IBM Technical Newsletter

File Number 1130-30

Re: Form No. C26-5929-1, -2

This Newsletter No. N26-0557

Date August 15, 1966

Previous Newsletter Nos. N26-0551
N26-0553

IBM 1130 Subroutine Library (Form C26-5929-1, -2)

The attached pages bring the above publication up to date. Changes are indicated by a vertical line at the left of affected text, a bullet (•) at the left of the title of a changed illustration, and a bullet beside the page number of a page that should be reviewed in its entirety. Pages that contain changes are coded in the upper outside corner.

REPLACE THE FOLLOWING PAGES:

iii and iv	41 and 42
1 and 2	43 and 44
15 and 16	45 and 46
17 and 18	46.1 and 46.2
29 and 30	47 and 48
35 and 36	49 and 50
37 and 38	51 and 52
39 and 40	53 and 54
	55 and 56

ADD PAGES:

47.1 and 47.2
57 and 58
59 and 60

Note that all references in this publication to floating-point have been changed to real by this TNL. These changes occur on pages 35 through 46 and are not identified by a vertical line, although each page number is marked with a bullet.

File this Newsletter at the back of the manual. It will provide a reference to changes, a method of determining that all amendments have been received, and a check for determining if the manual contains the proper pages.

IBM Corporation, Programming Publications, Dept. 452, San Jose, Calif. 95114

PRINTED IN U.S.A.

CONTENTS

It is often necessary to repeat a group, or block, of instructions many times during the execution of a program (examples include conversion of decimal values to equivalent binary values, computation of square roots, and reading data from a card reader). It is not necessary to write the instructions each time a function is required. Instead, the block of instructions is written once, and the main program transfers to that block each time it is required. Such a block of instructions is called a subroutine. Subroutines normally perform such basic functions that they can assist in the solution of many different kinds of problems.

When a main program uses a subroutine several times, which is the common situation, the block of instructions constituting the subroutine need appear only once. Control is transferred from a main program to the subroutine by a set of instructions known as a calling sequence, or basic linkage. A calling sequence transfers control to a subroutine and, through parameters, gives the subroutine any control information required.

The parameters of a calling sequence vary with the type of subroutine called. An input/output subroutine requires several parameters to identify an input/output device, storage area, amount of data to be transferred, etc.; whereas an arithmetic/functional subroutine usually requires one parameter representing an argument. Each calling sequence used with the 1130 System subroutines consists of a CALL or LIBF statement (whichever is required to call the specific subroutine), followed by DC statements that make up the parameter list. The calling sequences for the various subroutines in the subroutine library are presented later in the manual. Each subroutine is self-contained, so that only those routines required by the current job are in core storage at program execution time.

The interrupt service subroutines (ISS) transfer data from and to the various input/output devices attached to the computer. The subroutines handle all of the details peculiar to each device, including the usually complex interrupt functions, and can control input/output devices simultaneously and asynchronously.

ISS CHARACTERISTICS

To fully comprehend subsequent descriptions of each ISS, the user should be familiar with the following characteristics, which are common to all ISS:

- Methods of data transfer
- Interrupt processing
- ILS (interrupt level subroutine)
- ISS operation
- General error handling procedures
- Basic calling sequence

METHODS OF DATA TRANSFER

IBM 1130 I/O devices and their related subroutines can be differentiated according to their methods of transmitting and/or receiving data.

Direct Program Control

The serial I/O devices operate via direct program control, which requires a programmed I/O operation for each word or character transferred. A character interrupt occurs whenever a character I/O operation is completed. Direct program control of data transfer is used for the serial devices including the card read punch, paper tape reader and punch, console printer, input keyboard, 1132 Printer, and plotter.

Data Channel

Disk storage operates via a data channel, which requires an I/O operation only to initiate data transfer. A device is provided with control information, word-counts, and data from the user's I/O area. Once initiated, data transfer proceeds asynchronously to program execution. An operation-complete interrupt signals the end of an I/O operation when all data has been transferred.

INTERRUPT PROCESSING

Interrupt processing is divided into two parts, level processing and device processing. The flow of logic in response to an interrupt is: user program interrupted, level processing begun, device processing begun and completed, level processing completed, and user program continued.

Level Processing

Level processing consists of selecting the correct device processing routine, performing certain housekeeping functions, and clearing the level by a BOSC instruction when interrupt processing is complete.

Level processing is done by the ILS (interrupt level subroutines). Entered by interrupts, ILS give temporary control to a device processing subroutine (ISS) and eventually return control to the user program. The interrupt entrance address is stored, at load time, in the appropriate interrupt branch address; location 8 for interrupt level zero (ILS 00), location 9 for interrupt level one (ILS 01), . . . , location 12 for interrupt level four (ILS 04). The device processing entrance address is computed at load time from identifying information, stored in the ILS, in the compressed ISS header card, and in the loader interrupt transfer vector.

Device Processing

Device processing consists of operating an I/O device, processing the interrupts, and clearing the device by an XIO (sense DSW) instruction when interrupt processing is complete.

Device processing is done by the ISS (interrupt service subroutines). They can be entered by a calling instruction (LIBF or CALL), which either requests certain initialization to be done or requests an I/O device operation. They can also be entered by the ILS as part of the interrupt processing. The calling entry point is specified by an ISS statement.

The interrupt entry point(s) is set up in the ISS and identified in the ILS. It is entered indirectly through a branch address table within the ILS.

INTERRUPT LEVEL SUBROUTINES

The ISS package services all input/output interrupts with a set of ILS (interrupt level subroutines), loaded as part of the subroutine library.

Description

There is one ILS for each interrupt level used. Each routine determines which device on its level caused a particular interrupt; preserves the contents of the accumulator, the accumulator extension, index register one (XR1), and the Carry and Overflow indicators; and transmits identifying information to the ISS.

Interrupt service subroutines are loaded first so that the loader loads only the ILS that are required. For example, if a main program does not call the 1132 printer subroutine, the routine for interrupt level 1 need not be loaded because no interrupts will occur on that level. An ILS cannot be called. Therefore, an ILS is loaded only if requested by a loaded ISS (see "ISS-Define Interrupt Service Entry Point" in IBM 1130 Assembler Language, Form C26-5927).

When the ILS are loaded, the core addresses assigned to them are inserted into the computer words, reserved for that purpose, starting at word 8. Interrupts occurring during execution of a user program cause an automatic Branch Indirect, via the interrupt level word, to the correct ILS.

Recurrent Subroutine Entries

Recurrent entries to a subroutine can result from subsequent interrupts. For example, during execution of the console printer subroutine, a disk interrupt can start execution of a subroutine to handle the condition that caused the disk interrupt. If this handling includes calling the console printer subroutine, certain information is destroyed, the most important of which is the return address of the program that originally called the console printer.

To prevent the loss of data resulting from a recurrent entry, the user must provide the programming required to save the return address and any other data needed to continue an interrupted subroutine after an interrupt has been serviced.

NOTE: All ISS were written with the assumption that all LIBF's would be executed from the mainline level of interrupt priority. There are no provisions in any ISS to handle recurrent entries.

ISS OPERATION

This section briefly describes the operation of the ISS (interrupt service subroutines). This description, along with some basic flowcharts, should make it easier for the reader to understand the descriptions of individual subroutines presented later.

ISS Subdivision

Each ISS is divided into a call routine and an interrupt response routine. The call routine is entered when a user's calling sequence is executed; the interrupt response routine is entered as a result of an I/O interrupt.

Call Routine

Each ISS saves and restores the contents of the accumulator and extension, index registers, and the Carry and Overflow indicators. The call routine, illustrated in Figure 1, has four basic functions:

1. Determine if any previous operations on the specified device are still in progress.
2. Check the calling sequence for legality.
3. Save the calling sequence.
4. Initiate the requested I/O operation.

The flow diagram (Figure 1) is not exact for any one ISS. It is only a general picture of the internal operation of a call routine.

