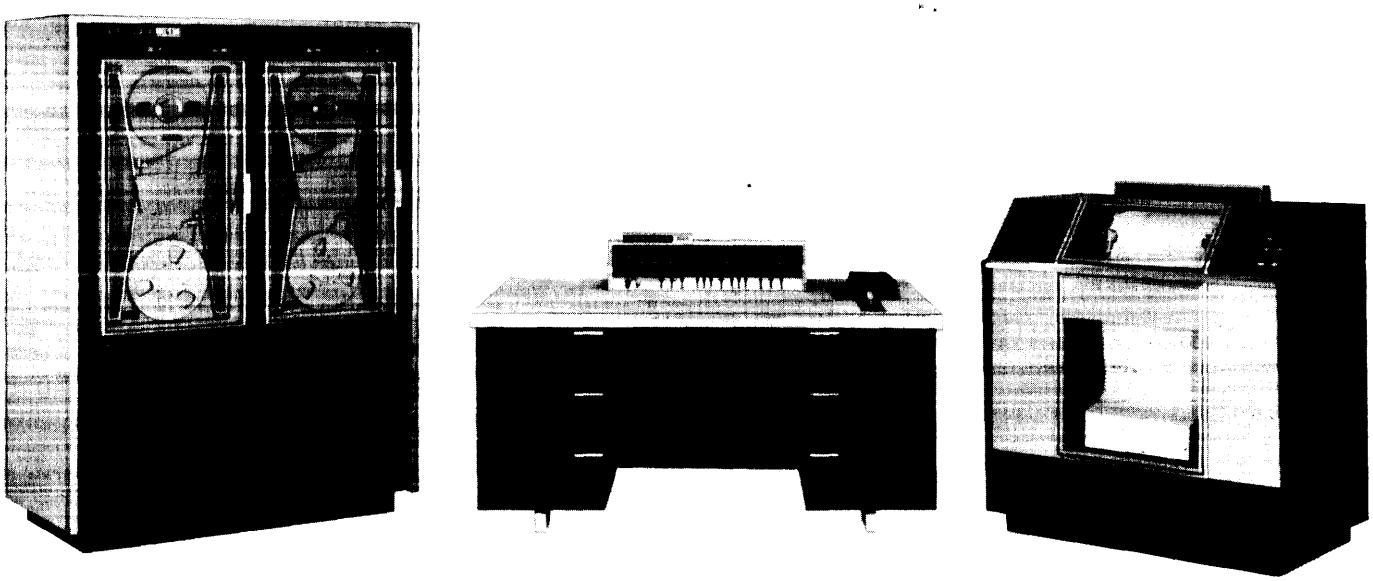
CONTROL DATA

160-A COMPUTER

160-A

INTERFOR/REFERENCE MANUAL

CONTROL DATA 160-A COMPUTER



INTERFOR/REFERENCE MANUAL

CONTROL DATA CORPORATION
8100 34th Avenue South
Minneapolis 20, Minnesota

SEPTEMBER 1962
Pub. No. 512

PREFACE

The INTERFOR programming system for the Control Data Corporation 160-A computer is a joint development of Control Data Corporation and Mr. Gordon Stanley of General Motors Corporation*. Mr. Stanley provided flow charts and documentation for the INTERFOR system, and developed the matrix instructions.

Control Data Corporation gratefully acknowledges the assistance of Mr. Stanley in this development.

*Aerospace Operations Group
Defense Research Laboratories,
Goleta, California

TABLE OF CONTENTS

	Page
INTRODUCTION	
PART I	
INTERFOR SYSTEM DESCRIPTION	1
Interpreter Routine	1
PART II	
INTERFOR WORD STRUCTURE	5
Memory Allocation	5
Instruction Word Format	6
Data Format	11
PART III	
INTERFOR INSTRUCTIONS	17
Data Transmission	19
Address Modification	21
Floating Point Arithmetic	21
No Address	23
Jumps and Stops	23
Storage Test	26
Storage Search	26
PART IV	
INTERFOR SUBROUTINES	29
Internal I/O Subroutines	29
Machine Language Subroutines	33

Page

PART V

INTERFOR ASSEMBLY PROGRAM (FLAP) 37

Symbolic Instruction Format 39

FLAP Pseudo Instructions 44

Machine Language Coding 45

Instruction Word Pairing 46

FLAP Input 48

FLAP Output 50

FLOADER 52

PART VI

INTERFOR OPERATOR'S GUIDE 53

FLAP Operating Procedures 53

FLOADER Operating Procedures 55

Interpreter Operating Procedures 57

APPENDIX

EXTERNAL FUNCTION SUBROUTINES 65

TABLES

TABLE I

Interpreter Pseudo Registers 2

TABLE II

Summary of Instruction Codes Accepted by FLAP 38

TABLE III

Flexowriter and Punched Card Characters Accepted by FLAP 43

INTRODUCTION

The purpose of the INTERFOR system is to facilitate the solution of scientific and engineering problems on a basic Control Data Corporation 160-A computer with up to four banks of core storage. The following on-line peripheral equipment will increase the usefulness of the INTERFOR system: A 161 typewriter, a 166 line printer and a 167 card reader. A 165 plotter may also be included.

The system, based on a 33-bit floating point interpreter, provides the programmer with up to 6521_8 48-bit words of storage and a repertoire of 22 single address instructions.

The INTERFOR system consists of the interpreter routine, the assembly program (FLAP), the binary load routine (FLOADER), nine function subroutines, and a PLOT subroutine for the Control Data 165 plotter. The interpreter routine is described in parts I, II and IV; INTERFOR word formats are in part II; the instruction repertoire is described in part IV. FLAP and FLOADER are described in part V. All operating instructions are in part VI. The function and PLOT subroutines are described in the appendix, which also includes calling sequences and octal listings.

PART I

INTERFOR SYSTEM DESCRIPTION

The INTERFOR system provides the programmer with a means of solving problems in floating point notation on the Control Data 160-A computer. An INTERFOR program is executed under control of the interpreter routine and may be coded either in octal or in symbolic format. Octal instructions are directly loaded and executed by the interpreter routine. Symbolic instructions are first assembled by FLAP, the INTERFOR assembly program to produce a listable output and a binary object program tape. The listable output, which may be on paper tape or on the 166 line printer, consists of a side-by-side listing showing the symbolic input and the assembled program. The binary object program tape is first loaded by FLOADER, the INTERFOR binary load routine before being executed by the interpreter.

Provision is made to include 160-A instructions within INTERFOR programs. The external function subroutines provided with the system simplify the computing of various mathematical functions. These subroutines utilize a portion of the interpreter routine but are loaded separately as described in the Appendix. See part V for a detailed description of FLAP and FLOADER.

INTERPRETER ROUTINE

The interpreter routine simulates twenty Control Data 1604 floating point instructions and two floating point matrix operations, using the Control Data 160-A computer. Each data word is converted into a 33-bit signed binary fraction and a 10-bit signed binary exponent. Data words and instruction words each occupy 48 bits of storage in the 160-A (four 160-A words). Each 48-bit word is addressed by a 4-digit octal number called the INTERFOR address. Since arithmetic is done in binary floating-point format, a pseudo accumulator is built into the interpreter routine. To implement the indexing features provided by several of the INTERFOR instructions, certain

locations in low core of bank 0 in the 160-A are used as pseudo index registers. Table I identifies the contents of low core storage locations (00-77) in the 160-A that are utilized by the interpreter. The argument locations are identified by X, the pseudo accumulator locations are identified by A, and the low-order portion of the result of a multiply instruction is identified by Q. The remainder of the pseudo registers in table I are used for housekeeping by the interpreter.

TABLE I

INTERPRETER PSEUDO REGISTERS

160-A Address	Contents	Name	Function
0000	7157		JFI Start; with address in A
0001	6720	ENTRY	Entrance to INTERPRETER
0002		PAK	160 Program Address Counter
0003		Y	160 Address of Operand
0004		B	Index Designator
0005		U	Upper Lower CTR: 2=lower, 0=upper
0006		YY	INTERFOR Address of Operand
0007	5733	EXIT	Goes to NORMALIZE
0012		AEXP	A Exponent
0013		ASIGN	A Sign
0014		AH	A Fraction, high 11 bits
0015		AM	A Fraction, middle 11 bits
0016		AL	A Fraction, low 11 bits
0017		EXPDIG	Input Exponent Digit Counter
0022		XEXP	X Exponent
0023		XSIGN	X Sign
0024		XH	X Fraction, high 11 bits

160-A Address	Contents	Name	Function
0025		XM	X Fraction, middle 11 bits
0026		XL	X Fraction, low 11 bits
0036	4000	MI	Mask
0037	3777	M2	Mask
0044		QH	Q Fraction, high 11 bits
0045		QM	Q Fraction, middle 11 bits
0046		QL	Q Fraction, low 11 bits
0051		B1	Index Register 1
0052		B2	Index Register 2
0053		B3	Index Register 3
0054		B4	Index Register 4
0055		B5	Index Register 5
0056		B6	Index Register 6
0057	6713	START	INTERPRETER Entrance, ADR in A
0060		P-REG	INTERFOR Program Address
0070		TEM 0	
0071		TEM 1	
0072		TEM 2	
0073		TEM 3	
0074		TEM 4	Temporary Storage
0075		TEM 5	
0076		TEM 6	
0077		TEM 7	

NOTE: All low core locations (0000---0077) are used by INTERFOR except: 0010, 0011, 0020, 0021, 0030, 0031, 0040, 0041. These locations are left open to provide interrupt capabilities for the INTERFOR system.

PART II

INTERFOR WORD STRUCTURE

The INTERFOR interpreter routine requires that the instructions and data in a program to be executed by the interpreter be in a specific format. Each instruction word and data word is converted by the interpreter to a particular internal format. Since each converted word is 48 bits, a special INTERFOR address identifies the storage location of each word. An instruction word consists of two 24-bit INTERFOR instructions. A data word consists of one floating point decimal number in binary form.

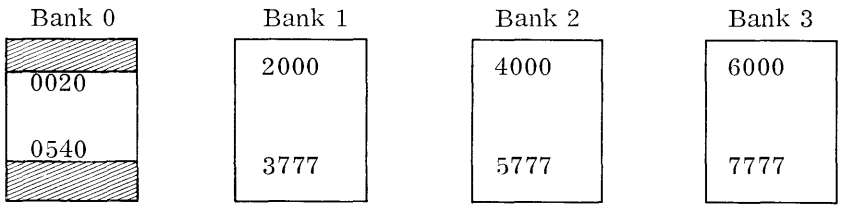
INTERFOR instructions written in octal notation may be loaded from previously prepared Flexowriter paper tape or manually inserted on the console typewriter under control of I/O subroutines within the interpreter routine. Data may be entered from previously prepared paper tape, from the console typewriter, and from the 167 card reader.

Segments of object programs may contain 160-A machine language coding. The interpreter does not execute instructions coded in machine language. Machine language coding under the INTERFOR system is explained in part V together with symbolic coding.

Instruction and data words, input formats, and the INTERFOR addressing scheme are described below.

MEMORY ALLOCATION

The following description of the interpreter memory allocation applies to a 4-bank 160-A computer. Storage for a 2-bank 160-A consists of banks 0 and 1 only.



Each bank contains 1024_{10} 48-bit words. Each octal (INTERFOR) address refers to a 48-bit word. The interpreter routine occupies INTERFOR locations 0000_8 though 0020_8 and 0540_8 through 1777_8 of bank 0 (shaded area). Remaining storage space in bank 0 may be used by the programmer. However, external system subroutines * must be located in bank 0; therefore it is advisable that object programs be located in other banks. When a program is executed under control of the interpreter, bank switching is provided by the interpreter routine; for those portions of programs that are coded in 160-A machine language, bank switching must be programmed.

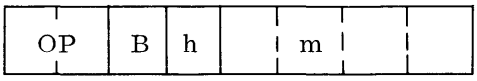
NOTE: For machine language programming under the INTERFOR system, SPECIFIC mode instructions may be used as the interpreter does not normally use machine location 7777_8 in bank 0. However, an MEL or an MES interpreter instruction will destroy the previous contents of this location. Refer to 160-A programming manual for SPECIFIC mode explanation.

