From ASI's continuing program of new product development...

ASI

ADVANCE SERIES 6020 REFERENCE MANUAL

Advanced Scientific Instruments / 8001 Bloomington Freeway / Minneapolis, Minnesota 55420

ASI

REFERENCE MANUAL FOR THE *ADVANCE* Series 6020 GENERAL PURPOSE COMPUTER

ADVANCED SCIENTIFIC INSTRUMENTS

Division of Electro-Mechanical Research, Inc.

TABLE OF CONTENTS

SECTION 1. INTRODUCTION

General 1-1
Central Processor 1-2
Principle Registers 1-2
Memory Section 1-2
Arithmetic and Control Section 1-2
Input/Output Section 1-3
Word Format 1-4
Peripheral Equipment 1-5

SECTION 2. ARITHMETIC

General 2-1
Binary Arithmetic 2-1
Signed Magnitude Arithmetic 2-2
Fixed Point Numbers 2-2
Floating Point Numbers 2-2
Floating Point Arithmetic 2-3
Conversion of Fixed Point Numbers to Floating Point Numbers 2-3
Conversion of Floating Point Numbers to Fixed Point Numbers 2-4

SECTION 3. INSTRUCTION REPERTOIRE

General 3-1
Instruction Operation 3-2
- Data Transfer and Arithmetic Operations 3-2
- Register Change Instructions 3-5
- Logical Operations 3-7
- Shift Operations 3-7
- Index Operations 3-9
- Jump and Skip Operations 3-10
- Trap, Flag, and Flag Indicator Light Operations 3-12
- Input/Output Operations 3-13
- No Operation 3-15
- Programmed Instruction 3-15

SECTION 4. INPUT/OUTPUT

Instructing External Devices . . . 4-1

Program Interrupts . . . 4-3
- Standard Interrupts . . . 4-3
- Traps . . . 4-3
- Scanner . . . 4-3
- Priority Interrupts . . . 4-4

Buffered Input/Output . . . 4-4
- Input/Output Channel . . . 4-4
- Assembly Register Instruction . . . 4-4
- Data Flow . . . 4-5
- Character Transfer . . . 4-5
- Word Transfer . . . 4-5
- Field Transfer . . . 4-5
- Cyclic Transfer . . . 4-6

Direct Communication . . . 4-7

SECTION 5. OPERATOR'S CONSOLE

Display Register . . . 5-1

Registers T and S . . . 5-1

Indicators . . . 5-1

Controls . . . 5-2

APPENDIX

List of Instructions . . . A-1
- A. Instructions Arranged by Octal Codes . . . A-1
- B. Instructions Arranged by Mnemonic Codes . . . A-3

Commonly Used Constants . . . A-7
- Factorials . . . A-7
- Powers of 10_{10} . . . A-7
- Miscellaneous Constants . . . A-8
- Reciprocals . . . A-9
- Negative Powers of 2 and 8 . . . A-10
- Positive Powers of 2 and 8 . . . A-11

SECTION ONE INTRODUCTION

Advanced Scientific Instruments has met the challenge presented by the scientific and engineering field with the ADVANCE Series computers. This new family of computers highlights upward program compatibility; that is, programs written for one computer will operate without modification on all other computers above it in the Series. Peripheral equipment is also upward compatible.

The foundation of the ADVANCE Series is the 6020. This low-cost system for the small scale user is ideally suited to a wide range of applications, including scientific and engineering computations, on-line systems control, and data reduction. Expansion within the 6020 of magnetic core memory size and input/output channels, and external expansion with a variety of peripheral equipment keynote this versatile computer.

Hardware

The ADVANCE Series 6020 features a short memory cycle time of 1.90 microseconds; a new, larger, and more powerful instruction repertoire; and a large variety of optional input/output channels. Other outstanding features include a 24-bit memory word plus a parity bit for automatic parity checking on all memory data, double precision hardware, a console typewriter as a standard feature, indirect addressing, indexing, convenient subroutine access including sixteen programmed instructions that provide one instruction access to subroutines and return, sixteen individual levels of priority interrupt, floating-point operation, not to mention rapid instruction execution such as a 3.8 microsecond add and a 11.4 microsecond double-precision add. Standard input/output capabilities include a buffered Character Assembly Input/Output Channel and provisions for up to seven additional buffered I/O channels of various types as well as a programmed input/output channel which permits convenient program handling of words or character information from external devices. Character Assembly, Field Length, Complete Computer Word, and Cyclic are among the buffered I/O channels offered. External device operations with buffered channels require program attention for initiation only. Up to 16 external devices may be connected to each channel, to a system maximum of 64 devices.

Installation of the 6020 Central Processor requires no special temperature and humidity controls, no special power requirements, and no costly false flooring for cable runs. A complete array of ADVANCE Series peripheral equipments are available to increase the efficiency and usefulness of the 6020.

Software

ALL ADVANCE Series Computing Systems include a new and distinctive software package. Based on programming systems thoroughly tested and proven on earlier ASI computers, the ADVANCE Series software package features a one-pass FORTRAN II Compiler, a one-pass Symbolic Assembler, a Monitor program which allows for the centralized control of program operation, Mathematical Subroutines, Utility Routines, and Diagnostic Routines.

Support

ASI is proud of the services offered by the new Support Program for ADVANCE Series customers. As important as hardware and software, the Support Program brings the ADVANCE Series full-circle. Not only does the Support Program provide assistance before, during, and after purchase, but covers the design of the complete system including any special interfacing required. It is the aim of the Support Program to be complete as possible; to help the customer in every way.

CENTRAL PROCESSOR

The ADVANCE Series 6020 Central Processor may be considered in three main sections: Memory, Arithmetic and Control, and Input/Output. The Memory Section provides fast access storage for data and instructions. In the Arithmetic and Control Section; arithmetic, logical, and shifting operations are performed in the Arithmetic portion. The Control portion contains logic for controlling and sequencing all of the events that occur in the Central Processor. The Input/Output Section contains the logic for instructing external devices, scanning for external interrupts, and data communication with external devices. In addition, an operator's console provides a convenient means of manual control and information display.

PRINCIPLE REGISTERS

MEMORY ADDRESS REGISTERS

M — Memory address register (15 bits). M supplies the address to the memory during a memory reference.

S — Sequence address register (15 bits). S contains the address of the next instruction to be performed.

T — Temporary Storage Register (15 bits). T holds the memory address temporarily during input/output memory references. T also serves as a display and entry address register for the operator's console.

ARITHMETIC REGISTER

A — Accumulator (24 bits). A is the principle register in most arithmetic, logical, and shift instructions.

E — Extension register (24 bits). E is involved in many arithmetic, logical, and shift instructions. A and E together form a 48-bit register for double precision arithmetic with E usually containing the least significant bits.

C — Console Display Register (24 bits). C is the word display register on the operator's console and also serves as an auxiliary arithmetic register.

BUFFERED INPUT/OUTPUT CHANNEL REGISTERS

B — Buffer Register (24 bits). B is the assembly register in a buffered input/output channel. Character information going to or from external devices is assembled or disassembled in Register B. It communicates with the memory in 24-bit words.

BM — Buffer Memory Address Register (15 bits). At the start of a data communication, BM is loaded with the starting address of the data in memory. BM is incremented by one on each memory reference until the end of the data communication. Therefore, BM holds the address of the next memory location with which a buffered input/output channel will communicate.

BL — Buffer Limit Address Register (15 bits). BL holds the limit address (the last allowable address + 1) for a block of buffered data.

INDEX REGISTERS

X1, X2, X3 — Index Registers (15 bits). These registers are contained in Memory Locations 115, 116, and 117.

MEMORY SECTION

The standard 6020 magnetic core memory contains 4,096 24-bit words (plus one parity bit) with a complete read-write cycle time of 1.90 microseconds. The memory is expandable to 32,768 24-bit words, all of which are directly addressable. Parity is automatically generated on all write operations and checked on all read operations. Memory parity failure leads to the second highest priority interrupt (the highest being power failure) if the parity fail trap is armed. The resulting interrupt routine will respond to the parity failure according to the dictates of the running program.

ARITHMETIC AND CONTROL SECTION

The Arithmetic and Control Section is shown in Figure 1-1. The arithmetic portion performs all arithmetic, logical, and shift operations, while the timing source and cycle counter, the address registers and address modification logic, and the instruction and sequencing logic are contained in the control portion.

The timing source generates two-phase clock pulses with a repetition rate of 525 kc. These pulses are counted in the P counter which supplies time sequencing for the instruction logic.

The three index registers are in memory in Locations 115 through 117. Any of these index registers may be used to modify operand addresses. The addition of the index to the basic operand address makes use of the arithmetic adder with the result being transferred to M from the arithmetic portion.

Multiple indirect addressing with indexing at each step may be performed in the 6020 Central Processor.

The instruction and sequencing logic makes use of timing information from the P counter, instruction codes from the instruction register (Register L), and information from other flip-flops in the system. It generates signals to the various transfer gates and sets flip-flops at times that are appropriate for the instruction that is being performed.

Six flags which may be set by the running program are included in the control portion. Two of these flags operate the operator indicator lights on the console, and may be used to draw the operator's attention to special conditions.

INPUT-OUTPUT SECTION

The ADVANCE Series 6020 Input/Output Section has a buffered character assembling channel. Once initiated by means of its EXD instruction, external device operations are controlled by logic associated with the external device. When an external device has completed an operation, it may initiate a program interrupt if it had previously been instructed to do so.

Up to eight buffered input/output channels of the following types may be attached to the 6020:

1. Character Assembling Channel.
2. Variable Field Channel
3. Parallel Word and Character Assembling Channel
4. Cyclic (continuous) Parallel Word Channel

Any of these channels may be used for input/output and may be time-shared by up to 16 external devices per channel. Buffered channels, which have priority over the central processor for memory reference, require one memory cycle time from the running program for each memory reference. Buffered communication may be variable block length and may use any part of memory that is specified by the program. The maximum character transfer rate of a buffered channel is 180 kc, while the maximum rate for parallel word

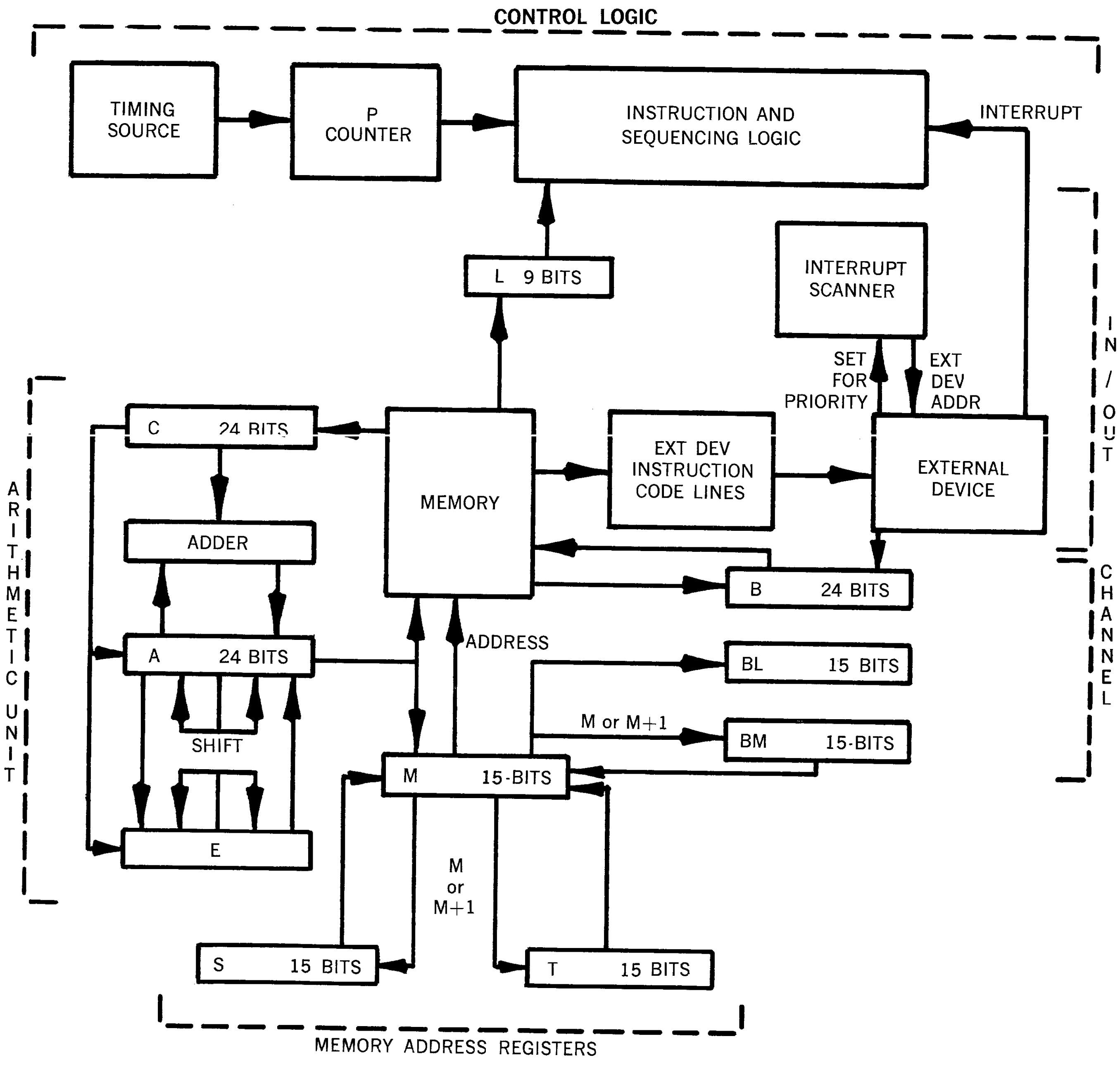

Figure 1-1. 6020 Block Diagram

communication is 130 kc. The 6020 Central Processor can handle a total buffered input/output word rate of 525 kc.

Up to 64 non-priority program interrupts are available on the 6020. Each interrupt will lead to a unique interrupt routine associated with the particular interrupt event. At the completion of an interrupt routine, control may be returned to the running program at the point of interruption or transferred to a new program. These interrupts may be used to notify the processor of the completion of some external operation, the failure of some external operation, or the occurrence of some other significant external event. The processor may recognize or ignore these interrupts by setting the external device interrupt to the allow or disallow condition, respectively.

Up to 16 levels of optional priority interrupts are available on the 6020. Any priority interrupt takes precedence over a lower priority interrupt and may interrupt any lower priority program. Each level of priority interrupt may be separately allowed or disallowed by the program and each leads to a unique interrupt routine.

As an option, the 6020 may be provided with a programmed input/output channel which sends information to or from the accumulator (Register A) by programmed instructions. The instruction specifies which device is to communicate; thus, the time-sharing of the programmed input/output lines is controlled directly by the program.

OPERATOR'S CONSOLE

The 6020 Central Processor has an operator's console with six sense switches, two program-controlled operator indicator lights, convenient display and entry features, a continuous or one-instruction mode switch, a preset switch for loading card or paper tape information simply, and many other features. This console is more thoroughly described in Section 5.

CONSOLE TYPEWRITER

The console typewriter is a standard feature of the 6020. The primary function of the typewriter is to monitor system and program operations. Such system conditions as ADD OVERFLOW, EXPONENT OVERFLOW, etc. and program conditions as SYNTAX ERROR, SYMBOL LENGTH, INTEGER SIZE, etc. are brought to the operator's attention via the typewriter. The typewriter also may be programmed to request information from the operator.

The typewriter also may be used to enter programs and data into the central processor and to type out the results in lieu of other peripheral equipment specifically designed for these functions.