Determine Status of Previous Operation. This function can be performed by using a programmed routine-busy indicator to determine if a previous operation is complete. The CARD1 subroutine is a good example. When an operation is started on the 1442, a subsequent LIBF CARD1 for the 1442 is not honored until the routine-busy indicator is turned off. A call to any other ISS subroutine, such as TYPE0, is not affected by the fact that the CARD1 subroutine is busy.

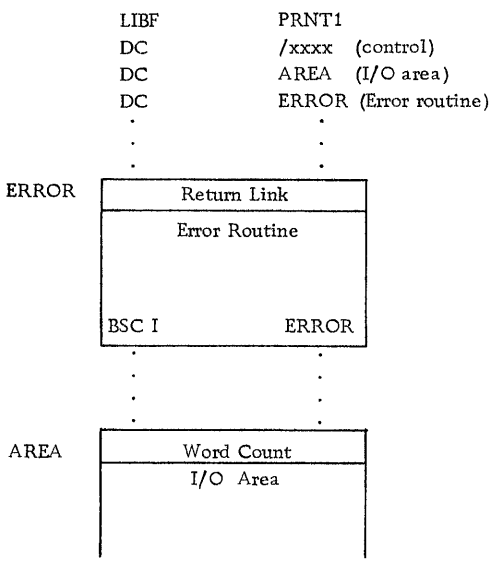
Each ISS, except PAPTN, can use one programmed routine-busy indicator to determine if a previous operation is complete. The PAPTN subroutine uses two busy indicators, one for the paper tape reader and one for the punch. If an operation is started on the reader, a subsequent LIBF PAPTN for the reader is not honored until the Reader Busy indicator is turned off. However, an LIBF PAPTN for the paper tape punch is treated in the same manner as a call to any other ISS and is not affected by the fact that the reader is busy.

The information was stored on the disk by the DPIR.

PRINTER SUBROUTINES

The printer subroutine PRNT1 handles all print and carriage control functions relative to the IBM 1132 Printer. Only one line of data can be printed, or one carriage operation executed, with each call to the printer subroutine. The data in the output area must be in EBCDIC form, packed two characters per computer word. (See Data Codes.)

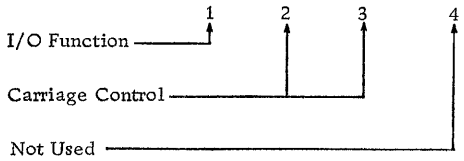
Calling Sequence



The calling sequence parameters are described in the following paragraphs.

Control Parameter

This parameter consists of four hexadecimal digits which are used as shown below.



I/O Function

The I/O function digit specifies the operation to be performed on an 1132 Printer. The functions, their associated digital values, and the required parameters are listed and described below.

Function	Digital Value	Required Parameters*
Test	0	Control
Print	2	Control, I/O Area, Error
Control Carriage	3	Control
Print Numerical	4	Control, I/O Area, Error

*Any parameter not required for a particular function must be omitted.

Test. Branches to LIBF+2 if the previous operation has not been completed or to LIBF+3 if the previous operation has been completed.

Print. Prints characters from the user's I/O area, checking for channel 9 and 12 indications. If either of these conditions is detected, the subroutine branches to the user's error routine after the line of data has been printed. Upon return from this error routine, a skip to channel 1 is initiated or the function is terminated, depending upon whether the Accumulator is non-zero or zero.

Control Carriage. Controls the carriage as specified by the carriage control digits listed in Table 2.

Print Numerical. Prints only numerals and special characters from the user's I/O area and checks for channel 9 and channel 12 indications. See Print above.

Carriage Control

Digits 2 and 3 specify the carriage control functions listed in Table 2. An immediate request is executed before the next print operation; an after-print request is executed after the next print operation and replaces the normal space operation.

If the I/O function is print, only digit 3 is examined; if the I/O function is control, and digits 2 and 3 both specify carriage operations, only digit 2 is used.

NOTE: An after-print request will be lost if it is followed by an immediate request or by a print with spacing suppressed. If a series of after-print requests is given, only the last one will be executed.

Table 2. Carriage Control Operations

Digit #2: Immediate Carriage Operations
Print Functions Not Used Control Function 1 - Immediate Skip To Channel 1 2 - Immediate Skip To Channel 2 3 - Immediate Skip To Channel 3 4 - Immediate Skip To Channel 4 5 - Immediate Skip To Channel 5 6 - Immediate Skip To Channel 6 9 - Immediate Skip To Channel 9 C - Immediate Skip To Channel 12 D - Immediate Space Of 1 E - Immediate Space Of 2 F - Immediate Space Of 3
Digit #3: After-Print Carriage Operations
Print Functions 0 - Space One Line After Printing 1 - Suppress Space After Printing Control Function 1 - Skip After Print To Channel 1 2 - Skip After Print To Channel 2 3 - Skip After Print To Channel 3 4 - Skip After Print To Channel 4 5 - Skip After Print To Channel 5 6 - Skip After Print To Channel 6 9 - Skip After Print To Channel 9 C - Skip After Print To Channel 12 D - Space 1 After Print E - Space 2 After Print F - Space 3 After Print

I/O Area Parameter

The I/O area parameter is the label of the control word that precedes the user's I/O area. The control word consists of a word count that specifies the number of computer words of data to be printed. The data must be in EBCDIC format, packed two characters per computer word.

Error Parameter

See Basic Calling Sequence.

CONSOLE PRINTER/INPUT KEYBOARD

There are two ISS for the transfer of data to and from the Console Printer and the Input Keyboard.

TYPE0. The TYPE0 subroutine handles input and output.

WRTY0. The WRTY0 subroutine handles output only. If a program does not require keyboard input, it is advantageous to use the WRTY0 subroutine because it occupies less core storage than the TYPE0 subroutine.

Only the TYPE0 subroutine is described below; the WRTY0 subroutine is identical, except that it does not allow the Read-Print function.

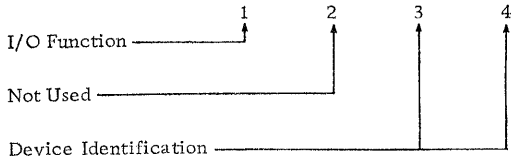
Calling Sequence

```
        LIBF    TYPE0 or WRTY0
        DC      /xxxx  (Control)
        DC      AREA   (I/O area
        .
        .
        .
AREA    +---------------+
        | Word Count    |
        +---------------+
        | I/O Area      |
        +---------------+
```

The parameters used in the above calling sequence are described in the following paragraphs.

Control Parameter

This parameter consists of four hexadecimal digits, as shown below:



I/O Function

The I/O function digit specifies the operation to be performed on the input keyboard and/or console printer. The functions, their associated digital values, and the required parameters are listed and then described below.

Function	Digital Value	Required Parameters*
Test	0	Control
Read-Print	1	Control, I/O Area
Print	2	Control, I/O Area

*Any parameter not required for a particular function must be omitted.

Test. Branches to LIBF+2 if the previous operation has not been completed or to LIBF+3 if the previous operation has been completed.

Read-Print. Reads from the keyboard and prints the requested number of characters on the console printer. The operation sequence is as follows:

1. The calling sequence is analyzed by the Call routine, which then unlocks the keyboard.
2. When a key is pressed, a character interrupt signals the Interrupt Response Routine that a character is ready to be read into core storage.
3. The Interrupt Response Routine converts the keyboard data to console printer output code (see Data Codes). Each character is printed as it is read; the keyboard is then unlocked for entry of the next character.
4. Printer interrupts occur whenever the console printer has completed a print operation. When the interrupt is received, the routine checks to determine if the final character has been read and printed. If so, the operation is considered complete. If the console printer becomes not ready during printing, the subroutines loop internally, waiting for the console printer to become ready.
5. Steps 2 to 4 are repeated until the specified number of characters have been read and printed. The characters read into the I/O area are in IBM card code; that is, each 12-bit image is left-justified in one 16-bit word.