INSTRUCTION WORD FORMAT

UPPER	LOWER

An INTERFOR instruction word contains two 24-bit INTERFOR instructions--the upper instruction and the lower instruction. Each instruction is made up of four functional parts:

*Appendix A



where

OP 2 octal digits	OP code specifies the instruction that is to be executed.
B 1 octal digit	Index designator specifies which (if any) of 6 index registers is to be used in modifying the execution address of the instruction.
h 1 octal digit	Breakpoint designator causes a conditional halt in processing (before the execution of the instruction) if certain selective stop console switches are set. (See Selective Stop Switch Table).
m 4 octal digits	Depending on type of instruction specified by OP code, m may be used as the base address of an operand (modified by the contents of the specified index register) or as the operand itself.

The INTERFOR instruction repertoire is contained in part III.

Address Modification

The interpreter recognizes two modes of addressing: direct, b = 0, and relative, b = 1 through 6. In the direct mode, the m term is interpreted as the execution address. In the relative mode, the execution address is formed by adding b (contents of index register) to m. (b = 7, illegal value.)

EXAMPLE: Given the symbolic instruction

OP	b	m
LDA	5	0106

where the contents of index register 5 is 7000, the execution address M is $m + (B^b)$.

$M = 0106 + 7000$

$= 7106$, the final execution address.

Breakpoint Designator

The conditions of the selective stop switches for the various values of h are shown below (x indicates a stop switch in the UP position):

SELECTIVE STOP SWITCH TABLE

Switch / h	4	2	1
0			
1			x
2		x	
3		x	x
4	x		
5	x		x
6	x	x	
7	x	x	x

EXAMPLE: A breakpoint stop will occur if an instruction contains h = 5 and either selective stop switch 4 or 1 (or both) are up.

Breakpoints in Symbolic Coding

The breakpoint designator, h, in symbolic coding is a single octal digit in the third character position of the comments field preceded by bp in the first two character positions of the comments field. A more complete explanation of symbolic coding is contained in part V.

Instruction Load Formats

INTERFOR object programs are punched on paper tape and loaded via the paper tape reader, or loaded manually via the console typewriter. For either medium, each

instruction must be 8 octal characters; the format must be one of the two shown below. Octal information consisting of 8 characters per word may be loaded in the same manner as instructions. Object programs are relocatable by use of a relocation constant (INTERFOR address) at load time. (Instruction Load Operating Instructions, part VII.)

The following symbols are used in defining the two instruction load formats:

Symbol	Meaning
A	Four octal digits specifying the simulator storage address.
I	The 8 octal digits specifying the information to be loaded at the specified address. NOTE: Any number of spaces and/or tabs may be contained within I, but I must contain 8 octal digits.
( )	The parentheses enclose a quantity which is optional within the format.
-	The minus code indicates that the A field is not to be modified by the relocation constant that may be manually inserted in the 160-A accumulator prior to initiating the load routine. If the minus code is not used, the load address will be modified by the relocation constant at load time.
/	The slash code immediately follows I and causes the last 4 octal digits of I to be increased by the relocation constant before being stored.
.	The period code, following the first group of 4 octal digits of I, causes these 4 digits to be increased by 4 times the relocation constant. Similarily, a period code following the second group of 4 octal digits causes these 4 digits to be increased by 4 times the relocation constant.
CR	The carriage return terminates a line of information. The loading routine anticipates a new storage address as the first item of the subsequent line. If a new address is specified, the information, I, of that line will be stored in the

Symbol	Meaning
	upper half of the storage address. If a new address is not specified (skipped over by a tab) the information will be stored in the half-word following that used for the previous storage.
TAB	Each tab code in a line indicates the beginning of an I field.
;	The semicolon, following a CR, terminates the instruction load routine.

FORMAT A

One instruction per line (one tab)

(CR)

(-) A Tab I (/.) CR

(-) (A) Tab I (/.) CR
⋮ (Instructions)
⋮
(-) (A) Tab I (/.) CR

;

The I field of the first line is loaded as an upper instruction. Subsequent I fields will be placed in sequence, lower, upper, lower, upper, lower, etc., unless a subsequent I field is preceded by an A field, resulting in a new sequence of upper and lower I fields. If the lower half of an instruction word is not specified, the previous contents are preserved.

FORMAT B

Two instructions per line (two tabs)

(CR)

(-) A Tab I (/.) Tab I (/.) CR

(-) (A) Tab I (/.) Tab I (/.) CR
⋮ (Instructions)
⋮
(-) (A) Tab I (/.) Tab I (/.) CR

;

In Format B, each line represents a complete 48-bit instruction word. The first I field in the line represents the upper instruction.

NOTE: Formats A and B may be mixed.

EXAMPLE: Relocation constant of 0500 specified at load time. Format A is used.

RESULTING STORAGE

Input Listing			Location	Contents
A	I		(INTERFOR Address)	
-0100	12345677	CR	0100	12345677 (lower = previous contents)
0000	11112222	CR	0500	1111222233334444
TAB	33334444	CR		
TAB	5555 6666	CR	0501	55556666 (lower = previous contents)
0010	7777 0000/	CR	0510	7777050024012402
TAB	0001.0002.	CR		

160-A machine coding may be loaded in the same manner by packing two octal machine instructions in each octal INTERFOR instruction, and supplying the proper INTERFOR address in the A field. Each INTERFOR address contains four 160-A machine instructions.

DATA FORMAT

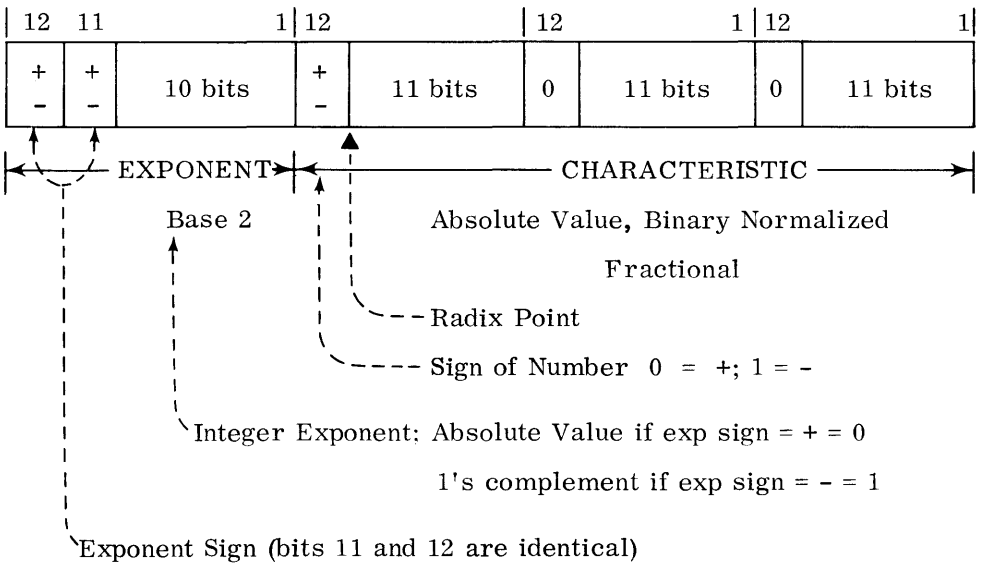
Data words in the INTERFOR system must be floating point decimal numbers. The format for entering floating point data is:

$$\pm \text{ .DDDDDDDDD } \begin{matrix}\text{TAB} \\ \text{or} \\ \text{CR}\end{matrix} \quad \pm \text{ EEE } \begin{matrix}\text{TAB} \\ \text{or} \\ \text{CR}\end{matrix}$$

where D represents a decimal digit of the fractional field of the data word and E represents a decimal digit of the exponent. Both the fractional part and the exponent may be preceded by a + or - sign.

If the sign is omitted, the field is interpreted to be positive. The fractional field which may contain up to 9 decimal digits is terminated by a TAB if followed by an exponent, otherwise the field is terminated by a CR. The exponent field which may contain up to 3 decimal digits may be terminated by either a TAB or a CR.

The interpreter converts each floating point decimal word into a 33-bit normalized binary fraction ($\frac{1}{2} \leq$ fraction < 1) and a 10-bit exponent; each 12 bits occupies a consecutive 160-A storage location. In the two low-order machine locations bit 12 is always zero. The exponent sign occupies two bit positions.



The following examples demonstrate how floating point decimal numbers will appear in octal format after being converted to binary numbers.

10_{10} = 0.10100------00*2^4 (binary fraction)

= 0004 2400 0000 0000 (octal notation)

-5_{10} = 1.100------00*2^0 (binary fraction)

= 0000 6000 0000 0000 (octal notation)

$+.25_{10}$ = 0.100------00*2^{-1} (binary fraction)

= 7776 2000 0000 0000 (octal notation)

NOTES:
1. Input data may be prepared on cards or Flexowriter tape, or entered from the console typewriter.
2. Delete codes may appear on paper tape as they are not recognized by the interpreter load subroutines.
3. The approximate magnitude of range for decimal numbers is between the values of 10_{10}^{-308} and 10_{10}^{+307}

Floating point data words are loaded and converted by one of two types of internal interpreter subroutines as described in part IV. One type is for loading data under program control, and the other is for manually inserting data after interrupting the program. The manual load subroutines accept data from either the paper tape reader or from the console typewriter. The programmable data input subroutines accept data from either the 167 card reader, the paper tape reader, or the typewriter. Each subroutine requires that data be in a specific format as shown below. The format is the same for paper tape as typewriter input.

Paper Tape Format for Program Control

The format for loading data under program control is:

$$\pm \text{ .DDDDDDDDD } \begin{matrix}\text{TAB}\\ \text{or}\\ \text{CR}\end{matrix} \pm \text{ EEE } \begin{matrix}\text{TAB}\\ \text{or}\\ \text{CR}\end{matrix}$$

No additional control characters are required. The number of data words that may be punched under this format is limited only by storage capacity. The first word

on tape may be preceded by a CR; it will not be recognized by the subroutine. Trailing zeros in the fractional field need not be punched.

EXAMPLE: Five decimal numbers, -0.1, -0.05, 1.25, 5.75, 11.6, are to be punched in floating point format and loaded under program control. The data tape is punched in the following manner:

CR (optional)
-.1 CR
-.5 TAB -1 CR
.125 TAB 1 CR
.575 TAB 1 CR
.116 TAB 2 CR

NOTES:
1. Use same format for entering the data from the console typewriter. If a typing error is made, an X character inserted in the faulty word will cause that word to be ignored.
2. The exponent fields may be terminated by a TAB.
3. Delete codes may appear on the data tape.

Data Card Format for Program Control

The card input format for floating point data is as follows:

± .DDDDDDDDD ±EEE*

Column	
1	Sign of fractional field: -, + or blank
2	Decimal point (must be punched)
3-11	Fractional field (up to nine decimal digits)
12,13	Must be blank
14	Sign of exponent field: -, + or

Column	
15-17	Exponent Field (up to three decimal digits)
18	Asterisk (must be punched to terminate read operation)
19-80	Ignored

NOTES: 1. Blanks in fractional and exponent fields are interpreted as spaces, not as zeros.

2. Fractional field should be normalized.

3. Only one word may be punched on a card.

Paper Tape Format for Manual Load

The manual load subroutine format differs from the program control format in that an INTERFOR address field must precede the fractional field of each word. Regardless of the sequence of loading, a load address must accompany each data word. The rules for the fractional and exponent fields are the same for both load formats. Operating instructions for the manual load subroutine are in part VII. Format for manual load is shown in the following example.

EXAMPLE: Load the decimal numbers -0.1, -0.05, 1.25, 5.75, and 11.6 using the manual data load subroutine. The numbers are to be stored in sequential INTERFOR locations starting at 3000. The data tape is punched as follows:

CR

3000 TAB -.1 CR

3001 TAB -.5 TAB + 1 CR

3002 TAB .125 TAB 1 CR

3003 TAB .575 TAB 1 CR

3004 TAB .116 TAB 2 CR

NOTES:

1. A relocation constant used with the manual data load subroutine will modify the A field of all words that do not have a - sign preceding the A field. In the example, if a relocation constant of 2000 is used at load time, the resulting INTERFOR addresses will be 5000, 5001, 5002, 5003, and 5004. If the A fields are preceded by - signs, the relocation constant is ignored and the words will be stored consecutively starting at 3000.
2. The format for manual data load from the console typewriter is the same as above.
3. Exponent fields must be terminated by a CR.
4. The ; symbol must be placed after carriage return of the last word in order to terminate the manual data load subroutine.
5. Delete codes may be used to correct typing errors.