WORD FORMAT

The 6020 computer word contains 24 bits which are numbered 1 to 24 from right to left. There are four types of computer words: the Instruction Word, the Index Word, the Control Word, and the Data Word. The Control Word is discussed in Section 4, Input/Output.

Instruction Word Format

F	I	X	Y
24 19	18	17 16	15 1
Operation Code	Indirect Address	Index Address	Base Operand Address

Operation Code (Bits 24-19). The Operation Code (F), which is the six highest order bits of the Instruction Word, tells the computer what operation is to be performed. The Operation Code of each Instruction Word is interpreted and performed separately by the computer.

In certain instructions, the Base Operand Address is treated as part of the Operation Code. This permits many more instructions to be specified than can be uniquely identified in the 6-bit Operation Code of each instruction.

Indirect Address (Bit 18). The Indirect Address Designator (I) specifies whether the Base Operand Address is to be used as the address of the operand (direct) or as the address of the memory location where the address of the operand will be found (indirect). The Indirect Address Designator indicates an indirect address if it is a "1."

Successive indirect addressing is possible as long as Bit 18 is a "1" in each Instruction Word referenced in memory. The last word to be referenced must have a "0" in the Bit 18 location, as the address it references is the address of the operand. Modification of the addresses through indexing (see following paragraph) may take place at each step when successive indirect addressing is performed.

Index Address (Bits 17-16). The Index Address (X) portion of the Instruction Word tells the computer whether the Base Operand Address of the instruction is to be modified or not; and, if the Base Operand Address is to be modified, where the modifying word will be found. The Index Address uses Bits 17 and 16 of the instruction Word. If they are both "0's," no modification of the Base Operand Address will take place. If either or both of the Index Address bits are "1's," they indicate a modification of the address is required and also specify the address of the modifying word. There are three index registers on the 6020 and they occupy actual memory locations in core storage.

Base Operand Address (Bits 15-1). The Base Operand Address (Y) is the lowest order 15 bits of the Instruction Word and is the address of the operand that is to be used for this instruction. The Base Operand Address may be modified as described in the two preceding paragraphs of this section.

Index Word Format

(Zeros) (9 bits)	Index Base (15 bits)
24 16	15 1

Index Words are stored in memory locations in the 6020. When the Index Address of the Instruction Word is 0, no indexing is specified. However, when the Index Address is a 1, 2, or 3, the Index Words in Memory Locations 115, 116, and 117, respectively, are referenced.

When the Index Address in an Instruction Word exceeds zero, the Index Word is extracted from memory and the contents of the Index Base (Bits 15-1) are added to the Base Operand Address of the Instruction Word.

Indexing with the 6020 is carried out in the two's complement system to eliminate the possibility of minus zero (see Section 2). When referring to an Index Word, it must be thought of as an absolute number that has no sign. If it is desired to use an Index Word to decrease the relative position of the specified Base Operand Address, the contents of the Index Address must be the two's complement of the number of positions the address is to be decreased. The two's complement of a number is formed by adding one to the one's complement of that number.

Data Word Format (numeric)

Sign (1 bit)	Magnitude (23 bits)
24	23 — 1

Numbers within the central processor are represented in fixed-point, binary, signed magnitude form. Bit 24 of the Data Word is the sign bit of the number; the remaining bits are magnitude bits. The sign bit is a "0" for positive numbers and a "1" for negative numbers. When examining a number for its absolute value, it must be remembered that this value is merely the number presented by Bits 23-1 with Sign Bit 24 ignored. (Section 2 of this publication provides a complete description of the arithmetic used by the 6020.)

Data Word Format (alphanumeric)

Character 1 (6 bits)	Character 2 (6 bits)	Character 3 (6 bits)	Character 4 (6 bits)
24 — 19	18 — 13	12 — 7	6 — 1

Four alphanumeric 6-bit characters are contained in each alphanumeric data word. The most significant character occupies the highest order bits of the word.

PERIPHERAL EQUIPMENT

Peripheral equipments increase the versatility, flexibility, and efficiency of the 6020 Computer System. The resulting increased usefulness means an even further reduction in the already low cost-per-answer ratio.

Line Printer. The Line Printer prints 400 lines/minute alphanumeric, 120 columns wide with the standard 48-character drum.

Magnetic Tape Units. Magnetic Tape Units are available to provide either low or high density operation. IBM compatibility permits the use of tapes written in this accepted format.

Punch Card Control Units. The Punch Card Control Units contain all logic, control, and power electronics to make card formats compatible with 6020 formats.

Paper Tape Unit. The Paper Tape Unit consists of a separate high speed reader and a punch.

Digital Incremental Plotters. Digital Incremental Plotters are available in four models that offer varied paper size, speed, and plotting increments.

SECTION TWO
ARITHMETIC

ARITHMETIC SYSTEMS

Since the ADVANCE Series 6020 utilizes bistable devices in its counters, registers, and pulse storing components, its "language" is the binary number system. As implied, the binary number system has only two symbols: "1" and "0." In all other respects, it is identical to the familiar decimal system.

Consider one more number system. The octal system is based on 8 unique symbols (0 through 7). Figure 2-1 shows that a binary 3-bit register can be utilized completely to represent all 8 unique symbols of the octal system. Because of the compatibility of the binary and octal systems and for ease of recognition when discussing the 6020, all numbers are displayed in octal form. While it is still somewhat difficult to comprehend the decimal equivalent of an octal number, especially a large one, it is considerably easier to work with than with its binary equivalent. For example, the memory location $0,010,100,011,101_2$ is much more meaningful when it is expressed as 2435_8.

BINARY ARITHMETIC

There are only four possible combinations for addition in the binary number system. They are:

(1) $0 + 0 = 0$
(2) $0 + 1 = 1$
(3) $1 + 0 = 1$
(4) $1 + 1 = 10$

With these combinations in mind, the following two binary numbers may be added:

$$\begin{array}{r} 11011010 \\ +\ 01011001 \\ \hline 100110011 \end{array}$$

In subtraction, there are also only four possible combinations:

(1) $1 - 1 = 0$
(2) $1 - 0 = 1$
(3) $0 - 1 = 1$ (Actually $10 - 1 = 1$, the "1" of the "10" is borrowed from the adjacent column to the left, as with subtraction in the decimal system.)
(4) $0 - 0 = 0$

With these combinations in mind, the following two binary numbers may be subtracted:

$$\begin{array}{r} 11011010 \\ -\ 01011001 \\ \hline 10000001 \end{array}$$

Decimal Number	Binary Number	Octal Number
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	10
9	1001	11
10	1010	12
11	1011	13
12	1100	14
13	1101	15
14	1110	16
15	1111	17
16	10000	20

Figure 2-1. Comparison of Binary, Octal, and Decimal Systems.

SIGNED MAGNITUDE ARITHMETIC

The 6020 is a signed magnitude computer; that is, all the arithmetic operations are accomplished by the process of addition and subtraction of magnitudes. Since multiplication and division can be broken down into a series of additions and subtractions, respectively, the 6020 will perform these operations as well.

Operations with signed magnitude numbers are identical with algebraic addition using a pencil and paper. Consider the two rules for the addition of signed quantities:

(1) If the signs are alike, add the magnitudes of the signed numbers. The sum will take the sign of the numbers.

$$(+24) + (+12) = +36$$
$$(-24) + (-12) = -36$$

(2) If the signs are not alike, subtract the smaller magnitude from the larger of the signed numbers. The difference will take the sign of the larger magnitude.

$$(+24) + (-12) = +12$$
$$(-24) + (+12) = -12$$

In the case of subtraction of signed quantities, one rule applies:

(1) Change the sign of the subtrahend and add it to the minuend, following the rules of algebraic addition.

$$(+24) - (+8) = (+24) + (-8) = +16$$
$$(-24) - (-8) = (-24) + (+8) = -16$$
$$(+24) - (-8) = (+24) + (+8) = +32$$
$$(-24) - (+8) = (-24) + (-8) = -32$$

Signed magnitude operations are the same in any number system, and are just as valid for the binary system as the decimal.

FIXED POINT NUMBERS

The 6020 is a fixed-point computer, which means the binary point of each operand is always in the same location in the registers. A good example of a fixed point "computer" is the office adding machine which has a fixed decimal point located between the second and third column from the right (0,000,000,000.00) in order to show one-hundredths of the dollar.

The binary point in the 6020 is located between the 23rd and 24th bits of the operand. Since the 24th bit is the sign bit, the operand is always a fraction. The results of all arithmetic operations in the 6020 must be less than 1. The product of two fractions will always be less than 1, but the addition, subtraction, or division of two fractions may yield a result greater than 1. Whenever the result of an arithmetic operation is 1 or greater, overflow occurs, which is an illegal condition in the 6020.

Two types of overflow may occur in the 6020; add and divide. Overflow may also occur in certain algebraic subtraction operations such as a negative quantity subtracted from a positive quantity. In reality, of course, this is the same as an additive operation.

As was mentioned in Section I, Introduction, the operand is positive when the 24th bit is "0" and negative when the 24th bit is "1." For all positive numbers (Bit 24 = 0), the octal digit will be 0, 1, 2, or 3; for all negative numbers, the octal digit will be 4, 5, 6, or 7. Therefore, positive numbers range from 00000000_8 to 37777777_8 and negative numbers from 40000000_8 to 77777777_8.

The two numbers 37246012_8 and 21422531_8 are positive numbers since their most significant digits are less than 4. The sum of these two numbers is:

$$\begin{array}{r} 37246012_8 \\ (+)\ 21422531_8 \\ \hline 60670543_8 \end{array}$$

The most significant digit of the sum indicates that the sum is negative, since it is greater than 3. Obviously, the sum of two positive numbers cannot be negative. What has happened is that the sum of the two numbers has exceeded the register length of the 6020 and an add overflow has occurred. Correspondingly, if the sum of two negative numbers is positive, an add overflow has occurred. This leads to a simple generalization that states: If the sign of the result is different from both the sign of the augend and addend, then an add overflow has occurred. Similarly, for subtraction: If the sign of the result is the same as the sign of the subtrahend and different than the sign of the minuend, overflow has occurred.

When dividing in the 6020, the dividend is placed in Registers A and E, the most significant bits being in A. The quotient appears in Register E and the remainder in Register A. If on division, one or more of the most significant bits of the quotient cannot be contained in Register E, a divide overflow condition exists. Both the quotient and remainder will be in error.

For example, dividing the number $302{,}560{,}000_8$ by 2_8:

```
                      1  41270000
                    ______________
         00000002 ) 00000003  02560000
                           A         E
                           2
                          ___
                           1  0
                           1  0
                          _____
                              2
                              2
                             ___
                              5
                              4
                             ___
                              16
                              16
                             ___
                               0
```

According to the long division, the quotient is $141{,}270{,}000_8$ with a remainder of 0_8, which is correct. However, the computer would show the quotient to be $41{,}270{,}000_8$ with a remainder of 1_8, if it were not for the fact that a divide overflow condition existed. To prevent a divide overflow condition, the most significant bit of the quotient must appear in Register E.

FLOATING POINT NUMBERS

In many cases, the solution of a problem requires values of numbers that are either too large or too small to be expressed by the computer. The physical size of the number can be reduced by "scaling" or shifting the number to the right or left a predetermined number of places so that the most significant bits of the number may be used. For instance, the decimal number 6,510,000 may be expressed as 0.651×10^7, 0.0651×10^8, 0.00651×10^9, etc. The exponent of the number system base is the scale factor or the number of places the number is shifted.

The 6020 has fixed-point arithmetic, and there is no automatic hardware feature for handling the scaling factor or exponent. The programmer is responsible for remembering the scale factors. Also, the possibility of an overflow during intermediate operations must be considered. As was shown, if two fractions are multiplied, the result is a number that is less than 1. But, if the same two fractions are added, subtracted, or divided, the result may be greater than 1 and an overflow may occur.

As an alternative to fixed point operations, a method involving a variable radix point, called floating point, is used. The floating point system is similar to the scaling system except that the exponent is also contained in the data word of the 6020. The format of floating-point numbers is a fraction, or mantissa, multiplied by an exponent. Since the 6020 recognizes only binary numbers, the fractions are multiplied by powers of two.

In order to achieve a greater degree of accuracy, two computer words are used to express one floating-point number. The mantissa of a floating-point word is a 37-bit fraction plus the sign. The sign bits of each word (Bit 24) both apply to the mantissa and must agree for proper operation. Negative mantissas have sign bits of 1's; positive mantissas have sign bits of 0's. Exponents are not complemented.

The exponent of the floating point number is a 9-bit quantity that is contained in the least significant portion of the second word of the expression. It is formed by adding the true exponent to a "bias" of 400_8. This results in a range of biased exponents with true values from -377_8 to $+377_8$. Biasing of the exponents is performed so that they can be handled with greater ease by the computer.

FLOATING POINT ARITHMETIC

In order to add two floating point numbers, it is first necessary to equalize the exponents of the numbers. This is accomplished by shifting the mantissa of the smaller expression right the number of places that equals the difference of the two exponents. For example, in adding the floating point decimal numbers 0.3×10^4 and 0.27×10^6, 0.3×10^4 is written as 0.003×10^6 and then the two numbers are added which gives the results of 0.173×10^6.

$$\begin{array}{r} .3 \times 10^4 \\ + .27 \times 10^6 \\ \hline \end{array} \quad = \quad \begin{array}{r} .003 \times 10^6 \\ + .27 \times 10^6 \\ \hline .273 \times 10^6 \end{array}$$

The same procedure is required for subtraction except that the subtrahend is subtracted from the minuend in the final step of the operation.

$$\begin{array}{r} .27 \times 10^6 \\ - .3 \times 10^4 \\ \hline \end{array} \quad = \quad \begin{array}{r} .27 \times 10^6 \\ + .003 \times 10^6 \\ \hline .267 \times 10^6 \end{array}$$

To perform this operation with the binary numbers contained in the 6020, the exponents are first differenced. Then the mantissa of the number with the smallest exponent is shifted right the specified number of places, that is, the difference between the two exponents.

When this is accomplished, the two resulting floating point expressions are added with a double precision add instruction and the exponent of the larger number is affixed to the result.

The operation is the same for subtraction except that the sign of the subtrahend is changed before the double precision add is performed. The procedure outlined above is a much simplified analysis of floating point addition and subtraction, but serves to explain the basic principle.

Multiplication and division of the mantissa of the floating point expression is performed in the same manner as for normal fixed point numbers. The exponents, however, are added in multiplication and subtracted in division. For example:

$$\begin{array}{l} .2 \times 10^3 \\ \times .3 \times 10^4 \\ \hline .06 \times 10^7 \text{ or } .6 \times 10^6 \end{array}$$

$$\frac{.4 \times 10^7}{.2 \times 10^2} = 2 \times 10^5 \text{ or } .2 \times 10^6$$

Note that the result of all arithmetic operations are normalized and the exponent is corrected at the completion of the operation. If overflow occurs, as in the example of division, the mantissa is shifted right until the overflow is contained in the fractional representation; the exponent is then increased by the number of shifts required. In the binary circuits of the computer the number of shifts required to correct for overflow will never exceed 1. The shifts required for underflow (the result is less than ½) may be any number within the capability of the machine.

CONVERSION OF FIXED POINT NUMBERS TO FLOATING POINT NUMBERS

There are actually only two steps in converting a fixed point number to a floating point expression. They are (1) Normalize the number, and (2) calculate the biased exponent. The following examples show the procedure of fixed to floating point conversion.