Print. Prints the specified number of characters on the console printer. A printer interrupt occurs when the console printer has completed a print operation. When an interrupt is received, the character count is checked. If the specified number of characters has not been written, printing is initiated for the next character. This sequence continues until the specified number of characters has been printed. Data to be printed must be in console printer code, (see Data Codes) packed two characters per 16-bit word. Control characters can be embedded in the message where desired.

In Read-Print and Print operations, printing begins where the printing element is positioned; that is, carrier return to a new line is not automatic when the subroutine is called.

Device Identification

Device identification digits can be 00 or 01; either value specifies the console printer.

Keyboard Functions

Keyboard functions provide for control by the TYPE0 subroutine and by the operator.

TYPE0 Subroutine Control

Three keyboard functions are recognized by the TYPE0 subroutine.

Backspace. The operator presses the backspace key whenever the previous character is in error. The interrupt response routine senses the control character, backspaces the console printer, and prints a slash (/) through the character in error. In addition, the subroutine prepares to replace the incorrect character in the I/O area with the next character.

If the backspace is depressed twice, the character address is decremented by +2, but only the last graphic character is slashed. For example, if ABCDE was entered and then the backspace key depressed three times, the next graphic character replaces the C but only the E is slashed each time. If XYZ is the new entry, the print-out shows ABCDE/XYZ, but the buffer contains ABXYZ.

Erase Field. When the interrupt response routine recognizes the erase field control character, it assumes that the entire message is in error and is to be entered again. The routine prints two slashes on the console printer, restores the carrier to a new line, and prepares to replace the old message in the I/O area with the new message.

The old message in the I/O area is not cleared. Instead, the new message overlays the old, character by character. If the old message is longer than the new, the remainder of the old message follows the NL character terminating the new message.

End-of-Message. When the interrupt response routine recognizes the end-of-message control character, it assumes the message has been completed, stores an NL character in the I/O area, and terminates the operation.

Operator Request Function

By pressing the operator request key (INT REQ) on the keyboard, the operator can inform the program that he wishes to enter data from the keyboard or the Console Entry switches. The interrupt that results causes the TYPE0 or WRTY0 routine to execute an indirect BSI instruction to core location 44, where the user must have the address of an operator request routine stored. Bit 1 of the accumulator contains the keyboard/console identification bit; that is, the device status word, shifted left two bits.

The user's operator request routine must return to the ISS subroutine via the return link. The user's routine is executed as a part of the interrupt handling. The interrupt level remains ON until control is returned to the ISS subroutine (see General Error Handling Procedures, Post-operation Checks).

I/O Area Parameter

The I/O area parameter is the label of the control word that precedes the user's I/O area. The control word consists of a word count that specifies the number of words to be read or printed. This word count is equal to the number of characters if the Read-Print function is requested, but to one-half the number of characters if the Print function is requested.

PAPER TAPE SUBROUTINES

The paper tape subroutines handle the transfer of data from a Paper Tape Reader to core storage and from core storage to a Paper Tape Punch. Any number of characters can be transferred via one calling sequence.

The PAPTN subroutine must be used if simultaneous reading and punching are desired.

The PAPT1 operates both devices, but only one at a time.

When called, the paper tape subroutine starts the reader or punch and then, as interrupts occur, transfers data to or from the user's I/O area. Input data is packed two characters per computer word by the subroutine; output data must be in that form when the subroutine is called for a punch function.

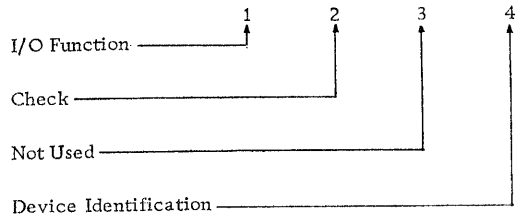
Calling Sequence

```
        LIBF        PAPT1 or PAPTN
        DC          /xxxx (Control)
        DC          AREA (I/O area)
        DC          ERROR (Error Routine)
         .           .
         .           .
ERROR   Return Link
        Error Routine
        BSC I       ERROR
         .           .
         .           .
         .           .
AREA    Word Count
        I/O Area
```

The parameters used in the above calling sequence are described in the following paragraphs.

Control Parameter

This parameter consists of four hexadecimal digits, as shown below:



I/O Function

The I/O function digit specifies the operation to be performed on a Paper Tape Attachment. The functions, their associated digital value, and the required parameters are listed and described below.

Function	Digital Value	Required Parameter*
Test	0	Control
Read	1	Control, I/O area, Error
Punch	2	Control, I/O area, Error

*Any parameter not required for a particular function must be omitted.

Input

Input is four IBM card code hexadecimal digits in INPUT through INPUT +3.

Output

Output is a 16-bit binary word in the accumulator.

Errors Detected

Any input character other than an IBM card code 0 through 9 or A through F is an error.

HOLEB

Description

This subroutine converts IBM card code subset to the EBCDIC subset or converts the EBCDIC subset to IBM card code subset. Code conversion is illustrated in Figure 10.

Calling Sequence

```
          LIBF      HOLEB
          DC        /xxxx    (Control)
          DC        INPUT
          DC        OUTPT
          DC        nnnn     (Character count)
           .          .
           .          .
INPUT     [= = = = = = = = = = =]
OUTPT     [= = = = = = = = = = =]
```

Control Parameter

The control parameter consists of four hexadecimal digits. Digits 1-3 are not used. The fourth digit specifies the direction of conversion:

0 - IBM card code to EBCDIC
1 - EBCDIC to IBM card code

Input

Input is either IBM card code or EBCDIC characters, (as specified by the control parameter) starting in location INPUT. EBCDIC characters must be packed two characters per binary word. IBM card code characters are stored one character to each binary word.

I/O Locations	Conversion Data	Bits in Core Storage 0 ◄——► 15
INPUT	JS	1101 0001 1110 0010
OUTPT	J	0101 0000 0000 0000
OUTPT + 1	S	0010 1000 0000 0000

Figure 10. HOLEB Conversion (EBCDIC to IBM Card Code)

Output

Output is either IBM card code or EBCDIC characters starting in location OUTPT. Characters are packed as described above.

If the direction of the conversion is IBM card code input to EBCDIC output, the input area can overlap the output area if the address INPUT is equal to or greater than the address OUTPT. If the direction of the conversion is EBCDIC input to IBM card code output, the input area can overlap the output area if the address INPUT + n/2 is equal to or greater than the address OUTPT + n, where n is the character count specified. The subroutine starts processing at location INPUT.

Character Count

This number specifies the number of characters to be converted; it is not equal to the number of binary words used for the EBCDIC characters because those characters are packed two per binary word. If an odd count is specified for EBCDIC output, bits 8 through 15 of the last word in the output area are not altered.

Errors Detected

Any input character not asterisked in Appendix D is an error.

SPEED

Description

This subroutine converts IBM card code to EBCDIC or EBCDIC to IBM card code. SPEED accepts all 256 characters defined in Appendix D.

If the input is IBM card code, the conversion time is much faster than that of HOLEB because a different conversion method is used when all 256

EBCDIC characters are accepted. If the SPEED subroutine is called before a card reading operation is completed, the SPEED subroutine synchronizes with a CARD subroutine read operation by checking bit 15 of the word to be processed before converting the word. If bit 15 is a one, the SPEED subroutine waits in a loop until the CARD subroutine sets the bit to a zero.

Calling Sequence

```
        LIBF    SPEED
        DC      /xxxx    (Control)
        DC      INPUT
        DC      OUTPT
        DC      nnnnn    (Character count)
        .       .
        .       .
INPUT   [- - - - - - - - - -]
        .       .
        .       .
OUTPT   [- - - - - - - - - -]
```

Control Parameter

This parameter consists of four hexadecimal digits. Digits 1 and 2 are not used. The third digit indicates whether the EBCDIC code is packed or unpacked.