PART III

INTERFOR INSTRUCTIONS

In the 22 INTERFOR instructions described on the following pages, the title line of each instruction contains the mnemonic code and format, name, and absolute coding. Abbreviations and symbols are defined as follows:

A	48-bit pseudo accumulator
b	Index designator
B^b	Designated index register
Exit (full)	Proceed to upper instruction of next program step
Half Exit	Proceed to lower instruction of same program step
j	The condition designator for jump and stop instructions
LA	Lower address
UA	Upper address
m	Unmodified operand address
M	Modified operand address $M = m + (B^b)$
( )	Contents of (register or storage location)
y	Unmodified operand
Y	Modified operand. $Y = y + (B^b)$
h	Breakpoint designator

A description of the error codes included in the specification of certain instructions is contained in the operator's guide in part IV.

INSTRUCTION REPERTOIRE

Mnemonic Code	Name	Absolute Code
LDA	Load A	12 bh mmmm
LAC	Load A Complement	13 bh mmmm
STA	Store A	20 bh mmmm
MEL	Matrix Element Load	34 bh mmmm
MES	Matrix Element Store	35 bh mmmm
LIU	Load Index Upper	52 bh mmmm
LIL	Load Index Lower	53 bh mmmm
SIU	Store Index Upper	56 bh mmmm
SIL	Store Index Lower	57 bh mmmm
ISK *	Index Skip	54 bh yyyy
FAD	Floating Add	30 bh mmmm
FSB	Floating Subtract	31 bh mmmm
FMU	Floating Multiply	32 bh mmmm
FDV	Floating Divide	33 bh mmmm
ENI	Enter Index	50 bh yyyy
INI	Increase Index	51 bh yyyy
	Exit to 160	51 0h yyyy
AJP	A Jump	22 jh mmmm
SLJ	Selective Jump	75 jh mmmm
SLS **	Selective Stop	76 jh mmmm
SSK *	Storage Skip	36 bh mmmm
EQS *	Equality Search	64 bh mmmm
THS *	Threshold Search	65 bh mmmm

* Upper Instruction Only

** j = 0 or 4 only

DATA TRANSMISSION

LDA b m — Load A — 12 bh mmmm

Replaces (A) with a 48-bit operand contained in the location specified by M. Initial (A) are changed during execution; (m) remain unchanged. Negative zero is formed in A if the operand at M is equal to negative zero.

LAC b m — Load A Complement — 13 bh mmmm

Replaces (A) with the complement* of the operand contained in the location specified by M. Initial (A) are changed during execution; (M) remain unchanged. Negative zero is formed in A if the operand at M is equal to positive zero.

STA b m — Store A — 20 bh mmmm

Replaces contents of designated location, M, with contents of A. Initial (A) remain unchanged; initial (M) are changed.

MEL b m bph — Matrix Element Load — 34 bh mmmm

Replaces (A) with a 48-bit operand from the b row and h column of a columnwise stored matrix. The a_{11} element of the matrix must be stored at address m + 1. The column length of the matrix must be stored in octal form in the upper half of word at address m. (B^b) = current value of column index. The breakpoint designator bph cannot be 0.

---

* Changes sign of the number only.

SIL b m — Store Index Lower — 57 bh mmmm

Replaces lower address portion of m with contents of designated index register. Remaining bits of the word in storage are unchanged. Initial (B^b) are unchanged; initial (m) are changed. If b = 0, this instruction becomes a pass instruction.

ADDRESS MODIFICATION

ISK b y — Index Skip — 54 bh yyyy

This instruction compares the quantity in the designated index register with the operand, y. If (B^b) are less than y, (B^b) are increased by one, and a half exit is performed. When the two quantities y and (B^b) become equal, the designated index register is cleared to zero and a full exit is performed. A half exit proceeds to the lower instruction of the program step, and a full exit proceeds to the upper instruction of the next program step. If the quantity in the index register is greater than the operand y, an error stop occurs. That is, the interpreter assumes that y is the maximum value the index register will be allowed to attain.

FLOATING POINT ARITHMETIC

An exponent fault error, e, will occur if the absolute value of the binary exponent exceeds 1777_8.

FAD b m — Floating Add — 30 bh mmmm

Forms the sum of two operands packed in floating point format. A floating point operand is read from storage location M and added to the floating point word in A. The result is normalized, rounded, and retained in A at the end of the operation. Exponent equalization takes place prior to performing the arithmetic operation.

NOTE: If the result is zero, the sign of the original contents of the accumulator is assigned to the answer.

FSB b m Floating Subtract 31 bh mmmm

Forms the difference of two 48-bit operands in floating point format. The subtrahend is acquired from storage address M and is subtracted from the minuend in A. The result is rounded and normalized if necessary and retained in A. Exponent equalization occurs prior to performing the arithmetic operation. See note under FAD.

FMU b m Floating Multiply 32 bh mmmm

Forms the product of two 48-bit operands in floating point format. One operand must be loaded into A prior to executing the instruction. The other operand is read from storage location M.

The product is rounded and normalized if necessary and retained in A.

FDV b m Floating Divide 33 bh mmmm

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

NO ADDRESS

In these instructions, the operand y is treated as a one's complement number. Numbers 4000 to 7777 are considered to be negative numbers; 4000 = -3777, 7777 = -0.

ENI b y	Enter Index	50 bh yyyy

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

INI b y	Increase Index	51 bh yyyy
	Exit to 160-A Coding	51 0h yyyy

Increases (B^b) by the operand y. If b = 0, this instruction causes an exit from the interpreter to basic 160-A coding. In this case yyyy specifies the interpreter return address when a normal exit is made from machine language back to the interpreter. The 160-A word immediately following this instruction is the first 160-A instruction to be executed.

NOTE: See part IV, Machine Language Subroutines.

JUMPS AND STOPS

Address modification does not apply to these instructions.

Normal Jump

A jump instruction terminates a current program sequence and initiates a new sequence at a different location in storage. In all jump instructions, the execution address, m, specifies the beginning address of the new program sequence. The word at address m is read from storage, and the upper instruction (first instruction of the new sequence) is executed.

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

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

AJP j m — A Jump — 22 jh mmmm

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

j = 0 jump if (A) = 0

j = 1 jump if (A) $\neq$ 0

j = 2 jump if (A) = +

j = 3 jump if (A) = -

NOTE: The zero tests are made on the bits of the fractional part of the number in the accumulator. The sign bit is not included.

SLJ j m — Selective Jump — 75 jh mmmm

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

j = 0 Jump unconditionally. (Key setting does not reference jump.)

j = 1 Jump if selective jump key 1 is set.

j = 2 Jump if selective jump key 2 is set.

j = 3 Jump if selective jump key 4 is set.

Normal Stop

SLS 0 m — Selective Stop — 76 0h mmmm

Stops at present step in the sequence and executes an unconditional jump to address m when the run-step key is moved to RUN or STEP. If $j \neq 0$ or 4, an error stop d occurs during the halt of a SLS instruction with the following display:

P = 6351 A = INTERFOR Address Z = 7700

Return Jump

A return jump begins a new sequence at the lower instruction of the program step to which the jump is made. At the same time, the execution address of the upper instruction of that program step is replaced with the address of the next step in the main program. This instruction is usually an unconditional jump instruction and allows a return to the main program after completing the sub-program sequence.

AJP j m — A Jump — 22 jh mmmm

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

j = 4 Return jump if (A) = 0

j = 5 Return jump if (A) $\neq$ 0

j = 6 Return jump if (A) = +

j = 7 Return jump if (A) = -

The same conditions apply to the zero tests as outlined with AJP Normal.

SLJ j m | Selective Jump | 75 jh mmmm

Executes a return jump to m on condition j (selective jump key setting). If the condition is not satisfied, the next instruction is executed.

j = 4 Return jump unconditionally. (Does not reference jump keys.)

j = 5 Return jump if jump key 1 is set.

j = 6 Return jump if jump key 2 is set.

NOTE: The set position of a jump key is UP.

Return Stop

SLS 4 m | Selective Stop | 74 4h mmmm

Stops unconditionally and executes a return jump to m if the run key is moved to the RUN or STEP. If $j \neq 0$ or 4, an error stop d occurs during the halt of an SLS instruction with the following display:

P = 6351 A = INTERFOR Address m Z = 7700

STORAGE TEST

SSK b m | Storage Skip | 36 bh mmmm

Senses the sign bit of the operand in M. If the sign is negative, a full exit is taken. If the sign is positive, a half exit is taken. The contents of the operational registers are left unmodified. SSK is restricted to an upper instruction. If used as a lower instruction an error stop, S, occurs.

STORAGE SEARCH

If b = 0 or if $(B^b) = 0$ in the following instructions, only the word at storage location m will be searched. The search area must be entirely within one bank.

EQS b m — Equality Search — 64 bh mmmm

Searches a list of operands to find one that is equal to A. The number of items to be searched is specified by B^b. These items are in sequential addresses beginning at the location specified by m. The search begins with the last address, m + (B^b) -1. For each word that is searched (B^b) is reduced by one until an operand is found that equals A or until (B^b) equals zero. If the search is terminated by finding an operand that equals A, a full exit is made. The address of the operand satisfying this condition is given by the sum of m and the final (B^b). If no operand is found that equals A, a half exit is taken. Positive zero and minus zero are recognized as the same quantity. If EQS is used as a lower instruction, an error stop "S" occurs.

THS b m — Threshold Search — 65 bh mmmm

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

PART IV

INTERFOR SUBROUTINES

Subroutines for the INTERFOR system fall in three categories:

Internal I/O subroutines
External subroutines
Machine language subroutines

Internal I/O subroutines are an integral part of the interpreter routine and perform input/output functions. External subroutines consist of arithmetic and output subroutines that are a part of the INTERFOR system but are not included with the interpreter routine. The external subroutines are described in Appendix A. Machine language subroutines are those in an object program that are coded in machine (160-A) language. An explanation of the methods used for entrance and exit for each type of subroutine, as well as a description of the internal I/O subroutines, is given below. Internal and external subroutines are entered via return jump instructions in the object program. Machine language coding under INTERFOR also requires the use of return jumps to leave and re-enter the interpreter routine. Input formats are described in part II.

INTERNAL I/O SUBROUTINES

The internal I/O subroutines consist of programmable data input and output subroutines, manual data input and output subroutines, and the instruction load subroutines. Input/Output media for internal subroutines are: paper tape reader, paper tape punch, console typewriter, 166 printer and 167 card reader. Programmable I/O subroutines are called by the programmer in an object program. Manual I/O subroutines and the instruction load subroutines are initiated by the computer operator at the 160-A console. Operating instructions for the latter subroutines are in part VII.

Programmable Data I/O Subroutines

These routines provide for programmable input and output of floating-point decimal data between the accumulator and the I/O equipment. These subroutines, which are integral to the INTERFOR interpreter, are stored in fixed locations.

Each data input subroutine converts one floating point decimal data word into binary format and leaves the converted binary word in the accumulator. Therefore the programmer must enter a data input subroutine each time a new data word is to be loaded, and also store the converted binary word before loading the next word.

Each time a data output subroutine is entered, the contents of the accumulator are converted and dumped as a single floating point decimal number on the specified output medium. The word to be dumped must be in the accumulator prior to entering the output subroutine.

To enter the programmable data input subroutines, the programmer must use a return jump instruction in which the M term contains the INTERFOR entrance address of the required subroutine. When the word has been loaded or dumped, the subroutine will exit to the upper half of the next instruction word in the main program.

Data Input Subroutine Entrances

Entrance locations to data input subroutines depend upon the source of input data.

Source	Entrance Address
167 Card reader	0754
Paper tape reader	1224
Typewriter	1226

Format for floating point input data is in part II.

Data Output Subroutine Entrances

Entrance locations to data output subroutines depend on the output device and whether the word is terminated with TAB or CR.