Convert $+36.0_8$ to floating point:

1. Normalizing

$$+36.0_8 = 000\ 000\ 000\ 000\ 000\ 000\ 011\ 110_2$$
$$= 0.11\ 110 \times 2^5$$

2. Calculating the Exponent

   2^5 = an exponent of 5

   5 + 400 (bias) = 405 (biased exponent)

   therefore, $+36.0_8$ fixed point = 36000000 00000405_8 floating point

Convert -36.0_8 to floating point:

1. Normalizing

   $-36.0_8 = 000\ 000\ 000\ 000\ 000\ 000\ 011\ 110_2$

   $= 0.11\ 110 \times 2^5$

2. Calculating the Exponent

   2^5 = an exponent of 5

   5 + 400 (bias) = 405 (biased exponent)

   therefore, -36.0_8 fixed point = 76000000 40000405_8 floating point

Convert $+0.0067_8$ to floating point:

1. Normalizing

   $+0.0067_8 = 000\ 000\ 000\ 000\ 000\ 000\ 110\ 111_2$

   $= 0.11\ 011\ 1 \times 2^{-6}$

2. Calculating the Exponent

   2^{-6} = an exponent of -6

   -6 + 400 (bias) = 372 (biased exponent)

   therefore, $+0.0067_8$ fixed point = 33400000 00000372 floating point

CONVERSION OF FLOATING POINT NUMBERS TO FIXED POINT NUMBERS

The operations necessary to convert a floating point number to fixed point is merely the reversal of the fixed to floating procedure. The exponent is first unbiased and the mantissa of the floating point word is shifted right or left the number of places specified by the exponent.

Convert 27300000 00000403 to fixed point:

1. Unbias the exponent

   $403 - 400 = +3$

2. Shift the Mantissa

   $273_8 = 0.10\ 111\ 011_2 \times 2^3$

   $= +5.66_8$ fixed point

Convert 77140000 40000374 to fixed point:

1. Unbias the exponent

   $374 - 400 = -4$

2. Shift the Mantissa

   $-3714 = -0.11\ 111\ 001\ 100_2 \times 2^{-4}$

   $= -0.000\ 011\ 111\ 001\ 100$

   $= -0.03714_8$ fixed point

SECTION THREE
INSTRUCTION REPERTOIRE

This section defines all computer instructions and describes their operation. The instructions are grouped into areas of common operations.

A diagram representing the format of the Instruction Command Word is given for each instruction. Preceding the diagram is the mnemonic code for the instruction and its name. See Figure 3-1.

Timing is given in computer cycles where one cycle equals 1.90 microseconds. The timing includes memory reference but does not include indexing and indirect addressing time. Add two cycles to each instruction for indexing and one cycle for each indirect address reference.

OP CODE	I	X	Y
24 19	18	17 16	15 1

Figure 3-1. Instruction Diagram.

Each instruction diagram is divided into four fields representing the Operation Code, Indirect Address (I), Index Address (X), and Base Operand Address (Y) portions of the instruction. The small numbers below each diagram designate the bits which each field occupies in an instruction word. See figure 3-1.

The Operation Code for each instruction is expressed as a two-digit octal number and always appears as the left-most part of an instruction diagram.

If the Indirect Address and Index Address portions of an instruction diagram contain I and X, respectively, then indirect addressing and indexing can be used. The I and X portions will be replaced by an octal number in some instructions. In these cases, the I and X portions of the instruction are used in conjunction with the Operation Code to determine the operation.

The symbol "Y" appearing in the diagram denotes the Base Operand Address portion of the instruction and at times may be replaced by octal numbers. In these cases, the Base Operand Address portion of the instruction is used in conjunction with the Operation Code portion to determine the operation. The octal numbers of the Base Operand Address of the instruction along with the Operation Code determines the defined operation.

A short paragraph that explains the operation of the instruction follows the diagram. Included with the explanation is a shorthand notation of the instruction.

The terms commonly used in instruction descriptions and their definitions are:

1. **Effective Operand Address.** The final effective address of an instruction after all indexing and indirect addressing has occurred.
2. **Operand.** The contents of the memory location specified by the Effective Operand Address.
3. **Base Operand Address.** This refers to Bits 15 through 1 of the instruction or register being discussed. For example, the Base Operand Address portion of the Operand would mean the contents of Bits 15 — 1 of the memory location specified by the Effective Operand Address.

A list of symbols commonly used in the instruction descriptions and their definitions is given in Figure 3-2. All numbers used in the examples in this section are in octal form.

SYMBOL	DESIGNATION
A	Register A, or Accumulator
A_y	The base operand address portion (Bits 15-1) of the A register
E	Register E, a second major arithmetic register
AE	Designates Register A and Register E
E_{exp}	The exponent portion (Bits 9-1) of the E register. Specified index location
X	Specified index location
X_b	Bits 15-1 of the specified index location
i	The address of the instruction being performed
k	Shift count (Bits 6-1)
m	The effective address of the instruction after all indexing and indirect addressing has been performed; i.e., the effective operand address
( )	Contents of. (A) signifies the contents of the A register
(m)	The contents of the location specified by the effective operand address; i.e., the operand
(m_y)	The address portion (Bits 15-1) of the operand
S	Register S, the program sequence register
T	Register T, the temporary storage register and display register of the address of the last instruction completed
Y	The address portion of an instruction before indexing or indirect addressing takes place; i.e., the base operand address
$+$	Add
$-$	Subtract
$\bullet$	Multiply
$\div$	Divide
$\rightarrow$	Is placed in
$\vert\ \vert$	Absolute value. $\vert(A)\vert$ signifies the absolute value of the contents of the A register.
$\overline{}$	Complement of. $\overline{(A)}$ signifies the 1's complement of the A register.
$\oplus$	Logical OR (inclusive). $\overline{(E)} \oplus (m)$ signifies the logical OR of the 1's complement of the contents of the E register and the operand.
$\odot$	Logical AND. $(E) \odot (m)$ signifies the logical AND of the contents of the E register and the operand.

Figure 3-2. Table of Symbols.

INSTRUCTION OPERATION

DATA TRANSFER AND ARITHMETIC OPERATIONS

LDA: LOAD A Timing: 2

01	I	X	Y
24 … 19	18	17 16	15 … 1

$(m) \rightarrow A$

Place the operand in Register A. The operand in memory remains unchanged.

LDE: LOAD E Timing: 3

02	I	X	Y
24 … 19	18	17 16	15 … 1

$(m) \rightarrow E$

Place the operand in Register E. The operand in memory remains unchanged.

DLD: LOAD AE Timing: 3

03	I	X	Y
24 … 19	18	17 16	15 … 1

$(m) \rightarrow A,\ (m + 1) \rightarrow E$

Place the operand in Register A and the contents of the memory location sequentially following the effective operand address in Register E. The contents of both locations in memory remain unchanged.

LXP: LOAD EXPONENT Timing: 3

52	0	2	Y
24 … 19	18	17 16	15 … 1

$(m)_{exp} \rightarrow E_{exp}$

Place Bits 9-1 of the operand into Bits 9-1 of Register E without changing the remaining bits of Register E. The operand in memory remains unchanged. No indirect addressing or indexing will occur.

STA: STORE A Timing: 2

04	I	X	Y
24 … 19	18	17 16	15 … 1

$(A) \rightarrow m$

Replace the operand with the contents of Register A. Register A remains unchanged.

STE: STORE E **Timing 3**

05	I	X	Y
24 19	18	17 16	15 1

$(E) \rightarrow m$

Replace the operand with the contents of Register E. Register E remains unchanged.

DST: STORE AE **Timing: 3**

06	I	X	Y
24 19	18	17 16	15 1

$(A) \rightarrow m, (E) \rightarrow m + 1$

Replace the contents of the memory location specified by the effective operand address with the contents of Register A and replace the contents of the memory location sequentially following the effective operand address with the contents of Register E. The contents of Registers A and E remain unchanged.

SXP: STORE EXPONENT **Timing: 4**

52	0	3	Y
24 19	18	17 16	15 1

$(E)_{exp} \rightarrow m_{exp}, 0 \rightarrow m_{24-10}, 0 \rightarrow E_{exp}$

Place "0's" in Bits 24-10 of the operand and the contents of Bits 9-1 of Register E in Bits 9-1 of the operand. In Register E, Bits 24-10 remain unchanged and Bits 9-1 are cleared. No indirect addressing or indexing will occur.

SAM: STORE A ADDRESS **Timing: 3**

10	I	X	Y
24 19	18	17 16	15 1

$(A_y) \rightarrow (m_y)$

Replace the base operand address of the operand with the base operand address portion of Register A. Bits 24-16 of the operand and Register A remain unchanged.

ADD: ADD **Timing: 2**

11	I	X	Y
24 19	18	17 16	15 1

$(A) + (m) \rightarrow A$

Add the contents of Register A to the operand and place the sum in Register A. The operand remains unchanged.

Add overflow occurs and the AO flip-flop will be set if both numbers are of the same sign and a carry is generated into the sign bit. In this case, the sum is incorrect but the sign bit remains correct. An AO interrupt to Location 100 will occur when AO is set, the AO trap is armed, and the program is not already in an interrupt routine. AO will remain set until the AO interrupt occurs or until it is cleared by the program.

Note: A sum of —0 can result only from the operation: $(-0) + (-0)$.

MPY: MULTIPLY **Timing: 16**

12	I	X	Y
24 19	18	17 16	15 1

$(A) \bullet (m) \rightarrow AE$

Multiply the contents of Register A by the operand and place the product in Registers A and E with the most significant bits of the product in Register A and the least significant bits in Register E. The product appears as a 46-bit signed number with A_{24} and E_{24} set to the correct sign.

Memory Locations 110 and 111 are used during execution of the multiply instruction for the formation of partial products.

Note: A minus zero can result from the following operations:

$(-0) \times$ positive number

$(+0) \times$ negative number

$(-0) \times (+0)$

SUB: SUBTRACT **Timing: 2**

13	I	X	Y
24 19	18	17 16	15 1

$(A) - (m) \rightarrow A$

Subtract the operand from the contents of Register A and place the difference in Register A. The operand remains unchanged.

Add overflow occurs and the AO flip-flop will be set if the minuend and subtrahend are of opposite sign and a carry is generated into the sign bit. In this case, the difference is incorrect but the sign bit remains correct. An AO interrupt to Location 100 will occur when AO is set, the AO trap is armed, and the program is not already in an interrupt routine. AO will remain set until the AO interrupt occurs or until it is cleared by the program.

Note: A difference of —0 can result only from the operation: $(-0) - (+0)$.

AMA: ADD MAGNITUDE **Timing: 2**

43	I	X	Y
24 19	18	17 16	15 1

$|(A)| + |(m)| \longrightarrow A$

Add the contents of Register A to the operand as 24-bit magnitudes (without sign) and place the sum in Register A. The operand remains unchanged. Add overflow cannot occur.

DAD: DOUBLE PRECISION ADD **Timing: 6**

15	I	X	Y
24 19	18	17 16	15 1

$(AE) + (m, m + 1) \longrightarrow AE$

Add the double precision contents of Registers A and E and the double precision contents of the memory location specified by the operand address and the next sequential memory location. The result appears as a 46-bit number in Registers A_{23} — A_1 and E_{23} — E_1 with the sign of the result in A_{24}. The addend and augend should be two 46-bit numbers in the same format as the result with the signs in the upper halves of the numbers.

Add overflow occurs and the AO flip-flop will be set if both numbers are of the same sign and a carry is generated into the sign bit. In this case, the sum is incorrect but the sign bit remains correct. An AO interrupt to Location 100 will occur when AO is set, the AO trap is armed, and the program is not already in an interrupt routine. AO will remain set until the AO interrupt occurs or until it is cleared by the program.

Memory Location 110 is used during execution of the double add instruction for temporary storage.

Note: A sum of —0 can result only from the operations:

$(-0) + (-0)$

$(-N) + (+N)$

ADM: ADD TO MEMORY **Timing: 4**

17	I	X	Y
24 19	18	17 16	15 1

$(A) + (m) \longrightarrow m$

Add the contents of Register A and the operand and place the sum in the memory location specified by the effective operand address. The contents of Register A remain unchanged.

Add overflow occurs and the AO flip-flop will be set if both numbers are of the same sign and a carry is generated into the sign bit. In this case, the sum is incorrect but the sign bit remains correct. An AO interrupt to Location 100 will occur when AO is set, the AO trap is armed, and the program is not already in an interrupt routine. AO will remain set until the AO interrupt occurs or until it is cleared by the program.

Note: A sum of —0 can result only from the operation:

$(-0) + (-0)$.

DVD: DIVIDE **Timing: 25**
3 if divide fault

14	I	X	Y
24 19	18	17 16	15 1

$(AE) \div (m) \longrightarrow E$

Remainder $\longrightarrow A$

If $|(A)| < |(m)|$, $(S) + 1 \longrightarrow S$

Divide the 46-bit signed fraction contained in Registers A and E by the operand and place the quotient in Register E and the remainder in Register A. A true remainder is generated with the sign algebraically correct.

For proper division to occur, the dividend should have its most significant half in A_{23} — A_1, and its least significant half in E_{23} — E_1, with the correct sign appearing in A_{24}. E_{24} is ignored.

Before division, if the absolute value of Register A is greater than or equal to the absolute value of the operand, a divide fault will occur. If divide fault occurs, the original contents of Registers A and E will remain unchanged, and the next instruction will be taken in sequence. If divide fault does not occur, the next instruction will be skipped, and the instruction following will be taken.

Note: A minus zero quotient or remainder can only result from the following operations:

$(+0) \div$ negative number, remainder $= (+0)$, Quotient $= (-0)$

$(-0) \div$ positive number, remainder $= (-0)$, Quotient $= (-0)$

$(-0) \div$ negative number, remainder $= (-0)$, Quotient $= (+0)$

AOA: ADD ADDRESS **Timing: 2**

52	0	0	Y
24 19	18	17 16	15 1

$(A) + Y \longrightarrow A$

No indirect addressing or indexing will occur. Add the contents of Register A and the base operand address and place the sum in Register A.

Add overflow occurs and the AO flip-flop will be set if both numbers are of the same sign and a carry is generated into the sign bit. In this case, the sum is incorrect but the sign bit remains correct. An AO interrupt to Location 100 will occur when AO is set, the AO trap is armed, and the program is not already in an interrupt routine. AO will remain set until the AO interrupt occurs or until it is cleared by the program.

SOA: SUBTRACT ADDRESS **Timing: 2**

52	0	1	Y
24 19	18	17 16	15 1

$(A) - Y \rightarrow A$

No indirect addressing or indexing will occur. Subtract the base operand address from the contents of Register A and place the difference in Register A.

Add overflow occurs and the AO flip-flop will be set if the minuend and subtrahend are of opposite sign and a carry is generated into the sign bit. In this case, the difference is incorrect but the sign bit remains correct. An AO interrupt to Location 100 will occur when AO is set, the AO trap is armed, and the program is not already in an interrupt routine. AO will remain set until the AO interrupt occurs or until it is cleared by the program.

Note: A difference of -0 can result only if the original content of Register A is -0 and the base operand address is $+0$.

LOA: LOAD ADDRESS **Timing: 3**

07	I	X	Y
24 19	18	17 16	15 1

$Y \rightarrow A_y$, $0 \rightarrow A_{24}$ thru A_{16}

Place the base operand address in Bits 15 through 1 of Register A. Clear Bits 24 through 16 of Register A.

RND: ROUND **Timing: 2**

50	0	1	00000
24 19	18	17 16	15 1

If E_{23}: $(A) + 1 \rightarrow A$

If E_{23} is a "1," increase the magnitude of contents of Register A by one.

