list contains two parts: (1) data, and (2) the location (storage address) of the next element of the list. An element may occupy one storage word or several. If the element occupies several storage words, the words are usually in consecutive storage locations.

The format of the word holding an element (or the format of the first word if the element occupies several words) is as follows:



Note that the format permits several options for positioning data within the word:

1) If the lower 18 bits are used to specify the address of the next element (s = 1 in the instruction word), the entire upper 30 bits may be used to hold data.

2) If the upper address portion is used to specify the address of the next element (s = 0 in the instruction word), the lower 24 bits and the uppermost 6-bit portion may be used to hold data.

Interpreting data is determined by the list containing it, its location in a particular list, or by an identifying tag in the data portion of the element.

Executing the Locate List Element instruction locates the 'nth' element of the list.

Before executing the Locate List Element instruction, two operations must have been accomplished. These are:

1) The first element of the list is located at the storage address designated by the contents of V^v. (The first element must be in the storage bank currently in use as the operand bank.) Therefore, this index register must be loaded with the appropriate address before executing this instruction.

2) B^b designates another index register (distinct from V^v) which holds the count field. B^b must be preset to contain one less than the count desired. That is, if one wishes to locate the "nth" element, he must load B^b with the value (n-1).

Note that neither 'b' nor 'v' can be set to 7 in this instruction.

Operation then proceeds as follows:

1) Examine (B^b). If $(B^b) = 0$, the 'nth' element has been located and the operation is complete. If $(B^b) \neq 0$, reduce (B^b) by one.

2) Read element from storage at address specified by (V^v).

3) If s = 0 (refer to instruction format), replace the old (V^v) with the upper address portion of the new storage word (refer to the format of the word holding an element).

If s = 1, replace the old (V^v) with the lower address portion of the new storage word.

NOTE

Once the value of 's' is set in the instruction word, it cannot be changed during the course of the instruction. For example, if 's' is set to "1", the lower address portion of the storage word will be used to locate the next element each time until the 'nth' element is located.

4) Examine (V^v). If the new $(V^v) = 0$, the operation is complete. (The programmer must previously have loaded all zeros in the designated address portion of the last storage word of the list.) Address 00000 may be used as the first list element location, but operation will halt (since the check on V^v occurs after the first storage reference) if address 00000 is used to locate a subsequent list element.

If $(V^v) \neq 0$, (V^v) now designates the address of the next element. Also, if the new $(V^v) \neq 0$, the Operand Bank register is switched to 'a' (where 'a' is the bank designator adjacent to the designated address portion of the storage word).

5) Return to step 1.

At the end of operations:

1) If the 'nth' element has been located, B^b holds a count of zero, V^v holds the address of the 'nth' element, and the Operand Bank register is set to the bank containing that element.

2) If the 'nth' element has not been located and the last element has been reached, B^b holds a non-zero count (indicating an erroneous count was initially placed in B^b), V^v holds 00000, and the Operand Bank register holds the bank address of the last element.

The flow diagram in figure 3-4 details the operation of this instruction.

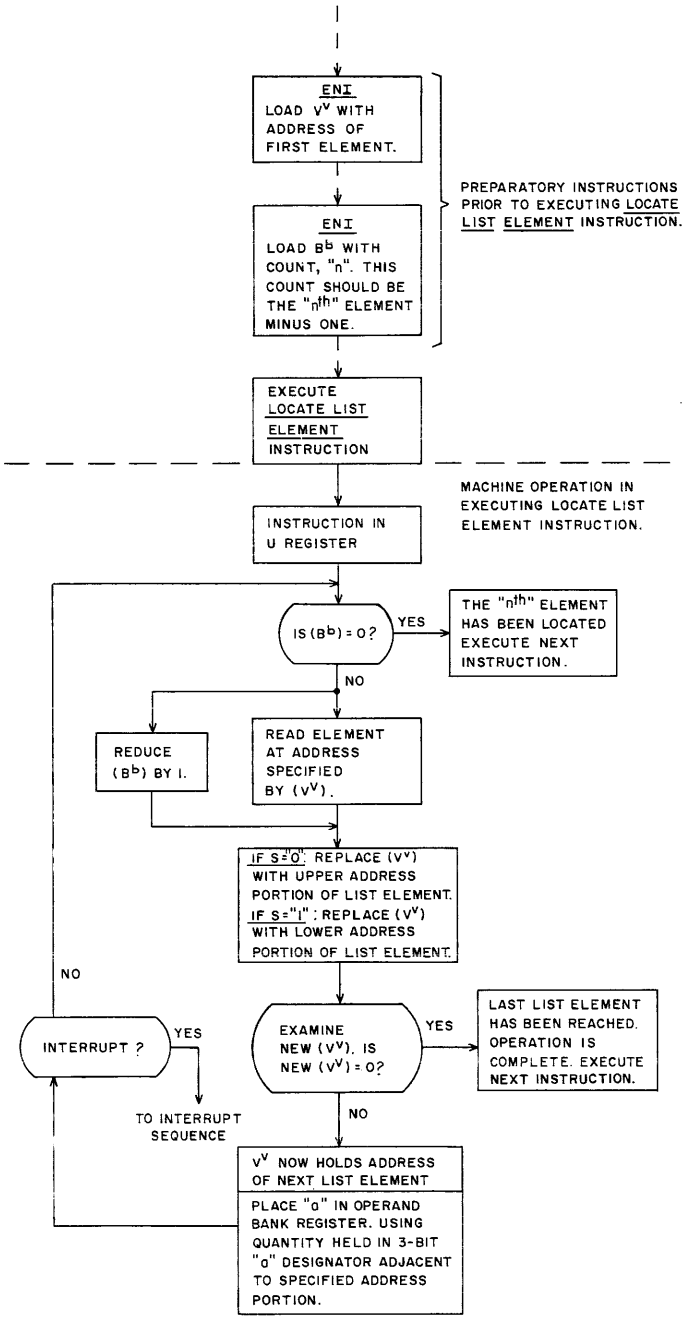


Figure 3-4. Locate List Element Operations

Jumps and Stops

Normal Jump

A Jump instruction causes a current program sequence to terminate and initiates a new sequence at a different location in storage. The Program Address register, P, provides continuity between program steps and always contains the storage location of the current program step.

When a jump instruction occurs, P is cleared and a new address is entered. In all jump instructions, the execution address 'm' specifies the beginning address of the new program sequence. The word at address 'm' is read from storage, placed in U and the upper instruction (first instruction of the new sequence) is executed.

Some of the jump instructions are conditional upon a register containing a specific value or upon the position of a Jump or Stop switch on the console. If the criterion is satisfied, the jump is made to location 'm'. If it is not satisfied, the program proceeds in regular sequence to the next instruction.

A jump instruction may appear in either position in a program step. If the jump instruction appears in the first (upper) part of the program step and the jump is taken, the second (lower) part of the program step is never executed. If the instruction appears in the lower part, the upper part is executed in the normal manner.

Return Jump

A return jump begins a new program sequence at the lower instruction portion of the program step to which the jump is made. At the same time, the execution address of the upper instruction of that program step is replaced with the address of the next program step in the main program. This instruction is usually an Unconditional Jump instruction and allows a return to the main program after completing the subprogram sequence (figure 3-5).

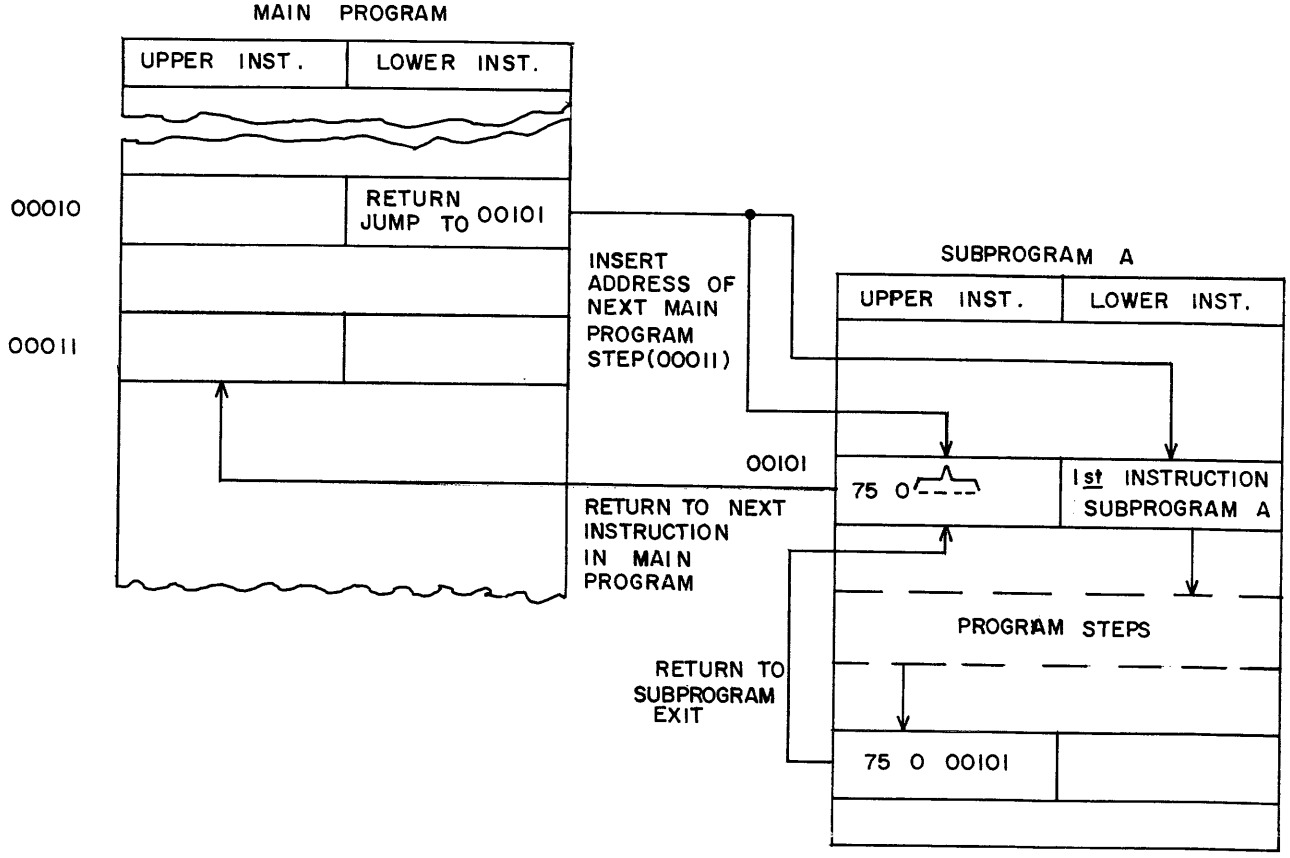


Figure 3-5. Return Jump

AJP j m (22) A Jump (Normal)

Jumps to 'm' if the conditions of the A register specified by the jump designator 'j' exist. If not, the next instruction is executed.

j = 0 Jump if (A) = 0
j = 1 Jump if (A) ≠ 0
j = 2 Jump if (A) = +
j = 3 Jump if (A) = −

When (A) is negative zero the interpretation is:

j = 0 The jump is executed because, in this case, negative zero is recognized as positive zero.
j = 1 The jump is not executed when (A) = +0 or −0.
j = 2 The jump is not executed because the sign bit is a "1".
j = 3 The jump is executed because the sign bit is a "1".

ARJ j m (22) A Jump (Return)

Executes a return jump to storage location 'm' if the condition of the A register specified by 'j' exists. If not, the next instruction is executed.

j = 4 Return jump if (A) = 0
j = 5 Return jump if (A) ≠ 0
j = 6 Return jump if (A) = +
j = 7 Return jump if (A) = −

Note: If (A) = negative zero, the AJP (Normal) interpretation applies.

QJP j m (23) Q Jump (Normal)

Jumps to 'm' if the condition of the Q register specified by the jump designator 'j' exists. If not, the next instruction is executed.

j = 0 Jump if (Q) = 0
j = 1 Jump if (Q) ≠ 0
j = 2 Jump if (Q) = +
j = 3 Jump if (Q) = −

When (Q) = negative zero the AJP interpretation applies.

QRJ j m (23) Q Jump (Return)

Executes a return jump to storage location 'm' if the condition of the Q register specified by 'j' exists. If not, the next instruction is executed.

j = 4 Return jump if (Q) = 0
j = 5 Return jump if (Q) ≠ 0
j = 6 Return jump if (Q) = +
j = 7 Return jump if (Q) = −

Note: If (Q) = negative zero, the AJP interpretation applies.

IJP b m (55) Index Jump

Examines (B^b). If this quantity is not zero, the quantity is reduced one count and a jump is executed to program step 'm'. The index jump can be used in the upper or lower instruction without reservation; it executes a normal jump upon satisfaction of the jump condition. If b = 0, a normal exit is taken. (Counting in this instruction is performed in two's complement notation, regardless of whether or not the computer is in two's complement mode.)

SLJ j m (75) Selective Jump (Normal)

Jumps to 'm' if the condition of the Jump switches specified by 'j' exists. If not, the next instruction is executed.

SLJ j = 0 Jump unconditionally (does not reference Jump switch setting).
SJ1 j = 1 Jump if Jump switch 1 is set
SJ2 j = 2 Jump if Jump switch 2 is set
SJ3 j = 3 Jump if Jump switch 3 is set

SRJ j m (75) Selective Jump (Return)

Executes a return jump to storage location 'm' on condition 'j' where condition 'j' represents the setting of the Jump switches. If the condition is not satisfied, the next instruction is executed.

RTJ j = 4 Return jump unconditionally (does not reference Jump switches).
RJ1 j = 5 Return jump if Jump switch 1 is set
RJ2 j = 6 Return jump if Jump switch 2 is set
RJ3 j = 7 Return jump if Jump switch 3 is set

Note: The Jump switch is illuminated when it is in the Set position.

SLS j m (76) Selective Stop (Normal)

Stops at present step in the sequence if the condition of the Stop switch specified by 'j' exists. If the stop condition exists, the stop is executed, and the jump is executed unconditionally when the Go switch is pressed. If the stop condition is not satisfied, the jump is executed unconditionally.

SLS j = 0 Stop unconditionally (does not reference Stop switch setting).
SS1 j = 1 Stop if Stop switch 1 is set
SS2 j = 2 Stop if Stop switch 2 is set
SS3 j = 3 Stop if Stop switch 3 is set

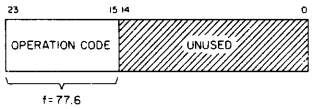
SLS j m (76) Selective Stop (Return)

Stops on condition 'j' and executes a return jump when the Go switch is pressed. If the stop condition is not satisfied, the stop is not executed and the return jump is executed unconditionally.

SRJ j = 4 Stop unconditionally; return jump on Go (does not reference Stop switches).
SR1 j = 5 Stop if Stop switch 1 is set; return jump on Go.
SR2 j = 6 Stop if Stop switch 2 is set; return jump on Go.
SR3 j = 7 Stop if Stop switch 3 is set; return jump on Go.

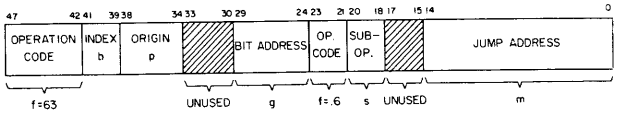
Note: The Stop switch is illuminated when it is in the Set position.

DRJ# (77.6) D Register Jump



This instruction scans the D (Flag) register from left to right. If a non-zero bit is detected, a jump is executed to address P + i + 1 (where P is the current address and 'i' is the location of the first non-zero bit in the D register).* If the D register is zero, the next instruction is executed. This instruction is restricted to use as an upper instruction.

ZBJP, NBJP (63.6) Bit Sensing



This instruction examines a single bit of 48 (specified by 'g') in a designated register ('p' specifies the register - see Inter-Register instruction). Note: When the A Upper Address or Q Upper Address register is specified by 'p', the lowest order bit in that register is numbered 00. A 3-bit suboperation designator 's' specifies the operation to be performed on this bit.

The lower order two bits of 's' specify the operation to be performed after the sense operation:

s = 0 Leave bit alone
s = 1 Set bit to "1"
s = 2 Clear bit
s = 3 Complement (toggle) the bit

If, in the execution of this instruction, an attempt is made to alter the contents of registers designated by codes 20-31 (except 25), the following occurs:

1) That operation which would alter the contents of register 20-31 (except 25) is not performed, and

2) The instruction continues to completion.

This operation does not constitute a fault or interruptible condition.

The upper order bit of 's' conditions a jump operation.

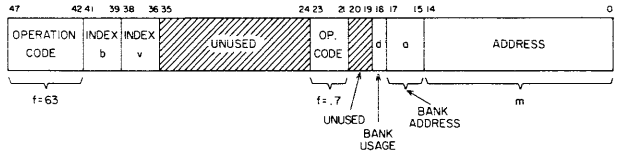
If upper order bit of s = 1 Jump if bit being examned in the specified register is a "0".

If upper order bit of s = 0 Jump if bit being examined in the specified register is a "1".

If the condition is met, the jump is to the address specified by M. If the jump condition is not met, the next instruction is executed. If, in the execution of this instruction, 'p' specifies 00, no register is specified, and zeros are used in the operation.

* 'i' = the location of the particular bit within the register. Though the register is numbered from right to left, scanning is from left to right. The first bit scanned is bit 47. If a "1" exists in bit 47, i = 47, and a jump is effected to [P + 47 + 1] (decimal notation).

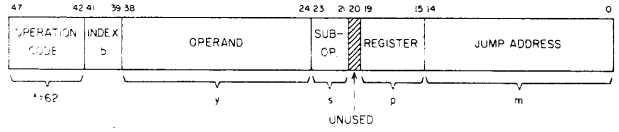
EXEC b v m (63.7) Execute



Jumps to M [M = m + (B^b) + (V^v)] and executes the instruction(s) at M. (An index designator of v = 7 in this instruction is illegal; if used, the result of address modification is 2 [m + (B^b)].) The Operand Bank register is set to 'a' if 'd' is a "1". After executing M, the main sequence continues unless the instruction executed was a jump. In this case, a new sequence is initiated at the jump address in the same program bank as EXEC. This instruction is effectively an indirect instruction, or a subroutine of a single instruction. Note that the instruction bank address is not changed by the Execute instruction and the contents of P remain at the address of the EXEC instruction.

During execution of the instruction pair specified by the Execute instruction, interrupt conditions are not recognized. Once execution of the upper instruction of the pair has begun, an interrupt will not be recognized until the next upper instruction is performed. Thus, if that upper instruction (specified by Execute) produces an interruptible fault condition, the lower instruction will be executed before interrupt can occur.

RGJP b m (62) Register Jump



Jumps to M [M = m + (B^b)] if, in the operation specified by 's', the condition is met. If the condition is not met, it executes the next instruction. If b = 7 in this instruction, the jump address is obtained indirectly.

If, in the execution of this instruction, an operation would attempt to alter the contents of registers designated by codes 20-31 (except 25), the following occurs:

1) The operation (s = 6 or 7) which could alter registers 20-31 (except 25) is not performed, and

2) The instruction continues to completion.

This operation does not constitute a fault or interruptible condition.

Octal values for 's' and the specific operations are:

s	Operation
0	(p) = y?
1	(p) > y?
2	(p) < y?
3	(p) ≠ y?
4	(p) ≤ y?
5	(p) ≥ y?
6	(p) < y? } if (p) < y, then (p) − y → (p)
7	(p) ≥ y? }

In executing the above operations, the following arithmetic properties hold:

a) If 'p' designates a 48-bit register, the sign bit of 'y' is extended, and signed quantities are compared in the operation.

b) If 'p' designates a 15-bit register, the operands are compared as 48-bit quantities (15-bit quantities with "0's" extended in the upper bit positions). If 'p' designates a register < 15 bits (e.g., the SCR), "0's" are extended.

Operand registers are indicated below according to their octal values of 'p'. (If, in the execution of this instruction, 'p' specifies 00, no operand is specified and zeros are used in the operation.)

01 - B^1
02 - B^2
03 - B^3
04 - B^4
05 - B^5
06 - B^6
07 - A Lower Address
10 - A Upper Address
11 - Q Lower Address
12 - Q Upper Address
13 - A Full
14 - Q Full
15 - D Full
16 - Bounds Register
17 - Interrupt Mask Register

* These are not registers, but forced operands which may be referenced in the operation.

20 - Interrupt Register
* { 21 - All "0's"
* { 22 - +1
* { 23 - All "1's"
24 - Instruction Bank Register
25 - Operand Bank Register
26 - Shift Count Register
27 - Miscellaneous Mode Selections
30 - P Register
31 - Time Register
32 - Time Limit Register

Bank Jumps (63.0)

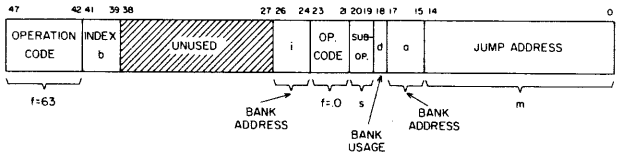
1) The bank usage designator 'd' must be set to "1" to effect an inter-bank jump. If 'd' = 0, all jumps are within the storage bank currently in use.

2) The bank address designator 'a' must be set to the bank to which the jump is desired.

3) The Jump and Set Index instruction effects changes in bank addresses. These bank addresses remain in effect until explicitly changed.

4) Interpreting the bank designators (both 'a' and 'i') depends on the value of 'd'.

The Bank Jumps are similar in operation to the Normal and Return Jumps. These instructions provide the additional capability of inter-bank jumping. Provision is also made for returning to the initial bank.



Three jump instructions, using the format above, are designated by the suboperation designator 's'. Jumps may be within the same storage bank or to another storage bank. In the latter case, provision is made for returning to the initial storage bank (BRTJ or BJSX) upon completion of the instruction(s) in the bank to which the jump was effected. Interpretations of 's' and the operations are outlined below.

s = 0 Unconditional Jump *(UBJP)*

Jumps unconditionally to the modified jump address M [M = m + (B^b)]. The jump is to the storage bank designated by 'a'. The operand bank is set to 'i'. If b = 0, the jump is to 'm'. If b = 7, the jump address is obtained by indirect addressing.

s = 1 Unconditional Return Jump *(BRTJ)*

Stores an Unconditional Bank Jump instruction (63.0) (s=0) in address designated by 'aM' [M = m + (B^b)].

Designator values in this stored (UBJP) instruction are selected to enable return to the initial storage bank. Values for these designators are:

1) m = P + 1	Enables return to the next main program step.
2) a = instruction bank	Set to bank in which main program is located to enable return to appropriate bank in which P + 1 is located.
3) i = operand bank	Set to bank in which main program operands are located (may be other than main program instruction bank).
4) b = 0	Index designator set to zero so that P + 1 (in 'm' portion) is not modified when UBJP instruction is executed.

After storing this instruction, jumps to M + 1 (address immediately following store address); sets operand bank to 'i'.

s = 2 Jump and Set Index *(BJSX)*

The Jump and Set Index instruction provides the programmer with a convenient method of bridging and processing constants which may be used at various places in a program loop. In executing this instruction, bank addresses are changed. The operation of this instruction provides for returning to the initial bank. Operations for this instruction are:

1) Store contents of the Program Address register (P) in B^b.

2) Store Unconditional Bank Jump instruction (63.0) (s = 0) in storage address designated by 'a m'. Values for designators in the UBJP instruction (that is stored) are automatically set as follows:

   a) m = 00000.

   b) a = value of the Instruction Bank register at the time the (63.0) (s = 2) instruction was executed.

   c) i = the value of the current contents of the Operand Bank register.

d) b = the index register in which (P) was stored.

3) Jump to address 'a m' + 1. The Operand Bank register is set to the value of the Instruction Bank register at the time the (63.0) (s = 2) instruction was executed.

Example (see figure 3-6):

Step 1: The Jump and Set Index instruction is executed; the following operations occur.

a) Stores P (00010) in B^b.

b) Stores UBJP instruction at address 305000 (bank 3; address 05000). Values for 'a' and 'i' are set to enable a return to bank 0; 'm' is set to 00000.

c) Jumps to 'a m' + 1 and sets the operand bank to the previous instruction bank.

Step 2: Subprogram instructions are executed; operands used are in bank 0.

Step 3: At conclusion of subprogram, execute Increase Index instruction. Since (P) is stored in B^b, the INI instruction effectively increases the count in P[00010 + 00005 = 00015], (B^b) = 00015.

Step 4: An unconditional jump is executed, returning the program to the UBJP instruction at address 05000.

Step 5: The UBJP instruction is executed. Modifying execution address [M = m + (B^b) = 00000 + 00015] provides for returning to the next main program step.

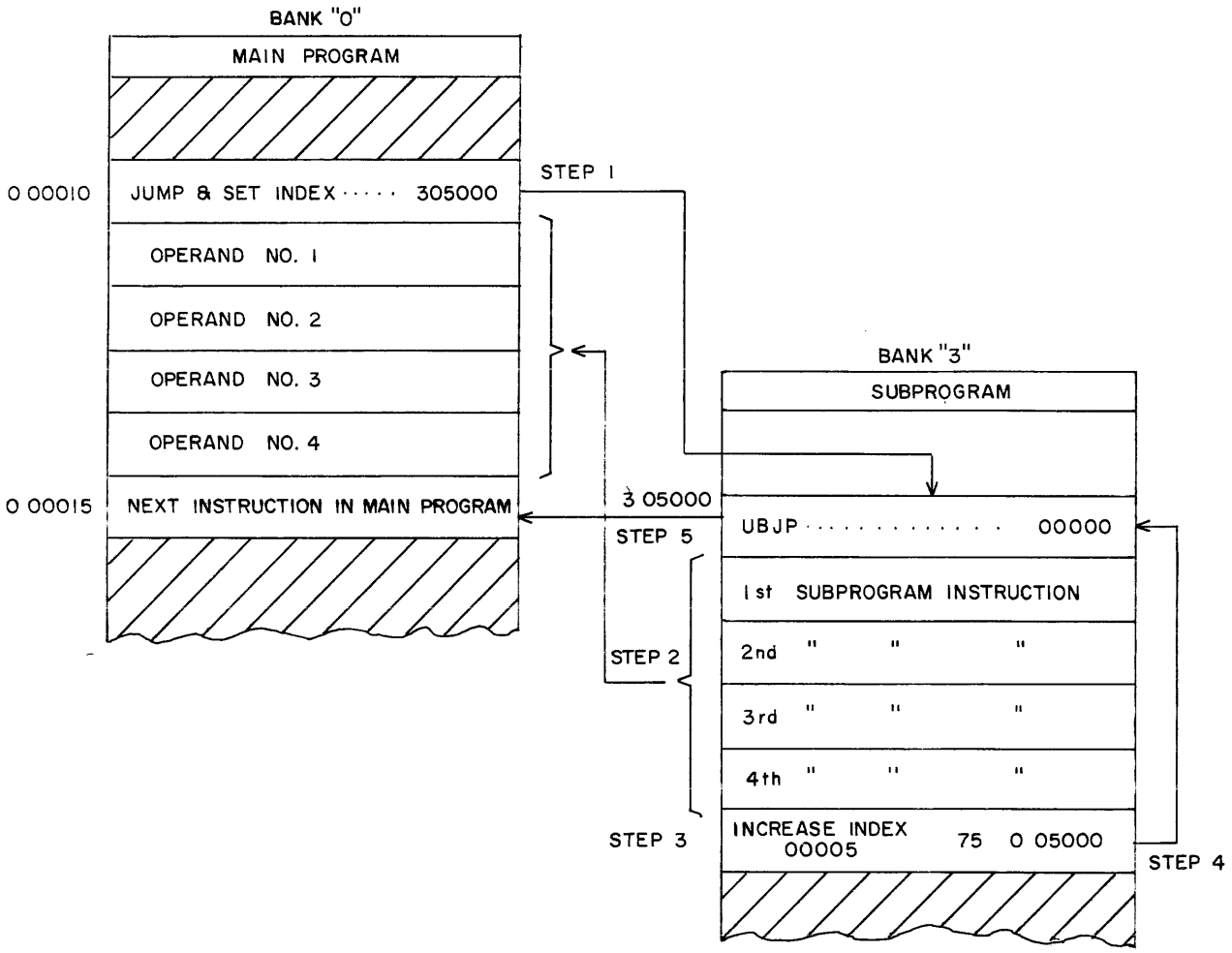
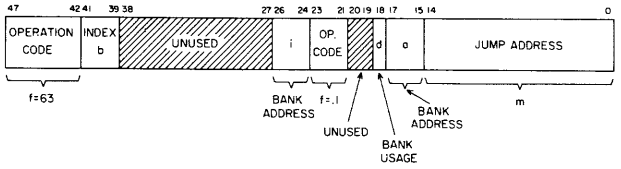


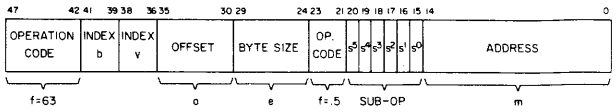
Figure 3-6. Jump and Set Index

Unconditional Jump To Lower (*BJPL*)(63.1)



This instruction, specified by operation code 63.1, performs an unconditional jump to the lower instruction contained in the storage address designated by M [M = m + (B^b)]. The jump is to the storage bank designated by 'a'. The operand bank is set to 'i'. If b = 0, the jump is to the lower instruction in 'm'; if b = 7, the jump address is obtained by indirect addressing.

Variable Data Field (63.5) Byte (LBYT, SBYT, SCAN)



The byte instruction performs two general operations on specified portions (bytes) of A, Q, or a designated storage operand. These operations are:

Transmit

1) Load - Loads a byte of the specified register (A or Q) with the designated byte of M.
2) Store - Stores a byte of A or Q in the selected byte of M.

Scan

Searches storage operands in byte-size increments until the specified condition is met or until (A) = 0 (the A register holds the byte count). Operation designators are tabulated below.

Designator	Operation
o	Offset designator; holds address of rightmost bit of byte in A or Q (bytes are left to right).
e	Byte-size designator; specifies size of byte (number of bits) used in the operation. (Legitimate values: 1, 2, 3, ... 48; 0 is not legitimate.)
* b	Index designator; used as an address modifier. If b = 7, indirect addressing occurs; address modification then occurs using the new 'b' and 'm'.
* v	Index designator; holds address of rightmost bit of byte of M. If v = 7 on Load Byte, v is read as "0".
s^5	Suboperation designator: if a "0", execute transmit operation. if a "1", execute scan operation.
s^0 - s^4	Suboperation designators; significance of each depends on the value of s^5 (i.e., whether transmit or scan operation is to be performed).

s^0 - s^4 values for transmit operation:

Designator	Operation	
	If a "0"	If a "1"
s^4	Execute operation without indexing	Execute operation with indexing
s^3	Use right indexing	Use left indexing
s^2	Execute load operation	Execute store operation
s^1	Use A register	Use Q register
s^0	Clear entire word	Replace byte

* Both B^b and V^v must be loaded by program.

The transmit operations are:

Load (for this example, the A register has been selected)

1) Load specified byte of M into specified byte of A.

2) If right indexing is selected, bytes of M are loaded from left to right into designated byte of A. After byte is loaded, operation is then:

   a) Subtract 'e' (byte-size) from (V^v); V^v now holds decremented byte address in M.

   b) Examine (V^v). If (V^v) $\geq$ 0, a skip exit is taken. If (V^v) $<$ 0, the load operation on bytes of M is complete, and a normal exit is taken.

3) If left indexing is selected, bytes of M are loaded from right to left into the designated byte of A. After the byte is loaded, operation is:

   a) Add 'e' (byte-size) to (V^v); (V^v) now holds incremented byte address in M.

   b) Examine (V^v). If (V^v) $\leq$ 48-e, a skip exit is taken. If (V^v) $>$ 48-e, the load operation on bytes of M is complete, and a normal exit is taken.

Note that only one byte is transmitted in the above operation. If subsequent byte operations are desired, the appropriate indexing steps must be accomplished by the main program (refer to figure 3-7).

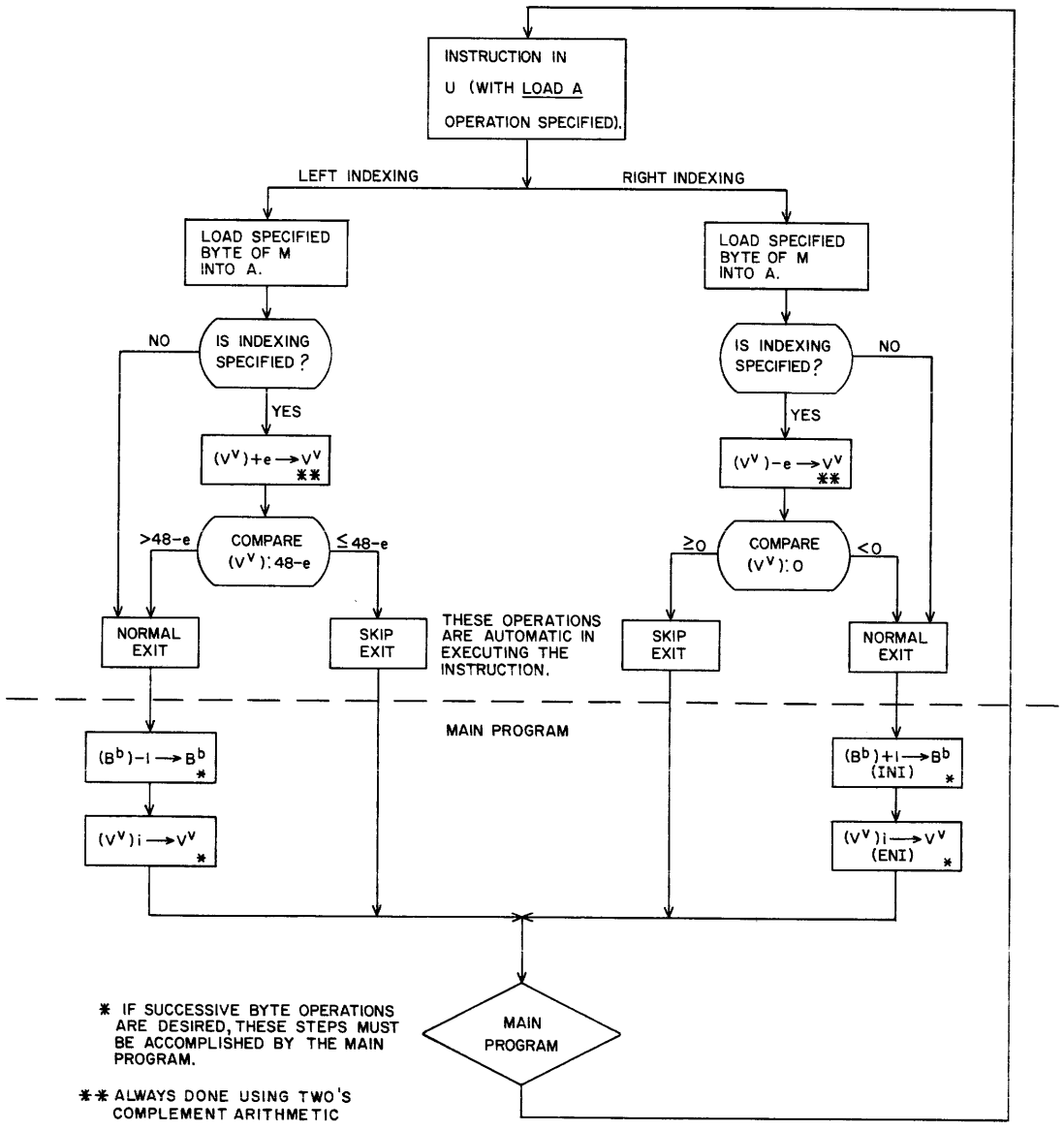


Figure 3-7. Load Byte Operation

Store

1) The source operand is the specified byte of the indicated register.

2) The destination is the specified byte of M.

3) Indexing (right or left) is accomplished in the same manner as in the load operation.

Scan

s^0 - s^4 values for scan operation:

Designator	Operation
s^4	. . . . Not Interpreted . . . .
s^3	. . . . Not Interpreted . . . .
If s^2 - s^0 count is:	Operation is:
000	Scan until m = (Q) or (A) = 0
001	Scan until m > (Q) or (A) = 0
010	Scan until m < (Q) or (A) = 0
011	Scan until m ≠ (Q) or (A) = 0
100	Scan until m ≤ (Q) or (A) = 0
101	Scan until m ≥ (Q) or (A) = 0
110	Equivalent to operation when s = 100
111	Equivalent to operation when s = 101

The following rules apply to comparisons in the scan operation:

a) If byte size is 48 bits, signed quantities are compared.

b) If byte size < 48 bits, magnitude quantities are compared.

The scan operation is:

1) Test (A) for 0 (A holds the number of bytes to be scanned). If (A) = 0, a normal exit is taken.

2) If A ≠ 0, read M from storage.

3) Compare byte of Q with specified byte of M.

4) If the condition is met, the operation is complete, and a skip exit is taken.

5) If the condition is not met, reduce (A) by one. Again test (A) for 0. If (A) = 0, a normal exit is taken. If (A) ≠ 0, index.

6) Indexing is right indexing only. Decrementing is accomplished in the following manner:

a) Subtract 'e' (byte-size) from (V^v); V^v now holds decremented byte address in M.

b) Examine (V^v). IF (V^v) ≥ 0, return to step 3. If (V^v) < 0 bytes of M have been scanned, and a new storage word must be obtained.

c) Increment (B^b) by and place in (B^b).

d) Place (48-e) in V^v. Leftmost byte of new M can now be referenced.

e) Return to step 2.

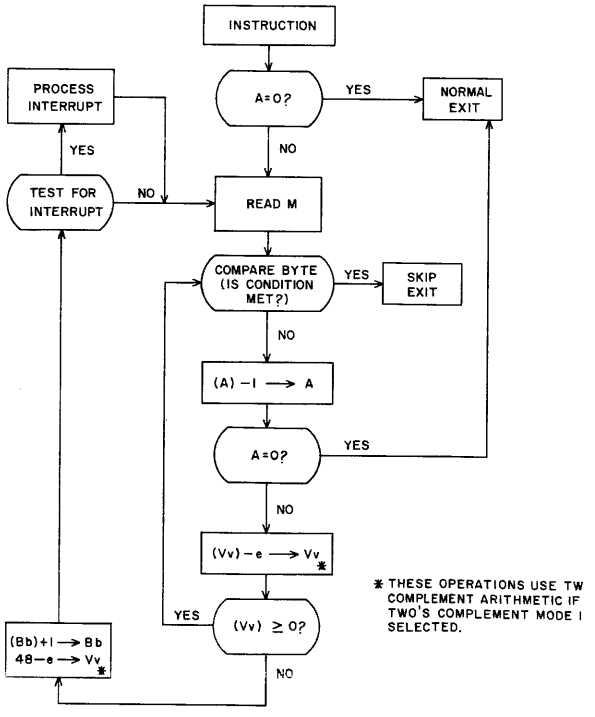
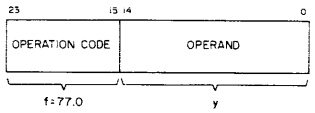


Figure 3-8. Byte Scan Operation

Internal Function (*INF*)(77.0)

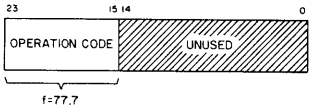


The Internal Function instruction establishes the internal operating mode or executes the operation specified by the function code 'y'. These codes (values) for 'y' and their designated functions are tabulated below.

Code	Function	Comments
00001	Clear Shift Fault Interrupt	Clear the designated bit of the Interrupt register.
00002	Clear Divide Fault Interrupt	
00003	Clear Exponent Overflow Interrupt	
00004	Clear Exponent Underflow Interrupt	
00005	Clear Arithmetic Overflow Interrupt	
00007	Clear Internal Reject Interrupt	
00011	Clear Real Time Clock Interrupt	
00012	Clear Storage Reference Fault Interrupt	
00013	Clear 1604 Mode Interrupt	
00014	Clear Trace Mode Interrupt	
00015	Clear Bounds Interrupt	
00016	Clear Illegal Instruction Interrupt	
00017	Clear Operand Parity Error Interrupt	
00020	Clear Manual Interrupt	
00021	Clear All Internal Interrupts	Clears the entire Interrupt register.
00022	Set Interrupt Active	Sets the Interrupt Active FF which enables the Interrupt system. Interrupts will now occur if a particular bit in the Interrupt Mask register is set when the interrupt condition occurs.
00025	Clear Interrupt Active	Clears the Interrupt Active FF. Regardless of the condition of bits in the Interrupt Mask register, entrance into the interrupt routine does not occur when the interrupt system is inactive.
00026	Set I/O Illegal Instruction FF	Sets the I/O Illegal Instruction bit. Interrupt will occur when: a.) an input/output instruction (74.0 - 74.6) is to be executed, and b.) the Illegal Instruction Mask bit is set, and c.) the Interrupt system is active. When Interrupt is entered, the Illegal Instruction bit in the Interrupt register is set. This feature protects I/O operations done under a Monitor or Master Control program.