Output	Entrance Address
Paper tape punch	
followed by TAB	1230
followed by CR	1232
Typewriter	
followed by TAB	1234
followed by CR	1236
166 Printer	
followed by TAB	0612
followed by CR	1126
output TAB only	0654
output CR only	1222

EXAMPLE: The following example in symbolic notation demonstrates the programmable data input and output subroutines. A block of ten words is to be read from the paper tape reader and stored in consecutive locations starting at BLOCK. The first five words of this record are then dumped on the console typewriter. Index register #1 is used as a counter.

```
START   (U)   ENI   1   0
              ENI   0           Pass
LOAD    (U)   SLJ   4   1224    Enter input subroutine
              ENI   0           Pass
        (U)   STA   1   BLOCK
```

```
        ENI   0                    Pass
(U)     ISK   1     9D
        SLJ   0     LOAD
(U)     SLS   0     OUT            Temporary Halt
        ENI   0
OUT (U) ENI   1     0
        ENI   0                    Pass
DUMP (U) LDA  1     BLOCK
        SLJ   4     1236
(U)     ISK   1     4D
        SLJ   0     DUMP
(U)     SLS   0     START
```

U = upper instruction

Instruction Load Subroutines

The instruction load subroutines allow INTERFOR instructions in octal format to be loaded from paper tape or the typewriter. To use the instruction load subroutine, the interpreter routine must be halted. Operating procedures for loading instructions in octal format are contained in part VII. The octal format for instructions is specified in part II. Symbolic instructions cannot be loaded by these subroutines. Refer to part V for symbolic format.

Manual Data Load Subroutines

The manual data load subroutines provide for relocation of floating-point data at load time by requiring that a storage address be entered with each floating point data word. Each load address may be incremented by a relocation constant if desired. The manual data load subroutines differ from the programmable data load subroutines in that (1) the interpreter routine must be at a normal stop, and (2) the subroutine

(paper tape or typewriter) is entered by forcing the P register to the required starting address (operating instructions part VI). Format for manual input data is in part II as well as with the operating instructions.

MACHINE LANGUAGE SUBROUTINES

An object program to be executed by the INTERFOR interpreter routine may contain 160-A machine instructions (in octal form) as well as INTERFOR instructions. Since 160-A instructions are not recognized by the interpreter routine, provision is made for exit and re-entry.

Interpreter Exit

To exit from the interpreter routine to machine language coding, the INTERFOR instruction INI 0 y (51 Oh yyyy) must be in the lower half of the INTERFOR instruction word immediately preceding the first word of 160-A coding. The y field of the INI instruction contains the INTERFOR location of the next instruction to be executed by the interpreter routine. When the INI 0 y instruction is executed, control is transferred to the 160-A machine language subroutine or program segment immediately following the INI instruction. The subroutine is then executed independently of the interpreter routine.

Returning to Interpreter Control

To re-enter the interpreter routine, the last two 160-A machine instructions must be:

Mnemonic		Octal
LDC	00	2200
	6720	6720
ACJ	70	0070

The above instructions clear all bank settings to zero and enter the interpreter routine at INTERFOR address 1564 (machine address 6720).

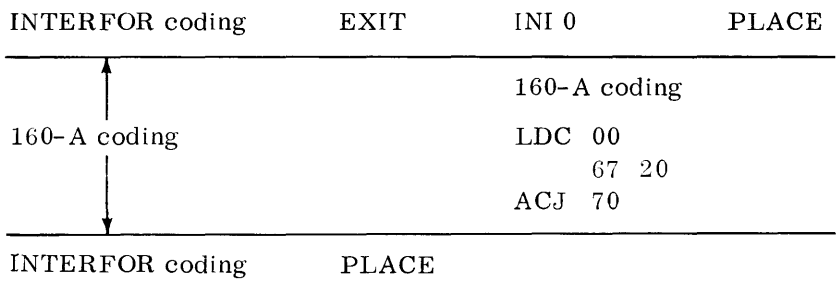
NOTES: 1. Both INTERFOR and machine language instructions must be in octal notation unless assembled by FLAP, the INTERFOR assembly program. All 160-A instructions in a program to be assembled by FLAP must be in octal form.

2. The relative bank setting is equal to the direct bank setting when an exit from the interpreter is performed. The two bank settings depend on the location of the last INTERFOR instruction executed. The indirect bank setting may or may not be equal to the relative and direct banks when an exit from the interpreter is performed.

EXAMPLES:

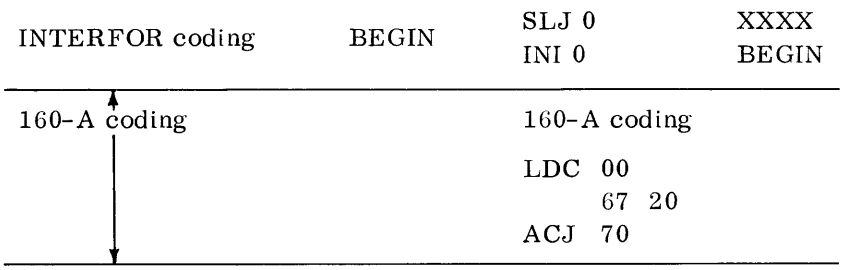
Open Subroutine

In this example the ACJ instruction returns control to location PLACE in the INTERFOR coding.



Closed Subroutine

In this example the ACJ instruction returns control to the SLJ 0 instruction, which in turn causes a jump to another segment of the INTERFOR coding.



PART V

INTERFOR ASSEMBLY PROGRAM
(FLAP)

FLAP, the INTERFOR assembly program, is included as a part of the INTERFOR system to allow coding of INTERFOR programs in symbolic language. FLAP processes symbolic coding to produce a binary object program that is executed under control of the interpreter routine. A listable output is produced that lists the symbolic input with the side-by-side translation of the symbolic coding. Input to FLAP may be on either cards or paper tape. Listable output may be on paper tape or on the 166 printer; binary output is on paper tape. The binary object program is loaded by FLOADER, the binary loader routine. Operating instructions for both FLAP and FLOADER are contained in part VI.

FLAP is normally a two-pass assembler, unless available storage is exceeded during assembly in which case a third pass is required to complete the assembly. During the first pass, FLAP reads the symbolic input and compiles the symbol table, translates all mnemonic operation codes into machine language, and condenses each line of coding for processing during the second pass. If storage capacity is exceeded during the first pass, the input for the second pass is automatically punched out on paper tape. This paper tape must be positioned for reading before starting the second pass. The second pass generates the assembled listing and the binary object program. A third pass is required only if the storage space required for the binary output exceeds available core storage. The third pass requires the second pass input, and will generate the binary object program tape. Generation of the assembled listing will be completed during the second pass. The capacity of the symbol table is approximately 500 symbols; if capacity is exceeded as indicated by an error stop, the number of symbols in the symbolic program must be reduced.

TABLE II

SUMMARY OF INSTRUCTION CODES ACCEPTED BY FLAP

Mnemonic Code	Octal Equivalent	Instruction Description
LDA	12	Load A
LAC	13	Load A, Complement
STA	20	Store A
AJP	22	A Jump
FAD	30	Floating Add
FSB	31	Floating Subtract
FMU	32	Floating Multiply
FDV	33	Floating Divide
SSK	36	Storage Skip, Upper Only
ENI	50	Enter Index
INI	51	Increase Index
LIU	52	Load Index Upper
LIL	53	Load Index, Lower
ISK	54	Index Skip, Upper Only
SIU	56	Store Index, Upper
SIL	57	Store Index, Lower
EQS	64	Equality Search, Upper Only
THS	65	Threshold Search
SLJ	75	Unconditional Jump
SLS	76	Unconditional Stop
MEL	34	Matrix Element Load
MES	35	Matrix Element Store

Mnemonic Code	Octal Equivalent	Instruction Description
REM		Remarks
EQU		Equivalence
ORG	(Pseudo instructions)	Origin
DEC		Decimal Constant
BSS		Block Reserve
OCT		Octal Constant
END		End of Assembly

SYMBOLIC INSTRUCTION FORMAT

Each symbolic INTERFOR instruction to be assembled by FLAP is represented by one line of coding in the following format:

Location Field	Operation Field	B- Field	M- Field	Comments Field

Each line is assembled as one-half of an INTERFOR instruction word; every line with a symbol in the location field will be assembled as the upper half of the instruction word. If consecutive lines have symbols in the location field, the assembly program will insert a pass instruction (ENI 0) in the lower half of the appropriate preceding instruction word. If an instruction is to be assembled in the upper half of an INTERFOR instruction word, but the programmer does not want to use a symbol in the location field, a comma or a + character may be entered in the location field instead. A comma (Flexowriter) or + (card input) appearing by itself in a location field will cause that line to be assembled as an upper instruction.

Location Field

The location field symbolic identifier may contain up to eight alphanumeric characters. All identifiers must contain at least one non-numeric character. Any legal Flexowriter character may be used in this field. If a symbol is punched in this field, the line will be assembled as an upper instruction.

Operation Field

This field must contain a 3-character mnemonic operation code representing one of the INTERFOR instructions or one of the FLAP pseudo-instructions in Table II. This field cannot be omitted.

B-Field

The B-Field must contain a single octal digit (0 through 7) that normally specifies one of the two modes of INTERFOR addressing - direct or relative. Digits 1 through 6 specify index registers and imply relative addressing. If the field is zero, no index register is specified, indicating direct mode of addressing. If the field contains the illegal B term of 7, the line is flagged with error "B" except when used with the AJP instruction. For the selective jump and stop instructions, the j term is placed in the B-field. (An alphabetic character may precede the digit in the B-field; it will be ignored by the assembly program.)

M-Field

The M-field normally specifies a memory address that may be modified by the contents of the index register designated by the B-field. The M-field may have any of the following forms:

1. A symbolic identifier that references a symbol in the location field. This symbol must be identical to the symbol referenced in the location field. The line will be flagged with u if the symbol in the M-field does not appear in a location field within the symbolic program.

2. A symbolic identifier followed by a + or - sign and up to 4 numeric characters. If D is placed after the numeric characters, they will be interpreted as a decimal number, otherwise they will be interpreted as an octal number. The entire M-field is evaluated by computing the algebraic sum of the numeric value of the identifier and the number.

3. Four numeric characters preceded by a + or - sign. If no sign appears, the number is assumed to be positive. D following the last digit indicates that the number is decimal, otherwise it is octal.

4. A / character. This character indicates that the M-field has the same value as the INTERFOR address of the line being assembled.

5. A symbolic identifier followed by a + or - sign and another identifier.

The M-field may be modified by a relocation constant when loading the binary object program, depending on the type of format used within the M-field. The examples in group I below will be modified by a relocation constant at load time. The examples in group II (all numeric) will not be modified by a relocation constant.

Group I

ABCDEFGH

ABCDEFGH ± 1234

ABCDEFGH ± 1234D

ABCDEFGH ± IJKLMNOP

/

/ ± 1234

/ ± 1234D

/ ± ABCDEFGH

Group II

± 1234

± 1234D

NOTE: The + sign in group II may be omitted. A + sign is replaced by the , character for Flexowriter input.

Comments Field

The comments field is optional except for instructions that contain breakpoints. The breakpoint designator h must appear in the third character position of the comments field preceded by bp in the first and second character positions of the comments field. The only restrictions on the characters in the comments is that they must be legal Flexowriter characters (see table III). If the field contains more than 30 characters (including space codes and control characters), the comments field will appear on two or more lines on the listable output of FLAP. The first 30 characters will be included with the line being assembled. The excess characters will appear on subsequent lines with an REM pseudo instruction appearing in the operation field.

TABLE III

FLEXOWRITER AND PUNCHED CARD CHARACTER CODES
ACCEPTED BY FLAP

Character	Flexowriter Code	Punched Card Code
A, a	30	12-1
B, b	23	12-2
C, c	16	12-3
D, d	22	12-4
E, e	20	12-5
F, f	26	12-6
G, g	13	12-7
H, h	05	12-8
I, i	14	12-9
J, j	32	11-1
K, k	36	11-2
L, l	11	11-3
M, m	07	11-4
N, n	06	11-5
O, o	03	11-6
P, p	15	11-7
Q, q	35	11-8
R, r	12	11-9
S, s	24	0-2
T, t	01	0-3
U, u	34	0-4
V, v	17	0-5
W, w	31	0-6
X, x	27	0-7
Y, y	25	0-8
Z, z	21	0-9

Character	Flexowriter Code	Punched Card Code
0	56	0
1	74	1
2	70	2
3	64	3
4	62	4
5	66	5
6	72	6
7	60	7
8	33	8
9	37	9
+	46*	12
-	52	11
/	44	0-1
.	42	12-3-8
;	50	
)	54	12-4-8
Functions		
Back space	61	
Upper Case	47	
Lower Case	57	
Space	04	blank
Tab	51	
Carriage return	45	

*Upper Case, Lower case prints as a comma.

