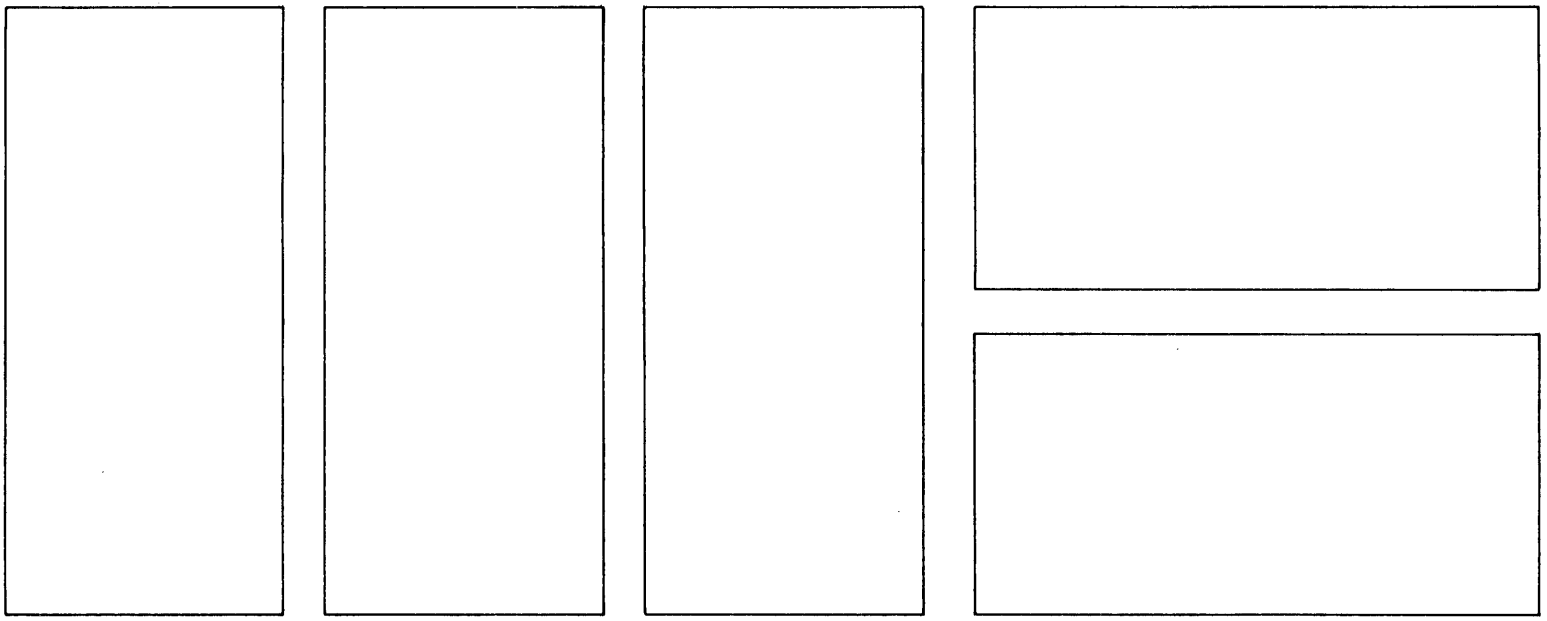
# CONTROL DATA® 1700 Computer System



## COMPUTER REFERENCE MANUAL

| 60153100 | **Record of Revisions** |
|---|---|
| REVISION | NOTES |
| Rev. 02 | This edition obsoletes all previous editions. |
| (9-17-65) | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |
| | |

Pub. No. 60153100
September, 1965



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 back of this manual.

PRELIMINARY EDITION

Information in this manual is subject to correction and change. Comments and suggestions should be communicated to Computer Division, Technical Publications Department.

# PREFACE

The 1700 Computer System Reference Manual, Pub. No. 60153100, describes the characteristics, operation, and capabilities of the computer. The instructions are described in terms of their use and parameters. For more information, refer to the following Control Data publications.

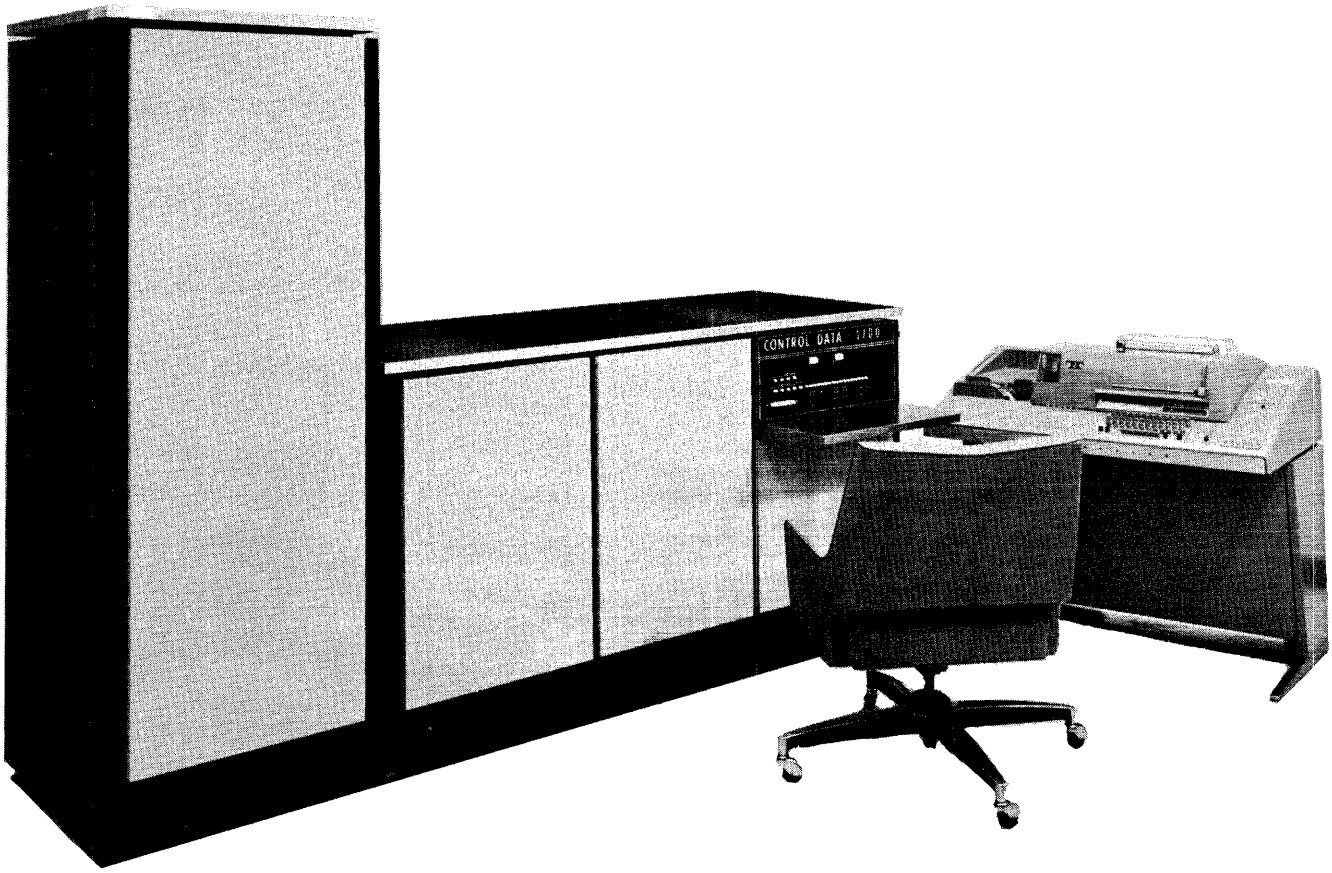
1700 Computer System, System Manual, Pub. No. 60153100
1700 FORTRAN General Information Manual, Pub. No. 60134000
1700 UTILITY/ASSEMBLER General Information Manual, Pub. No. 60133900

# CONTENTS

## FIGURES



## TABLES


CONTROL DATA 1700

# BASIC SYSTEM DESCRIPTION 1

## INTRODUCTION

The CONTROL DATA* 1704 Computer is a stored program, small, real-time, digital computer. It is designed for on-line or control applications requiring special program protection features as well as high reliability and speed under a wide range of environmental conditions.

The 1704 Computer is designed to interface with a combination of slow-speed and high-speed I/O (input/output) devices. With the available options, a variety of system configurations is possible. Refer to Table 1-1 for system characteristics.

TABLE 1-1. 1700 COMPUTER SYSTEM CHARACTERISTICS

| | |
|---|---|
| Stored-program, digital computer<br>Completely solid-state, 6000-type logic<br>Parallel mode of operation<br>18-bit storage word<br>16 data bits<br>1 parity bit<br>1 program protect bit<br>16-bit instruction word<br>Two 16-bit Index registers<br>Multilevel indirect addressing<br>Magnetic core storage (options available):<br>4,096 18-bit words, expandable to 32,768 words<br>Input/output (options available):<br>Transmission of 8/16-bit words<br>Console includes:<br>Register contents displayed in binary<br>Operating switches and indicators | Reliability (calculated):<br>Approximately 8,000 hours mean time between failures for the 1704 Computer<br>Environment:<br>40°F to 120°F<br>Relative humidity 0% to 80%<br>Cooling:<br>Forced air<br>System interrupt<br>Flexible repertoire of instructions:<br>Arithmetic operations<br>Logical and masking operations<br>Interregister transfers<br>Base 16 (hexadecimal) number system<br>Binary arithmetic:<br>Modulus $2^{16}-1$ (one's complement)<br>Intercomputer communications:<br>1700 to 1700<br>Satellite operations |

*Registered trademark of Control Data Corporation

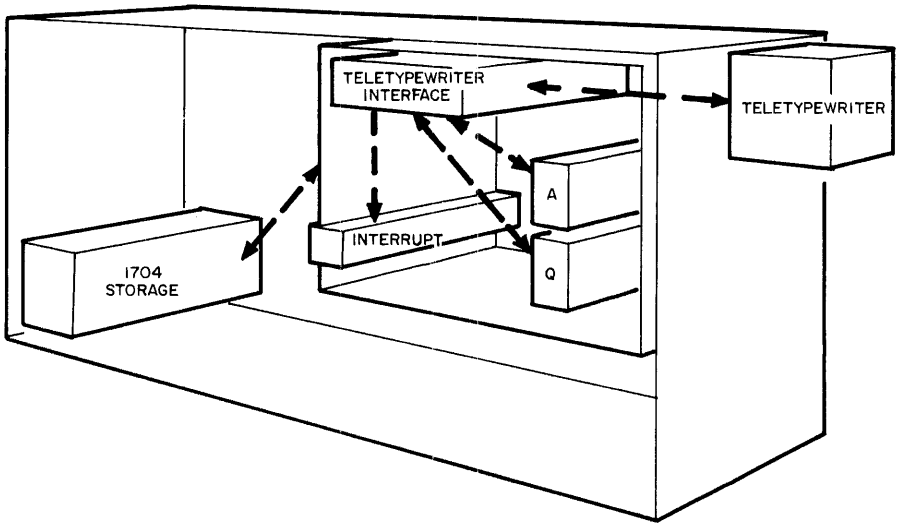



Figure 1-1. Basic 1700 Computer System

## BASIC 1700 COMPUTER SYSTEM

The basic 1700 Computer System (see Figure 1-1) consists of the 1704 Computer and the 1720 Card/Paper Tape Data Channel. The data channel is mounted in the same cabinet as the computer.

### Computer

The 1704 Computer (also referred to as "1704" or "computer") performs Arithmetic and Logical operations required by the instructions of a stored program. The computer also generates the commands necessary to execute input and output operations via the A/Q interface.

#### Computation Section

This section executes the stored program contained in the storage section of the computer. The basic computer contains one internal and one external interrupt.

#### Storage

The basic 1700 Computer System provides high-speed, random-access magnetic core storage for 4,096 18-bit words (16 data bits, 1 parity bit, 1 program protect bit).

#### A/Q Interface

The A/Q interface is the path between the computer and external equipment during I/O operations. Data is transferred to and from I/O equipment in 16-bit words.

Teletypewriter Interface

The teletypewriter interface is the path between the computer and external teletypewriter equipment. Data is transferred to and from teletypewriter equipment in 8-bit words.

Storage Interface

The storage interface provides the internal link between the computation section and storage. It also provides a direct link between optional external devices and storage.

Console

The console contains all the controls and indicators necessary to operate the 1700 Computer System.

## Peripheral Controller

Space for a 1720 Card/Paper Tape Data Channel is included in a basic 1700 computer cabinet. It may be installed in an expanded system (see Figure 1-2). It is a low-speed data channel and controller which connects internally to the A/Q interface. The 1720 Card/Paper Tape Data Channel can control the 1721 or 1722 Paper Tape Input, the 1723 or 1724 Paper Tape Output, and the 1729 Card Reader (low speed).

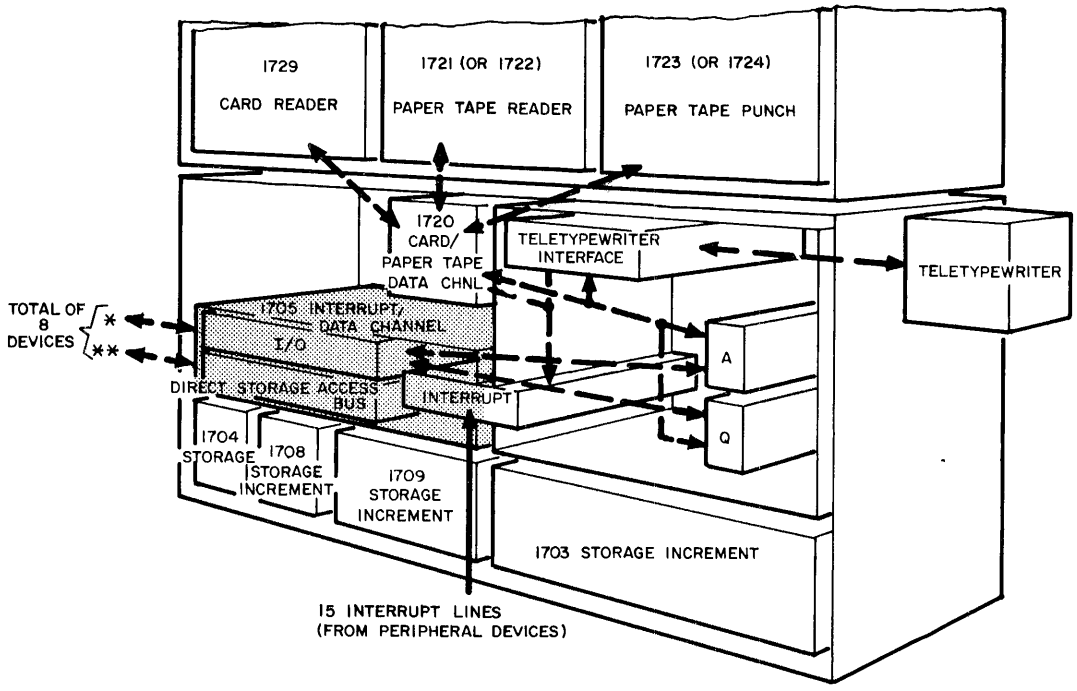



Figure 1-2. Expanded 1700 Computer System

## OPTIONS

For greater systems capability, the 1700 Computer System may be expanded with storage, I/O, and computer-to-computer options. Figure 1-2 shows an expanded 1700 Computer System.