Code	Function	Comments
00027	Clear I/O Illegal Instruction FF	When the I/O Illegal Instruction FF is cleared, execution of an Input/Output instruction does not set the Illegal Instruction bit in the Interrupt register.
00030	Select Two's Complement Mode	Entering Two's Complement mode results in performing all address modification, indexing instructions, and indexing in the Byte instruction in two's complement notation.
00031	Select Negate BCD Conversion	Selecting Negate BCD conversion sets the Negate BCD Conversion FF and transmits a Negate BCD Conversion signal to the external equipments, inhibiting BCD conversion.
00032	Release Selection of Negate BCD Conversion	Clearing the Negate BCD Conversion FF removes the inhibit on BCD conversion in the external equipments if the 1604 Mode bit in the Interrupt Mask register is also clear.
00040	Clear Two's Complement Mode	All index arithmetic is accomplished in one's complement notation when Two's Complement mode is cleared.
00100 through 07700	Reserved for Direct Interrupt	

Fault (77.7)



The Illegal Instruction bit in the Interrupt register is set by the Fault instruction if:

a) the Interrupt sequence is entered because 1604 mode is selected and the Interrupt system is active, and
b) the Illegal Instruction bit in the Interrupt Mask register is set.

If 1604 mode is not selected or if the Interrupt system is inactive, the Fault instruction is a pass instruction.

CHAPTER IV
INTERRUPT SYSTEM

The 3600 interrupt system provides for testing whether certain conditions (internal or external) exist without having these tests in the main program. Examples of these conditions are faults (internal) and end of operation (in an external equipment). After executing each main program instruction, a test is made for these conditions. If one of the conditions exists, execution of the main program halts. The contents of the Program Address register, P, and the bank designators are stored and an interrupt routine is initiated. This interrupt routine takes the necessary action for the condition and then jumps back to the next unexecuted main program step.

For each condition that can cause an interrupt, the program has two alternatives. It may select an interruptible condition, such that interrupt occurs when that condition occurs or it may choose to have the interrupt system ignore the condition. This is accomplished by not selecting interrupt on the condition. The program also has the choice of whether the interrupt system is to be used. The Internal Function instruction activates or deactivates the interrupt system.

WHEN COMPUTER MAY BE INTERRUPTED

The computer main program may be interrupted at the following times:

1) After reading an instruction from storage, but prior to executing that instruction.

2) Between upper and lower instructions of a program step, except for two cases:

   a) Interrupt is not recognized immediately after executing an Augment instruction (Single or Double Precision Augment). If an interrupt condition occurs, the main program will be interrupted after the augmented instruction is complete.

   b) Interrupt is not recognized between the upper and lower instructions specified by the Execute instruction. If an interrupt condition occurs, the main program will be interrupted after the instruction pair has been executed.

3) Between levels of multi-level indirect addressing. (If an interrupt occurs between levels of multi-level indirect addressing, the instruction involving this operation will begin over again when the main program resumes.)

4) Between iterations of instructions of considerable length. Examples of this type of instruction are the Searches and an Augmented Transmit instruction. If an interrupt occurs between iterations 'n' and 'n+1' of such an instruction, the 'n + 1' iteration will begin when the main program resumes. Thus, interruption does not necessitate repeating the entire instruction.

5) When the computer is operating in 1604 mode, before executing an instruction with operation codes (in octal) 00, 62, 63, 74, and 77. Upon return to the main program, the unexecuted instruction will be executed unless provision to skip that instruction was made in the interrupt routine.

6) In Trace mode, during the execution of a Jump instruction in which the Jump condition(s) is met but before the jump is taken. When the main program resumes, the Jump instruction which is interrupted will be executed, unless the interrupt routine provides for returning elsewhere.

LOGICAL DESCRIPTION OF INTERRUPT SYSTEM

Four registers are directly involved in the 3600 interrupt system. These are:

1) a 48-bit Interrupt register

2) a 48-bit Interrupt Mask register

3) a 48-bit Main Product register, and

4) an 11-bit Channel Product register

The bits in these registers are numbered from right to left in ascending order. The rightmost bit is numbered zero, and the leftmost bit (in a 48-bit register) is numbered 47.

Each of the 48 bits of the register is associated with a particular internal or external condition. This bit association is identical in all three 48-bit registers. Bits 00 through 04 and 07 through 15 are associated with internal conditions and bits 05 through 06 and 16 through 47 are associated with external conditions. The specific condition associated with each bit is as follows:

Bit	Association	Bit	Association
00	Shift Fault	24	Data Channel 10
01	Divide Fault	25	Data Channel 11
02	Exponent Overflow Fault	26	Data Channel 12
03	Exponent Underflow Fault	27	Data Channel 13
04	Arithmetic Overflow Fault	28	Data Channel 14
05	Interrupt 1 } See Direct	29	Data Channel 15
06	Interrupt 2 } Interrupt	30	Data Channel 16
07	Internal Reject	31	Data Channel 17
08	Real Time Clock	32	Data Channel 20
09	Storage Reference Fault	33	Data Channel 21
10	1604 Mode	34	Data Channel 22
11	Trace Mode	35	Data Channel 23
12	Bounds Fault	36	Data Channel 24
13	Illegal Instruction Fault	37	Data Channel 25
14	Operand Parity Error	38	Data Channel 26
15	Manual Interrupt	39	Data Channel 27
16	Data Channel 00	40	Data Channel 30
17	Data Channel 01	41	Data Channel 31
18	Data Channel 02	42	Data Channel 32
19	Data Channel 03	43	Data Channel 33
20	Data Channel 04	44	Data Channel 34
21	Data Channel 05	45	Data Channel 35
22	Data Channel 06	46	Data Channel 36
23	Data Channel 07	47	Data Channel 37

Though the uses of these registers will be detailed later, brief statements concerning the registers follow:

Interrupt Register

Each of the internal conditions which can cause an interrupt is wired to a particular bit position of this register. The bit positions associated with external conditions receive an interrupt line from each of the 32 possible data channels.

Interrupt Mask Register

The programmer selects to have a given interrupt condition (internal) tested by setting the appropriate bit of the Interrupt Mask register. The upper 32 bit positions associated with the 32 possible data channels are always set to "1's".

Main Product Register

The Main Product register contains the logical product of the Interrupt register and the Interrupt Mask register. The logical product of the registers is the logical product of their corresponding bits. For example, if one considers the logical product of two 4-bit registers, the result is:

4-bit register	1010
4-bit register	1101
	1000 (their logical product)

An interrupt results from a "1" in a bit of the Main Product register. To set to "1" any bit in the Main Product register requires that the Interrupt Mask Register bit to be set to "1".

Channel Product Register

Associated with each of the 32 possible data channels is an 11-bit Channel Product register. Each data channel may have up to eight equipments attached to it. The 11-bit Channel Product register contains the interrupt lines from these equipments in the eight lower order bit positions (bits 0-7). The three higher order bits indicate an interrupt from the data channel as follows:

Bit 08 = "1"	Interrupt from channel on End of Chain.
Bit 09 = "1"	Interrupt from channel on Parity Error (Storage or I/O).
Bit 10 = "1"	Interrupt from channel on Control Word Parity Error.

The mask associated with the Channel Product register is always set to "1's".

PROGRAMMING THE INTERRUPT SYSTEM

If an interrupt is desired when one or more specific conditions arise, a number of preparatory steps must first be accomplished by the programmer. These steps are:

1) The interrupt system must be activated.

2) External and internal conditions to be tested must be selected.

3) An interrupt routine must be programmed to determine the cause(s) of interrupt and to process the interrupt condition(s).

Interrupt Mode

The interrupt system is activated by executing an Internal Function instruction (77.0 00022 - Set Interrupt Active). The interrupt system remains active until:

1) An Internal Function instruction (77.0 00025 - Clear Interrupt Active) is executed,

2) An internal master clear is performed, or

3) The system is automatically deactivated (temporarily) by entering the interrupt routine.

Selecting Interrupt

Selecting the interrupt condition to be tested is accomplished in two ways, depending upon the condition - internal or external.

Internal

Interrupts on internal conditions are selected by setting the associated bits in the Interrupt Mask register to "1". The internal conditions on which the computer can be interrupted are listed below according to their bit position in the Interrupt Mask register.

Bit	Internal Condition
00	Shift Fault
01	Divide Fault
02	Exponent Overflow
03	Exponent Underflow
04	Arithmetic Overflow Fault
05	Interrupt 1 } Direct
06	Interrupt 2 } Interrupt
07	Internal Reject Interrupt
08	Real Time Clock Interrupt
09	Storage Reference Fault
10	1604 Mode
11	Trace Mode
12	Bounds Fault
13	Illegal Instruction
14	Operand Parity Error
15	Manual Interrupt

Setting the particular bit or bits in the Interrupt Mask register to provide for interrupt may be accomplished by an Inter-Register or Bit Sensing instruction. These bits remain set until cleared by an internal master clear or by one of the above instructions. (Refer to the appendix on Interruptible Conditions and Faults).

Direct Interrupt

By means of cables, certain external devices (usually another computer or some specialized external equipment) may be connected directly to the 3604. This direct connection allows the external device to act upon the 3604 interrupt system without going through the normal input/output interrupt facility.

Signal lines for the direct interrupt facility are as follows:

1) Interrupt 1: Bit 05 of the Interrupt register in the 3604 is the Interrupt 1 line from the external device. A "1" in bit 05 indicates an interrupt condition has occurred in the external device. If the associated bit of the Interrupt Mask register is set, and the Interrupt system is active, interrupt will occur in the 3604.

2) Interrupt 2: Bit 06 of the Interrupt register in the 3604 is the Interrupt 2 line from the external device. A "1" in bit 06 indicates an interrupt condition has occurred in the external device. If the associated bit of the Interrupt Mask register is set, and the interrupt system is active, interrupt will occur in the 3604.

3) Stop Computer: The Stop Computer signal is transmitted by the external device to the 3604. When received by the 3604, the 3604 halts activity, just as though the Stop switch had been pressed.

4) Computer Running: The Computer Running signal is transmitted continuously to the external device when the 3604 is running.

5) Internal Master Clear: When the Internal Master Clear switch is pressed on the 3601 console, the 3604 transmits an Internal Master Clear signal to the external device.

6) External Master Clear: When the External Master Clear switch is pressed on the 3601 console or when the Clear Channel instruction is executed with the channel designator set to 40_8 (equivalent to an external master clear), the 3604 transmits an External Master Clear signal to the external device.

7) Data Cables (bits 30-35 of U^{000} register): When the 3604 executes an internal function instruction with codes 00100 through 07700 selected, bits 30-35 of the internal function code are transmitted to the external device via the data cables.

External

Interrupts on external conditions are selected in a manner different from internal conditions. Selection is not accomplished by setting the Interrupt Mask register bits, since the upper 32 bits of the Interrupt Mask register are automatically always set to "1's". To provide for an interrupt if a specific external condition arises, a Function instruction (refer to Input/Output section) must be executed to select interrupt on the condition. This condition remains selected until cleared by another Function instruction or an external master clear is performed.

Interrupt Routine

An interrupt routine must be programmed. The interrupt routine typically:

1) Stores the contents of the operational registers (including SCR if its contents are to be retained) if these registers are to be used in the interrupt routine.

2) Processes the interrupt condition(s).

3) Clears the condition causing the interrupt.

4) Reloads the operational registers upon completing the routine.

5) Returns control to the main program.

AUTOMATIC OPERATIONS IN THE INTERRUPT SYSTEM

Certain operations in the interrupt system occur automatically and require no intervention by the programmer. These automatic operations are:

1) Setting the particular bit in the Interrupt register when the particular condition arises (e.g., the Shift Fault bit in the Interrupt register is set when a shift fault has occurred). For a description of conditions which set the individual bits of the Interrupt register, refer to the appendix on Interruptible Conditions and Faults.

2) Examining the Main Product register as described on page 4-1 (if the interrupt system has been activated by the Internal Function instruction).

3) Halting the main program when interrupt occurs.

4) Deactivating the interrupt system when interrupt occurs (to prevent interrupts from occurring while in the interrupt routine).

5) When interrupt occurs, executing a wired-in return jump to address 0 00001, the entrance to the interrupt routine. At address 0 00000*, an Unconditional Bank Jump instruction (63.0) (s=0) or an Unconditional Jump to Lower instruction (63.1) is stored, depending on whether return is to the upper or lower instruction. The address of the interrupted main program step, P, is placed in the address portion of the instruction stored at address 0 00000. This instruction (63.0 or 63.1) provides for return to the main program (upper or lower instruction) after executing the interrupt routine.

Internal interrupt conditions may occur:

1) as a result of executing certain instructions, or

2) because of the nature of certain instructions, or

3) simply at random times without any specific relationship to instructions (e. g. Real Time Clock).

Depending on (1) which of the above three conditions caused an internal interrupt condition and (2) the position of the instruction being executed, the instruction stored at address 0 00000 is (63.0) (s = 0) or 63.1. The value of P stored in either of these instructions may be P or P + 1, depending upon the instruction and the time the interrupt routine was entered. Table 4-1 lists the various cases for each of the internal interrupt conditions.

* The first digit refers to the storage bank; the remaining five digits refer to the storage address within that bank.

Table 4-1. Instructions stored on Internal Interrupt Conditions

Internal Condition Causing Interrupt	Instruction Upper/Lower?	Bank Jump Instruction Stored At Address 0 00000.	Value Of Program Address Register In Jump Instruction Stored At Address 0 00000.
Shift Fault*	upper	63.1	P
	lower	63.0	P + 1
Divide Fault*	upper	63.1	P
	lower	63.0	P + 1
Exponent Overflow*	upper	63.1	P
	lower	63.0	P + 1
Exponent Underflow*	upper	63.1	P
	lower	63.0	P + 1
Arithmetic Overflow*	upper	63.1	P
	lower	63.0	P + 1
Internal Reject*	48-bit	63.0	Reject jump address "n"
Storage Reference*[1]	upper	63.0	P
Fault	lower	63.1	P
	instruction fetch	63.0	P
1604 Mode	upper	63.0	P
	lower	63.1	P
Trace Mode*	upper	63.0	P
	lower	63.1	P
Bounds Fault*[1]	upper	63.0	P
	lower	63.1	P
Illegal Instruction	upper	63.0	P
	lower	63.1	P
Operand Parity*	upper	63.1	P
Error	lower	63.0	P + 1
Asynchronous Interrupts:			
Real Time Clock Interrupt		63.0 or 63.1	P
Manual Interrupt		63.0 or 63.1	P
Direct Interrupt		63.0 or 63.1	P
External Interrupts		63.0 or 63.1	P

1 Whenever a Bounds Fault interrupt or Storage Reference Fault interrupt occurs: (a) while executing an augmented instruction, or (b) while executing the lower instruction of the pair specified by the Execute instruction, the following occurs:

a) a 63.0 instruction is stored at address 0 00000, and

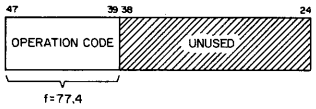
b) the value of P is stored in the jump instruction at address 0 00000.

*Internal interrupt conditions flagged by an asterisk are conditions which arise during the execution of certain instructions (i.e., instruction execution causes the condition). Internal interrupt conditions not flagged occur because of the nature of certain instructions, or simply occur at random times as asynchronous interrupts.

INTERRUPT INSTRUCTIONS

Two interrupt instructions used in the interrupt routine facilitate determining the cause(s) of the interrupt. These instructions are explained below.

Main Product Register Jump (MPJ)(77.4)

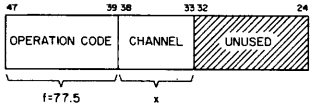


This instruction, restricted to the upper instruction position of a program step, scans the Main Product register from left to right; that is, from bit 47 to bit 0. If a non-zero bit is detected, a jump is executed to the location (P+i+1). P is the current program address and 'i' is the location of the non-zero bit in the Main Product register.* If the Main Product register is zero, the next instruction is executed.

Thus, this instruction determines which channel (a non-zero bit in one of the upper 32 bit positions), or internal condition (a non-zero bit in one of the lower 16 bit positions) caused an interrupt.

If a non-zero bit is detected in one of the upper 32 bit positions (indicating an interrupt on channel 'x'), it is necessary to determine which equipment on that channel caused the interrupt. The Channel Product Register Jump instruction, described below, is used for this purpose.

Channel Product Register Jump (CPJ)(77.5)



This instruction, restricted to the upper instruction position of a program step, enables the designated Channel Product register into the main computer to be scanned. Scanning is from left to right; that is, from bit 10 to bit 0. If a non-zero bit is detected, a jump is executed to location (P+i+1). (P is the current program address and 'i' is the location of the non-zero bit in the Channel Product register.)** If the Channel Product register is zero, the next instruction is executed.

Thus, after determining the channel on which interrupt has occured (via the Main Product Register Jump instruction), the Channel Product Register Jump instruction determines which equipment, equipments, or condition caused the interrupt.

To determine the particular condition within the equipment causing the interrupt, a Copy Status instruction must be executed (refer to Input/Output section).

INTERRUPT PROCESSING

During execution of the main program (if the interrupt system has been activated by the Internal Function instruction), the Main Product register is automatically examined as outlined on page 4-1. If, in this examination, a non-zero bit is detected in the Main Product register, interrupt occurs.

When an interrupt occurs, the main program is halted and a previously programmed routine of instructions (interrupt routine) is performed. Entrance to this interrupt routine is accomplished by an automatic wired-in unconditional return jump to address 0 00001. The following automatic operations also occur:

1) An Unconditional Bank Jump instruction (63.0) (s=0) or Unconditional Jump to Lower instruction (63.1) is stored at address 0 00000. Values for the operand and instruction bank designators and P (current address) are set in this instruction such that a return may be effected to the storage bank and current address of the main program at the time interrupt occurred. (The 63.0, s=0 instruction is stored if return is to the upper instruction; the 63.1 instruction is stored if return is to the lower instruction.)

---

* 'i' - the location of the particular bit within the register. Though the register is numbered from right to left, scanning is from left to right. The first bit scanned is bit 47. If a "1" exists in bit 47, i= 47, and a jump is effected to [P+ 47+ 1] (decimal notation).

** 'i' - the location of the particular bit within the register. Though the register is numbered from right to left, scanning is from left to right. The first bit scanned is bit 10. If a "1" exists in bit 10, i= 10, and a jump is effected to [P+ 10+ 1] (decimal notation).

2) Operand Bank and Instruction Bank registers are set to bank "0".

3) The interrupt system is deactivated; that is, examination of the Product register is no longer automatic, but is program controlled.

Once the entrance to the interrupt routine is made, processing begins to determine the type (internal or external) and cause of the interrupt. Since processing internal and external interrupt conditions differs, each will be examined in detail.

Internal Interrupt Processing

Refer to figure 4-1.

The following description of internal interrupt processing outlines pertinent facts necessary for a satisfactory interrupt program. In outlining the operation of interrupt processing, a list of steps for a sample interrupt routine is given. Steps flagged by a single asterisk indicate those steps which are essential to processing the interrupt. Otherwise, no constraints are placed on the interrupt routine.

NOTE

Since the interrupt system is inactive while in the interrupt routine, neither Bounds checking nor Trace mode are in effect to constrain the interrupt program. In step 7, execution of the Bank Jump instruction at address 0 00000 to return to the main program is accomplished in the interrupt routine. If this jump is taken to an address out of bounds, an out-of-bounds indication is not revealed until an attempt is made to read the first instruction at the jump address.

Steps in Interrupt Routine (Sample)

*1) Enter interrupt routine at address 0 00001.

2) Execute subroutine to store contents of Shift Count register and operational registers.

*3) Test the Main Product register. This may be accomplished by:

a) Main Product Register Jump instruction,

b) Bit Sensing instruction, or

c) Other logical instructions.

4) Execute subroutine to process particular interrupt condition.

*a) Clear the interrupt condition.

*5) Test for further interrupt conditions. This test may be accomplished by steps 3a, 3b, or 3c.

6) Execute subroutine to restore contents of operational registers including Shift Count register.**

*7) Execute jump instruction at address 0 00000 to return to main program.

** Restoring the normalize count in the SCR before returning to the main program may be accomplished as follows:

Since the SCR cannot be written into directly by the Inter-Register instruction, etc., restoring the normalize count in the SCR must be accomplished via a shift instruction. Before restoring the contents of A or Q, shifting A or Q right by k (where k is the normalize count) places k in the SCR.

Sample Interrupt Program

Address	Contents	Comments
0 00000	63.0 or 63.1 instruction	Instruction stored when interrupt occurred; address portion of instruction is P (address of main program). Executing this instruction tests the Product register: if non-zero, goes to 0 00001; if zero, jumps to P and activates interrupt system.
0 00001 ↓	Subroutine	Subroutine stores contents of operational registers, including Shift Count register.
X XXXXX ↓	MPJ instruction in upper; Jump instruction to Y YYYYY is lower.	Test Product register; if Product register zero, jump to Y YYYYY.
P+i+1	→	Jump to subroutine for processing interrupt on shift fault.
. . . . .	Jump instructions	
P+i+1	→	Jump to subroutine for processing interrupt on channel 32_{10}.
Y YYYYY . . .	Subroutine	Subroutine restores operational registers.
Z ZZZZZ	Jump instruction	Jump to address 0 00000 to return to main program.

The operation of steps in the interrupt routine is detailed below:

1) The interrupt routine is automatically entered at address 0 00001.

2) If interrupt occurs immediately following a Floating Point instruction, it is often desirable to preserve the normalize count (held in the Shift Count register). To prevent destroying the contents of the SCR, the following may be performed:

   Upon entering the interrupt routine, the SCR must be stored prior to executing the Main Product Register Jump instruction. Executing the MPJ instruction changes the contents of the SCR; this precludes the use of address 0 00001 for the MPJ instruction. The subroutine for storing the SCR (and other operational registers) may begin at address 0 00001 or a jump may be taken elsewhere from address 0 00001 to this subroutine. After storing the SCR, the MPJ instruction may then be executed.

   Instructions used to store the contents of the Shift Count register must be instructions which, in their execution, do not harm the contents of the SCR. Listed in table 4-2 are all instructions which, in their execution, destroy the contents of the Shift Count register.

Table 4-2. Instructions Which Destroy Contents of SCR

All Shift Instructions	Register Jump
A Jump, Q Jump, Selective Jump, and Selective Stop with b=4-7	Transmit with s=4, 5, 6, or 7
Scale Instructions	Locate List Element
Storage Skip	Search Order with s=0, 1, 2, or 3
Storage Shift	Byte
Selective Set	Bit Sensing
Selective Clear	All 1604 Type Search Instructions
Selective Complement	All Replace Instructions
Selective Substitute	Main Product Register Jump
Load Logical	Channel Product Register Jump
Add Logical	D Register Jump
Subtract Logical	Multiply Integer and Multiply Fractional
Load Index Upper	Divide Integer and Divide Fractional
Load Index Lower	All Floating Point Instructions
Store Index Upper	Add to Exponent
Substitute Address Upper	Inter-Register Swap

3) Executing the Main Product Register Jump instruction scans the Main Product register to test for a non-zero bit. The location of a non-zero bit is designated by 'i'* and indicates a specific condition (internal or external - refer to the register description) is present which caused the interrupt.

   a) If a non-zero is detected, a jump is executed to address [P+i+1]. Since the Main Product register is 48_{10} bits, 'i' may have 48_{10} values. In order to branch to the individual subroutines to process each interrupt condition, the programmer must previously have stored jump instructions in the first 48_{10} storage locations following the location of the Main Product Register Jump instruction.

   b) The jump address specified by the jump instruction at [P+i+1] is the starting location of the subroutine to process the particular interrupt condition.

   c) Within the appropriate subroutine, the programmer must clear the interrupt condition, or the subroutine is re-entered when the Main Product register is again scanned. Clearing the interrupt condition may be accomplished by the Internal Function instruction (refer to the Faults section in the appendix).

4) Upon completing this subroutine, a jump is taken to either address 0 00000 or the location of the Main Product Register Jump Instruction.

   a) If return is to address 0 00000, the Bank Jump instruction stored there tests the Main Product register. If non-zero, the Jump instruction becomes a pass instruction and the Main Product Register Jump instruction is executed. If zero, a jump is effected back to the main program.

* 'i' - the location of the particular bit within the register. Though the register is numbered from right to left, scanning is from left to right. The first bit scanned is bit 47. If a "1" exists in bit 47, i= 47, and a jump is effected to [P+ 47+ 1] (decimal notation).

b) If return is to the address containing the Main Product Register Jump instruction, this instruction is again executed to scan for further interrupt conditions. When the Main Product register is zero (no further interrupt conditions to process), the next instruction executed is a jump instruction. This jump may be to a subroutine to restore the contents of operational registers or to address 0 00000.

5) Returning to the main program via the jump instruction at address 0 00000 automatically activates the entire interrupt system again to permit the automatic testing of interrupt conditions.

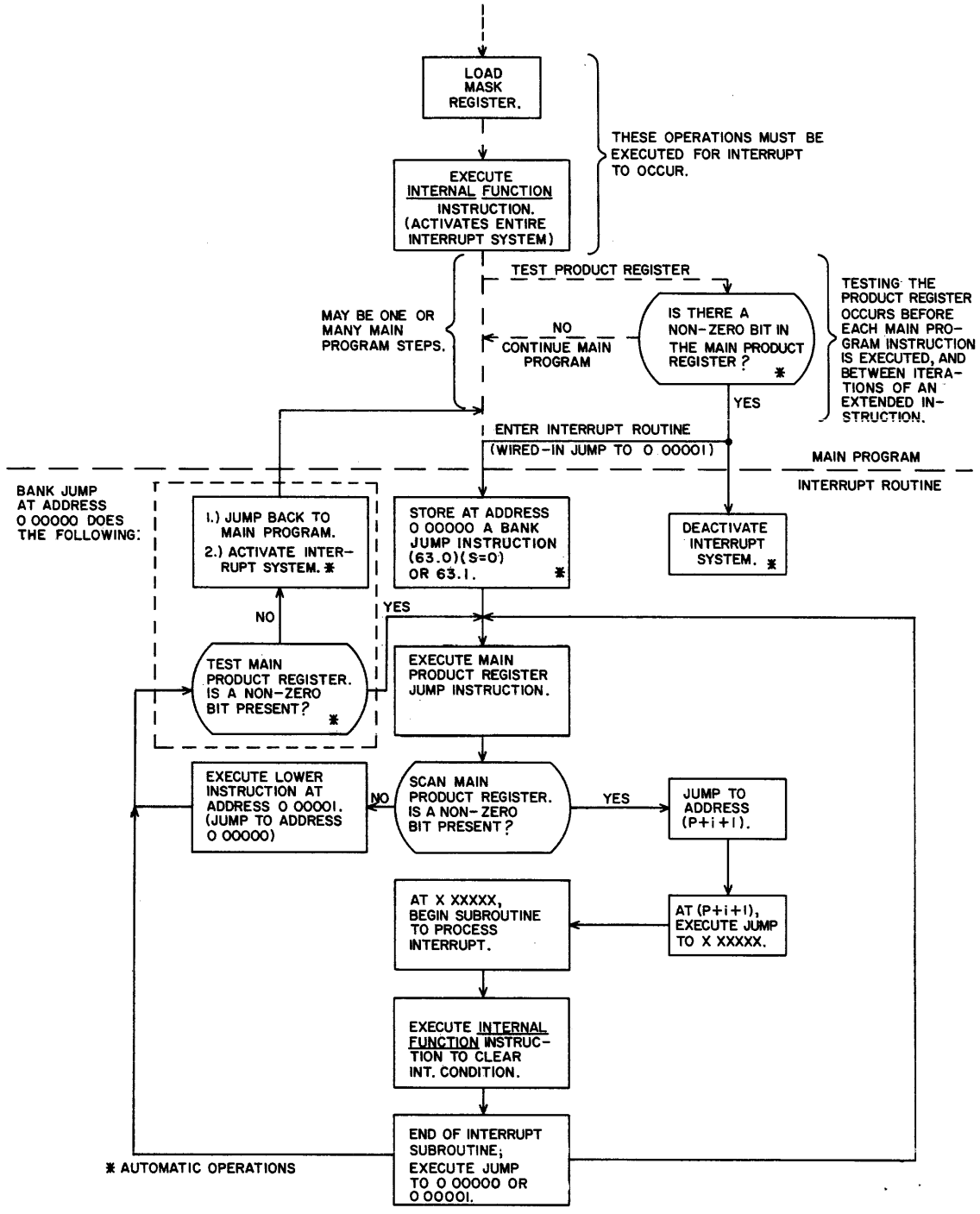


Figure 4-1. Typical Internal Interrupt Processing

External Interrupt Processing

For each external equipment, external interrupt conditions are recognized only on one of the following three situations:

1) Interrupt on ready and not busy

2) Interrupt on end of operation

3) Interrupt on abnormal end of operation

Since the meaning of these three situations depends on the external equipment, refer to the reference manual for that equipment.

Two external interrupt conditions are recognized within the data channel itself. These are interrupt on end of chain and interrupt on error (refer to the description of the Function instruction).

The preliminary steps in processing external interrupt conditions occur as described in the Internal Interrupt Processing section. The following steps are directly pertinent to processing external interrupt conditions (figure 4-2):

1) The Main Product Register Jump instruction is executed. A non-zero bit detected in one of the upper 32 bit positions indicates an interrupt condition is present on one of the 32 possible data channels. Thus, executing this instruction isolates the interrupt condition to a specific channel.

   a) Detecting a non-zero bit effects a jump to address [P+i+1]. Address [P+i+1] contains a previously stored Jump instruction to the starting location of the particular external interrupt subroutine.

2) At the jump address, a Channel Product Register Jump instruction has been previously stored. Its channel designator 'x' must have a value corresponding to the channel number associated with bit "i". (Refer to the Channel Product Register Jump instruction description.)

   a) The Channel Product Register Jump directs its 6-bit channel designator 'x' to the specified channel attached to the communication module.

   b) An 11-bit Channel Product register (one bit for each equipment on that channel and one bit for each of the other interrupt conditions) is enabled into the computer to be scanned by this instruction.

   c) Detecting a non-zero bit in the Channel Product register causes a jump to [P+i+1]. The source of the interrupt has now been isolated to a particular equipment or a particular condition on a particular channel.

3) The address [P+i+1] is the starting location of the subroutine to process the particular interrupt condition.

4) To determine the particular condition causing the interrupt in the equipment, the programmer must execute a Copy Status instruction in this subroutine (refer to the Input/Output section).

5) As in internal interrupt processing, the interrupt condition must be cleared within the subroutine to prevent re-recognition of that condition when the Channel Product register is again scanned. Clearing the interrupt condition is accomplished by a Function instruction. Clearing a particular interrupt condition may not necessarily clear the bit in the Interrupt register, since more than one interrupt condition may exist at one time on the same channel.

6) Upon completion of the subroutine, the Channel Product register is again scanned. If no further interrupt conditions are present, a return is effected to address 0 00000 after restoring the contents of the operational registers.

7) The Main Product register is again scanned. If there are no further interrupt conditions to process, a return is effected to the main program via the same steps taken for internal interrupt processing.

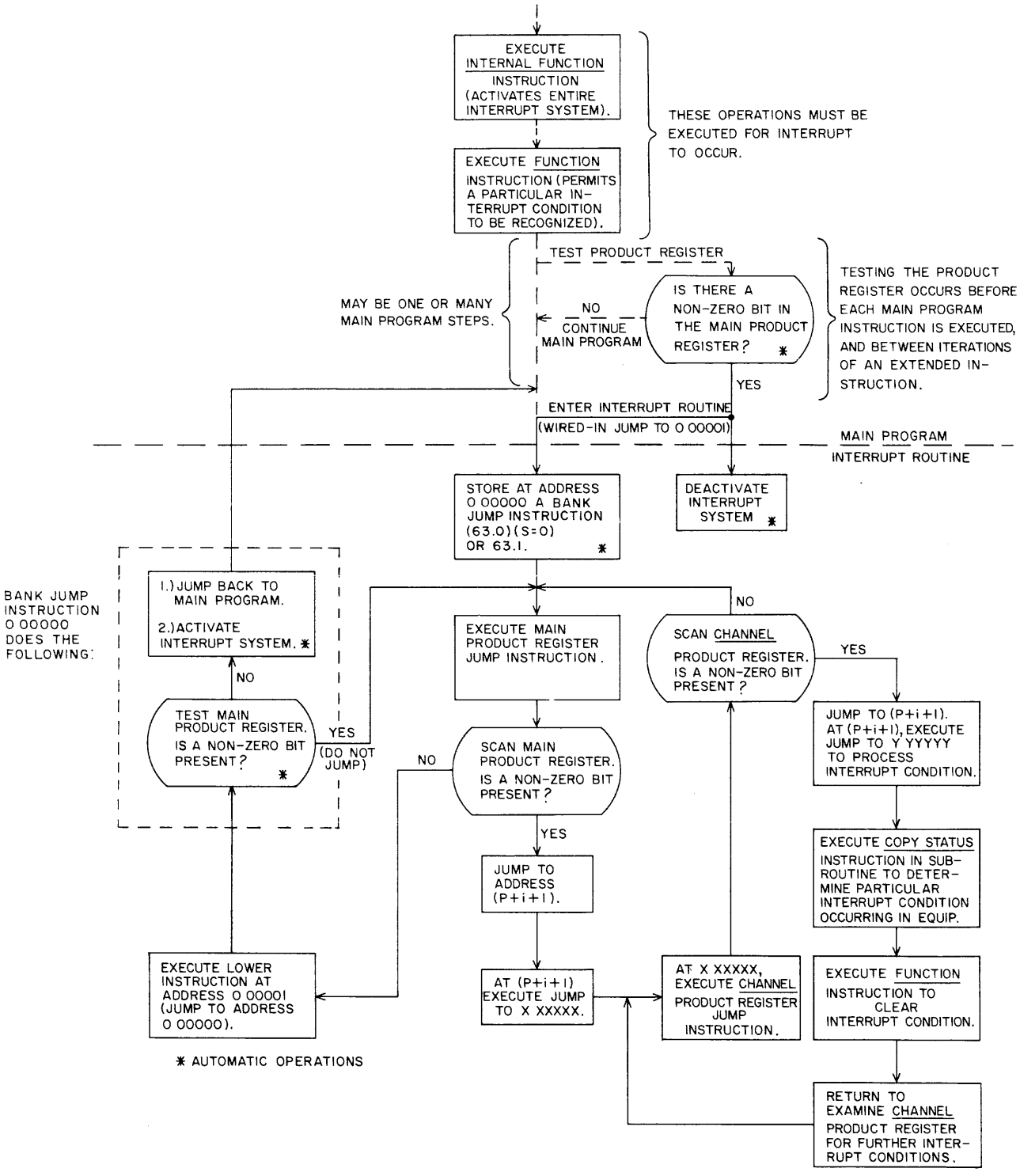


Figure 4-2. Typical External Interrupt Processing

CHAPTER V
INPUT/OUTPUT

Input/output facility for the 3600 system is provided by two modules: the 3602 communication module and the 3606 data channel.

These two modules operate conjunctively and are both located in a data interchange cabinet. Together they provide the method for bi-directional data exchange and for proper control of information transmission between the 3600 system and its various external equipments.

A simplified block diagram of a 3600 system is shown in figure 5-1. For purposes of illustration, the system in figure 5-1 shows only one 3602 communication module and one 3606 data channel. Block diagrams of the communication module and data channel are shown in figures 5-2 and 5-3.

A basic 3600 system includes one 3602 communication module and four 3606 data channels. Additional data channels, up to a total of eight, may be added. For maximum I/O capability, a 3600 system may include up to 32 bi-directional data channels. This is obtained by expanding the basic system to its limit of four communication modules with each controlling eight data channels.

A 3606 data channel may control a maximum of eight external equipments. Typical external devices are line printers, punched card equipment, and magnetic tape equipment. The Connect Instruction selects the equipments individually to communicate with the 3600 system via the data channel and communication module.

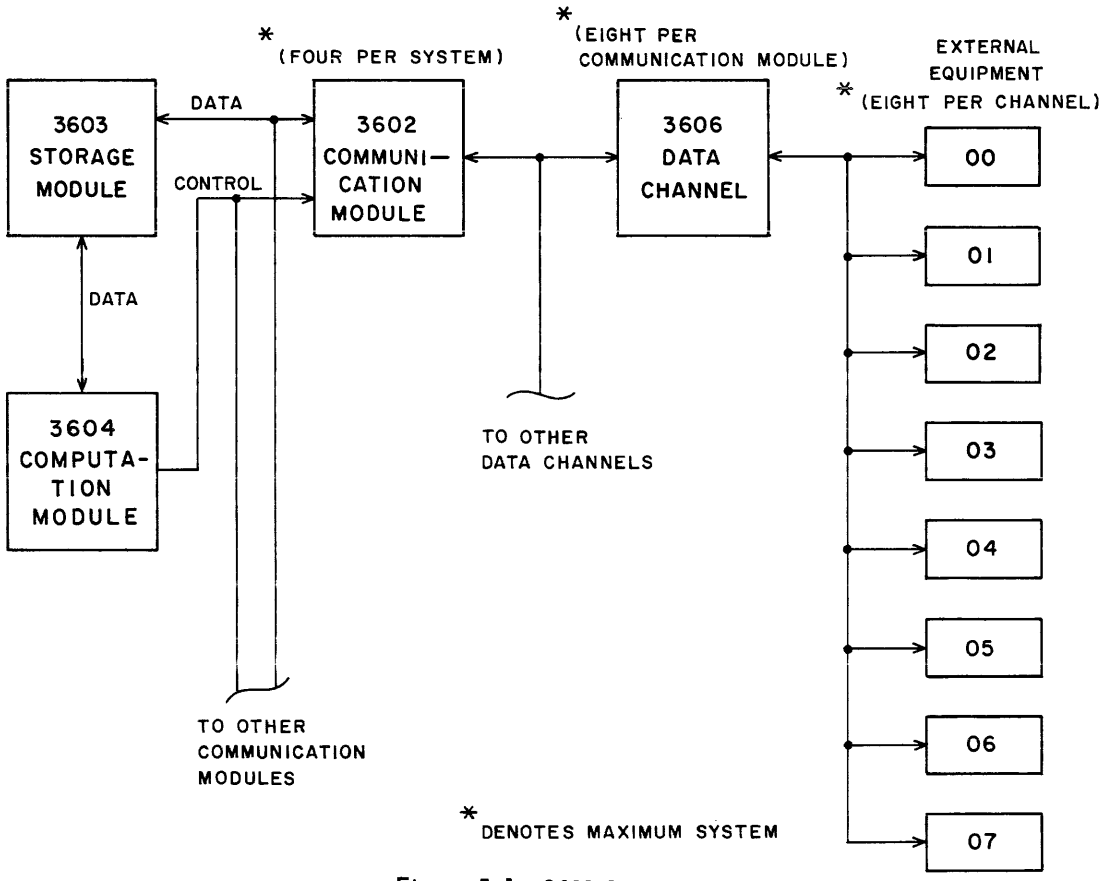


Figure 5-1. 3600 System

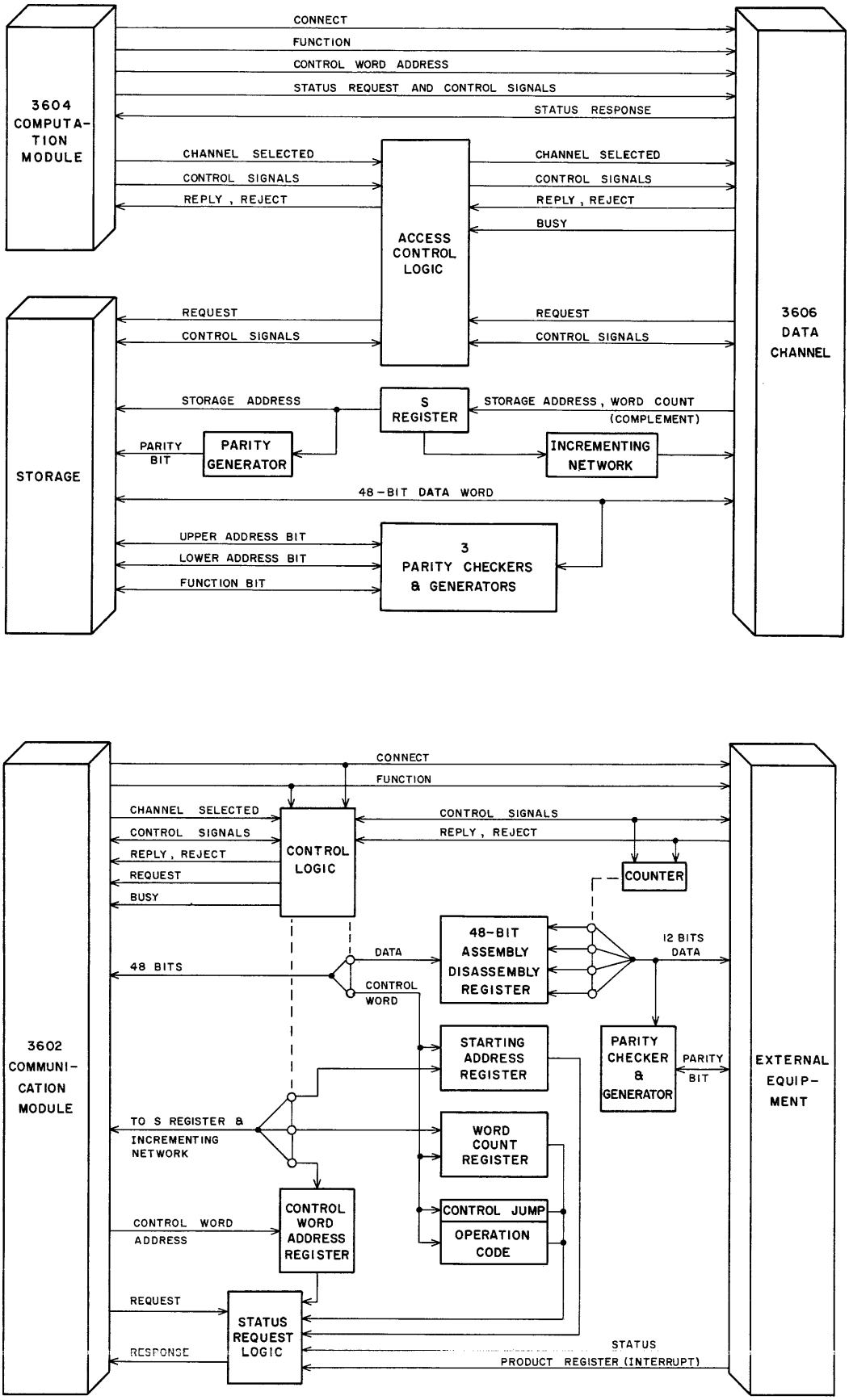


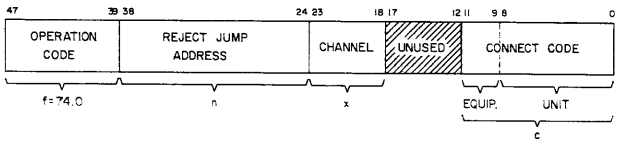
Figure 5-3. 3606 Data Channel

INPUT OUTPUT INSTRUCTIONS

Seven instructions govern input/output operations in the 3600 system. These instructions establish operating modes within external equipments and provide for transferring data between the storage and communication modules. Provision is made for sampling the status of operating conditions in the external equipments, and for monitoring the progress of data transmission.