Add overflow occurs and the AO flip-flop will be set if the number in A is of maximum magnitude. In this case, the sum is incorrect but the sign bit remains correct. An AO interrupt to Location 100 will occur when AO is set, the AO trap is armed, and the program is not already in an interrupt routine. AO will remain set until the AO interrupt occurs or until it is cleared by the program.

REGISTER CHANGE INSTRUCTIONS

All the Register Change Instructions use the 15-bit base address portion of the instruction as part of the Operation Code in order to determine the operation to be performed. Indexing and indirect addressing are not allowed.

ZOA: ZERO A **Timing: 2**

50	0	0	00002
24 19	18	17 16	15 1

$0 \rightarrow A$

Clear Register A to all zeroes.

ZOE: ZERO E **Timing: 2**

50	0	0	00001
24 19	18	17 16	15 1

$0 \rightarrow E$

Clear Register E to all zeroes.

ZOD: ZERO AE **Timing: 2**

50	0	0	00003
24 19	18	17 16	15 1

$0 \rightarrow A$
$0 \rightarrow E$

Clear Register A to all zeros and clear Register E to all zeros.

ZSA: CLEAR A TO SIGN OF E **Timing: 2**

50	0	0	01002
24 19	18	17 16	15 1

$\pm 0 \rightarrow A$ (Same sign as E)

Place "0's" in Bits 23 — 1 of Register A. Place E_{24} in A_{24}. The contents of Register E remain unchanged.

CMA: COMPLEMENT A **Timing: 3**

50	0	0	00010
24 19	18	17 16	15 1

$\overline{(A)} \rightarrow A$

Complement the contents of Register A.

CME: COMPLEMENT E **Timing: 3**

50	0	0	00004
24 19	18	17 16	15 1

$\overline{(E)} \rightarrow E$

Complement the contents of Register E.

CMD: COMPLEMENT AE **Timing: 3**

50	0	0	00014
24 19	18	17 16	15 1

$\overline{(A)} \rightarrow A$

$\overline{(E)} \rightarrow E$

Complement the contents of Register A and complement the contents of Register E.

MNA: MINUS A **Timing: 2**

50	0	0	00040
24 19	18	17 16	15 1

$-(A) \rightarrow A$

Complement A_{24}. A_{23} through A_1 remain unchanged.

MNE: MINUS E **Timing: 2**

50	0	0	00020
24 19	18	17 16	15 1

$-(E) \rightarrow E$

Complement E_{24}. E_{23} through E_1 remain unchanged.

MND: MINUS AE **Timing: 2**

50	0	0	00060
24 19	18	17 16	15 1

$-(A) \rightarrow A$

$-(E) \rightarrow E$

Complement A_{24} and E_{24}. A_{23} through A_1 and E_{23} through E_1 remain unchanged.

AVA: ABSOLUTE VALUE A **Timing: 2**

50	0	0	00200
24 19	18	17 16	15 1

$0 \rightarrow A_{24}$

Clear A_{24}.

AVE: ABSOLUTE VALUE E **Timing: 2**

50	0	0	00100
24 19	18	17 16	15 1

$0 \rightarrow E_{24}$

Clear E_{24}.

AVD: ABSOLUTE VALUE AE **Timing: 2**

50	0	0	00300
24 19	18	17 16	15 1

$0 \rightarrow A_{24}$

$0 \rightarrow E_{24}$

Clear A_{24} and E_{24}.

SAE: SIGN OF A TO E **Timing: 2**

50	0	0	00400
24 19	18	17 16	15 1

$A_{24} \rightarrow E_{24}$

Place A_{24} in E_{24}. A_{24} is unchanged.

SEA: SIGN OF E TO A **Timing: 2**

50	0	0	01000
24 19	18	17 16	15 1

$E_{24} \rightarrow A_{24}$

Place E_{24} in A_{24}. E_{24} is unchanged.

ATE: COPY A IN E **Timing: 2**

50	0	0	02000
24 19	18	17 16	15 1

$(A) \rightarrow E$

Place the contents of Register A in Register E. The contents of Register A remain unchanged.

ETA: COPY E IN A **Timing: 2**

50	0	0	04000
24 19	18	17 16	15 1

$(E) \rightarrow A$

Place the contents of Register E in Register A. The contents of Register E remain unchanged.

XAE: EXCHANGE AE **Timing 2**

50	0	0	06000
24 19	18	17	16 15 1

(A) $\rightarrow$ E

(E) $\rightarrow$ A

Place the original contents of Register A in Register E and place the original contents of Register E in Register A.

LOGICAL OPERATIONS

Note: The logical complement of Register A or Register E may be obtained by using the CMA and CME instructions.

AND: LOGICAL "AND" **Timing: 2**

40	I	X	Y
24 19	18	17	16 15 1

$(E) \odot (m) \rightarrow A$

Form the logical AND of the contents of Register E and the operand and place the result in Register A. The original contents of Register E remain unchanged.

XOR: EXCLUSIVE "OR" **Timing: 2**

41	I	X	Y
24 19	18	17	16 15 1

$(E) \odot \overline{(m)} \oplus \overline{(E)} \odot (m) \rightarrow A$

Form the exclusive OR of the contents of Register E and the operand and place the result in Register A. The original contents of Register E remain unchanged.

LOR: LOGICAL "OR" **Timing: 2**

42	I	X	Y
24 19	18	17	16 15 1

$(E) \oplus (m) \rightarrow A$

Form the logical OR of the contents of Register E and the operand and place the result in Register A. The original contents of Register E remain unchanged.

SHIFT OPERATIONS

Shift instructions use Bits 15 - 11 of the operand address to determine the type of shift operation. Bits 6 - 1 of the operand address contain the shift count (k). The maximum number of shifts is 63_{10}. Bits 10 — 7 are unspecified.

Indirect addressing and indexing can be used with these instructions. If indexing is used, and the Index Base contains bits in Positions 15 - 11, then the original operation may be changed. When indexing is specified, the contents of the index register are added to the base operand address and this may change the type of shift operation and the shift count.

If indirect addressing is used, then the base operand address determines the type of shift operation and the shift count. Therefore, indirect addressing with or without indexing may change the operation.

RSA: RIGHT SHIFT A **Timing: 2 + k**

20	I	X	400 k
24 19	18	17	16 15 1

(A) right k places $\rightarrow$ A

Shift Bits 23 through 1 of Register A right k places. All bits shifted out of A_1 will be lost and "0's" will be entered into A_{23}. Sign Bit 24 of Register A is unchanged. The original contents of Register E remain unchanged.

LSA: LEFT SHIFT A **Timing: 2 + k**

20	I	X	500 k
24 19	18	17	16 15 1

(A) left k places $\rightarrow$ A

Shift Bits 23 through 1 of Register A left k places. All bits shifted out of A_{23} will be lost and "0's" will be entered into A_1. Sign Bit 24 of Register A is unchanged. The original contents of Register E remain unchanged.

CSA: CIRCULAR LEFT SHIFT A **Timing: 2 + k**

20	I	X	540 k
24 19	18	17	16 15 1

(A) left circular k places $\rightarrow$ A

Shift the contents of Register A left circular k places. All bits shifted out of A_{24} will be placed into A_1. Sign Bit A_{24} is shifted with the contents of Register A. The original contents of Register E remain unchanged.

RLA: LOGICAL RIGHT SHIFT A **Timing: 2 + k**

20	I	X	420 k
24 19	18	17	16 15 1

(A) left k places $\rightarrow$ A

Shift the contents of Register A right k places. All bits shifted out of A_1 will be lost and "0's" will be entered into A_{24}. The original contents of Register E remain unchanged.

LLA: LOGICAL LEFT SHIFT A Timing: 2 + k

20	I	X	520 k
24 19	18	17 16	15 1

(A) left k places $\rightarrow$ A

Shift the contents of Register A left k places. All bits shifted out of A_{24} will be lost and "0's" will be entered into A_1. The original contents of Register E remain unchanged.

RSE: RIGHT SHIFT E Timing: 2 + k

20	I	X	200 k
24 19	18	17 16	15 1

(E) right k places $\rightarrow$ E

Shift Bits 23 through 1 of Register E right k places. All bits shifted out of E_1 will be lost and "0's" will be entered into E_{23}. Sign Bit 24 of Register E is unchanged. The original contents of Register A remain unchanged.

LSE: LEFT SHIFT E Timing: 2 + k

20	I	X	300 k
24 19	18	17 16	15 1

(E) left k places $\rightarrow$ E

Shift Bits 23 through 1 of Register E left k places. All bits shifted out of E_{23} will be lost and "0's" will be entered into E_1. Sign Bit 24 of Register E is unchanged. The original contents of Register A remain unchanged.

CSE: CIRCULAR LEFT SHIFT E Timing: 2 + k

20	I	X	340 k
24 19	18	17 16	15 1

(E) left circular k places $\rightarrow$ E

Shift the contents of Register E left circular k places. All bits shifted out of E_{24} will be placed into E_1. Sign Bit E_{24} is shifted with the contents of Register E. The original contents of Register A remain unchanged.

RLE: LOGICAL RIGHT SHIFT E Timing: 2 + k

20	I	X	220 k
24 19	18	17 16	15 1

(E) right k places $\rightarrow$ E

Shift the contents of Register E right k places. All bits shifted out of E_1 will be lost and "0's" will be entered into E_{24}. The original contents of Register A remain unchanged.

LLE: LOGICAL LEFT SHIFT E Timing: 2 + k

20	I	X	320 k
24 19	18	17 16	15 1

(E) left k places $\rightarrow$ E

Shift the contents of Register E left k places. All bits shifted out of E_{24} will be lost and "0's" will be entered into E_1. The original contents of Register A remain unchanged.

RSD: RIGHT SHIFT AE Timing: 2 + k

20	I	X	600 k
24 19	18	17 16	15 1

(AE) right k places $\rightarrow$ AE

Shift Bits 23 through 1 of Register A and Bits 23 through 1 of Register E right k places. All bits shifted out of A_1 will be placed into E_{23} and all bits shifted out of E_1 will be lost. "0's" will be entered into A_{23}. Sign Bits 24 of Registers A and E are unchanged.

LSD: LEFT SHIFT AE Timing: 2 + k

20	I	X	700 k
24 19	18	17 16	15 1

(AE) left k places $\rightarrow$ AE

Shift Bits 23 through 1 of Register A and Bits 23 through 1 of Register E left k places. All bits shifted out of E_{23} will be placed into A_1 and all bits shifted out of A_{23} will be lost. "0's" will be entered into E_1. Sign Bits 24 of Registers A and E are unchanged.

CSD: CIRCULAR LEFT SHIFT AE Timing: 2 + k

20	I	X	740 k
24 19	18	17 16	15 1

(AE) left circular k places $\rightarrow$ AE

Shift the contents of Registers A and E left circular k places. All bits shifted out of A_{24} will be placed in E_1 and all bits shifted out of E_{24} will be placed in A_1. Sign Bits 24 of Registers A and E are shifted with the contents of the registers.

RLD: LOGICAL RIGHT SHIFT AE Timing: 2 + k

20	I	X	620 k
24 19	18	17 16	15 1

(AE) right k places $\rightarrow$ AE

Shift the contents of Registers A and E right k places. All bits shifted out of A_1 will be placed in E_{24}. All bits shifted out of E_1 will be lost and "0's" will be entered into A_{24}.

LLD: LOGICAL LEFT SHIFT AE Timing: 2 + k

20	I	X	720 k
24 19	18	17 16	15 1

(AE) left k places $\rightarrow$ AE

Shift the contents of Registers A and E left k places. All bits shifted out of E_{24} will be placed in A_1. All bits shifted out of A_{24} will be lost and "0's" will be entered into E_1.

NRM: NORMALIZE Timing: 6 + k

50	1	2	00000
24 19	18	17 16	15 1

(AE) left until A_{23} is a "1"

$(i + 1) - k \rightarrow i + 1$

$(S) + 1 \rightarrow S$

Shift Bits 23 through 1 of Registers A and E left until A_{23} is a "1" or until 46 shifts have been performed. All bits shifted out of E_{23} will be placed in A_1 and "0's" will be entered into E_1. Sign Bits 24 of Registers A and E are unchanged.

The count of the number of shifts required will be subtracted from the 24-bit absolute value of the memory location sequentially following the location of the NRM instruction. After execution of the NRM instruction, the next instruction is skipped.

INDEX OPERATIONS

SXM: STORE INDEX Timing: 4

53	0	X	Y
24 19	18	17 16	15 1

$X_b \rightarrow m_y$

No indirect addressing or indexing will occur. Place the contents of the specified index base in the base operand address portion of the operand. Bits 24 through 16 of the operand and the contents of the index location remain unchanged.

ZOM: ZERO MEMORY Timing: 4

53	0	0	Y
24 19	18	17 16	15 1

$0 \rightarrow m_y$

If an index location is not specified in the SXM instruction, a ZOM instruction occurs. Bits 16 through 1 of the operand are cleared, while Bits 24 through 16 remain unchanged.

AUX: AUGMENT INDEX Timing: 4

54	0	X	Y
24 19	18	17 16	15 1

$Y + X_b \rightarrow X_b$, $0 \rightarrow X_{24}$ through X_{16}

If $Y + X_b = 0$, $(S) + 1 \rightarrow S$

No indirect addressing or indexing will occur. Add the base operand address portion of this instruction to the specified index base and place the result in the specified index location. If this results in all "0's" in the index, the next instruction is skipped. If not, the next instruction is taken in sequence. Bits 24 through 16 of the specified index location are cleared.

Decrementing of an index base can be affected by placing the 2's complement of the decrement value in the base operand address portion of the AUX instruction.

KOZ: SKIP IF BASE OPERAND ADDRESS IS ZERO Timing: 4

54	0	0	Y
24 19	18	17 16	15 1

If $Y = 0$, $(S) + 1 \rightarrow S$

If an index location is not specified in the AUX instruction, a KOZ instruction occurs. When the base operand address of this instruction is "0," the next sequential instruction is skipped.

JDX: JUMP AND DECREMENT INDEX Timing: 3

54	1	X	Y
24 19	18	17 16	15 1

$X_b - 1 \rightarrow X_b$; if $X_b - 1 \neq 0$, $m \rightarrow S$

No indirect addressing or indexing will occur. Subtract +1 from the specified index base and place the result in the specified index. If the result is not equal to zero, the next instruction from the location specified by the effective operand address is taken. If the result is equal to zero, the next instruction in sequence is taken. Bits 24 through 16 of the index location are cleared.

LOX: LOAD INDEX WITH ADDRESS Timing: 2

55	0	X	Y
24 19	18	17 16	15 1

$Y \rightarrow X_b$

$0 \rightarrow X_{24}$ through X_{16}

No indirect addressing or indexing will occur. Place the base operand address portion of this instruction in the specified index base. Clear Bits 24 through 16 of the specified index location.

LAX: LOAD INDEX WITH A ADDRESS Timing: 2

55	1	X	Y
24 19	18	17 16	15 1

$(A_y) \longrightarrow X_b$

$0 \longrightarrow X_{24}$ through X_{16}

No indirect addressing or indexing will occur. Place the base operand address portion of Register A in the specified index base. Clear Bits 24 through 16 of the specified index location.

IXT: INCREMENT INDEX AND TEST Timing: 5 if skip 4 if no skip

56	0	X	Y
24 19	18	17 16	15 1

$X_b + 1 \longrightarrow X_b$

If $X_b + 1 = Y$, then $(S) + 1 \longrightarrow S$

No indirect addressing or indexing will occur. Add the number +1 to the specified index base and place the result in the specified index base. Clear Bits 24 through 16 of the specified index location. If the result is equal to the base operand address, the next instruction is skipped. If the result is not equal, the next instruction is taken in sequence.