### Storage Options

1703

A 16,384-word storage increment may be added to a 1704 Computer to which 1708 and 1709 storage increments have been added. This gives the computer a total storage capacity of 32,768 words.

1708

A 4,096-word storage increment which may be added to a 1704 Computer. This gives the 1704 a total storage capacity of 8,192 words.

1709

A 8,192-word storage increment which may be added to a 1704 Computer to which a 1708 storage increment has been added. This gives the 1704 a total storage capacity of 16,384 words.

Note that normal storage sizes are 4K, 8K, 16K, and 32K. The computer is designed to permit installation of 12K and 24K storage as a special option. The special option is not supported by standard software packages.


1659

CONTROL DATA 1700 Computer applications range throughout the spectrum of industrial and commercial uses. Versatility is expressed through expandable modular hardware and versatile software packages encompassing real-time and time-shared program applications.

## I/O Options

1705

This Interrupt/Data Channel option increases the I/O capability of the 1704 Computer by adding 14 interrupt levels, giving the computer a total of 16. This further implements the nonbuffered AQ I/O channel for use external to the computer, and implements the direct-access channel into storage from external sources.

1706

A Buffered Data Channel which has an external interface identical to the basic AQ channel. The internal interface of the BDC connects to both the AQ channel and the direct-access bus to storage.

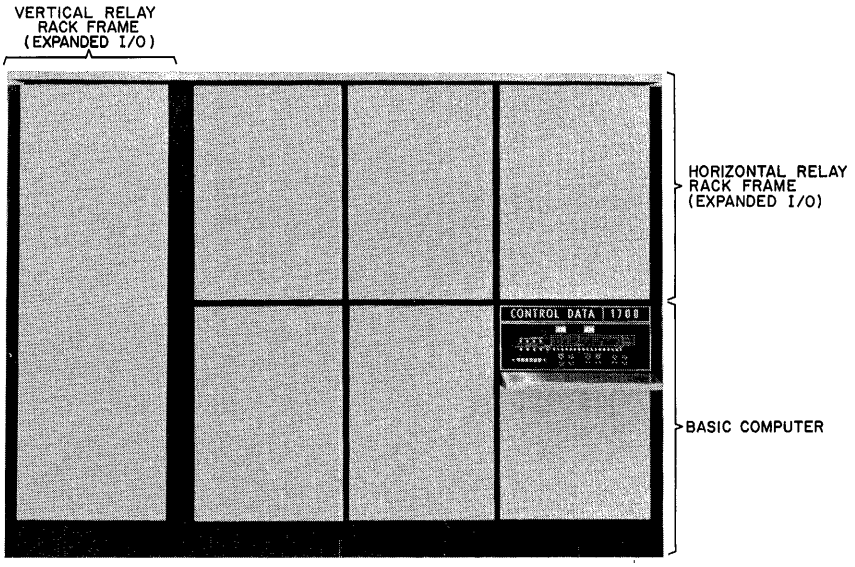
1716

A Coupling Data Channel which connects to the standard AQ interfaces of two 1704 Computers and to the direct storage access points of the computers. It provides for a common set of peripheral devices for both computers.

## Intercomputer Communication

1718

This Satellite Coupler provides coupling between the 1704 Computer and 3000 or 6000 Series computers. It connects to a 3X06 or 6681 Data Channel and operates like a 3682 Satellite Coupler. Also, the 1718 can be connected to a 1705, 1706, or 1716 Data Channel. Programs are provided to facilitate generation of user programs. Transfer is in 12-bit bytes.




1553

Modular design of the 1700 Computer allows the user wide choice of options. Paper tape punch and reader, and card reader may be incorporated into the horizontal relay rack frame while the user may select a typical vertical relay rack frame containing a 1706 Buffered Data Channel and other associated equipment.

# STORAGE 2

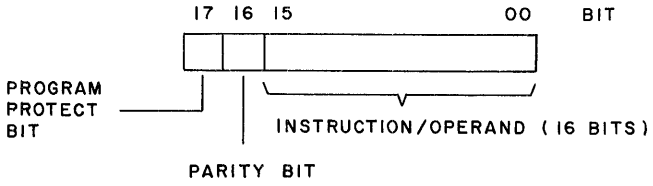
The basic 1700 Computer System provides high-speed, random-access magnetic core storage for 4,096 18-bit words. The storage capacity may be expanded from 4K, to 8K, 16K and 32K as standard increments.

Storage cycle time is 1.1 microseconds. This is defined as the shortest possible time between successive Read/Write operations in storage.

## STORAGE WORD

A storage word may be a 16-bit instruction, a 16-bit operand, or one-half of a 32-bit operand (multiply or divide). A parity bit and a program protect bit are appended to each 16-bit storage word; thus a storage word is 18 bits long.

Format:



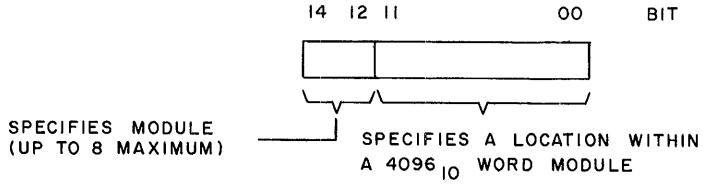

Bit 16 is the parity bit. It takes on a value so that the total number of 1 bits is odd (total number of bits includes the program protect bit). For example; if all 16 data bits are 1's and the program protect bit is 0, the parity bit is a 1.

Bit 17 is the program protect bit. If it is a 1, it indicates the word is part of a protected program.

## STORAGE ADDRESSING

The location of each word in storage is identified by an assigned number (address). An address consists of 15 bits.

Format:



Bits 00-11 specify a location within a $4096_{10}$-word storage increment (module). Bits 12-14 specify one of up to eight storage increments (modules).

Sometimes a condition arises in which the program references an address in a nonexistent storage module. In this case, the computer references the address specified by bits 00-11 in some existing storage module (storage addressing wraparound). Table 2-1 lists the actual 1700 storage size, the storage module addressed, and the effective module addressed.

TABLE 2-1. STORAGE MODULE ADDRESSING RELATIONSHIPS

| | STORAGE MODULE ADDRESSED | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Storage Size | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |
| 4K* | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Effective Module Addressed |
| 8K* | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | |
| 12K | 0 | 1 | 2 | 2 | 0 | 1 | 2 | 2 | |
| 16K* | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | |
| 24K | 0 | 1 | 2 | 3 | 4 | 5 | 4 | 5 | |
| 32K* | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | |

*Standard Storage Sizes

For example; if the computer has 16K ($16,284_{10}$) words of storage, the highest permissible address is $3FFF_{16}$. If the program attempts to address location $5040_{16}$ (located in a nonexistent storage module 5), it actually references location $1040_{16}$ in module 1 (see Table 2-1).

## STORAGE REGISTERS

### Z Register

The Z register is the 18-bit data storage register. It transfers data between the computer, external storage access, and the magnetic core storage. Each time a word is read from storage, it is transferred to the Z register and parity is checked. Word parity is also generated when the data in the Z register and parity is checked. Word parity is also generated when the data is in the Z register prior to writing the word into storage. Prior to writing a word into storage, the existing word at that location is first read from storage and a parity check is performed. See the interrupt chapter for a detailed description of Storage Parity Errors.

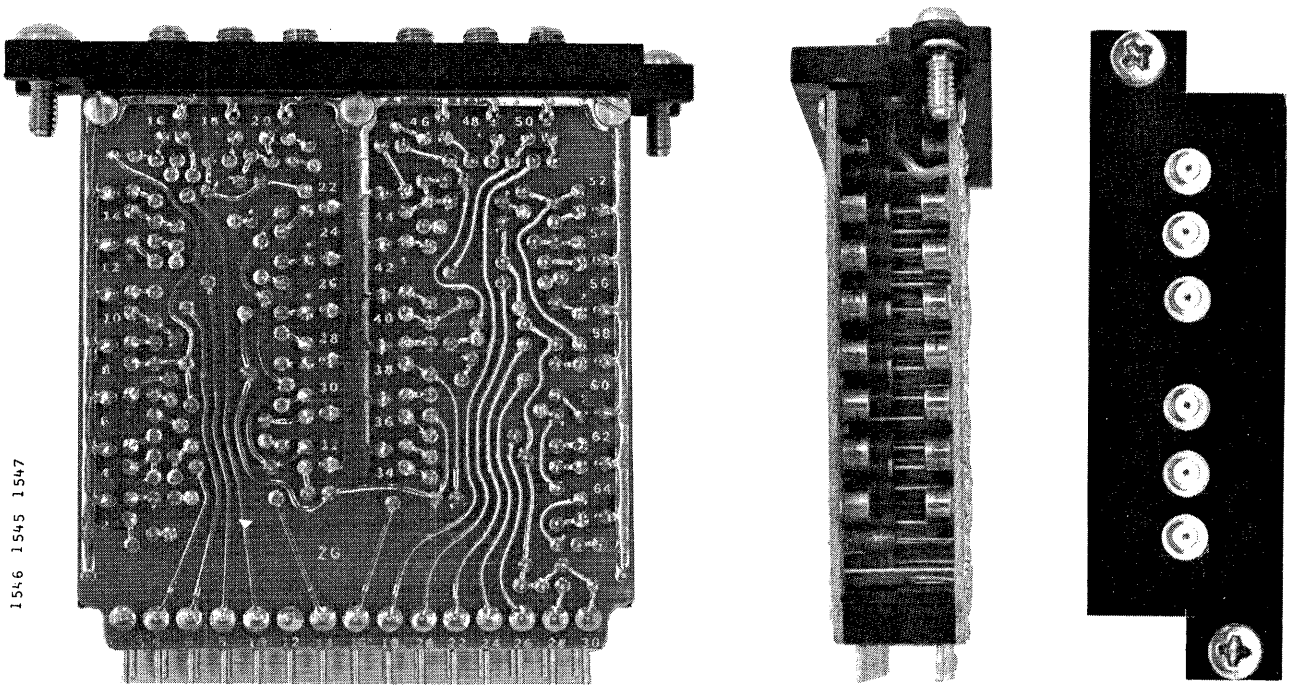
### S Register

The S register is the 15-bit address storage register. It holds the address specifying where a word is to be read from or written into storage.

## STORAGE ACCESSES

There are two accesses to the computer storage. One is internal from the computer itself and is included in the basic 1700 Computer System. The computer uses the internal access for all computation operations which reference storage.

The external access is included in the 1705 Interrupt/Data Channel. This access is via the terminated twisted-pair, transmission line technique used in the I/O of the 3000 Series computers.* The cables, connectors, signal levels, and termination are identical to the 3000 Series I/O.**



1546 1545 1547

HIGH SPEED OF THE 1700 COMPUTER is achieved through use of the same circuit design developed for the super-scale CONTROL DATA 6000 Computer Systems. Circuit packaging also utilizes the 6000 techniques. Components are mounted on and between two printed circuit boards. A 30-pin connector provides in-out electrical access for each circuit module. Six screw-on test points facilitate oscilloscope monitoring of circuit performance.

*Maximum total cable length from the external storage access is 80 feet.

**The 3000 Series controllers, synchronizers, etc. are not compatible with the 1700 Computer System.

The 1704 Computer 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 magnetic core storage of the computer system.

Functionally, the computer 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. In all discussions of registers, the rightmost bit is the least significant and is defined as bit 00. The operational registers are described below. Figure 3-1 is a block diagram of the computer.

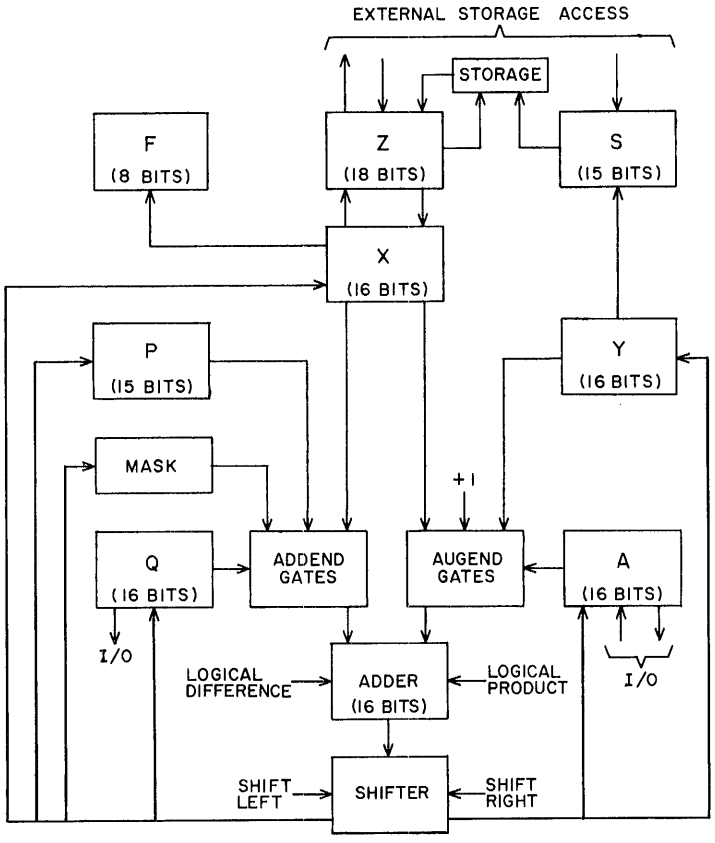



Figure 3-1. Block Diagram: 1704 Computer

**A Register**

The A register is the principal arithmetic register containing 16 bits (15-00) of which bit 16 is the sign bit. The A register also serves as the data interface during I/O operations.

**Q Register**

The Q register is an auxiliary arithmetic register containing 16 bits (15-00). This register is also used as an index register for instructions requiring indexing. The Q register also holds the address of a peripheral device during I/O operations.

**P Register**

The 15-bit P register functions as a program address counter. The P register holds the address of each instruction. After executing the instruction at address P, the quantity in P is advanced to the address of the next instruction. The amount by which P is advanced is determined by the type of instruction being executed.

Since the count in P is advanced by a 16-bit one's complement adder, P can generate storage addresses in sequence from $0000_{16}$ to $7FFF_{16}$. When a count of $7FFF_{16}$ is reached, the next count in P reduces its value to $0000_{16}$.

**X Register**

The X register is an exchange register containing 16 bits (15-00). This register holds data going to or from storage. It also holds one of the parameters in most arithmetic operations.

**Y Register**

The Y register is an address register containing 16 bits (16-00). In this register, storage addresses are formed and held for transfer during a storage reference. The Y register is also used as a counter during Multiply, Divide, and Shift instructions.

**F Register**

The F register is a function register containing 8 bits (07-00). This register holds the instruction identification and/or addressing mode bits during the execution of an instruction.

## CONTROL SECTION

The control section of the computer directs the operations required to execute instructions and establishes the timing relationships needed to perform these operations in the proper sequence. It also controls interrupt processing, program protection, and operations involving I/O and storage.

The control section acquires an instruction from storage, interprets it, and sends the necessary commands to other sections. The program address counter, P, provides program continuity by generating in sequence the storage addresses which contain the individual instructions. Usually, at the completion of each instruction, the count in P is advanced by one to specify the address of the next instruction in the program.

## INSTRUCTION FORMATS

There are three types of instructions in the 1704 Computer: storage reference, register reference, and skip.

- Storage reference instructions reference storage for operands.
- Register reference instructions operate on the computer registers or control logic.
- Skip instructions sense the existence of specific conditions within the computer.

Five instruction formats are required, one for each type of instruction plus one for the Shift instructions and one for the Interregister instructions which are subgroups of the register reference instructions.