The Connect instruction must be used to connect the external equipment to the 3600 system before data transmission can be initiated. If various operating conditions within the connected equipment are to be specified, a Function instruction is executed. After initial operating conditions have been established, the Read or Write instruction may be executed to transmit data into or out of the system automatically and independent of the main program. In addition, the Read or Write instruction specifies the address of the control word which contains all other information necessary to perform the operation; i.e., starting address, word count, and operation code. While input/output operations are in progress, the status of data transmission or operating conditions within the external equipments may be sampled by using the Copy Status instruction. The Channel Clear instruction is used to clear the designated channel at the start of a program or in the event of a program failure. For direct entry of keyboard or punched card information into the 3604, the Input to A instruction is used.

Connect (CONN) (74.0)



An external equipment is connected to the system via a data channel and communication module. The upper octal digit of the channel designator 'x' specifies one of four possible communication modules. The lower digit of 'x' specifies one of the eight possible data channels attached to the communication module. Data channels are numbered in octal as:

Communication Module 0	Channels 0 - 7
Communication Module 1	Channels 10 - 17
Communication Module 2	Channels 20 - 27
Communication Module 3	Channels 30 - 37

Up to eight equipments may be attached to each data channel. The desired equipment is specified by the upper octal digit of the Connect code 'c'. The equipments are numbered 0 through 7 (octal). Each equipment, via an eight position switch located on the equipment, may assume any one of the eight possible equipment codes.* Certain equipments, such as magnetic tape controllers, control more than one unit. A number specifies each unit. Unit numbers are the three lower octal digits of 'c' and may range from 000_8 to 777_8. Normally, the legitimate range of unit numbers is from 000_8 to 017_8.

The Connect instruction connects an equipment and/or unit to the computing system by specifying:

1) The 6-bit channel code (one of 32 channels divided among four communication modules).

2) The 3-bit equipment code (one of eight equipments assigned to the selected channel).

3) The 9-bit unit code, if any (one of 512 possible units; 16 possible units in all ordinary systems).

When the Connect instruction has been executed:

1) The Function instruction may be used to establish various operating modes within the equipment, if desired, and

2) The Read or Write instruction may be executed to transmit data from or to the designated equipment.

The external equipment remains connected until another Connect instruction is executed for the same channel, or the Clear Channel instruction is executed. Only one equipment may be connected to a channel at any one time. All 32 channels may have an equipment connected, and all may be active at once. Thus, 32 equipments may be simultaneously communicating with the 3600 system.

Under certain conditions, it may not be possible to connect the designated equipment. When one or more of these conditions occurs, a Reject signal is sent to the 3604, and a jump is effected to the address

* If the operator has selected the same octal equipment number for more than one equipment on a particular channel, a Connect instruction will connect each equipment. Subsequent Read or Write instructions will reference each connected equipment resulting in loss of data or in a program malfunction.

specified by the 15-bit reject jump address. This address will be within the bank specified by the Program Address Bank register.

A Reject signal will be sent to the 3604 only under one or more of the following conditions:

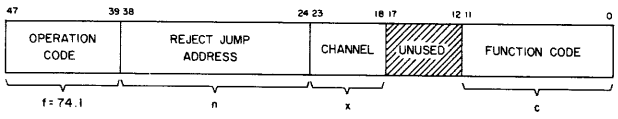
1) Channel Busy: The selected channel is currently performing a Read or Write operation.

2) Unit Unavailable: The unit referenced is in use by another data channel. This may occur only if an equipment is multi-channel (such as a magnetic tape controller).

3) False Reference: When the 3604 receives no response within 100 μsec, it generates its own Reject signal and performs the jump. This case may occur if the referenced equipment is not attached to the specified channel or the equipment is inoperative.

The program may determine whether a reject is internally or externally generated by examining bit 07 of the Interrupt register. This bit is set whenever an internal reject occurs.

Function (EXTF) (74.1)



The Function instruction specifies operating conditions within an external equipment or a condition on which interrupt will occur. This instruction contains a 12-bit Function code 'c' which specifies operating conditions. If the upper bit of channel designator 'x' = "0", the Function code 'c' is sent to the external equipment connected to channel 'x'. A list of codes for each external equipment is included in its associated chapter.

The Function code 'c' is sent to the data channel if the upper bit of channel designator 'x' = "1". The codes to which the channel will respond are as follows:

Bit 23
74.1 n 1x 0001 Stop Channel Activity

1. Selects data channel x.
2. Cuts off any I/O operation in progress.
3. Does not clear any registers.
4. Does not disconnect equipment.
5. Contents of the operating registers may be sampled by means of the Copy Status instruction. These register contents may be stored in the form of a Control Word. A Read or Write instruction containing the address of this new Control Word will cause the truncated I/O operation to resume at the point at which it was cut off.*

Bit 23
74.1 n 1x 0002 Select Interrupt on End of Chain

1. Selects data channel x.
2. Data channel interrupts main program in 3604 when current Read or Write operation is completed and no further chaining is specified.

Bit 23
74.1 n 1x 0004 Clear Interrupt on End of Chain

1. Selects data channel x.
2. Removes interrupt selection by clearing control FFs.

Bit 23
74.1 n 1x 0010 Select Interrupt on Error

1. Selects data channel x.
2. Data channel interrupts main program in 3604 upon:
   a. Storage reference fault
   b. I/O transmission parity error
   c. Storage address parity error
   d. Parity error on word from storage.

Bit 23
74.1 n 1x 0020 Clear Interrupt on Error

1. Selects data channel x.
2. Removes interrupt selection by clearing control FFs.

NOTE

An Interrupt signal from a data channel may be dropped using the Clear Channel instruction, the master clear facility, or by sending any Function code to the data channel (with bit 23 = "1"). The latter method does not remove the interrupt selection, allowing Interrupt to occur again the next time the condition is reached.

* In tape units, tape will move to the next inter-record gap after a Stop Channel Activity function. Thus a truncated I/O operation on tapes cannot resume where it was cut off.

If an equipment to which the Function instruction is directed has not been previously connected to the system via a Connect instruction, the Function code cannot be recognized and a Reject signal will be generated. The Reject signal causes the program to jump to the 15-bit reject jump address (within the bank specified by the 3-bit Program Address Bank register). The following conditions or combination of conditions will result in a reject:

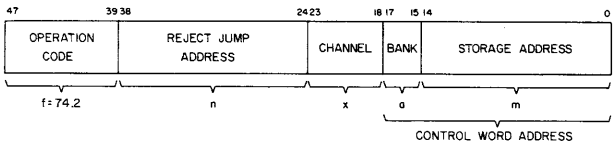
1) No Unit or Equipment Connected: The referenced device is not connected to the system and cannot recognize a Function instruction. If no response is received within 100 μsec, the Reject signal is generated automatically by the 3604.

2) Illegal Code: The Function code 'c' cannot be interpreted by the specified device. The Reject signal is generated by the external equipment after the attempted reference.

3) Equipment or Unit Busy or Not Ready: The device cannot perform the operation specified by 'c' without damaging the equipment or losing data. For example, a Write End of File code will be rejected by a tape unit if the tape unit is rewinding.

4) Channel Busy: The selected data channel is currently performing a Read or Write operation.

Read (BEGR) (74.2)



The Read instruction initiates input activity on channel 'x'. The 18-bit control word address 'a m' is transmitted to the data channel.

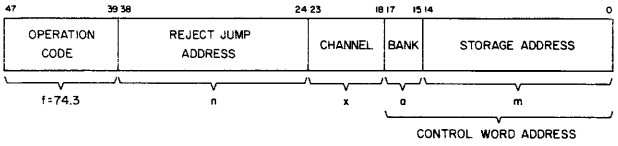
Unless a Reject signal is generated, the main computer program proceeds independent of activity in the communication module. If a Reject signal is generated, a jump is effected to the 15-bit reject jump address (located in the storage bank specified by the Program Address Bank register). A Reject signal is generated by a Channel Busy (the designated channel is currently performing a Read or Write operation).

If no Reject signal is given, the control word (see page 5 - 8) is fetched from the storage address designated by 'a m'. The control word and its associated 18-bit address are placed in their respective registers in data channel 'x'.

This activity occurs simultaneously with main computer program activity. The communication module and the designated data channel control all activity until the channel becomes inactive.

The diagram in figure 5-4 represents Read and Write operations in flow chart form.

Write (BEGW) (74.3)



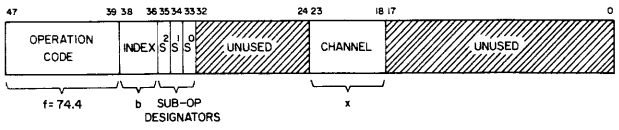
The Write instruction initiates output activity on channel 'x'. The 18-bit control word address 'a m' is transmitted to the data channel.

Unless a Reject signal is generated, the main computer program proceeds independent of activity in the 3602. If a Reject signal is generated, a jump is effected to the 15-bit reject jump address (located in the storage bank specified by the Program Address Bank register). A Reject signal is generated by a Channel Busy (the designated channel is currently performing a Read or Write operation).

If no Reject signal is issued, the control word is fetched from the storage address designated by 'a m'. The control word and its associated 18-bit address are placed in the proper registers in channel 'x'.

This activity occurs simultaneously with main computer program activity. The 3602 and the designated data channel control all activity until the channel becomes inactive.

Copy Status (COPY) (74.4)



After a connection has been performed, the Copy Status instruction may be used to determine if the data channel is busy and to sample the status of the external equipment, the control word, and the control word address.

The status of the data channel and external equip-

ment may be examined via the 15-bit channel and equipment status response.

The external equipment issues a 12-bit status code (bits 00-11) to indicate operating conditions. (See individual chapters for list of status codes.) This code is present at all times on lines from the external equipment to the data channel to which it is connected. The upper bits are appended by the data channel to indicate the following:

Bit 14 = "1", Parity error has occurred.

Bit 13 = "1", Write operation is in progress.

Bit 12 = "1", Read operation is in progress.

The status of a data transmission may be determined by examining the word count and the current address. These quantities are held in registers in the data channel. Since the word count is reduced by one and the address is increased by one for each data word transmitted, the number of words processed in the operation may be determined at any time by examining these quantities. (The sum of the word count and the current address is always constant.)

During chaining operations, it may be necessary to examine the status of the control word address. Since the control word address is always advanced by one immediately after reading a new control word from storage, the Control Word Address register always contains the address of the next control word.

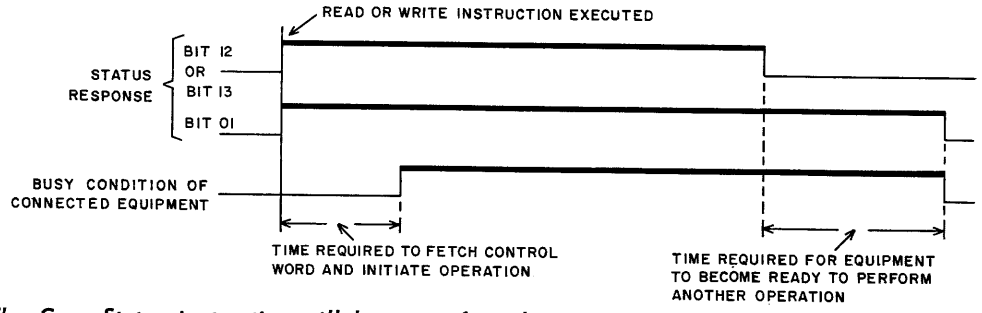
The Copy Status instruction determines if the data channel is busy, and samples the status of the external equipment, the control word and the control word address. The three 1-bit designators determine whether one, two, or all three quantities are to be examined *(See Table 5-1).*

Table 5-1. Copy Status Designators

Designator	If a "0"	If a "1"
$_{s}0$	No operation	Places channel and equipment status bits in B^{b}; if b is 0 or 7 these status bits are lost.
$_{s}1$	No operation	Places control word address in the lower order 18 bits of the Q register.
$_{s}2$	No operation	Places control word in the A register in the following bit positions: current address 00-17 control jump 44 word count 24-38 operation code 45-47

NOTES

1. The Busy bit of the Status Response (bit 01) is the logical OR combination of the Read In Progress (bit 12) and the Write In Progress (bit 13) from the data channel, and the Busy Condition of from the connected equipment. Bit 01 becomes "1" as soon as the Read or Write instruction is executed, and stays up until the external equipment has finished all activity and is ready to perform another operation. Bits 12 and 13 come up as soon as the respective instruction is executed and drop when the data channel becomes inactive. The external equipment does not become busy until the data channel has fetched the control word and initiated the operation, but may remain busy for some time after the data channel becomes inactive. This is shown in the following diagram.



2. The Copy Status instruction will hang up if used to examine the control word during a series of control word jumps. If the 3604 executes a Copy Status to examine the Control Word while the data channel is in the process of fetching a new control word, the 3604 will not be allowed to complete the instruction until the new control word has been entered into the data channel registers. During a series of jump operations, the contents of the Control Word registers in the data channel are continuously changing. A Copy Status instruction executed at this time will be automatically repeated as long as the jump operations continue; therefore, if the jump operations are proceeding in a ring, the Copy Status instruction will remain hung up indefinitely.

3. In systems containing multi-channel peripheral equipment, a Connect code from a data channel may be rejected by an equipment because it is busy, reserved, or otherwise occupied by another data channel. The equipment will, however, enable its status lines to the channel which attempted to connect, so that the reason for the reject may be determined using the Copy Status instruction. The status lines remain enabled to that channel until it attempts to connect another of its equipments.

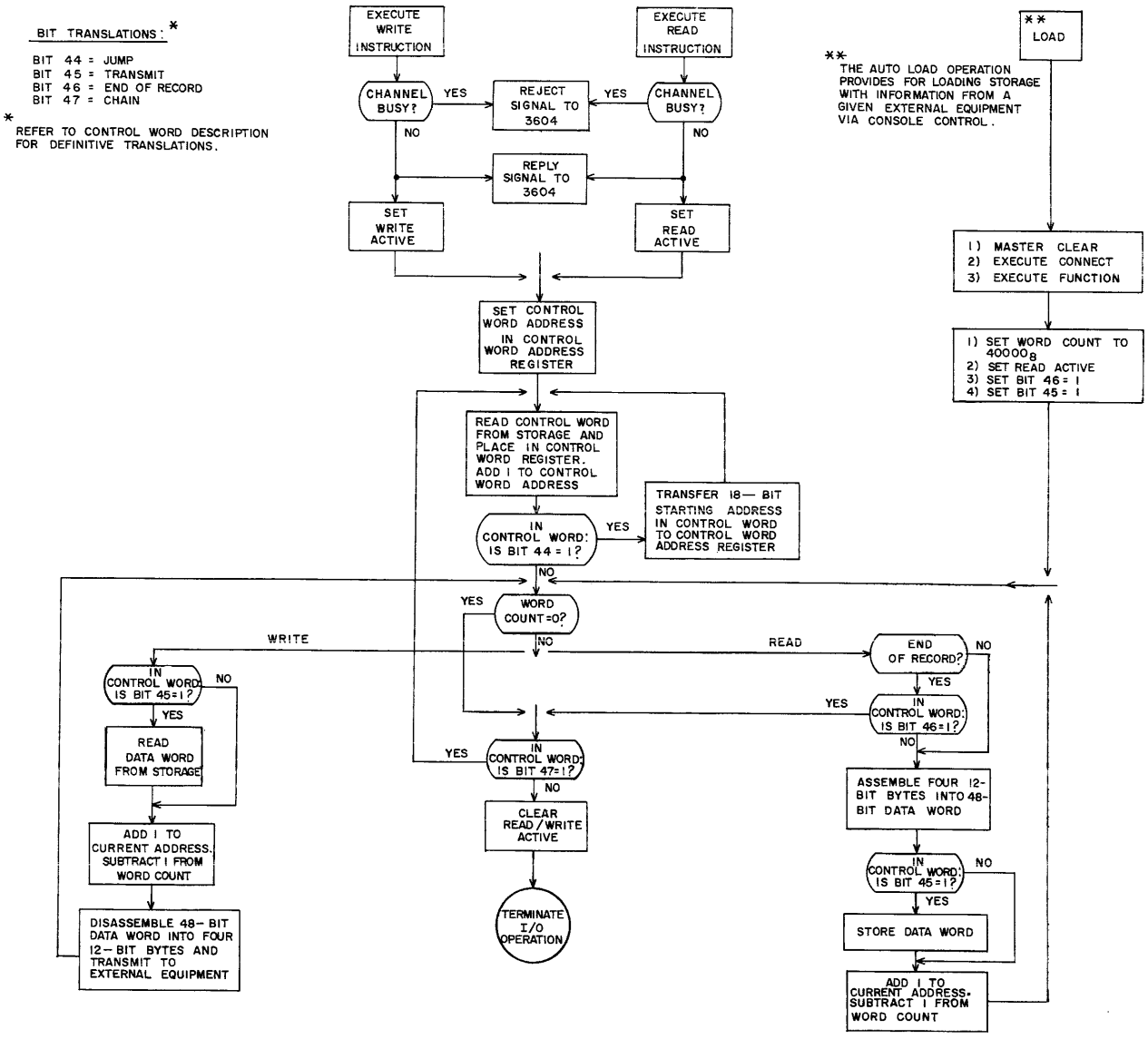
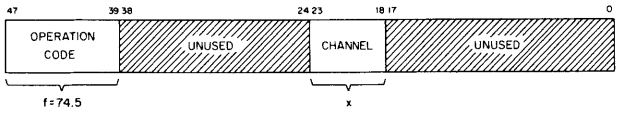


Figure 5-4. Read and Write Operations

Clear Channel (CLCH) (74.5)



The Clear Channel instruction:

1) Disconnects all equipments from the specified channel, preventing any communication until a Connect instruction is executed.

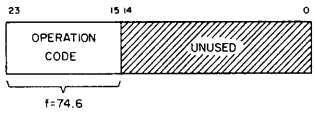
2) Disconnects all units within an equipment.

3) Clears the channel control word to its original state; i.e., a control word containing all zeros. This cuts off any data transmission in progress.

4) If channel 40_8 is specified (a non-existent channel), all channels are cleared.

This instruction assures a cleared channel prior to use when the previous condition of the channel is unknown, or removes a hang-up condition caused by a malfunction.

Input To A (IPA) (74.6)

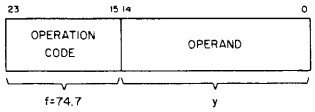


The Input to A instruction enters a single byte into the cleared A register from an external equipment. Note that the A register is cleared before the byte is placed in A. If successive Input to A operations are programmed, the contents of A must be stored after each operation or the succeeding Input to A instruction will clear A and information will be lost. A switch on the maintenance section of the console selects the equipment to be used. The following equipments may be used with the Input to A instruction:

1) Keyboard: 6-bit byte entered into positions 0 through 5 of the A register.

2) Card Reader: 12-bit byte entered into positions 0 through 11 of the A register.

Perform Algorithm (ALG) (74.7)



Certain special functions may be accomplished in the 3600 computing system with the addition of an algorithm module. The 3604 computation module is equipped to:

1) Receive command signals from the algorithm module.

2) Send certain signals to the algorithm module.

3) Send (continuously) the upper address portion of the U^{000} register to the algorithm module.

4) Receive information from the algorithm module (information received by the 3604 is placed in index register B^6).

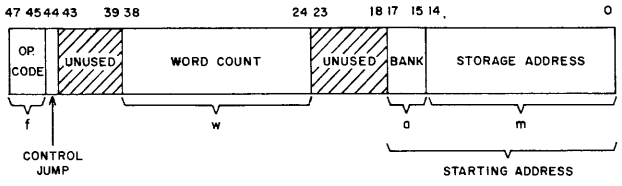
The Perform Algorithm instruction transmits a Begin Algorithm signal to an algorithm module. The 15-bit 'y' portion of the instruction activates a combination of fifteen lines connecting the 3604 to the algorithm module. (Refer to the appropriate Algorithm Manual for 'y' values.) The 3604 ceases all activity (other than that noted above) and gives total control of the system to the algorithm module.

If this instruction is executed and no algorithm module is attached to the 3604, the following occurs:

1) If the Illegal Instruction Fault bit in the Interrupt Mask register is set, and interrupt is active, interrupt occurs.

2) If the Illegal Instruction Fault bit in the Interrupt Mask register is not set, this instruction is a pass (do nothing) instruction.

CONTROL WORD



All data transmission between the 3600 system and external units is governed by control words associated with each data channel. The control word specifies the number of data words to be transmitted into or out of the system, the starting address in storage for the list of words, and eight possible operating modes for the input or output activity.

The control word, prior to the input or output operation, is located in one of the storage modules. Its location is designated by an 18-bit control word address. A Read or Write instruction transmits the 18-bit control word address to the data channel. The control word specifies an 18-bit starting address 'a m' from which the first output word will be read, or in which the first input word will be stored. A 15-bit word count 'w' specifies the number of data words to be transmitted. Any Read or Write instruction specifying a control word in which the word count is zero will be treated as a pass (do nothing) instruction, except as noted under chaining. The control word is fetched from storage and the starting address, word count operation code, and the incremented control word address are placed in their respective registers within the designated data channel.

When the current operation is complete (word count is zero), the data channel:

1) Reads another control word from storage (if chaining is selected), or

2) Halts the input or output activity.

The word count must be a value such that the sum of the word count and the lower order 15 bits of the starting address do not exceed 2^{15}. A sum greater than 2^{15} exceeds the capacity of the designated storage module. The last address which can be referenced in an input or output operation is address 77777. If the word count is not reduced to zero when address 77777 is referenced, the next word written into or read from storage will reference address 00000 (within the same storage module). Subsequent data words are transmitted to or from consecutive storage locations in the normal manner.

If a return to address 00000 is not desired, the word count must be such that it is reduced to zero when address 77777 is reached. If additional input or output data is to be processed, chaining may be effected to fetch a new control word.

Before examining the 3-bit operation code 'f' in the control word, the 1-bit designator (bit 44) is examined. This bit provides for a control jump. If bit 44 is a "1", the following operations occur:

1) The 3-bit 'f' portion is not interpreted.

2) The address portion of the control word is placed in the Control Word Address register.

3) A new control word is read from that address.

The 3-bit operation code 'f' is specified by bits 47, 46, and 45 of the control word. Their individual meanings are as follows:

Bit 47 = "1", Chain
(When word count = zero, a new control word is fetched from storage and the operation continues.)

Bit 47 = "0", Chaining not selected

Bit 46 = "1", End of record
(Operation terminates at end of record, although word count may not be zero.)

Bit 46 = "0", End of record not selected

Bit 45 = "1", Transmit
(Send information to storage or to external equipment.)

Bit 45 = "0", Skip
(The specified number of words are read from the external equipment, but are not sent to storage. If information is being written from storage into the external equipment, the words skipped will be written as all "0's".)

The 3-bit operation code 'f' specifies eight possible modes for a Read operation, and eight possible modes for a Write operation. These conditions are:

For a Read Operation:

'f'	Operation
0	Skip 'w' words and terminate
1	Read 'w' words and terminate
2	Skip to end of record and terminate
3	Read 'w' words or to end of record and terminate
4	Skip 'w' words and chain to next control word
5	Read 'w' words and chain to next control word
6	Skip to end of record and chain to next control word
7	Read 'w' words or to end of record and chain to next control word

For a Write Operation:

'f'	Operation
0*	Write 'w' words of zeros, insert record gap and terminate
1	Write 'w' words, insert record gap and terminate

* Refer to Record Gap definition on page 5-10.

2 Write 'w' words of zeros, insert record gap and terminate

3 Write 'w' words, insert record gap and terminate

4 Write 'w' words of zeros and chain to next control word

5 Write 'w' words and chain to next control word

6 Write 'w' words of zeros, insert record gap and chain to next control word

7 Write 'w' words, insert record gap and chain to next control word

The following definitions apply to the terms used in the above description:

Skip — The number of words specified by the word count designator 'w' are read from the external equipment, but are not transmitted to storage.

Read (Transmit) — The number of words specified by 'w' are transmitted to storage; the first word of the list is stored in the location specified by the 18-bit starting address (lower order 18 bits of the control word).

Write — The number of words specified by 'w' are transmitted from storage to the external equipment. The first word of the list is read from the storage location specified by the 18-bit starting address (lower order 18 bits of the control word).

Chain — If chaining is selected, the operation specified by the operation code in the control word is executed. The number of words 'w' (or to the end of record, as specified) are read, skipped, or written, depending on whether the operation is read or write. A new control word is then fetched from storage, and operation continues without a halt. If chaining operations are to be performed, the appropriate control words must previously have been stored in consecutive storage locations within the same storage module.

Record Gap — In writing data on magnetic tape with chaining not selected, the number of words 'w' constitutes a record. If chaining is selected and 'f' = 4 or 5, everything written under the control of the chain constitutes a record. If chaining is selected and 'f' = 6 or 7, the number of words 'w' in each control word constitutes a record; the chain may contain several records. When all data words in a record have been transmitted, the Write signal drops, causing the tape unit to write a record gap.

End of Record — When end of record is selected and the word count is reduced to zero, before end of record is reached, the Terminate or Chain operation (whichever is selected) occurs. If end of record is selected and this condition is reached before the word count reaches zero, the Terminate or Chain operation occurs.

Terminate — The operation terminates when the word count is reduced to zero at end of chaining, or end of record is reached before the word count reaches zero (Read operation with End of Record selected). This inactivates the channel and removes the console channel active indication.

INPUT/OUTPUT CHAINING

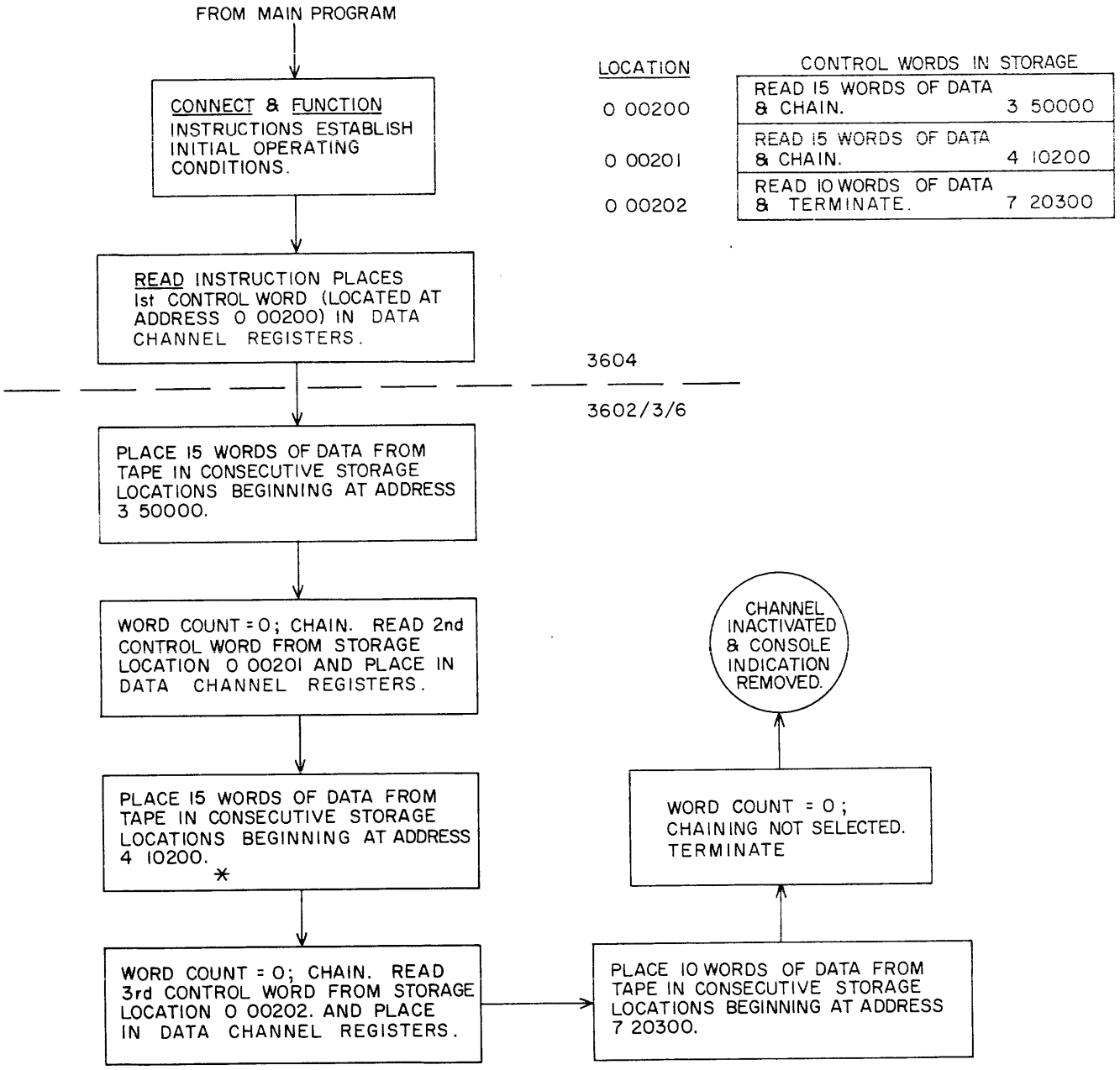
Data may be transmitted to or from several block locations in storage by a process called chaining. If chaining is selected (by the appropriate operation code in the control word), the programmer must previously have stored the correct control words in consecutive storage locations within the same storage bank. After performing the input or output operation on the first block of data (the number of data words in the block is determined by the word count in the control word), the second control word is read from storage and the operation continues without interruption. A sample chaining operation is diagrammed in figure 5-5.

NOTE

All blocks of data involved in the chaining operation must contain at least one word, except the final block which may be zero words in length. Thus, a control word in which bit 47 = "1" specifying the chain operation must contain a word count of 00001 or greater. Any control word specifying the chain operation in which the word count is zero constitutes an illegal condition and will result in a malfunction.

Problem:

Read one record of data from magnetic tape. Break this record into three portions and store in three non-consecutive block locations in storage. The record is 40 data words in length. The first portion is 15 words, the second portion is 15 words, and the third portion is 10 words.



* NOTE: THOUGH THE CONTROL WORDS MUST BE LOCATED IN CONSECUTIVE LOCATIONS WITHIN THE SAME STORAGE BANK, BLOCKS OF DATA IN THE CHAINING PROCESS MAY BE IN DIFFERENT STORAGE BANKS.

Figure 5-5. Chaining Operation

AUTO-LOAD

The auto-load feature provides a means by which instructions or data contained on an external storage medium may be automatically loaded into storage in the 3600 system. The operator can specify the channel and equipment to be connected, and the function to be executed by means of switches on the 3604 chassis.

Pressing the Auto-Load switch on the console initiates the following sequence of operations:

1) Performs internal and external master clear.

2) Connects the channel, equipment and unit as specified by the Auto-Load switches on the 3604 chassis.

3) Performs the function specified on the Function switches.

4) Begins a Read operation; the starting address is always 0 00000, the word count is $40{,}000_8$. The Read operation continues until $40{,}000_8$ words are read or until End of Record.

5) After completing the Read operation, execution of the program automatically begins at address 0 00000.

Any external storage medium may be auto-loaded, including magnetic tape, disc file, and punched cards. The magnetic tape produces an End of Record signal after each record. The disc file produces an End of Record signal at the end of a segment (a segment is thrity-two 48-bit words). In the case of punched cards, each individual card constitutes a logical record. If several cards are to be auto-loaded, the first card must contain a "loader". This is a programmed Read instruction covering the remaining cards.

SEQUENCE OF STEPS IN READ AND WRITE OPERATIONS

The order in which activity progresses during I/O Read and Write operations is presented in the following outline:

I. The program must execute a Connect instruction. The equipment remains connected until cleared or until another Connect instruction is executed for the same channel.

   A. This instruction directs the 3604 to send a 6-bit code to select the channel, a 12-bit Connect code to select the external equipment, and a Connect signal.

      1. The 6-bit Channel code contains one extra bit since only 5 bits are necessary to select one of the 32 possible channels. The lower 3 bits of the 5-bit code select one of eight data channels and the upper 2 bits select one of four communication modules, thus specifying only one of 32 possible channels.

      2. The 12-bit Connect code is composed of a 3-bit Equipment code and a 9-bit Unit code. The 3-bit Equipment code selects one of the eight equipments which may be attached to the selected channel. The 9-bit Unit code provides for selecting one of the units controlled by a piece of external equipment. As an example, if the selected equipment is a magnetic tape controller, the 9-bit code must select one of up to 16 tape units which may be controlled by it.

      3. The Connect signal directs the external equipment to examine the 12-bit Connect code and to respond if selected. The Connect signal stays up until a response is returned. The response may be a Reply or a Reject, depending on whether or not the connection can be made.

   B. The signals which may be produced in response to the Connect signal are listed below. In all cases, the response drops as soon as the Connect signal drops.

      1. The Reject signal means that the connection cannot be made. It may be produced because of any or all of the following reasons:

         a. Channel Busy: The selected channel is currently performing a Read or Write operation. The channel indicates this condition by a Busy signal which is sent to the communication module. If the Busy signal is present, the communication module returns a Reject to the 3604 immediately upon receipt of a Connect signal.

         b. False Reference: The referenced equipment or unit is not attached to that channel. If no response is received within 100 μsec, the Reject is generated automatically by the 3604.

         c. Unit Unavailable: The unit is currently being used by another source. This may occur only if an equipment is multi-channel, such as a magnetic tape controller.

2. The Reply signal is returned to the 3604 when the instruction has been performed, meaning, in this case, that the selected external equipment has been connected. It is generated by the external equipment and indicates that the 3604 is free to resume its main program.

II. The program may execute a Function instruction if the designated equipment is capable of performing more than one operation. For instance, if the designated equipment were a printer, the Function instruction might specify the Double Space mode.

A. This instruction directs the 3604 to send a 6-bit code to select the channel, a 12-bit code to specify the function, and a Function signal.

1. The 6-bit code selects the desired channel; the external equipment has already been connected to that channel by the Connect instruction.

2. The 12-bit Function code specifies the operating conditions of the external equipment. The various function codes for each equipment are listed in the associated reference manuals.

3. The Function signal directs the external equipment to examine the 12-bit Function Code. The Function signal stays up until a response is received. The response may be either a Reply or a Reject, depending on whether or not the specified operation can be performed.

B. The signals which may be produced in response to a Function signal are listed below. In all cases, the response drops as soon as the Function signal drops.

1. The Reject signal means that the specified function cannot be performed. It may be produced because of any or all of the following reasons:

a. Illegal Code: The Function code cannot be interpreted by the specified device.

b. Equipment or Unit Busy or Not Ready: The device cannot perform the specified operation without damaging the equipment or losing data.

c. Channel Busy: The selected channel is currently performing a Read or Write operation. The channel indicated this by sending a Busy signal to the 3602 communication module. If the Busy signal is present, the 3602 returns a Reject to the 3604 immediately upon receipt of a Function signal.

d. No Equipment or Unit Connected: The referenced device is not connected to the system and cannot recognize a Function instruction. The Reject for this condition is generated by the 3604 after a 100 μsec delay.

2. The Reply signal is returned to the 3604 when the instruction has been performed, meaning, in this case, that the external equipment has accepted the Function code. It is generated by the external equipment and indicates that the 3604 is free to resume its main program.

III. The program may execute a Read instruction, initiating input activity on the selected channel and external equipment. During a Read operation, the 3600 system obtains information from the selected external equipment. Examples of external equipment of this type are magnetic tape units or punched card readers. A Read operation is initiated by the 3604 in response to a programmed instruction, but the remainder of the operation is performed by the 3602 and the data channel.

The operation is governed by a control word, which is a 48-bit word that has been previously stored in memory. The control word specifies the starting address in memory, and the word count, which is the number of words involved in the operation.

A. This instruction directs the 3604 to send a 6-bit Channel Selection code, an 18-bit control word address consisting of a 3-bit bank address and a 15-bit storage address, and a Read signal.

1. The 6-bit code selects the data channel to which the external equipment is connected.

2. The 18-bit control word address specifies one of eight possible storage modules by means of the 3-bit bank address, and the location within that storage module by means of the 15-bit storage address.

3. The Read signal directs the data channel to examine the 18-bit control word address and to start the Read operation. The Read signal stays up until a response is received by the 3604. The response may be a Reply or a Reject, depending on whether or not the operation can be performed.

B. The signals which may be produced in response to a Read signal are listed below. In all cases, the response drops as soon as the Read signal drops.

1. The Reject signal means that the Read operation cannot be performed because:

a. Channel Busy: The selected channel is currently performing a Read or Write operation. This condition is indicated by a Busy signal which the channel sends to the 3602 communication module. If the Busy signal is present, the 3602 returns a Reject to the 3604 immediately upon receipt of the Read signal.

2. The Reply signal indicates that the instruction has been performed and the 3604 is free to resume its main program. If the Busy signal is not present when the Read signal is received, the 3602 immediately returns a Reply to the 3604.