FLAP PSEUDO INSTRUCTIONS

In addition to the INTERFOR instructions listed in table II, the assembly program recognizes the following pseudo instructions.

Symbol	Definition
REM	REMARKS causes the succeeding characters in the current line of coding to be treated as comments. Remarks may consist of any number of Flexowriter characters.
EQU	EQUIVALENCE assigns to the symbolic identifier in the location field, the value of the expression appearing in the M-field.
ORG	ORIGIN precedes a block of INTERFOR instructions to be loaded sequentially in INTERFOR locations starting at the address specified by the M-field of the ORG instruction. The first INTERFOR instruction in the block will appear in the upper half of the instruction word as specified by the M-field. The second instruction will be placed in the lower half of the first instruction word if no symbol appears in the location field. Consecutive instructions will be placed in the respective upper and lower halves of sequential addresses. If no ORG is specified, the first word of a program is loaded in INTERFOR location 20_8.
DEC	DECIMAL CONSTANT indicates that the number in the M-field is in floating binary form. This constant will be converted by FLOADER at load time. The decimal number must be in the format: ±.DDDDDDDDDe±EEE where (1) the fractional coefficient consists of 1 to 9 decimal digits (D's) (2) the decimal point must not be omitted (3) the exponent, if used, contains from 1 to 3 decimal digits and must be preceded by an e, and a - sign if negative (4) the M-field must be terminated with a TAB or CR.

A blank M-field is converted to zero; the + sign (, for Flexowriter) may be omitted. DEC numbers are converted by the binary load routine using one of the conversion subroutines in the interpreter.

BSS — BLOCK RESERVE reserves a block of memory. If a location symbol is used, it will be assigned to the first word of that block. The word length of the block to be reserved is placed in the M-field. The next line of coding will be placed in the location following the last word in the reserved block.

OCT — OCTAL CONSTANT provides a means of entering signed octal constants. The sign is optional; up to 16 octal digits may be placed in the M-field. If fewer than 16 characters are entered, the octal number is right-adjusted (leading zeros are appended to less than 16 digits). The OCT pseudo instruction is used to load 160A machine language coding.

END — END OF ASSEMBLY appears at the end of a symbolic program to indicate the end of the program tape or deck. If an address appears in the M-field, the END pseudo instruction will also generate a transfer card to be interpreted by the loader as the start execution location.

MACHINE LANGUAGE CODING

The programmer may wish to include 160-A machine language coding within a symbolic program. This is accomplished by writing the desired 160-A instructions in octal notation, and placing them in the M-field of OCT pseudo instructions, four instructions in each M-field. Since the symbolic portion of a program is normally relocatable, the 160-A coding should consist entirely of relative (forward or backward), no address, and constant type 160-A instructions in order to maintain relocatability. Instructions that reference low core (00-77) must not be used. The first machine language instruction must be preceded by an INI 0 instruction in the INTERFOR coding. The programmer must also provide for a return to the interpreter routine. Methods of exits and entrances are described under machine language subroutines in Part IV.

INSTRUCTION WORD PAIRING

Certain INTERFOR instructions must be placed either in the upper or lower half of an instruction word because of the inherent characteristic of each instruction. Therefore the programmer must explicitly or implicitly indicate the part of the instruction word that each symbolic INTERFOR instruction must occupy. Instructions may be explicitly loaded only in the upper half of an instruction word. Instructions may be implicitly loaded in either half of an instruction word.

To explicitly load a symbolic instruction in the upper half of a word, a symbol or a , character (+ character) must be placed in the location field. Conversely, every instruction that has a symbol in the location field will be placed in the upper half of an instruction word. The exception to this rule is the first INTERFOR instruction following an ORG pseudo instruction in a symbolic program. This first instruction will be placed in the upper half of the word at the location specified by the M-field of the ORG pseudo instruction, even without a symbol in the location field. Consecutive instructions will be placed alternately in the lower and upper halves of the first and succeeding instructions words until a location field symbol is detected. A new sequence of upper and lower instruction assignments commences with each instruction having a symbol in the location field. If two or more consecutive instructions have symbols in the location field, each will be an upper instruction and the assembly program will generate pass instructions (ENI 0) in the lower half of the appropriate words. This same situation occurs when an instruction identified with a symbol follows an instruction that normally falls in the upper part of an instruction word.

To implicitly load an instruction in either the upper or lower half of a word, the programmer must note the location of the instruction last specified as an upper instruction, and keep an account of the number of succeeding instructions he has included in his program. If the number of succeeding instructions is an odd number, the next instruction will be an upper instruction; otherwise it will be a lower instruction. Pass orders may be used to cause a sequence to be odd or even.

The following example demonstrates how a sequence of instructions in a symbolic program appears on the listable output. The coding represents a routine that uses programmable input/output subroutines.

EXAMPLE:

	Symbolic Input		
Loc	Op	B	M
	ORG		2000
START	ENI	1	0
LOAD	SLJ	4	1224
,	STA	1	BLOCK
,	ISK	1	9D
	SLJ	0	LOAD
	SLS	0	OUT
BLOCK	EQU		3000
OUT	ENI	1	0
DUMP	LDA	1	BLOCK
	SLJ	4	1236
	ISK	1	4D
	SLJ	0	DUMP
	SLS	0	START
	END		START

Address Assignment

INTERFOR Address	Loc	Op	B	M
2000	START	ENI	1	0
		ENI	0	
2001	LOAD	SLJ	4	1224
		ENI	0	
2002	,	STA	1	BLOCK
		ENI	0	
2003	,	ISK	1	9D
		SLJ	0	LOAD
2004		SLS	0	OUT
	BLOCK	EQU		3000
		ENI	0	
2005	OUT	ENI	1	0
		ENI	0	
2006	DUMP	LDA	1	BLOCK
		SLJ	4	1236
2007		ISK	1	4D
		SLJ	0	DUMP
2010		SLS	0	START
		ENI	0	
		END		2000

FLAP INPUT

Symbolic programs to be assembled by FLAP may be punched on paper tape or cards. Paper tape input is via the 350 paper tape reader; card input is via the 167 card reader.

Paper Tape Format

Each word on paper tape must be in the form shown under INSTRUCTION FORMAT. Each field is separated by a TAB, and each line of coding is terminated by a carriage return. All fields except the comments field must appear in each line. If, within a line, a field (except comments) is to be skipped, a TAB must be punched for that field. For example, if the location field is to be blank, a TAB must be punched at the beginning of the line. Space codes are ignored except in the comments field. The last word on tape must be an END pseudo instruction to terminate the read operation. The maximum number of characters permitted in each field is as follows:

Location field	-	8 characters
Operation Field	-	3 characters (always)
B-Field	-	1 character (0-7)
M-Field	-	17 characters
Comments Field (Optional)	-	30 characters (in card line)

NOTES:

1. If the comments field contains more than 30 characters, FLAP generates additional lines of REM pseudo instructions, and the excess in any line will appear as succeeding lines in the listable output. As many REM lines will be generated as are necessary.

2. Remarks may be included in a line of symbolic input in any field, provided that not more than 30 characters are used in one line and the last character is followed by a / and a CR. The combination of / CR causes the line containing the remarks to be ignored by the assembly program. This same combination of / CR may be used for deleting lines during tape preparation, as long as no more than 30 characters appear in the entire line.

CAUTION: If an M-field contains only a / character as described earlier, the field must be terminated by a TAB, even if the comments field is not used. Otherwise, the line will be ignored by FLAP.

Card Input Format

The restrictions placed on each field for card input are the same as those described for paper tape input. Only one line may be punched on a card. Card input format is as follows:

Column	
1-8	Location Field (Up to 8 alphanumeric characters)
9	Blank
10-15	Operation Field (3 characters always)
16	Blank
17-18	B-Field (One character, 0-7)
20-40	M-Field (Up to 17 characters)
41-80	Comments Field (Up to 30 characters)

The last card in a symbolic deck must be an END card (END in operation field) to terminate read operation. See NOTES under paper tape input format for further information on remarks field.

FLAP OUTPUT

Output from the FLAP assembly program is of two kinds; listable and binary. The listable output is in a 9-field format as follows:

Error Flag	INTERFOR Instruction (octal)	INTERFOR Address	Location Field	Operation Field	B-Field	M-Field	Comments Field

and represents an assembled line of symbolic coding.

The binary output is the object program to be executed by the interpreter routine and is loaded under control of FLOADER, the INTERFOR load routine. Binary output is in binary card format on paper tape. Listable output may be on paper tape or on the 166 printer. If both listable and binary output are on paper tape, the listable output appears first. The listable output medium is specified by inserting a parameter in the A register as described in the operating instructions.

FLAP Error Flags

The following symbols may occur in the Error Flag column of the listable output.

L - Location symbol contains illegal character. Correct translation of M-fields containing this symbol is not guaranteed

or

Location symbol inappropriate for pseudo instruction in OP-field, such as the END pseudo instruction.

D - Location field contains a symbol appearing in another location field. The value of the symbol in the symbol table will be the address of the last location using this symbol.

O - Illegal mnemonic code in OP-field. OP-field of current instruction is cleared to zero.

B - Illegal B term.

M - Illegal character in M-field, or term does not conform to M-field rules.

U - Undefined symbol in M-field (i.e., does not appear in a location field within program being assembled).

P - Undefined M-field in pseudo instruction.

FLOADER

The FLOADER routine loads the binary object program tapes that are prepared by FLAP. When loaded, the object program is ready for execution by the interpreter routine. Programs loaded by FLOADER may be relocated by specifying a relocation constant as described in the FLOADER operating instructions in part VI. This relocation constant is added to the address assigned to each word in the object program and to the M terms as described in the instruction format in this section.

FLOADER examines two frames of the binary input tape at a time. When a load address is encountered, FLOADER replaces the previous load address with the new address. When INTERFOR instructions are encountered, they are stored in sequential locations starting at the last load address specified. If a DEC number is encountered in the object program tape, the load routine converts from floating point decimal to floating point binary. The last two frames on the binary output tape are the checksum which is examined by FLOADER to check the accuracy of the read operation.

If the interpreter routine is in storage when FLOADER is executed, FLOADER will set up the first address to be executed by the interpreter. This address is specified by the M-field of the END pseudo instruction in the symbolic program.

PART VI

INTERFOR OPERATOR'S GUIDE

The INTERFOR system operating instructions are arranged in order of use. They include the operating procedures for FLAP, FLOADER, and the interpreter. Each procedure consists of the required console operations, and the normal and error stops and printout, if any. Corrective action for error stops is included where feasible. SWAP catalog codes are given for the FLAP, FLOADER and INTERFOR tapes.

FLAP OPERATING PROCEDURES

The FLAP program is contained on paper tape AA1.02 in machine loadable bi-octal format, and must be loaded at location 0000, bank 0. The input and listable output media (cards, paper tape, or printer) for FLAP is determined by entering the proper parameter in the A register before executing FLAP. Binary output is always on paper tape.

To load the FLAP assembly tape and execute the assembly program:

1. Position FLAP program tape in paper tape reader; turn reader on.
2. Master clear; clear all bank settings to 0.
3. Machine load FLAP program tape starting at location 0.
4. At completion of FLAP load, place RUN switch in neutral, and position symbolic program input in reader. Turn punch on.
5. Set P register to 0241. (Do not master clear.)
6. Enter one of the following parameters in the A register to determine FLAP input-output media:

   0000 = Paper tape input, paper tape listable output

   0001 = Paper tape input, 166 printer listable output

   0010 = 167 card input, paper tape listable output

   0011 = 167 card input, 166 printer listable output.
7. Press RUN switch. FLAP will proceed to assemble symbolic program.