0 – Packed, two EBCDIC characters per binary word
1 – Unpacked, one EBCDIC character per binary word (left-justified)

The fourth digit indicates the direction of conversion:

0 – IBM card code to EBCDIC
1 – EBCDIC to IBM card code

Input

Input is either IBM card code or EBCDIC characters (as specified by the control parameter) starting in location INPUT. EBCDIC characters can be packed or unpacked. IBM card code characters are stored one character to each binary word.

Output

Output is EBCDIC or IBM card code characters starting in location OUTPT. EBCDIC characters can be packed or unpacked; IBM card code characters are not packed.

The input area should not overlap the output area because of restart problems that can result from card feed errors.

Character Count

This parameter specifies the number of EBCDIC or IBM card code characters to be converted. If the character count is odd and the output code is EBCDIC, bits 8 through 15 of the last word are unaltered.

Errors Detected

Any input character code not listed in Appendix D is an error. All IBM card code punch combinations, except multiple punches in rows 1-7, are legal.

PAPEB

Description

This subroutine converts PTTC/8 subset to EBCDIC subset or EBCDIC subset to PTTC/8 subset. PAPEB conversion of EBCDIC to PTTC/8 with the initialize case option selected is illustrated in Figure 11.

Calling Sequence

```
        LIBF    PAPEB
        DC      /xxxx    (Control)
        DC      INPUT
        DC      OUTPT
        DC      nnnnn    (Character count)
        .       .
        .       .
INPUT   [- - - - - - - - - -]
        .       .
        .       .
OUTPT   [- - - - - - - - - -]
```

I/O Locations	Conversion Data	Bits in Core Storage 0 ◄——► 15			
INPUT	JS	1101	0001	1110	0010
OUTPT +0	UC J	0000	1110	0101	0001
+1	S DEL	0011	0010	0111	1111

Figure 11. PAPEB Conversion (EBCDIC to PTTC/8)

ARITHMETIC AND FUNCTIONAL SUBROUTINES

The IBM 1130 Subroutine Library includes the arithmetic and functional subroutines that are the most frequently required because of their general applicability. There are 44 subroutines, some of which have several entry points.

Table 4 lists the arithmetic and functional subroutines that are included in the Subroutine Library.

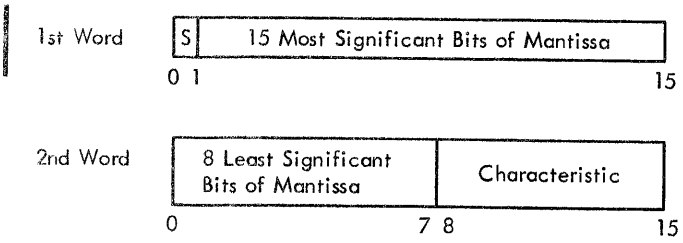
REAL DATA FORMATS

Many of the IBM 1130 arithmetic and functional subroutines offer two ranges of precision: standard and extended. The standard precision provides 23 significant bits, and the extended precision provides up to 31 significant bits.

To achieve correct results from a particular subroutine, the input arguments must be in the proper format.

Standard Precision Format

Standard precision real numbers are stored in core storage as shown below:

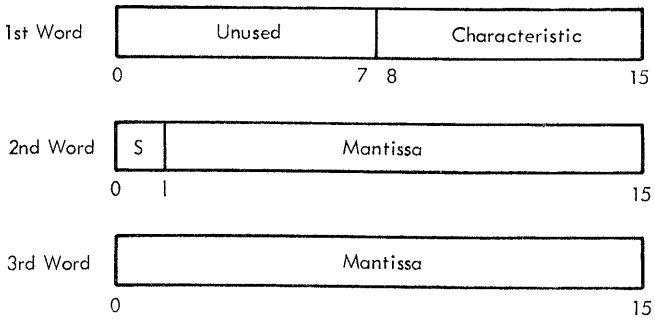


Numbers can consist of up to 23 significant bits (mantissa) with a binary exponent ranging from -128 to +127. Two adjacent storage locations are required for each number. The first (lowest) location must be even-numbered. The sign of the mantissa is in bit zero of the first word. The next 23 bits represent the mantissa (2's complement) and the remaining 8 bits represent the characteristic. The mantissa is normalized to fractional form, i.e., the implied binary point is between bits zero and one.

The characteristic is formed by adding +128 to the exponent. For example, an exponent of -32 is represented by a characteristic of 128-32, or 96. An exponent of +100 is represented by a characteristic of 100 + 128, or 228. Since $128_{10} = 80_{16}$ the characteristic of a nonnegative exponent always has a 1-bit in position 1, while the characteristic of a negative exponent always produces a 0-bit in position 1. A normal zero consists of all zero bits in both the characteristic and the mantissa.

Extended Precision Format

Extended precision real numbers are stored in three adjacent core locations as shown below:



Numbers can consist of up to 31 significant bits with a binary exponent ranging from -128 to +127.

Bits zero through seven of the first word are unused; bits eight through 15 of the first word represent the characteristic of the exponent (formed in the same manner as in the standard range format); bit zero of the second word contains the sign of the mantissa; and the remaining 31 bits represent the mantissa (2's complement).

Real Negative Number Representation

Real negative numbers differ from real positive numbers in only one respect; the mantissa is always the 2s complement of the equivalent positive value.

Example:

+.53125 is represented in core as 44000080
-.53125 is represented in core as BC000080
+4.0 is represented in core as 40000083
-4.0 is represented in core as C0000083

Note that a real negative number is never represented by a value of 800000xx, where xx is any characteristic between 00 and FF. The mantissa value of 800000 is its own 2s complement and therefore lies outside the definition of a real negative number, i.e., the 2s complement of its absolute value.

Fixed Point Format

Fractional numbers, as applied to the fixed-point subroutines, XSQR, XMDS, XMD, and XDD, are defined as binary fractions with implied binary points of zero. That is, the binary point is positioned between the sign (bit 0) and the most significant bit (bit 1).

The user can consider the binary point to be in any position in his fixed-point numbers. To correctly interpret the results the following rules must be observed.

1. Only numbers with binary points in equivalent positions can be correctly added or subtracted.
2. The binary point location in the product of two numbers is the sum of the binary point locations of the multiplier and the multiplicand.
3. The binary point location in the quotient of two numbers is the difference between the binary point locations of the dividend and the divisor.
4. The binary point location in a number that is input to the fixed-point square root subroutine (XSQR) must be an even number from 0-14. The binary point location in the output root is half the binary point location of the input number.

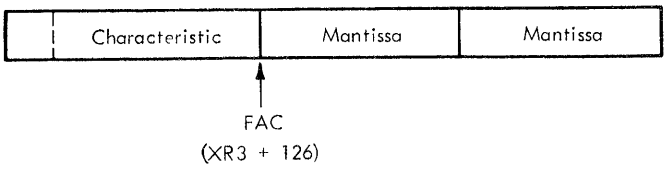
● Table 4. Arithmetic and Functional Subroutines