C. The 3604, having initiated and established the Read operation, continues its main program. The Read operation is performed by the 3602 communication module and the data channel, and is completely independent of the main program in the 3604. The Read operation proceeds as follows:

1. The control word is fetched from storage. The control word address is transmitted to the 3602 S register, is incremented by 1, and returned to the Control Word Address register in the data channel. The starting address and word count are placed in their respective registers. Bits 47, 46, and 45 specify the operation code, and bit 44 determines whether a control word jump will occur.

2. The starting address in storage is specified by the lower 18 bits of the control word.

3. The word count is specified by bits 24 through 38 of the control word. The 15-bit word count limits the operation to a maximum of 32,768 words, which is the capacity of a single storage module. Operations involving more words than this must use the technique of chaining to other modules.

4. The Read operation continues with the data channel sending a Read signal and a Data signal to the external equipment. The Read signal stays up during the entire operation. The Data signal indicates that the data channel is ready to receive a 12-bit word from the external equipment. When the Reply signal is received from the external equipment, the Data signal drops.

5. The Reply signal is returned by the external equipment to the data channel. In a Read operation, the Reply signal indicates that the external equipment has a 12-bit word of data ready for transmission, and directs the data channel to sample it. Upon receipt of the Reply signal, the data channel drops the Data signal, causing the external equipment to drop the Reply signal.

a. The data channel transmits the Word Mark signal to the external equipment to indicate the completion of each 48-bit word. The Word Mark signal comes up simultaneously with the Data signal for the final 12-bit byte and drops when the Data signal drops. It is not repeated until the data channel finishes assembling another 48-bit word.

6. The data channel sends another Data signal to the external equipment approximately 0.1 μsec after the Reply signal drops. The operation continues until four 12-bit bytes have been assembled into a 48-bit word.

7. When the 48-bit word is assembled, the data channel sends the storage address and a Request to the 3602.

8. The 3602 obtains access to storage and returns an Enabling signal to the data channel. The word of information is then stored in memory.

9. While the word is being stored, the 3602 increments the storage address by one and returns it to the Storage Address register in the data channel. The channel then sends the complement of the word count to the 3602 where it is incremented

by one and returned. By incrementing its complement, the actual word count is effectively reduced by one.

10. If the operation code does not specify chain or end of record, the preceding steps are repeated until the word count in the data channel reaches zero. The data channel then inactivates the external equipment by dropping the Read signal.

IV. The program may execute a Write instruction, initiating output activity on the selected channel and external equipment. During a Write operation, the 3600 system sends information to the selected external equipment. An example of an external equipment of this type would be a line printer.

A Write operation is initiated by the 3604 in response to a programmed instruction, but the remainder of the operation is performed by the 3602 and the data channel. The operation is governed by a control word, which is a 48-bit word that has been previously stored in memory. The control word specifies the starting address in memory, and the word count which is the number of words involved in the operation.

A. This instruction directs the 3604 to send a 6-bit Channel Selection code, an 18-bit control word address consisting of a 3-bit bank address and a 15-bit storage address, and a Write signal.

1. The 6-bit code selects the data channel to which the external equipment is connected.
2. The 18-bit control word address specifies one of eight possible storage modules by means of the 3-bit bank address, and the location within that storage module by means of the 15-bit storage address.
3. The Write signal directs the data channel to examine the 18-bit control word address and to start the Write operation. The Write signal stays up until the 3604 receives a response. The response may be a Reply or a Reject, depending on whether or not the operation can be performed.

B. The signals which may be produced in response to a Write signal are listed below. In all cases, the response drops as soon as the Write signal drops.

1. The Reject signal means that the Write operation cannot be performed because:
   a. Channel Busy: The selected channel is currently performing a Read or Write operation. This condition is indicated by a Busy signal which the channel sends to the 3602 communication module. If the Busy signal is present, the 3602 returns a Reject to the 3604 immediately upon receipt of the Write signal.
2. The Reply signal indicates that the instruction has been performed and the 3602 immediately returns a Reply to the 3604.

C. The 3604, having initiated and established the Write operation, continues with its main program. The Write operation is performed by the 3602 communication module and the data channel, and is completely independent of the main program in the 3604. The Write operation proceeds as follows:

1. The control word is fetched from storage. The control word address is transmitted to the 3602 S register, is incremented by 1, and returned to the Control Word Address register in the data channel. The starting address and word count are placed in their respective registers. Bits 47, 46, and 45 specify the operation code and bit 44 determines whether a control jump will occur.
2. The starting address in storage is specified by the lower 18 bits of the control word.
3. The word count is specified by bits 24 through 38 of the control word. The 15-bit word count limits the operation to a maximum of 32,768 words, which is the capacity of a single storage module. Operations involving more words than this must use the technique of chaining to other modules.
4. The Write operation continues with the data channel sending a Write signal to the external equipment and a Request, together with the storage address, to the 3602. The Write signal to the external equipment stays up during the entire operation. The Request signal drops when the channel has been honored. The 3602 then obtains access to storage and the 48-bit word is fetched and entered into the Assembly/Disassembly register in the data channel.

5. While the word is being fetched, the 3602 increments the storage address by one and returns it to the data channel. The channel then sends the word count to the 3602 where it is incremented by one and returned. By incrementing its complement, the actual word count is effectively reduced by one.
6. The channel sends the Data signal to the external equipment. This indicates that a 12-bit byte of the 48-bit word is available and directs the external equipment to sample it. The Data signal drops as soon as the Reply signal is received from the external equipment.
    a. The data channel transmits the Word Mark signal to the external equipment to indicate the completion of each 48-bit word. The Word Mark signal comes up simultaneously with the Data signal for the final 12-bit byte and drops when the Data signal drops. It is not repeated until the data channel finishes disassembling another 48-bit word.
7. The Reply signal is returned by the external equipment to the data channel. In a Write operation, the Reply signal indicates that the external equipment has accepted the 12-bit word of data. Upon receipt of the Reply signal, the data channel drops the Data signal, causing the external equipment to drop the Reply signal.
8. The preceding operation continues until the four 12-bit bytes of the 48-bit data word have been transmitted to the external equipment. The data channel then sends another Request to the 3602 and another 48-bit data word is fetched from storage.
9. If the operation code does not specify Chain, the operation ends when the word count in the data channel reaches zero. The data channel then inactivates the external equipment by dropping the Write signal.

TRANSMISSION RATE

The rate of transmitting each 48-bit word of I/O information is within the following boundaries:

Best Case	1.5 μsec per word
Worst Case	60 μsec per word

The best case refers to a situation in which only one data channel is active and demanding access to storage. It must be communicating with an external equipment of sufficiently high speed so that the rate of information transfer is limited by the memory cycle time.

The worst case is a situation in which one storage module is providing storage capacity for the 3604 and a total of 32 data channels, all of which are simultaneously active and demanding access to storage. The 3604 has exclusive use of one of the five access channels to storage; thus, 40 memory cycles are required to honor the 32 data channels and a total of eight requests from the 3604.

All I/O information is transmitted on a word-by-word basis. As soon as each word is delivered, the storage module disconnects itself and will not communicate further with the selected 3602 until it has honored its four remaining access channels. In addition, the 3602 will have no more communication with the selected data channel until it has honored the seven other channels which may be attached to it.

The storage module and the 3602 each contain a free-running scanner which sequentially monitors their communication channels. If no other channels require servicing, the scanners will return to their original position after approximately 0.3 μsec.

STORAGE REFERENCE FAULT

Each 3602, like the 3604, has exclusive use of one of the five access channels to storage. When one of its associated data channels is ready to fetch or store a word, the 3602 sends a Request signal to storage. Requests to storage are honored sequentially and, in the worst case, the 3602 will be required to wait approximately 6 μsec. If the request is not honored within 10 μsec, a Storage Reference Fault has occurred. The 3602 then releases the data channel and will accept requests from other data channels. The data channel which was unable to reference storage does not continue the operation. It is left with its storage address incremented by 1, but the word count remains the same as before the attempted reference.

The Storage Reference Fault condition is shown by a lighted indicator on the 3602 and can be removed only by a master clear. This condition will cause the data channel to interrupt if it previously has been selected by the Function code.

PARITY

All transmissions of data between storage and external equipment are accompanied by at least one parity bit. In all cases, odd parity is used.

Parity Error

Bit 14 of the Channel and Equipment Status Response is set to "1" if a parity error occurs. Interrupt will occur on parity error if it has been selected by the Function code.

I/O Transmission Parity Error

Each 12 bits of data transmitted between the data channel and external equipment are accompanied by a parity bit. During a Read operation, the parity bit is generated by the external equipment and checked by the data channel. During a Write operation, the parity bit is generated by the data channel and checked by the external equipment, which returns a Parity Error signal to the data channel if an error has occurred.

Storage Parity Error

The 18-bit storage address transmitted from the 3602 communication module to the storage module is accompanied by a parity bit generated by the 3602. The address and parity bit are checked by the storage module, which returns a Storage Address Parity Error signal to the 3602 if an error has occurred. The 3602, in turn, forwards the signal to the data channel.

Each 48-bit word of data transmitted between the data channel and storage passes through the 3602. During a Read operation, the 3602 generates three parity bits which are stored together with the 48 bits of data. During a Write operation, the 3602 examines the 51 parity and data bits to determine if the previously generated parity condition is still present. If an error has occurred, the 3602 sends a Parity Error on Word from Storage Signal to the data channel.

The 48-bit word of data is divided into three portions with a parity bit for each. These are:

Lower Address:	bits 00 through 14
Upper Address:	bits 24 through 38
Function:	bits 15 through 23, and 39 through 47

Control Word Parity Error

The Read or Write operation halts if a parity error occurs during the fetching of a control word. The data channel will not proceed further until the Control Word Parity Error condition is cleared. This may be done using the Clear Channel instruction, the master clear facility, or by sending another Function code to the data channel (with bit 23 = "1").

Parity Error On Connect Or Function Code

The external equipment does not connect and does not return a Parity Error signal if a parity error exists on the Connect code. If a parity error exists on the Function code, the connected equipment returns a Parity Error signal but does not perform the specified function.

CHAPTER VI
PARITY

In the 3600 system, parity bits are generated and checked to determine one of three possible conditions:

1) Errors in data channel transmissions

2) Errors in storage address transmissions

3) Errors in storage transmission of data or instruction words

The presence of a parity error may indicate one or more of the above has occurred. Odd parity is used; that is, the parity bit is set such that the total number of ones in the address or storage word is odd.

In addition to the three cases listed above, the external equipment itself may generate and check parity on its internal operations. Treatment of this case is discussed later in this chapter. For operational details, refer to the chapter for the specific equipment.

PARITY GENERATION

Parity bits are generated in the following cases:

1) On each storage address transmitted to storage from either the 3604 or the 3602.

2) On an instruction transmitted to storage from either the 3604 or the 3602.

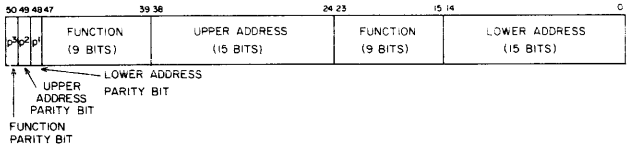
3) On a data word transmitted to storage from either the 3604 or the 3602.

4) On each 12-bit byte of data transmitted to an external equipment from the 3606 data channel.

5) On each 12-bit byte of data transmitted to a 3606 data channel from an external equipment.

Data Parity Generation

Each module (3604 or 3602) attached to a storage access channel provides three parity bits along with the 48-bit instruction or data word on a Write operation. The format of this word with its associated parity bits follows.



A parity bit is generated by the 3602 or 3604 for each of the three portions of the instruction or data word.

Storage Address Parity Generation

Each 3604 or 3602 module attached to a storage access channel must provide an address along with the word to be stored or from which a word will be read. A parity bit is generated by the 3604 or 3602 and accompanies the address to storage.

Parity Generation For Data Channel Transmissions

Each 12-bit byte of data being transmitted into a 3606 data channel from an external equipment has an accompanying parity bit. This parity bit is generated by the external equipment. The 3606 data channel generates a parity bit to accompany a 12-bit byte of data being transmitted to an external equipment.

PARITY CHECKING

Parity Checking On Storage Address Transmissions

Each storage address transmitted to the 3609 storage module from the 3602 or 3604 for a read or write reference is checked (in the 3609) for a parity error. If a transmission error has occurred, the 3609 returns a Parity Error signal to the appropriate module.

Parity Checking On Storage Or Transmission Of Data

When an instruction or data word is read from storage by the 3602 or 3604, the module initiating the storage reference checks for a parity error. The existence of a parity error may indicate one or more of the following:

1) A transmission error may have occurred either when the word was initially transmitted to storage or when the word was transmitted from storage.

2) The information was garbled in the storage read/write process itself.

Parity Checking On Data Channel Transmissions

Each 12-bit byte of data being transmitted from an external equipment into the 3606 data channel is checked for a parity error by the data channel. A parity error indicates a transmission error has occurred. A 12-bit byte of data being transmitted to an external equipment is checked for a parity error by the external equipment.

PARITY ERRORS

When parity errors occur, the manner in which they are handled depends on the module (3602, 3606 or 3604) to which the Parity Error signal is sent, or in which parity is checked. Descriptions of parity errors given below list the cases for each module.

Storage Address Parity Error

A storage address transmitted to the 3609 storage module from the 3604 is checked for parity in the 3609. If a transmission error has occurred, the 3609 returns a Parity Error signal to the 3604. This Parity Error signal lights an indicator on the console and stops the computer.

A storage address transmitted to the 3609 storage module from the 3602 is checked for parity in the 3609. If a transmission error has occurred, the 3609 returns a Parity Error signal to the 3606 via the 3602. This Parity Error signal sets an I/O Parity Error bit in the data channel. If interrupt is not selected to examine this bit, addressing operations will result in error. (Refer to following discussion on interrupt selection on I/O parity errors.)

Instruction Parity Error

An instruction word read from storage is checked for parity in the 3604. If an instruction parity error occurs, the instruction will not be executed and the computer will stop. An instruction parity error lights an indicator on the console.

Storage Or Data Transmission Parity Error

An operand read from storage by the 3604 is checked for parity in the 3604. If an operand parity error occurs, and if the Operand Parity Error bit in the Interrupt Mask register is set (assuming interrupt is active), interrupt occurs. If the Operand Parity Error bit is not set, interrupt does not occur and operations using this operand will result in error. An operand parity error lights an indicator on the console, regardless of whether the Operand Parity Error bit has selected interrupt.

A data word read from storage by the 3602 is checked for parity in the 3602. If a parity error occurs, the I/O Parity Error bit in the data channel is set. If interrupt is not selected to examine this bit, operations using this data will result in error. (Refer to following discussion on interrupt selection on I/O parity errors.)

Data Channel Transmission Parity Error

Data channel transmission parity errors may occur during several operations involving the data channel. Each of these cases is outlined below.

1) If a parity error occurs in transmitting the 12-bit Function code to an external equipment from the 3606, the following actions take place:

   a) The function specified by the code is not executed.

   b) No external Reject signal is generated by the external equipment; the Reject signal is generated internally by the 3604.

c) No Reply signal is sent to the 3604 from the external equipment.

d) A Parity Error Signal is sent to the 3606 data channel. This signal sets the I/O Parity Error bit in the data channel.

e) In order to initiate operation, either an external master clear (from the console) or a Clear Channel instruction must be executed to clear the condition.

2) If a parity error occurs in transmitting the 12-bit Connect code to an external equipment, the following actions occur:

a) The connection specified by the Connect code is not effected.

b) No external Reject signal is generated by the external equipment; the Reject signal is generated internally by the 3604.

c) No Reply signal is sent to the 3604 from the external equipment.

d) No Parity Error signal is generated.

e) In order to initiate operation, either an external master clear (from the console) or a Clear Channel instruction must be performed to clear the condition.

3) If a parity error occurs in transmitting a 12-bit byte of data from the data channel to an external equipment, the following actions occur:

a) A Parity Error signal is returned to the 3606 data channel. This Parity Error signal sets the I/O Parity Error bit in the data channel.

b) The Write operation continues as specified by the control word unless interrupt on I/O parity error is selected. (Refer to following discussion on interrupt section on I/O parity errors.)

4) If a parity error occurs in transmitting a 12-bit byte of data into the data channel from an external equipment, the following actions occur:

a) The I/O Parity Error bit in the data channel is set.

b) The Read operation continues as specified by the control word unless interrupt on I/O parity error is selected. (Refer to following discussion on interrupt selection on I/O parity errors.)

c) In order to initiate subsequent operation, either an external master clear (from the console) or a Clear Channel instruction must be performed to clear the condition.

Interrupt Selection On I/O Parity Error

Interrupt may be selected to recognize parity errors occurring in input/output operations. Selecting interrupt causes the I/O Parity Error bit in the 3606 data channel to be examined. The conditions which set this bit are:

1) address parity error

2) data parity error

3) data transmission parity errors:

a) parity error on Function code

b) parity error on Connect code

c) parity error on 12-bit data transmission to external equipment from channel

d) parity error on 12-bit data transmission from external equipment to channel

Selecting interrupt to recognize one of these occurrences is accomplished as follows:

1) Execute Function instruction (74.1 X0010 - Select Interrupt on Error).

2) Activate interrupt system via Internal Function instruction (77.0 XXX22 - Set Interrupt Active).

When interrupt is selected and one of the above conditions occurs, an interrupt routine is entered. Within the interrupt routine, scanning the Channel Product register isolates the cause of interrupt to an I/O parity error. An I/O parity error constitutes a major machine malfunction. In re-executing the I/O operations, if the error persists, call a maintenance engineer. (Refer to Interrupt chapter and appendix on Interruptible Conditions and Faults.)

External Equipment Parity Error

Parity errors within an external equipment (e.g., longitudinal parity error during a Read/Write operation on magnetic tape) are handled differently from the I/O parity errors. These parity errors indicate a difficulty in reading or writing information on the equipment's medium.

If a parity error occurs within the external equipment, a Parity Error signal is returned to the data channel via one of the twelve status lines.

Once the interrupt routine has isolated the cause of interrupt to a particular equipment (via the Channel Product Register Jump instruction), a Copy Status instruction must be executed to determine the particular condition causing interrupt (e.g., longitudinal parity error). Refer to the Interrupt chapter.

CHAPTER VII
CONSOLE

The basic 3600 computing system includes a 3601 console. The console, connected to the 3604 computation module and a 3606 data channel, contains switches for operating and maintaining the system, displays of register contents, and an electric typewriter for input to A and output operations.

The console is divided into two areas, one for operating the system and one for maintenance.

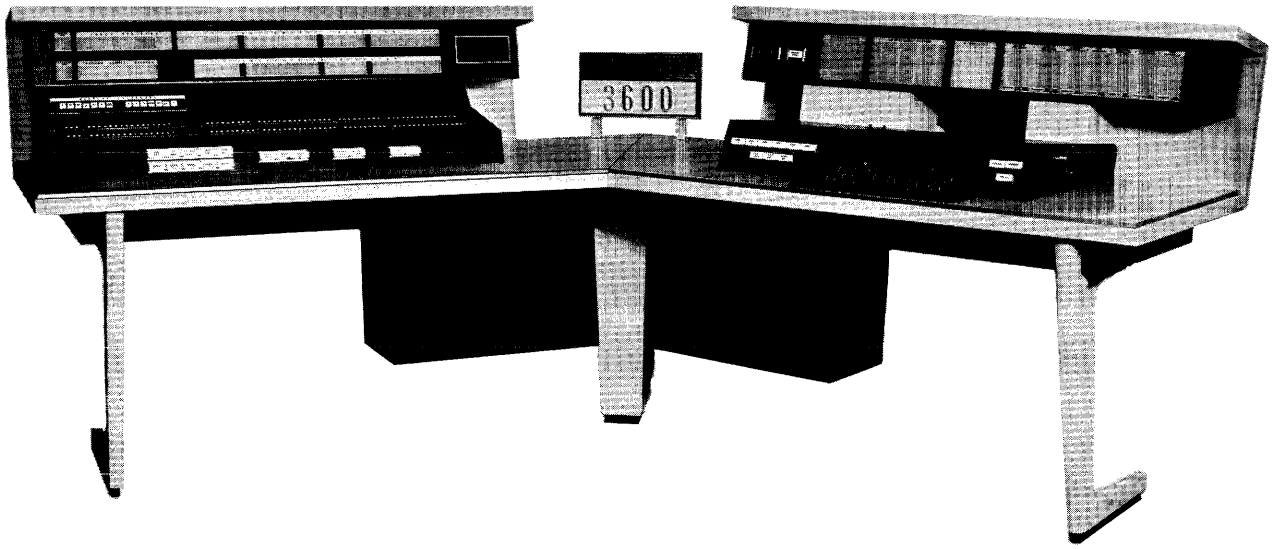


Figure 7-1. 3601 Console

MAINTENANCE SECTION

The maintenance section of the console, with its indicator panels and switches, is shown in figure 7-2.

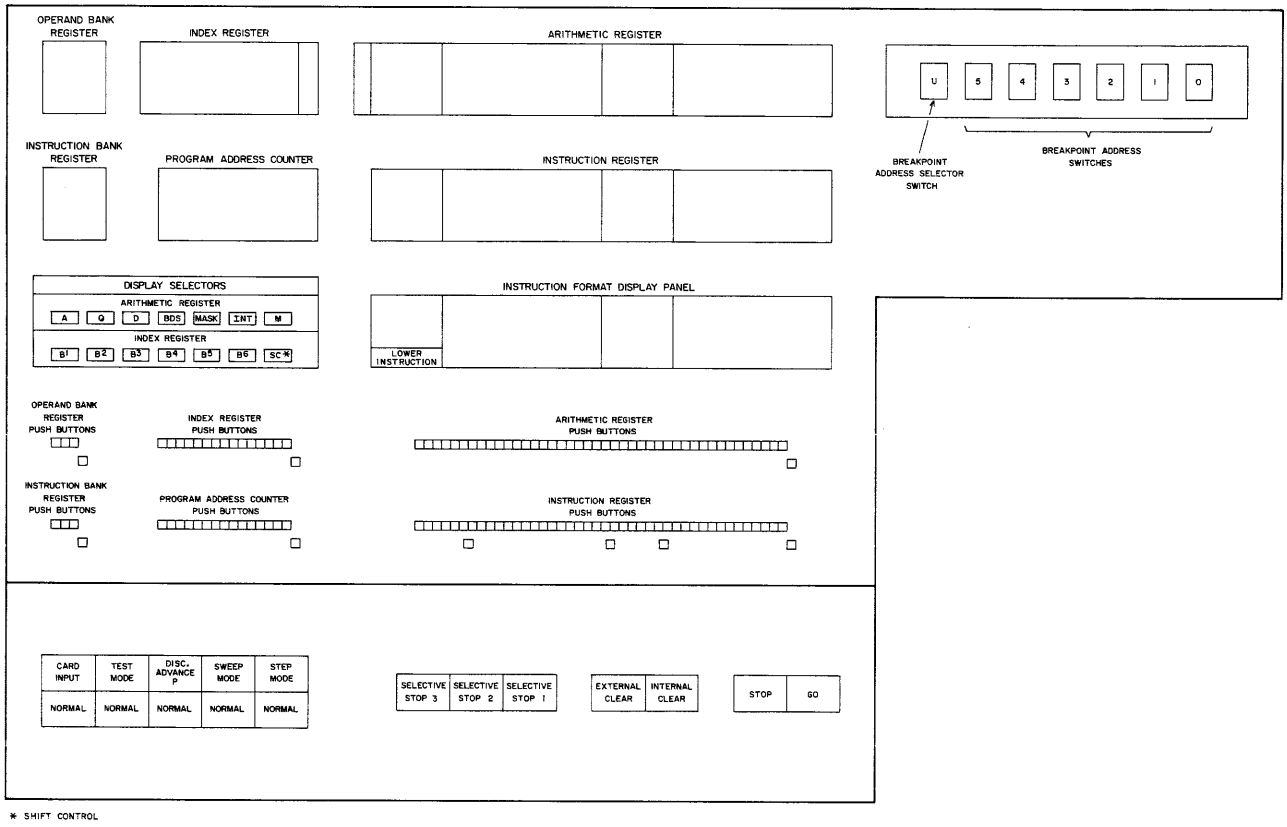


Figure 7-2. Maintenance Section of Console

Register Displays

Six indicator panels on the maintenance section of the console display the contents of all operational registers. The light indicators within the panels are lamp modules, each of which displays a single octal digit. The lamps, in response to binary signals from the computer, display the register contents in octal form*only when the computer is stopped; the display is blank when the computer is running. Exceptions to this are the Operand and Instruction Bank registers, the Program Address register, and the Instruction register. The contents of these registers are displayed at all times, whether the computer is running or stopped. (However, this display is binary rather than octal. The Set push buttons beneath the octal displays for these registers are illuminated.) Registers displayed are listed in table 7-1.

*Lamp modules within the Set push buttons associated with each of the register displays provide binary displays in addition to the octal displays of register contents when the computer is stopped. These binary displays are blank when the computer is running (note the exceptions).

Table 7-1. Register Displays

Register	No. of Octal Digits	Display Panel
Program Address Register	5	Program Address Register display panel.
Operand Bank Register	1	Operand Bank Register display panel.
Instruction Bank Register	1	Instruction Bank Register display panel.
Index Register (B^1 - B^6) Shift Count Register	5 3	The six index registers and the Shift Count register (Shift Control) share the single Index Register display panel. The register to be displayed (B^1 - B^6 or SCR) is selected by pressing the appropriate Selector switch. Located at the left end of the Index Register display panel is a light which indicates the register whose contents are being displayed (B^1, B^2, etc.).
Instruction Register	16	Instruction Register display panel.
M Register A Register Q Register D Register Interrupt Register Interrupt Mask Register Bounds Register	16 16 16 16 16 16 13	These registers share the single Arithmetic Register display panel. The register to be displayed is selected by pressing the appropriate Selector switch. Located at the left end of the Arithmetic Register display panel is a light which indicates the register whose contents are being displayed (M, A, etc.).

Instruction Format Display Panel

Located immediately beneath the Instruction Register display is an Instruction Format display panel. This panel lights words and/or phrases interpreting the designator values in the instruction held in the Instruction Register display panel. For example, if the quantity 10 appears in the upper two octal digits of the Instruction Register display panel, Operation Code is illuminated in the identical positions of the Format display panel. Thus, one can compare the two indicator panels and readily determine the octal value of a particular operation designator within the instruction being displayed.

Also included in this display is an indicator which is illuminated when the lower instruction of a pair of 24-bit instructions is being executed.

Switches and Push Buttons

Several switches appear on the maintenance section of the console. These switches and their operations are listed in table 7-2.

Table 7-2. Maintenance Switches

Switch	No.	M or L *	Function
Card Input	1	M	Pressing the Card Input switch selects the card reader for an Input to A operation. This operation loads the lower 12 bits of the A register with a 12-bit character from the card reader. This switch is illuminated when pressed.
Normal	1	M	Pressing the Normal switch (paired with the Card Input switch) selects the console typewriter for an Input to A operation and turns off the Card Input light. This operation loads the lower 6 bits of the A register with a 6-bit character for each typewriter key pressed.
Test Mode	1	M	Pressing the Test Mode switch places the computer in the Test mode. The following then occurs: 1) Instructions are executed in the normal manner. 2) After 300 μsec, the computer stops. 3) Approximately 50 μsec after stopping, an internal master clear is performed. 4) After 50 μsec, instructions are again executed, beginning with the instruction(s) at address 0 00000. Steps 2 through 4 are repeated as long as the computer is in Test mode.
Normal	1	M	Pressing the Normal switch removes the selection of Test mode and turns off its light. When the Go switch is pressed, computer operation proceeds in the normal manner.
Disconnect Advance P	1	M	Pressing the Disconnect Advance P switch disables advancing the count in the Program Address register. When the Go switch is pressed, the same instruction is repeated. The Disconnect Advance P switch is illuminated when pressed.
Normal	1	M	Pressing the Normal switch removes the selection made by the Disconnect Advance P switch and turns off its light. When the Go switch is pressed, the program steps will be executed in normal sequential order.
Sweep Mode	1	M	Pressing the Sweep Mode switch places the computer in Sweep mode. When the Go switch is pressed, instructions are read from consecutive storage locations in the normal manner, but are not executed. The Sweep Mode switch is illuminated when pressed.
Normal	1	M	Pressing the Normal switch removes the selection of Sweep mode and turns off its light. When the Go switch is pressed, instructions are read from storage and executed in the normal manner.

* M - momentary, L - locking

Table 7-2. (Cont'd)

Switch	No.	M or L	Function
Step Mode	1	M	Pressing the Step Mode switch places the computer in Step mode. A step may be defined as that portion of a program step executed between successive pressings of the Go switch (where a program step is an instruction pair or a single 48-bit instruction). Since steps can differ, the possible cases are outlined below: Case 1: Perform internal master clear; select Step mode; press Go switch. In this case, the program step (instruction word) is read from storage and the computer stops before executing the instruction. Pressing the Go switch again results in the following operations: a) If indirect addressing is specified by the instruction, the indirect address is determined and the computer stops. The next time the Go switch is pressed, the instruction is executed (if indirect addressing is not called for again) and the computer stops after: (1) reading (from the U register) the lower instruction of an instruction pair, or (2) reading a new instruction word from storage (if the previous instruction word word was a 48-bit instruction). b) If indirect addressing is not specified by the instruction, the instruction is executed and the computer stops after (1) or (2) above. Case 2: No internal master clear has been performed; select Step mode; press Go switch. In this case, the following operations occur: a) If indirect addressing is specified by the instruction, the indirect address is determined and the computer stops before executing that instruction. b) If no indirect addressing is specified, the instruction is executed and the computer stops after (1) or (2) above in case 1. The Step Mode switch is illuminated when pressed.
Normal	1	M	Pressing the Normal switch removes the selection of Step mode and turns off its light. When the Go switch is pressed, instructions are read from storage and executed in the normal high-speed manner.
Selective Stop 1* Selective Stop 2* Selective Stop 3*	1 1 1	L L L	Pressing the Selective Stop switches provides the manual conditions for stopping the computer on the Selective Stop instruction (76, b = 1, 2, 3, 5, 6 or 7). The Stop switches are illuminated when pressed.

* An asterisk denotes switch is active when running or stopped; others are active only when stopped.

Table 7-2. (Cont'd)

Switch	No.	M or L	Function
External Clear	1	M	Pressing the External Clear switch master clears all external equipment, the data channels to which they are attached, all registers and controls in the 3602 and the data channels, and registers and controls in storage.
Internal Clear	1	M	Pressing the Internal Clear switch master clears the 3604 computation module; i.e., all registers and most control FFs.
Go	1	M	Pressing the Go switch initiates execution of the instruction currently in the Program Control register if Master Clear was not previously pressed. If previous master clear was performed, reads instruction from address specified by current value of P and initiates execution of that instruction. Pressing the Go switch initiates execution of the program at the rate governed by the paired Step Mode/ Normal switches (whichever has been pressed).
Stop *	1	M	Pressing the Stop switch stops the computer. When the computer stops, selected operational registers are displayed on the console.
Display Selector Switches (M, A, Q, D, Interrupt, Mask, and Bounds Register Switches)	7	M	Pressing the particular register Display Selector switch enables that register to be displayed in the Arithmetic Register display panel when the computer is stopped. Pressing the Selector switch also permits quantities to be manually entered into the selected register or allows clearing that register via the Set and Clear push buttons.
Display Selector Switches (Index Registers B^1-B^6 and Shift Count Register)	7	M	Pressing the particular register Display Selector switch enables that register to be displayed in the Index Register display panel when the computer is stopped. This selection also permits quantities to be manually entered into the selected register or allows clearing that register via the Set and Clear push buttons. ***NOTE*** ***The contents of the Shift Count register may not be altered. Pressing the Set or Clear push buttons has no effect when this register is being displayed.***

* An asterisk denotes switch is active when running or stopped; others are active only when stopped.

Table 7-2. (Cont'd)

Switch	No.	M or L	Function
Set Push Buttons			
Instruction Register	48	M	The Set push buttons are located beneath the register indicator panels such that each register indicator panel has provision for manual entry. The Set push buttons are binary switches numbered in the powers of 2, beginning with zero. Each group of 3 is an octal digit. Pressing a Set push button forces that particular stage of a register to the set (or "1") state. The register to be manually loaded must have previously been selected by pressing one of the register Display Selector switches. The Set push buttons beneath the Instruction, Program Address, Instruction Bank and Operand Bank registers are illuminated while the computer is running.
Arithmetic Register	48	M	
Program Address Register	15	M	
Index Register	15	M	
Instruction Bank Register	3	M	
Operand Bank Register	3	M	
Clear Push Buttons			
Instruction Register	5	M	Pressing the Clear push button associated with the register display panel clears (to "0") every stage of the selected register. Five Clear push buttons are associated with various portions of the Instruction register. Pressing a particular push button results in clearing the following: 1) The entire register 2) Lower address portion 3) Lower operation code portion 4) Upper address portion 5) Upper operation code portion
Arithmetic Register	1	M	
Program Address Register	1	M	
Index Register	1	M	
Instruction Bank Register	1	M	
Operand Bank Register	1	M	
Breakpoint Address*	6	L	Six 8-position thumb-nail wheel switches can be set to octal addresses 000000 through 777777 (banks 0 through 7, addresses 00000 through 77777).
Breakpoint Address Selector Switch*	1	L	Breakpoint Address Selector switch is a four-position switch permitting selection of: a) Operand address b) Instruction address c) Operand or instruction address d) Off Computer stops when breakpoint address and address selected by Selector switch are equal.

* An asterisk denotes switch is active when running or stopped; others are active only when stopped.

OPERATOR SECTION

The operator section of the console, with its indicator panels and switches, appears in figure 7-3.

Displays

By means of lights, the operator section of the console displays a single register, a variety of fault conditions, and the activity of input/output operations. All displays on the operator section of the console are active when the condition arises, whether the computer is running or stopped. Conditions indicated by console lights are listed in table 7-3. For definitive descriptions of various fault conditions, refer to the appendix on Interruptible Conditions and Faults.

Table 7-3. Conditions Associated With Console Lights

Display	Condition When Illuminated
D Register	The 48-bit Flag (D) register is a binary display arranged in eight 6-bit portions, roughly forming a square. The contents of this register are displayed at all times. No push buttons are provided for entry into this register. (However, the D register may be manually manipulated on the maintenance section of the console.)
Input/Output Status	The Input/Output Status display provides an indication of data channel activity by illuminating the channel number and the operation (input or output) occurring on that channel (refer to figure 7-2).
Two's Complement	Computer is in Two's Complement mode of operation.
Address Parity Error	An error has occurred in transmitting an address from the 3604 to storage.
Interrupt Mode	Computer is in interrupt routine.
Interrupt Active	Interrupt system is enabled, permitting examination of selected interrupt conditions in an interrupt routine when these conditions occur.
Bounds Fault Light	A storage reference has been attempted outside the bounds specified by the contents of the Bounds register. Refer to Bounds register description, chapter III.
Exponent Underflow Light	An exponent underflow fault has occurred; in a Floating Point instruction, exponent is $< 2^{-10}-1$.
Exponent Overflow Light	An exponent overflow fault has occurred; in a Floating Point instruction, exponent is $> 2^{10}-1$.
Arithmetic Overflow Light	The result of an add or subtract operation exceeds the capacity of the A register.
Divide Fault Light	An improper Divide instruction has been executed.
Shift Fault Light	Shift count is greater than 60_8 or 140_8 (depending on whether 48 or 96-bit register was to be shifted).

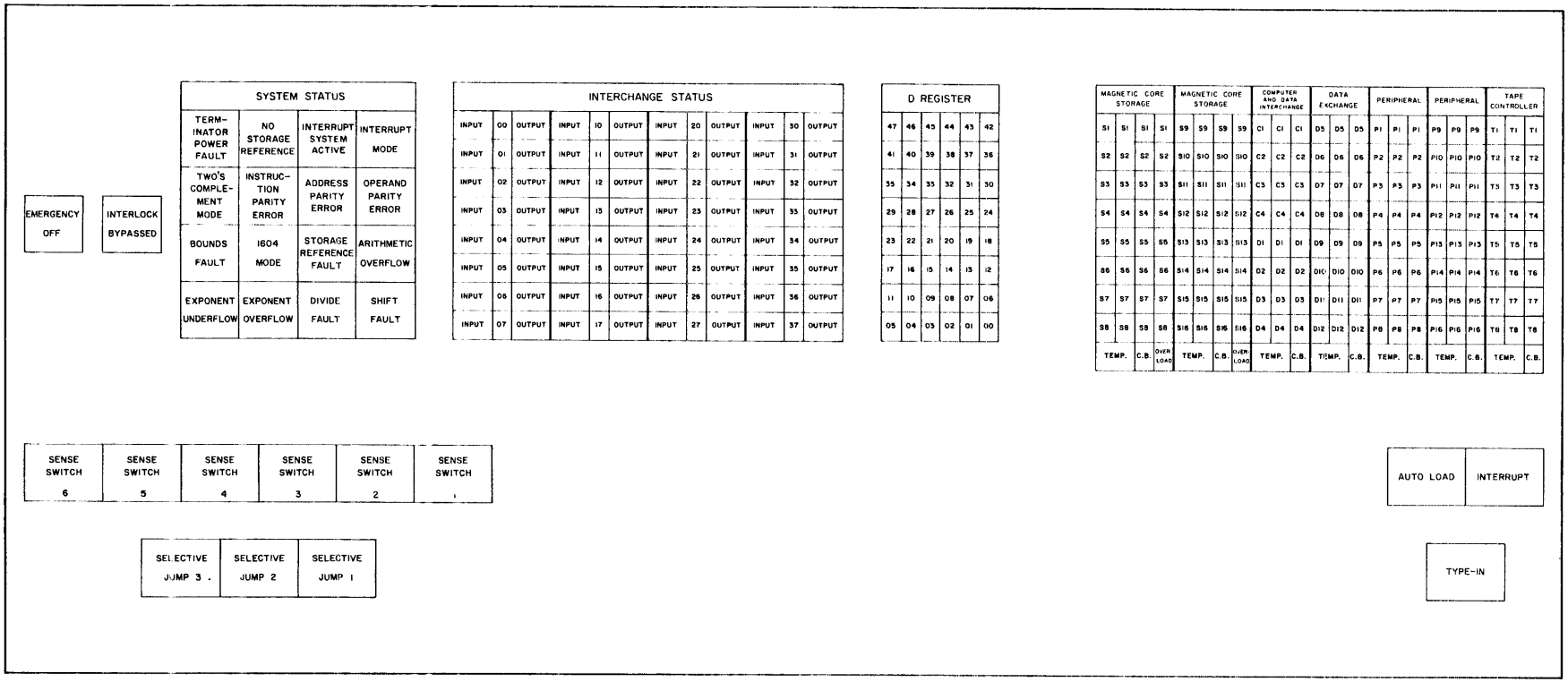


Figure 7-3. Operator Section of Console

Table 7-3. (Cont'd)

Display	Condition When Illuminated
Instruction Parity Error Light	A parity error has occurred in reading an instruction from storage to be executed in the 3604.
Operand Parity Error Light	A parity error has occurred in reading an operand from storage.
Storage Reference Fault Light	An attempt has been made to address a non-existent storage bank.
No Storage Reference Light	Indicates a deepend condition; computer has failed to complete an operation.
1604 Mode Light	Computer is in 1604 mode of operation.
Type In Light	Indicates an Input to A operation is to be performed, either with the console typewriter or the card reader. (If the card reader has been selected, the Card Input switch will also be illuminated.)
Terminator Power Fault Light	Terminator power has been lost or one side of the terminator power line has been shorted to ground.

Temperature and Circuit Breaker Lights

Various cabinets of the system are equipped with temperature sensing devices and circuit breakers to insure operations proceed without harm to the equipments.

Light indicators on the operator section of the console monitor these conditions. If the temperature in a given cabinet exceeds the normal range, an amber light comes on. If the temperature reaches a point which might harm the equipment, a red light comes on and the computer halts.

When a circuit breaker in a particular cabinet trips, light indicators for circuit breakers are illuminated red and the computer halts.

An Interlock Bypassed switch is provided to override stopping the computer in the cases outlined above. Pressing the Interlock Bypassed switch: (1) allows computation to proceed regardless of ambient temperature conditions, and (2) lights the Interlock Bypassed light on the console.

Switches

Several switches appear on the operator section of the console. These switches and their operations are listed in table 7-4.