Hexadecimal notation is used in this computer for ease in expressing the 4-bit groups which occur in the instruction format. Hexadecimal is base 16 and requires the additional characters A, B, C, D, E, and F. The relationships between binary, decimal, and hexadecimal are shown in Table 3-1.

TABLE 3-1. BINARY-DECIMAL-HEXADECIMAL RELATIONSHIPS

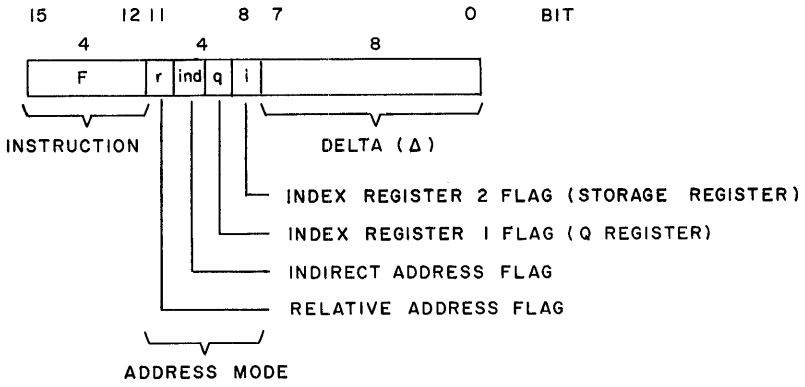
| DECIMAL | HEXADECIMAL | BINARY | DECIMAL | HEXADECIMAL | BINARY |
|---|---|---|---|---|---|
| 0 | 0 | 0000 | 8 | 8 | 1000 |
| 1 | 1 | 0001 | 9 | 9 | 1001 |
| 2 | 2 | 0010 | 10 | A | 1010 |
| 3 | 3 | 0011 | 11 | B | 1011 |
| 4 | 4 | 0100 | 12 | C | 1100 |
| 5 | 5 | 0101 | 13 | D | 1101 |
| 6 | 6 | 0110 | 14 | E | 1110 |
| 7 | 7 | 0111 | 15 | F | 1111 |

## Storage Reference Instructions

The storage reference instructions contain three fields: instruction, address mode, and delta.

The instruction field contains the 4-bit operation code, F. The address mode contains 4 flags for indexing, indirect addressing, and relative addressing. The sign bit of delta is extended in all cases except those noted. Delta is treated as zero when its contents = $00000000_2$.

Format:



The following definitions apply to the descriptions of addressing modes.

- Instruction Address: the address of the instruction being executed = P.
- Indirect Address: a storage address which contains an address rather than an operand.
- Base Address: the operand address after all indirect addressing but before modification by Index registers. The base address is the effective address if no indexing is specified.
- Effective Address: the final address of the operand. In certain cases, the effective address equals the operand for read operand type instructions. These cases are noted in Table 3-2.
- Indexing: The computer has two Index registers. Index register number 1 is the Q register, and Index register number 2 is storage location $FF_{16}$ The base address may be modified by either one or both of the Index registers. If the Index register number flag is set, the contents of the Q register are added to the base address to form the effective address. If the Index register number 2 flag is set, the contents of storage location $FF_{16}$ are added to the base address to form the effective address. If both Index register flags are set, the contents of Q are added to the base address; then the contents of $FF_{16}$ are added to the result to form the effective address. Indexing occurs after indirect addressing has been completed.

  The computer uses the 16-bit one's complement adder during Indexing operations. Consequently, Index register contents are treated as signed quantities (bit 15 = sign bit).

Storage reference instructions has seven different types of addressing modes. They are:

1) Absolute (address mode bits = 0, 1, 2 or 3). Relative and indirect flags both equal "0". The base address equals delta. The sign bit of delta is not extended. The contents of the Index registers when specified are added to the base address to form the effective address.

2) Indirect (address mode bits = 4, 5, 6, or 7). Relative flag equals "0", and indirect address flag equals "1". Delta is not equal to "0". The 8-bit value of delta is an indirect address. The sign bit of delta is not extended.

3) Relative (address mode bits = 8, 9, A, or B). The relative flag equals "1", indirect address flag equals "0", and delta does not equal "0". The base address is equal to the instruction address, P, plus the value of delta with sign extended. The contents of the Index registers when specified are added to the base address to form the effective address.

4) Relative Indirect (address mode bits = C, D, E, or F). Both relative and indirect address flags equal "1", delta does not equal "0", and the value of the instruction address, P, plus the value of delta with sign extended is an indirect address.

   If bit 15 of the contents of this indirect address is "0", the contents of this indirect address is the base address. If bit 15 of the contents of the indirect address is a "1", the contents of the indirect address is another indirect address.

   Indirect addressing continues until bit 15 of the contents of an indirect address is "0". The contents of the Index registers when specified are added to the base address to form the effective address.

5) Constant (address mode bits = 8, 9, A, or B). Relative flag equals "1"; indirect address flag and delta equal "0". If no indexing is specified, the instruction address, P, plus 1 is the effective address. If indexing is specified, the contents of the Index registers are added to the contents of the storage location P plus 1 to form the operand for read operand type instructions, or the effective address for store instructions.

6) Storage (address mode bits = C, D, E, or F). Relative and indirect address flags equal "1"; delta equals "0". Bit 15 of the contents of P plus 1 equals "0". The contents of P plus 1 is the base address. The contents of the Index registers when specified are added to the base address to form the effective address.

7) Storage Indirect (address mode bits = C, D, E, or F). Relative and indirect address flags equal "1", delta equals "0". Bit 15 of the contents of P plus 1 equals "1". The contents of P plus 1 is an indirect address.

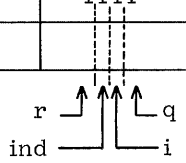
   If bit 15 of the contents of this indirect address is "0", the contents of this indirect address is the base address. If bit 15 of the contents of the indirect address is "1", the contents of the indirect address is another indirect address.

   Indirect addressing continues until bit 15 of the contents of an indirect address is "0". The contents of the Index registers, when specified, are added to the base address to form the effective address.

Table 3-2 shows all the addressing possibilities for storage reference instructions which may be obtained through combinations of flag bits.

TABLE 3-2. STORAGE ADDRESSING RELATIONSHIPS

| Mode | Address Mode Bits (11-08) Binary | Hex. | | Effective Address | Address of Next Instruction |
|---|---|---|---|---|---|
| Absolute | 0000 | 0 | | Δ | P + 1 |
| | 0001 | 1 | | Δ + (00FF) | P + 1 |
| | 0010 | 2 | | Δ + (Q) | P + 1 |
| | 0011 | 3 | | Δ + (Q) + (00FF) | P + 1 |
| Indirect | 0100 | 4 | | (Δ) | P + 1 |
| | 0101 | 5 | | (Δ) + (00FF) | P + 1 |
| | 0110 | 6 | | (Δ) + (Q) | P + 1 |
| | 0111 | 7 | | (Δ) + (Q) + (00FF) | P + 1 |
| Relative | 1000 | 8 | Δ ≠ 0 | P + Δ | P + 1 |
| Constant | | | Δ = 0 | P + 1 | P + 2 |
| Relative | 1001 | 9 | Δ ≠ 0 | P + Δ + (00FF) | P + 1 |
| Constant | | | Δ = 0 | (P + 1) + (00FF)* | P + 2 |
| Relative | 1010 | A | Δ ≠ 0 | P + Δ + (Q) | P + 1 |
| Constant | | | Δ = 0 | (P + 1) + (Q)* | P + 2 |
| | | | | | |
| Relative | 1011 | B | Δ ≠ 0 | P + Δ + (Q) + (00FF) | P + 1 |
| Constant | | | Δ = 0 | (P + 1) + (Q) + (00FF)* | P + 2 |
| Relative Ind. | 1100 | C | Δ ≠ 0 | (P + Δ) | P + 1 |
| Storage | | | Δ = 0 | (P + 1) | P + 2 |
| Relative Ind. | 1101 | D | Δ ≠ 0 | (P + Δ) + (00FF) | P + 1 |
| Storage | | | Δ = 0 | (P + 1) + (00FF) | P + 2 |
| Relative Ind. | 1110 | E | Δ ≠ 0 | (P + Δ) + (Q) | P + 1 |
| Storage | | | Δ = 0 | (P + 1) + (Q) | P + 2 |
| Relative Ind. | 1111 | F | Δ ≠ 0 | (P + Δ) + (Q) + (00FF) | P + 1 |
| Storage | | | Δ = 0 | (P + 1) + (Q) + (00FF) | P + 2 |




*Effective Address = operand for read operand type instructions.

Data Transmission

STQ (F = 4)

Store Q Store the contents of the Q register in the storage location specified by the effective address. The contents of Q are not changed.

STA (F = 6)

Store Q Store the contents of the A register in the storage location specified by the effective address. The contents of A are not altered.

SPA (F = 7)

Store A, Parity to A Store the contents of the A register in the storage location specified by the effective address. Clear A if the number of "1" bits in A is odd. Set A = $0001_{16}$ if the number of "1" bits in A is even. The contents of A are not altered if the write into storage is aborted because of parity error or protect fault.

LDA (F = C)

Load A Load the A register with the contents of the storage location specified by the effective address. The contents of the storage location are not altered.

LDQ (F = E)

Load Q Load the Q register with the contents of the storage location specified by the effective address. The contents of the storage location are not altered.

Arithmetic

All the following arithmetic operations use one's complement arithmetic.

MUI (F = 2)

Multiply Integer Multiply the contents of the storage location specified by the effective address by the contents of the A register. The 32-bit product replaces the contents of Q and A with the most significant bits in the Q register.

DVI (F = 3)

Divide Integer Divide the combined contents of the Q and A registers by the contents of the effective address. The Q register contains the most significant bits before execution. The quotient is in the A register and the remainder in the Q register at the end of the Divide operation. If a 16-bit dividend is loaded into A, the sign of A must be set (the sign of the dividend A must be extended throughout A).

The OVERFLOW indicator is set if the magnitude of the quotient is greater than the capacity of the A register. Once set, the OVERFLOW indicator remains set until a Skip On Overflow instruction is executed.

ADD (F = 8)

Add to A Add the contents of the storage location specified by the effective address to the contents of the A register.

The OVERFLOW indicator is set if the magnitude of the sum is greater than the capacity of the A register. Once set, the OVERFLOW indicator remains set until a Skip On Overflow instruction is executed.

SUB (F = 9)

Subtract From A Subtract the contents of the storage location specified by the effective address from the contents of the A register. Operation on overflow is the same as for an Add to A instruction.

RAO (F = D)

Replace Add One in Storage Add one to the contents of the storage location specified by the effective address. The contents of A and Q are not changed. Operation on overflow is the same as for an Add to A instruction.

ADQ (F = F)

Add to Q Add the contents of the storage location specified by the effective address to the contents of the Q register. Operation on overflow is the same as for an Add to A instruction.

Logical

The AND (AND With A) instruction achieves its result by forming a logical product. A logical product is a bit-by-bit multiplication of two binary numbers according to the following rules:

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

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

Example:
```
0011  Operand A
0101  Operand B
----
0001  Logical Product
```

A logical product is used, in many cases, to select only specific portions of an operand for use in some operation. For example; if only a specific portion of an operand in storage is to be entered into the A register, the operand is subjected to a mask in A. This mask is composed of a predetermined pattern of "0's" and "1's". Executing the AND instruction causes the operand to retain its original contents only in those stages which have "1's" in the mask in A.

The EOR (Exclusive OR With A) instruction achieves its result by forming an exclusive OR. Executing the EOR instruction causes the operand to complement its original contents only in those stages which have "1's" in the mask in A. An exclusive OR is a bit-by-bit logical subtraction of two binary numbers according to the following rules:

Exclusive OR

| A | B | A ∀ B |
|---|---|---|
| 1 | 1 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 0 | 0 | 0 |

Example:
```
0011  Operand A
0101  Operand B
0110  Exclusive Or
```

AND (F = A)

AND With A  Form the logical product, bit by bit, of the contents of the storage location specified by the effective address and the contents of the A register. The result replaces the contents of A. The contents of storage are not altered.

EOR (F = B)

Exclusives OR With A  Form the logical difference (exclusive OR), bit by bit, of the contents of the storage location specified by the effective address and the contents of the A register. The result replaces the contents of A. The contents of storage are not altered.

Jumps

A Jump (JMP) 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 instructions and always contains the storage location of the current instruction in the program.

When a Jump instruction occurs, P is cleared and a new address is entered*. In the Jump instruction, the effective address specifies the beginning address of the new program sequence. The word at the effective address is read from storage and interpreted as the first instruction of the new sequence.

A Return Jump (RTJ) instruction enables the computer to leave the main program, jump to some subprogram, execute the subprogram, and return to the main program via another instruction. The Return Jump provides the computer with the necessary information to enable returning to the main program. Figure 3-2 shows how a Return Jump instruction can be used.

*Jumps or Return Jumps from unprotected to protected storage cause a fault, but the address that is saved in the trap location is the destination address, i.e., the address of the next logical instruction. See Program Protection in Section 4.

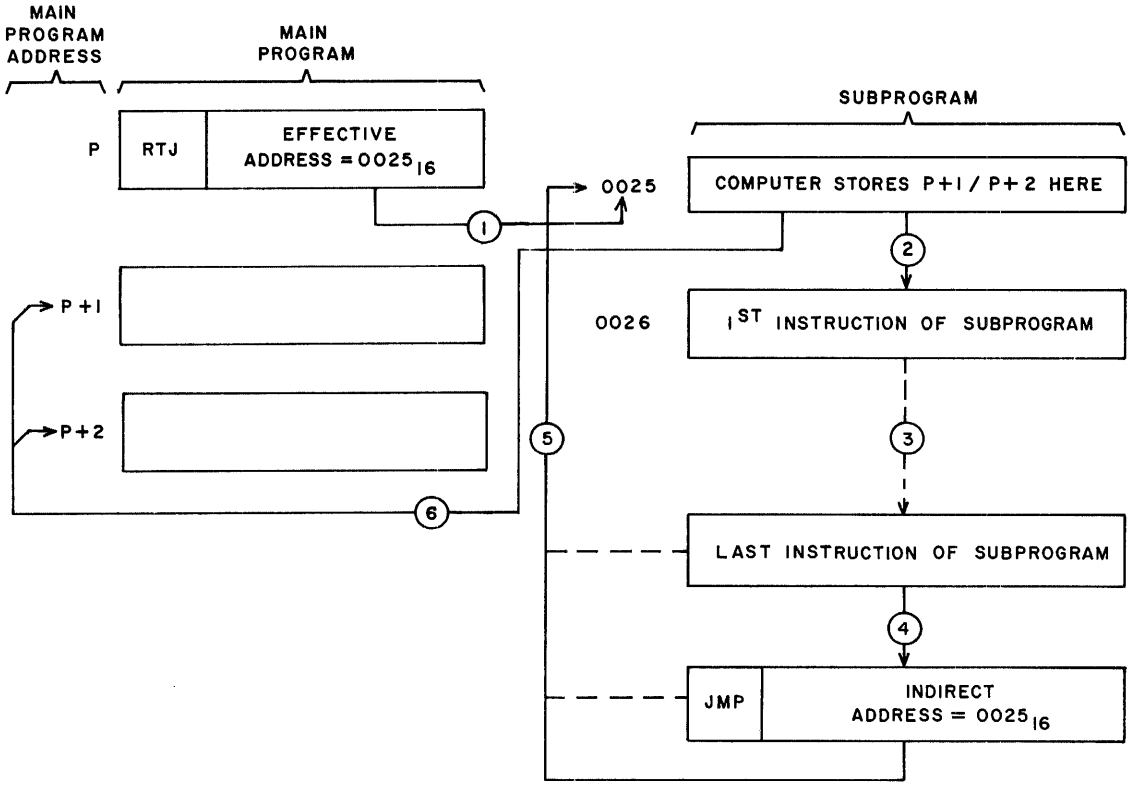



Figure 3-2. Program Using Return Jump Instruction

A Return Jump instruction is executed at main program address P. The computer jumps to effective address $0025_{16}$ and stores P plus 1 or P plus 2 (depending upon the addressing mode of RTJ) at this location. Then the program address counter, P, is set to $0026_{16}$ and the computer starts executing the subprogram. At the end of the subprogram, the computer executes a Jump instruction (JMP) with indirect addressing. This causes the computer to jump to the address specified by the subprogram address $0025_{16}$ which is P plus 1 or P plus 2 of the main program. Now main program execution continues at P plus 1 or P plus 2.