KON: SKIP IF BASE OPERAND ADDRESS IS ONE Timing: 5 if skip 4 if no skip

56	0	0	Y
24 19	18	17 16	15 1

If $Y = 1$, $(S) + 1 \longrightarrow S$

If an index location is not specified in the IXT instruction, a KON instruction occurs. When the base operand address of this instruction is "1," the next sequential instruction is skipped.

KXH: SKIP INDEX HIGH Timing: 3 if skip 2 if no skip

57	0	X	Y
24 19	18	17 16	15 1

If $X_b > Y$, then $(S) + 1 \longrightarrow S$

No indirect addressing or indexing will occur. If the specified index base is greater than the base operand address, the next instruction is skipped. If the specified index base is not greater, the next instruction is taken in sequence.

JUMP AND SKIP OPERATIONS

JMP: JUMP Timing: 1

21	I	X	Y
24 19	18	17 16	15 1

$m \longrightarrow S$

XEC: EXECUTE Timing: 1 + CYC

22	I	X	Y
24 19	18	17 16	15 1

Execute the operand as an instruction, but do not leave this instruction sequence unless the operand instruction leads to a SKIP or JUMP. If it leads to a SKIP, the instruction following the XEC instruction is skipped. If it leads to a JUMP, the next instruction from the location specified by the effective operand address is taken.

JDI: JUMP ALLOW INTERRUPT Timing: 2

23	I	X	Y
24 19	18	17 16	15 1

$m \longrightarrow S$

Allow Interrupt

Clear the highest priority interrupt flip-flop that is set. If none is set, clear the non-priority interrupt flip-flop. Take the next instruction from the location specified by the effective operand address.

JPO: JUMP TO M + 1 Timing: 2

24	I	X	Y
24 19	18	17 16	15 1

$m + 1 \longrightarrow S$

Take the next instruction from the location sequentially following the location specified by the effective operand address. Only one level of indirect addressing will occur; i. e., Bit 18 in the location specified by the indirect address will be ignored.

RTJ: RETURN AND JUMP Timing: 3

25	I	X	Y
24 19	18	17 16	15 1

$(S) \longrightarrow m_Y$

$m + 1 \longrightarrow S$

Place the address of the instruction sequentially following the RTJ instruction in the base operand address portion of the operand. Bits 24 through 16 of the operand remain unchanged. Take the next instruction from the location sequentially following the location specified by the effective operand address.

JLZ: JUMP IF A < 0 — Timing: 2 if Jump, 3 if No Jump

30	I	X	Y
24 19	18	17 16	15 1

If $(A) < 0$, $m \longrightarrow S$

If the contents of Register A is algebraically less than zero, take the next instruction from the location specified by the effective operand address. The jump does not occur if (A) is —0.

JGZ: JUMP IF A > 0 — Timing: 2 if Jump, 3 if No Jump

31	I	X	Y
24 19	18	17 16	15 1

If $(A) > 0$, $m \longrightarrow S$

If the contents of Register A is algebraically greater than zero, take the next instruction from the location specified by the effective operand address.

JEZ: JUMP IF A = 0 — Timing: 2 if Jump, 3 if No Jump

32	I	X	Y
24 19	18	17 16	15 1

If $(A) = \pm 0$, $m \longrightarrow S$

If the contents of Register A equals zero, take the next instruction from the location specified by the effective operand address. This jump occurs when (A) is +0 or —0.

KAL: SKIP A LOW — Timing: 4

34	I	X	Y
24 19	18	17 16	15 1

If $(A) < (m)$, $(S) + 1 \longrightarrow S$

If the contents of Register A is algebraically less than the operand, skip the next instruction. If not, take the next instruction in sequence.

KAH: SKIP A HIGH — Timing: 4

35	I	X	Y
24 19	18	17 16	15 1

If $(A) > (m)$, $(S) + 1 \longrightarrow S$

If the contents of Register A is algebraically greater than the operand, skip one instruction. If not, take the next instruction in sequence.

KAQ: SKIP A EQUAL — Timing: 4

36	I	X	Y
24 19	18	17 16	15 1

If $(A) = (m)$, $(S) + 1 \longrightarrow S$

If the contents of Register A is equal to the operand, skip one instruction. If not, take the next instruction in sequence. Note: +0 = —0.

IMT: INCREMENT MEMORY AND TEST — Timing: 4

37	I	X	Y
24 19	18	17 16	15 1

$(m) + 1 \longrightarrow m$

If $(m) + 1 = \pm 0$, then $(S) + 1 \longrightarrow S$

Add the number +1 to the algebraic value of the operand and place the result in the memory location specified by the effective operand address. If the result is ±0, the next instruction is skipped. If not, the next instruction is taken in sequence. +0 is generated by adding +1 to —1 and —0 is generated by adding +1 to +37777777

Add overflow cannot occur.

HLT: HALT

51	0	0	Y
24 19	18	17 16	15 1

HALT

$Y \longrightarrow T$

Halt and take the next instruction (when the RUN switch is pressed) in sequence.

KIF: CONDITIONAL SKIP — Timing: 3 if skip, 2 if no skip

51	1	0	Y
24 19	18	17 16	15 1

If the specified condition is true and the corresponding bit of Y is a "1," the next instruction is taken in sequence. If not, the next instruction is skipped.

If bit 7 is set, the specified flip-flop is cleared.

Bit Position	Condition
1	Sense Switch 1
2	Sense Switch 2
3	Sense Switch 3
4	Sense Switch 4
5	Sense Switch 5
6	Sense Switch 6
7	Add Overflow Flip-Flop
8	————
9	————
10	Flag 1 (Indicator Light 1)
11	Flag 2 (Indicator Light 2)
12	Flag 3
13	Flag 4
14	Flag 5
15	Flag 6

Any combination of bits can be used to simultaneously test more than one condition. If this is done, a skip occurs if none of the tested conditions are true.

KEX: SKIP ON EXTERNAL SIGNAL

Timing:
3 if skip
2 if no skip

51	1	1	Y
24 19	18	17 16	15 1

If the external signal specified by Y is a "0," the next instruction is skipped. If not, the next instruction is taken in sequence.

TRAP, FLAG, AND FLAG INDICATOR LIGHT OPERATIONS

All Trap, Flag, and Flag Indicator Light instructions use the address portion of the instruction to specify the operation. Indexing and indirect addressing may not be used. In discussing these instructions, all reference is to the bit configuration appearing in the address portion of the instruction.

Trap settings control which interrupt signals will be allowed to interrupt a program in process. If a trap is armed, then the associated interrupt conditions will be permitted to interrupt the main program when they occur. A trap which has not been armed or has been disarmed inhibits the occurrence of interrupt signals.

The set and clear flag operations are used to change the settings of four flag flip-flops. The condition of the flag flip-flops can be tested by other instructions. These provide built-in hardware switches which can be used to control program branching.

Two Flag Indicator Lights can be set and cleared by the trap instructions. These can also be tested by other instructions and can be used as the flags to control program branching. In addition, their condition is displayed on the operator's console thus providing a means for visible program controlled operator directives.

CTP: CLEAR FLIP-FLOPS

Timing: 2

51	0	2	Y
24 19	18	17 16	15 1

Clear the flip-flops specified by the bit position in Y.

Bit Position	Flip-Flop
1	Priority Trap
2	ED Trap
3	Programmed I/O Channel Trap
4	Operator Trap
5	Power Fail Trap
6	Add Overflow Trap
7	Add Overflow Flip-Flop
8	Memory Parity Fail Trap
9	————
10	Flag 1 (Flag Light 1)
11	Flag 2 (Flag Light 2)
12	Flag 3
13	Flag 4
14	Flag 5
15	Flag 6

A "1" in the bit position will clear the flip-flop. A "0" has no effect. Any number of flip-flops can be cleared in one instruction by specifying the correct combination of "1" bits.

STP: SET FLIP-FLOPS

Timing: 2

51	0	3	Y
24 19	18	17 16	15 1

Set the flip-flops specified by the bit position in Y.

Bit Position	Flip-Flop
1	Priority Trap

2	ED Trap
3	Programmed I/O Channel Trap
4	Operator Trap
5	Power Fail Trap
6	Add Overflow Trap
7	Add Overflow Flip-Flop
8	Memory Parity Fail Trap
9	("0")
10	Flag 1 (Flag Light 1)
11	Flag 2 (Flag Light 2)
12	Flag 3
13	Flag 4
14	Flag 5
15	Flag 6

A "1" in the bit position will set the flip-flop. A "0" has no effect. Any number of flip-flops can be set in one instruction by specifying the correct combination of "1" bits.

If any of the interrupt conditions (as specified by Bit Positions 1, 2, 3, 5, 6, and 8) have occurred prior to execution of this instruction and the particular trap is armed, an interrupt will occur immediately after executing this instruction.

SCT: SAVE AND CLEAR FLIP-FLOPS Timing: 2

51	1	2	Y
24 19	18	17 16	15 1

Clear Register A. Place the condition of the flip-flops specified by Y in the corresponding bit positions of Register A. Then, clear the flip-flops specified by Y.

Bit Position	**Flip-Flop**
1	Priority Trap
2	ED Trap
3	Programmed I/O Channel Trap
4	Operator Trap
5	Power Fail Trap
6	Add Overflow Trap
7	Add Overflow Flip-Flop
8	Memory Parity Fail Trap
9	————
10	Flag 1 (Flag Light 1)
11	Flag 2 (Flag Light 2)
12	Flag 3
13	Flag 4
14	Flag 5
15	Flag 6

A "1" in the bit position will save and clear the flip-flops. A "0" has no effect. Any number of flip-flops can be saved and cleared in one instruction by specifying the correct combination of "1" bits.

SST: SAVE AND SET FLIP-FLOPS Timing: 2

51	1	3	Y
24 19	18	17 16	15 1

Clear Register A. Place the condition of the flip-flops specified by Y in the corresponding bit positions of Register A. Then, set the flip-flops specified by Y.

Bit-Position	**Flip-Flop**
1	Priority Trap
2	ED Trap
3	Programmed I/O Channel Trap
4	Operator Trap
5	Power Fail Trap
6	Add Overflow Trap
7	Add Overflow Flip-Flop
8	Memory Parity Fall Trap
9	("0")
10	Flag 1 (Flag Light 1)
11	Flag 2 (Flag Light 2)
12	Flag 3
13	Flag 4
14	Flag 5
15	Flag 6

A "1" in the bit position will save and set the flip-flop. A "0" has no effect. Any number of flip-flops can be set in one instruction by specifying the correct combination of "1" bits.

If any of the interrupt conditions (as specified by Bit Positions 1, 2, 3, 5, 6, 7, and 8) have occurred prior to the execution of this instruction and the particular trap is armed, an interrupt will occur immediately after executing this instruction.

INPUT/OUTPUT OPERATIONS

ASR: ASSEMBLY REGISTER Timing: 2 to 7

26	I	X	Y
24 19	18	17 16	15 1

Interpret the operand as the first of a set of Assembly Register Control Words (ARCW) that occupy consecutive memory locations. If the specified channel is not busy, the next instruction is skipped. See Section 4, Input/Output, for a detailed explanation of ARCW.

The format of the ARCW is as follows:

Bits 24-22: **Channel Address.** The channel address specifies which channel will be operated on. Eight different channels can be specified.

Bits 21-19: **Designator.** The designator specifies which operation will be performed.

0 - LOAD BM with the contents of Bits 15-1 of the ARCW if the specified channel is not busy.

1 - LOAD BL with the contents of Bits 15-1 of the ARCW if the specified channel is not busy.

2 - STORE BM into Bits 15-1 of the memory location determined by Bits 15-1 of the ARCW if the specified channel is not busy. Bits 24-16 of the memory location determined by Bits 15-1 of the ARCW remain unchanged.

3 - LOAD BG. If Bit 15 is a "1," this transfer will occur if the specified channel is not busy. If Bit 15 is a "0," this transfer will occur regardless of the busy condition. LOAD BG is used in some of the buffered I/O channels. For details of operation for each particular channel, see the description for that channel in the Input/Output Section.

4 - LOAD BM with the contents of Bits 15-1 of the ARCW regardless of the busy condition of the specified channel.

5 - LOAD BL with the contents of Bits 15-1 of the ARCW regardless of the busy condition of the specified channel.

6 - STORE BM into Bits 15-1 of the memory location specified by Bits 15-1 of the ARCW regardless of the busy condition of the specified channel. Bits 24-16 of the memory location specified by Bits 15-1 of the ARCW remain unchanged.

7 - Unspecified.

NOTE: The above operations apply to the Character and Word channels only. For operations pertaining to other channels, see the description for that particular channel in Section 4, Input/Output.

Bit 18: **Last Control Word:** If Bit 18 is a "1," the control word is not the last word in the set. The last control word of the set contains a "0" in Bit 18 or specifies a Store BM operation.

Bits 17-16: **Index Location.** These two bits specify an index location just as they do in an instruction word.

Bits 15-1: **Memory Address.**

EXD: EXTERNAL DEVICE Timing: 6

27	I	X	Y
24 19	18	17 16	15 1

Interpret the operand as the first of a set of control words that occupy consecutive memory locations. The last control word of the set is either an External Device Control Word (EDCW) or an Assembly Register Control Word (ARCW) that causes a Store BM operation. All other control words are ARCW's that do not Store BM. See Section 4, Input/Output, for a detailed explanation of EDCW.

The format of the EDCW is as follows:

Bits 24-19: **ED Address.** The ED address specifies which external device this particular instruction is referring to. Sixty-four different external devices can be specified.

Bit 18: **Last Control Word.** If Bit 18 is a "1," the control word is an ARCW and not the last word in the set. The last control word of the set contains a "0" in Bit 18 specifying an EDCW.

Bit 17: **SKIP if not busy.** If Bit 17 of the EDCW is a "0" and the specified external device is not busy, the next instruction is skipped. If not, the next instruction is taken in sequence.

Bit 16: **Inhibit ED START signal.** If Bit 16 of the EDCW is a "1," the ED START signal will be inhibited.

Bit 15: **Interrupt.** If Bit 15 of the EDCW is a "1" and the specified external device is given an ED START signal, the external device will send an interrupt when it has completed the specified operation.

Bits 14-1: **ED Operation.** These bits specify what operation the specified external device will perform.

The ED START signal occurs if the specified external device is not busy, the channel the specified external device is connected to is not busy, and Bit 16 of the EDCW is a "0." The ED START signal is transmitted from the central processor to the specified external device and initiates the specified operation.

XAK: EXTERNAL LINES TO A AND SKIP IF READY

Timing: 4 if skip
3 if no skip

50	0	2	Y
24 19	18	17 16	15 1

The original contents of Register A is automatically stored in Memory Location 110 as the first step in the execution of XAK regardless of the READY status. If the READY signal is present when the instruction is given, the data from the external lines is transferred to Register A and the next sequential instruction is skipped. If the READY signal is **not** present, the data is not transferred to Register A and the next instruction is taken in sequence. See "Buffered Input/Output" in Section 4 for bit designation.

AXK: A TO EXTERNAL LINES AND SKIP IF READY

Timing: 4 if skip
3 if no skip

50	1	0	Y
24 19	18	17 16	15 1

If the READY signal is present when the instruction is given, the data in Register A is transferred to the external lines and the next sequential instruction is skipped. If the READY signal is **not** present, the data is not transferred to the external lines and the next instruction is taken in sequence. See "Buffered Input/Output" in Section 4 for bit designation.

SEN: SEND EXTERNAL SIGNAL Timing: 3

51	0	1	Y
24 19	18	17 16	15 1

Send the external signal that is specified by Y.

NO OPERATION

The no operation instructions cause nothing to occur as far as the computer is concerned. They may be used as "fill-in" instructions in lieu of instructions to be added later or to consume time for special applications.