Table 7-4. Operator Switches

Switch	No.	M or L*	Function
Sense Switches**	6	L	Each of the six Sense switches may be pressed to flag an internal condition. Pressing a Sense switch places a "1" in its matching bit position of the Miscellaneous Modes Selections register. The Sense switches are illuminated when pressed.
Selective Jump 1** Selective Jump 2** Selective Jump 3**	1 1 1	L L L	Pressing the Selective Jump switches provides the manual conditions for executing a program jump on the Selective Jump instruction (75, b = 1, 2, 3, 5, 6 or 7). The Selective Jump switches are illuminated when pressed.
Auto Load**	1	M	Pressing the Auto-Load switch provides for automatically loading storage with information from a given external equipment.
Manual Interrupt**	1	M	Pressing the Manual Interrupt switch forces the computer into an interrupt routine if: a) The Manual Interrupt bit in the Interrupt Mask register is set to "1", and b) The interrupt system is active.
Interlock Bypassed**	1	M	Pressing the Interlock Bypassed switch overrides stopping the computer on reaching abnormal temperature conditions. This switch is illuminated when pressed.
Emergency Off**	1	M	Pressing the Emergency Off switch removes power from all the system.

* M - momentary, L - locking

** Denotes switch is active when running or stopped; others are active only when stopped.

CONSOLE TYPEWRITER

The 731 IBM Selectric typewriter (mounted on the right wing of the console) is used as an input/output device with the 3600 system. The typewriter is connected directly to the computer and also to a 3606 data channel. The typeout speed is approximately 15 characters per second.

All input from the typewriter is done directly by the Input to A instruction. Output operations with the typewriter occur in the normal manner via the 3606 data channel and the 3602 communication module. The input/output facilities of the typewriter are entirely independent (see Input to A section, last paragraph).

SWITCHES AND INDICATORS

Equipment Number Switch (0 -- 7)

The Equipment Number switch designates the typewriter as equipment N. Bits 9, 10, and 11 of the Connect code must match the octal setting of this switch.

I/O Fake Reply Switch

This switch is used in ON position only during maintenance. A Ready signal will be available on the status lines only if this switch is in the OFF position.

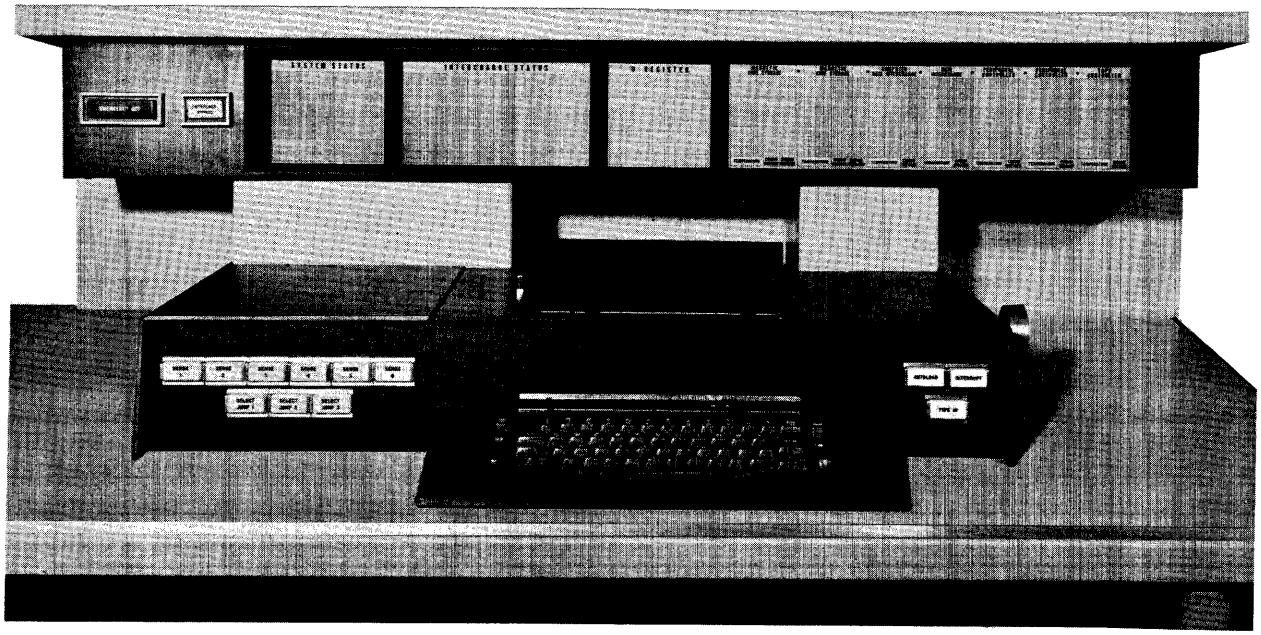


Figure 7-4. Console Typewriter

Transmission Parity Indicator

1) In a Connect code:
   If a parity error occurs during output of a Connect code, the red Transmission Parity Error indicator on the console typewriter chassis lights. The typewriter will not be connected and no signals will be returned to the 3606 data channel. This light will go out if an external master clear or a Clear Channel instruction is performed.

2) In a Function code:
   If a parity error occurs during output of a Function code, the red Transmission Parity Error indicator lights. A Parity Error signal is returned to the 3606 data channel if the typewriter is connected. This sets the I/O Parity Error bit in the data channel. The desired function will not be executed. The light goes out and the Parity Error signal drops when an external master clear or a Clear Channel instruction is performed.

3) During a Write operation:
   If a parity error occurs during a Write operation, the red Transmission Parity Error indicator lights. A Parity Error signal is returned to the 3606 which sets the I/O Parity Error bit in the data channel. The Write operation continues without stopping. The light goes out and the I/O Parity Error bit is cleared when an external master clear or a Clear Channel instruction is performed.

Even if a parity error occurs in any of these cases, an Input to A operation may proceed, or be initiated.

Connect Indicator

The white Connect light on the console typewriter chassis lights when the typewriter is connected to a data channel. This light goes out when the typewriter is no longer connected.

Typewriter On/Off Switch

The Typewriter On/Off switch is located on the Selectric typewriter. This switch must be in the ON position for any typewriter operation. The power for the typewriter control logic is applied when the system power is on.

Card Input Mode/Normal Switches

The pair of switches, Card Input Mode/Normal, determines whether the card reader or the console typewriter will have the use of the Input to A path. The Normal switch selects the typewriter.

Type In Indicator

The Type In indicator is located on the operator section of the console. When this light comes on, the program has reached an Input to A instruction and the Input to A operation may begin via the typewriter.

CODES

The codes in table 7-6 are used for both input and output operations. (See the Input to A section for additional comments on the codes in table 7-6.) The codes in table 7-5 are used mainly in output operations.

Table 7-5. Connect, Function, and Status Codes

CONNECT	
Connect Equipment N	NXXX
FUNCTION	
Set Interrupt On Abnormal Operation	XXX1
Clear Interrupt On Abnormal Operation	XXX2
Clear Interrupt	XXX4
STATUS	
Ready	XXX1
Busy	XXX2
Upper/Lower Case	XXX4
End of Line	XX4X
Type Parity Error	2XXX

Connect

Connect Equipment N (NXXX)

Bits 9, 10, and 11 of the Connect code must match the octal setting of the Equipment Number switch. If the switch setting and these bits do not agree, the typewriter will not be connected. No signals will be returned to the 3606 data channel. A new Connect code not equal to NXXX clears a previous connection.

Function

Set Interrupt on Abnormal Operation (XXX1)

If bit 0 is present in the Function code (accompanied by a Function signal), an Interrupt on Abnormal Operation condition is established in the typewriter. (In all discussion of codes, bit 0 is in the rightmost position, and bits are numbered from right to left in ascending order.) This allows the typewriter to send an Interrupt signal to the computer when either or both of two conditions occur:

1) End of Line
   The typewriter carriage has reached the right margin of the line it is typing as indicated by the right margin stop setting.

2) Type Parity Error
   A parity error has occurred in the typewriter logic. (See the Type Parity Error discussion under Status Lines for additional comments.)

The Interrupt signal is transmitted on the line which corresponds to the equipment number (N) of the typewriter.

Table 7-6. Console Typewriter Codes

Lower Case	Code	Upper Case	Lower Case	Code	Upper Case
A	12	A			
B	01	B	0 (zero)	43	)
C	11	C	1	77	±
D	55	D	2	37	@
E	51	E	3	33	#
F	30	F	4	47	$
G	74	G	5	57	%
H	45	H	6	13	¢
I	16	I	7	53	& (and)
J	70	J	8	17	* (asterisk)
K	15	K	9	07	(
L	41	L	!	76	° (degree)
M	72	M	.	32	.
N	31	N	'	52	"
O	42	O	;	50	:
P	54	P	,	14	,
Q	10	Q	/	44	?
R	56	R	=	34	+
S	46	S	Dash -	04	Underline _
T	75	T	Space	60	Space
U	35	U	Backspace	61	Backspace
V	36	V	Tab	62	Tab
W	02	W	C. R.	63	C. R.
X	71	X	U. C.	64	U. C.
Y	40	Y	L. C.	66	L. C.
Z	73	Z			

Clear Interrupt On Abnormal Operation (XXX2)

If bit 1 is present in the Function code, the Interrupt on Abnormal Operation condition is cleared.

Clear Interrupt (XXX4)

If bit 2 is present in the Function code, the Interrupt signal is cleared.

Status

Ready (XXX1)

If status bit 0 is present, the console typewriter is connected, the typewriter on/off switch is ON and the Fake I/O Reply switch is OFF.

Busy (XXX2)

If bit 1 is present in the Status code, the typewriter is doing one of the following operations:

1) Typing a character

2) Backspacing

3) Carriage return or tab

4) Shifting to upper case

5) Shifting to lower case

6) Processing a 00, 65, 67, or Correct Case code. If one or both 6-bit frames of the 12-bit word received by the typewriter during a normal output (Write) operation equal 00, 65, or 67, nothing is typed or spaced corresponding to that code. The time interval required to process these codes is approximately 3 μsec. This time interval also applies when an Upper/Lower Case code is specified and the typewriter is already shifted to the desired case.

Upper/Lower Case (XXX4)

If bit 2 is present in the Status code, the typewriter is in upper case. If bit 2 is not present in the Status code, the typewriter is in lower case.

End of Line (XX4X)

If bit 5 is present in the Status code, the typewriter has reached the end of a line as indicated by the right margin stop setting. During output operations there is nothing to prevent typing beyond the right margin stop. If this is done, the End of Line signal will remain up only for a distance of 7 to 8 characters beyond the right margin.

Type Parity Error (2XXX)

If bit 10 is present in the Status code, a type parity error has occurred. This indicates some logic has failed in the typewriter control and the character typed is not necessarily the one specified by the code. A type parity error can occur during an output operation. The Type Parity Error signal drops when an external master clear or a Clear Channel instruction is performed. It also drops if a Carriage Return or Clear Interrupt Function code is received.

PROGRAMMING

Output

The general order of events when using the typewriter for an output operation via a 3606 data channel is:

1) Set tabs, margins, and spacing. Turn on typewriter

2) Clear

3) Connect

4) Check status

5) Function

6) Write

Set Tabs, Margins, and Spacing

All tabs, margins, and paper spacing must be set manually prior to the output operation. A tab may be set for each space on the typewriter between the margins.

Clear

There are two types of clears which may be used to clear all conditions existing in the typewriter control. These are:

1) External Master Clear
   This signal is sent out on all data channels. In the case of the typewriter, it clears all functions, control logic, and an Interrupt signal if one exists. It also turns off the Transmission Parity Error light on the 3606 and a Parity Error signal if one exists.

2) Clear Channel Instruction
   This instruction performs the same operation as an external master clear, except it normally only applies to one data channel.

Connect

The 12-bit portion of the Connect instruction (accompanied by a Connect signal) connects the typewriter to a data channel. A Reply signal is returned to the 3606 when the connection has been made. There is no external reject under any circumstances. The connection is cleared by a Clear signal or another Connect code.

Check Status

The programmer may wish to check the status of the connected typewriter before proceeding. This is done with a Copy Status instruction. Status information is returned to the data channel on five of twelve status lines. The Bit Sensing instruction may be used to determine the status of the connected typewriter. If the programmer is certain of the status of the typewriter, this operation may be omitted.

Function

The 12-bit portion of the Function instruction (accompanied by a Function signal) performs a certain operation depending on the code. These codes are listed in table 7-6. Since only 1 bit of the Function code needs to be interpreted to perform a specified operation, it is possible to combine operations using one code. For example: $XXX6_8 = (XXX110)_2$ would Clear Interrupt on Abnormal Operation and Clear Interrupt.

Example:

XXX4 Clear Interrupt
XXX2 Clear Interrupt on Abnormal Operation
XXX6 Clear both

Write

A Write instruction starts the output operation using the codes listed in table 7-5. The upper 6 bits of the 12-bit data word received from the data channel are translated and the character matching the code is typed. Then the lower 6 bits of the code are translated, the character is typed, and a reply is returned to the data channel.

The typewriter is automatically placed in lower case at the beginning and end of every Write (output) operation (upper case can still be selected when the Write operation begins).

The typewriter keyboard does not have a lockout during an output operation. If a key, space bar, etc., is accidently pressed during output, the typewriter may miss a character or type something other than the normal output character.

Input To A

When the typewriter is used for an Input to A operation, the A register is initially cleared and 6 bits are entered into bit positions 0 through 5 in the A register. Nothing is entered into positions 6 through 47. Therefore, the A register cannot be used as an assembly register.

The order of events involved when using the typewriter for an Input to A operation is:

1) Set tabs, margins, and spacing. Turn on typewriter.

2) Press Normal switch (paired with Card Input Mode switch).

3) Wait for Input to A indicator to light.

4) Begin Input to A operation.

After setting tabs, margins, and spacing, and turning on the typewriter, press the Normal switch (paired with the Card Input Mode switch). The indicator on this switch lights when the typewriter is selected for an Input to A operation.

When the Type In indicator lights, the program is waiting for the Input to A operation and the operator may enter information on the typewriter.

The mechanical operations, such as backspace, carriage return, etc., enter their corresponding codes into the A register. For example: shifting from lower to upper case enters a 64_8; shifting from upper case to lower case enters a 66_8.

The program may determine whether the typewriter has the Input to A path by examining the Miscellaneous Modes Selections register. Bit 13 of this register indicates which of the two switches (Card Input Mode/Normal) has been pressed (refer to Miscellaneous Modes Selections register description).

Note that the program may also monitor the status of the typewriter during an Input to A operation by connecting the typewriter on the data channel and executing a Copy Status instruction.

An Input to A can occur during an output operation when all of the following three conditions are met:

1) The typewriter is connected for an output operation.

2) The output is in progress; i.e., information is being typed.

3) The main program in the computer encounters an Input to A instruction.

When step 3 is reached, the information which is being sent to the typewriter via the data channel will be returned to the A register of the computer.

APPENDIX SECTION

APPENDIX I
NUMBER SYSTEMS

Any number system may be defined by two characteristics, the radix or base and the modulus. The radix or base is the number of unique symbols used in the system. The decimal system has ten symbols, 0 through 9. Modulus is the number of unique quantities or magnitudes a given system can distinguish. For example, an adding machine with ten digits, or counting wheels, would have a modulus of 10^{10}. The decimal system has no modulus because an infinite number of digits can be written, but the adding machine has a modulus because the highest number which can be expressed is 9, 999, 999, 999.

Most number systems are positional, that is, the relative position of a symbol determines its magnitude. In the decimal system, a 5 in the units column represents a different quantity than a 5 in the tens column. Quantities equal to or greater than 1 may be represented by using the 10 symbols as coefficients of ascending powers of the base 10. The number 984_{10} is:

$$\begin{aligned} 9 \times 10^2 &= 9 \times 100 = 900 \\ +8 \times 10^1 &= 8 \times 10 = 80 \\ +4 \times 10^0 &= 4 \times 1 = 4 \\ &\hline 984_{10} \end{aligned}$$

Quantities less than 1 may be represented by using the 10 symbols as coefficients of ascending negative powers of the base 10. The number 0.593_{10} may be represented as:

$$\begin{aligned} 5 \times 10^{-1} &= 5 \times .1 = .5 \\ +9 \times 10^{-2} &= 9 \times .01 = .09 \\ +3 \times 10^{-3} &= 3 \times .001 = .003 \\ &\hline 0.593_{10} \end{aligned}$$

BINARY NUMBER SYSTEM

Computers operate faster and more efficiently by using the binary number system. There are only two symbols, 0 and 1; the base = 2. The following shows the positional value.

. . .	2^5	2^4	2^3	2^2	2^1	2^0	
	=32	=16	=8	=4	=2	=1	Binary point

The binary number 011010 represents:

$$\begin{aligned} 0 \times 2^5 &= 0 \times 32 = 0 \\ +1 \times 2^4 &= 1 \times 16 = 16 \\ +1 \times 2^3 &= 1 \times 8 = 8 \\ +0 \times 2^2 &= 0 \times 4 = 0 \\ +1 \times 2^1 &= 1 \times 2 = 2 \\ +0 \times 2^0 &= 0 \times 1 = 0 \\ &\hline 26_{10} \end{aligned}$$

Fractional binary numbers may be represented by using the symbols as coefficients of ascending negative powers of the base.

	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	. . .
Binary Point	=1/2	=1/4	=1/8	=1/16	=1/32	

The binary number 0.10110 may be represented as:

$$\begin{aligned} 1 \times 2^{-1} &= 1 \times 1/2 = 1/2 = 8/16 \\ +0 \times 2^{-2} &= 0 \times 1/4 = 0 = 0 \\ +1 \times 2^{-3} &= 1 \times 1/8 = 1/8 = 2/16 \\ +1 \times 2^{-4} &= 1 \times 1/16 = 1/16 = 1/16 \\ &\hline 11/16_{10} \end{aligned}$$

OCTAL NUMBER SYSTEM

The octal number system uses eight discrete symbols, 0 through 7. With the base 8 the positional value is:

. . .	8^5	8^4	8^3	8^2	8^1	8^0
	32,768	4,096	512	64	8	1

The octal number 513_8 represents:

$$\begin{aligned} 5 \times 8^2 &= 5 \times 64 = 320 \\ +1 \times 8^1 &= 1 \times 8 = 8 \\ +3 \times 8^0 &= 3 \times 1 = 3 \\ &\hline 331_{10} \end{aligned}$$

Fractional octal numbers may be represented by using the symbols as coefficients of ascending negative powers of the base.

8^{-1}	8^{-2}	8^{-3}	8^{-4}	. . .
1/8	1/4	1/512	1/4096	

The octal number 0.4520 represents:

$$4 \times 8^{-1} = 4 \times 1/8 = 4/8 = 256/512$$
$$+5 \times 8^{-2} = 5 \times 1/64 = 5/64 = 40/512$$
$$+2 \times 8^{-3} = 2 \times 1/512 = 2/512 = 2/512$$
$$298/512 = 149/256_{10}$$

ARITHMETIC

Addition and Subtraction

Binary numbers are added according to the following rules:

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = 0$ with a carry of 1

The addition of two binary numbers proceeds as follows (the decimal equivalents verify the result):

Augend	0111	(7)
Addend	+0100	+(4)
Partial Sum	0011	
Carry	1	
Sum	1011	(11)

Subtraction may be performed as an addition:

8	(minuend)	8	(minuend)
−6	(subtrahend) or	+4	(ten's comp. of sub.)
2	(difference)	2	(difference - omit carry)

The second method shows subtraction performed by the "adding the complement" method. The omission of the carry in the illustration has the effect of reducing the result by 10.

One's Complement

The 3604 performs all arithmetic operations in the binary one's complement mode. In this system, positive numbers are represented by the binary equivalent and negative numbers in one's complement notation.

The one's complement representation of a number is found by subtracting each bit of the number from 1. For example:

1111	
−1001	9
0110	(one's complement of 9)

This representation of a negative binary quantity may also be obtained by substituting "1's" for "0's" and "0's" for "1's".

The value zero can be represented in one's complement notation in two ways:

$0000 \rightarrow 00_2$ positive (+) zero

$1111 \rightarrow 11_2$ negative (−) zero

The rules regarding the use of these two forms for computation are:

1) Both positive and negative zero are acceptable as arithmetic operands.

2) If the result of an arithmetic operation is zero, it will be expressed as positive zero. The one exception to this rule is when negative zero is added to negative zero. In this case, the result is negative zero.

One's complement notation applies not only to arithmetic operations performed in A, but also to the modification of execution addresses. During address modification, the modified address will equal 77777_8 only if the unmodified exception address equals 77777_8 and b = 0 or $(B^b) = 77777_8$.

Two's Complement

The additive counter in the computer uses two's complement arithmetic. An additive counter is a register with provisions for increasing its contents by one (P register). A two's complement counter is open-ended; there is no end-around carry or borrow.

Positive numbers have the same representation in both systems; negative values differ by one count.

Count	Two's Comp. Rep.	One's Comp. Rep.
+2	00010	00010
+1	00001	00001
0	00000	00000
−1	11111	11110
−2	11110	11101

The difference in the representation of negative values in these two systems is due to the skipping of the all "1's" count in one's complement notation. In the one's complement system the end-around-carry feature of the register automatically changes a count of all "1's" to all "0's". (Note exception under One's Complement.)

As an example, if the content of a subtractive counter is positive seven (0111) and is to be reduced by one, add the two's complement expression of negative one (1111) to 0111 as shown below. The result is six.

$$\begin{array}{r} 0111 \\ +1111 \\ \hline 0110 \end{array}$$

Note that the two's complement expression for a negative number may also be formed by adding one to the one's complement representation of the number.

Multiplication

Binary multiplication proceeds according to the following rules:

$$0 \times 0 = 0$$
$$0 \times 1 = 0$$
$$1 \times 0 = 0$$
$$1 \times 1 = 1$$

Multiplication is always performed on a bit-by-bit basis. Carries do not result from multiplication since the product of any two bits is always a single bit.

Decimal example:

multiplicand	14	
multiplier	12	
partial products	28	
	14	(shifted one place left)
product	168_{10}	

The shift of the second partial product is a shorthand method for writing the true value 140.

Binary example:

multiplicand	(14)	1110	
multiplier	(12)	1100	
partial products		0000	shift to place digits in proper columns
		0000	
		1110	
		1110	
product	(168_{10})	10101000_2	

The computer determines the running subtotal of the partial products. Rather than shifting the partial product to the left to position it correctly, the computer right shifts the summation of the partial products one place before the next addition is made. When the multiplier bit is "1", the multiplicand is added to the running total and the results are shifted to the right (in effect, the quantity has been multiplied by 10_2).

Division

The following example shows the familiar method of decimal division:

		14	quotient
divisor	13	185	dividend
		13	
		55	partial dividend
		52	
		3	remainder

The computer performs division in a similar manner (using binary equivalents):

		1110	quotient (14)
divisor	1101	10111001	dividend
		1101	
		10100	partial dividend
		1101	
		1110	partial dividend
		1101	
		11	remainder (3)

However, instead of shifting the divisor right to position it for subtraction from the partial dividend (shown above), the computer shifts the partial dividend left, accomplishing the same purpose and permitting the arithmetic to be performed in the A register. The

computer counts the number of shifts, which is the number of quotient digits to be obtained; after the correct number of counts, the routine is terminated.

CONVERSIONS

The procedures that may be used when converting from one number system to another are power addition, double dabble, and substitution.

RECOMMENDED CONVERSION PROCEDURES (INTEGER AND FRACTIONAL)

Conversion	Recommended Method
Binary to Decimal	Power Addition
Octal to Decimal	Power Addition
Decimal to Binary	Double Dabble
Decimal to Octal	Double Dabble
Binary to Octal	Substitution
Octal to Binary	Substitution

GENERAL RULES

* $r_i > r_f$: use Double Dabble, Substitution

$r_i < r_f$: use Power Addition, Substitution

r_i = Radix of initial system

r_f = Radix of final system

Power Addition

To convert a number from r_i to r_f ($r_i < r_f$), write the number in its expanded r_i polynomial form and simplify using r_f arithmetic.

Example 1 Binary to Decimal (Integer)

$$\begin{aligned} 010\ 111_2 &= 1\ (2^4) + 0\ (2^3) + 1\ (2^2) + 1\ (2^1) + 1\ (2^0) \\ &= 1\ (16) + 0\ (8) + 1\ (4) + 1\ (2) + 1\ (1) \\ &= 16 + 0 + 4 + 2 + 1 \\ &= 23_{10} \end{aligned}$$

Example 2 Binary to Decimal (Fractional)

$$\begin{aligned} .0101_2 &= 0\ (2^{-1}) + 1\ (2^{-2}) + 0\ (2^{-3}) + 1\ (2^{-4}) \\ &= 0 + 1/4 + 0 + 1/16 \\ &= 5/16_{10} \end{aligned}$$

Example 3 Octal to Decimal (Integer)

$$\begin{aligned} 324_8 &= 3\ (8^2) + 2\ (8^1) + 4\ (8^0) \\ &= 3\ (64) + 2\ (8) + 4\ (1) \\ &= 192 + 16 + 4 \\ &= 212_{10} \end{aligned}$$

Example 4 Octal to Decimal (Fractional)

$$\begin{aligned} .44_8 &= 4\ (8^{-1}) + 4\ (8^{-2}) \\ &= 4/8 + 4/64 \\ &= 36/64_{10} \end{aligned}$$

Double Dabble

To convert a whole number from r_i to r_f ($r_i > r_f$):

1) Divide r_i by r_f using r_i arithmetic.

2) The remainder is the lowest order bit in the new expression.

3) Divide the integral part from the previous operation by r_f.

4) The remainder is the next higher order bit in the new expression.

5) The process continues until the division produces only a remainder which will be the highest order bit in the r_f expression.

To convert a fractional number from r_i to r_f:

1) Multiply r_i by r_f using r_i arithmetic.

2) The integral part is the highest order bit in the new expression.

3) Multiply the fractional part from the previous operation by r_f.

4) The integral part is the next lower order bit in the new expression.

5) The process continues until sufficient precision is achieved or the process terminates.

* r refers to the radix of the number system used.

Example 1 Decimal to Binary (Integer)

45 ÷ 2 = 22 remainder 1; record 1
22 ÷ 2 = 11 remainder 0; record 0
11 ÷ 2 = 5 remainder 1; record 1
5 ÷ 2 = 2 remainder 1; record 1
2 ÷ 2 = 1 remainder 0; record 0
1 ÷ 2 = 0 remainder 1; record 1

101101

Thus: $45_{10} = 101101_2$

Example 2 Decimal to Binary (Fractional)

.25 x 2 = 0.5; record 0
.5 x 2 = 1.0; record 1
.0 x 2 = 0.0; record 0

.010

Thus: $.25_{10} = .010_2$

Example 3 Decimal to Octal (Integer)

273 ÷ 8 = 34 remainder 1; record 1
34 ÷ 8 = 4 remainder 2; record 2
4 ÷ 8 = 0 remainder 4; record 4

421

Thus: $273_{10} = 421_8$

Example 4 Decimal to Octal (Fractional)

.55 x 8 = 4.4; record 4
.4 x 8 = 3.2; record 3
.2 x 8 = 1.6; record 1
--- --- -
--- --- -

Thus: $.55_{10} = .431\ldots_8$ .431 . . .

Substitution

This method permits easy conversion between octal and binary representations of a number. If a number in binary notation is partitioned into triplets to the right and left of the binary point, each triplet may be converted into an octal digit. Similarly each octal digit may be converted into a triplet of binary digits.

Example 1 Binary to Octal

Binary = 110 000 . 001 010
Octal = 6 0 . 1 2

Example 2 Octal to Binary

Octal = 6 5 0 . 2 2 7
Binary = 110 101 000 . 010 010 111

COMMON PURE NOTATIONS

Decimal	Binary	Octal
00	00000	00
01	00001	01
02	00010	02
03	00011	03
04	00100	04
05	00101	05
06	00110	06
07	00111	07
08	01000	10
09	01001	11
10	01010	12
11	01011	13
12	01100	14
13	01101	15
14	01110	16
15	01111	17
16	10000	20
17	10001	21

POWERS OF COMMON NUMBER SYSTEMS

$2^0 = 1$	$8^0 = 1$	$10^0 = 1$
$2^1 = 2$	$8^1 = 8$	$10^1 = 10$
$2^2 = 4$	$8^2 = 64$	$10^2 = 100$
$2^3 = 8$	$8^3 = 512$	$10^3 = 1,000$
$2^4 = 16$	$8^4 = 4,096$	$10^4 = 10,000$
$2^5 = 32$	$8^5 = 32,768$	$10^5 = 100,000$
$2^6 = 64$	$8^6 = 262,144$	$10^6 = 1,000,000$
$2^7 = 128$	$8^7 = 2,097,152$	
$2^8 = 256$	$8^8 = 16,777,216$	
$2^9 = 512$		
$2^{10} = 1,024$		

FIXED POINT AND FLOATING POINT NUMBERS

Any number may be expressed in the form kB^n, where k is a coefficient, B a base number, and the exponent n the power to which the base number is raised.

A fixed point number assumes:

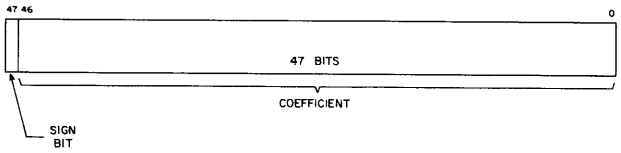
1) The exponent n = 0 for all fixed point numbers.

2) The coefficient k occupies the same bit positions within the computer word for all fixed point numbers.

3) The radix (binary) point remains fixed with respect to one end of the expression.

A 3604 fixed point number consists of a sign bit and coefficient as shown below. The upper bit of any 3604 fixed point number designates the sign of the coefficient (47 lower order bits). If the bit is "1", the quantity is negative since negative numbers are represented in one's complement notation; a "0" sign bit signifies a positive coefficient.

FIXED POINT FORMAT



The coefficient may be an integer or fraction. The radix (binary) point, in the case of an integer, is assumed to be immediately to the right of the lowest order bit (00). In the case of the fraction, the point is just to the right of the sign bit.

In many instances, the values in a fixed point operation may be too large or too small to be expressed by the computer. The programmer must position the numbers within the word format so they can be represented with sufficient precision. The process, called scaling, consists of shifting the values a predetermined number of places. The numbers must be positioned far enough to the right in the register to prevent overflow but far enough to the left to maintain precision. The scale factor (number of places shifted) is expressed as the power of the base. For example, $5,100,000_{10}$ may be expressed as 0.51×10^7, 0.051×10^8, 0.0051×10^9, etc. The scale factors are 7, 8, and 9.

Since only the coefficient is used by the computer, the programmer is responsible for remembering the scale factors. Also, the possibility of an overflow during intermediate operations must be considered. For example, if two fractions in fixed point format are multiplied, the result is a number less than 1.

If the same two fractions are added, subtracted, or divided, the result may be greater than 1 and an overflow will occur. Similarly, if two integers are multiplied, divided, subtracted or added, the likelihood of an overflow is apparent.

As an alternative to fixed point operation, a method involving a variable radix point, called floating point, is used. This significantly reduces the amount of bookkeeping required on the part of the programmer.

By shifting the radix point and increasing or decreasing the value of the exponent, widely varying quantities which do not exceed the capacity of the machine may be handled.

Floating point numbers within the computer are represented in a form similar to that used in "scientific" notation, that is, a coefficient or fraction multiplied by a number raised to a power. Since the computer uses only binary numbers, the numbers are multiplied by powers of two.

$F \times 2^E$ where: F = fraction
E = exponent

In floating point, different coefficients need not relate to the same power of the base as they do in fixed point format. Therefore, the construction of a floating point number includes not only the coefficient but also the exponent.

Coefficient

The single precision coefficient consists of a 36-bit fraction in the 36 lower order positions of the floating point word. The coefficient is a normalized fraction; it is equal to or greater than ½ but less than 1. The highest order bit position (47) is occupied by the sign bit of the coefficient. If the sign bit is a "0", the coefficient is positive; a "1" bit denotes a negative fraction (negative fractions are represented in one's complement notation).

Exponent

The floating point exponent is expressed as an 11-bit quantity with a value ranging from 0000_8 to 3777_8. Within this range, both positive and negative exponents must be expressed. Biasing the exponent provides the ability to distinguish between positive and negative exponents. It is formed by adding a true positive exponent and a bias of 2000_8 or a true negative exponent and a bias of 1777_8. This results in a range of biased exponents as shown below.

True Positive Exponent	Biased Exponent	True Negative Exponent	Biased Exponent
+0	2000	−0	2000*
+1	2001	−1	1776
+2	2002	−2	1775
---	----	---	----
---	----	---	----
+1776	3776	−1776	0001
$+1777_8$	3777_8	-1777_8	0000_8

When bias is used with the exponent, floating point operation is more versatile since floating point operands can be compared with each other in the normal fixed point mode.

As an example, compare the unbiased exponents of $+52_8$ and $+0.02_8$ (example 1).

Example 1, Number = +52

0	00 000 000 110	(36 bits)
Coefficient Sign	Exponent	Coefficient

Number = +0.02

0	11 111 111 011	(36 bits)
Coefficient Sign	Exponent	Coefficient

In this case, +0.02 appears to be larger than +52 because of the larger exponent. If, however, both exponents are biased (example 2), changing the sign of both exponents makes +52 greater than +0.02.

Example 2, Number = $+52_8$

0	10 000 000 110	(36 bits)
Coefficient Sign	Exponent	Coefficient

Number = $+0.02_8$

0	01 111 111 011	(36 bits)
Coefficient Sign	Exponent	Coefficient

Conversion Procedures

Fixed Point to Floating Point

1) Express the number in binary.

2) Normalize the number. A normalized number has the most significant 1 positioned immediately to the right of the binary point and is expressed in the range $1/2 \leq k < 1$.

3) Inspect the sign of the true exponent. If the sign is positive, add 2000_8 (bias) to the true exponent of the normalized number. If the sign is negative, add the bias 1777_8 to the true exponent of the normalized number. In either case, the resulting exponent is the biased exponent.

4) Assemble the number in floating point.

5) Inspect the sign of the coefficient. If negative, complement the assembled floating point number to obtain the true floating point representation of the number. If the sign of the coefficient is positive, the assembled floating point number is the true representation.

Example 1 Convert +4.0 to Floating Point

1) The number is expressed in octal.

* Minus zero is sensed as positive zero by the computer and is therefore biased by 2000_8 rather than 1777_8.

2) Normalize. $4.0 = 4.0 \times 8^0 = 0.100 \times 2^3$.

3) Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 3.

4) Assemble number in floating point format.
Coefficient = 400 000 000 000_8
Biased exponent = 2003_8
Assembled word = 2003 400 000 000 000_8

5) Since the sign of the coefficient is positive, the floating point representation of +4.0 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

Example 2 Convert −4.0 to Floating Point

1) The number is expressed in octal.

2) Normalize. $-4.0 = -4.0 \times 8^0 = -0.100 \times 2^3$.

3) Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 3.

4) Assemble number in floating point format.
Coefficient = 400 000 000 000_8
Biased exponent = 2003_8
Assembled word = 2003 400 000 000 000_8

5) Since the sign of the coefficient is negative, the assembled floating point word must be complemented. Therefore, the true floating point representation for −4.0 = 5774 377 777 777 777_8.

Example 3 Convert 0.5_{10} to Floating Point

1) Convert to octal. $0.5_{10} = 0.4_8$.

2) Normalize. $0.4 = 0.4 \times 8^0 = 0.100 \times 2^0$.

3) Since the sign of the true exponent is positive, add 2000_8 (bias) to the true exponent. Biased exponent = 2000 + 0.

4) Assemble number in floating point format.
Coefficient = 400 000 000 000_8
Biased exponent = 2000_8
Assembled word = 2000 400 000 000 000_8

5) Since the sign of the coefficient is positive, the floating point representation of $+ 0.5_{10}$ is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word. This example is a special case of floating point, since the exponent of the normalized number is 0 and could be represented as −0. The exponent would then be biased by 1777_8 instead of 2000_8 because of the negative exponent. The 3604, however, recognizes −0 as +0 and biases the exponent by 2000_8.

Example 4 Convert 0.04_8 to Floating Point

1) The number is expressed in octal.

2) Normalize. $0.04 = 0.04 \times 8^0 = 0.4 \times 8^{-1} = 0.100 \times 2^{-3}$.

3) Since the sign of the true exponent is negative, add 1777_8 (bias) to the true exponent. Biased exponent = $1777_8 + (-3) = 1774_8$.

4) Assemble number in floating point format.
Coefficient = 400 000 000 000_8
Biased exponent = 1774_8
Assembled word = 1774 400 000 000 000_8

5) Since the sign of the coefficient is positive, the floating point representation of 0.04_8 is as shown. If, however, the sign of the coefficient were negative, it would be necessary to complement the entire floating point word.

Floating Point to Fixed Point

1) If the floating point number is negative, complement the entire floating point word and record the fact that the quantity is negative. The exponent is now in a true biased form.

2) If the biased exponent is equal to or greater than 2000_8, subtract 2000_8 to obtain the true exponent. If less than 2000_8, subtract 1777_8 to obtain true exponent.

3) Separate the coefficient and exponent. If the true exponent is negative, the binary point should be moved to the left the number of bit positions indicated by the true exponent. If the true exponent is positive, the binary point should be moved to the right the number of bit positions indicated by the true exponent.

4) The coefficient has now been converted to fixed binary. The sign of the coefficient will be negative if the floating point number was complemented in step 1. (The sign bit must be extended if the quantity is placed in a register.)

5) Represent the fixed binary number in fixed octal notation.

Example 1 Convert Floating Point Number 2003 400 000 000 000$_8$ to Fixed Octal

1) The floating point number is positive and remains uncomplemented.

2) The biased exponent > 2000$_8$, therefore subtract 2000$_8$ from the biased exponent to obtain the true exponent of the number. 2003 - 2000 = +3.

3) Coefficient = 400 000 000 000$_8$ = .100$_2$. Move binary point to the right three places. Coefficient = 100.0$_2$.

4) The sign of the coefficient is positive because the floating point number was not complemented in step 1.

5) Represented in fixed octal notation. 100.0 x 2^0 = 4.0 x 8.0

Example 2 Convert Floating Point Number 5774 377 777 777 777$_8$ to Fixed Octal

1) The sign of the coefficient is negative, therefore complement the floating point number. Complement = 2003 400 000 000 000$_8$

2) The biased exponent (in complemented form) is > 2000$_8$, therefore subtract 2000$_8$ from the biased exponent to obtain the true exponent of the number. 2003 - 2000 = +3.

3) Coefficient = 400 000 000 000$_8$ = 0.100$_2$ Move binary point to the right three places. Coefficient = 100.0$_2$

4) The sign of the coefficient will be negative because the floating point number was originally complemented.

5) Convert to fixed octal. -100.0$_2$ = -4.0$_8$.

Example 3 Convert Floating Point Number 1774 400 000 000 000$_8$ to Fixed Octal

1) The floating point number is positive and remains uncomplemented.

2) The biased exponent < 2000$_8$, therefore subtract 1777$_8$ from the biased exponent to obtain the true exponent of the number. 1774$_8$ - 1777$_8$ = -3.

3) Coefficient = 400 000 000 000$_8$ = .100$_2$ Move binary point to the left three places. Coefficient = .000100$_2$

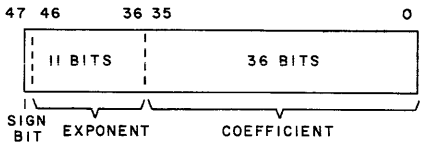
4) The sign of the coefficient is positive because the floating point number was not complemented in step 1.

5) Represent in fixed octal notation. .000100$_2$ = .04$_8$.

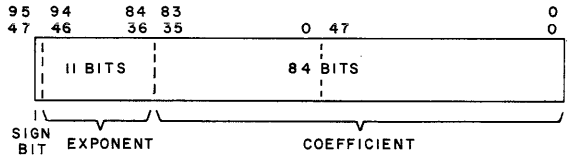
Notes on 3600 Floating Point Operations

1) All Floating Point Instructions
   a) 3600 floating point numbers have the following format.

SINGLE PRECISION



DOUBLE PRECISION



Single precision results are in the A register and double precision results are in the A and Q registers.

In double precision floating point operations, 48 additional bits of coefficient are added to form a 96-bit floating point word. The upper 36 bits of coefficient are of greater significance than the lower 48 bits. The highest order bit (95) is the sign bit of the 84-bit coefficient.