JMP (F = 1)

Jump Jump to the address specified by the effective address. This effectively replaces the contents of the program address counter, P, with the effective address specified in the jump instruction.

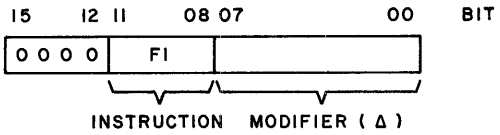
RTJ (F = 5)

Return Jump Replace the contents of the storage location specified by the effective address with the address of the next consecutive instruction. The address stored in the effective address will be P plus 1 or P plus 2, depending upon the addressing mode of RTJ. The contents of P are then replaced with the effective address plus 1.

## Register Reference Instructions

Register reference instructions use the address mode field for the operation code. Register reference instructions are identified when the upper 4 bits (15-12) of an instruction are "0's".

Format:




SLS (F1 = 0)

Selective Stop Stops the computer if this instruction is executed when the STOP switch is on. This becomes a Pass instruction when the switch is off. On restart, executes P plus 1.

INP (F1 = 2)

Input to A Read one word from an external device into the A register. The word in the Q register selects the sending device. If the device sends a Reply, the next instruction comes from P plus 1. If the device sends a Reject, the next instruction comes from P plus 1 plus Δ, where Δ is an 8-bit signed number, including sign. If an Internal Reject occurs, the next instruction comes from P plus Δ.

OUT (F1 = 3)

Output from A Output one word from the A register to an external device. The word in the Q register selects the receiving device. If the device sends a Reply, the next instruction comes from P plus 1. If the device sends a Reject, the next instruction comes from P plus 1 plus Δ, where Δ is an 8-bit signed number, including sign. If an Internal Reject occurs, the next instruction comes from P plus Δ.

INA (F1 = 9)

Increase A Replaces the contents of A with the sum of the initial contents of A and Δ. Where Δ is treated as a signed number with the sign extended into the upper 8 bits. Operation on overflow is the same as for an Add to A instruction.

ENA (F1 = A)

Enter A Replaces the contents of the A register with the 8-bit Δ, sign extended.

ENQ (F1 = C)

Enter Q Replaces the contents of the Q register with the 8-bit Δ, sign extended.

INQ (F1 = D)

Increase Q Replaces the contents of Q with the sum of the initial contents of Q and Δ, where Δ is treated as a signed number, with the sign extended into the upper 8 bits. Operation on overflow is the same as for an Add to A instruction.

The following instructions (F1 = 4, 5, 6, 7, E) are legal only if the PROGRAM PROTECT switch is off or if the instructions themselves are protected. If an instruction is illegal, it becomes a selective stop, and an interrupt on program protect fault is possible (if selected).

- Switch ON: Pass unless instruction is protected (program protect bit set).
- Switch OFF: Normal instruction execution (no program protection).

EIN (F1 = 4)

Enable Interrupt Activates the interrupt system after one instruction following EIN has been executed. The interrupt system must be active and the appropriate mask bit set for an interrupt to be recognized.

IIN (F1 = 5)

Inhibit Interrupt Deactivates the interrupt system. If interrupt occurs during execution of this instruction, the interrupt is not recognized until one instruction after the next EIN instruction is executed.

SPB (F1 = 6)

Set Program Protect Bit Sets the program protect bit in the address specified by Q.

CPB (F1 = 7)

Clear Program Protect Bit Clears the program protect bit in the address specified by Q.

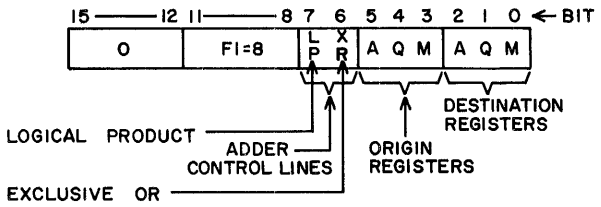
EXI (F1 = E)

Exit Interrupt State This instruction must be used to exit from any interrupt state. Delta defines the interrupt state from which the exit is taken (see Table 4-1). This instruction automatically reads the address containing the return address, resets the OVERFLOW indicator according to bit 16, activates the interrupt system, and jumps to the return address.

Interregister

This instruction causes data from certain combinations of two origin registers to be sent through the adder to any combination of destination registers. Various operations, selected by the adder control lines, are performed on the data as it passes through the adder.

Format:




*If bit 0 of IR = "1" and the instruction is not protected, this is a nonprotected Selective Stop instruction. The program protect fault bit is set and interrupt occurs if selected. See Section 4 for additional information.

The origin registers are considered as operands of which there are two kinds defined as follows:

Operand 1 may be:

- FFFF (bit 5 = 0) or
- the contents of A (bit 5 = 1)

Operand 2 may be:

- FFFF (bit 4 = 0 and bit 3 = 0) or
- the contents of M (bit 4 = 0 and bit 3 = 1) or
- the contents of Q (bit 4 = 1 and bit 3 = 0) or
- the inclusive OR, bit by bit, of the contents of Q and M (bit 4 = 1 and bit 3 = 1)

Operations possible are:

- LP = 0 and XR = 0. The data placed in the destination register(s) is the arithmetic sum of operand 1 and operand 2. The OVERFLOW indicator operates the same as for an Add to A instruction.
- LP = 1 and XR = 0. The data placed in the Destination register(s) is the logical product, bit by bit, of operand 1 and operand 2.
- LP = 0 and XR = 1. The data placed in the destination register(s) is the exclusive OR, bit by bit, of operand 1 and operand 2.
- LP = 1 and XR = 1. The data placed in the destination register(s) is the complement of the logical product, bit by bit, of operand 1 and operand 2.

TABLE 3-3. INTERREGISTER INSTRUCTION TRUTH TABLE

| Operand 1 | Operand 2 | LP = 0 XR = 1 | LP = 1 XR = 0 | LP = 1 XR = 1 | LP = 0 XR = 0 |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | Arith-metic Sum |
| 0 | 1 | 1 | 0 | 1 | |
| 1 | 0 | 1 | 0 | 1 | |
| 1 | 1 | 0 | 1 | 0 | |

Notes:

a. Register transfers can be accomplished with LP = 0, XR = 0, and making operand 1 or operand 2 equal to $FFFF_{16}$.

b. Magnitude comparisons without destroying either operand can be done with LP = 0, XR = 0, no destination register selected, and testing the OVERFLOW indicator.

c. Complementing registers can be done with LP = 0, XR = 1, and making operand 1 or operand 2 equal to $FFFF_{16}$.

Interregister Mnemonics:

| | |
|---|---|
| SET | (F1 = 8, bits 7-3 = 10000) Set To Ones |
| CLR | (F1 = 8, bits 7-3 = 01000) Clear To Zero |
| TRA | (F1 = 8, bits 7-3 = 10100) Transfer A |
| TRM | (F1 = 8, bits 7-3 = 10001) Transfer M |
| TRQ | (F1 = 8, bits 7-3 = 10010) Transfer Q |
| TRB | (F1 = 8, bits 7-3 = 10011) Transfer Q + M |
| TCA | (F1 = 8, bits 7-3 = 01100) Transfer Complement A |
| TCM | (F1 = 8, bits 7-3 = 01001) Transfer Complement M |
| TCQ | (F1 = 9, bits 7-3 = 01010) Transfer Complement Q |
| TCB | (F1 = 8, bits 7-3 = 01011) Transfer Complement Q + M |
| AAM | (F1 = 8, bits 7-3 = 00101) Transfer Arithmetic Sum A, M |
| AAQ | (F1 = 8, bits 7-3 = 00110) Transfer Arithmetic Sum A, Q |
| AAB | (F1 = 8, bits 7-3 = 00111) Transfer Arithmetic Sum A, Q + M |
| EAM | (F1 = 8, bits 7-3 = 01101) Transfer Exclusive Or A, M |
| EAQ | (F1 = 8, bits 7-3 = 01110) Transfer Exclusive Or A, Q |
| EAB | (F1 = 8, bits 7-3 = 01111) Transfer Exclusive Or A, Q + M |

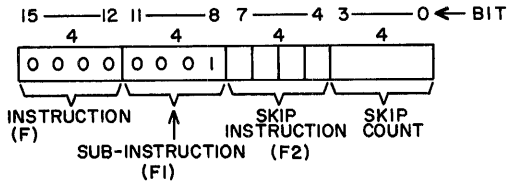
| | |
|---|---|
| LAM | (F1 = 8, bits 7-3 = 10101) Transfer Logical Product A, M |
| LAQ | (F1 = 8, bits 7-3 = 10110) Transfer Logical Product A, Q |
| LAB | (F1 = 8, bits 7-3 = 10111) Transfer Logical Product A, Q + M |
| CAM | (F1 = 8, bits 7-3 = 11101) Transfer Complement Logical Product A, M |
| CAQ | (F1 = 8, bits 7-3 = 11110) Transfer Complement Logical Product A, Q |
| CAB | (F1 = 8, bits 7-3 = 11111) Transfer Complement Logical Product A, Q + M |

Note: " + " symbol implies an Or.

Shifts

These Shift instructions shift A, Q, or QA left or right the number of places specified by the 5-bit shift count. Right shifts are end-off with sign extension in the upper bits. Left shifts are end-around. The maximum long right or long left shift is $31_{10}$ places.

Format:

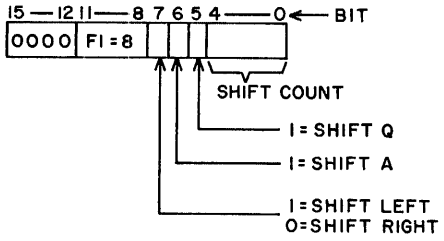



| | |
|---|---|
| ARS | (F1 = F) A Right Shift |
| QRS | (F1 = F) Q Right Shift |
| LRS | (F1 = F) Long Right Shift (QA) |
| ALS | (F1 = F) A Left Shift |
| QLS | (F1 = F) Q Left Shift |
| LLS | (F1 = F) Long Left Shift (QA) |
| NOP | (F1 = B) No Operation |

## Skip Instructions

Skip instructions are identified when the instruction mode field is 0 and the subinstruction mode field is 1.

Format:




When the skip condition is met, the contents of the skip count plus 1 is added to P to obtain the address of the next instruction (e.g., when the skip count is zero, go to P plus 1). When the skip condition is not met, the address of the next instruction is P plus 1 (skip count ignored). The skip count does not have a sign bit.

| | |
|---|---|
| SAZ | (F2 = 0) Skip if A = +0 |
| SAN | (F2 = 1) Skip if A ≠ +0 |
| SAP | (F2 = 2) Skip if A = + |
| SAM | (F2 = 3) Skip if A = - |
| SQZ | (F2 = 4) Skip if Q = +0 |
| SQN | (F2 = 5) Skip if Q ≠ +0 |
| SQP | (F2 = 6) Skip if Q = + |
| SQM | (F2 = 7) Skip if Q = - |
| SWS | (F2 = 8) Skip if Switch set |
| SWN | (F2 = 9) Skip if Switch not set |

SOV (F2 = A)

Skip on Overflow This instruction skips on the condition and also clears the Overflow indicator.

SNO (F2 = B) Skip on No Overflow

SPE (F2 = C)

Skip on Storage Parity Error This instruction skips on the condition and also clears the Storage Parity Error interrupt signal and the Storage Parity Error indicator.

SNP (F2 = D) Skip on no Storage Parity Error

SPF (F2 = E)

Skip on Program Protect Fault Program protect fault is set by:

- A nonprotected instruction attempting to write into an address which is protected.
- An attempt to execute a protected instruction immediately following a nonprotected instruction except if an interrupt caused the instruction sequence.
- Execution of any nonprotected instruction affecting interrupt mask or enables.

The program protect fault is cleared when it is sensed by the SPF instruction. The program protect fault cannot be set if the program protect system is disabled.

SNF (F2 = F)

Skip on No Program Protect Fault Refer to program protection in Section 4 for conditions causing a program protect fault.

# INTERRUPT SYSTEM AND PROGRAM PROTECTION 4

## INTERRUPT SYSTEM

The computer 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 these conditions exists and the conditions for interrupting are present, execution of the main program halts. The contents of the Program Address register, P, is stored at a fixed address and an interrupt routine is initiated. This interrupt routine takes the necessary action for the condition and then returns control to the next unexecuted instruction in the main program.

For each condition that can cause an interrupt, the program has two alternatives. It may select an interruptible condition, so that interrupt occurs when that condition arises, or it may choose to have the interrupt system ignore the condition. The program also has the choice of whether the interrupt system is to be used. The EIN and IIN instructions activate and deactivate the interrupt system.

The interrupt system gives the program the ability to establish priority of interrupts so that an interrupt of high priority can interrupt the machine while processing an interrupt of a lower priority. The return path to the lower priority interrupt routine(s) and then to the main program is clearly established and saved.

If all conditions for interrupting have been met, the main program is interrupted just before the next storage reference. Consequently:

- If conditions for interrupting occur while the computer is reading up an instruction which references storage, the main program is interrupted before that instruction is executed.
- If conditions for interrupting occur while the computer is reading up an instruction which does not reference storage (e.g., Inter-register instruction), interrupt does not occur until after the computer has executed that instruction.
- If conditions for Interrupting occur while the computer is reading up an indirect address and bit 15 is set, the interrupt occurs before that instruction is executed.

In all three preceeding cases, the value of P stored at the fixed interrupt trap location is such as to enable return to the next unexecuted instruction in the main program after interrupt processing.

### Logical Description of Interrupt System

The interrupt system consists of fixed Interrupt Trap Locations and the interrupt Mask register.

## Basic and Optional Interrupts

The basic computer has two interrupts, one internal (storage parity error or program protect fault = interrupt state 00), and one external (interrupt state 01). The 1705 option may be added which gives an additional 14 external interrupt lines. Thus, the computer may have up to 16 different interrupts. The discussions that follow assume the computer has 16 interrupts.

## Interrupt Trap Locations

Interrupt trap locations are established for each interrupt line. They are in the range of addresses 0100 through $013C_{16}$. The assignment for each interrupt state or line is shown in Table 4-1. The first column is the interrupt state. The second column is the value of Δ to be used in the Exit Interrupt instruction to exit from that state. The third column is the address where the contents of the Program Address register is stored when an interrupt occurs. The fourth column is the address of the first instruction to be executed following an interrupt. These addresses are reserved exclusively for interrupts unless that particular interrupt is not being used.