NOP: NO OPERATION Timing: 2 + k

20	I	X	000 k
24 19	18	17 16	15 1

NOP: NO OPERATION Timing: 2

50	0	0	00000
24 19	18	17 16	15 1

PROGRAMMED INSTRUCTION

There are sixteen programmed instructions which permit routines assigned to these instructions to be handled as hardware instructions. As various routines are assigned to these instructions, each instruction will receive an individual mnemonic code.

Each of the sixteen programmed instructions has a fixed address as follows:

Octal Code	Fixed Address	Octal Code	Fixed Address
60	00120	70	00130
61	00121	71	00131
62	00122	72	00132
63	00123	73	00133
64	00124	74	00134
65	00125	75	00135
66	00126	76	00136
67	00127	77	00137

(—): PROGRAMMED INSTRUCTION Timing: 3

a	I	X	Y
24 19	18	17 16	15 1

where a is the Octal Code.

Place the address of the programmed instruction being performed in the base operand address portion of the memory location specified by the programmed instruction fixed address. Clear Bits 24-19, 17, and 16, and place a "1" in Bit 18 of the above specified memory location. Take the next instruction from the memory location sequentially following the one specified by the programmed instruction fixed address.

No indirect addressing or indexing will occur. The programmed instruction does not use the operand; therefore, address modification is not appropriate. If address modification is specified, it will take place in the subroutine to which the programmed instruction has transferred control.

SECTION FOUR
INPUT/OUTPUT

INSTRUCTING EXTERNAL DEVICES

The 6020 sends instructions to External Devices which tell those devices to initiate specified operations. After an External Device operation has been initiated by the 6020, all control of events passes to the logic associated with that External Device until the operation is completed. During the operation, the External Device will respond busy to attempts by the 6020 to initiate further operations. The sequencing of events during this operation derives its timing from the External Device and its logic.

All External Devices which are instructed by the EXD instruction or which use external device interrupts are connected to the 6020 by a "common cable" that carries an External Device address code and a code which specifies what operation is to be performed. Only that device whose address is on the lines will respond to an instruction on the common cable. No instruction will be initiated unless it is accompanied by a START signal. When a device recognizes its address and receives a START signal, it will store the essential information from the operation code in flip-flops and initiate the specified operation. When the operation is complete, the External Device will interrupt the computer program if it was instructed to do so by the operation code. Otherwise the External Device just becomes not busy when it has completed its operation. It is then available for further instruction.

When an External Device recognizes its address and is not busy, it sends a response on the NOT BUSY line to the 6020. If no such response is received, the 6020 will assume that the addressed device is busy. The 6020 will send a START signal only if a NOT BUSY response is received. If a device is disconnected, it will appear to be busy to the 6020.

EXD Instruction. The program instructs External Devices by using the EXD instruction which has the same format as other 6020 instructions (see Instruction Repertoire), but may have more than one operand. The last operand of the EXD instruction is the External Device Control Word (EDCW). Its format is shown in Figure 4-1.

ED Address	Last	Skip	Prevent Start	Interrupt	ED Operation Code
(6 bits)	(1 bit)	(1 bit)	(1 bit)	(1 bit)	(14 bits)
24 19	18	17	16	15	14 1

Figure 4-1. EDCW Format

Bits 24 through 19 of the EDCW are the 6-bit address of the External Device. Table 4-1 lists the standard addresses for the available External Devices. Addresses other than standard addresses may be ordered.

Table 4-1

STANDARD EXTERNAL DEVICE ADDRESSES

ADDRESS	EXTERNAL DEVICE
02	Paper Tape Reader
04	Paper Tape Punch
06	Card Reader
10	Card Punch
12	Line Printer
14	Plotter #1
16	Plotter #2
20	Magnetic Tape Unit #1
22	Magnetic Tape Unit #2
24	Magnetic Tape Unit #3
26	Magnetic Tape Unit #4

Note: Other External Devices will be assigned Addresses 30 to 77 as required. Addresses 60-77 are priority addresses and are available as options in groups of four.

Bit 18 of the EDCW will always be a "0" since it is always the last control word in an EXD instruction.

If Bit 17 of the EDCW is a "0," the next instruction in sequence will be skipped if the External Device is not busy. The programmer may enter a busy subroutine when the External Device is busy by placing a "0" in Bit 17 of the EDCW and by making the instruction following the EXD instruction a jump to the subroutine. The jump instruction will be skipped if the External Device is not busy. Similarly, if the programmer wishes to wait until the device is not busy, he may put a "0" in Bit 17 of the EDCW and follow the EXD instruction with a jump back to the EXD instruction. Thus, the 6040 will keep trying to execute the EXD instruction until the External Device becomes not busy. The START pulse will then be sent and the jump instruction will be skipped.

If Bit 16 is a "1," the START signal will be prevented regardless of the busy status of the External Device.

If Bit 15 is a "1," the External Device is instructed to interrupt the program when it has completed the specified operation. If Bit 15 is a "0," the External Device is not required to interrupt when it is done. In any case, fail interrupts may still occur if the appropriate conditions occur. Bit 15 is sent to all External Devices along with the Operation Code.

Bits 14 through 1 of the EDCW are the Operation Code that is sent to the External Device to specify what operation is to be performed. The operation code is interpreted by the particular device that is addressed. The same Operation Code may have different meanings to different devices. Table 4-2 lists Operation Codes of standard External Devices.

Table 4-2

OPERATION CODES FOR STANDARD EXTERNAL DEVICES

EXTERNAL DEVICE	COMMAND	CODE
Single Density Magnetic Tape Unit	Test End of Tape	00
	Test File Mark	01
	Test Fail	02
	Write Alpha	05
	Write File Mark	06
	Write Binary	07
	Move Reverse N Records	10
	Rewind	12
	Move Forward N Records	14
	Read Alpha	15
	Read Binary	17
	Test Busy	03
Multiple Density Magnetic Tape Unit	Test End of Tape	00
	Test File Mark	01
	Test Fail	02
	Test Busy	03
	Write Alpha	05
	Write File Mark	06
	Write Binary	07
	Backspace	10
	Reverse to File Mark	11
	Rewind	12
	Space	14
	Read Alpha	15
	Forward to File Mark	16
	Read Binary	17
Paper Tape Reader	Read Packed Mode	00
	Read Character Mode	01
Paper Tape Punch	Punch Packed Mode	00
	Punch Character Mode	01
High Speed Card Reader	Read Column Binary (multiple card)	00
	Read Hollerith (multiple card)	01
	Read Column Binary (single card)	04
	Read Hollerith (single card)	05
High Speed Card Punch	Punch Column Binary (multiple card)	00
	Punch Hollerith (multiple card)	01
	Punch Column Binary (single card)	04

Table 4-2 (cont.)

EXTERNAL DEVICE	COMMAND	CODE
	Punch Hollerith (single card)	05
Low Speed Card Reader	Read Column Binary (multiple card)	00
	Read Hollerith (multiple card)	01
	Read Column Binary (single card)	04
	Read Hollerith (single card)	05
Low Speed Card Punch	Punch Column Binary (multiple card)	00
	Punch Hollerith (multiple card)	01
	Punch Column Binary (single card)	04
	Punch Hollerith (single card)	05
Plotter	Test Busy	00
	Move +X	01
	Move —X	02
	Move +Y	04
	Move —Y	10
	Pen Up	20
	Pen Down	40

PROGRAM INTERRUPTS

Standard Interrupts. Various events can lead to a program interrupt. Each interrupt is to a unique fixed memory address which is associated with the event that caused it. Figure 4-2 shows the addresses which are reserved for these interrupts. Each External Device has an interrupt address which is equal to its External Device address. An External Device may have more than one interrupt event and each event may have its own interrupt address. Interrupts may occur only at the end of program instructions.

It is important to the programmer that each type of interrupt results in transfer of control to a different memory address. This makes it unnecessary for the program to scan interrupt events to see what has happened. A subroutine for each interrupt event may be in memory.

An add overflow interrupt takes precedence over non-priority External Device interrupts if they occur at the same time. The External Device interrupts will still occur but they will be delayed until a JDI has been executed. Only a priority External Device may interrupt an interrupt routine.

External Device interrupts may occur for any of several reasons. An External Device may interrupt when it has completed an operation if it was told to do so by the program when the operation was initiated. An External Device may interrupt at a different address to indicate that a failure (parity fail, end of tape, etc.) has occurred. Different devices have different failure conditions and different failure interrupt addresses. An External Device may be notifying the program that some specific real-time event has occurred.

Octal Address	
0.57	Non-Priority External Devices
60-77	Priority External Devices
100	Add Overflow Interrupt
101	Operator Interrupt
102	Memory Parity Fail Interrupt
103	Memory I/O Parity Fail Interrupt
104	Power Fail Interrupt
105	Programmed I/O Channel Interrupt
106	Typewriter Interrupt
110-111	Temporary storage used during execution of MPY, DAD, and XAK instructions
115	Index Register 1
116	Index Register 2
117	Index Register 3
120-137	Programmed Instruction Entries

Figure 4-2. Fixed Memory Addresses

Operator interrupt is initiated by a switch on the operator's console.

Add overflow is an event that may result from normal program instructions (see Instruction Repertoire).

Traps. For each type of interrupt there is a program controlled trap which may prevent or allow the corresponding interrupts. If a trap is in the "1" condition, the corresponding interrupt is allowed. If it is a "0," the corresponding interrupt is prevented. The following is a list of traps and their corresponding interrupt events:

Power Fail Trap	Power Fail Interrupt
Memory Parity Fail Trap	Memory Parity Fail Interrupt; I/O Memory Parity Fail Interrupt
Priority ED Trap	Priority ED Interrupts
Add Overflow Trap	Add Overflow Interrupt
ED Trap	Non-Priority ED Interrupts
Programmed I/O Trap	Programmed I/O Interrupt
Operator Trap	Typewriter Interrupt; Operator Interrupt

If the ED trap is a "1," standard External Device interrupts are allowed. Priority interrupts are prevented if the priority trap is a "0." Setting a trap to the "1" condition is sometimes referred to as "arming" the trap. Conversely, clearing a trap to the "0" condition may be called "disarming" the trap.

Scanner. External Devices are continuously scanned for interrupt requests. A scanner in the 6020 counts through the non-priority External Device addresses sequentially (at 5.7 microseconds per address), each time asking if the addressed device is requesting an interrupt. When an interrupt request is found, the scanner will stop at that address and attempt to interrupt the computer program. The only other time when the scanner stops scanning for interrupt requests is during an EXD instruction. The EXD instruction uses the same address lines as the scanner on a time sharing basis. Whenever an EXD instruction occurs, the scanner is stopped and switched off the cable lines, and the address from the EDCW is switched onto the lines. The process is reversed at the end of the EXD instruction.

External Devices respond with both their busy status and their interrupt request status whenever they recognize their own address. They do not clear out an interrupt request until the interrupt succeeds. The 6020 notifies an External Device that its interrupt has been recognized by sending out an interrupt reset signal along with the address of the interrupt that has been recognized.

Priority Interrupts. Priority ED interrupts differ from standard ED interrupts in the following respects:

1. Priority ED interrupts are optional.
2. There are 16 priority ED interrupts available in groups of four.
3. There can be 16 levels of priority with the lowest number (Address 60) having the highest priority.
4. Priority interrupts may interrupt standard interrupt programs and priority interrupt programs of lower priority.
5. Two priority internal interrupts, Power Fail and Memory Parity Fail are of higher priority than the priority external interrupts. Power Fail has the highest priority.
6. Each priority ED interrupt may be separately allowed or disallowed by the program since each has its own allow flip-flop.

BUFFERED INPUT/OUTPUT

Input/Output Channel. The standard mode of input/output data flow for the 6020 is by sequentially transmitted six-bit bioctal or alphanumeric characters. These characters are assembled (or disassembled) into 24-bit computer words by an input/output Assembly Register (B), one of which is employed for each input/output channel. In addition, each input/output channel is provided with two 15-bit address registers. These registers are called Buffer Memory Address Register (BM) and Buffer Limit Address Register (BL). (See Figure 4-3, Input/Output Channel Block Diagram.) BM and BL define the beginning and limit locations in memory into (from) which a block of data is to be transferred via the particular input/output channel.

These two address registers must be set to the correct values (beginning and ending locations of the data area) before an External Device is directed to begin a data transfer. BL should always be set to one greater than the address of the last memory location involved in the transfer.

During the actual data transfer, Registers B, BM, and BL are under control of the specified External Device. For input, the B register assembles six-bit characters sent by the External Device into 24-bit computer words and then transmits these words to memory locations. In an output operation, the B register receives 24-bit words from the core memory and disassembles these into six-bit characters and directs them to the External Device. The contents of the BM address register are advanced by one after to each memory transfer. Whenever the contents of BM are equal to the contents of BL, communication with the External Device is stopped as the specified amount of data has been communicated between the central processor and the External Device.

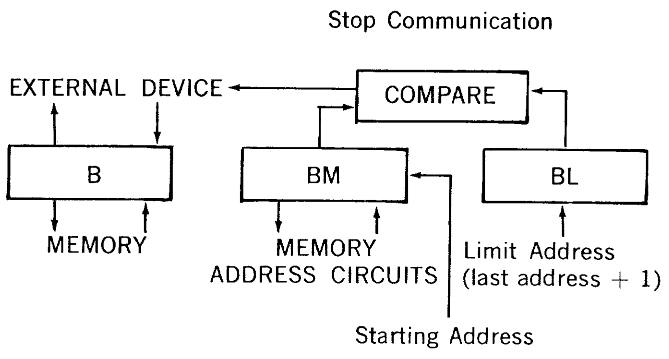

Figure 4-3. Input/Output Channel Block Diagram

Assembly Register Instruction. The operation of each input/output channel is controlled by the assembly register instruction (ASR). The ASR instruction has the same format as all other machine instructions (see Instruction Repertoire) and its operands are the Assembly Register Control Words (ARCW). The ARCW's may also occur in the EXD instruction. The format of the ARCW is shown in Figure 4-4.

Each ARCW is decoded and interpreted by the control circuits of the 6020 when an ASR instruction is performed. The ARCW contains the codes and control function designators that are necessary to accomplish control of the input/output channels. An explanation of each possible interpretation of an ARCW is given in the following paragraphs.

Channel No. (3 bits)	Operation Code (3 bits)	Last (1 bit)	Index (2 bits)	Base Operand Address (15 bits)
24 22	21 19	18	17 16	15 1

Figure 4-4. ARCW Format

The operation codes are as follows:

000	Load BM if channel is not busy
001	Load BL if channel is not busy
010	Store BM if channel is not busy
011	"Set BG" (See Field Transfer and Cyclic Transfer Channels)
100	Load BM regardless of busy status of channel
101	Load BL regardless of busy status of channel
110	Store BM regardless of busy status of channel
111	Unspecified

The index address portion of the ARCW operates in the same manner as the index address of the machine instructions. That is, the base operand address portion of the ARCW may be indexed to yield the effecitve operand address. Indirect addressing cannot be used in the ARCW as the indirect address bit (Bit 18) is used for another purpose.

The effective operand address is used as indicated by Bits 19-21 of the ARCW. The effective operand address is either placed in Register BM or BL of the specified channel or specifies the memory location where the contents of Register BM of the indicated channel will be stored.

If Bit 18 is "0" or if the operation code specifies "Store BM" (010 or 110) in an ARCW, the ARCW will be the last control word to be executed during the ASR instruction. If Bit 18 is a "0" in a control word for the EXD instruction, the control word will be interpreted as an EDCW. If Bit 18 is a "1" in **any** control word, it will be interpreted as an ARCW.