Output of FLAP consists of the listable output, either on paper tape or the 166 printer, and the binary object program, which is always on paper tape.

If both listable and binary output are on paper tape, the listable output appears before the binary output, separated by blank tape. The binary output is to be loaded by FLOADER and executed by the interpreter routine.

A second and third pass may be required as indicated by appropriate stop. If this occurs, first pass generates paper tape output that is used as input by second and third pass.

Normal
FLAP Program Stops

3072	An intermediate output tape has been generated by FLAP. To load this tape, remove from punch, position in paper tape reader, and run. (This stop may occur before second and third pass.)
4321	Final Stop. Program assembly has been completed. Another program may be assembled by repeating steps 4 through 7.

FLAP Error Stops

2757 2772	Parity check error on intermediate output tape. Frame just read has lateral parity error. Program must be reassembled.
2745	Longitudinal check error. Frame just read is not longitudinal check character. Program must be reassembled.
2577	Symbol table capacity has been exceeded. Number of symbols in object program must be reduced. Symbol table capacity is approximately 500 symbols.
2322	Length of field in symbolic line greater than 77_8.

3416	Binary output storage and intermediate input storage areas have overlapped. Reposition symbolic input tape for loading and restart assembly at 0570. Starting at 0570 produces intermediate output on paper tape. Proceed as outlined in program stop 3072.
3746	Assembly program has dropped information while processing. Most probable occurrence is during second pass when intermediate input is on paper tape. In this case, error is probably due to punch malfunction. Program must be re-assembled.

FLOADER OPERATING PROCEDURES

The FLOADER routine is contained on paper tape A4.11 and loads the binary output tapes produced by FLAP in proper format to be executed by the interpreter routine. FLOADER also provides for relocation of the binary object program by adding a relocation constant to the INTERFOR load addresses contained in the object program, as well as to the appropriate M terms. The relocation constant is entered in the A register prior to executing the load routine. FLOADER may be loaded after the interpreter routine is loaded; however, the loader occupies that area in storage assigned to the manual service subroutines contained within the interpreter, and will replace these subroutines if loaded after the interpreter routine.

If DEC pseudo instructions are contained in the symbolic program from which the binary output was obtained, FLOADER must be loaded after the interpreter routine is loaded, because the conversion of DEC constants is accomplished by an internal subroutine within the interpreter. If the manual service subroutines are to be used, then the interpreter must be reloaded.

To load the FLOADER routine and the binary program tapes:

1. Turn on paper tape reader. Master clear, set bank settings 1 to 0, and position FLOADER tape (A4.11) in reader.

2. Set P register to 7400 and machine load FLOADER tape.
3. Position binary program tape in reader.
4. Set P register to 7400, set A register to relocation constant. If no relocation constant, A register must be zero.
5. Set RUN switch. Binary object program will be loaded in the proper storage locations ready for execution by the interpreter routine.

Normal Program Stops

7543	Object program correctly loaded. Z register contains 7700, A register contains starting INTERFOR address of object program as specified by M term of END pseudo instruction in the symbolic program. If the interpreter was loaded before FLOADER, running from this halt begins execution of the object program starting with the INTERFOR address in the A register.

Error Stops

7411	Incorrect first frame on binary tape. Symbolic tape must be reassembled.
7534	Checksum error on binary tape. Run from this stop to ignore checksum error. Loader routine will proceed to normal stop.
7766	Illegal decimal number. Run from this point to obtain error printout on typewriter. Error printouts are the same as those described in the interpreter operating procedures.

CAUTION: The FLOADER routine does not check for storage range errors. Storage references in illegal banks are detected by a machine halt and a red background in the P register display.

INTERPRETER OPERATING PROCEDURES

The interpreter routine executes the binary object programs assembled by FLAP and loaded by FLOADER. The interpreter routine contains instruction load subroutines that load octal INTERFOR programs, and various service subroutines. Load formats are contained in part II.

Interpreter Tape Load

The INTERFOR interpreter routine is contained on paper tape AB1.03 and is in machine load format. To load tape:

1. Turn on paper tape reader.
2. Position interpreter tape (AB1.03) in reader.
3. Master clear and clear all bank settings to 0.
4. Set P register to 3050.
5. Machine load simulator tape

   Loading stops with P register = 0100

   A register = 0410 (checksum)

   Z register = 0000

   If tape fails to load correctly, reload starting with step 2.

Octal Instruction Load

After loading the interpreter routine, octal instructions may be loaded from Flexowriter tape or from the console typewriter.

From Flexowriter Tape

1. Turn on reader
2. Position tape in reader
3. Master clear

4. Set P register to 7400, set relocation constant in A register, and run. At completion of load, program stops with 7463 in P register.

NOTE: Last instruction on tape must be followed by a CR and a semicolon to terminate the load operation.

From Typewriter

1. Master clear
2. Set P register to 7402 and run.
3. Type in octal instructions in the format given in part II. Last instruction must be followed by a CR and a semicolon to terminate the load routine.

Interpreter Execution

After loading the object program with FLOADER or the instruction load routine, execute the object program by performing the following steps:

1. Master clear
2. Enter starting INTERFOR address in A register and run. Object program will be executed. If output is to be on paper tape, turn punch on before running.

Interpreter Manual Service Subroutines

The following manual service subroutines are part of the interpreter routine and serve as debugging aids for the programmer. However, if the FLOADER routine is loaded after the interpreter is loaded, the manual service subroutines are not available since FLOADER occupies the same storage locations. In this case, the interpreter must be reloaded if the service subroutines are to be used.

Instruction Dump

The contents of consecutive locations may be punched on paper tape or listed on the typewriter in octal notation by performing the following steps:

1. Master clear
2. Set P register to:

   7401 = paper tape dump

   7403 = typewriter dump
3. In A register, insert INTERFOR location of first word to be dumped and run.
4. At halt, insert in A register last INTERFOR location to be dumped and run. Each instruction will be dumped on the typewriter or paper tape in octal format with the INTERFOR location of each word. Subroutine halts with 7653 in P register.

Manual Data Load

Floating point decimal data may be loaded by the manual data load subroutine. Data from paper tape or the typewriter must be in the following format as explained in part II:

CR

(-) A TAB ± .DDDDDDDDD (TAB or CR) ±EEE CR

⋮

(-) A TAB ± .DDDDDDDDD (TAB or CR) ±EEE CR

;

The load address A must be specified for each word. The minus sign before A is optional and indicates that the load address will not be modified by the relocation constant. The + signs are also optional.

To manually load data from paper tape:

1. Turn on reader, insert data tape in reader
2. Master clear
3. Set P register to 7404 and run.
4. Program stops at 7745 after correctly loading data. If error stop 7766 occurs, run to obtain error printout on typewriter, and continue loading by repeating steps 2 and 3. Error printouts are described under Error Code Identification.

To manually load data from the typewriter:

1. Master clear
2. Set P register to 7406 and run.
3. Type in data in the above format. Program stops at 7754 after entering each word. If error stop 7766 occurs, run to obtain typewriter printout and restart with step 1.
4. Return RUN switch to neutral after all data words have been entered.
5. Master clear.

NOTE: If a typing error is made return carriage and retype the line.

Data Dump

The contents of INTERFOR locations may be non-destructively listed on the typewriter in floating point decimal format.

To dump data:

1. Master clear
2. Set P register to 7407
3. Set A register to first INTERFOR location to be dumped and run.

4. Data word will be printed out on typewriter. A register contains INTERFOR address of next consecutive location. P register contains 4106.

5. Move RUN switch to neutral and back to run to obtain next consecutive word.

Accumulator Dump

Dumps the contents of the INTERFOR accumulator in decimal floating point form. The dump routine destroys the contents of the accumulator.

To dump the accumulator:

1. Master clear
2. Set P register to 7410 and run.
3. Word is dumped on typewriter and program stops at 6351.

Interpreter Normal Program Stops

0077 - End interpreter load, A register contains checksum of interpreter tape

7463 - End instruction load

7754 - End manual data load

7653 - End instruction dump

4106 - End data dump, A register contains INTERFOR address of next word to be dumped

6351 - End accumulator dump, A register contains 1244

6751 - Breakpoint stop, A register contains INTERFOR address of breakpoint stop instruction. The least significant digit in the Z register is the breakpoint stop designator h.

6351 - Selective stop, A register contains INTERFOR address of selective stop instruction.

Simulator Error Stop

The simulator routine has only one error stop. To identify the type of error, the operator may call for an error printout on the typewriter or on paper tape.

The following display indicates that an error has occurred:

P = 7766 A = 0013 Z = 7700

Typewriter Printout

To identify the type of error by typewriter printout, move RUN switch to neutral and back to run; the error code will be typed.

Paper Tape Output

To obtain the error code on paper tape:

1. Master clear
2. Turn on paper tape punch
3. Set P register to 7762 and run.

A stop code precedes the error code on the paper tape output to identify the beginning of the error code words.

Error Code Identification

Error codes are printed or punched in the following format:

Line 1. Error code

Line 2. Indicates the index register used by instruction causing error.

Lines 3 and 4. Instruction word causing error and its INTERFOR location. L or U indicates which instruction is in error.

Error Codes

c - Illegal op code

d - Index designator error

e - Exponent error. Exponent exceeds permitted range.

$$(10_{10}^{-308} \leq e < 10_{10}^{307}).$$

i - Indicates index register errors:

(1) Index register contents negative for search instruction, or

(2) contents of index register greater than y for an index skip instruction.

v - INTERFOR address range error (outside areas 0020-0537, 2000-7777).

s - Skip instruction placed in lower half of word where used in equality or threshold search, or in storage or index skip instructions.

a - Illegal argument used for input to a subroutine.

Example of error output:

```
c
b  0005
u  2042   6010   0020
          7540   1157
```

Indicates an illegal op code in the upper instruction at INTERFOR location 2042. 6010 0020 indicates the instruction in error. The 1 in 6010 indicates index register 1 was in use. The printout b 0005 indicates that the contents of index register number 1 is 0005. The lower instruction in location 2042 is listed below the upper instruction. If the error had occurred in the lower instruction, an L would be printed before the address, in place of the U.

APPENDIX

EXTERNAL INTERFOR SUBROUTINES

The INTERFOR subroutine library consists of subroutines and a PLOT subroutine for programming the 165 Plotter. Subroutines are written in a combination of INTERFOR instructions and basic 160-A instructions, and are relocatable in the available storage locations by use of the instruction load subroutine in the interpreter. All of the subroutines are on punched paper tape in Flexowriter code, which is recognized by the instruction load subroutine. Octal listings of the subroutines are included.

External subroutines are:

ARCTANGENT X
SINE X
COSINE X
ARCCOSINE X
ARCSINE X
BASIC (SERIES EXPANSION)
EXPONENTIAL (2^x, e^x 10^x)
SQUARE ROOT
LOG TO BASE 2
PLOT

All routines are written starting at location 0000 and must be relocated at load time. All subroutines must be loaded in bank 0, therefore they may only occupy INTERFOR locations 0020-00537.

A slash indicates that an address is written relative to the beginning location. For example, 0034/ refers to an address which is 34_8 locations following the base location in which a routine is relocated.

Routines are entered by a return jump (7540xxxx) instruction to the location specified as the entrance of the subroutine. The return jump may be either an upper or a lower instruction. Return from the subroutine will, in all cases, be to the upper instruction of the next word of the main program.

A. IDENTIFICATION

TITLE: INTERFOR Subroutine - Arctangent

IDENTIFICATION NUMBER: INTERFOR-1

PROGRAMMER: Payne

B. PURPOSE

Given a number X, compute the arctangent of X using the following Maclaurin series approximation:

$$\text{Arctan}^* X = \sum_{i=0}^{7} C^{2i+1} X^{2i+1}$$

where $C_1 = .99999,93329$ $C_9 = .09642,00441$

$C_3 = -.33329,85605$ $C_{11} = -.05590,98861$

$C_5 = .19946,53599$ $C_{13} = .02186,12288$

$C_7 = -.13908,53351$ $C_{15} = -.00405,40580$

C. USAGE

1. Operational Procedure:

   If $-1 \leq X \leq 1$, use the series directly.

   If $X > 1$, then take Arctangent of $1/X$ and subtract this value from $\pi/2$ to get arctangent of X.