TABLE 4-1. INTERRUPT STATE DEFINITIONS

| | Interrupt state$_{16}$ | Value of Δ to exit state$_{16}$ | Location of return address$_{16}$ | Location of first instruction after interrupt occurs$_{16}$ |
|---|---|---|---|---|
| * | 00 | 00 | 0100 | 0101 |
| * | 01 | 04 | 0104 | 0105 |
| | 02 | 08 | 0108 | 0109 |
| | 03 | 0C | 010C | 010D |
| | 04 | 10 | 0110 | 0111 |
| | 05 | 14 | 0114 | 0115 |
| | 06 | 18 | 0118 | 0119 |
| | 07 | 1C | 011C | 011D |
| ** | 08 | 20 | 0120 | 0121 |
| | 09 | 24 | 0124 | 0125 |
| | 10 | 28 | 0128 | 0129 |
| | 11 | 2C | 012C | 012D |
| | 12 | 30 | 0130 | 0131 |
| | 13 | 34 | 0134 | 0135 |
| | 14 | 38 | 0138 | 0139 |
| | 15 | 3C | 013C | 013D |

*Interrupts in basic computer

**Interrupts added by 1705 Interrupt/Data Channel option

Mask Register

The 16-bit Mask register is the enable for each interrupt state or line. Bit 00 of the Mask register corresponds to interrupt line 0, bit 01 to line 1, etc. To enable an interrupt line, its corresponding bit in the Mask register must be set. The Mask register is set by the interregister instruction. The basic computer has mask bits 0 through 3. The 1705 option adds mask bits 4 through 15.

## Programming and Operation of 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:

- The interrupt system must be activated.
- Internal and external conditions to be tested must be selected with various masks.
- Interrupt routines must be programmed to determine the cause(s) of interrupt and to process and clear the interrupt.

The computer can distinguish up to 16 different interrupts. Each of these interrupts has its respective bit in the interrupt Mask register and its respective address to which control is transferred upon recognizing the interrupt.

When the computer is processing a particular interrupt, it is defined as being in that interrupt state (state 00 through 15). Thus, the interrupts and their respective bits in the interrupt Mask register are numbered 00 through 15. An interrupt in bit 7 puts the computer in interrupt state 7, etc.

Before the computer can recognize any interrupt, the mask bit for that interrupt must be set and the interrupt system must be activated. The Mask register can be set by an interregister instruction, and the interrupt system is activated by an Enable Interrupt instruction.

Upon recognizing an interrupt, the computer automatically stores the return address in the lower 15 bits of the storage location reserved for that interrupt state. Bit 16 of the storage location is set or cleared to record the current state of the OVERFLOW indicator. The OVERFLOW indicator itself is then cleared.

The computer then deactivates the interrupt system and transfers control to another address also specified by the interrupt state. The program then stores all registers, including the Mask register, in addresses reserved for this interrupt state and loads the Mask register with the mask to be used while in this state. One's in the mask denote interrupts that have higher priority than the interrupt being processed. The mask should not have a "1" in the position of the interrupt being processed. If an interrupt was allowed into the same state which is being processed, the return link is lost. The program then activates the interrupt system and processes the interrupt.

The computer exits from an interrupt state as follows. The program inhibits interrupt and restores the registers, including the Mask register. After loading the register, the program executes the Exit Interrupt instruction with Δ equal to the lower 8 bits of the base address of the interrupt state. This instruction reads the storage location where the return address is stored. The OVERFLOW indicator is set or cleared in accordance with bit 16, the interrupt system is activated, and control transfers to the return address.

## Interrupt Priority

The priority of interrupts is under control of the computer program. The program assigns priority by establishing an interrupt mask for each interrupt state which enables all higher priority interrupts and disables all lower priority interrupts. When an interrupt state is entered, the mask for that state is placed in the Mask register. There may be up to 16 levels of priority. It is possible to change priority during execution of a program.

If two or more interrupts have equal priority and occur at the same time, the computer recognizes the lowest interrupt line.

The following table and sample program steps apply if there are five different possible interrupts and the programmer wants three levels of priority so that interrupt 01 has high priority, interrupts 02 and 05 have next priority, and interrupt 03 and 04 have low priority. Interrupt 00 has highest priority, but this example does not consider interrupt 00.

| Bit | 5 | 4 | 3 | 2 | 1 | 0 | |
|---|---|---|---|---|---|---|---|
| Mask 1 | 1 | 1 | 1 | 1 | 1 | 1 | Mask used for Main Program |
| Mask 2 | 1 | 0 | 0 | 1 | 1 | 1 | Mask used for State 03, 04 |
| Mask 3 | 0 | 0 | 0 | 0 | 1 | 1 | Mask used for State 02, 05 |
| Mask 4 | 0 | 0 | 0 | 0 | 0 | 1 | Mask used for State 01 |

Main Program

```
Set mask reg. to Mask 1
Enable int.
   ---
   ---
   ---
   ---
   ---
   ---
```

State 01 Program

```
Store Registers
Set Mask to Mask 4
Enable int.
      -
      -
Inhibit int.
Replace registers
Exit int. 01
```

State 02 Program

```
Store Registers
Set Mask to Mask 3
Enable int.
      -
      -
Inhibit int.
Replace registers
Exit int. 02
```

State 03 Program

```
Store Registers
Set Mask to Mask 2
Enable int.
      -
Inhibit int.
Replace registers
Exit int. 03
```

| State 04 Program | State 05 Program |
|---|---|
| Store Registers | Store Registers |
| Set Mask to Mask 2 | Set Mask to Mask 3 |
| Enable int. | Enable int. |
| - | - |
| - | - |
| Inhibit int. | Inhibit int. |
| Replace registers | Replace registers |
| Exit int. 04 | Exit int. 05 |

Sharing Subroutines Between Interrupt Levels

Properly programmed,it is possible for programs in different Interrupt states to reference the same subroutine. The first instruction in the subroutine must be an IIN, and the last two instructions must be EIN and JMP.

Example:

| Main Program | α [return link] |
|---|---|
| Interrupt State 1 | α + 1 IIN - inhibit interrupt |
| - | - |
| - | - |
| - | - |
| RTJ α | - |
| - | - |
| Interrupt State 2 | EIN - enable Interrupt |
| - | JMP (Indirect α ) |
| - | |
| - | |
| RTJ α | |
| - | |
| - | |
| - | |

Interrupts occurring after the execution of the RTJ are blocked, because the IIN is executed. These interrupts are not recognized until after the jump is executed, because one instruction must be executed after an EIN before the interrupt system is active.

## Internal Interrupts

Certain internal interrupts are generated by conditions arising within the computer. If such a condition occurs, it generates interrupt 00 (corresponding interrupt mask bit = 00). Normally, internal interrupts are assigned the highest priority. These interrupts are:

- Storage parity error
- Program protect fault

Storage Parity Error

A storage parity bit is generated and entered with every word written into storage. Storage parity is checked (and consequently an error could occur) in two cases:

1) When instructions/data are read from storage, parity is checked.
2) When a word is written into storage, the existing word at the address is first read from storage and its parity is checked.

A 00 interrupt state occurs if conditions for interrupting (mask bit 00 set and interrupt active) are present and a storage parity error occurs in either operation 1 or 2.

Note that once a storage parity error occurs and the PROGRAM PROTECT switch is set whether or not interrupt is selected), no instruction can write into storage unless the instruction is protected. The operation that store P or P plus 1 when interrupt occurs is the protected operation.

The Storage Parity Error Interrupt signal and indicator are cleared when the computer executes a Skip On Storage Parity Error instruction.

Refer to Program Protect in this section for some special cases of storage parity errors as related to program protection and for a discussion of the Program Protect Fault interrupt.

## PROGRAM PROTECTION

The computer has a program protect system which makes it possible to protect a program in the computer from any other nonprotected program also in the computer. The system is built around a program protect bit (bit 17) contained in each word of storage. If the bit is set, that word is an operand or an instruction of the protected program. All operands and instructions in the protected program must have the program protect bit set. None of the instructions or operands of the nonprotected program may have the program bit set.

### Clearing/Setting the Program Protect Bit

The program protect instructions (SPB or CPB) are the only way in which the program protect bit may be set or cleared in each word of storage.

### Program Protect Switch

Program protect is manually enabled by a two-position switch on the computer console. If the switch is not enabling Program Protect, none of the following violations are recognized.

### Program Protect Violations

Whenever a violation of the program protect system is detected, the program protect fault is set and an internal interrupt is enabled. A 00 interrupt occurs if mask bit 00 is set and the interrupt system is active. A violation indicates that the nonprotected program has attempted an operation which could harm the protected program. The four program protect violations are:

1) A nonprotected instruction attempts to write into a storage location containing a protected instruction/operand. The contents of the storage location is not changed.
2) An attempt is made to write into a protected storage location via the external storage access when a nonprotected instruction was the ultimate source of the attempt. The contents of the storage location is not changed.
3) An attempt is made to execute a protected instruction following the execution of a nonprotected instruction. The protected instruction is executed as a nonprotected Selective Stop instruction. It is not a violation, however, if an interrupt caused this sequence of instructions.
4) An attempt is made to execute the following instructions when they are not protected: interregister with bit 0 = 1, EIN, IIN, EXI, SPB, or CPB. These instructions become nonprotected Selective Stop instructions under these circumstances.

### Storage Parity Errors as Related to Program Protection

If a nonprotected instruction is attempting to write into storage and a storage parity error is present or occurs, the word in storage is not altered and a Storage Parity Error interrupt is enabled.

If a protected instruction is attempting to write into storage and a storage parity error occurs, the word is written into storage and a Storage Parity Error interrupt is enabled.

If the computer attempts to execute a SPB or CPB instruction and a storage parity error occurs, these become pass instructions and a storage parity error interrupt is enabled.

### Peripheral Equipment Protection

All peripheral equipment essential to operation of the protected program have a Program Protect switch. If the switch is on, the peripheral device responds with a Reject to all nonprotected commands (except status requests) addressed to it. The peripheral device responds to all protected commands in the normal manner.

If the switch is off, the peripheral device responds in the normal manner to protected and nonprotected commands.

### Programming Requirements

In order for the program protect system to work, the following program requirements must be met:

- There must be completely checked out program package which handles all interrupts for the nonprotected program. This program must also be part of the protected program.
- The protected program must be a completely checked out program.

This section covers the I/O section of the computer in a general manner. Since so many I/O configurations are possible, it is impossible to cover all in detail in this manual. Refer to the specific external equipment reference manuals for detailed codes and operating information.

The computer handles I/O operations via the AQ channel (nonbuffered) or the buffered data channel.

## AQ CHANNEL

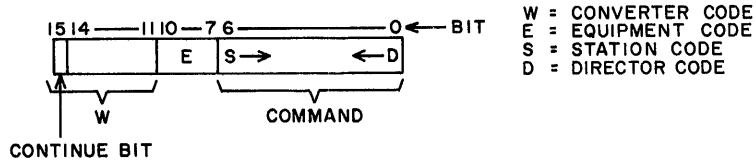
The normal I/O channel for the computer is called the AQ channel. The 16-bit A and Q registers of the computer operate together to transfer data, status, and commands on the AQ channel.

The Q register contains the address and command for the peripheral device (external equipment), and the A register receives or transmits the data or status. A maximum of eight devices may be attached to the AQ channel.

### AQ Channel Peripheral Device Addressing

The AQ channel operates so that the Q register of the computer contains the address of the peripheral device, and the data transfer occurs to/from the A register of the computer.

Format:




The following definitions apply for the controller levels shown in Figure 5-1:

- Equipment — The equipment contains logic shared by the stations. The equipment compares bits 10-07 of Q with the setting of its equipment number switch. If these settings match, the equipment responds.
- Station — The station contains logic unique for each unit (e.g., card reader controller). The station translates bit 6 and lower of Q as required, if the equipment code is correct, and responds to its station code.

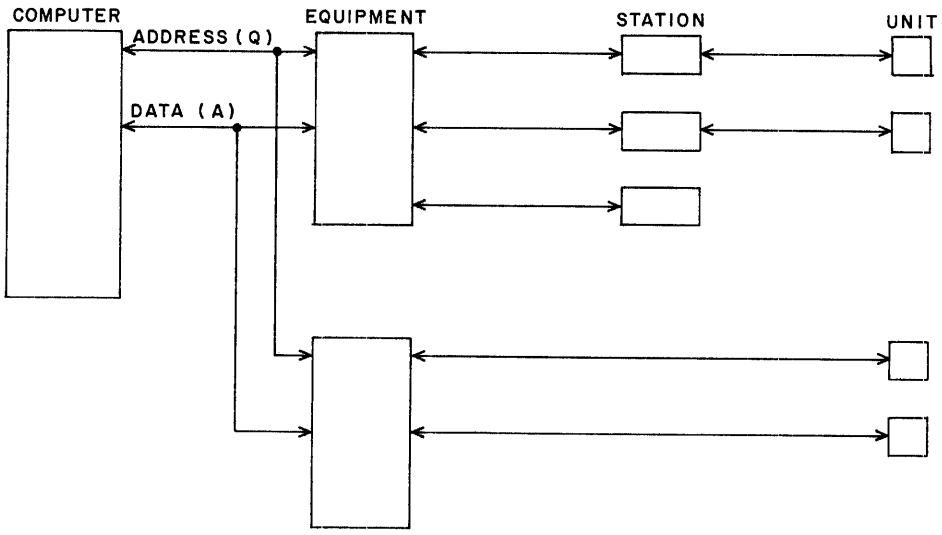



Figure 5-1. Controller Levels

- Units — Units usually refer to some physical hardware such as a magnetic tape handler, paper tape punch, etc. The units are controlled by a higher level controller and respond only to the controller. In some cases, the controller is the station; in other cases, the controller is the equipment. Different units on the controller are selected by a function which directs bits in A to change the selected unit.

W

Bits 11-15 of Q (=W) are reserved to address special devices called converters. These converters are devices which are never operated from the buffered data channel. Therefore, the buffered data channel is itself one of these converters.

$W = 00000_2$ is reserved for devices which can be operated from both the AQ and buffered channels. The lower 11 bits of Q must be coded in such a way as to distinguish between devices on the AQ channel (buffered or nonbuffered). When W = 00000, the W = 0 line to the external equipment is a 1.

Equipment Code

Bits 07-10 of Q select the desired equipment. Each equipment is assigned an equipment number (hexadecimal 0-F) via wiring or an Equipment Number switch. If the switch or wired setting matches bits 07-10 of Q, the equipment responds.

Station Code

The station code selects a station within an equipment. The station code may vary from 1 bit to 6 bits. Bit 6 of Q is the lowest station code bit, bit 5 of Q is the next higher, etc.

Director Code

The director code (bit 0) specifies the type of operation to be performed by the combination of the Q and A registers.

Command

The combination of the station code and director code are referred to as a command. The following commands are possible:

- Data Transfer When all bits of the command (Q6-Q0) are zero except for the station designator, a 16-bit input or output operation is performed via the A register, using either an Input to A or Output from A computer instruction (see Input on AQ Channel and Output on AQ Channel).
- Director Function When bit 00 of Q = 1, and the computer executes an Output from A instruction, and all station bits (if any assigned) are zero, bits in A control the functions of the station within the equipment. Assigned bits in A are:

| | | | |
|---|---|---|---|
| A0 | Clear Controller | A4 | Alarm Interrupt Request |
| A1 | Clear Interrupts | A5 | Start Motion |
| A2 | Data Interrupt Request | A6 | Stop Motion |
| A3 | End of Operation Interrupt Request | | |

  Refer to the associated external equipment reference manual for an exact defintion of the codes in A. Bits A7 through A11 are unassigned and can be used when necessary.
- Director Status When bit 00 of Q = 1, and the computer executes an Input to A instruction, and all station bits (if any assigned) are zero, the status of the equipment is loaded into the A register. When bit 00 of Q = 1, and the computer executes an Input to A instruction, and station bits are present, the status of the station is loaded into the A register.