The ASR instruction will always cause a skip of the next sequential instruction if the channel specified in any ARCW is not busy. If the channel is busy, no skip will occur.

Data Flow. The transfer of data between External Devices and their associated assembly registers proceeds under the control of the External Device.

Input/Output channels may transfer data simultaneously in multi-channel operation. The access to main memory from the in/out channel is made available as needed, subject to channel priority. This is provided by channel priority logic which selects the channel of highest priority requesting transfer, that is, the lowest numbered channel requesting transfer of data. The data rate and channels active are not restricted except that the program must not require data transfer which exceeds a peak word rate of 525 kc considered over all input/output channels.

External Device action occurs only as a result of a computer External Device instruction. The amount of data transferred and the memory locations employed are previously determined by computer Assembly Register instructions which set BM and BL. Once an ED is placed in action, the information transfer is completely and irrevocably under the control of the ED until the operation is complete except that BM or BL may still be changed by the program. At the completion of the operation, the ED may cause an ED interrupt if appropriate.

Character Transfer. The normal mode of communication is by sequential transmission of 6-bit (in parallel) characters with provision for computer specification of the number of words to be transferred.

The basic operation of the character input communication cycle is as follows: After an External Device has been addressed and commanded to perform a specific input operation by an ED instruction, four characters are input to the assembly register of the communication channel. The word is then stored in memory at the location specified by Register BM.

This operation continues with a memory reference being made at each fourth character until BM = BL or the external device signals the end of its information. When BM = BL, the channel signals the external device with a complete signal and no further information is sent to the computer. If the external device has no more information, it sends a transfer signal to the channel and the word or partial word is placed into memory and the communication cycle is terminated.

The output operation occurs in about the same manner as input. The differences are: (1) a memory reference is made prior to the first character output and (2) the complete signal appears when BM = BL and the last character of the last word is sent to the external device.

Word Transfer. For applications requiring more generality or increase in speed of data transmission, three types of word channels are available. Transmission is of entire words (24 bits in parallel). The word transfer feature is optional and not part of the standard 6020.

The three types of word channels that are available are: word in, word out, and word in-and-out. Only with the word in-and-out channel is character transfer included to permit standard character communications to take place. The word in channel is for input only, while the word out is for output only.

The operation of word transfer is the same as for character transfer except that 24 bits are transferred between the computer and the external device at one time. This means that no breakdown of words into characters on output or assembling of characters into words on input takes place. This provides an increase of four times the input/output speed over character transfer assuming the external device can accept or send words at that rate.

Field Transfer. The field channel is composed of a character channel and a field counter. The channel operates as a character channel unless the field flip-flop is set. An ASR instruction called "Set BG" (see Figure 4-5) is used to set

Channel No. (3 bits)	Operation Code "0" "1" "1"	Last (1 bit)	Index (2 bits)	Busy (1 bit)	Field Flip-Flop (1 bit)		Field Count (4 bits)
24 22	21 20 19	18	17 16	15	14	13 5	4 1

Figure 4-5. "Set BG" format for Field Channel

the field flip-flop. If Bit 15 is a "1," Bit 14 is recognized regardless of busy status. If Bit 15 is a "0," Bit 14 is recognized only if the channel is not busy.

In operation, transfer is the same as for character transfer. However, when the last character of the "field" word has been processed, as determined by the contents of the Field Length Register, the remaining bits of the word in memory are not filled. The first character of the next field word is placed in the six most significant bits of the next memory location. The result is that each field word in the memory has the same pattern, the same number of locations and always begins with the most significant bit of a memory location. The field word may be from one to 15 characters long.

Cyclic Transfer. This optional channel provides continuous cyclic word communication with from one to four equal-length blocks of memory. The blocks are contiguous and contain from one to 4,096 words. Both input and output modes are accommodated. The cyclic feature of this channel is beneficial in such applications as telemetry or other high-speed repetitive operations because once initiated, this channel continues to function without the need of a program instruction to start each cycle.

The Load BM instruction loads Registers BM and BF with the same address. BM is the current address register and always holds the address of the next memory location to be referenced. BF is only changed by the Load BM instruction and therefore contains the address of the first memory location to be referenced. From the Load BL instruction Bits 12-1 go to the BZ (Size) register, Bits 14-13 go to the BK (Block) register, and Bit 15 to the BINT (Interrupt Enable) flip-flop. An ASR instruction called "Set BG" (see Figure 4-6) is used to set or clear the stop flip-flop. Bit 13 prevents the channel from communicating when it is a "1." If Bit 15 is a "1," Bit 13 is recognized regardless of busy status. If Bit 15 is a "0," Bit 13 is recognized only if the channel is not busy.

When the stop flip-flop is set and the end of a block is reached, a complete signal is sent to the external device and the channel is prevented from further communication until a "Load BM" or a "Clear Stop Flip-Flop" code is generated by an ASR instruction. If the BINT flip-flop is set, a priority interrupt will be sent to the central processor at the end of a block.

Channel No. (3 bits)	Operation Code "0" "1" "1"	Last (1 bit)	Index (2 bits)	Busy Status (1 bit)		Stop Flip-Flop (1 bit)	
24 22	21 20 19	18	17 16	15	14	13	12 1

Figure 4-6. "Set BG" format for Cyclic Channel

BZ is composed of a 12-bit register and a 12-bit counter. The contents of the BZ register is the number of words in a block of information, while the contents of the BZ counter at a given time is the number of words left to be transmitted in that particular block. The contents of the BZ register can only be changed by the Load BL instruction.

BK is composed of a two-bit register and a two-bit counter. The contents of the BK register is the number of blocks in a cycle, while the contents of the BK counter at a given time is the number of blocks left to be transmitted in that particular cycle. The contents of the BK register can only be changed by the Load BL instruction.

Every memory reference increments the BM and BZ counters. When the BZ counter equals the BZ register, the BK counter is incremented while the BZ counter is cleared. When the BZ counter equals the BK register, the contents of BF is transferred to the BM register and the BZ and BK counters are cleared. At this point, one complete cycle has been completed and the second cycle is just starting. All subsequent cycles are identical to the first cycle unless a Load BM or Load BL instruction is given to modify information contained in the cyclic channel registers.

Pressing the Initial Clear button causes the cyclic channel to cease memory reference and prevents further communication until a Load BM or a "Clear Stop Flip-Flop" code is recognized by the channel.

DIRECT COMMUNICATION

Up to 16 bits of data may be sent directly to any External Device as an operation code. This direct communication is under direct program control and requires no buffered communication channel.

The operation code may also be used to specify one of up to 16,000 control lines to be signaled. Here again, no buffered channel is required and the communication is under direct program control.

An optional programmed input/output channel is available which is independent of all the buffered I/O channels. Four computer instructions allow direct input and output of words containing 24 bits or less to and from the accumulator under program control. This input and output can be programmed to occur regardless of external conditions or with a "wait" for an external READY signal.

SECTION FIVE
OPERATOR'S CONSOLE

DISPLAY REGISTER

There is a 25-bit binary display on the 6020's operator console. Display Bit 25 indicates the memory word parity, and Display Bits 24 through 1 may indicate:

(1) the next instruction,

(2) the contents of any memory location,

(3) the contents of Register A (Accumulator),

(4) the contents of Register E,

(5) the contents of Index Memory Location No. 1,

(6) the contents of Index Memory Location No. 2,

(7) the contents of Index Memory Location No. 3, or

(8) the status of the traps.

When the computer halts, the Display Register will indicate the next instruction word, while Register T will contain the address of the halt instruction that stopped the computer. To display anything else, the appropriate push-button display switch must be pressed.

When the DISPLAY M button is pressed, the contents of the memory location specified by Register T will be displayed. When either the DISPLAY A, E, 1, 2, 3, or TRAPS button is pressed; the contents of Register A; Register E; Index Memory Locations 1, 2, or 3; or the condition of the traps is displayed; respectively.

The Display Register is also used as an entry register. Push-buttons are provided which may clear all or various parts of the Display Register. The contents of the Display Register may be entered in any memory location; Register A; Register E; or Index Memory Locations 1, 2, or 3.

Pushing any of the ENTRY buttons will accomplish entry of the Display Register contents into the appropriate memory location or register. To enter a number in Register A, for example, the operator must push the clear button for the whole register, press those set buttons which correspond to the bits where "1's" are desired in the number, and then press the ENTER A button.

REGISTERS T AND S

Two address registers are also displayed on the operator's console. Register T is a temporary memory address register. Register S is the program sequence register which specifies the address of the next instruction. These registers may be set to any desired value by push-button switches.

A clear button is associated with each of these registers to clear the entire register. Each bit of each register may be set by a particular button. Setting either register is accomplished by first clearing the register and then pushing the buttons that correspond to the bits where "1's" are desired.

INDICATORS

There are 10 display lights on the operator's console which indicate various conditions of the computer system that may be of interest to the operator. A list of the indicators and their functions follows:

1. **MEMORY ALARM.** Indicates that the memory temperature is out of limits.
2. **ED FAIL.** Indicates that some external device has failed. (Fail indicator lights are also located on each external device.)

3. **ED OPERATE.** Indicates that some external device operations are in progress.
4. **FLAG 1 and FLAG 2.** These lights are controlled by the STP instruction. They may be used to indicate anything the programmer wishes them to indicate and may be used to indicate different things in different programs.
5. **INTERRUPT and PRIORITY INTERRUPT.** Indicate that interrupt or priority interrupt routines are in progress. These must be clear before additional standard interrupts may occur.
6. **MEMORY FAIL.** Indicates an unrecognized memory fail. (Recognized memory fails lead to interrupts.)
7. **POWER FAIL.** Indicates that a momentary power fail has occurred.
8. **I/O PARITY FAIL.** Indicates that a parity failure has been detected in input/output communication.

CONTROLS

Push-button switches control power turn-on and turn-off. The power turn-on sequence is built-in and is triggered by the power ON switch. The power turn-off sequence is also built-in and includes timing so that any memory cycle that has been initiated will be completed. The power OFF switch triggers these events.

The RUN button will cause the program to start by taking its first instruction from the address specified by Register S. Thereafter, the computer will follow sequential instructions unless these instructions specify otherwise. When the computer is halted, Register S may be set to any desired value as described above.

The HALT button will halt the computer program but will not interrupt input/output communication. When the program halts as a result of pressing this button, it will halt at the completion of the instruction in progress. The next instruction following the halt will be in the Display Register.

Any one of four separate external devices may be selected for presetting with the PRESET SELECT control. The positions are labeled PT (Address 02), CD (Address 06), MT 20 (Address 20), and MT 30 (Address 30). If a device other than the one listed is using that particular address, it can be used for presetting if it has that capability. For instance, if a magnetic tape unit has Address 02, it can be used for presetting by selecting "PT."

The PRESET button will cause a block of information from the specified external device to be loaded into the memory beginning with the address in Register S which may be set to any memory address by use of the console buttons. This switch is armed by pressing the CLEAR button.

The CLEAR switch will halt the computer program and all external device operation. Normally this button will not be pressed until the ED OPERATE light indicates the input/output communication is complete.

An operator interrupt switch (OPERATOR) will cause the program to be interrupted and to go to the Operator Interrupt Subroutine if the operator interrupt trap is armed. If the operator trap is not armed, the OPERATOR switch will be ignored. The operator interrupt trap is controlled by the program which may arm it or disarm it with a trap instruction (STP, CTP, etc). If the trap is armed, the OPERATOR indicator is lit.

The STEP button allows the 6020 to be operated in one of two modes. The normal mode of operation (STEP indicator unlit) is continuous. In this mode, the computer will halt only as a result of a HALT instruction, an error halt, or the HALT button. The other mode of operation (STEP indicator lit) is one-instruction. In this mode, the computer will halt after each instruction is performed. This is useful in code checking and other special cases.

When the central processor is in the SCAN mode as indicated by the illuminated SCAN indicator, pressing the RUN button will not cause the central processor to run but will transfer the contents of Register S to Register T. The memory location specified by Register T will be shown in the Display Register and the contents of Register S will be incremented. Only one memory location will be displayed each time the RUN button is pressed regardless of the state of STEP push-button.

Six SENSE buttons mounted on the control console, numbered 1 through 6, may be used to cause branching of the computer program when the sense switch instruction is given.

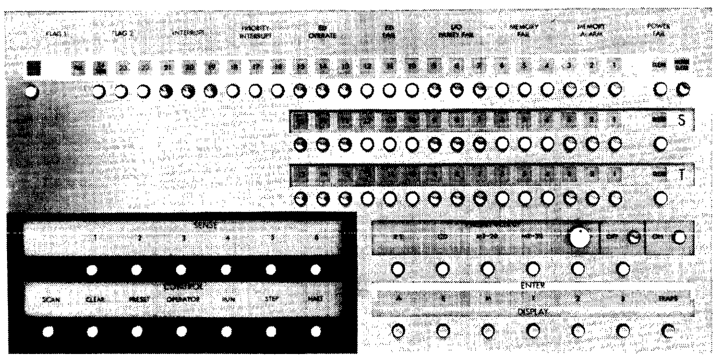

Figure 5-1. 6020 Operator's Console

APPENDIX

LIST OF INSTRUCTIONS

A. Instruction Arranged by Octal Codes

Octal Code	Mnemonic Code	Description	Timing	Section 3 Page Reference
01	LDA	Load A	2	3-2
02	LDE	Load E	2	3-2
03	DLD	Load AE	3	3-2
04	STA	Store A	2	3-2
05	STE	Store E	3	3-3
06	DST	Store AE	3	3-3
07	LOA	Load Address	3	3-5
10	SAM	Store A Address	3	3-3
11	ADD	Add	2	3-3
12	MPY	Multiply	16	3-3
13	SUB	Subtract	2	3-3
14	DVD	Divide	25; 3 if divide fault	3-4
15	DAD	Double Add	6	3-4
17	ADM	Add to Memory	4	3-4
20	RSA	Right Shift A	2 + k	3-7
20	LSA	Left Shift A	2 + k	3-7
20	CSA	Circular Left Shift A	2 + k	3-7
20	RLA	Logical Right Shift A	2 + k	3-7
20	LLA	Logical Left Shift A	2 + k	3-8
20	RSE	Right Shift E	2 + k	3-8
20	LSE	Left Shift E	2 + k	3-8

20	CSE	Circular Left Shift E	2 + k	3-8
20	RLE	Logical Right Shift E	2 + k	3-8
20	LLE	Logical Left Shift E	2 + k	3-8
20	RSD	Right Shift AE	2 + k	3-8
20	LSD	Left Shift AE	2 + k	3-8
20	CSD	Circular Left Shift AE	2 + k	3-8
20	RLD	Logical Right Shift AE	2 + k	3-8
20	LLD	Logical Left Shift AE	2 + k	3-9
20	NOP	No Operation	2 + k	3-15
21	JMP	Jump	1	3-10
22	XEC	Execute	1 + CYC	3-10
23	JDI	Jump Disable Interrupt	2	3-10
24	JPO	Jump to M + 1	2	3-10
25	RTJ	Return and Jump	4	3-10
26	ASR	Assembly Register	2 to 7	3-13
27	EXD	External Device	6	3-14
30	JLZ	Jump if $A < 0$	2 if jump; 3 if no jump	3-11
31	JGZ	Jump if $A > 0$	2 if jump; 3 if no jump	3-11
32	JEZ	Jump if $A = 0$	2 if jump; 3 if no jump	3-11
34	KAL	Skip A Low	4	3-11
35	KAH	Skip A High	4	3-11
36	KAQ	Skip A Equal	4	3-11
37	IMT	Increment Memory and Test	4	3-11
40	AND	Logical "And"	2	3-7
41	XOR	Exclusive "Or"	2	3-7
42	LOR	Logical "Or"	2	3-7
43	AMA	Add Magnitude	2	3-4
50	NOP	No Operation	2	3-15
50	ZOA	Zero A	2	3-5
50	ZOE	Zero E	2	3-5
50	ZOD	Zero AE	2	3-5
50	ZSA	Clear A to Sign of E	2	3-5
50	CMA	Complement A	3	3-5
50	CME	Complement E	3	3-6
50	CMD	Complement AE	3	3-6
50	MNA	Minus A	2	3-6
50	MNE	Minus E	2	3-6
50	MND	Minus AE	2	3-6
50	AVA	Absolute Value A	2	3-6
50	AVE	Absolute Value E	2	3-6
50	AVD	Absolute Value AE	2	3-6
50	SAE	Sign of A to E	2	3-6