SUBROUTINE	NAME	
Real	Standard Precision	Extended Precision
Add/Subtract	*FADD/*FSUB	*EADD/*ESUB
Multiply	*FMPY	*EMPY
Divide	*FDIV	*EDIV
Load/Store FAC	*FLD/*FSTO	*ELD/*ESTO
Trigonometric Sine/Cosine	FSINE/FCOSN, FSIN/FCOS	ESINE/ECOSN, ESIN/ECOS :COS
Trigonometric Arctangent	FATN, FATAN	EATN, EATAN
Square Root	FSQR, FSQRT	ESQR, ESQRT
Natural Logarithm	FLN, FALOG	ELN, EALOG
Exponential (e^X)	FXPN, FEXP	EXPN, EEXP
Hyperbolic Tangent	FTNH/FTANH	ETNH/ETANH
Real Base to an Integer Exponent	*FAXI	*EAXI
Real Base to a Real Exponent	*FAXB	*EAXB
Real to Integer	IFIX	IFIX
Integer to Real	FLOAT	FLOAT
Normalize	NORM	NORM
Real Binary to Decimal/Real Decimal to Binary	FBTD/FDTB	FBTD/FDTB
Real Arithmetic Range Check	FARC	FARC
Fixed-Point		
Integer Base to an Integer Exponent	*FIXI	*FIXI
Fixed-Point Square Root	XSQR	XSQR
Fixed-Point Fractional Multiply (short)	XMDS	
Fixed-Point Double Word Multiply	XMD	XMD
Fixed-Point Double Word Divide	XDD	XDD
Special Function		
Real Reverse Subtract	*FSBR	*ESBR
Real Reverse Divide	*FDVR	*EDVR
Real Reverse Sign	SNR	SNR
Real Absolute Value	FAVL, FABS	EAVL, EABS
Integer Absolute Value	IABS	IABS
Miscellaneous		
Get Parameters	FGETP	EGETP

NOTE: By adding an X to those names prefixed with an asterisk, the user can cause the contents of Index Register 1 to be added to the address of the argument specified in the subroutine calling sequence to form the effective argument address. For example, FADDX would be the modified form of FADD.

REAL NUMBER PSEUDO ACCUMULATOR

IBM 1130 real number subroutines sometimes require a register or accumulator that can accommodate numbers in real number format. Since all of the 1130 registers are only 16 bits in length, a pseudo accumulator must be set up to contain two- or three-word real numbers. The pseudo accumulator (designated FAC for floating accumulator) is a three-word register occupying the three highest locations of the transfer vector (see IBM 1130 Assembler Language, C26-5927). The user can refer to these words by using Index Register 3 plus a fixed displacement (XR3 + 125, 126, or 127). The format of the FAC is shown below.



The effective address of the mantissa is always even.

NOTE: Arithmetic and functional subroutines do not save and restore the contents of the 1130 accumulator or the accumulator extension. The main program should provide for this if the contents are significant.

CALLING SEQUENCES

The arithmetic and functional subroutines are called via a CALL or LIBF statement (whichever is required) followed, in some cases, by a DC statement containing the actual or symbolic address of an argument. In the descriptions that follow, the notations (ARG) and (FAC) refer to the contents of the operand rather than its address. The name FAC refers to the real number pseudo accumulator. The extended precision subroutine names are prefixed with the letter E (subroutines which handle both precisions have the same name and do not have a prefix).

Note also that some of the functional subroutines can be called via two different calling sequences. One calling sequence assumes the argument is in FAC; the other specifies the location of the argument with a DC statement.

In addition, some subroutines can have indexed linkage to the argument. The calling sequence is the same except for the subroutine name which contains an X suffix. Also, some subroutines perform more than one type of arithmetic or function. For example, FSIN and FCOS are different entry points to the same subroutine. Each subroutine is listed in Table 4 with the corresponding entry points.

Real Add

LIBF	FADD, FADDX, EADD or EADDX
DC	ARG
Input	Real augend in FAC Real addend in location ARG
Result	(FAC) + (ARG) replaces (FAC)

Real Subtract

LIBF	FSUB, FSUBX, ESUB or ESUBX
DC	ARG
Input	Real minuend in FAC Real subtrahend in location ARG
Result	(FAC) - (ARG) replaces (FAC)

Real Multiply

LIBF	FMPY or EMPY
DC	ARG
Input	Real multiplicand in FAC Real multiplier in location ARG
Result	(FAC) times (ARG) replaces (FAC)

Real Divide

LIBF	FDIV, FDIVX, EDIV or EDIVX
DC	ARG
Input	Real dividend in FAC Real divisor in location ARG
Result	(FAC) / (ARG) replaces (FAC)

Load FAC

LIBF	FLD, FLDX, ELD or ELDX
DC	ARG
Input	Real number in location ARG
Result	(ARG) replaces (FAC)

Store FAC

LIBF	FSTO, FSTOX, ESTO or ESTOX
DC	ARG
Input	Real number in FAC
Result	(FAC) replaces (ARG)

Real Trigonometric Sine

CALL	FSINE or ESINE
Input	Real argument (in radians) in FAC
Result	Sine of (FAC) replaces (FAC)

or

CALL	FSIN or ESIN
DC	ARG
Input	Real argument (in radians) in location ARG
Result	Sine of (ARG) replaces (FAC)

Real Trigonometric Cosine

CALL	FCOSN or ECOSN
Input	Real argument (in radians) in FAC
Result	Cosine of (FAC) replaces (FAC)

or

CALL	FCOS or ECOS
DC	ARG
Input	Real argument (in radians) in location ARG
Result	Cosine of (ARG) replaces (FAC)

Real Trigonometric Arctangent

CALL	FATN or EATN
Input	Real argument in location ARG
Result	Arctangent of (FAC) replaces (FAC); the result lies within the range $\pm\frac{\pi}{2}$ radians (±90 degrees)

or

CALL	FATAN or EATAN
DC	ARG
Input	Real argument in location ARG
Result	Arctangent of (ARG) replaces (FAC); the result lies within the range $\pm\frac{\pi}{2}$ radians (±90 degrees)

Real Square Root

CALL	FSQR or ESQR
Input	Real argument in FAC
Result	Square root of (FAC) replaces (FAC)

or

CALL	FSQRT or ESQRT
DC	ARG
Input	Real argument in location ARG
Result	Square root of (ARG) replaces (FAC)

Real Natural Logarithm

CALL	FLN or ELN
Input	Real argument in FAC
Result	Log_e (FAC) replaces (FAC)

or

CALL	FALOG or EALOG
DC	ARG
Input	Real argument in location ARG
Result	Log_e (ARG) replaces (FAC)

Real Exponential

CALL	FXPN or EXPN
Input	Real argument in FAC = n
Result	e^n replaces (FAC)

or

CALL	FEXP or EEXP
DC	ARG
Input	Real argument in location ARG = n
Result	e^n replaces (FAC)

Real Hyperbolic Tangent

CALL	FTNH or ETNH
Input	Real argument in FAC
Result	TANH (FAC) replaces (FAC)

or

CALL	FTANH or ETANH
DC	ARG
Input	Real argument in location ARG
Result	TANH (ARG) replaces (FAC)

Real Base to an Integer Exponent

LIBF	FAXI, FAXIX, EAXI, or EAXIX
DC	ARG
Input	Real base in FAC Integer exponent in location ARG
Result	(FAC), raised to the exponent (ARG), replaces (FAC)

Real Base to a Real Exponent

CALL	FAXB, FAXBX, EAXB or EAXBX
DC	ARG
Input	Real base in FAC Real exponent in location ARG
Result	(FAC) raised to the exponent (ARG) replaces (FAC)

Real to Integer

LIBF	IFIX
Input	Real number in FAC
Result	Integer in the Accumulator

Integer to Real

LIBF	FLOAT
Input	Integer in the Accumulator
Result	Real number in FAC

Normalize

LIBF	NORM
Input	Real unnormalized number in FAC
Result	The mantissa portion of FAC is shifted until the most significant bit resides in bit position 1. The characteristic is changed to reflect the number of bit positions shifted.

Real to Decimal

CALL	FBTD
DC	LDEC
Input	Real number in FAC
Output	A string of EBCDIC-coded decimal data, starting at location LDEC. Each EBCDIC character occupies the rightmost eight bits of a word. The output format is exactly as follows. sd. ddddddddEsdd where s represents a sign (plus or minus) and d represents one of the decimal digits 0-9.