  All devices must respond to status requests with a Reply since the status must always be available within 4 microseconds. If a Reject is received by the computer, it is an Internal Reject and indicate the device is not connected on the AQ channel.

  Additional bits may be used in conjunction with $Q_{00}$ so that levels of status may be selected, such as interrupt conditions, addresses, etc. The following codes for status apply for primary status returns (equipment status):

| | | | |
|---|---|---|---|
| A0 | Ready | A5 | Alarm |
| A1 | Busy | A6 | Lost Data |
| A2 | Interrupt | A7 | Protected |
| A3 | Data | A8 | Parity Error |
| A4 | End of Operation | A9-A15 | Unassigned |

  Again, refer to the associated external equipment reference manual for an exact definition of the status codes in A. Bits A9-A15 are unassigned and can be used when necessary.

Continue Bit

Bit 15 of Q is a continue bit used to speed addressing of devices requiring addresses greater than can be contained in Q. This works as follows:

- Address the device with Q15 = 0 and the remainder of Q set to select this device. The device is now connected.
- All succeeding addresses with Q15 = 1 are recognized by this device. Thus 15 bits of address or more are available in the Q register to this device.
- The next address with Q15 = 0 disconnects this device unless it is the address of this device.

### Output on AQ Channel

A single word is output from the A register whenever the computer executes the Output from A instruction (Q00 = 0). The presence of the output data is signified by the presence of a "1" on the write line. The peripheral equipment whose address is in the Q register must respond with a Reply or a Reject signal within 4 microseconds.

If no response occurs within 6 microseconds, the computer generates an Internal Reject and reinitiates execution of instructions. If a Reply is received by the computer, the next instruction executed is the one following the output instruction (P plus 1). If an external Reject is received, the next instruction executed is located at P plus 1 plus Δ, where Δ is the lowest 8 bits of the output instruction, of which the highest bit is a sign bit. If an Internal Reject is received, the next instruction executed is located at P plus Δ.

### Input on AQ Channel

A single word is input to the A register whenever the computer executes the Input to A instruction (Q00 = 0). The request for data by the computer is signified by a "1" signal on the read line. The peripheral device whose address is in the Q register responds with a Reply when data is available to the A register.

If no data is available, the peripheral device responds with a Reject. In either case, the peripheral device must respond with a Reject or a Reply within 4 microseconds. If no response is obtained in 6 microseconds, the computer generates an Internal Reject. Reply causes the computer to go to address P plus 1. (P is the address of the input instruction.) External Reject causes the computer to go to address P plus 1 plus Δ, where Δ is the lowest 8 bits of the input instruction, the highest of which is a sign bit. Internal Reject causes the computer to go to address P plus Δ.

TABLE 5-1. AQ CHANNEL SIGNALS

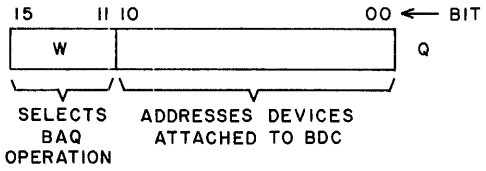
| SIGNAL | DEFINITION |
|---|---|
| Data Bits (To/from the A register) | These 16 lines are bidirectional and perform as follows:<br>1. In a Read (input) operation, the 16-bit data word is transferred from the peripheral device to the A register.<br>2. In a Write (output) operation, the 16-bit word is transferred from the A register to the peripheral device. |
| Address Bits | These 16 lines from the computer specify the address of the peripheral device with which data transfer is desired. The high-order bits uniquely specify the device. The low-order bits can specify functions of subdevices of that device. |
| Read | A "1" signal from the computer to the peripheral equipment, initiating an input transfer of one data word. |
| Write | A "1" signal from the computer to the peripheral equipment, initiating an output transfer of one data word. |
| Reply | A "1" signal, originating in the external equipment in response to a signal on the read or write line. The Reply drops after the Read or Write drops. |
| Reject | A "1" signal originating in the external equipment in response to a Read or Write signal, indicating that the operation cannot be performed. |
| Master Clear | A "1" signal sent by the computer, clearing the I/O channels and disconnecting the external equipment. |
| Program Protect | A "1" signal sent by the computer indicating that this data transfer was intitated by an I/O instruction whose program protect bit is set. It permits buffered I/O devices to write via direct access to computer storage into a program protected storage location. |
| Interrupt | A "1" signal sent to the computer on one of 16 lines when an external equipment reaches a predetermined condition. |
| Character Input | A "1" signal sent to the computer during input operations. If this signal is present during an Input to A, the lower 8-bit character is loaded into the lower 8 bits of A without disturbing the upper 8 bits of A. |
| W = 0 | This signal is present whenever bits 11, 12, 13, 14, and 15 of the address in the Q register are all zeros. |

## 1706 BUFFERED DATA CHANNEL

The 1706 Buffered Data Channel (called BDC) is a device on the AQ channel. It receives commands via the AQ channel and then transfers data via the external storage access of the computer. The external BDC interface is identical to the AQ interface. A maximum of eight devices may be connected to the BDC. Up to three 1706's may be connected to a 1700 Computer System. A 1716 may be used in place of any or all of the BDC 1706.

### BDC Addressing

The format of the address in the Q register which directs the BDC operation is:




The BDC is addressed via the upper 5 bits of the Q register which is denoted as W. W is written as two digits. The left digit is binary; the right digit is hexadecimal. The binary digit is always zero since it has to do with special devices connected on the AQ channel. Refer to Table 5-2 for definitions of addresses and operations for the BDC.

TABLE 5-2. 1706 ADDRESSES AND OPERATIONS

| W | | | COMPUTER OPERATION | |
|---|---|---|---|---|
| 1706 #3 | 1706 #2 | 1706 #1 | INPUT | OUTPUT |
| 0C | 07 | 02 | Direct input | Direct output |
| 0D | 08 | 03 | Terminate buffer: BDC current address | Function (See 1706 Table 5-3) |
| 0E | 09 | 04 | BDC status | Buffered output |
| 0F | 0A | 05 | BDC current address | Buffered input |

TABLE 5-3. BUFFERED DATA CHANNEL FUNCTIONS

| BIT IN A REGISTER | MEANING |
|---|---|
| A15 = 1 | Set condition for ones in bits A14 - A00 |
| A15 = 0 | Clear condition for ones in bits A14 - A00 |
| A14 - A01 | Not defined |
| A00 | Enable interrupt on BDC End of Operation |

## Output

### Direct Output

Whenever the computer executes an output instruction to a BDC with the W field of Q selecting a direct output, Q is stored in the BDC and presented to the devices attached to the BDC, and one 16-bit word is transferred from A to the selected device. This mode of operation is identical to that on the AQ channel. The continue bit (Q15) functions as described previously.

### Buffered Output on the 1706

A buffered output is initiated when the computer executes an output instruction and the W field of Q selects a buffered output. Q is stored in the BDC, then placed on the BDC channel, and the contents of A is transferred to the BDC. The contents of A specifies the first word address minus 1 (FWA-1) of the block to be transferred. The contents of location FWA-1 specifies the last word address plus one (LWA+1) of the block to be transferred. The BDC begins the data transfer by raising the Write signal to the device. The The device responds within 4 microseconds by raising the Reply signal to the computer. The BDC then advances to the next data word, repeating this cycle until the block of data has been transferred. If the BDC receives a Reject, it indefinitely repeats the transfer of the word until the word is accepted. The address is not reissued to storage. The BDC does not generate an Internal Reject. If an attempt is made to establish a buffered output when the BDC is busy, the BDC responds with a Reject.

## Input

### Direct Input

Whenever the computer executes an input instruction to a BDC with the W field of Q selecting direct input, Q is stored in the BDC and presented to the devices attached to the BDC, and one 16-bit word is transferred into the A register from the selected device. The word may be a data word or the status of the device connected to the BDC. This mode of operation is identical to that on the AQ channel. The continue bit (Q15) functions as described previously.

### Buffered Input on the 1706

A buffered input is initiated when the computer executes an output instruction and the W field of Q selects a buffered input. Q is stored in the BDC, then placed on the BDC channel, and the contents of A is transferred to the BDC. The contents of A specifies the location of the first word address minus one (FWA-1) of the block where data is to be stored. The contents of location FWA-1 specifies the last word address plus 1 (LWA+1) of this block. The BDC begins the data transfer by raising the Read signal to the device. The device responds within 4 microseconds by raising the Reply signal to the computer. The BDC then advances to the next data word and repeats this cycle until the block of data has been transferred.

If the BDC receives a Reject from the device, it repeats the transfer of the word indefinitely until the word is accepted. The BDC does not generate an Internal Reject. If an attempt is made to establish a buffered input when the BDC is busy, the BDC responds with a Reject.

## Status

There are two status associated with the BDC. When either one is specified by a computer input instruction, the appropriate data is entered into the computer A register by the BDC. The BDC always gives a Reply to either of its status within 4 microseconds. If a Reject occurs for either status, the BDC is not attached to the AQ channel.

BDC Status

This status shows the current operating conditions of the BDC. It is loaded into the A register during an input instruction when BDC status is selected. The bits are defined in Table 5-4.

Device Reply This bit, if set, means the peripheral device accepted the last word transfer attempted from the BDC.

Device Reject This bit, if set, means the peripheral device rejected the last word transfer attempted from the BDC.

Program Protect Fault A reference to computer storage caused a program protect fault.

End of Operation A buffer transfer input or output has been completed.

Interrupt An End of Operation interrupt is being sent to the computer from the BDC.

Busy This bit is set from the time the BDC accepts an output word from the computer, initiating a block transfer, until the block transfer is terminated, or during a direct operation.

Ready This bit is set when power is on.

Current Address

This status shows the address of the current word being transferred. It is loaded into the A register of the computer when the computer executes an input instruction with the W field of Q selecting terminate buffer or BDC Current Address.

## Interrupt

An interrupt to the computer is generated when the BDC terminates a buffered transfer (i.e., End of Operation). Interrupt is enabled by an output to the BDC with the W field of the Q register selecting a function to the BDC, bit 15 of A set, and bit 00 of A is set.

TABLE 5-4. 1706 STATUS BITS

| BIT | MEANING |
|---|---|
| 15-10 | Not used |
| 9 | Device reply |
| 8 | Device reject |
| 7 | Not used |
| 6 | Program protect fault |
| 5 | Not used |
| 4 | End of operation |
| 3 | Not used |
| 2 | Interrupt |
| 1 | Busy |
| 0 | Ready |

Interrupt on End of Operation is disabled by an output to the BDC with the W field of the Q register selecting a function to the BDC, bit 15 of A clear, and bit 00 of A set. An interrupt from the BDC is cleared by a function to the BDC with bit 00 of A set.

## 1716 COUPLING DATA CHANNEL

The 1716 is a Buffered Data Channel (1706) which can be shared by two 1700 Computer Systems and provides a means of direct data coupling between the two 1700 Computer Systems. The programming functions and operations are identical to the 1706 BDC, but additional functions and controls have been added to effectively provide the added capability.

### Addressing

A maximum of three 1716's may be connected to a 1700 Computer System. In any case, the sum total of 1706's and 1716's cannot be greater than three. Refer to Table 5-5 for definitions of addresses and operations for the 1716.

### Buffered Data Channel Mode

In this mode, either of the computers may initiate buffered data transfers to the peripheral devices connected to the buffered channel of the 1716.

In addition to this, either computer can cause the data transfer to be to or from either computer's core storage. The control of which computer's core storage is to be used is provided by the highest order bit in the addresses used to define the limits of the block to be transferred (i.e., FWA-1 and LWA+1). This bit, if set, means the "other" computer; if cleared, means "this" computer.

### Flags

Five flags are provided for use in controlling sharing of the BDC or direct transfer of data between computers. They may be set and cleared as shown in Table 5-6. Their state may be determined via the 1716 status word as defined in Table 5-7. Each computer has a mask associated with each flag. If the computer sets its mask bit, it is interrupted whenever the associated flag is set.

TABLE 5-5. 1716 ADDRESSES AND OPERATIONS

| W | | | COMPUTER OPERATIONS | |
|---|---|---|---|---|
| 1716 #3 | 1716 #2 | 1716 #1 | INPUT | OUTPUT |
| 0B | 06 | 01 | | Buffered transfer |
| 0C | 07 | 02 | Direct input | Direct output |
| 0D | 08 | 03 | Terminate buffer<br>1716 current address | Function<br>(See 1716 Table 5-6) |
| 0E | 09 | 04 | 1716 status | Buffered output |
| 0F | 0A | 05 | 1716 current address | Buffered input |

TABLE 5-6. 1716 FUNCTIONS

| BIT IN A REGISTER | MEANING |
|---|---|
| A15 { = 1 | Set condition for ones in bits A14 - A00. * |
| A15 { = 0 | Clear condition for ones in bits A14 - A00. * |
| A14 | Flag 4 |
| A13 | Flag 3 |
| A12 | Flag 2 |
| A11 | Flag 1 |
| A10 | Flag 0 |
| A09 | Mask 4 |
| A08 | Mask 3 |
| A07 | Mask 2 |
| A06 | Mask 1 |
| A05 | Mask 0 |
| A04 | Not defined |
| A03 | Not defined |
| A02 | Not defined |
| A01 | Not defined |
| A00 | Interrupt on BDC End of Operation* |

*Functions also used in the 1706

TABLE 5-7. 1716 STATUS BITS

| BIT | MEANING |
|---|---|
| 15 | Not used |
| 14 | Flag 4 |
| 13 | Flag 3 |
| 12 | Flag 2 |
| 11 | Flag 1 |
| 10 | Flag 0 |
| 9 | Device Reply * |
| 8 | Device Reject * |
| 7 | Not used |
| 6 | Program Protect Fault * |
| 5 | Not used |
| 4 | End of Operation * |
| 3 | Not used |
| 2 | Interrupt * |
| 1 | Busy * |
| 0 | Ready * |

*Status also used in the 1706

**Status**

The status bits for the 1716 are defined in Table 5-7. These status bits are common to both computers except for interrupt and end of operations. Each computer has its own status bits in these two cases.

**Buffered Transfer Mode**

In this mode, data is transferred directly to computer core storage via the 1716. The transfer is initiated by one computer executing an output instruction with the W field of Q selecting a buffered transfer (see Table 5-5). The transfer is accomplished by defining the bounds of the source data and the FWA-1 of the destination location. The output to the 1716 initiates the buffer transfer and sends via A the source data FWA-1 (SFWA-1). The contents of SFWA-1 is the source data LWA+1 (SLWA+1). The contents of SLWA+1 is the destination data FWA-1 (DFWA-1). Bit 16 of all these addresses refers to the "other" computer if it is set; "this" computer if not set. It is thus possible to make a block transfer within a single computer's core storage. All Flag, Mask, and Status functions of the 1716 are available in Buffered Transfer mode.