In arithmetic operations using double precision, the 96-bit operand is read from consecutive storage locations M and M + 1. Refer to Order of Instructions section, Double Precision Arithmetic.

b) Floating point instructions follow the sequence shown below.

reference memory
arithmetic operation
round
normalize
end sign correction

c) Operands do not have to be normalized. However, in an FDV operation, the coefficient of the dividend must be less than two times the coefficient of the divisor; if not, a divide fault will result.

d) When a right shift is required to normalize, it is performed whether the augment bit t1 (t1 = 1 for un-normalized arithmetic) is set or not.

e) When a number is shifted left to normalize, sign bits are entered into the least significant positions.

f) In single precision, the Q register is never touched during normalize or round operations.

g) In single precision instructions, if the normalize count is checked to obtain the proper significance, meaning can be attached to the residue in Q.

h) On an exponent underflow, A and Q are cleared after the end sign correction. On an exponent overflow, the result is left as formed.

i) When the coefficient of the result is ±0 the exponent portion of A is cleared before the end sign correction.

j) The 2's complement mode is disabled during floating point operations.

h) Rounding in single precision is accomplished by adding ±1 to the A register; in double precision by adding ±1 to the Q register.

2) FAD

a) The operand with the smallest exponent is placed in A and is shifted right into Q until the exponents are equal. The information (residue) in Q is untouched during the rest of the instruction. If the exponents are equal initially, Q is cleared and Q^{47} is set equal to A^{47}.

b) The round decision (to add -1, 0 or +1) is made after the shift to equalize exponents and before the addition of the coefficients.

	A^{47}		Q^{47}
Add +1 if	0		1
Add -1 if	1		0
Add 0 if	A^{47}	=	Q^{47}

Note that when the exponents are equal, $A^{47} = Q^{47}$ (See a above). Therefore, there is never a round.

3) DFAD

a) The operand with the smallest exponent is placed in AQ and is shifted right end off. The last bit shifted off is retained in the Round Flip Flop. If the exponents are equal, the machine will never round.

b) Sec. 2-b applies here except the Round Flip Flop should be substituted for Q^{47}.

4) FMU

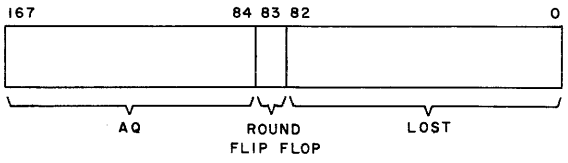
a) The magnitudes of the operands are multiplied to obtain a 72 bit product in AQ. Except for end sign correction (Q carries the sign of the result) Q is not changed after the iterative step.

b) The round decision (to add 0 or +1) is made after the iterative step. Since the result at this time is always positive, the decision is simpler than in FAD.

	A^{47}		Q^{47}
Add +1	0		1
Add 0	A^{47}	=	Q^{47}

5) DFMU

a) The magnitudes of the operands are multiplied to obtain a 168 bit product. The bits are distributed as shown below.



b) Sec. 4-b applies here except the Round Flip Flop should be substituted for Q^{47}.

6) FDV

a) The magnitudes of the operands are divided to obtain a 36 bit coefficient in A and a 36 bit remainder in the lower part of Q. After the iterative step, two times the remainder is compared with the divisor. This is essentially another iteration in divide step. If 2 x remainder $\geq$ divisor the Round Flip Flop is set.

b) The round decision (to add 0 or +1) is made after the probe to set the Round flip flop. Since the result at this time is always positive, the decision is simpler than in FAD.

	A^{47}		Round Flip Flop
Add +1	0		1
Add 0	A^{47}	=	RND

c) During the normalize operation, the remainder in Q is untouched. Proper significance can be attached to the remainder by checking the normalize count. The remainder carries the sign of the result.

7) DFDV

a) The magnitudes of the operands are divided to obtain a 84 bit coefficient in AQ. The setting conditions for the Round Flip Flop are the same as those in FDV. The remainder is lost.

b) Sec. 6-b applies here.

8) FSB, DFSB

FSB and DFSB are identical to FAD and DFAD except the operands read from memory are complemented before the execution of the instruction.

APPENDIX II
INTERRUPTIBLE CONDITIONS AND FAULTS

Under certain internal conditions in the execution of a computer program, faults may occur. Most of these fault conditions are associated with a particular bit of the Interrupt register, and may be tested in an interrupt routine. Faults which are not associated with the interrupt register are denoted by an asterisk. In most cases, a fault condition does not stop operation, but a visual indication of a fault occurrence is provided on the console.

SHIFT FAULT

When a single register shift of more than 48_{10} places or a double register shift of more than 96_{10} places is specified by the shift count in a Shift instruction, bit 00 of the Interrupt register is automatically set to "1". The Shift operation does not occur under these conditions. This Interrupt Register bit remains set until: (1) an Internal Function instruction (77.0 00001 - Clear Shift Fault Interrupt) is executed, or (2) a manual internal master clear is performed.

A shift fault lights an indicator on the console.

DIVIDE FAULT

Bit 01 of the Interrupt register is automatically set to "1" when:

1) The absolute value of the quotient resulting from a Divide Integer instruction is $\geq 2^{47}$,

2) The absolute value of the result of a Divide Fractional instruction is ≥ 1, or

3) A fixed point or floating point divide by zero is executed except when the dividend is zero.

4) A floating point divide is executed with the dividend $\geq$ two times the divisor.

This Interrupt Register bit remains set until:

1) The Internal Function instruction (77.0 00002 - Clear Divide Fault Interrupt) is executed, or

2) A manual internal master clear is performed.

A divide fault lights an indicator on the console.

EXPONENT OVERFLOW FAULT

Bit 02 of the Interrupt register is automatically set to "1" when the value of the exponent formed in a floating point add, subtract, multiply, or divide is $> 2^{10}-1$ (1777_8). This bit is also set whenever a floating point divide by zero is executed. This Interrupt Register bit remains set until:

1) The Internal Function instruction (77.0 00003 - Clear Exponent Overflow Interrupt) is executed, or

2) A manual internal master clear is performed.

An exponent overflow fault lights an indicator on the console.

EXPONENT UNDERFLOW FAULT

Bit 03 of the Interrupt register is automatically set to "1" when the value of the exponent formed in a floating point add, subtract, multiply, or divide is $< 2^{-10}-1$ (-1777_8). This Interrupt Register bit remains set until:

1) The Internal Function instruction (77.0 00004 - Clear Exponent Underflow Fault) is executed, or

2) A manual internal master clear is performed.

An exponent underflow fault lights an indicator on the console.

ARITHMETIC OVERFLOW FAULT

Bit 04 of the Interrupt register is automatically set to "1" when:

1) The absolute value of the sum or difference of two fixed point integers is $\geq 2^{47}$, or

2) The absolute value of the sum or difference of two fixed point fractions is ≥ 1. This Interrupt Register bit remains set until:

   a) The Internal Function instruction (77.0 00005 - Clear Arithmetic Overflow Interrupt) is executed, or

   b) A manual internal master clear is performed.

An arithmetic overflow fault lights an indicator on the console.

DIRECT INTERRUPT

Bits 05 and 06 of the Interrupt register may be set to "1" when another equipment in the system (usually a computer) is causing an interrupt.

These bits remain set until cleared by the attached equipment.

INTERNAL REJECT INTERRUPT

Bit 07 of the Interrupt register is automatically set to "1" when an Internal Reject signal is generated by the 3604. Internal Reject signals are generated when an external equipment fails to send an External Reject signal to the 3604. Internal Reject signals are generated in the following cases:

1) In the execution of Connect and Function instructions --

   a) The equipment or unit referenced is not attached to the specified channel, or

   b) The equipment or unit referenced is attached to the specified channel, but its power is off.

2) In the execution of Read and Write instructions -- the 3602 specified by the upper octal digit of the channel designator is not attached to the system.

3) Some failure in the data channel itself prevents generating an External Reject signal.

This Interrupt Register bit remains set until:

1) The Internal Function instruction (77.0 00007 - Clear Internal Reject Interrupt) is executed, or

2) A manual internal master clear is performed.

REAL TIME CLOCK INTERRUPT

Bit 08 of the Interrupt register is automatically set to "1" when the contents of the Time register and the Time Limit register become equal. This bit remains set until:

1) Cleared by an Internal Function instruction (77.0 00011 - Clear Real Time Clock Interrupt), or

2) An internal master clear is performed.

STORAGE REFERENCE FAULT

Bit 09 of the Interrupt register is automatically set to "1" when a storage reference is attempted to a non-existent storage bank (e.g., a reference is made to storage bank 6 and only three storage banks are in the system). If the Storage Reference Fault bit in the Interrupt Mask register is not set, the computer will stop.

This Interrupt Register bit remains set until:

1) The Internal Function instruction (77.0 00012 - Clear Storage Reference Fault Interrupt) is executed, or

2) Cleared by a manual internal master clear.

1604 MODE

Most programs written for the CONTROL DATA 1604 Computer can be executed by the 3600 computing system through the 1604 compatibility package. Since the function codes 00, 62, 63, 74, and 77 refer to different instructions in the 1604 and 3600, these instructions in a 1604 program must be executed interpretively by the 3600. This is accomplished by interrupting after interpreting the function code, but before executing an instruction with a function code of 00, 62, 63, 74, or 77.

The instruction is then interpretively executed in the interrupt routine.

Bit 10 (1604 Mode bit) of the Interrupt register is automatically set to "1" when:

1) The interrupt system is active,

2) Bit 10 of the Interrupt Mask register is set to "1" (select 1604 mode), and

3) The computer is in the Interrupt mode, after interpreting a 00, 62, 63, 74, or 77 function code.

This Interrupt Register bit remains set until:

1) The Internal Function instruction (77.0 00013 - Clear 1604 Mode Interrupt) is executed, or

2) A manual internal master clear is performed.

TRACE MODE

The 3600 computing system can, by using the interrupt system, trace all jumps occurring in a program. When interrupt occurs, the jump is executed by the interrupt routine. This feature, called tracing, is useful in debugging complex programs.

Bit 11 of the Interrupt register is automatically set to "1" when:

1) The interrupt system is active,

2) Bit 11 of the Interrupt Mask register is set to "1" (select Trace mode), and

3) The instruction being processed will result in a jump.

This Interrupt Register bit remains set until:

1) The Internal Function instruction (77.0 00014 - Clear Trace Mode Interrupt) is executed, or

2) A manual internal master clear is performed.

BOUNDS FAULT

Bit 12 of the Interrupt register is automatically set to "1" when:

1) The interrupt system is active,

2) Bit 12 of the Interrupt Mask register is set to "1" (check bounds), and

3) A write reference is attempted out of bounds (the bounds addresses are defined by the contents of the 37-bit Bounds register), or a jump is attempted out of bounds, or an attempt is made to read an instruction from out of bounds. (When 1604 Mode is selected and conditions (1) and (2) are met, any Read reference from out of bounds sets the Interrupt Register bit.)

This Interrupt Register bit remains set until:

1) The Internal Function instruction (77.0 00015 - Clear Bounds Interrupt) is executed, or

2) A manual internal master clear is performed.

A bounds fault lights an indicator on the console.

ILLEGAL INSTRUCTION FAULT

Bit 13 of the Interrupt register is automatically set to "1" when bit 13 of the Interrupt Mask register is set to "1" (the Interrupt system is active) and:

a) The Fault instruction (77.7) is executed, or

b) The Perform Algorithm instruction is executed, and no algorithm module is attached to the system, or

c) The Inter-Register instruction is executed with 'q' or 'r' equal to zero, or

d) An Input/Output instruction (74.0 – 74.6) is executed and the I/O Illegal Instruction bit is set.

This Interrupt Register bit remains set until:

1) The Internal Function instruction (77.0 00016 - Clear Illegal Instruction Interrupt) is executed, or

2) A manual internal master clear is performed.

OPERAND PARITY ERROR

Bit 14 of the Interrupt register is automatically set to "1" if a parity error occurs when an operand is read from storage.

This Interrupt Register bit remains set until:

1) The Internal Function instruction (77.0 00017 - Clear Operand Parity Error Interrupt) is executed, or

2) A manual internal master clear is performed.

An operand parity error lights an indicator on the console.

MANUAL INTERRUPT

When the Manual Interrupt button on the console is pressed, bit 15 of the Interrupt register is automatically set to "1". This bit remains set until:

1) The Internal Function instruction (77.0 00020 - Clear Manual Interrupt) is executed, or

2) A manual internal master clear is performed.

* ADDRESS PARITY ERROR

If a parity error occurs on an address transmitted to storage for a read or write reference, the Address Parity Error bit is set. Since this bit is not in the Interrupt register, an interrupt cannot occur on an address parity error.

Setting the Address Parity Error bit halts computer operation and lights an indicator on the console.

The Address Parity Error bit remains set until cleared by a manual internal master clear.

* INSTRUCTION PARITY ERROR

If a parity error occurs when reading an instruction from storage, the Instruction Parity Error bit is automatically set to "1". Setting this bit halts computer operation and lights an indicator on the console. Since this bit is not in the Interrupt register, an interrupt cannot occur on an instruction parity error.

The Instruction Parity Error bit remains set until cleared by a manual internal master clear.

I/O PARITY ERROR

This bit, located in the 3606 data channel, is set to "1" when:

1) An address parity error has occurred,

2) A data parity error has occurred, or

3) A data channel transmission parity error has occurred.

If this bit is set by one of the above conditions, operations using this data will result in error.

The I/O Parity Error bit remains set until:

1) An external master clear (from the console) is performed,

2) The Clear Channel instruction is executed, or

3) A Function instruction clears the selection of interrupt on error.

APPENDIX III
3600 SOFTWARE SYSTEMS

SCOPE

SCOPE is a supervisory control system which facilitates job processing and simplifies programming and operating. SCOPE includes the following features:

JOB PROCESSING

equipment assignments
memory allocation
subprogram loading and linking
overlay processing

DEBUGGING AIDS

error diagnostics
octal corrections facility
debugging dumps
memory map

INPUT/OUTPUT CONTROL

input/output routines
external interrupt control
tape handling
request stacking

SPECIAL STATEMENTS AND REQUESTS

internal interrupt control
sampling of current equipment status
preparation of a new library tape
editing of an existing library tape

SCOPE is programmed with control statements and requests. Control statements direct equipment assignments, compilations, assemblies, execution, data transfer, and library preparation. They are punched into cards and placed before and after the programs which they control. Requests are assembly language pseudo instructions which are assembled into calls to SCOPE routines. These routines do input/output processing, interrupt control, equipment status sampling and numerous other jobs.

Jobs to be run on the 3600 computer, together with the necessary SCOPE control cards, are placed on a designated input unit. The jobs are then processed sequentially by SCOPE with a minimum of intervention by the operator. Using the control statements and requests, SCOPE keeps accounting information, makes equipment assignments, initiates compilations, assemblies, and executions, processes input/output requests and provides recovery dumps for programs which terminate abnormally.

COMPASS

COMPASS is a comprehensive assembly system for the CONTROL DATA 3600 Computer. Operating within the SCOPE monitor system, COMPASS provides a convenient form for writing machine language programs. The assembly system accepts as input cards or card images containing assembly language instructions and produces relocatable binary programs as output. These programs are punched into cards or written as card images on magnetic tape and may be loaded by SCOPE into any section of memory for execution. A listing of the assembled program and a compressed symbolic output deck may also be obtained from an assembly.

COMPASS programs are composed of one or more subprograms; each is compiled independently, but may communicate with other subprograms. A subprogram is composed of a sequence of symbolic machine instructions and pseudo instructions. Symbolic machine instructions are alphabetic codes for the 3600 machine language instructions. The address fields in mnemonic instructions may contain constants, variables, arithmetic expressions, or literal expressions.

COMPASS pseudo instructions are used for the following operations:

identifying subprograms
linking subprograms
reserving storage locations
defining data
controlling the assembler
defining blocks of instructions by a single name

FORTRAN

FORTRAN is a mathematically oriented programming language designed to simplify programming while efficiently utilizing the 3600 instruction set. The FORTRAN complier accepts as input cards or card images on magnetic tape and produces as output relocatable binary programs.

A FORTRAN program is composed of one or more independently compiled subprograms, each of which is written as a series of FORTRAN statements. FORTRAN statements are translated by the compiler into machine language instructions. Both FORTRAN and COMPASS subprograms can be included in a single program. FORTRAN statements are used to transfer control from one subprogram to another and for transmitting data between subprograms. Some typical FORTRAN statements are shown below:

Replacement statement: A = Expression

This statement specifies that the expression is evaluated and the result stored in the location labeled A. An expression is composed of constants and variables, connected by arithmetic, logical or masking operations. Constants and variables may be defined as integer, real, double precision, complex, or logical quantities.

Data allocation: DIMENSION A(n_1, n_2, n_3), . . . , P(m_1, m_2, m_3)

This statement reserves memory locations for the arrays A, . . . , P. Arrays may have one, two or three dimensions; each element of the array is referenced by the array name and its subscripts. For example, the element in row 6, column 5, plane 3 of the array A is A (6, 5, 3).

Control statement: GO TO (n_1, n_2, . . . , n_m),i

Control will be transferred to one of the statements n_1, . . . , n_m according to the value of i. This value is set by other FORTRAN statements.

Input statement: BUFFER IN (i,p) LIST

This statement initiates the input of one record from logical unit i to the locations given in the LIST. The record parity is indicated by the parameter p.

COBOL

COBOL is a problem-oriented compiling system especially designed for business data processing. Programs written in a language which resembles English are translated into 3600 machine language programs.

COBOL programs are composed of a sequence of sentences, arranged into the four divisions of the COBOL language; IDENTIFICATION, ENVIRONMENT, DATA, and PROCEDURE.

The IDENTIFICATION division identifies the program and supplies information to the monitor accounting routine. The ENVIRONMENT division describes the computer which will be used for executing the program. The DATA division describes the organization and format of the data which the program is to process, and the PROCEDURE division describes the actual processing of the data.

Compilation and execution of a COBOL program are handled by the SCOPE operating system. COBOL statements punched into cards are submitted along with control statements for processing under the monitor system. The COBOL compiler produces a relocatable binary program which may be executed immediately and may also be saved on magnetic tape or on cards for later execution.

SIMSCRIPT

SIMSCRIPT is a language used to simulate dynamic processes. It is designed to facilitate the writing of such programs as those which trace the performance of alternate system configurations for a manufacturing process or simulate the flow of crude oil through a cracking plant. A SIMSCRIPT program consists of (1) descriptions of entities, like manufacturing equipment or oil products, and attributes of the entities, like cost, speed, yield per barrel, (2) subprograms that describe different events which can occur and how they will affect other events (3) a report generator for which the programmer specifies the form, content, and frequency of the output report.

ALGOL

A version of ALGOL-60, operating under the SCOPE monitor system, will be available for the 3600. This system will include the following features:

full call-by-name facility
dynamic arrays
separately compiled procedures
ability to call FORTRAN subroutines and functions
versatile input/output routines

Input to the compiler can use either the 42- or 62-characters set.

Linear Programming

CDM3, a linear programming system originally developed for the CONTROL DATA 1604 and 1604-A Computers, is being rewritten in FORTRAN-63.

APPENDIX IV
TABLE OF POWERS OF 2

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

APPENDIX V
OCTAL-DECIMAL CONVERSION TABLE

0000 to 0777 (Octal)	0000 to 0511 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
0000	0000	0001	0002	0003	0004	0005	0006	0007
0010	0008	0009	0010	0011	0012	0013	0014	0015
0020	0016	0017	0018	0019	0020	0021	0022	0023
0030	0024	0025	0026	0027	0028	0029	0030	0031
0040	0032	0033	0034	0035	0036	0037	0038	0039
0050	0040	0041	0042	0043	0044	0045	0046	0047
0060	0048	0049	0050	0051	0052	0053	0054	0055
0070	0056	0057	0058	0059	0060	0061	0062	0063
0100	0064	0065	0066	0067	0068	0069	0070	0071
0110	0072	0073	0074	0075	0076	0077	0078	0079
0120	0080	0081	0082	0083	0084	0085	0086	0087
0130	0088	0089	0090	0091	0092	0093	0094	0095
0140	0096	0097	0098	0099	0100	0101	0102	0103
0150	0104	0105	0106	0107	0108	0109	0110	0111
0160	0112	0113	0114	0115	0116	0117	0118	0119
0170	0120	0121	0122	0123	0124	0125	0126	0127
0200	0128	0129	0130	0131	0132	0133	0134	0135
0210	0136	0137	0138	0139	0140	0141	0142	0143
0220	0144	0145	0146	0147	0148	0149	0150	0151
0230	0152	0153	0154	0155	0156	0157	0158	0159
0240	0160	0161	0162	0163	0164	0165	0166	0167
0250	0168	0169	0170	0171	0172	0173	0174	0175
0260	0176	0177	0178	0179	0180	0181	0182	0183
0270	0184	0185	0186	0187	0188	0189	0190	0191
0300	0192	0193	0194	0195	0196	0197	0198	0199
0310	0200	0201	0202	0203	0204	0205	0206	0207
0320	0208	0209	0210	0211	0212	0213	0214	0215
0330	0216	0217	0218	0219	0220	0221	0222	0223
0340	0224	0225	0226	0227	0228	0229	0230	0231
0350	0232	0233	0234	0235	0236	0237	0238	0239
0360	0240	0241	0242	0243	0244	0245	0246	0247
0370	0248	0249	0250	0251	0252	0253	0254	0255

	0	1	2	3	4	5	6	7
0400	0256	0257	0258	0259	0260	0261	0262	0263
0410	0264	0265	0266	0267	0268	0269	0270	0271
0420	0272	0273	0274	0275	0276	0277	0278	0279
0430	0280	0281	0282	0283	0284	0285	0286	0287
0440	0288	0289	0290	0291	0292	0293	0294	0295
0450	0296	0297	0298	0299	0300	0301	0302	0303
0460	0304	0305	0306	0307	0308	0309	0310	0311
0470	0312	0313	0314	0315	0316	0317	0318	0319
0500	0320	0321	0322	0323	0324	0325	0326	0327
0510	0328	0329	0330	0331	0332	0333	0334	0335
0520	0336	0337	0338	0339	0340	0341	0342	0343
0530	0344	0345	0346	0347	0348	0349	0350	0351
0540	0352	0353	0354	0355	0356	0357	0358	0359
0550	0360	0361	0362	0363	0364	0365	0366	0367
0560	0368	0369	0370	0371	0372	0373	0374	0375
0570	0376	0377	0378	0379	0380	0381	0382	0383
0600	0384	0385	0386	0387	0388	0389	0390	0391
0610	0392	0393	0394	0395	0396	0397	0398	0399
0620	0400	0401	0402	0403	0404	0405	0406	0407
0630	0408	0409	0410	0411	0412	0413	0414	0415
0640	0416	0417	0418	0419	0420	0421	0422	0423
0650	0424	0425	0426	0427	0428	0429	0430	0431
0660	0432	0433	0434	0435	0436	0437	0438	0439
0670	0440	0441	0442	0443	0444	0445	0446	0447
0700	0448	0449	0450	0451	0452	0453	0454	0455
0710	0456	0457	0458	0459	0460	0461	0462	0463
0720	0464	0465	0466	0467	0468	0469	0470	0471
0730	0472	0473	0474	0475	0476	0477	0478	0479
0740	0480	0481	0482	0483	0484	0485	0486	0487
0750	0488	0489	0490	0491	0492	0493	0494	0495
0760	0496	0497	0498	0499	0500	0501	0502	0503
0770	0504	0505	0506	0507	0508	0509	0510	0511

1000 to 1777 (Octal)	0512 to 1023 (Decimal)

	0	1	2	3	4	5	6	7
1000	0512	0513	0514	0515	0516	0517	0518	0519
1010	0520	0521	0522	0523	0524	0525	0526	0527
1020	0528	0529	0530	0531	0532	0533	0534	0535
1030	0536	0537	0538	0539	0540	0541	0542	0543
1040	0544	0545	0546	0547	0548	0549	0550	0551
1050	0552	0553	0554	0555	0556	0557	0558	0559
1060	0560	0561	0562	0563	0564	0565	0566	0567
1070	0568	0569	0570	0571	0572	0573	0574	0575
1100	0576	0577	0578	0579	0580	0581	0582	0583
1110	0584	0585	0586	0587	0588	0589	0590	0591
1120	0592	0593	0594	0595	0596	0597	0598	0599
1130	0600	0601	0602	0603	0604	0605	0606	0607
1140	0608	0609	0610	0611	0612	0613	0614	0615
1150	0616	0617	0618	0619	0620	0621	0622	0623
1160	0624	0625	0626	0627	0628	0629	0630	0631
1170	0632	0633	0634	0635	0636	0637	0638	0639
1200	0640	0641	0642	0643	0644	0645	0646	0647
1210	0648	0649	0650	0651	0652	0653	0654	0655
1220	0656	0657	0658	0659	0660	0661	0662	0663
1230	0664	0665	0666	0667	0668	0669	0670	0671
1240	0672	0673	0674	0675	0676	0677	0678	0679
1250	0680	0681	0682	0683	0684	0685	0686	0687
1260	0688	0689	0690	0691	0692	0693	0694	0695
1270	0696	0697	0698	0699	0700	0701	0702	0703
1300	0704	0705	0706	0707	0708	0709	0710	0711
1310	0712	0713	0714	0715	0716	0717	0718	0719
1320	0720	0721	0722	0723	0724	0725	0726	0727
1330	0728	0729	0730	0731	0732	0733	0734	0735
1340	0736	0737	0738	0739	0740	0741	0742	0743
1350	0744	0745	0746	0747	0748	0749	0750	0751
1360	0752	0753	0754	0755	0756	0757	0758	0759
1370	0760	0761	0762	0763	0764	0765	0766	0767

	0	1	2	3	4	5	6	7
1400	0768	0769	0770	0771	0772	0773	0774	0775
1410	0776	0777	0778	0779	0780	0781	0782	0783
1420	0784	0785	0786	0787	0788	0789	0790	0791
1430	0792	0793	0794	0795	0796	0797	0798	0799
1440	0800	0801	0802	0803	0804	0805	0806	0807
1450	0808	0809	0810	0811	0812	0813	0814	0815
1460	0816	0817	0818	0819	0820	0821	0822	0823
1470	0824	0825	0826	0827	0828	0829	0830	0831
1500	0832	0833	0834	0835	0836	0837	0838	0839
1510	0840	0841	0842	0843	0844	0845	0846	0847
1520	0848	0849	0850	0851	0852	0853	0854	0855
1530	0856	0857	0858	0859	0860	0861	0862	0863
1540	0864	0865	0866	0867	0868	0869	0870	0871
1550	0872	0873	0874	0875	0876	0877	0878	0879
1560	0880	0881	0882	0883	0884	0885	0886	0887
1570	0888	0889	0890	0891	0892	0893	0894	0895
1600	0896	0897	0898	0899	0900	0901	0902	0903
1610	0904	0905	0906	0907	0908	0909	0910	0911
1620	0912	0913	0914	0915	0916	0917	0918	0919
1630	0920	0921	0922	0923	0924	0925	0926	0927
1640	0928	0929	0930	0931	0932	0933	0934	0935
1650	0936	0937	0938	0939	0940	0941	0942	0943
1660	0944	0945	0946	0947	0948	0949	0950	0951
1670	0952	0953	0954	0955	0956	0957	0958	0959
1700	0960	0961	0962	0963	0964	0965	0966	0967
1710	0968	0969	0970	0971	0972	0973	0974	0975
1720	0976	0977	0978	0979	0980	0981	0982	0983
1730	0984	0985	0986	0987	0988	0989	0990	0991
1740	0992	0993	0994	0995	0996	0997	0998	0999
1750	1000	1001	1002	1003	1004	1005	1006	1007
1760	1008	1009	1010	1011	1012	1013	1014	1015
1770	1016	1017	1018	1019	1020	1021	1022	1023

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7
2000	1024	1025	1026	1027	1028	1029	1030	1031
2010	1032	1033	1034	1035	1036	1037	1038	1039
2020	1040	1041	1042	1043	1044	1045	1046	1047
2030	1048	1049	1050	1051	1052	1053	1054	1055
2040	1056	1057	1058	1059	1060	1061	1062	1063
2050	1064	1065	1066	1067	1068	1069	1070	1071
2060	1072	1073	1074	1075	1076	1077	1078	1079
2070	1080	1081	1082	1083	1084	1085	1086	1087
2100	1088	1089	1090	1091	1092	1093	1094	1095
2110	1096	1097	1098	1099	1100	1101	1102	1103
2120	1104	1105	1106	1107	1108	1109	1110	1111
2130	1112	1113	1114	1115	1116	1117	1118	1119
2140	1120	1121	1122	1123	1124	1125	1126	1127
2150	1128	1129	1130	1131	1132	1133	1134	1135
2160	1136	1137	1138	1139	1140	1141	1142	1143
2170	1144	1145	1146	1147	1148	1149	1150	1151
2200	1152	1153	1154	1155	1156	1157	1158	1159
2210	1160	1161	1162	1163	1164	1165	1166	1167
2220	1168	1169	1170	1171	1172	1173	1174	1175
2230	1176	1177	1178	1179	1180	1181	1182	1183
2240	1184	1185	1186	1187	1188	1189	1190	1191
2250	1192	1193	1194	1195	1196	1197	1198	1199
2260	1200	1201	1202	1203	1204	1205	1206	1207
2270	1208	1209	1210	1211	1212	1213	1214	1215
2300	1216	1217	1218	1219	1220	1221	1222	1223
2310	1224	1225	1226	1227	1228	1229	1230	1231
2320	1232	1233	1234	1235	1236	1237	1238	1239
2330	1240	1241	1242	1243	1244	1245	1246	1247
2340	1248	1249	1250	1251	1252	1253	1254	1255
2350	1256	1257	1258	1259	1260	1261	1262	1263
2360	1264	1265	1266	1267	1268	1269	1270	1271
2370	1272	1273	1274	1275	1276	1277	1278	1279

	0	1	2	3	4	5	6	7
2400	1280	1281	1282	1283	1284	1285	1286	1287
2410	1288	1289	1290	1291	1292	1293	1294	1295
2420	1296	1297	1298	1299	1300	1301	1302	1303
2430	1304	1305	1306	1307	1308	1309	1310	1311
2440	1312	1313	1314	1315	1316	1317	1318	1319
2450	1320	1321	1322	1323	1324	1325	1326	1327
2460	1328	1329	1330	1331	1332	1333	1334	1335
2470	1336	1337	1338	1339	1340	1341	1342	1343
2500	1344	1345	1346	1347	1348	1349	1350	1351
2510	1352	1353	1354	1355	1356	1357	1358	1359
2520	1360	1361	1362	1363	1364	1365	1366	1367
2530	1368	1369	1370	1371	1372	1373	1374	1375
2540	1376	1377	1378	1379	1380	1381	1382	1383
2550	1384	1385	1386	1387	1388	1389	1390	1391
2560	1392	1393	1394	1395	1396	1397	1398	1399
2570	1400	1401	1402	1403	1404	1405	1406	1407
2600	1408	1409	1410	1411	1412	1413	1414	1415
2610	1416	1417	1418	1419	1420	1421	1422	1423
2620	1424	1425	1426	1427	1428	1429	1430	1431
2630	1432	1433	1434	1435	1436	1437	1438	1439
2640	1440	1441	1442	1443	1444	1445	1446	1447
2650	1448	1449	1450	1451	1452	1453	1454	1455
2660	1456	1457	1458	1459	1460	1461	1462	1463
2670	1464	1465	1466	1467	1468	1469	1470	1471
2700	1472	1473	1474	1475	1476	1477	1478	1479
2710	1480	1481	1482	1483	1484	1485	1486	1487
2720	1488	1489	1490	1491	1492	1493	1494	1495
2730	1496	1497	1498	1499	1500	1501	1502	1503
2740	1504	1505	1506	1507	1508	1509	1510	1511
2750	1512	1513	1514	1515	1516	1517	1518	1519
2760	1520	1521	1522	1523	1524	1525	1526	1527
2770	1528	1529	1530	1531	1532	1533	1534	1535

2000 to 2777 (Octal)	1024 to 1535 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
3000	1536	1537	1538	1539	1540	1541	1542	1543
3010	1544	1545	1546	1547	1548	1549	1550	1551
3020	1552	1553	1554	1555	1556	1557	1558	1559
3030	1560	1561	1562	1563	1564	1565	1566	1567
3040	1568	1569	1570	1571	1572	1573	1574	1575
3050	1576	1577	1578	1579	1580	1581	1582	1583
3060	1584	1585	1586	1587	1588	1589	1590	1591
3070	1592	1593	1594	1595	1596	1597	1598	1599
3100	1600	1601	1602	1603	1604	1605	1606	1607
3110	1608	1609	1610	1611	1612	1613	1614	1615
3120	1616	1617	1618	1619	1620	1621	1622	1623
3130	1624	1625	1626	1627	1628	1629	1630	1631
3140	1632	1633	1634	1635	1636	1637	1638	1639
3150	1640	1641	1642	1643	1644	1645	1646	1647
3160	1648	1649	1650	1651	1652	1653	1654	1655
3170	1656	1657	1658	1659	1660	1661	1662	1663
3200	1664	1665	1666	1667	1668	1669	1670	1671
3210	1672	1673	1674	1675	1676	1677	1678	1679
3220	1680	1681	1682	1683	1684	1685	1686	1687
3230	1688	1689	1690	1691	1692	1693	1694	1695
3240	1696	1697	1698	1699	1700	1701	1702	1703
3250	1704	1705	1706	1707	1708	1709	1710	1711
3260	1712	1713	1714	1715	1716	1717	1718	1719
3270	1720	1721	1722	1723	1724	1725	1726	1727
3300	1728	1729	1730	1731	1732	1733	1734	1735
3310	1736	1737	1738	1739	1740	1741	1742	1743
3320	1744	1745	1746	1747	1748	1749	1750	1751
3330	1752	1753	1754	1755	1756	1757	1758	1759
3340	1760	1761	1762	1763	1764	1765	1766	1767
3350	1768	1769	1770	1771	1772	1773	1774	1775
3360	1776	1777	1778	1779	1780	1781	1782	1783
3370	1784	1785	1786	1787	1788	1789	1790	1791

	0	1	2	3	4	5	6	7
3400	1792	1793	1794	1795	1796	1797	1798	1799
3410	1800	1801	1802	1803	1804	1805	1806	1807
3420	1808	1809	1810	1811	1812	1813	1814	1815
3430	1816	1817	1818	1819	1820	1821	1822	1823
3440	1824	1825	1826	1827	1828	1829	1830	1831
3450	1832	1833	1834	1835	1836	1837	1838	1839
3460	1840	1841	1842	1843	1844	1845	1846	1847
3470	1848	1849	1850	1851	1852	1853	1854	1855
3500	1856	1857	1858	1859	1860	1861	1862	1863
3510	1864	1865	1866	1867	1868	1869	1870	1871
3520	1872	1873	1874	1875	1876	1877	1878	1879
3530	1880	1881	1882	1883	1884	1885	1886	1887
3540	1888	1889	1890	1891	1892	1893	1894	1895
3550	1896	1897	1898	1899	1900	1901	1902	1903
3560	1904	1905	1906	1907	1908	1909	1910	1911
3570	1912	1913	1914	1915	1916	1917	1918	1919
3600	1920	1921	1922	1923	1924	1925	1926	1927
3610	1928	1929	1930	1931	1932	1933	1934	1935
3620	1936	1937	1938	1939	1940	1941	1942	1943
3630	1944	1945	1946	1947	1948	1949	1950	1951
3640	1952	1953	1954	1955	1956	1957	1958	1959
3650	1960	1961	1962	1963	1964	1965	1966	1967
3660	1968	1969	1970	1971	1972	1973	1974	1975
3670	1976	1977	1978	1979	1980	1981	1982	1983
3700	1984	1985	1986	1987	1988	1989	1990	1991
3710	1992	1993	1994	1995	1996	1997	1998	1999
3720	2000	2001	2002	2003	2004	2005	2006	2007
3730	2008	2009	2010	2011	2012	2013	2014	2015
3740	2016	2017	2018	2019	2020	2021	2022	2023
3750	2024	2025	2026	2027	2028	2029	2030	2031
3760	2032	2033	2034	2035	2036	2037	2038	2039
3770	2040	2041	2042	2043	2044	2045	2046	2047

3000 to 3777 (Octal)	1536 to 2047 (Decimal)

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

4000 (Octal)	2048 (Decimal)
to	to
4777 (Octal)	2559 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
4000	2048	2049	2050	2051	2052	2053	2054	2055
4010	2056	2057	2058	2059	2060	2061	2062	2063
4020	2064	2065	2066	2067	2068	2069	2070	2071
4030	2072	2073	2074	2075	2076	2077	2078	2079
4040	2080	2081	2082	2083	2084	2085	2086	2087
4050	2088	2089	2090	2091	2092	2093	2094	2095
4060	2096	2097	2098	2099	2100	2101	2102	2103
4070	2104	2105	2106	2107	2108	2109	2110	2111
4100	2112	2113	2114	2115	2116	2117	2118	2119
4110	2120	2121	2122	2123	2124	2125	2126	2127
4120	2128	2129	2130	2131	2132	2133	2134	2135
4130	2136	2137	2138	2139	2140	2141	2142	2143
4140	2144	2145	2146	2147	2148	2149	2150	2151
4150	2152	2153	2154	2155	2156	2157	2158	2159
4160	2160	2161	2162	2163	2164	2165	2166	2167
4170	2168	2169	2170	2171	2172	2173	2174	2175
4200	2176	2177	2178	2179	2180	2181	2182	2183
4210	2184	2185	2186	2187	2188	2189	2190	2191
4220	2192	2193	2194	2195	2196	2197	2198	2199
4230	2200	2201	2202	2203	2204	2205	2206	2207
4240	2208	2209	2210	2211	2212	2213	2214	2215
4250	2216	2217	2218	2219	2220	2221	2222	2223
4260	2224	2225	2226	2227	2228	2229	2230	2231
4270	2232	2233	2234	2235	2236	2237	2238	2239
4300	2240	2241	2242	2243	2244	2245	2246	2247
4310	2248	2249	2250	2251	2252	2253	2254	2255
4320	2256	2257	2258	2259	2260	2261	2262	2263
4330	2264	2265	2266	2267	2268	2269	2270	2271
4340	2272	2273	2274	2275	2276	2277	2278	2279
4350	2280	2281	2282	2283	2284	2285	2286	2287
4360	2288	2289	2290	2291	2292	2293	2294	2295
4370	2296	2297	2298	2299	2300	2301	2302	2303

	0	1	2	3	4	5	6	7
4400	2304	2305	2306	2307	2308	2309	2310	2311
4410	2312	2313	2314	2315	2316	2317	2318	2319
4420	2320	2321	2322	2323	2324	2325	2326	2327
4430	2328	2329	2330	2331	2332	2333	2334	2335
4440	2336	2337	2338	2339	2340	2341	2342	2343
4450	2344	2345	2346	2347	2348	2349	2350	2351
4460	2352	2353	2354	2355	2356	2357	2358	2359
4470	2360	2361	2362	2363	2364	2365	2366	2367
4500	2368	2369	2370	2371	2372	2373	2374	2375
4510	2376	2377	2378	2379	2380	2381	2382	2383
4520	2384	2385	2386	2387	2388	2389	2390	2391
4530	2392	2393	2394	2395	2396	2397	2398	2399
4540	2400	2401	2402	2403	2404	2405	2406	2407
4550	2408	2409	2410	2411	2412	2413	2414	2415
4560	2416	2417	2418	2419	2420	2421	2422	2423
4570	2424	2425	2426	2427	2428	2429	2430	2431
4600	2432	2433	2434	2435	2436	2437	2438	2439
4610	2440	2441	2442	2443	2444	2445	2446	2447
4620	2448	2449	2450	2451	2452	2453	2454	2455
4630	2456	2457	2458	2459	2460	2461	2462	2463
4640	2464	2465	2466	2467	2468	2469	2470	2471
4650	2472	2473	2474	2475	2476	2477	2478	2479
4660	2480	2481	2482	2483	2484	2485	2486	2487
4670	2488	2489	2490	2491	2492	2493	2494	2495
4700	2496	2497	2498	2499	2500	2501	2502	2503
4710	2504	2505	2506	2507	2508	2509	2510	2511
4720	2512	2513	2514	2515	2516	2517	2518	2519
4730	2520	2521	2522	2523	2524	2525	2526	2527
4740	2528	2529	2530	2531	2532	2533	2534	2535
4750	2536	2537	2538	2539	2540	2541	2542	2543
4760	2544	2545	2546	2547	2548	2549	2550	2551
4770	2552	2553	2554	2555	2556	2557	2558	2559