Real Decimal to Binary

CALL	FDTB
DC	LDEC
Input	Same as output from FBTD subroutine. The input field may not contain any embedded blanks. The first blank encountered is interpreted as the end of the string.
Output	Real number in FAC

Real Arithmetic Range Check

LIBF	FARC
Result	This subroutine checks for real number overflow or underflow, and sets programmed indicators for interrogation by a FORTRAN program.

Integer Base to an Integer Exponent

LIBF	FIXI or FIXIX
DC	ARG
Input	Integer base in the accumulator Integer exponent in location ARG
Result	(Accumulator) raised to the exponent contained in ARG replaces (accumulator)

Fixed-Point Square Root

CALL	XSQR
Input	Fixed-point fractional argument (16 bits only) in the accumulator.
Result	Square root of (accumulator) replaces (accumulator). If the argument is negative the absolute value is used and the overflow indicator is turned on.

Fixed-Point Double-Word Multiply

LIBF	XMD
Input	Double-word fractional multiplier in FAC (addressed by XR3 + 126) Double-word fractional multiplicand in the accumulator and extension
Result	Double-word fractional product in the accumulator and extension

Fixed-Point Fractional Multiply

LIBF	XMDS
Input	Double-word fractional multiplier in the accumulator and extension Double-word fractional multiplicand in FAC (addressed by XR3 + 126)
Result	Product in the accumulator and extension (XMDS is shorter and faster than XMD; however, the resulting precision is 24 bits).

Fixed-Point Double-Word Divide

LIBF	XDD
Input	Double-word fractional dividend in FAC (addressed by XR3 + 126) Double-word fractional divisor in accumulator and extension
Result	Double-word fractional quotient in the accumulator and extension. The double dividend in FAC is destroyed by the execution of the subroutine.

Real Reverse Subtract

LIBF	FSBR, FSBRX, ESBR or ESBRX
DC	ARG
Input	Real minuend in location ARG Real subtrahend in FAC
Result	(ARG) - (FAC) replaces (FAC)

Real Reverse Divide

LIBF	FDVR, FDVRX, EDVR or EDVRX
DC	ARG
Input	Real dividend in location ARG Real divisor in FAC
Result	(ARG) / (FAC) replaces (FAC)

Real Reverse Sign

LIBF	SNR
Input	Real number in FAC
Result	-(FAC) replaces (FAC)

Real Absolute Value

CALL	FAVL or EAVL
Input	Real number in FAC
Result	Absolute value of (FAC) replaces (FAC)

or

CALL	FABS or EABS
DC	ARG
Input	Real number in location ARG
Result	Absolute value of (ARG) replaces (FAC)

Integer Absolute Value

CALL	IABS
DC	ARG
Input	An integer in ARG
Result	Absolute value of (ARG) replaces (accumulator)

Get Parameters (FGETP or EGETP)

Example:

```
MAIN    CALL    SUBR
        DC      ARG
NEXT    etc.
  .       .       .
  .       .       .
  .       .       .
SUBR    DC        0
        LIBF    FGETP or EGETP
SUBEX   DC        0
        etc.
  .       .       .
  .       .       .
  .       .       .
  .       .       .
        BSC I   SUBEX
```

The FGETP subroutine performs two functions for a subroutine accessed by a CALL statement. It loads FAC with the contents of ARG; it sets SUBEX to return to NEXT in the main program.

ARITHMETIC AND FUNCTIONAL SUBROUTINE ERROR INDICATORS

The highest three-word entry in the transfer vector is reserved for the real number pseudo accumulator (FAC). The next to highest three-word entry is reserved for the arithmetic and functional subroutine error indicators.

The first word (addressed XR3 +122) of the second entry is used for real number arithmetic overflow and underflow indicators. The second word (XR3 + 123) is used for a divide check indicator, and the third word (XR3 + 124) is used for functional subroutine indicators. The loader initializes all three words to zero.

Word One

Each real number subroutine checks for exponent underflow and overflow. If either occurs, word one and FAC are set as follows.

1, if overflow has occurred (FAC = ± maximum).
3, if underflow has occurred (FAC = zero).

When an overflow occurs, FAC is set to the largest valid number of the same algebraic sign as the contents of FAC when the overflow was detected. The last error condition replaces any previous indication.

Also, when an underflow occurs, FAC is set to zero.

Word Two

The real number divide subroutines check for division by zero. If this occurs, word two is set to 1. The dividend is not changed.

Word Three

The functional subroutines check for the following error conditions and set word three as described. All error conditions detected by the functional subroutines are indicated in word three.

Real Natural Logarithm

When the argument is zero, FAC is set to the largest negative value and a bit is ORed into position 15 of word three. When the argument is negative, the absolute value of the argument is used and a bit is ORed into position 15 of word three.

Real Trigonometric Sine and Cosine

When the absolute value of the argument is equal to or greater than 2^{24}, FAC is set to zero and a bit is ORed into position 14 of word three.

Real Square Root

When the argument is negative, the square root of the argument's absolute value is returned, and a bit is ORed into position 13 of word three.

Real to Integer

When the absolute value of the argument is greater than $2^{15}-1$, the largest possible signed result is placed in the accumulator and a bit is ORed into position 12 of word three.

Integer Base to an Integer Exponent

When the base is zero and the exponent is zero or negative, a zero result is returned and a bit is ORed into position 11 of word three.

Real Base to an Integer Exponent

When the base is zero and the exponent is zero or negative, a zero result is returned and a bit is ORed into position 10 of word three.

Real Base Raised to a Real Exponent

When the base is zero and the exponent is zero or negative, a zero result is returned and a bit is ORed into position 9 of word three. When the base is negative and the exponent is not zero, the absolute value of the base is used and a bit is ORed into position 15 of word three.

FUNCTIONAL SUBROUTINE ACCURACY

Given:

$e \equiv$ Maximum error
$f(x) \equiv$ True value of the function
$f^*(x) \equiv$ Value generated by subroutine
$(<+\infty) \equiv \leq$ Largest valid real number
$(>-\infty) \equiv \geq$ Most negative real number

EXTENDED PRECISION SUBROUTINES

The following statements of accuracy apply to extended precision subroutines.

ESIN

$$e \equiv \left| \frac{\sin(x) - \sin^*(x)}{x} \right| < 3.0 \times 10^{-9}$$

for the range

$$-1.0 \times 10^{6} \leq x < 0$$

$$1.0 \times 10^{6} \geq x > 0$$

for $x = 0$ $\sin(x) \equiv 0$

ECOS

$$e \equiv \left| \frac{\cos(x) - \cos^*(x)}{|x| + \frac{\pi}{2}} \right| < 3.0 \times 10^{-9}$$

for the range

$$-1.0 \times 10^{6} \leq x \leq 1.0 \times 10^{6}$$

EATAN

$$e \equiv \left| \frac{\text{atn}(x) - \text{atn}^*(x)}{\text{atn}(x)} \right| < 2.0 \times 10^{-9}$$

for the range

$$-3.88336148 \times 10^{37} \leq x \leq 3.88336148 \times 10^{37}$$

EEXP

$$e \equiv \left| \frac{e^{x} - (e^{x})^*}{e^{x}} \right| < \left\{ \begin{matrix} 2.0 \times 10^{-9} \\ \text{or} \\ 2.0 \times 10^{-9} \end{matrix} \right. \begin{matrix} |x| \\ \end{matrix} \left. \begin{matrix} \\ \\ \\ \end{matrix} \right\} \begin{matrix} \text{whichever} \\ \text{is} \\ \text{greater} \end{matrix}$$

for the range

$$-\ln(\infty) < x < \ln(\infty)$$

i.e., $0 < e^{x} < \infty$

ELN

$$e \equiv \left| \frac{\ln(x) - \ln^*(x)}{\ln(x)} \right| < 3.0 \times 10^{-9}$$

for the range

$$0 < x < \infty$$

ETANH

$$e \equiv \left| \tanh(x) - \tanh^*(x) \right| < 3.0 \times 10^{-9}$$

for the range

$$-\infty < x < \infty$$

ESQRT

$$e \equiv \left| \frac{\sqrt{x} - \sqrt{x}^*}{\sqrt{x}} \right| < 1.0 \times 10^{-9}$$

for the range

$$0 < x < \infty$$

STANDARD PRECISION SUBROUTINES

The following statements of accuracy apply to the standard precision subroutines.