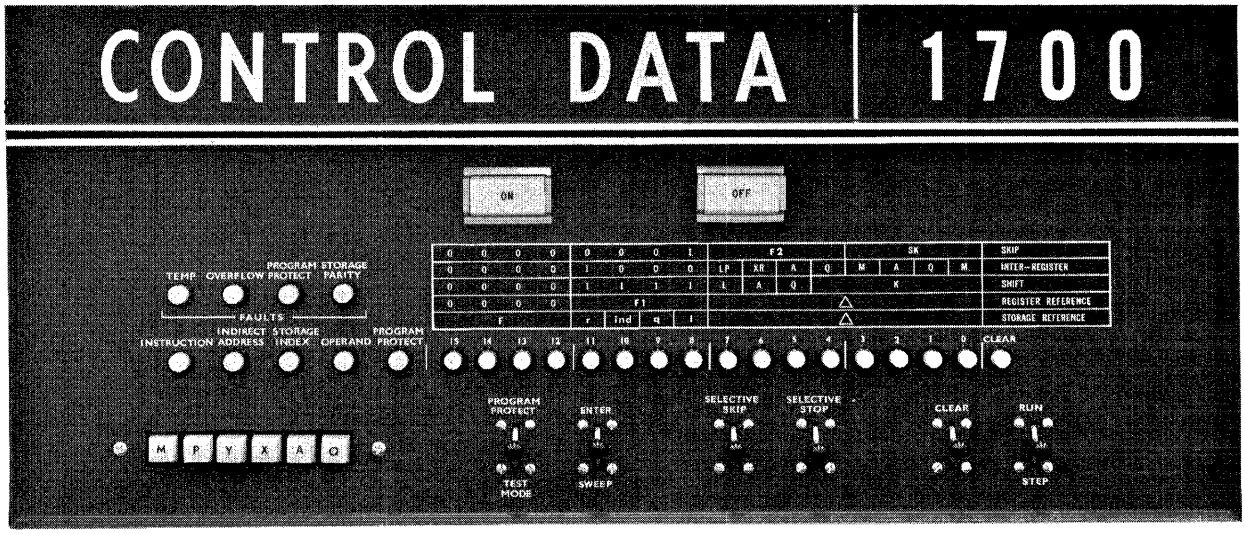
**Interrupt**

The 1716 has one interrupt line to each computer. The interrupts are generated from end of operation, if enabled, and from the flags if the associated mask is set. The End of Operation interrupt is sent to the computer that initiated the transfer.

The 1716 timing is identical to that on the AQ channel.

# MANUAL CONTROLS AND INDICATORS

This section discusses switches and indicators used to operate and maintain the computer. Figure 6-1 shows the computer console.




1554

Figure 6-1. Computer Console

## SWITCHES

**Master Clear**

This is a three-position key/lever switch. A Master Clear is executed whenever it is momentarily operated either up or down. The Master Clear returns the computer and peripheral devices to initial conditions.

**Run/Step**

This is a three-position key/lever switch. When the switch is momentarily placed in the up position, the computer begins program execution, starting with the instruction whose address is in the P register. The computer is stopped by momentarily placing the switch in the down position.

If the switch is repeatedly placed in the down position, the computer steps through the program, stopping after each storage reference. The significance of the storage reference just made is indicated by the instruction sequence indicators.

**Enter/Sweep**

This is a three-position key/lever switch, maintained in all positions. The center position is off.

Enter

Up position selects the Enter mode. In this mode, each step operation of the RUN/STEP switch stores the contents of the X register at the location specified by P plus 1 and then advances the P register by 1. The first step after a Master Clear or Clear P stores X at the location specified by P.

To store a few instructions in unprotected storage, proceed as follows:

1) Power on but computer stopped.
2) Operate CLEAR Switch.
3) Operate P Register Selector switch and Clear Register switch, in that order. Set desired address for instruction in P by use of indicator pushbuttons.
4) Set ENTER/SWEEP switch to ENTER.
5) Operate X Register Selector switch.
6) Operate Clear Register switch, then enter word to be stored by use of indicator pushbuttons.
7) Operate RUN/STEP switch to STEP one time. To store additional words in successive storage locations, repeat steps 6 and 7 until finished.

To change to a new sequence of addresses, start at step 2 for the first one, then repeat steps 6 and 7 for each successive word.

NOTE: A lighted indicator pushbutton indicates a "1", a dark one a "0".

Sweep

The down position selects the Sweep mode. In this mode, each step operation of the RUN/STEP switch reads the contents of the storage location whose address is in P plus 1 into the X register and then advances the P register by 1. The first step after a Master Clear or Clear P displays the location specified by P. Instructions are not executed.

**Selective Stop**

This is a three-position key/lever switch. The computer stops when it executes a Selective Stop instruction if this switch is in either the up or down position. Up position is maintained; down position is momentary.

**Selective Skip**

This is a three-position key/lever switch. The computer Selective Skip instruction is conditioned by this switch. This switch is off in the center position, on in the up or down position. The up position is maintained; the down position in momentary.

**Program Protect/ Test Mode**

This is a three-position key/lever switch, maintained in all positions. The center position is off.

Program Protect

The up position selects program protection.

Test Mode

The down position selects Test mode. When in Test mode, the computer executes the following sequence of events:

1) 20-usec Master Clear. This clears the P register and all other operational registers to zeros.
2) 100-usec program run starting from P = $0000_{16}$.
3) Return to 1) and repeat.

**Emergency Off**

Pressing this switch shuts down power for the entire system.

## INDICATORS

**Registers**

The X, A, Q, P, Y, and M registers are available for display and manual entry of values via a single set of switch/indicators. A six-position push-button switch called a REGISTER SELECTOR selects the register for display and entry.

To enter a value into a register, select the register via the REGISTER SELECTOR, push the REGISTER CLEAR pushbutton to clear that register, and then set bits via the 16 switch/indicator pushbuttons.

**Program Protect**

The PROGRAM PROTECT bit indicator displays the state of the program protect bit of the last storage location referenced by the computer.

**Faults**

There are five fault indicators. When lighted, the fault condition is present.

- HI TEMP The temperature inside the computer has exceeded safe operating limits.
- OVERFLOW An arithmetic register overflow has occurred.
- PROGRAM PROTECT A violation of the program protect system has been detected.
- STORAGE PARITY A parity error has been detected in an operand or instruction read from storage.
- TEMP WARNING The temperature of the incoming air is exceeding safe operating limits.

**Instruction Sequence Indicators**

When an instruction is being stepped, this group of four indicators describes the meaning of the storage reference just completed. The data of the storage reference (read or write) is in the X register. The four indicators and their meaning when lighted are:

- Instruction: The contents of the X register is an instruction.
- Indirect Address: The contents of the X register is the result of indirect addressing. The indirect address may also be another indirect address, hence, this indicator may remain lighted for several consecutive storage references.
- Storage Index: The contents of the X register is the value of the Storage Index register.
- Operand: The contents of the X register is the value of the operand either written into or read from storage.

# APPENDIX SECTION

# 1700 INSTRUCTION EXECUTION TIMES



STORAGE REFERENCE

| | INSTRUCTION | EXECUTION TIME (microseconds)* |
|---|---|---|
| LDA | Load A | 2.2 |
| STA | Store A | 2.2 |
| LDQ | Load Q | 2.2 |
| STQ | Store Q | 2.2 |
| ADD | Add A | 2.2 |
| SUB | Subtract | 2.2 |
| ADQ | Add Q | 2.2 |
| AND | AND with A | 2.2 |
| EOR | Exclusive OR with A | 2.2 |
| RAO | Replace Add One in Storage | 3.3 |
| MUI | Multiply Integer | 7.0 |
| JMP | Jump | 1.1 |
| RTJ | Return Jump | 2.2 |
| DVI | Divide Integer | 9.0 |
| SPA | Store A, Parity to A | 2.2 |

*Add 1.1 microsecond if Storage Index Register is used.
Add 1.1 microsecond for each level of Indirect Addressing.

## REGISTER REFERENCE

| | INSTRUCTION | EXECUTION TIME (microseconds) |
|---|---|---|
| SLS | Selective Stop | 1.1 |
| INP | Input to A | 1.1 min., 10 max. |
| OUT | Output from A | |
| ENA | Enter A | 1.1 |
| ENQ | Enter Q | 1.1 |
| INA | Increase A | 1.1 |
| INQ | Increase Q | 1.1 |
| ARS | A Right Shift | $1.1 + \left(\frac{\text{shift count}}{2}\right)(.2)$ |
| QRS | Q Right Shift | |
| ALS | A Left Shift | |
| QLS | Q Left Shift | |
| LRS | Long Right Shift | 1.1 + (shift count) (.2) |
| LLS | Long Left Shift | |
| NOP | No Operation | 1.1 |
| EIN | Enable Interrupt | 1.1 |
| IIN | Inhibit Interrupt | 1.1 |
| EXI | Exit Interrupt State | 2.2 |
| SPB | Set Program Protect | 2.2 |
| CPB | Clear Program Protect | 2.2 |

## REGISTER REFERENCE

| INTER-REGISTER | INSTRUCTION | EXECUTION TIME (usec) |
|---|---|---|
| SET | Set to Ones | 1.1 |
| CLR | Clear to Zero | |
| TRA | Transfer A | |
| TRM | Transfer M | |
| TRQ | Transfer Q | |
| TRB | Transfer Q + M | |
| TCA | Transfer Complement A | |
| TCM | Transfer Complement M | |
| TCQ | Transfer Complement Q | |
| TCB | Transfer Complement Q + M | |
| AAM | Transfer Arithmetic Sum A, M | |
| AAQ | Transfer Arithmetic Sum A, Q | |
| AAB | Transfer Arithmetic Sum A, Q + M | |
| EAM | Transfer Exclusive or A, M | |
| EAQ | Transfer Exclusive or A, Q | |
| EAB | Transfer Exclusive or A, Q + M | |
| LAM | Transfer Logical Product A, M | |
| LAQ | Transfer Logical Product A, Q | |
| LAB | Transfer Logical Product A, Q + M | |
| CAM | Transfer Complement Logical Product A, M | |
| CAQ | Transfer Complement Logical Product A, Q | |
| CAB | Transfer Complement Logical Product A, Q + M | |

## SKIPS

| | INSTRUCTION | EXECUTION TIME (microseconds) |
|---|---|---|
| SAZ | Skip if A = +0 | 1.1 |
| SAN | Skip if A ≠ +0 | |
| SAP | Skip if A = + | |
| SAM | Skip if A = - | |
| SQZ | Skip if Q = +0 | |
| SQN | Skip if Q ≠ +0 | |
| SQP | Skip if Q = + | |
| SQM | Skip if Q = - | |
| SWS | Skip if Switch Set | |
| SWN | Skip if Switch Not Set | |
| SOV | Skip on Overflow | |
| SNO | Skip on No Overflow | |
| SPE | Skip on Storage Parity Error | |
| SNP | Skip on No Storage Parity Error | |
| SPF | Skip on Program Protect Fault | |
| SNF | Skip on No Program Protect Fault | |

FUNCTION LISTING OF 1700 INSTRUCTIONS

| FUNCTION | MNEMONIC CODE | INSTRUCTION DESCRIPTION |
|---|---|---|
| Transfers | LDA | Load A (Storage Reference) |
| | STA | Store A (Storage Reference) |
| | LDQ | Load Q (Storage Reference) |
| | STQ | Store Q (Storage Reference) |
| | SPA | Store A, Parity to A (Storage Reference) |
| | ENA | Enter A (Register Reference) |
| | ENQ | Enter Q (Register Reference) |
| | TRA | Transfer A (Register Reference) |
| | TRM | Transfer M (Register Reference) |
| | TRQ | Transfer Q (Register Reference) |
| Arithmetic | ADD | Add A (Storage Reference) |
| | SUB | Subtract (Storage Reference) |
| | ADQ | Add Q (Storage Reference) |
| | RAO | Replace Add One in Storage (Storage Reference) |
| | MUI | Multiply Integer (Storage Reference) |
| | DVI | Divide Integer (Storage Reference) |
| | INA | Increase A (Register Reference) |
| | INQ | Increase Q (Register Reference) |
| | SET | Set to Ones (Register Reference) |
| | TRB | Transfer Q + M (Register Reference) |
| | AAM | Transfer Arithmetic Sum A, M (Register Reference) |
| | AAQ | Transfer Arithmetic Sum A, Q (Register Reference) |
| | AAB | Transfer Arithmetic Sum A; Q + M (Register Reference) |
| Logical | AND | AND with A (Storage Reference) |
| | EOR | Exclusive OR with A (Storage Reference) |
| | CLR | Clear to Zero (Register Reference) |
| | TCA | Transfer Complement A (Register Reference) |
| | TCM | Transfer Complement M (Register Reference) |
| | TCQ | Transfer Complement Q (Register Reference) |
| | TCB | Transfer Complement Q + M (Register Reference) |
| | EAM | Transfer Exclusive OR of A, M (Register Reference) |

FUNCTION LISTING OF 1700 INSTRUCTIONS (Cont'd)

| FUNCTION | MNEMONIC CODE | INSTRUCTION DESCRIPTION |
|---|---|---|
| Logical (Cont'd) | EAQ | Transfer Exclusive OR of A, Q (Register Reference) |
| | EAB | Transfer Exclusive OR of A, Q + M (Register Reference) |
| | LAM | Transfer Logical Product A, M (Register Reference) |
| | LAQ | Transfer Logical Product A, Q (Register Reference) |
| | LAB | Transfer Logical Product A, Q + M (Register Reference) |
| | CAM | Transfer Complement Logical Product A, M (Register Reference) |
| | CAQ | Transfer Complement Logical Product A, Q (Register Reference) |
| | CAB | Transfer Complement Logical Product A, Q + M (Register Reference) |
| Jumps & Stops | JMP | Jump (Storage Reference) |
| | RTJ | Return Jump (Storage Reference) |
| | SLS | Selective Stop (Register Reference) |
| | NOP | No Operation (Register Reference) |
| Decisions | SAZ | Skip if A = +0 (Register Reference) |
| | SAN | Skip if A ≠ +0 (Register Reference) |
| | SAP | Skip if A = + (Register Reference) |
| | SAM | Skip if A = - (Register Reference) |
| | SQZ | Skip if Q = +0 (Register Reference) |
| | SQN | Skip if Q ≠ +0 (Register Reference) |
| | SQP | Skip if Q = + (Register Reference) |
| | SQM | Skip if Q = - (Register Reference) |
| | SWS | Skip if Switch set (Register Reference) |
| | SWN | Skip if Switch not set (Register Reference) |
| | SOU | Skip on overflow (Register Reference) |
| | SNO | Skip on no overflow (Register Reference) |
| | SPE | Skip on Storage Parity Error (Register Reference) |
| | SNP | Skip on no Storage Parity Error (Register Reference) |
| | SPF | Skip on Program Protect Fault (Register Reference) |
| | SNF | Skip on no Program Protect Fault (Register Reference) |

FUNCTION LISTING OF 1700 INSTRUCTIONS (Cont'd)

| FUNCTION | MNEMONIC CODE | INSTRUCTION DESCRIPTION |
|---|---|---|
| Shifts | ARS | A Right Shift (Register Reference) |
| | QRS | Q Right Shift (Register Reference) |
| | ALS | A Left Shift (Register Reference) |
| | QLS | Q Left Shift (Register Reference) |
| | LRS | Long Right Shift (Register Reference) |
| | LLS | Long Left Shift (Register Reference) |
| Input/Output | INP | Input to A (Register Reference) |
| | | Output from A (Register Reference) |
| Interrupt | EIN | Enable Interrupt (Register Reference) |
| | IIN | Inhibit Interrupt (Register Reference |
| | EXI | Exit Interrupt State (Register Reference) |
| Program Protect | SPB | Set Program Protect (Register Reference) |
| | CPB | Clear Program Protect (Register Reference) |