   If $X < -1$ then subtract Arctangent $1/X$ from $-\pi/2$ to get Arctangent of X.

2. Entry:

   A contains the number X in floating point

3. Exit:

   A contains the answer in radians

   B6 is used, but is reset to its original value before exit.

4. Error Conditions:

   None

5. Subroutines Used:

   Basic

6. Remarks and Restrictions:

   Uses 51_8 permanent locations.

   Basic subroutine is located at 0040/ (incorporated on an instruction load tape)

   See INTERFOR-8 for subroutine listing.

A. IDENTIFICATION

TITLE: INTERFOR Subroutine - Sine Cosine

IDENTIFICATION NUMBER: INTERFOR-2

PROGRAMMER: Payne

B. PURPOSE

Given X, compute the Sin X or Cos X (where X is in radians)

C. USAGE

1. Operational Procedure:

   The framework for the program is around the Sin X routine.

   Cos X uses the Sin X as a subroutine where Cos X = Sin ($\pi/2$ + X)

2. Entry:

   A contains the angle X in radians

3. Exit:

   A contains the value of Sin X or Cos X.

   B6 is used, but reset to its original value before exit.

4. Error Conditions:

   None.

5. Subroutines Used:

   Basic is incorporated in Sin X. Cos X incorporates both Basic and Sin X.

6. Remarks and Restrictions:

   Sin X uses 44 permanent INTERFOR locations.

   Cos X uses 4 permanent INTERFOR locations.

This is a relocatable program on flex tape with entry address of Sin X at 0000/ and Cos X at 0045/. Basic entry address is at 0033/. Accuracy is within 1 or 2 in the ninth decimal place.

```
0000    75007777    entrance to sin x subroutine
        32000032/

        20000016/
        30000015/

        31000016/
        32000021/

        22200004/
        31000021/

        65000024/
        30000021/

        65000022/
        75000007/

        75000010/
        00000000

        32000020/
        30000023/

        20000016/
        32000016/

        20000017/
        50000000

        12000014/
        75400033/    entrance to basic to evaluate series

        32000016/
        75000051/   go to check  limit of one

        00400017/
        00000025/

        00410000
        00000000

        00000000
        00000000

        00000000
        00000000

        00016000
        00000000
```

INTERFOR-2 (Cont.)

```
00036000
00000000

00012000
00000000

00022000
00000000

00023000
00000000

00013110
17662440

00006452
35700553

77742431
24253546

77706311
06303104

77632366
27562212

77752427
31403333

7500 7777
51000041/       exit to basic language

2056 4067
0404 4056
2016 4220
2014 4214

0400 4014
4015 4016

2200 4011
4012 7001

3200 7777       LOOP
5000 0000

3060 7777       TABLE
51000033/       exit to basic language
```

INTERFOR-2 (Cont.)

```
        2056 0701
        4056 6205

        2067 4056
        2341 7057

        2202 7057
        0040/0000

        75007777          cos x entrance
        30000050/

        75400000/
        00000000

        75000045/
        00000000

        00013110
        17662504

        3100 0022/
        22300053/

        1200 0022/
        7500 0000/

        3000 0023/
        2230 0055/

        3100 0022/
        7500 0000/

        1300 0022/
        7500 0000/
;;;;;
```

A. IDENTIFICATION

TITLE: INTERFOR Subroutine - Arcsine, Arccosine

IDENTIFICATION NUMBER: INTERFOR-3

PROGRAMMER: Payne

B. PURPOSE

Given a number X, this subroutine will find the Arccos of X or the Arcsin of X using the following approximation:

$$\text{Arccos } X = \sqrt{1 - X} \cdot \psi^*(X)$$

where $\psi^*(X) = a_0 + a_1 X + a_2 X^2 + \ldots + a_7 X^7$

and

a_0	= 1.5707,9630,50	a_4	= .0308,9188,10
a_1	= -.2145,9880,16	a_5	= -.0170,8812,56
a_2	= .0889,7898,74	a_6	= .0066,7009,01
a_3	= -.0501,7430,46	a_7	= -.0012,6249,11

C. USAGE

1. Operational Procedure:

   Arcsin X is evaluated by using Arccos X as a subroutine where Arcsin X = $\pi/2$ - Arccos X.

2. Entry:

   A contains the number X.

3. Exit:

   A contains Arcsin X or Arccos X.

   B6 is used, but is reset to its original value before exit.

4. Error Conditions:

   None.

5. Subroutines Used:

   Arccos X incorporates SQRT and Basic.

   Arcsin X incorporates SQRT, Basic, and Arccos X.

6. Remarks and Restrictions:

   Arccos X uses 55 permanent locations.

   Arcsin X uses 5 permanent locations.

   This is a relocatable program on flex tape. Entry is: Arccos X at 0000/, Arcsin X at 0055/. Basic is used to evaluate the series.

   The SQRT subroutine is also incorporated with the following entry points: Basic at 0022/, SQRT at 0034/. Accuracy is within 3 or 4 in the eighth decimal place.

```
0000    75007777
        20000007/

        12000011/
        31000007/

        75400034/
        00000000

        20000010/
        12000006/

        75400022/
        00000000

        32000010/
        75000000/

        00700007/
        00000012/

        00000000
        00000000

        00000000
        00000000

        00012000
        00000000

        00013110
        17662376

        77757335
        37621646

        77742661
        32471415

        77737154
        03442255

        77723750
        20760246

        77726137
        34311450

        77703324
        20611326
```

INTERFOR-3 (Cont.)

```
77666453
32123762

7500 7777
5100 0030/  exit to basic language

2056 4067
2012 0111

0110 4056
2016 4216

2014 4210
0400 4014

4016 2226
4012 7001   return to simulator

3200 7777   LOOP
5000 0000

3060 7777   TABLE
5100 0022/  exit to basic language

2056 0701
4056 6205

2067 4056
2341 7057

2202 7057
0027/4011

75000000
20000054/

51000042/
20126203

22556102
22544206

20120111
01100103

10377777
42462037

42457001
00000047/
```

INTERFOR-3 (Cont.)

```
33000052/
30000052/

32000053/
51000034/

20143630
63102015

36266305
20163624

12116624
23237057

20000052/
50000000

12000054/
75000042/

77763036
03000000

00000000
00000000

00002000
00000000

00000000
00000000

75007777
75400000/

20000060/
12000061/

31000060/
75000055/

00000000
00000000

00013110
17662504

;;
```

A. IDENTIFICATION

TITLE: INTERFOR Subroutine - Basic (series expansion)

IDENTIFICATION NUMBER: INTERFOR-4

PROGRAMMER: Mansfield

B. PURPOSE

Compute the value of a given series $\sum_{i=0}^{n} C_i x^i$ of n terms.

(n = 1, 2 . . . n)

C. USAGE

1. Operational Procedure:

   A code word must have been previously loaded in A before entering the Basic subroutine. The code word is in the octal format:

   NNNOXXXX
   OOOOTTTT

   where NNN is the number of terms in the series not including C_a, XXXX is the address of the floating point number X to be used by the series and TTTT is the address of the first coefficient C_a in the table of coefficients. The O character positions are ignored by the subroutine and may be any digit.

2. Entry:

   A contains the code word.

3. Exit:

   A contains the result of the series expansion.

   B6 is used, but reset to its original value before exit.

4. Error Conditions:

   None

5. Remarks and Restrictions:

Uses 11_8 permanent INTERFOR locations. This routine is on relocatable Flexowriter tape.

See INTERFOR-8 for subroutine listing.

A. IDENTIFICATION

TITLE: INTERFOR Subroutine - Exponential

IDENTIFICATION NUMBER: INTERFOR-5

B. PURPOSE

Calculate 2^x, e^x, or 10^x

C. USAGE

1. Operational Procedure:

This basic subroutine calculates 2^x, but e^x and 10^x may be calculated by performing one preliminary multiplication on X using a constant contained in the subroutine. The multiplication factors for e^x and 10^x are stored in the subroutine as shown:

e^x factor is stored at 0033_8 plus the starting address

10^x factor is stored at 0034_8 plus the starting address

As an example of the use of this subroutine, it is assumed that the subroutine has been loaded starting at location 0300. The main program to enter the subroutine by a return jump is as follows for 2^x:

Loc.	Upper inst.		Lower inst.		Remarks
2400	1200	2100	7540	0300	Load x in accumulator from 2100 and Jump to the subroutine.

For e^x a possible entrance is

Loc.	Upper inst.		Lower inst.		Remarks
2377	1200	2100	3200	0333	Load x into acc. Mult. by factor in 0733 (in subroutine)
2400	7540	0300	0000	0000	Return jump to sub-routine, lower instruction is skipped.
2401	2000	2200	0000	0000	On return from sub-routine, store e^x in 0200, the lower in-struction is a pass order.

2. Entry:

   A contains the number x (premultiplied if e^x or 10^x are desired)

3. Remarks and Restrictions:

   Entry is by a return jump to the first address in the subroutine. 35_8 permanent locations are required. This routine is on a relocatable flex tape.

INTERFOR-5 EXPONENTIAL

```
0000    75000000
        20000027/

        22300002/
        30000025/

        31000024/
        30000026/

        20000030/
        13000030/

        30000027/
        20000027/

        32000027/
        20000031/

        30000023/
        20000032/

        12000022/
        33000032/

        30000021/
        32000031/

        30000020/
        31000027/

        20000032/
        30000027/

        30000027/
        33000032/

        51000000/
        22576202

        14374254
        26520701

        42504650
        56466502

        20123244
        40127001

        00022705
        12163123
```

INTERFOR-5 (Cont.)

```
77732156
31470403

00033422
34153701

00072567
04132057

00002000
00000000

00012000
00000000

00410000
00000000

00000000
00000000

00000000
00000000

00000000
00000000

00000000
00000000

00012705
12163123

00023244
32360230
```

A. IDENTIFICATION

TITLE: INTERFOR Subroutine - Square Root

IDENTIFICATION NUMBER: INTERFOR-6

B. PURPOSE

Given a number X in the accumulator, find the square root of the number by the use of the Newton iteration method and leave the square root in the accumulator.

C. USAGE

1. Entry:

   A contains the number X in floating point

2. Exit:

   A contains the square root in floating point

3. Remarks:

   This routine used 21 permanent locations and is on relocatable flex tape.

```
0000    75000000
        20000020/

        51000005/
        20126203

        22556102
        22544204

        20120114
        10377777

        42502037
        42477001

        33000016/
        30000016/

        32000017/
        51000000/

        20143634
        63122015

        36326307
        20163630

        12116303
        70010013/

        23017057
        00000000

        20000016/
        50000000/

        12000020/
        75000005/

        77763036
        03000000

        00000000
        00000000

        00002000
        00000000

        00000000
        00000000
;am9
```

A. IDENTIFICATION

TITLE: INTERFOR Subroutine - Log to Base 2

IDENTIFICATION NUMBER: INTERFOR-7

B. PURPOSE

Given a floating point number in A, calculate the log to the base 2 of this number.

C. USAGE

1. Entry:

   A contains the number X in floating point.

2. Exit:

   A contains the log to the base 2 of the value of X.

3. Remarks and Restrictions:

   Routine uses 47_8 permanent locations and is on relocatable flex tape.

INTERFOR-7 **LOG BASE 2**

```
0000    75000000
        51000004/

        20136204
        04307101

        77562012
        07014233

        04014012
        70017777

        20000023/
        30000024/

        20000025/
        12000023/

        31000024/
        33000025/

        20000023/
        32000023/

        20000025/
        12000027/

        75400035/
        00000000

        32000023/
        30000026/

        51000022/
        23106205

        27124313
        20366102

        04004316
        04004350

        23226103
        23036110

        33266304
        43305760

        65100413
        37634255
```

INTERFOR-7 (Cont.)

```
23361736
42537001

30000034/
75000000/

00000000
00000000

00012650
04742720

00000000
00000000

00002000
00000000

00300025/
00000030/

00022705
12163073

00003661
30443103

00002234
33020030

77763362
27223225

00000000
00000000

75007777
51000043/

2056 4067
0403 4056

2016 4220
2014 4212

0400 4014
4015 4016

2200 4011
4012 7001
```

INTERFOR-7 (Cont.)

```
3200 7777
5000 0000

3060 7777
5100 0035/

2056 0701
4056 6205

2067 4056
2341 7057

2202 7057
0042/4011
;;
```

A. IDENTIFICATION

TITLE: INTERFOR Subroutine - Trig Routines

IDENTIFICATION: INTERFOR-8

B. PURPOSE

This tape contains the following subroutines with the given entrances:

SIN X	0000/
COX X	0045/
ARCTAN X	0051/
BASIC	0033/

C. USAGE

See writeups of the original routines.