FSIN

$$e \equiv \left| \frac{\sin(x) - \sin^*(x)}{x} \right| < 2.5 \times 10^{-7}$$

for the range

$$-1.0 \times 10^{6} \leq x < 0$$

$$1.0 \times 10^{6} \geq x > 0$$

for $x = 0$ $\sin(x) \equiv 0$

FCOS

$$e \equiv \left| \frac{\cos(x) - \cos^*(x)}{|x| + \frac{\pi}{2}} \right| < 2.5 \times 10^{-7}$$

for the range

$$-1.0 \times 10^{6} \leq x \leq 1.0 \times 10^{6}$$

FATAN

$$e \equiv \left| \frac{atn(x) - atn^*(x)}{atn(x)} \right| < 5.0 \times 10^{-7}$$

for the range

$$-3.883361 \times 10^{37} \leq x \leq 3.883361 \times 10^{37}$$

FEXP

$$e \equiv \left| \frac{e^x - (e^x)^*}{e^x} \right| < \left\{ \begin{matrix} 2.5 \times 10^{-7} \, |x| \\ \text{or} \\ 2.5 \times 10^{-7} \end{matrix} \right\} \begin{matrix} \text{whichever} \\ \text{is} \\ \text{greater} \end{matrix}$$

for the range

$$-\ln(\infty) < x < \ln(\infty) \text{ i.e., } 0 < e^x < \infty$$

FLN

$$e \equiv \left| \frac{\ln(x) - \ln^*(x)}{\ln(x)} \right| < 4.0 \times 10^{-7}$$

for the range

$$0 < x < \infty$$

FTANH

$$e \equiv \left| \tanh(x) - \tanh^*(x) \right| < 2.5 \times 10^{-7}$$

for the range

$$-\infty < x < +\infty$$

FSQRT

$$e \equiv \left| \frac{\sqrt{x} - \sqrt{x}^*}{\sqrt{x}} \right| < 2.5 \times 10^{-7}$$

for the range

$$0 < x < \infty$$

ELEMENTARY FUNCTION ALGORITHMS

The choice of an approximating algorithm for a given function depends on such considerations as expected execution time, storage requirements, and accuracy. For a given accuracy, and within reasonable limits, storage requirements vary inversely as the execution time. Polynomial approximating is used to evaluate the elementary functions to effect the desired balance between storage requirements and efficiency.

SINE-COSINE

Polynomial Approximation

Given a real number, x, n and y are defined such that

$$x\left(\frac{\pi}{2}\right) = n + y$$

where n is an integer and $0 \leq y < 1$. Thus, $x = 2\pi n + 2\pi y$, and the identities are

$$\sin x = \sin 2\pi y \text{ and } \cos x = 2\pi y .$$

The polynomial approximation, F(z), for the function $(\sin 2\pi z)/z$ is used where $-1/4 \leq z \leq 1/4$.

The properties of sines and cosines are used to compute these functions as follows.

$$\cos 2\pi y = F(z)$$

where:

$z = 1/4-y$ in the range $0 \leq y \leq 1/2$
$z = y-3/4$ in the range $1/2 \leq y < 1$

$$\sin 2\pi y = F(z)$$

where

$z = y$ in the range $0 \leq y < 1/4$
$z = 1/2-y$ in the range $1/4 \leq y < 3/4$
$z = y-1$ in the range $3/4 \leq y < 1$

Extended Precision

$$F(z) = a_1 z + a_2 z^3 + a_3 z^5 + a_4 z^7 + a_5 z^9 + a_6 z^{11}$$

where

$a_1 = 6.2831853071$
$a_2 = -41.341702117$
$a_3 = 81.605226206$
$a_4 = -76.704281321$
$a_5 = 42.009805726$
$a_6 = -14.394135365$

Standard Precision

$$F(z) = a_1 z + a_2 z^3 + a_3 z^5 + a_4 z^7 + a_5 z^9$$

where

$a_1 = 6.2831853$
$a_2 = -41.341681$
$a_3 = 81.602481$
$a_4 = -76.581285$
$a_5 = 39.760722$

ARCTANGENT

Polynomial Approximation

The routine for arctangent is built around a polynomial, $F(z)$, that approximates $\text{Arctan}(z)$ in the range $-.23 \leq z \leq .23$. The $\text{Arctan}(z)$ for z outside this range is found by using the identities:

$$\text{Arctan}(-z) = -\text{Arctan}(z)$$

$$\text{Arctan}(z) = a_k + \text{Arctan}\left[\frac{z - b_k}{zb_k + 1}\right]$$

where

$$a_k = \frac{k\pi}{7} \quad \text{and} \quad b_k = \tan a_k$$

and k is determined so that

$$\tan\frac{(2k-1)\pi}{14} \leq z < \tan\frac{(2k+1)\pi}{14} \qquad k = 1, 2, 3$$

Having determined the value of k appropriate to z, the transformation $x = (z-b_k)(zb_k + 1)$ puts x in the range $-\tan \pi/14 \leq x < \tan \pi/14$. The polynomial $F(z)$ was chosen to be good over a range slightly larger (i.e., $.23 > \tan \pi/14$) so that the comparisons to determine the interval in which z lies need be only standard precision accuracy.

Extended Precision

$$F(z) = x\,(1.0 - a_1 x^2 + a_2 x^4 - a_3 x^6 + a_4 x^8)$$

where

$a_1 = .33333327142$
$a_2 = .19999056792$
$a_3 = .14235177463$
$a_4 = .09992331248$

Standard Precision

$$F(z) = x(1.0 - a_1x^2 + a_2x^4 = a_3x^6)$$

where

$a_1 = .333329573$
$a_2 = .199641035$
$a_3 = .13177988$

SQUARE ROOT

Square Root (x)
Let $x = 2^{2b}F$ when $.25 \leq F < 1$
then $\sqrt{X} = 2^b\sqrt{F}$
where $\sqrt{F} = P_i$ i = number of approximation

$P_1 = AF + B$ as a first approximation followed by 2 Newton iterations

where

$A = .875$, $B = .27863$ when $.25 \leq F < .5$

or

$A = .578125$, $B = .421875$ when $.5 \leq F < 1$

$$P_2 = \frac{\left(P_1 + \frac{F}{P_1}\right)}{2}$$

$$P_3 = \frac{\left(P_2 + \frac{F}{P_2}\right)}{2}$$

NATURAL LOGARITHM

Polynomial Approximation

Given a normalized real number

$$x = 2^k x\ f$$

where the range of f is $1/2 \leq f < 1$, and j and g are found such that $x = 2^j g$ where $(\sqrt{2}/2 \leq g < \sqrt{2})$. This is done by setting $j = k-1$, $g = 2f$ if $f < \sqrt{2}/2$ and $j = k$, $g = f$ otherwise. Thus:

$$\ln(x) = j.\ln(2) + \ln(g).$$

The approximation for $\ln(g)$, $\sqrt{2}/2 \leq g < \sqrt{2}$, is based on the series

$$\ln\frac{v+x}{v-x} = 2\left[(x/v) + (x^3/3v^3) + (x^5/5v^5) + \ldots\right]$$

which converges for $(-v < x < v)$.
With the transformation

$$x = v\frac{f-1}{f+1}, \quad v = (\sqrt{2}+1)^2$$

so that $-1 \leq x < 1$ for $\sqrt{2}/2 \leq g < \sqrt{2}$.
Substituting,

$$\ln(g) = 2\ (z + z^3/3 + z^5/5 + \ldots)$$

where $z = x/v = (f-1)/(f+1)$. The approximation used is $G(z)$ for $\ln(g)/z$ in the range $\sqrt{2}/2 \leq g < \sqrt{2}$.