# POWERS OF TWO



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

# OCTAL-DECIMAL INTEGER CONVERSION TABLE



| | 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 |

| 0000 | 0000 |
|---|---|
| to | to |
| 0777 | 0511 |
| (Octal) | (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 |
|---|---|---|---|---|---|---|---|---|
| 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 |

| 1000 | 0512 |
|---|---|
| to | to |
| 1777 | 1023 |
| (Octal) | (Decimal) |

# OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

| Octal | Decimal |
|---|---|
| 2000 | 1024 |
| to | to |
| 2777 | 1535 |
| (Octal) | (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 |
|---|---|---|---|---|---|---|---|---|
| 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 |
| 2100 | 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 | 1519 | 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 |

| Octal | Decimal |
|---|---|
| 3000 | 1536 |
| to | to |
| 3777 | 2047 |
| (Octal) | (Decimal) |

| | 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 |

# OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

| | 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 |

| 4000 | 2048 |
|---|---|
| to | to |
| 4777 | 2559 |
| (Octal) | (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 |
|---|---|---|---|---|---|---|---|---|
| 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 |

| 5000 | 2560 |
|---|---|
| to | to |
| 5777 | 3071 |
| (Octal) | (Decimal) |

# OCTAL-DECIMAL INTEGER CONVERSION TABLE (Cont'd)

| Octal | Decimal |
|---|---|
| 6000 | 3072 |
| to | to |
| 6777 | 3583 |
| (Octal) | (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 |
|---|---|---|---|---|---|---|---|---|
| 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 | 3132 | 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 |

| Octal | Decimal |
|---|---|
| 7000 | 3584 |
| to | to |
| 7777 | 4095 |
| (Octal) | (Decimal) |

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| 7000 | 3584 | 3585 | 3586 | 3587 | 3588 | 3589 | 3590 | 3591 |
| 7010 | 3592 | 3593 | 3594 | 3595 | 3496 | 3497 | 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 | 3758 | 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 |

# 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 | [illegible]34375 | .270 | .359375 | .370 | .484375 |
| .071 | .111328 | .171 | [illegible]36328 | .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 | .001098 | .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 |

# DECIMAL/BINARY POSITION TABLE



| Largest Decimal Integer | Decimal Digits Req'd* | Number of Binary Digits | Largest Decimal Fraction |
|---|---|---|---|
| 1 | | 1 | .5 |
| 3 | | 2 | .75 |
| 7 | | 3 | .875 |
| 15 | 1 | 4 | .937 5 |
| 31 | | 5 | .968 75 |
| 63 | | 6 | .984 375 |
| 127 | 2 | 7 | .992 187 5 |
| 255 | | 8 | .996 093 75 |
| 511 | | 9 | .998 046 875 |
| 1 023 | 3 | 10 | .999 023 437 5 |
| 2 047 | | 11 | .999 511 718 75 |
| 4 095 | | 12 | .999 755 859 375 |
| 8 191 | | 13 | .999 877 929 687 5 |
| 16 383 | 4 | 14 | .999 938 964 843 75 |
| 32 767 | | 15 | .999 969 482 421 875 |
| 65 535 | | 16 | .999 984 741 210 937 5 |
| 131 071 | 5 | 17 | .999 992 370 605 468 75 |
| 262 143 | | 18 | .999 996 185 302 734 375 |
| 524 287 | | 19 | .999 998 092 651 367 187 5 |
| 1 048 575 | 6 | 20 | .999 999 046 325 683 593 75 |
| 2 097 151 | | 21 | .999 999 523 162 841 796 875 |
| 4 194 303 | | 22 | .999 999 761 581 420 898 437 5 |
| 8 388 607 | | 23 | .999 999 880 790 710 449 218 75 |
| 16 777 215 | 7 | 24 | .999 999 940 395 355 244 609 375 |
| 33 554 431 | | 25 | .999 999 970 197 677 612 304 687 5 |
| 67 108 863 | | 26 | .999 999 985 098 838 806 152 343 75 |
| 134 217 727 | 8 | 27 | .999 999 992 549 419 403 076 171 875 |
| 268 435 455 | | 28 | .999 999 996 274 709 701 538 085 937 5 |
| 536 870 911 | | 29 | .999 999 998 137 354 850 769 042 968 75 |
| 1 073 741 823 | 9 | 30 | .999 999 999 068 677 425 384 521 484 375 |
| 2 147 483 647 | | 31 | .999 999 999 534 338 712 692 260 742 187 5 |
| 4 294 967 295 | | 32 | .999 999 999 767 169 356 346 130 371 093 75 |
| 8 589 934 591 | | 33 | .999 999 999 883 584 678 173 065 185 546 875 |
| 17 179 869 183 | 10 | 34 | .999 999 999 941 792 339 086 532 592 773 437 5 |
| 34 359 738 367 | | 35 | .999 999 999 970 896 169 543 266 296 386 718 75 |
| 68 719 476 735 | | 36 | .999 999 999 985 448 034 771 633 148 193 359 375 |
| 137 438 953 471 | 11 | 37 | .999 999 999 992 724 042 385 816 574 096 679 687 5 |
| 274 877 906 943 | | 38 | .999 999 999 996 362 021 192 908 287 048 339 843 75 |
| 549 755 813 887 | | 39 | .999 999 999 998 181 010 596 454 143 524 169 921 875 |
| 1 099 511 627 775 | 12 | 40 | .999 999 999 999 090 505 298 227 071 762 084 960 937 5 |
| 2 199 023 255 551 | | 41 | .999 999 999 999 545 252 649 113 535 881 042 480 468 75 |
| 4 398 046 511 103 | | 42 | .999 999 999 999 772 626 324 556 767 940 521 240 234 375 |
| 8 796 093 022 207 | | 43 | .999 999 999 999 886 313 162 278 383 970 260 620 117 187 5 |
| 17 592 186 044 415 | 13 | 44 | .999 999 999 999 943 156 581 139 191 985 130 310 058 593 75 |
| 35 184 372 088 831 | | 45 | .999 999 999 999 971 578 290 569 595 992 565 155 029 296 875 |
| 70 368 744 177 663 | | 46 | .999 999 999 999 985 789 145 284 797 996 282 577 514 648 437 5 |
| 140 737 488 355 327 | 14 | 47 | .999 999 999 999 992 894 572 642 398 998 141 288 757 324 218 75 |
| 281 474 976 710 655 | | 48 | .999 999 999 999 996 447 286 321 199 499 070 644 378 662 109 375 |
| 562 949 953 421 311 | | 49 | .999 999 999 999 998 223 643 160 599 749 535 322 189 331 054 687 5 |
| 1 125 899 906 842 623 | 15 | 50 | .999 999 999 999 999 111 821 580 299 874 767 661 094 665 527 343 75 |
| 2 251 799 813 685 247 | | 51 | .999 999 999 999 999 555 910 790 149 937 383 830 547 332 763 671 875 |
| 4 503 599 627 370 495 | | 52 | .999 999 999 999 999 777 955 395 074 968 691 915 273 666 381 835 937 5 |
| 9 007 199 254 740 991 | | 53 | .999 999 999 999 999 888 977 697 537 484 345 957 636 833 190 917 968 75 |
| 18 014 398 509 481 983 | 16 | 54 | .999 999 999 999 999 944 488 848 768 742 172 978 818 416 595 458 984 375 |
| 36 028 797 018 963 967 | | 55 | .999 999 999 999 999 972 244 424 384 371 086 489 409 208 297 729 492 187 5 |
| 72 057 594 037 927 935 | | 56 | .999 999 999 999 999 986 122 212 192 185 543 244 704 604 148 864 746 093 75 |
| 144 115 188 075 855 871 | 17 | 57 | .999 999 999 999 999 993 061 106 096 092 771 622 352 302 074 432 373 046 875 |
| 288 230 376 151 711 743 | | 58 | .999 999 999 999 999 996 530 553 048 046 385 811 176 151 037 216 186 523 437 5 |
| 576 460 752 303 423 487 | | 59 | .999 999 999 999 999 998 265 276 524 023 192 905 588 075 518 608 093 261 718 75 |
| 1 152 921 504 606 846 975 | 18 | 60 | .999 999 999 999 999 999 132 638 262 011 596 452 794 037 759 304 046 630 859 375 |

*Larger numbers within a digit group should be checked for exact number of decimal digits required.

Examples of use:

1. *Q.* What is the largest decimal value that can be expressed by 36 binary digits?
   *A.* 68,719,476,735.
2. *Q.* How many decimal digits will be required to express a 22-bit number?
   *A.* 7 decimal digits.

# CONSTANTS



| | | |
|---|---|---|
| $\pi$ | = | 3.14159 26535 89793 23846 26433 83279 50 |
| $\sqrt{3}$ | = | 1.732 050 807 569 |
| $\sqrt{10}$ | = | 3.162 277 660 1683 |
| e | = | 2.71828 18284 59045 23536 |
| ln 2 | = | 0.69314 71805 599453 |
| ln 10 | = | 2.30258 50929 94045 68402 |
| $\log_{10} 2$ | = | 0.30102 99956 63981 |
| $\log_{10} e$ | = | 0.43429 44819 03251 82765 |
| $\log_{10} \log_{10} e$ | = | 9.63778 43113 00537 |
| $\log_{10} \pi$ | = | 0.49714 98726 94133 85435 |
| 1 degree | = | 0.01745 32925 11943 radians |
| 1 radian | = | 57.29577 95131 degrees |
| $\log_{10}(5)$ | = | 0.69897 00043 36019 |
| 7! | = | 5040 |
| 8! | = | 40320 |
| 9! | = | 362,880 |
| 10! | = | 3,628,800 |
| 11! | = | 39,916,800 |
| 12! | = | 479,001,600 |
| 13! | = | 6,227,020,800 |
| 14! | = | 87,178,291,200 |
| 15! | = | 1,307,674,368,000 |
| 16! | = | 20,922,789,888,000 |
| $\frac{\pi}{180}$ | = | 0.01745 32925 19943 29576 92369 07684 9 |
| $\left(\frac{\pi}{2}\right)^2$ | = | 2.4674 01100 27233 96 |
| $\left(\frac{\pi}{2}\right)^3$ | = | 3.8757 84585 03747 74 |
| $\left(\frac{\pi}{2}\right)^4$ | = | 6.0880 68189 62515 20 |
| $\left(\frac{\pi}{2}\right)^5$ | = | 9.5631 15149 54004 49 |
| $\left(\frac{\pi}{2}\right)^6$ | = | 15.0217 06149 61413 07 |
| $\left(\frac{\pi}{2}\right)^7$ | = | 23.5960 40842 00618 62 |
| $\left(\frac{\pi}{2}\right)^8$ | = | 37 0645 72481 52567 57 |
| $\left(\frac{\pi}{2}\right)^9$ | = | 58.2208 97135 63712 59 |
| $\left(\frac{\pi}{2}\right)^{10}$ | = | 91.4531 71363 36231 53 |
| $\left(\frac{\pi}{2}\right)^{11}$ | = | 143.6543 05651 31374 95 |
| $\left(\frac{\pi}{2}\right)^{12}$ | = | 225.6516 55645 350 |
| $\left(\frac{\pi}{2}\right)^{13}$ | = | 354.4527 91822 91051 47 |
| $\left(\frac{\pi}{2}\right)^{14}$ | = | 556.7731 43417 624 |

| | | |
|---|---|---|
| $\pi^2$ | = | 9.86960 44010 89358 61883 43909 9988 |
| $2\pi^2$ | = | 19.73920 88021 78717 23766 87819 9976 |
| $3\pi^2$ | = | 29.60881 32032 68075 85680 31729 9964 |
| $4\pi^2$ | = | 39.47841 76043 57434 47533 75639 9952 |
| $5\pi^2$ | = | 49.34802 20054 46793 09417 19549 9940 |
| $6\pi^2$ | = | 59.21762 64065 36151 71300 63459 9928 |
| $7\pi^2$ | = | 69.08723 08076 25510 33184 07369 9916 |
| $8\pi^2$ | = | 78.95683 52087 14868 95067 51279 9904 |
| $9\pi^2$ | = | 88.82643 96098 04227 56950 95189 9892 |

| | | |
|---|---|---|
| $\sqrt{2}$ | = | 1.414 213 562 373 095 048 801 688 |
| $1 + \sqrt{2}$ | = | 2.414 213 562 373 095 048 801 688 |
| $(1 + \sqrt{2})^2$ | = | 5.828 427 124 746 18 |
| $(1 + \sqrt{2})^4$ | = | 33.970 562 748 477 08 |
| $(1 + \sqrt{2})^6$ | = | 197.994 949 366 116 30 |
| $(1 + \sqrt{2})^8$ | = | 1153.999 133 448 220 72 |
| $(1 + \sqrt{2})^{10}$ | = | 6725.999 851 323 208 02 |
| $(1 + \sqrt{2})^{12}$ | = | 39201.999 974 491 027 40 |
| $(1 + \sqrt{2})^{14}$ | = | 228485.999 995 622 956 38 |
| $(1 + \sqrt{2})^{16}$ | = | 1331713.999 999 246 711 |
| $(1 + \sqrt{2})^{18}$ | = | 7761797.999 999 884 751 |

| | | |
|---|---|---|
| Sin .5 | = | 0.47942 55386 04203 |
| Cos .5 | = | 0.87758 25618 90373 |
| Tan .5 | = | 0.54630 24898 43790 |
| Sin 1 | = | 0.84147 09848 07896 |
| Cos 1 | = | 0.54030 23058 68140 |
| Tan 1 | = | 1.55740 77246 5490 |
| Sin 1.5 | = | 0.99749 49866 04054 |
| Cos 1.5 | = | 0.07073 72016 67708 |
| Tan 1.5 | = | 14.10141 99471 707 |

# HEXADECIMAL ADDITION TABLE



| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 |
| 3 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 |
| 4 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 |
| 5 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 |
| 6 | 6 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 |
| 7 | 7 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| 8 | 8 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 |
| 9 | 9 | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| A | A | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| B | B | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A |
| C | C | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B |
| D | D | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C |
| E | E | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D |
| F | F | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E |

CUT ALONG LINE

## COMMENT SHEET

CONTROL DATA 1700 Computer System
Computer Reference Manual
Pub. No. 60153100

**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

FOLD

FOLD

CUT ALONG LINE

STAPLE

STAPLE

**CONTROL DATA SALES OFFICES**

ALAMOGORDO • ALBUQUERQUE • ATLANTA • BILLINGS • BOSTON • CAPE CANAVERAL • CHICAGO • CINCINNATI • CLEVELAND • COLORADO SPRINGS DALLAS • DAYTON • DENVER • DETROIT • DOWNEY, CALIFORNIA • GREENSBORO, NORTH CAROLINA • HONOLULU • HOUSTON • HUNTSVILLE • MIAMI MONTEREY, CALIFORNIA • INDIANAPOLIS • 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 • SANTA BARBARA • SAN FRANCISCO SEATTLE • ST. LOUIS • TULSA • WASHINGTON, D. C.

AMSTERDAM • ATHENS • BOMBAY • CANBERRA • DUSSELDORF • FRANKFURT • HAMBURG • JOHANNESBURG • LONDON • LUCERNE • MELBOURNE MEXICO CITY • MILAN • MONTREAL • MUNICH • OSLO • OTTAWA • PARIS TEL AVIV • STOCKHOLM • STUTTGART • SYDNEY • TOKYO (C. ITOH ELECTRONIC COMPUTING SERVICE CO., LTD.) • TORONTO • ZURICH

CONTROL DATA CORPORATION

8100 34th AVE. SO., MINNEAPOLIS, MINN. 55440

Pub. No. 60153100

Litho in U. S. A.