5000 (Octal)	2560 (Decimal)
to	to
5777 (Octal)	3071 (Decimal)

	0	1	2	3	4	5	6	7
5000	2560	2561	2562	2563	2564	2565	2566	2567
5010	2568	2569	2570	2571	2572	2573	2574	2575
5020	2576	2577	2578	2579	2580	2581	2582	2583
5030	2584	2585	2586	2587	2588	2589	2590	2591
5040	2592	2593	2594	2595	2596	2597	2598	2599
5050	2600	2601	2602	2603	2604	2605	2606	2607
5060	2608	2609	2610	2611	2612	2613	2614	2615
5070	2616	2617	2618	2619	2620	2621	2622	2623
5100	2624	2625	2626	2627	2628	2629	2630	2631
5110	2632	2633	2634	2635	2636	2637	2638	2639
5120	2640	2641	2642	2643	2644	2645	2646	2647
5130	2648	2649	2650	2651	2652	2653	2654	2655
5140	2656	2657	2658	2659	2660	2661	2662	2663
5150	2664	2665	2666	2667	2668	2669	2670	2671
5160	2672	2673	2674	2675	2676	2677	2678	2679
5170	2680	2681	2682	2683	2684	2685	2686	2687
5200	2688	2689	2690	2691	2692	2693	2694	2695
5210	2696	2697	2698	2699	2700	2701	2702	2703
5220	2704	2705	2706	2707	2708	2709	2710	2711
5230	2712	2713	2714	2715	2716	2717	2718	2719
5240	2720	2721	2722	2723	2724	2725	2726	2727
5250	2728	2729	2730	2731	2732	2733	2734	2735
5260	2736	2737	2738	2739	2740	2741	2742	2743
5270	2744	2745	2746	2747	2748	2749	2750	2751
5300	2752	2753	2754	2755	2756	2757	2758	2759
5310	2760	2761	2762	2763	2764	2765	2766	2767
5320	2768	2769	2770	2771	2772	2773	2774	2775
5330	2776	2777	2778	2779	2780	2781	2782	2783
5340	2784	2785	2786	2787	2788	2789	2790	2791
5350	2792	2793	2794	2795	2796	2797	2798	2799
5360	2800	2801	2802	2803	2804	2805	2806	2807
5370	2808	2809	2810	2811	2812	2813	2814	2815

	0	1	2	3	4	5	6	7
5400	2816	2817	2818	2819	2820	2821	2822	2823
5410	2824	2825	2826	2827	2828	2829	2830	2831
5420	2832	2833	2834	2835	2836	2837	2838	2839
5430	2840	2841	2842	2843	2844	2845	2846	2847
5440	2848	2849	2850	2851	2852	2853	2854	2855
5450	2856	2857	2858	2859	2860	2861	2862	2863
5460	2864	2865	2866	2867	2868	2869	2870	2871
5470	2872	2873	2874	2875	2876	2877	2878	2879
5500	2880	2881	2882	2883	2884	2885	2886	2887
5510	2888	2889	2890	2891	2892	2893	2894	2895
5520	2896	2897	2898	2899	2900	2901	2902	2903
5530	2904	2905	2906	2907	2908	2909	2910	2911
5540	2912	2913	2914	2915	2916	2917	2918	2919
5550	2920	2921	2922	2923	2924	2925	2926	2927
5560	2928	2929	2930	2931	2932	2933	2934	2935
5570	2936	2937	2938	2939	2940	2941	2942	2943
5600	2944	2945	2946	2947	2948	2949	2950	2951
5610	2952	2953	2954	2955	2956	2957	2958	2959
5620	2960	2961	2962	2963	2964	2965	2966	2967
5630	2968	2969	2970	2971	2972	2973	2974	2975
5640	2976	2977	2978	2979	2980	2981	2982	2983
5650	2984	2985	2986	2987	2988	2989	2990	2991
5660	2992	2993	2994	2995	2996	2997	2998	2999
5670	3000	3001	3002	3003	3004	3005	3006	3007
5700	3008	3009	3010	3011	3012	3013	3014	3015
5710	3016	3017	3018	3019	3020	3021	3022	3023
5720	3024	3025	3026	3027	3028	3029	3030	3031
5730	3032	3033	3034	3035	3036	3037	3038	3039
5740	3040	3041	3042	3043	3044	3045	3046	3047
5750	3048	3049	3050	3051	3052	3053	3054	3055
5760	3056	3057	3058	3059	3060	3061	3062	3063
5770	3064	3065	3066	3067	3068	3069	3070	3071

OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

	0	1	2	3	4	5	6	7
6000	3072	3073	3074	3075	3076	3077	3078	3079
6010	3080	3081	3082	3083	3084	3085	3086	3087
6020	3088	3089	3090	3091	3092	3093	3094	3095
6030	3096	3097	3098	3099	3100	3101	3102	3103
6040	3104	3105	3106	3107	3108	3109	3110	3111
6050	3112	3113	3114	3115	3116	3117	3118	3119
6060	3120	3121	3122	3123	3124	3125	3126	3127
6070	3128	3129	3130	3131	2132	3133	3134	3135
6100	3136	3137	3138	3139	3140	3141	3142	3143
6110	3144	3145	3146	3147	3148	3149	3150	3151
6120	3152	3153	3154	3155	3156	3157	3158	3159
6130	3160	3161	3162	3163	3164	3165	3166	3167
6140	3168	3169	3170	3171	3172	3173	3174	3175
6150	3176	3177	3178	3179	3180	3181	3182	3183
6160	3184	3185	3186	3187	3188	3189	3190	3191
6170	3192	3193	3194	3195	3196	3197	3198	3199
6200	3200	3201	3202	3203	3204	3205	3206	3207
6210	3208	3209	3210	3211	3212	3213	3214	3215
6220	3216	3217	3218	3219	3220	3221	3222	3223
6230	3224	3225	3226	3227	3228	3229	3230	3231
6240	3232	3233	3234	3235	3236	3237	3238	3239
6250	3240	3241	3242	3243	3244	3245	3246	3247
6260	3248	3249	3250	3251	3252	3253	3254	3255
6270	3256	3257	3258	3259	3260	3261	3262	3263
6300	3264	3265	3266	3267	3268	3269	3270	3271
6310	3272	3273	3274	3275	3276	3277	3278	3279
6320	3280	3281	3282	3283	3284	3285	3286	3287
6330	3288	3289	3290	3291	3292	3293	3294	3295
6340	3296	3297	3298	3299	3300	3301	3302	3303
6350	3304	3305	3306	3307	3308	3309	3310	3311
6360	3312	3313	3314	3315	3316	3317	3318	3319
6370	3320	3321	3322	3323	3324	3325	3326	3327

	0	1	2	3	4	5	6	7
6400	3328	3329	3330	3331	3332	3333	3334	3335
6410	3336	3337	3338	3339	3340	3341	3342	3343
6420	3344	3345	3346	3347	3348	3349	3350	3351
6430	3352	3353	3354	3355	3356	3357	3358	3359
6440	3360	3361	3362	3363	3364	3365	3366	3367
6450	3368	3369	3370	3371	3372	3373	3374	3375
6460	3376	3377	3378	3379	3380	3381	3382	3383
6470	3384	3385	3386	3387	3388	3389	3390	3391
6500	3392	3393	3394	3395	3396	3397	3398	3399
6510	3400	3401	3402	3403	3404	3405	3406	3407
6520	3408	3409	3410	3411	3412	3413	3414	3415
6530	3416	3417	3418	3419	3420	3421	3422	3423
6540	3424	3425	3426	3427	3428	3429	3430	3431
6550	3432	3433	3434	3435	3436	3437	3438	3439
6560	3440	3441	3442	3443	3444	3445	3446	3447
6570	3448	3449	3450	3451	3452	3453	3454	3455
6600	3456	3457	3458	3459	3460	3461	3462	3463
6610	3464	3465	3466	3467	3468	3469	3470	3471
6620	3472	3473	3474	3475	3476	3477	3478	3479
6630	3480	3481	3482	3483	3484	3485	3486	3487
6640	3488	3489	3490	3491	3492	3493	3494	3495
6650	3496	3497	3498	3499	3500	3501	3502	3503
6660	3504	3505	3506	3507	3508	3509	3510	3511
6670	3512	3513	3514	3515	3516	3517	3518	3519
6700	3520	3521	3522	3523	3524	3525	3526	3527
6710	3528	3529	3530	3531	3532	3533	3534	3535
6720	3536	3537	3538	3539	3540	3541	3542	3543
6730	3544	3545	3546	3547	3548	3549	3550	3551
6740	3552	3553	3554	3555	3556	3557	3558	3559
6750	3560	3561	3562	3563	3564	3565	3566	3567
6760	3568	3569	3570	3571	3572	3573	3574	3575
6770	3576	3577	3578	3579	3580	3581	3582	3583

6000 to 6777 (Octal)	3072 to 3583 (Decimal)

Octal	Decimal
10000 -	4096
20000 -	8192
30000 -	12288
40000 -	16384
50000 -	20480
60000 -	24576
70000 -	28672

	0	1	2	3	4	5	6	7
7000	3584	3585	3586	3587	3588	3589	3590	3591
7010	3592	3593	3594	3595	3596	3597	3598	3599
7020	3600	3601	3602	3603	3604	3605	3606	3607
7030	3608	3609	3610	3611	3612	3613	3614	3615
7040	3616	3617	3618	3619	3620	3621	3622	3623
7050	3624	3625	3626	3627	3628	3629	3630	3631
7060	3632	3633	3634	3635	3636	3637	3638	3639
7070	3640	3641	3642	3643	3644	3645	3646	3647
7100	3648	3649	3650	3651	3652	3653	3654	3655
7110	3656	3657	3658	3659	3660	3661	3662	3663
7120	3664	3665	3666	3667	3668	3669	3670	3671
7130	3672	3673	3674	3675	3676	3677	3678	3679
7140	3680	3681	3682	3683	3684	3685	3686	3687
7150	3688	3689	3690	3691	3692	3693	3694	3695
7160	3696	3697	3698	3699	3700	3701	3702	3703
7170	3704	3705	3706	3707	3708	3709	3710	3711
7200	3712	3713	3714	3715	3716	3717	3718	3719
7210	3720	3721	3722	3723	3724	3725	3726	3727
7220	3728	3729	3730	3731	3732	3733	3734	3735
7230	3736	3737	3738	3739	3740	3741	3742	3743
7240	3744	3745	3746	3747	3748	3749	3750	3751
7250	3752	3753	3754	3755	3756	3757	3753	3759
7260	3760	3761	3762	3763	3764	3765	3766	3767
7270	3768	3769	3770	3771	3772	3773	3774	3775
7300	3776	3777	3778	3779	3780	3781	3782	3783
7310	3784	3785	3786	3787	3788	3789	3790	3791
7320	3792	3793	3794	3795	3796	3797	3798	3799
7330	3800	3801	3802	3803	3804	3805	3806	3807
7340	3808	3809	3810	3811	3812	3813	3814	3815
7350	3816	3817	3818	3819	3820	3821	3822	3823
7360	3824	3825	3826	3827	3828	3829	3830	3831
7370	3832	3833	3834	3835	3836	3837	3838	3839

	0	1	2	3	4	5	6	7
7400	3840	3841	3842	3843	3844	3845	3846	3847
7410	3848	3849	3850	3851	3852	3853	3854	3855
7420	3856	3857	3858	3859	3860	3861	3862	3863
7430	3864	3865	3866	3867	3868	3869	3870	3871
7440	3872	3873	3874	3875	3876	3877	3878	3879
7450	3880	3881	3882	3883	3884	3885	3886	3887
7460	3888	3889	3890	3891	3892	3893	3894	3895
7470	3896	3897	3898	3899	3900	3901	3902	3903
7500	3904	3905	3906	3907	3908	3909	3910	3911
7510	3912	3913	3914	3915	3916	3917	3918	3919
7520	3920	3921	3922	3923	3924	3925	3926	3927
7530	3928	3929	3930	3931	3932	3933	3934	3935
7540	3936	3937	3938	3939	3940	3941	3942	3943
7550	3944	3945	3946	3947	3948	3949	3950	3951
7560	3952	3953	3954	3955	3956	3957	3958	3959
7570	3960	3961	3962	3963	3964	3965	3966	3967
7600	3968	3969	3970	3971	3972	3973	3974	3975
7610	3976	3977	3978	3979	3980	3981	3982	3983
7620	3984	3985	3986	3987	3988	3989	3990	3991
7630	3992	3993	3994	3995	3996	3997	3998	3999
7640	4000	4001	4002	4003	4004	4005	4006	4007
7650	4008	4009	4010	4011	4012	4013	4014	4015
7660	4016	4017	4018	4019	4020	4021	4022	4023
7670	4024	4025	4026	4027	4028	4029	4030	4031
7700	4032	4033	4034	4035	4036	4037	4038	4039
7710	4040	4041	4042	4043	4044	4045	4046	4047
7720	4048	4049	4050	4051	4052	4053	4054	4055
7730	4056	4057	4058	4059	4060	4061	4062	4063
7740	4064	4065	4066	4067	4068	4069	4070	4071
7750	4072	4073	4074	4075	4076	4077	4078	4079
7760	4080	4081	4082	4083	4084	4085	4086	4087
7770	4088	4089	4090	4091	4092	4093	4094	4095

7000 to 7777 (Octal)	3584 to 4095 (Decimal)

OCTAL-DECIMAL FRACTION CONVERSION TABLE

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000	.000000	.100	.125000	.200	.250000	.300	.375000
.001	.001953	.101	.126953	.201	.251953	.301	.376953
.002	.003906	.102	.128906	.202	.253906	.302	.378906
.003	.005859	.103	.130859	.203	.255859	.303	.380859
.004	.007812	.104	.132812	.204	.257812	.304	.382812
.005	.009765	.105	.134765	.205	.259765	.305	.384765
.006	.011718	.106	.136718	.206	.261718	.306	.386718
.007	.013671	.107	.138671	.207	.263671	.307	.388671
.010	.015625	.110	.140625	.210	.265625	.310	.390625
.011	.017578	.111	.142578	.211	.267578	.311	.392578
.012	.019531	.112	.144531	.212	.269531	.312	.394531
.013	.021484	.113	.146484	.213	.271484	.313	.396484
.014	.023437	.114	.148437	.214	.273437	.314	.398437
.015	.025390	.115	.150390	.215	.275390	.315	.400390
.016	.027343	.116	.152343	.216	.277343	.316	.402343
.017	.029296	.117	.154296	.217	.279296	.317	.404296
.020	.031250	.120	.156250	.220	.281250	.320	.406250
.021	.033203	.121	.158203	.221	.283203	.321	.408203
.022	.035156	.122	.160156	.222	.285156	.322	.410156
.023	.037109	.123	.162109	.223	.287109	.323	.412109
.024	.039062	.124	.164062	.224	.289062	.324	.414062
.025	.041015	.125	.166015	.225	.291015	.325	.416015
.026	.042968	.126	.167968	.226	.292968	.326	.417968
.027	.044921	.127	.169921	.227	.294921	.327	.419921
.030	.046875	.130	.171875	.230	.296875	.330	.421875
.031	.048828	.131	.173828	.231	.298828	.331	.423828
.032	.050781	.132	.175781	.232	.300781	.332	.425781
.033	.052734	.133	.177734	.233	.302734	.333	.427734
.034	.054687	.134	.179687	.234	.304687	.334	.429687
.035	.056640	.135	.181640	.235	.306640	.335	.431640
.036	.058593	.136	.183593	.236	.308593	.336	.433593
.037	.060546	.137	.185546	.237	.310546	.337	.435546
.040	.062500	.140	.187500	.240	.312500	.340	.437500
.041	.064453	.141	.189453	.241	.314453	.341	.439453
.042	.066406	.142	.191406	.242	.316406	.342	.441406
.043	.068359	.143	.193359	.243	.318359	.343	.443359
.044	.070312	.144	.195312	.244	.320312	.344	.445312
.045	.072265	.145	.197265	.245	.322265	.345	.447265
.046	.074218	.146	.199218	.246	.324218	.346	.449218
.047	.076171	.147	.201171	.247	.326171	.347	.451171
.050	.078125	.150	.203125	.250	.328125	.350	.453125
.051	.080078	.151	.205078	.251	.330078	.351	.455078
.052	.082031	.152	.207031	.252	.332031	.352	.457031
.053	.083984	.153	.208984	.253	.333984	.353	.458984
.054	.085937	.154	.210937	.254	.335937	.354	.460937
.055	.087890	.155	.212890	.255	.337890	.355	.462890
.056	.089843	.156	.214843	.256	.339843	.356	.464843
.057	.091796	.157	.216796	.257	.341796	.357	.466796
.060	.093750	.160	.218750	.260	.343750	.360	.468750
.061	.095703	.161	.220703	.261	.345703	.361	.470703
.062	.097656	.162	.222656	.262	.347656	.362	.472656
.063	.099609	.163	.224609	.263	.349609	.363	.474609
.064	.101562	.164	.226562	.264	.351562	.364	.476562
.065	.103515	.165	.228515	.265	.353515	.365	.478515
.066	.105468	.166	.230468	.266	.355468	.366	.480468
.067	.107421	.167	.232421	.267	.357421	.367	.482421
.070	.109375	.170	.234375	.270	.359375	.370	.484375
.071	.111328	.171	.236328	.271	.361328	.371	.486328
.072	.113281	.172	.238281	.272	.363281	.372	.488281
.073	.115234	.173	.240234	.273	.365234	.373	.490234
.074	.117187	.174	.242187	.274	.367187	.374	.492187
.075	.119140	.175	.244140	.275	.369140	.375	.494140
.076	.121093	.176	.246093	.276	.371093	.376	.496093
.077	.123046	.177	.248046	.277	.373046	.377	.498046

OCTAL-DECIMAL FRACTION CONVERSION TABLE (Cont,d)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000000	.000000	.000100	.000244	.000200	.000488	.000300	.000732
.000001	.000003	.000101	.000247	.000201	.000492	.000301	.000736
.000002	.000007	.000102	.000251	.000202	.000495	.000302	.000740
.000003	.000011	.000103	.000255	.000203	.000499	.000303	.000743
.000004	.000015	.000104	.000259	.000204	.000503	.000304	.000747
.000005	.000019	.000105	.000263	.000205	.000507	.000305	.000751
.000006	.000022	.000106	.000267	.000206	.000511	.000306	.000755
.000007	.000026	.000107	.000270	.000207	.000514	.000307	.000759
.000010	.000030	.000110	.000274	.000210	.000518	.000310	.000762
.000011	.000034	.000111	.000278	.000211	.000522	.000311	.000766
.000012	.000038	.000112	.000282	.000212	.000526	.000312	.000770
.000013	.000041	.000113	.000286	.000213	.000530	.000313	.000774
.000014	.000045	.000114	.000289	.000214	.000534	.000314	.000778
.000015	.000049	.000115	.000293	.000215	.000537	.000315	.000782
.000016	.000053	.000116	.000297	.000216	.000541	.000316	.000785
.000017	.000057	.000117	.000301	.000217	.000545	.000317	.000789
.000020	.000061	.000120	.000305	.000220	.000549	.000320	.000793
.000021	.000064	.000121	.000308	.000221	.000553	.000321	.000797
.000022	.000068	.000122	.000312	.000222	.000556	.000322	.000801
.000023	.000072	.000123	.000316	.000223	.000560	.000323	.000805
.000024	.000076	.000124	.000320	.000224	.000564	.000324	.000808
.000025	.000080	.000125	.000324	.000225	.000568	.000325	.000812
.000026	.000083	.000126	.000328	.000226	.000572	.000326	.000816
.000027	.000087	.000127	.000331	.000227	.000576	.000327	.000820
.000030	.000091	.000130	.000335	.000230	.000579	.000330	.000823
.000031	.000095	.000131	.000339	.000231	.000583	.000331	.000827
.000032	.000099	.000132	.000343	.000232	.000587	.000332	.000831
.000033	.000102	.000133	.000347	.000233	.000591	.000333	.000835
.000034	.000106	.000134	.000350	.000234	.000595	.000334	.000839
.000035	.000110	.000135	.000354	.000235	.000598	.000335	.000843
.000036	.000114	.000136	.000358	.000236	.000602	.000336	.000846
.000037	.000118	.000137	.000362	.000237	.000606	.000337	.000850
.000040	.000122	.000140	.000366	.000240	.000610	.000340	.000854
.000041	.000125	.000141	.000370	.000241	.000614	.000341	.000858
.000042	.000129	.000142	.000373	.000242	.000617	.000342	.000862
.000043	.000133	.000143	.000377	.000243	.000621	.000343	.000865
.000044	.000137	.000144	.000381	.000244	.000625	.000344	.000869
.000045	.000141	.000145	.000385	.000245	.000629	.000345	.000873
.000046	.000144	.000146	.000389	.000246	.000633	.000346	.000877
.000047	.000148	.000147	.000392	.000247	.000637	.000347	.000881
.000050	.000152	.000150	.000396	.000250	.000640	.000350	.000885
.000051	.000156	.000151	.000400	.000251	.000644	.000351	.000888
.000052	.000160	.000152	.000404	.000252	.000648	.000352	.000892
.000053	.000164	.000153	.000408	.000253	.000652	.000353	.000896
.000054	.000167	.000154	.000411	.000254	.000656	.000354	.000900
.000055	.000171	.000155	.000415	.000255	.000659	.000355	.000904
.000056	.000175	.000156	.000419	.000256	.000663	.000356	.000907
.000057	.000179	.000157	.000423	.000257	.000667	.000357	.000911
.000060	.000183	.000160	.000427	.000260	.000671	.000360	.000915
.000061	.000186	.000161	.000431	.000261	.000675	.000361	.000919
.000062	.000190	.000162	.000434	.000262	.000679	.000362	.000923
.000063	.000194	.000163	.000438	.000263	.000682	.000363	.000926
.000064	.000198	.000164	.000442	.000264	.000686	.000364	.000930
.000065	.000202	.000165	.000446	.000265	.000690	.000365	.000934
.000066	.000205	.000166	.000450	.000266	.000694	.000366	.000938
.000067	.000209	.000167	.000453	.000267	.000698	.000367	.000942
.000070	.000213	.000170	.000457	.000270	.000701	.000370	.000946
.000071	.000217	.000171	.000461	.000271	.000705	.000371	.000949
.000072	.000221	.000172	.000465	.000272	.000709	.000372	.000953
.000073	.000225	.000173	.000469	.000273	.000713	.000373	.000957
.000074	.000228	.000174	.000473	.000274	.000717	.000374	.000961
.000075	.000232	.000175	.000476	.000275	.000720	.000375	.000965
.000076	.000236	.000176	.000480	.000276	.000724	.000376	.000968
.000077	.000240	.000177	.000484	.000277	.000728	.000377	.000972

OCTAL-DECIMAL FRACTION CONVERSION TABLE (Cont'd)

OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.	OCTAL	DEC.
.000400	.000976	.000500	.001220	.000600	.001464	.000700	.001708
.000401	.000980	.000501	.001224	.000601	.001468	.000701	.001712
.000402	.000984	.000502	.001228	.000602	.001472	.000702	.001716
.000403	.000988	.000503	.001232	.000603	.001476	.000703	.001720
.000404	.000991	.000504	.001235	.000604	.001480	.000704	.001724
.000405	.000995	.000505	.001239	.000605	.001483	.000705	.001728
.000406	.000999	.000506	.001243	.000606	.001487	.000706	.001731
.000407	.001003	.000507	.001247	.000607	.001491	.000707	.001735
.000410	.001007	.000510	.001251	.000610	.001495	.000710	.001739
.000411	.001010	.000511	.001255	.000611	.001499	.000711	.001743
.000412	.001014	.000512	.001258	.000612	.001502	.000712	.001747
.000413	.001018	.000513	.001262	.000613	.001506	.000713	.001750
.000414	.001022	.000514	.001266	.000614	.001510	.000714	.001754
.000415	.001026	.000515	.001270	.000615	.001514	.000715	.001758
.000416	.001029	.000516	.001274	.000616	.001518	.000716	.001762
.000417	.001033	.000517	.001277	.000617	.001522	.000717	.001766
.000420	.001037	.000520	.001281	.000620	.001525	.000720	.001770
.000421	.001041	.000521	.001285	.000621	.001529	.000721	.001773
.000422	.001045	.000522	.001289	.000622	.001533	.000722	.001777
.000423	.001049	.000523	.001293	.000623	.001537	.000723	.001781
.000424	.001052	.000524	.001296	.000624	.001541	.000724	.001785
.000425	.001056	.000525	.001300	.000625	.001544	.000725	.001789
.000426	.001060	.000526	.001304	.000626	.001548	.000726	.001792
.000427	.001064	.000527	.001308	.000627	.001552	.000727	.001796
.000430	.001068	.000530	.001312	.000630	.001556	.000730	.001800
.000431	.001071	.000531	.001316	.000631	.001560	.000731	.001804
.000432	.001075	.000532	.001319	.000632	.001564	.000732	.001808
.000433	.001079	.000533	.001323	.000633	.001567	.000733	.001811
.000434	.001083	.000534	.001327	.000634	.001571	.000734	.001815
.000435	.001087	.000535	.001331	.000635	.001575	.000735	.001819
.000436	.001091	.000536	.001335	.000636	.001579	.000736	.001823
.000437	.001094	.000537	.001338	.000637	.001583	.000737	.001827
.000440	.00109	.000540	.001342	.000640	.001586	.000740	.001831
.000441	.001102	.000541	.001346	.000641	.001590	.000741	.001834
.000442	.001106	.000542	.001350	.000642	.001594	.000742	.001838
.000443	.001110	.000543	.001354	.000643	.001598	.000743	.001842
.000444	.001113	.000544	.001358	.000644	.001602	.000744	.001846
.000445	.001117	.000545	.001361	.000645	.001605	.000745	.001850
.000446	.001121	.000546	.001365	.000646	.001609	.000746	.001853
.000447	.001125	.000547	.001369	.000647	.001613	.000747	.001857
.000450	.001129	.000550	.001373	.000650	.001617	.000750	.001861
.000451	.001132	.000551	.001377	.000651	.001621	.000751	.001865
.000452	.001136	.000552	.001380	.000652	.001625	.000752	.001869
.000453	.001140	.000553	.001384	.000653	.001628	.000753	.001873
.000454	.001144	.000554	.001388	.000654	.001632	.000754	.001876
.000455	.001148	.000555	.001392	.000655	.001636	.000755	.001880
.000456	.001152	.000556	.001396	.000656	.001640	.000756	.001884
.000457	.001155	.000557	.001399	.000657	.001644	.000757	.001888
.000460	.001159	.000560	.001403	.000660	.001647	.000760	.001892
.000461	.001163	.000561	.001407	.000661	.001651	.000761	.001895
.000462	.001167	.000562	.001411	.000662	.001655	.000762	.001899
.000463	.001171	.000563	.001415	.000663	.001659	.000763	.001903
.000464	.001174	.000564	.001419	.000664	.001663	.000764	.001907
.000465	.001178	.000565	.001422	.000665	.001667	.000765	.001911
.000466	.001182	.000566	.001426	.000666	.001670	.000766	.001914
.000467	.001186	.000567	.001430	.000667	.001674	.000767	.001918
.000470	.001190	.000570	.001434	.000670	.001678	.000770	.001922
.000471	.001194	.000571	.001438	.000671	.001682	.000771	.001926
.000472	.001197	.000572	.001441	.000672	.001686	.000772	.001930
.000473	.001201	.000573	.001445	.000673	.001689	.000773	.001934
.000474	.001205	.000574	.001449	.000674	.001693	.000774	.001937
.000475	.001209	.000575	.001453	.000675	.001697	.000775	.001941
.000476	.001213	.000576	.001457	.000676	.001701	.000776	.001945
.000477	.001216	.000577	.001461	.000677	.001705	.000777	.001949

APPENDIX VI

INDEX TO 3604 (MNEMONIC) INSTRUCTIONS

DESIGNATORS AND ABBREVIATIONS FOR OCTAL CODES

a	Specifies storage bank	k	Unmodified shift count
c	Ext. function code	K	k + (B^b)
CPR	Channel Product Reg.	m	Address portion of instruction
CW	Control word	M	m + (B^b)
CWA	Control word address	MPR	Main Product Reg.
d	Bank usage	n	Jump address
i	Specifies storage bank; or leftmost "1" bit in a register	Ni	Next instruction
		V^v	Index Register v
j	Designator for 22, 23, 75, 76	x	Channel number

3604 MNEMONIC INSTRUCTIONS

Op. Field	Add. Field	Op.	Function	Page
ADD, MG, CM	(a) m,b,v	14	Add (A) + (M) → A	3-26
ADL, CM, RP, MG	(a) m,b,v	45	Add Logical (A) + L (Q) (M) → A	3-30
ADX	y	77.3	Add to Exp. 'y' + exp. (A) → A	3-27
AJP, ZR, NZ, PL, MI	(a) m,v	22	A Jump to 'm'	3-37
ALG	y	74.7	Perform Algorithm	5-8
ALS, SS, EO	b,k	05	A left by K	3-31
ARJ, ZR, NZ, PL, MI	(a) m,b	22	A Ret. Jump to m	3-37
ARS, SS, EO	b,k	01	A right K places	3-31
BEGR	x,(a) m,n	74.2	Begin Read	5-5
BEGW	x,(a) m,n	74.3	Begin Write	5-5
BJPL	(a) m,b,i	63.1	Bank Jump lower	3-42
BJSX	(a) m,b,i	63.0	Bank Jump and Set Index	3-40
BRTJ	(a) m,b,i	63.0	Bank Ret. Jump	3-40
CLCH	x	74.5	Clear Channel	5-8
CONN	x,e,c,n	74.0	Connect	5-3
COPY, CW, CWA	X,b	74.4	Copy Status	5-5
CPJ		77.5	Chan. Prod. Jump	4-6
DFAD, UR, UN, MG, CM, RP	(a) m,b,v	30	D.P. Float. Add (AQ) + (M,M + 1) → AQ	3-29
DFDV, UR, MG, CM	(a) m,b,v	33	D.P. Float. Div. (AQ) / (M,M + 1) → A	3-29
DFMU, UR, UN, MG, CM	(a) m,b,v	32	D.P. Float. Mul. (AQ) + (M,M + 1) → AQ	3-29
DFSB, UR, UN, MG, CM	(a) m,b,v	31	D.P. Float, Sub. (AQ) – (M,M + 1) → AQ	3-29
DLDA, MG, CM	(a) m,b,v	12	D.P. Load AQ (M,M + 1) → AQ	3-29
DRJ		77.6	D Reg. Jump	3-38
DSTA, MG, CM	(a) m,b,v	20	D.P. Store AQ in (M,M + 1)	3-29
DVF, TR, MG, CM	(a) m,b,v	27	Div. Fract. (AQ) / (M) → A	3-26
DVI, MG, CM	(a) m,b,v	25	Div. Integer (QA) / M → A	3-26
ENA, CM	y,b	10	Enter A. Y → A, extend sign Y	3-23
ENI	y,b	50	Enter Index. Y → B^b	3-23
ENQ, CM	y,b	04	Enter Q. Y → Q, extend sign Y	3-23

Op. Field	Add. Field	Op.	Function	Page
EQS	(a) m,b,v	65	Equal. Srch. B^b words; if (M-1) etc. = A, skip	3-32
EXEC	(a) m,b,v	63.7	Execute inst. at address	3-39
EXTF	x,c,n	74.1	Ext. Function	5-4
FAD, UR, UN, MG, CM, RP	(a) m,b,v	30	Float Add (A) + (M) → A	3-27
FDV, UR, MG, CM	(a) m,b,v	33	Float. Div. (A) / (M) → A	3-27
FMU, UR, UN, MG, CM	(a) m,b,v	32	Float. Mult. (A) + (M) → A	3-27
FSB, UR, UN, MB, CM, RP	(a) m,b,v	31	Float. Sub. (A) – (M) → A	3-27
IJP	m,v	55	Index Jump. ($B^b \neq 0$: (B^b – 1 → B^b, NI = m (B^b) = 0:NI	3-37
INA, CM	y,b	11	Increase A.Y + (A) → A	3-29
INF	m	77.0	Internal Function y	3-45
INI	y,b	51	Increase Index. y + (B^b) → B^b	3-29
IPA		74.6	Input to A	5-8
ISK	y,b	54	Index Skip. (B^b) ≠ y: (B^b) + 1 → B^b, skip (B^b) = y: 0 → B^b, next upper	3-29
LAC, CM	(a) m,b,v	13	Load A Comp. (M) → A	3-21
LBYT, LI, CL, RI (Ao,Ee,Qo)*	m,b,v	63.5	Load Byte	3-42
LDA, MG, CM	(a) m,b,v	12	Load A. (M) → A	3-21
LDL	(a) m,b,v	14	Load Logical. L(Q) M → A	3-30
LDQ, MG, CM	(a) m,b,v	16	Load Q. (M) → Q	3-21
LIL	(a) m,b,v	53	Load Index. (M_L) → B^b	3-22
LIU	(a) m,b,v	52	Laad Index. (M_U) → B^b	3-22
LLS, SS, EO	b,k	07	Long Left Shift (AQ) K places	3-31
LQC, CM	(a), m,b,v	17	Load Q Comp. (M^1) → Q	3-21
LRS, SS, EO	b,k	03	Long Right Shift (AQ) K places	3-31
LSTL	b,v	63.3	Locate list Element Lower	3-34
LSTU	b,v	63.3	Locate list Element Upper	3-34
MEQ	(a) m,b,v	66	Masked Equal. Srch. (B^b) words if L(Q) (M) = (A), skip	3-32
MPJ		77.4	Main Prod. Reg. Jump	4-6
MTH	(a) m,b,v	67	Masked Thresh. Srch. (B^b) words if L(Q) (M)> (A), skip	3-32
MUF, MG, CM	(a) m,b,v	26	Mult. Fract. (A) + (M) → AQ	3-26
MUI, MG, CM	(a) m,b,v	24	Mult. Integ. (A) + (M) → QA	3-26
NBJP, CM, CL, ST	p,g,m,b	63.6	Non-zero Bit Jump	3-38
NOP	m	50.0	No. op.	3-23
QJP, ZR, NZ, PL, MI	(a) m,v	23	Q Jump	3-37
QLS, SS, EO	b,k	06	Q Left Shift K places	3-31
QRJ, ZR, NZ, PL, MI	(a) m,v	23	Q Return Jump	3-37
QRS, SS, EO	b,k	02	Q Right Shift K places	3-31
RAD	(a) m,b,v	70	Replace Add. (A) + (M) → M	3-31
RAO	(a) m,b,v	72	Replace Add one. (A) + 1 → M	3-32
RGJP,s	p,y,m,b	62	Register Jump	3-39

* LBYT require modifiers Ao or Qo and Ee.

Op. Field	Add. Field	Op.	Function	Page
RJ1-3	(a) m,v	75	Sel. Ret. Jump, key 1-3	3-37
ROP,s	p,q,r	00	Register Op. Inter Reg. Trans	3-19
RSB	(a) m,b,v	71	Replace Sub. (A) – (M) → M	3-31
RSO	(a) m,b,v	73	Replace Sub. one (A) – 1 → M	3-32
RSW, CQ, CR	q,r	00	Reg. Swap, q and r	3-19
RTJ	(a) m,v	75	Return Jump	3-37
RXT, CQ, CR	q,r	00	Reg. Transmit, q to r	3-19
SAL	(a) m,b,v	61	Substitute Add. Lwr. $(A_{00-14}) \rightarrow M_{LA}$	3-23
SAU	(a) m,b,v	60	Substitute Add Upp. $(A_{00-14}) \rightarrow M_{UA}$	3-23
SBL, CM, RP	(a) m,b,v	46	Subtract Logical. L(Q) (M) – (A) → A	3-30
SBYT, LI, CL, RI (Ao,Ee,Qo)*	m,b,v	63.5	Store Byte Ee	3-42
SCA	b,k	34	Scale A left until | (A) | ≥ .5 or K = 0, K – # shifts → B^b	3-31
SCAN, EQ, GT, LT, NE, LE, GE (Qo,Ee)*	m,b,v	63.5	Scan, use byte Ee	3-42
SCL	(a) m,b,v	41	Sel. Clear, A_n to 0 for M_n = 1	3-30
SCM	(a) m,b,v	42	Sel. Comp. A_n for M_n = 1	3-30
SCQ	b,k	35	Scale AQ left until | (AQ) | ≥ .5 or K = 0, K – # shifts → B^b	3-31
SEQU, I	(a) m,n	63.4	Search for Equality (M) = (A)	3-33
SEWL, I	(a) m,n	63.4	Search in Limits (A) ≥ (M) > (Q)	3-33
SIL	(a) m,n	57	Store Index Lower (B^b) → M_{LA}	3-23
SIU	(a) m,n	56	Store Index Upper (B^b) → M_{UA}	3-23
SJ1-3	(a) m,v	75	Sel. Jump Keys 1-3	3-37
SLJ	(a) m,b,v	75	Selective Jump	3-37
SLS	(a) m,v	76	STOP	3-37
SMEQ, I	(a) m,n	63.4	Search Masked Equal. L(Q) (M) = (A)	3-33
SMWL, I	(a) m,n	63.4	Search Mag. in Limits (A) ≥ | (M) | > (Q)	3-33
SR1-3	(a) m,v	76	Sel. Return Stop Keys 1-3	3-38
SRJ	(a) m,v	76	Stop Return Jump	3-37
SS1-3	(a) m,v	76	Sel. Stop Keys 1-3	3-38
SSH	(a) m,b,v	37	Storage Shift left one if (M) neg. skip	3-32
SSK	(a) m,b,v	36	Storage Skip if (M) neg.	3-32
SST	(a) m,b,v	40	Sel. Set (A_n) to 1 for (M_n) = 1	3-30
SSU	(a) m,b,v	43	Sel. Sub. (M_n) → (A_n) for (Q_n) = 1	3-30
STA, MG, CL, CM	(a) m,b,v	20	Store A (A) → M	3-21
STL	(a) m,b,v	47	Store Logical L(Q) (A) → M	3-30
STQ, MG, CL, CM	(a) m,b,v	21	Store Q (Q) → M	3-21
SUB, MG, CM	(a) m,b,v	15	Subtract (A) – (M) → A	3-26

* SBYT and SCAN require Ao or Qo and Ee modifiers.

Op. Field	Add. Field	Op.	Function	Page
THS	(a) m,b,v	65	Threshold Search (B^b) words if (M-1) etc. (A) skip	3-32
UBJP	(a) m,b,i	63.0	Uncon. Bank Jump	3-40
XMIT, PC, CM, AUG, MK	(a) m, (i) n	63.6	Transmit (am) to (in)	3-21
ZBJP, ST, CL, CM	p,g,m,b	63.6	Zero bit Jump	3-38

MNEMONIC CODES FOR INSTRUCTION MODIFIERS

Ao	A reg. in LBYT and SBYT; Ao= rightmost bit of byte
AUG	Augment
C	Chain to next control word
CL	Clear Source
CM	Complement
CQ, CR	Clear unused part of q or r in RSW, RXT
CW	Control word to A in COPY
CWA	Control word add. to Q in COPY
Ee	Byte size in # bits for LBYT, SBYT, SCAN
Eo	Shift end off with no sign exit
I	Indir. Add. in Searches
MG	Magnitude
MI	Minus
MK	Masked
NZ	Non Zero
PC	Plus constant in A
PL	Plus
Qo	Q. reg. in LBYT, SBYT. Qo=right bit of byte
RP	Replace
SS	Signed Shift. Direction is sign count
TR	Truncated
UN	Unnormalized
UR	Unrounded
ZR	Zero

VALUES OF S

ROP Inst.

Mnem.	Octal	Function
OR	0	Or
XOR	1	Exc. OR
AND	2	And
IMP	3	Impl.
EQ	4	Equiv.
+	5	Sum
−	6	Diff.

RGJP Inst.

Mnem.	Octal	Function
EQ	0	(p)=y
GT	1	(p)>y
LT	2	(p)<y
NE	3	(p)≠ y
LE	4	(p)≤ y
GE	5	(p)≥ y
LT,D	6	(p)<y, Cond. decrem.
GE,D	7	(p)≥ y, Cond. decrem.