Contains 112_8 locations.

```
0000    75007777
        32000032/

        20000016/
        30000015/

        31000016/
        32000021/

        22200004/
        31000021/

        65000024/
        30000021/

        65000022/
        75000007/

        75000010/
        00000000

        32000020/
        30000023/

        20000016/
        32000016/

        20000017/
        50000000

        12000014/
        75400033/

        32000016/
        75000000/

        00400017/
        00000025/

        00410000
        00000000

        00000000
        00000000

        00000000
        00000000

        00016000
        00000000
```

INTERFOR-8 (Cont.)

```
00036000
00000000

00012000
00000000

00022000
00000000

00023000
00000000

00013110
17662450

00006452
35700553

77742431
24253546

77706311
06303104

77632366
27562212

77752427
31403333

7500 7777
51000041/       exit to basic language

2056 4067
2012 0111

0110 4056
2016 4216

2014 4210
0400 4014

4016 2226
4012 7001

3200 7777       return to simulator
5000 0000       LOOP

3060 7777       TABLE
51000033/       exit to basic language
```

INTERFOR-8 (Cont.)

```
2056 0701
4056 6205

2067 4056
2341 7057

2202 7057
0040/4011

75007777        cos entrance
30000050/

75400000/
00000000

75000045/
00000000

00013110
17662523

75007777        arctan entrance
50000000

65000073/
75000054/

65000074/
75000063/

20000106/
12000073/

33000106/
75400065/

20000106/
22200061/

12000075/
31000106/

75000051/
00000000

12000110/
31000106/

75000051/
00000000
```

INTERFOR-8 (Cont.)

```
75400065/
00000000

75000051/
00000000

75007777
20000106/

32000106/
20000107/

12000071/
75400033/

32000106/
75000065/

00700107/
00000076/

00000000
00000000

00012000 00000000
00016000 00000000
00017110 17662523
00003777 37750635
77766525 06063706
77753142 00511553
77756163 14302654
77743053 27673200
77737450 01603612
77722630 26241657
77706046 27713206
00000000 00000000
00000000 00000000
00013110 17662523
;;;;
;;
```

A. IDENTIFICATION

TITLE: INTERFOR Subroutine - Plot

IDENTIFICATION: INTERFOR-9

CATEGORY: Mathematical Subroutine

PROGRAMMER: H. Theiste

DATE: March, 1961

B. PURPOSE

The purpose of this subroutine is to plot results on the on-line plotter in either of two ways:

1. Moving from previous plot point to present plot point in a straight line with pen down.
2. Moving from previous plot point to present plot point with pen up, plot a symbol to represent desired point.

As written, the subroutine uses output from routines written in INTERFOR language. The method is applicable for plotting either fixed or floating point results.

C. USAGE

1. Operational Procedure

   Before any plotting can be done, the subroutine must be provided with the X and Y scale factors. The scale factor (F) is determined by dividing 100 by the number of units per inch on the plotted output. Thus, if the X scale were 1 inch = 4 units and the Y scale were 1 inch = .5 units, the scale factors are

$$F(X) = \frac{100}{4} = 25$$

$$F(Y) = \frac{100}{0.5} = 200$$

These are stored in locations 0073/ and 0074/ respectively. It is the task of the user to store the X and Y scale factors in these locations. After the scale factors have been set up, the initial values of X_o and Y_o must be provided to the routine. This is accomplished by a return jump to location 0070/ of the plot subroutine with the address of X_o in B6.

Two entries are provided for plotting results from the interpreter routine, depending on whether the plot is to be a line plot or a point plot. In either case, X and any Y values must be stored in consecutive locations, with X preceding Y.

If a line plot is desired, i.e., move between points with the pen down, a return jump is made to location 0000/ with the address of the X value in B6. The pen will move from its present position to the point X, Y along a nearly straight line, and the subroutine will exit to the main program.

If a point plot is desired, i.e., move between points with the pen up and plot a symbol at the desired point, a return jump is made to location 0110/ with the address of X in B6. This subroutine uses B1, but restores the original contents.

To load the plot subroutine:

a. Interpreter routine must be in the computer.
b. Turn on reader and insert PLOT tape anywhere on leader.
c. Set P = 7400
d. A = First INTERFOR location of subroutine
e. Press Run Switch

f. Line Plot

Set B6 to location of X-coord. Return jump to first location of Plot routine.

g. Point Plot

Set B6 to location of X-coord. Return jump to location 0110/ of Plot routine.

h. Set X_o and Y_o

Set B6 to location of X_o-coord. Return jump to location 0070/ of Plot routine.

3. Space Required

110_8 INTERFOR locations (440_8 or 288_{10} 160-A locations).

4. Temporary Storage

The point plot subroutine uses B1, but restores the original contents before exit. The contents of B6 remain unchanged. Therefore, if several points are to be plotted, the address of X will always be in B6. Also X and Y are left unchanged.

11. Accuracy

The plotted point will be accurate to the nearest 1/100th of an inch.

13. Equipment Configuration

Minimum 160-A computer with CDC 165 Plotter.

D. METHOD

To plot a given point, the subroutine multiplies each coordinate (X_L or Y_L) by its scale factor and retains only the integer portion of the coefficient of the floating point number, discarding the fractional portion. It then subtracts the previous coordinates to obtain values of Δ X and Δ Y, representing actual pen motion.

The subroutine then determines which value has the greater magnitude, ΔX or ΔY. This indicates along which axis most of the pen motion will occur. The larger quantity (in magnitude) is called R_L and the other L.

R_L is divided by L to obtain the ratio R between the two motions. The directions of pen motion for R_L and L are determined and two plotter outputs are set up, P_1 and P_2. P_1 moves the pen one unit in the R_L and L directions, and P_2 moves the pen only in the R_L direction. The three quantities, R_L, L and R, are made positive to serve as counters and a fourth quantity T is obtained by subtracting 2 from R.

L is decremented by one and, if it remains positive, a P_1 output is given to the plotter; whereupon T is decremented by one and, if it remains positive, R_L is decremented by one and a P_2 pulse is given to the plotter. If, however, T goes negative R is added to T, R_L is decremented by one and the process is repeated, starting with decrementing L by one. If L goes negative, and if R_L is at least one, R_L is decremented and a P_2 output is given to the plotter until R_L goes negative, at which time the X and Y coordinate values are updated and an exit is made from the subroutine.

```
0000    75000000
        12600000

        32000073/
        30000075/

        20000024/
        31000026/

        20000026/
        12600001

        32000074/
        30000075/

        20000025/
        31000027/

        20000027/
        51000015/

        20123673
        63066114

        22711222
        34146310

        04043273
        42740401

        32704272
        61300401

        32644265
        04043261

        42637001
        00003777

        12000026/
        20000063/

        12000027/
        20000026/

        12000063/
        20000027/

        51000037/
        22336204
```

```
                22360207
                52342222

                62042232
                02075230

                22270217
                52246125

0030            74000037/
                74007400

                23131362
                43152312

                13654314
                23104247

                23114267
                75067420

                23246205
                22037057

                44010036/
                23257057

                12000026/
                75000041/

                12000026/
                33000027/

                20000063/
                31000071/

                31000071/
                20000072/

                12000027/
                31000071/

                22300052/
                20000027/

                51000045/
                74007001

                12000072/
                31000071/
```

INTERFOR-9 (Cont.)

```
                22300060/
                20000072/

                12000026/
                31000071/

                20000026/
                51000045/

                74007001
                00000000

                12000026/
                31000071/

                22300055/
                20000026/

                51000052/
                04006417

                12000024/
                20000026/

                12000025/
                20000027/

                75000000/
                00000000

                30000063/
                20000072/

                12000026/
                31000071/

                20000026/
                75000042/

0064            12600000
                32000073/

                30000075/
                20000026/

                12600001
                32000074/

                30000075/
                20000027/
```

INTERFOR-9 (Cont.)

```
          75000000
          75000064/

          00012000
          00000000

0075      00410000
          00000000

          57100107/
          50107440

          57100033/
          75400000/

          50107420
          57100033/

          53100112/
          51000103/

          74047404
          74207001

          75000000
          75000076/

          75000000
          75400103/

          51000104/
          74027402

          74107410
          74107410

          74017401
          74017401

          74047404
          74047404

          74027402
          74407410

          74107001
          00000000

          75000000
          75400103/
```

INTERFOR-9 (Cont.)

```
51000113/
74127412

74117411
74057405

74067406
04006425

75000000
75400103

51000117/
74107410

74107410
74407406

74067420
74017401

74017401
74407402

74027001
```

Other publications concerning programming and programming systems for the Control Data Corporation 160-A Computer are:

160-A Programming Manual	#145b
Fortran Autotester	#186a
Satellite Programming	#187
160-A Programming Systems	#502
160-A Fortran/General Information	#505
OSAS-A/160-A Assembly System	#507
INTERFOR	#512
Peripheral Processing Package	#517

CONTROL DATA
CORPORATION

PROGRAMMING SYSTEMS BULLETIN

Revisions to the INTERFOR/Reference Manual

Control Data Publication Number 512

This bulletin describes the necessary revisions that should be made to the INTERFOR Manual, Publication #512, to make the manual conform with recent modifications to the INTERFOR system.

Page 59 -	Item 4.	(middle of page)	Change	7653 to 7655
Page 60 -	Item 4.	(top of page)	Change	7745 to 7756 7766 to 7767
	Item 3.	(middle of page)	Change	7754 to 7756 7766 to 7767
Page 61 -	Under Normal Stops		Change	0077 to 0100 7463 to 7464 7754 to 7756 7653 to 7655

CONTROL DATA CORPORATION
Application Services Department
3330 Hillview Avenue
Palo Alto, California

November 1962 PSB-BB01

CONTROL DATA SALES OFFICES

ALBUQUERQUE, N. M., 937 San Mateo, N.E., Phone 265-7941

BEVERLY HILLS, CALIF., 8665 Wilshire Boulevard, Phone OL 2-6280

BIRMINGHAM 13, ALA., 16 Office Park Circle, Phone TR 1-0961

BOSTON, MASS., 594 Marrett Road, Lexington, Mass., Phone VO 2-0002

CHICAGO, ILL., 840 South Oak Park Avenue, Oak Park, Ill., Phone 386-1911

CLEVELAND, OHIO, Center Building, 46 West Aurora Road, Northfield, Ohio, Phone 467-8141

DALLAS 35, TEXAS, 2505 West Mockingbird Lane, Phone FL 7-7993

DAYTON 29, OHIO, 10 Southmoor Circle, Phone 298-7535

DENVER 3, COLORADO, 655 Broadway Building, Phone AC 2-8951

DETROIT, MICHIGAN, 12800 West Ten Mile Road, Huntington Woods, Michigan

HOUSTON 27, TEXAS, 4901 Richmond Avenue, Phone MA 3-5482

ITHACA, NEW YORK, Cornell University, Rand Hall, Phone AR 3-6483

KANSAS CITY 6, MISSOURI, 921 Walnut Street, Phone HA 1-7410

MINNEAPOLIS 20, MINN., 8100 34th Avenue South, Phone 888-5555

NEWARK, NEW JERSEY, Terminal Building, Newark Airport, Phone MI 3-6446

NORFOLK 2, VIRGINIA, P.O. Box 1226, Phone 341-2245

ORLANDO, FLORIDA, P.O. Box 816, Maitland, Florida, Phone 647-7747

SAN FRANCISCO, CALIF., 885 North San Antonio Road, Los Altos, Cal., Phone 941-0904

WASHINGTON 16, D.C., 4429 Wisconsin Avenue N.W., Phone EM 2-2604

WASHINGTON 10, D.C., 1515 Ogden Street N.W., Phone RA 6-4983

CONTROL DATA

8100 34TH AVENUE SOUTH, MINNEAPOLIS 20, MINNESOTA

PUB. NO. 512