50	SEA	Sign of E to A	2	3-6
50	ATE	Copy A in E	2	3-6
50	ETA	Copy E in A	2	3-6
50	XAE	Exchange AE	2	3-7
50	RND	Round	2	3-5
50	XAK	External Lines to A and Skip if Ready	4 if skip; 3 if no skip	3-14
50	AXK	A to External Lines and Skip if Ready	4 if skip; 3 if no skip	3-15
50	NRM	Normalize	6 + k	3-9
51	SEN	Send External Signal	3	3-15
51	CTP	Clear Flip-Flops	2	3-12
51	HLT	HALT		3-11
51	STP	Set Flip-Flops	2	3-12
51	KIF	Conditional Skip	2 if no skip; 3 if skip	3-11
51	KEX	Skip on External Signal	2 if no skip; 3 if skip	3-12
51	SCT	Save and Clear Flip-Flops	2	3-13
51	SST	Save and Set Flip-Flops	2	3-13
52	AOA	Add Address	2	3-4
52	SOA	Subtract Address	2	3-5
52	LXP	Load Exponent	2	3-2
52	SXP	Store Exponent	4	3-3
53	SXM	Store Index	4	3-9
53	ZOM	Zero Memory	4	3-9
54	AUX	Augment Index	4	3-9
54	KOZ	Skip if Base Operand Address is Zero	4	3-9
54	JDX	Jump and Decrement Index	4	3-9
55	LOX	Load Address in Index	2	3-9
55	LAX	Load A Address in Index	2	3-10
56	IXT	Increment Index and Test	4 if skip; 5 if no skip	3-10
56	KON	Skip if Base Operand Address is one	4 if skip; 5 if no skip	3-10
57	KXH	Skip Index High	2 if skip; 3 if no skip	3-10

B. Instructions Arranged by Mnemonic codes

Octal Code	Mnemonic Code	Description	Timing	Section 3 Page Reference
11	ADD	Add	2	3-3
17	ADM	Add to Memory	4	3-4
43	AMA	Add Magnitude	2	3-4
40	AND	Logical "And"	2	3-7
52	AOA	Add Address	2	3-4

26	ASR	Assembly Register	2 to 7	3-13
50	ATE	Copy A in E	2	3-6
54	AUX	Augment Index	4	3-9
50	AVA	Absolute Value A	2	3-6
50	AVD	Absolute Value AE	2	3-6
50	AVE	Absolute Value E	2	3-6
50	AXK	A to External Lines and Skip if Ready	3 if no skip; 4 if skip	3-15
50	CMA	Complement A	3	3-5
50	CMD	Complement AE	3	3-6
50	CME	Complement E	3	3-6
20	CSA	Circular Left Shift A	2 + k	3-7
20	CSD	Circular Left Shift AE	2 + k	3-8
20	CSE	Circular Left Shift E	2 + k	3-8
51	CTP	Clear Flip-Flops	2	3-12
15	DAD	Double Add	6	3-4
03	DLD	Load AE	3	3-2
06	DST	Store AE	3	3-3
14	DVD	Divide	25; 3 if divide fault	3-4
50	ETA	Copy E in A	2	3-6
27	EXD	External Device	6	3-14
51	HLT	HALT		3-11
37	IMT	Increment Memory and Test	4	3-11
56	IXT	Increment Index and Test	5 if skip; 4 if no skip	3-10
23	JDI	Jump Disable Interrupt	2	3-10
54	JDX	Jump and Decrement Index	4	3-9
32	JEZ	Jump if A = 0	2 if jump; 3 if no jump	3-11
31	JGZ	Jump if A > 0	2 if jump; 3 if no jump	3-11
30	JLZ	Jump if A < 0	2 if jump; 3 if no jump	3-11
21	JMP	Jump	1	3-10
24	JPO	Jump to M + 1	2	3-10
35	KAH	Skip A High	4	3-11
34	KAL	Skip A Low	4	3-11
36	KAQ	Skip A Equal	4	3-11
51	KEX	Skip on External Signal	3 if skip; 2 if no skip	3-12
51	KIF	Conditional Skip	3 if skip; 2 if no skip	3-11
56	KON	Skip if Base Operand Address is one	5 if skip; 4 if no skip	3-10
54	KOZ	Skip if Base Operand Address is Zero	4	3-9
57	KXH	Skip Index High	2 if skip; 3 if no skip	3-10
55	LAX	Load A Address in Index	2	3-10
01	LDA	Load A	2	3-2
02	LDE	Load E	2	3-2

20	LLA	Logical Left Shift A	2 + k	3-8
20	LLD	Logical Left Shift AE	2 + k	3-9
20	LLE	Logical Left Shift E	2 + k	3-8
07	LOA	Load Address	2	3-5
42	LOR	Logical "Or"	2	3-7
55	LOX	Load Address in Index	2	3-9
20	LSA	Left Shift A	2 + k	3-7
20	LSD	Left Shift AE	2 + k	3-8
20	LSE	Left Shift E	2 + k	3-8
52	LXP	Load Exponent	2	3-2
50	MNA	Minus A	2	3-6
50	MND	Minus AE	2	3-6
50	MNE	Minus E	2	3-6
12	MPY	Multiply	16	3-3
20	NOP	No Operation	2 + k	3-15
50	NOP	No Operation	2	3-15
50	NRM	Normalize	6 + k	3-9
20	RLA	Logical Right Shift A	2 + k	3-7
20	RLD	Logical Right Shift AE	2 + k	3-8
20	RLE	Logical Right Shift E	2 + k	3-8
50	RND	Round	2	3-5
20	RSA	Right Shift A	2 + k	3-7
20	RSD	Right Shift AE	2 + k	3-8
20	RSE	Right Shift E	2 + k	3-8
25	RTJ	Return and Jump	4	3-10
50	SAE	Sign of A to E	2	3-6
10	SAM	Store A Address	3	3-3
51	SCT	Save and Clear Flip-Flops	2	3-13
50	SEA	Sign of E to A	2	3-6
51	SEN	Send External Signal	3	3-15
52	SOA	Subtract Address	2	3-5
51	SST	Save and Set Flip-Flops	2	3-13
04	STA	Store A	2	3-2
05	STE	Store E	3	3-3
51	STP	Set Flip-Flops	2	3-12
13	SUB	Subtract	2	3-3
53	SXM	Store Index	4	3-9
52	SXP	Store Exponent	4	3-3
50	XAE	Exchange AE	2	3-7
50	XAK	External Lines to A and Skip if Ready	3 if no skip; 4 if skip	3-14
22	XEC	Execute	1 + CYC	3-10

50	XTA	External Word to A	3	3-7
50	ZOA	Zero A	2	3-5
50	ZOD	Zero AE	2	3-5
50	ZOE	Zero E	2	3-5
53	ZOM	Zero Memory	4	3-9
50	ZSA	Clear A to Sign of E	2	3-5

COMMONLY USED CONSTANTS

Factorials	Decimal	Octal
1!	1	1
2!	2	2
3!	6	6
4!	24	30
5!	120	170
6!	720	1 320
7!	5 040	11 660
8!	40 320	116 600
9!	362 880	1 304 600
10!	3 628 800	15 657 400
11!	39 916 800	230 212 400
12!	479 001 600	3 443 176 000
13!	6 227 020 800	56 312 146 000
14!	87 178 291 200	1 211 416 624 000

Powers of 10_{10}

Decimal	Octal
10^0	+ 000000 000001
10^1	+ 000000 000012
10^2	+ 000000 000144
10^3	+ 000000 001750
10^4	+ 000000 023420
10^5	+ 000000 303240
10^6	+ 000003 641100
10^7	+ 000046 113200
10^8	+ 000575 360400
10^9	+ 007346 545000
10^{10}	+ 112402 762000
10^{-1}	+.063146 314631 463146 314631
10^{-2}	+.005075 341217 270243 656050
10^{-3}	+.000406 111564 570651 767635
10^{-4}	+.000032 155613 530704 145451
10^{-5}	+.000002 476132 610706 643604
10^{-6}	+.000000 206157 364055 366615
10^{-7}	+.000000 015327 745152 745364
10^{-8}	+.000000 001257 143561 060430
10^{-9}	+.000000 000104 560276 404665
10^{-10}	+.000000 000006 676337 663536
10^{-11}	+.000000 000000 537657 770274

Miscellaneous Constants	**Decimal**	**Octal**
$\sqrt{2}$	1.414 213 562 4	1.324 047 463 201
$\sqrt{3}$	1.732 050 807 6	1.566 636 564 132
$\sqrt{5}$	2.236 067 977 5	2.170 673 633 460
$\sqrt{6}$	2.449 489 742 8	2.346 107 024 023
$\sqrt{7}$	2.645 751 311 1	2.512 477 651 650
$\sqrt{8}$	2.828 427 124 8	2.650 117 146 402
$\sqrt{10}$	3.162 277 660 2	3.123 054 072 667
π	3.141 592 653 6	3.110 375 524 211
2π	6.283 185 307 1	6.220 773 250 413
$1/\pi$	.318 309 886 2	.242 763 015 564
$1/2\pi$	.159 154 943 1	.121 371 406 672
$1° = 1/360$ of a circle	.002 777 777 8	.001 330 133 015
e	2.718 281 828 5	2.557 605 213 053
1/e	.367 879 441 2	.274 265 306 615
$\log_{10}e$	.434 294 481 9	.336 267 542 512
$\log_e 10$	2.302 585 093 0	2.232 730 673 553
$\log_e 2$	.693 147 180 6	.542 710 277 600
$\log_{10}\pi$	.497 149 872 7	.376 424 666 307
$\log_e \pi$	1.144 729 885 8	1.112 064 044 344

1°=.017 453 292 5 radians

Reciprocals

n	1/n (octal)	1/n (decimal)
2	+400000000000	+.5000000000
3	+252525252525	+.3333333333
4	+200000000000	+.2500000000
5	+146314631463	+.2000000000
6	+125252525252	+.1666666667
7	+111111111111	+.1428571428
8	+100000000000	+.1250000000
9	+070707070707	+.1111111111
10	+063146314631	+.1000000000
11	+056427213505	+.0909090909
12	+052525252525	+.0833333333
13	+047304730473	+.0769230769
14	+044444444444	+.0714285714
15	+042104210421	+.0666666667
16	+040000000000	+.0625000000
17	+036074170360	+.0588235294
18	+034343434343	+.0555555555
19	+032745032745	+.0526315789
20	+031463146314	+.0500000000
21	+030303030303	+.0476190476
22	+027213505642	+.0454545454
23	+026205441310	+.0434782609
24	+025252525252	+.0416666667
25	+024365605075	+.0400000000
26	+023542354235	+.0384615384
27	+022755022755	+.0370370370
28	+022222222222	+.0357142857
29	+021517345410	+.0344827586
30	+021042104210	+.0333333333

Negative Powers of 2 and 8

m	n	2^{-n} and 8^{-m}
0	0	1.0
	1	0.5
	2	0.25
1	3	0.125
	4	0.062 5
	5	0.031 25
2	6	0.015 625
	7	0.007 812 5
	8	0.003 906 25
3	9	0.001 953 125
	10	0.000 976 562 5
	11	0.000 488 281 25
4	12	0.000 244 140 625
	13	0.000 122 070 312 5
	14	0.000 061 035 156 25
5	15	0.000 030 517 578 125
	16	0.000 015 258 789 062 5
	17	0.000 007 629 394 531 25
6	18	0.000 003 814 697 265 625
	19	0.000 001 907 348 632 812 5
	20	0.000 000 953 674 316 406 25
7	21	0.000 000 476 837 158 203 125
	22	0.000 000 238 418 579 101 562 5
	23	0.000 000 119 209 289 550 781 25
8	24	0.000 000 059 604 644 775 390 625
	25	0.000 000 029 802 322 387 695 312 5
	26	0.000 000 014 901 161 193 847 656 25
9	27	0.000 000 007 450 580 596 923 828 125
	28	0.000 000 003 725 290 298 461 914 062 5
	29	0.000 000 001 862 645 149 230 957 031 25
10	30	0.000 000 000 931 322 574 615 478 515 625
	31	0.000 000 000 465 661 287 307 739 257 812 5
	32	0.000 000 000 232 830 643 653 869 628 906 25
11	33	0.000 000 000 116 415 321 826 934 814 453 125
	34	0.000 000 000 058 207 660 913 467 407 226 562 5
	35	0.000 000 000 029 103 830 456 733 703 613 281 25
12	36	0.000 000 000 014 551 915 228 366 851 806 640 625
	37	0.000 000 000 007 275 957 614 183 425 903 320 312 5
	38	0.000 000 000 003 637 978 807 091 712 951 660 156 25
13	39	0.000 000 000 001 818 989 403 545 856 475 830 078 125

Positive Powers of 2 and 8

m	n	2^n and 8^m	m	n	2^n and 8^m
0	0	1	7	21	2 097 152
	1	2		22	4 194 304
	2	4		23	8 388 608
1	3	8	8	24	16 777 216
	4	16		25	33 554 432
	5	32		26	67 108 864
2	6	64	9	27	134 217 728
	7	128		28	268 435 456
	8	256		29	536 870 912
3	9	512	10	30	1 073 741 824
	10	1 024		31	2 147 483 648
	11	2 048		32	4 294 967 296
4	12	4 096	11	33	8 589 934 592
	13	8 192		34	17 179 869 184
	14	16 384		35	34 359 738 368
5	15	32 768	12	36	68 719 476 736
	16	65 536		37	137 438 953 472
	17	131 072		38	274 877 906 944
6	18	262 144	13	39	549 755 813 888
	19	524 288			
	20	1 048 576			

ASI

8001 Bloomington Freeway, Minneapolis, Minnesota 55420 / Division Electro-Mechanical Research, Inc.

SALES OFFICES — **BOSTON** 272-5400, 1 Garfield Circle, Burlington • **COCOA BEACH, FLA.** 783-4903, Suite 1, Holiday Office Center, 1325 N. Atlantic Avenue • **DALLAS** DI 8-4170, Room 104, First Federal Building, 440 Northlake Center • **DENVER** SU 9-1834, 3600 S. Lincoln, Englewood • **EL PASO** 751-2344, 8888 Dyer St. • **HOUSTON** MI 4-1856, 7135 Office City Drive • **HUNTSVILLE, ALA.** 881-4822, 3322 S. Memorial Parkway • **LOS ANGELES** ST 2-7030, 15551 Cabrito Road, Van Nuys • **MINNEAPOLIS** 888-9581, 8001 Bloomington Freeway • **NEW YORK CITY** 679-5954, 475 Fifth Avenue • **SUNNYVALE, CALIF.** 245-3694, 505 West Olive • **WASHINGTON, D. C.** 864-6340, 5012 College Avenue, College Park, Md.

FORM NO. 406021 REV. 1

LITHO IN U.S.A. 4-65