REGISTER CODES

(for 00, 62, 63.6 instruction)

Mnem. Code	Octal Code	Register	
	00	−−	
B1	01	B^1	
B2	02	B^2	
B3	03	B^3	
B4	04	B^4	
B5	05	B^5	
B6	06	B^6	
AL	07	A lwr. addr.	
AU	10	A up. addr.	
QL	11	Q lwr. addr.	
QU	12	Q up. addr.	
A	13	A Full	
Q	14	Q Full	
D	15	D Full	
BR	16	Bounds Reg.	
IM	17	Int. Mask Reg.	
IR	20*	Int. Reg.	
PZ	21*	All "0's"	forced operands
P1	22*	+ 1	forced operands
MZ	23*	All "1's"	forced operands
IB	24*	Inst. Bank Reg.	
OB	25	Op. Bank Reg.	
NC	26*	Shift Count Reg.	
MS	27*	Misc. Mode Sel.	
P	30*	P Register	
CK	31*	Time Register	
LM	32	Time Limit Reg.	

* Used for operands only.

APPENDIX VII
INSTRUCTION EXECUTION TIMES

The time required to execute a given instruction may vary from application to application, depending on several factors. Factors to be considered in computing execution times for a particular program are as follows:

1) If consecutive storage references are made to the same 3609 storage module, the read access time from storage will be maximized.

2) If indirect addressing is specified, at least one additional storage reference will be needed to execute the instruction (the new index designator may itself specify indirect addressing).

3) If an instruction is augmented via the Single or Double Precision Augment instructions, the execution time of the augmented instruction may be increased.

4) In repetitive operations (e.g., Augmented Transmit, Search, etc.), the number of repetitions of an operation increases execution times.

5) If the length of cables between the 3604 and storage is increased appreciably, the time required to acquire an instruction or operand from storage increases.

In computing the instruction execution times tabulated below, the following criteria were used:

1) The storage cycle time was assumed to be approximately 1.4 microseconds.

2) Instructions and operands were located in different 3609 storage modules (unless otherwise indicated in the table).

3) Times listed include the time required for instruction acquisition from storage.

4) Indirect addressing was not specified for any of the instructions.

5) The time listed for each 24-bit instruction was derived by executing a long list of that instruction with both the upper and lower instruction positions of the instruction word holding the particular instruction. After executing this list of instructions, the total elapsed time was divided by the number of instructions executed to provide an approximate time for each instruction of the list.

6) The length of the cable between the 3604 and storage was approximately 15 feet.

Instructions	Comments	Approximate Time*
Inter-Register (ROP)		
007 (Transmit)		1.33
007 (Swap)		1.82
000-006 (Arith. or Logical operation)		1.88
Full Word Transmission		
12 (LDA)[1]		2.00
16 (LDQ)[1]		2.00
20 (STA)[2]		1.88
21 (STQ)[2]		1.88
13 (LAC)[1]		2.00
17 (LQC)[1]		2.00
63.2 (XMIT)	Transmission from one 3609 storage module to another.	3.63
63.2 (Augmented Transmit) n = no. of transmits		3.25(n-1)+5.00
Address Transmission		
53 (LIL)[1]		2.00
52 (LIU)[1]		2.00
57 (SIL)		2.07
56 (SIU)		2.07
61 (SAL)		2.07
60 (SAU)		2.25
50 (ENI)		1.00
04 (ENQ) Not Augmented		1.00
10 (ENA)		1.00
Instruction Augment		
77.1 Single Precision Augment		1.13
77.2 Double Precision Augment		1.13

* Times are in microseconds

Instructions	Comments	Approximate Time
Fixed Point Arithmetic		
14 (ADD)[1]		2.07
15 (SUB)[1]		2.07
24 (MUI)	A = 0	2.12
	ESC*	6.50
	No ESC	6.40
26 (MUF)	A = 0	2.12
	ESC	6.70
	No ESC	6.63
25 (DVI)	A = 0	2.12
	ESC	14.9
	No ESC	14.9
27 (DVF)	A = 0	2.12
	ESC	14.9
	No ESC	14.9
Truncated Divide (time includes augment)	Augmented with $t^0 = 1$	5.50
Address Arithmetic		
11 (INA)		1.13
51 (INI)		1.07
54 (ISK)	Assume for a lower instruction: $B^b = Y$	1.50
	Assume for an upper instruction: $B^b \neq Y$	
Single Precision Floating Point Arithmetic		
30 (FAD)		4.25
31 (FSB)		4.25
32 (FMU)		6.40
33 (FDV)		13.0
77.3 (ADX)		1.63

* End Sign Correction

Instructions	Comments	Approximate Time
Double Precision Floating Point Arithmetic	Times listed include time for execution of Augment instruction.	
Aug. 30 (DFAD)		6.38
Aug. 31 (DFSB)		6.38
Aug. 32 (DFMU)		26.90
Aug. 33 (DFDV)		27.20
Aug. 12 (DLDA)		5.00
Aug. 20 (DSTA)		4.50
Logical		
40 (SST)[1]		2.00
41 (SCL)[1]		2.13
42 (SCM)[1]		2.07
43 (SSU)[1]		2.07
44 (LDL)[1]		2.00
45 (ADL)[1]		2.07
46 (SBL)[1]		2.07
47 (STL)		2.13
Shifting		
01 (ARS)		1.25
02 (QRS)		1.25
03 (LRS)		1.25
	* Augmented with $t^7 = 1$	2.63
05 (ALS)		1.37
06 (QLS)		1.37
07 (LLS)		1.37
	* Augmented with $t^7 = 1$	2.63
34 (SCA)		2.38
35 (SCQ)	A = 0	2.63
	A ≠ 0	2.38

* Includes time for execution of Augment instruction.

Instructions	Comments	Approximate Time
Replace		
70 (RAD)[1]		3.13
71 (RSB)[1]		3.13
72 (RAO)[1]		3.13
73 (RSO)[1]		3.13
Storage Test		
36 (SSK)[1]	Assume: operand for upper instruction is positive; operand for lower instruction is negative.	2.13
37 (SSH)[1]		3.13
Search		
64 (EQS)	b = 0	3.13
	Search satisfied	1.40 (n–1)+3.13
	Search not satisfied	$1.40\left[(B^b)-1\right]+3.25$
* 65 (THS)	same as 64	↓
* 66 (MEQ)	same as 64	
* 67 (MTH)	same as 64	
* 63.4 (s = 0) (SEQU)	Search is satisfied	2.25 (n – 1)+3.63
	Search not satisfied	$2.25\ (B^b-1)+4.25$
(s = 1) (SMEQ)	same as s = 0	↓
(s = 2) (SEWL)	same as s = 0	
(s=3) (SMWL)	same as s = 0	
63.3 (LIST)	n = number of items to scan	1.88 (n–1)+3.5 terminate w/$V^v=0$ 1.88 (n – 1)+4.0 terminate w/$B^b=0$
Jumps and Stops		
22 (AJP)	No jump condition	1.12
	b=0–3	1.75 Upper 1.25 Lower
	b=4–7	2.38 Upper 1.88 Lower
	Trace or out-of-bounds interrupt (Time to start execution at address 0 00001).	3.00
23 (QJP)	same as 22	same as 22
55 (IJP)	$(B^b=0)$	1.25
	$(B^b) \neq 0$ jump	2.25

* n = number of words searched

Instructions	Comments	Approximate Time
Jumps and Stops (Cont'd.)		
75 (SLJ)	No jump condition	1.12
	b = 0 – 3	1.75 Upper 1.25 Lower
	b = 4 – 7	2.38 Upper 1.88 Lower
76 (SLS) Assume jump condition	b = 0 – 3	1.75 Upper 1.25 Lower
	b = 4 – 7	2.38 Upper 1.88 Lower
63.7 (EXEC)		1.75
62 (RGJP)	No jump condition	2.50
	Jump condition	3.13
63.6 (NBJP) (ZBJP)	No jump condition	2.00
	Jump condition	2.63
63.0 (s=0) (UBJP)		1.75
	Trace mode or out-of-bounds interrupt (Time to start execution of address 0 00001)	3.00
63.0 (s=1) (BRTJ)		2.88
63.0 (s=2) (BJSX)		2.88
63.1 (BJPL)		1.75
77.4 (MPJ)	MPR = 0	1.25
	MPR $\neq$ 0	3.13 Upper 2.63 Lower
77.5 (CPJ)	CPR = 0	2.50
	CPR $\neq$ 0	4.35 Upper 3.85 Lower
77.6 (DRJ)	DR = 0	1.25
	DR $\neq$ 0	3.13 Upper 2.63 Lower
Variable Data Field		
63.5, $s^5=0$, $s^2=0$ (LBYT)	No index	4.25
	Right indexing	4.25
	Left indexing	5.00
63.5, $s^5=0$, $s^2=1$ (SBYT)	No index	4.88
	Right indexing	4.88
	Left indexing	5.62
63.5, $s^5=1$, (SCAN)	k = no. of words n = no. of bytes	4.38+2.63 (n–1) +1.63 (k–1)
77.0 (INF)		1.18

Instructions	Comments	Approximate Time		
		Int. Reject	Ext. Reject	Reply
Input/output				
74.0 (CONN)	y**=time for reject or resume	102.5	2.5+y	2.13+y
74.1 (EXTF)	y**=time for reject or resume	102.5	2.5+y	2.13+y
74.2 (BEGR)	y**=time for reject or resume	102.5	2.5+y	2.13+y
74.3 (BEGW)	y**=time for reject or resume	102.5	2.5+y	2.13+y
74.4 (COPY)		1.63 + 1.22 ↑ for Status + 1.22 ↑ for C.W.A. + 2.44 ↑ for C.W. Assume no Ambiguity		
74.5 (CLCH)		102.0		
74.6 (IPA)	y=time for reply	1.62+y		
74.7 (ALG)	y=time in algorithm box	1.00+y		

1 – If instruction is augmented, add 0.250 μsec.

2 – If instruction is augmented, add 0.125 μsec.

** – For internal reject, y is a fixed 100 μsec. For external reject and for reply, y is a variable time dependent upon cable length and response time of external equipment.

GLOSSARY

ABSOLUTE ADDRESS	A specific storage location; contrast with relative address.
ACCESS TIME	The time needed to perform a storage reference, either read or write. In effect, the access time of a computer is one storage reference cycle.
ACCUMULATOR	A register with provisions for the addition of another quantity to its content. It is also the name of the A register.
ADDER	A device capable of forming the sum of two or more quantities.
ADDRESS	A 15-bit quantity which identifies a particular storage location; an 18-bit quantity identifies a particular storage location within a particular storage bank.
ALPHABETIC CODING	A system of abbreviation used in preparing information for input into a computer; e.g., Q Right Shift would be QRS.
ALL ONES REGISTER	A quantity composed of all ones which may be referenced as an operand in the Inter-Register, Bit Sensing, and Register Jump instructions.
ALL ZEROS REGISTER	A quantity composed of all zeros which may be referenced as an operand in the Inter-Register, Bit Sensing, and Register Jump instructions.
AND FUNCTION	A logical function in Boolean algebra that is satisfied (has the value "1") only when all of its terms are "1's". For any other combination of values it is not satisfied and its value is "0".
A REGISTER	Principal arithmetic register; operates as a 48-bit subtractive accumulator (modulus $2^{48}-1$).
AUGMENT	Noun: The Single or Double Precision instruction. Verb: To increase the capability of an ordinary instruction by prefacing the instruction with one of the above instructions.
BASE	A quantity which defines some system of representing numbers by positional notation; radix.
BIT	Binary digit, either "1" or "0".
BLOCK	A group of words transported in and out of storage as a unit.
BOOTSTRAP	The coded instructions at the beginning of an input tape, together with the manually entered instructions.
BOUNDS REGISTER	A 37-bit register holding two 18-bit addresses and a single bit which together define an upper and lower bound in storage. Operations are typically confined to the addresses within these bounds.
BRANCH	A conditional jump.
BREAKPOINT	A point in a routine at which the computer may be stopped by manual switches for a visual check of progress.

B^1 - B^6 REGISTERS	Index registers used primarily for modification of execution address.
BUFFER	A device in which data is stored temporarily in the course of transmission from one point to another; to store data temporarily. The operation in which either a word from storage is sent to an external equipment via an output channel (output buffer), or a word is sent from an external equipment to storage via an input channel (input buffer).
BYTE	A portion of a computer word.
CAPACITY	The upper and lower limits of the numbers which may be processed in a a register or the quantity of information which may be stored in a storage unit. If the capacity of a register is exceeded, an overflow is generated.
CARRY	In an additive counter or accumulator, a signal indicating that in stage 'n', a "1" was added to a "1". The signal is sent to stage n + 1, which it complements.
CHANNEL	A transmission path that connects the communication module to an external equipment.
CHARACTER	Two types of information handled by the computer: 1) A group of 6 bits which represents a digit, letter or symbol from the typewriter. 2) A group of 12 bits which represents a column of information from the card reader.
CLEAR	A command that removes a quantity from a register by placing every stage of the register in the "0" state. The initial contents of the register are destroyed by the Clear operation.
CLOCK PHASE	One of two outputs from the master clock, even or odd.
COMMAND	A signal that performs a unit operation, such as transmitting the contents of one register to another, or setting a FF.
COMPILER	A routine which automatically produces a specific program for a particular problem. The routine determines the meaning of information expressed in a pseudo code, selects or generates the required subroutine, transforms the subroutine into specific coding, assigns storage registers, and enters the information as an element of the problem program.
COMPLEMENT	Noun: See One's Complement or Two's Complement. Verb: A command which produces the one's complement of a given quantity.
CONTENT	The quantity or word held in a register or storage location.
CORE	A ferromagnetic toroid used as the bi-stable device for storing a bit in a memory plane.
COUNTER	A register with provisions for increasing or decreasing its contents by 1.
D REGISTER	An auxiliarly (Flag) register which may be specifically referenced in the Inter-Register, Bit Sensing, or Register Jump instructions.

DOUBLE PRECISION	Providing greater precision in the results of arithmetic operations by appending to the initial operands 48 additional bits (of lesser significance) of operand.
ENTER	The Enter operation is composed of two steps: a) The register is cleared, and b) The operand, Y, is copied in to the cleared register.
EQUIVALENCE	Refer to the truth table in the Order of Instructions section.
EXCLUSIVE OR	A logical function in Boolean algebra that is satisfied (has the value "1") when any one of its terms is "1". It is not satisfied when all its terms are "1" or when all its terms are "0".
EXECUTION ADDRESS	The lower 15 bits of a 24 or 48-bit instruction. Most often used to specify the storage address of an operand. Sometimes used as the operand.
EXIT	Initiation of a second control sequence by the first, occurring when the first is near completion; the circuit involved in exiting.
FAULT	Operational difficulty which lights an indicator or for which interrupt may be selected.
FIXED POINT	A notation or system of arithmetic in which all numerical quantities are expressed by a predetermined number of digits with the binary point implicitly located at some predetermined position; contrasted with floating point.
FLIP-FLOP (FF)	A bi-stable storage device. A "1" input to the set side puts the FF in the "1" state; a "1" input to the clear side puts the FF in the "0" state. The FF remains in a state indicative of its last "1" input. A stage of a register consists of a FF.
FLOATING POINT	A means of expressing a number X by a pair of numbers, Y and Z, such that $X = Yn^Z$. Z is an integer called the exponent or characteristic; n is a base, usually 2 or 10; and Y is called the fraction or mantissa.
IMPLICATION	Refer to the truth table in the Order of Instructions section.
INCREASE	The increase operation adds a quantity (y or Y) to the contents of the specified register.
INDEX CODE	A 3-bit quantity in an instruction; usually specifies an index register whose contents are to be added to the execution address; sometimes specifies the conditions for executing the instruction.
INSTRUCTION	A 24 or 48-bit quantity consisting of an operation code and several other designators.
INSTRUCTION BANK REGISTER	A 3-bit register whose contents specify the storage bank in which instructions are located.
INTERRUPT	Leaving the main program routine to execute a special sequence of instructions.

INTERRUPT REGISTER	A 48-bit register whose individual bits are set to "1" by the occurrence of specific interrupt conditions, either internal or external.
INTERRUPT MASK REGISTER	A 48-bit register whose individual bits match those of the Interrupt register. Setting the lower 16 bits of the Interrupt Mask register to "1's" (the upper bits are always "1's") is one of the conditions for selecting interrupt.
INVERTER	A circuit which provides as an output a signal that is opposite to its input. An inverter output is "1" only if all the separate OR inputs are "0".
JUMP	An instruction which alters the normal sequence control of the computer and, conditionally or unconditionally, specifies the location of the next instruction.
LOAD	The Load operation is composed of two steps: a) The register is cleared, and b) The contents of storage location M are copied into the cleared register.
LOCATION	A storage position holding one computer word, usually designated by a specific address.
LOGICAL PRODUCT	In Boolean Algebra, the AND function of several terms. The product is "1" only when all the terms are "1"; otherwise it is "0". Sometimes referred to as the result of bit-by-bit multiplication.
LOGICAL SUM	In Boolean algebra, the OR function of several terms. The sum is "1" when any or all of the terms are "1"; it is "0" only when all are "0".
LOOP	Repetition of a group of instructions in a routine.
LOWER ADDRESS	The execution address portion of a lower instruction; bits 0 through 14 of a 48-bit register or storage location.
LOWER INSTRUCTION	See Program Step.
MASK	In the formation of the logical product of two quantities, one quantity may mask the other; i.e., determine what part of the other quantity is to be considered. If the mask is "0", that part of the other quantity is cleared; if the mask is "1", the other quantity is left unaltered.
MASTER CLOCK	The source of standard signals required for sequencing computer operation. The clock determines the basic frequency of the computer.
MASTER CLEAR	A general command produced by pressing one of two switches: a) Internal Master Clear - Clears all operational registers and control FFs in the 3604; also clears certain control FFs and registers in storage. b) External Master Clear - Clears all external equipments, the data channels, and the communication module.
MNEMONIC CODE	A three- or four-letter code which represents the function or purpose of an instruction. Also called Alphabetic Code.

MODULUS	An integer which describes certain arithmetic characteristics of registers, especially counters and accumulators, within a digital computer. The modulus of a device is defined by r^n for an open-ended device and r^n-1 for a closed (end-around) device, where 'r' is the base of the number system used and 'n' is the number of digit positions (stages) in the device. Generally, devices with modulus r^n use two's complement arithmetic; devices with modulus r^n-1 use one's complement.
NORMALIZE	To adjust the exponent and mantissa of a floating point result so that the mantissa lies in the prescribed standard (normal) range.
NORMAL JUMP	An instruction that jumps from one sequence of instructions to a second, and makes no preparation for returning to the first sequence.
NUMERIC CODING	A system of abbreviation in which all information is reduced to numerical quantities.
ONE'S COMPLEMENT	With reference to a binary number, that number which results from subtracting each bit of the given number from "1". The one's complement of a number is formed by complementing each bit of it individually, that is, changing a "1" to "0" and a "0" to a "1". A negative number is expressed by the one's complement of the corresponding positive number.
ON-LINE OPERATION	A type of system application in which the input data to the system is fed directly from the external equipment to the computer.
OPERAND	Usually refers to the quantity specified by the execution address. This quantity is operated upon in the execution of the instruction.
OPERAND BANK REGISTER	A 3-bit register whose contents specify the storage bank from which operands will be obtained.
OPERATION CODE	A 6 or 9-bit quantity in an instruction specifying the operation to be performed.
OPERATIONAL REGISTERS	Registers which are displayed on the operator's section of the console.
OR FUNCTION	A logical function in Boolean algebra that is satisfied (has the value "1") when any of its terms are "1". It is not satisfied when all terms are "0". Often called the inclusive OR function.
OVERFLOW	The capacity of a register is exceeded.
PARITY CHECK	A summation check in which the binary digits in a character are added and the sum checked against a previously computed parity digit; i.e., a check which tests whether the number of ones is odd or even.
PARTIAL ADD	An addition without carries. Accomplished by toggling each bit of the augend where the corresponding bit of the addend is a "1".
P REGISTER	The Program Address Counter (P register) is a two's complement additive register (modulus 2^{15}) which generates in sequential order the storage addresses containing the individual program steps.

PROGRAM	A precise sequence of instructions that accomplishes a computer routine; a plan for the solution of a problem.
PROGRAM STEP	Either a single 48-bit instruction or two 24-bit instructions contained in one 48-bit storage address; the higher order 24 bits are the upper instruction; lower order 24 bits, the lower instruction. An instruction or pair of instructions is read from storage, and the upper instruction is executed first. The lower one is then executed, except when the upper one provides for skipping the lower one.
Q REGISTER	Auxiliary arithmetic register which assists the A register in the more complicated arithmetic operations (modulus $2^{48}-1$).
RANDOM ACCESS	Access to storage under conditions in which the next position from which information is to be obtained is in no way dependent on the previous one.
READ	To remove a quantity from a storage location.
REJECT	A signal generated under certain circumstances by either the external equipment or the 3604 during the execution of Input/Output instructions.
RELATIVE ADDRESS	Identifies a word in a subroutine or routine with respect to its position. Relative addresses are translated into absolute addresses by the addition of some specific reference address, usually that at which the first word of the routine is stored.
REPLACE	When used in the title of an instruction, the result of the execution of the instruction is stored in the location from which the initial operand was obtained. When replace is used in the description of an instruction, the contents of a location or register are substituted by the operand. The replace operation implies clearing the register or portion of the register in preparation for the new quantity.
REPLY	A response signal in I/O operations that indicates a positive response to some previous operation or request signal.
RESUME	A control signal sent by the 3609 to either the 3604 or 3606 indicating a Read or Write operation in storage is completed.
RETURN JUMP	An instruction that jumps from a sequence of instructions to initiate a second sequence and prepares for continuing the first sequence after the second is completed.
ROUTINE	The sequence of operations which the computer performs under the direction of a program.
SCALE FACTOR	One or more coefficients by which quantities are multiplied or divided so that they lie in a given range of magnitude.
SHIFT	To move the bits of a quantity right or left.
SIGN BIT	In registers where a quantity is treated as signed by use of one's complement notation, the bit in the highest order stage of the register. If the bit is "1", the quantity is negative; if the bit is "0", the quantity is positive.

SIGN EXTENSION	The duplication of the sign bit in the higher order stages of a register.
STAGE	The FFs and inverters associated with a bit position of a register.
STATUS	The state or condition of input/output operations.
STORE	To transmit information to a device from which the unaltered information can later be obtained. The Store operation is essentially the reverse of the Load operation. Storage location M is cleared and the contents of the register are copied into M.
SUBINSTRUCTION	In a typical 24-bit instruction, the Index code specifies one of eight forms of the instruction indicated by the Operation code. Such forms are called subinstructions.
SUBOPERATION CODE	In an instruction having several options, the Suboperation code specifies which of these options is to be performed.
TOGGLE	To complement each bit of a quantity as a result of an individual condition.
TRANSMISSION, FORCED	A transfer of bits into a register which has not been cleared previously.
TRANSMIT	The term transmit implies register contents are moved; i.e., the contents of register 1 are copied into register 2. Unless specifically stated, the contents are not changed during transmission. The term transfer is often used synonymously with transmit.
TWO'S COMPLEMENT	Number that results from subtracting each bit of a number from "0". The two's complement may be formed by complementing each bit of the given number and then adding one to the result, performing the required carries.
U REGISTER	Program Control register. Holds a program step while the single 48-bit instruction or the two 24-bit instructions contained in it are executed.
UNDERFLOW	That fault occurring in exponent arithmetic when the value of the exponent formed in a floating point sum, difference, product, or quotient of two numbers is $< 2^{-10-1}$ (-1777_8).
UPPER ADDRESS	The execution address portion of an upper instruction; bit positions 24 through 38 of a 48-bit register or storage address.
UPPER INSTRUCTION	See Program Step.
WORD	A unit of information which has been coded for use in the computer as a series of bits. The normal word length is 48 bits.
WRITE	To enter a quantity into a storage location.

CUT ALONG LINE

COMMENT SHEET

CONTROL DATA 3600

REFERENCE MANUAL – PUB. NO. 60021300

FROM NAME : _____

BUSINESS ADDRESS : _____

COMMENTS: (DESCRIBE ERRORS, SUGGESTED ADDITION OR DELETION AND INCLUDE PAGE NUMBER, ETC.)

NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE STAPLE

FOLD FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
8100 34TH AVENUE SOUTH
MINNEAPOLIS 20, MINNESOTA

ATTN: TECHNICAL PUBLICATIONS DEPT.
COMPUTER DIVISION
PLANT TWO

CUT ALONG LINE

FOLD FOLD

STAPLE STAPLE

INDEX TO INSTRUCTIONS
(OCTAL CODES)

* Instructions not in the 1604 computer.

CONTROL DATA SALES OFFICES

ALAMOGORDO • ALBUQUERQUE • ATLANTA • BOSTON • CAPE CANAVERAL
CHICAGO • CINCINNATI • CLEVELAND • COLORADO SPRINGS • DALLAS • DAYTON
DENVER • DETROIT • DOWNEY, CALIFORNIA • HONOLULU • HOUSTON • HUNTSVILLE
ITHACA • KANSAS CITY, KANSAS • LOS ANGELES • MADISON, WISCONSIN
MINNEAPOLIS • NEWARK • NEW ORLEANS • NEW YORK CITY • OAKLAND • OMAHA
PALO ALTO • PHILADELPHIA • PHOENIX • PITTSBURGH • SACRAMENTO
SALT LAKE CITY • SAN BERNARDINO • SAN DIEGO • SEATTLE • WASHINGTON, D.C.

INTERNATIONAL OFFICES

FRANKFURT, GERMANY • HAMBURG, GERMANY • STUTTGART, GERMANY
GENEVA, SWITZERLAND • ZURICH, SWITZERLAND • CANBERRA, AUSTRALIA
MELBOURNE, AUSTRALIA • SYDNEY, AUSTRALIA • ATHENS, GREECE
LONDON, ENGLAND • OSLO, NORWAY • PARIS, FRANCE • STOCKHOLM, SWEDEN
MEXICO CITY, MEXICO, (REGAL ELECTRONICA DE MEXICO, S.A.)
OTTAWA, CANADA, (COMPUTING DEVICES OF CANADA, LIMITED) • TOKYO, JAPAN,
(C. ITOH ELECTRONIC COMPUTING SERVICE CO., LTD.)

CONTROL DATA
CORPORATION

8100 34th AVENUE SOUTH, MINNEAPOLIS, MINNESOTA 55440

Pub. No. 60021300

Litho in

PUBLICATION NUMBER

60021300

REVISION E

TITLE 3600 Reference Manual

REASON FOR CHANGE

Change Order 8911

INSTRUCTIONS

Replace the following pages with the attached revised pages:

Record of Revisions
28
29

ORIGINATED BY P.A. Kraska APPROVED _____ DATE Sept. 16, 1964

FORM CA 114

3600

Control Data® 3600 Computer System
Reference Manual

Record of Revisions

REVISION	NOTES
D	Manual revised; obsoletes all previous editions.
E	Change Order 8911; pages 28, and 29 revised.

Pub. No. 60021300
September, 1964

Address comments concerning this manual to:

Control Data Corporation
Technical Publications Department
4201 North Lexington Avenue
St. Paul, Minnesota 55112

Or use comment sheet located in the rear of this manual.

APPENDIX VI

INDEX TO 3604 (MNEMONIC) INSTRUCTIONS

DESIGNATORS AND ABBREVIATIONS FOR OCTAL CODES

a	Specifies storage bank	k	Unmodified shift count
c	Ext. function code	K	k + (B^b)
CPR	Channel Product Reg.	m	Address portion of instruction
CW	Control word	M	m + (B^b)
CWA	Control word address	MPR	Main Product Reg.
d	Bank usage	n	Jump address
i	Specifies storage bank; or leftmost "1" bit in a register	NI	Next Instruction
		V^v	Index Register v
j	Designator for 22, 23, 75, 76	x	Channel number

3604 MNEMONIC INSTRUCTIONS

Op. Field	Add. Field	Op.	Function	Page
ADD, MG, CM	(a) m,b,v	14	Add (A) + (M) → A	3-26
ADL, CM, RP, MG	(a) m,b,v	45	Add Logical (A) + L (Q) (M) → A	3-30
ADX	y	77.3	Add to Exp. 'y' + exp. (A) → A	3-27
AJP, ZR, NZ, PL, MI	(a) m,v	22	A Jump to 'm'	3-37
ALG	y	74.7	Perform Algorithm	5-8
ALS, SS, EO	b,k	05	A left by K	3-31
ARJ, ZR, NZ, PL, MI	(a) m,b	22	A Ret. Jump to m	3-37
ARS, SS, EO	b,k	01	A right K places	3-31
BEGR	x,(a) m,n	74.2	Begin Read	5-5
BEGW	x,(a) m,n	74.3	Begin Write	5-5
BJPL	(a) m,b,i	63.1	Bank Jump lower	3-42
BJSX	(a) m,b,i	63.0	Bank Jump and Set Index	3-40
BRTJ	(a) m,b,i	63.0	Bank Ret. Jump	3-40
CLCH	x	74.5	Clear Channel	5-8
CONN	x,e,c,n	74.0	Connect	5-3
COPY, CW, CWA	X,b	74.4	Copy Status	5-5
CPJ		77.5	Chan. Prod. Jump	4-6
DFAD, UR, UN, MG, CM, RP	(a) m,b,v	30	D.P. Float. Add (AQ) + (M,M + 1) → AQ	3-29
DFDV, UR, MG, CM	(a) m,b,v	33	D.P. Float. Div. (AQ) / (M,M + 1) → A	3-29
DFMU, UR, UN, MG, CM	(a) m,b,v	32	D.P. Float. Mul. (AQ) + (M,M + 1) → AQ	3-29
DFSB, UR, UN, MG, CM	(a) m,b,v	31	D.P. Float, Sub. (AQ) – (M,M + 1) → AQ	3-29
DLDA, MG, CM	(a) m,b,v	12	D.P. Load AQ (M,M + 1) → AQ	3-29
DRJ		77.6	D Reg. Jump	3-38
DSTA, MG, CM	(a) m,b,v	20	D.P. Store AQ in (M,M + 1)	3-29
DVF, TR, MG, CM	(a) m,b,v	27	Div. Fract. (AQ) / (M) → A	3-26
DVI, MG, CM	(a) m,b,v	25	Div. Integer (QA) / M → A	3-26
ENA, CM	y,b	10	Enter A. Y → A, extend sign Y	3-23
ENI	y,b	50	Enter Index. Y → B^b	3-23
ENQ, CM	y,b	04	Enter Q. Y → Q, extend sign Y	3-23

Op. Field	Add. Field	Op.	Function	Page
EQS	(a) m,b,v	65	Equal. Srch. B^b words; if (M-1) etc. = A, skip	3-32
EXEC	(a) m,b,v	63.7	Execute inst. at address	3-39
EXTF	x,c,n	74.1	Ext. Function	5-4
FAD, UR, UN, MG, CM, RP	(a) m,b,v	30	Float Add (A) + (M) → A	3-27
FDV, UR, MG, CM	(a) m,b,v	33	Float. Div. (A) / (M) → A	3-27
FMU, UR, UN, MG, CM	(a) m,b,v	32	Float. Mult. (A) + (M) → A	3-27
FSB, UR, UN, MB, CM, RP	(a) m,b,v	31	Float. Sub. (A) – (M) → A	3-27
IJP	m,v	55	Index Jump. ($B^b \neq 0$: (B^b – 1 → B^b, NI = m (B^b) = 0:NI	3-37
INA, CM	y,b	11	Increase A.Y + (A) → A	3-29
INF	m	77.0	Internal Function y	3-45
INI	y,b	51	Increase Index. y + (B^b) → B^b	3-29
IPA		74.6	Input to A	5-8
ISK	y,b	54	Index Skip. (B^b) ≠ y: (B^b) + 1 → B^b, skip (B^b) = y: 0 → B^b, next upper	3-29
LAC, CM	(a) m,b,v	13	Load A Comp. (M)′→ A	3-21
LBYT, LI, CL, RI (Ao,Ee,Qo)*	m,b,v	63.5	Load Byte	3-42
LDA, MG, CM	(a) m,b,v	12	Load A. (M) → A	3-21
LDL	(a) m,b,v	14	Load Logical. L(Q) M → A	3-30
LDQ, MG, CM	(a) m,b,v	16	Load Q. (M) → Q	3-21
LIL	(a) m,b,v	53	Load Index. (M_L) → B^b	3-22
LIU	(a) m,b,v	52	Laad Index. (M_U) → B^b	3-22
LLS, SS, EO	b,k	07	Long Left Shift (AQ) K places	3-31
LQC, CM	(a), m,b,v	17	Load Q Comp. (M)′→ Q	3-21
LRS, SS, EO	b,k	03	Long Right Shift (AQ) K places	3-31
LSTL	b,v	63.3	Locate list Element Lower	3-34
LSTU	b,v	63.3	Locate list Element Upper	3-34
MEQ	(a) m,b,v	66	Masked Equal. Srch. (B^b) words if L(Q) (M) = (A), skip	3-32
MPJ		77.4	Main Prod. Reg. Jump	4-6
MTH	(a) m,b,v	67	Masked Thresh. Srch. (B^b) words if L(Q) (M)> (A), skip	3-32
MUF, MG, CM	(a) m,b,v	26	Mult. Fract. (A) + (M) → AQ	3-26
MUI, MG, CM	(a) m,b,v	24	Mult. Integ. (A) + (M) → QA	3-26
NBJP, CM, CL, ST	p,g,m,b	63.6	Non-zero Bit Jump	3-38
NOP	m	50.0	No. op.	3-23
QJP, ZR, NZ, PL, MI	(a) m,v	23	Q Jump	3-37
QLS, SS, EO	b,k	06	Q Left Shift K places	3-31
QRJ, ZR, NZ, PL, MI	(a) m,v	23	Q Return Jump	3-37
QRS, SS, EO	b,k	02	Q Right Shift K places	3-31
RAD	(a) m,b,v	70	Replace Add. (A) + (M) →M & A	3-31
RAO	(a) m,b,v	72	Replace Add one.(M) + 1 →M & A	3-32
RGJP,s	p,y,m,b	62	Register Jump	3-39

* LBYT require modifiers Ao or Qo and Ee.

Op. Field	Add. Field	Op.	Function	Page
RJ1-3	(a) m,v	75	Sel. Ret. Jump, key 1-3	3-37
ROP,s	p,q,r	00	Register Op. Inter Reg. Trans	3-19
RSB	(a) m,b,v	71	Replace Sub. (M) – (A) → M & A	3-31
RSO	(a) m,b,v	73	Replace Sub. one (M) – 1 → M & A	3-32
RSW, CQ, CR	q,r	00	Reg. Swap, q and r	3-19
RTJ	(a) m,v	75	Return Jump	3-37
RXT, CQ, CR	q,r	00	Reg. Transmit, q to r	3-19
SAL	(a) m,b,v	61	Substitute Add. Lwr. $(A_{00-14}) \rightarrow M_{LA}$	3-23
SAU	(a) m,b,v	60	Substitute Add Upp. $(A_{00-14}) \rightarrow M_{UA}$	3-23
SBL, CM, RP	(a) m,b,v	46	Subtract Logical. L(Q) (M) – (A) → A	3-30
SBYT, LI, CL, RI (Ao,Ee,Qo)*	m,b,v	63.5	Store Byte Ee	3-42
SCA	b,k	34	Scale A left until | (A) | ≥ .5 or K = 0, K – # shifts → B^b	3-31
SCAN, EQ, GT, LT, NE, LE, GE (Qo,Ee)*	m,b,v	63.5	Scan, use byte Ee	3-42
SCL	(a) m,b,v	41	Sel. Clear, A_n to 0 for $M_n = 1$	3-30
SCM	(a) m,b,v	42	Sel. Comp. A_n for $M_n = 1$	3-30
SCQ	b,k	35	Scale AQ left until | (AQ) | ≥ .5 or K = 0, K – # shifts → B^b	3-31
SEQU, I	(a) m,n	63.4	Search for Equality (M) = (A)	3-33
SEWL, I	(a) m,n	63.4	Search in Limits (A) ≥ (M) > (Q)	3-33
SIL	(a) m,n	57	Store Index Lower $(B^b) \rightarrow M_{LA}$	3-23
SIU	(a) m,n	56	Store Index Upper $(B^b) \rightarrow M_{UA}$	3-23
SJ1-3	(a) m,v	75	Sel. Jump Keys 1-3	3-37
SLJ	(a) m,b,v	76	Selective Jump	3-37
SLS	(a) m,v	76	STOP	3-37
SMEQ, I	(a) m,n	63.4	Search Masked Equal. L(Q) (M) = (A)	3-33
SMWL, I	(a) m,n	63.4	Search Mag. in Limits (A) ≥ | (M) | > (Q)	3-33
SR1-3	(a) m,v	76	Sel. Return Stop Keys 1-3	3-38
SRJ	(a) m,v	76	Stop Return Jump	3-37
SS1-3	(a) m,v	76	Sel. Stop Keys 1-3	3-38
SSH	(a) m,b,v	37	Storage Shift left one if (M) neg. skip	3-32
SSK	(a) m,b,v	36	Storage Skip if (M) neg.	3-32
SST	(a) m,b,v	40	Sel. Set (A_n) to 1 for $(M_n) = 1$	3-30
SSU	(a) m,b,v	43	Sel. Sub. $(M_n) \rightarrow (A_n)$ for $(Q_n) = 1$	3-30
STA, MG, CL, CM	(a) m,b,v	20	Store A (A) → M	3-21
STL	(a) m,b,v	47	Store Logical L(Q) (A) → M	3-30
STQ, MG, CL, CM	(a) m,b,v	21	Store Q (Q) → M	3-21
SUB, MG, CM	(a) m,b,v	15	Subtract (A) – (M) → A	3-26

* SBYT and SCAN require Ao or Qo and Ee modifiers.

Op. Field	Add. Field	Op.	Function	Page
THS	(a) m,b,v	65	Threshold Search (B^b) words if (M-1) etc. (A) skip	3-32
UBJP	(a) m,b,i	63.0	Uncon. Bank Jump	3-40
XMIT, PC, CM, AUG, MK	(a) m, (i) n	63.6	Transmit (am) to (in)	3-21
ZBJP, ST, CL, CM	p,g,m,b	63.6	Zero bit Jump	3-38

MNEMONIC CODES FOR INSTRUCTION MODIFIERS

Ao	A reg. in LBYT and SBYT; Ao= rightmost bit of byte
AUG	Augment
C	Chain to next control word
CL	Clear Source
CM	Complement
CQ, CR	Clear unused part of q or r in RSW, RXT
CW	Control word to A in COPY
CWA	Control word add. to Q in COPY
Ee	Byte size in # bits for LBYT, SBYT, SCAN
Eo	Shift end off with no sign exit
I	Indir. Add. in Searches
MG	Magnitude
MI	Minus
MK	Masked
NZ	Non Zero
PC	Plus constant in A
PL	Plus
Qo	Q. reg. in LBYT, SBYT. Qo=right bit of byte
RP	Replace
SS	Signed Shift. Direction is sign count
TR	Truncated
UN	Unnormalized
UR	Unrounded
ZR	Zero

VALUES OF S

ROP Inst.

Mnem.	Octal	Function
OR	0	Or
XOR	1	Exc. OR
AND	2	And
IMP	3	Impl.
EQ	4	Equiv.
+	5	Sum
−	6	Diff.

RGJP Inst.

Mnem.	Octal	Function
EQ	0	(p)=y
GT	1	(p)>y
LT	2	(p)<y
NE	3	(p)≠ y
LE	4	(p)≤ y
GE	5	(p)≥ y
LT,D	6	(p)<y, Cond. decrem.
GE,D	7	(p)≥ y, Cond. decrem.

REGISTER CODES

(for 00, 62, 63.6 instruction)

Mnem. Code	Octal Code	Register
	00	--
B1	01	B^1
B2	02	B^2
B3	03	B^3
B4	04	B^4
B5	05	B^5
B6	06	B^6
AL	07	A lwr. addr.
AU	10	A up. addr.
QL	11	Q lwr. addr.
QU	12	Q up. addr.
A	13	A Full
Q	14	Q Full
D	15	D Full
BR	16	Bounds Reg.
IM	17	Int. Mask Reg.
IR	20*	Int. Reg.
PZ	21*	All "0's" } forced
P1	22*	+ 1 } operands
MZ	23*	All "1's" }
IB	24*	Inst. Bank Reg.
OB	25	Op. Bank Reg.
NC	26*	Shift Count Reg.
MS	27*	Misc. Mode Sel.
P	30*	P Register
CK	31*	Time Register
LM	32	Time Limit Reg.

* Used for operands only.