Then for both entended and standard precision,

$$z = \frac{g-1}{g+1}$$
$\sqrt{2}/2 = .7071067811865$
$\ln(2) = .6931471805599$

Thus, the required calculation is

$$\ln(x) = j\ .\ln(2) + zG(z)$$

Extended Precision

$$G(z) = b_0 + b_2z^2 + b_4z^4 + b_6z^6$$

where

$b_0 = 2.0$
$b_2 = .666666564181$
$b_4 = .400018840613$
$b_6 = .28453572660$
$b_8 = .125$

SELECTIVE DUMP SUBROUTINES

The IBM 1130 Subroutine Library includes three dump subroutines: Dump Selected Data on the console printer, Dump Selected Data on the 1132 Printer, and Dump Status Area. These subroutines allow the user to dump selected portions of core storage during the execution of an object program.

DUMP SELECTED DATA ON CONSOLE PRINTER OR 1132 PRINTER

Two subroutines are available to select an area of core storage and dump it out on the console printer or the 1132 Printer. Each of these subroutines has two entry points, one for hexadecimal output and one for decimal output. The entry points for the various configurations are shown below:

DMTX0 Dump on console printer in hexadecimal form, using the WRTY0 subroutine

DMTD0 Dump on console printer in decimal form, using the WRTY0 subroutine

DMPX1 Dump on 1132 Printer in hexadecimal form, using the PRNT1 subroutine

DMPD1 Dump on 1132 Printer in decimal form, using the PRNT1 subroutine

Calling Sequence

The calling sequence for any of the above functions is as follows:

```
CALL    ENTRY POINT
DC      START
DC      END
```

START and END represent the starting and ending addresses of the portion of core storage to be dumped. A starting address greater than the ending address results in the error message, ERROR IN ADDRESS, and a return to the main program.

Format

Before the actual dump appears on the selected output device, the user is given one line of status information. This line indicates the status of the Overflow and Carry triggers (ON or OFF), the contents of the Accumulator and Extension, and the contents of the three index registers. The index register contents are given in both hexadecimal and decimal form, regardless of which type of output was requested. The format of the status information is shown below:

OFF	ON	HHHHH (±DDDDD)	HHHH (±DDDDD)
Overflow	Carry	Accumulator	Extension
HHHHH (±DDDDD)		HHHH (±DDDDD)	HHHH (±DDDDD)
Index Register 1		Index Register 2	Index Register 3

All other data is dumped eight words to a line; the address of the first word in each line is printed to the left of the line. Hexadecimal data is printed four characters per word; decimal data is printed five digits per word, preceded by a plus or minus sign.

Page numbers are not printed for either subroutine. However, the 1132 Printer subroutine does provide for automatic page overflow upon the sensing of a channel 12 punch in the carriage tape.

DUMP STATUS AREA

This subroutine provides a relatively easy and efficient means of dumping the first 80 words of core storage. These words contain status information relating to index registers, interrupt addresses, etc., which may be required frequently during the testing of a program. It may also be desirable to dump these words before loading because pressing the Load key destroys the data in the first 80 words of core storage.

The Dump Status Area subroutine is called via the following statement:

```
CALL    DMP80
```

The console printer prints the first 80 words of core storage in hexadecimal form; the printing format provides spacing between words. After typing the last word, the subroutine returns control to the main program.

INTERRUPT SERVICE SUBROUTINES

The following rules must be adhered to when writing an ISS:

1. Precede the ISS statement with an LIBR statement if the subroutine is to be called by LIBF rather than CALL.
2. Precede the subroutine with an EPR (extended) or an SPR (standard) statement if precision specification is necessary.
3. Precede the subroutine with one ISS statement defining the entry point (one only), the ISS number, and the ILS subroutines required. The device interrupt level assignments, and the ISS numbers used in the IBM-provided ISS and ILS routines, are shown in Table 5.
4. The entry points of an ISS are defined by the related ILS. This must be taken into consideration when a user-written ISS is used with an IBM supplied ILS. The ILS executes a Branch and Store I instruction to the ISS at the ISS entry point plus n (see Table 5). The ISS must return to the ILS via a BSC instruction (not a BOSC).

INTERRUPT LEVEL SUBROUTINES

An ILS is loaded only if requested by a loaded ISS. The following are rules for writing an ILS:

1. Precede the subroutine with an ILS statement identifying the interrupt level involved. In the Disk Monitor System, the subroutine must also be preceded by the correct number of LEVEL control records (see "Assembler," IBM 1130 Disk Monitor System Reference Manual, Form C26-3750).
2. Precede all instructions by an ISS branch table and include one word per ILSW bit used. If the ILSW will not be scanned, (i.e., a single ISS routine to handle all interrupts on the level), then a one word table is sufficient. The minimum table size is one word. Table words must be non-zero.

Table 5. ISS/ILS Correspondence

ISS Number	Device	Device Interrupt Level Assignments	n
1	1442 Card Reader Punch	0, 4	+4, +7
2	Input Keyboard/ Console Printer	4	+4
3	1134/1055 Paper Tape Reader/Punch	4	+4
4	Disk Storage	2	+4
6	1132 Printer	1	+4
7	1627 Plotter	3	+4

ILSW Bit 15 word
ILSW Bit 14 word
.
.
.
ILSW Bit 0 word
} ISS Branch Table

The ISS Branch table identifies both the ISS subroutine and the point within the ISS which should be entered for each bit used in the ILSW. The actual linkage is generated by the relocating loader or core image converter. Basic to this generation is the ISS number implied by bits 8-15 of the branch table word and specified in the ISS statement. This number identifies a core location in which the loader or converter has stored the address of the called entry point in the ISS. This entry point address is incremented by the value in bits 0-7 of the branch table word, producing the branch linkage. The loader or converter replaces the ISS branch table word with the generated branch linkage.

At execution time the ISS branch table contains actual addresses. It may be used with an indirect branch and store I (BSI) instruction to reach the ISS corresponding to that ILSW bit position. The ILSW bit that is ON can be determined by the execution of a SLCA instruction. At the completion of this instruction, the index register specified contains a relative value

Standard Precision

$$G(z) = b_0 + b_2 z^2 + b_4 z^4 + b_6 z^6$$

where

$b_0 = 2.0$
$b_2 = .66664413786$
$b_4 = .4019234697$
$b_6 = .25$

EXPONENTIAL

Polynomial Approximation

To find e^x, the following identity is used.

To reduce the range, we let

$$x \log_2 e = n + d + z$$

where

- n is the integral portion of the real number,
- d is a discreet fraction (1/8, 3/8, 5/8, or 7/8) of the real number, and
- z is the remainder which is in the range $-1/8 \leq z \leq 1/8$.

Thus,

$$e^x = 2^n \times 2^d \times 2^z$$

and it is necessary to only approximate 2^z for $-1/8 \leq z \leq 1/8$ by using the polynomial $F(z)$.

Extended Precision

$$F(z) = a_0 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4 + a_5 z^5$$

where

$a_0 = 1.0$
$a_1 = .69314718057$
$a_2 = .24022648580$
$a_3 = .055504105406$
$a_4 = .0096217398747$
$a_5 = .0013337729375$

Standard Precision

$$F(z) = a_0 + a_1 z + a_2 z^2 + a_3 z^3 + a_4 z^4$$

where:

$a_0 = 1.0$
$a_1 = .693147079$
$a_2 = .240226486$
$a_3 = .0555301557$
$a_4 = .00962173985$

HYPERBOLIC TANGENT

$$\text{Tanh}(x) = \frac{e^{2x}-1}{e^{2x}+1}$$

for

$x \geq 32$ Tanh (x) = 1
$x \leq -32$ Tanh (x) = -1

REAL BASE TO REAL EXPONENT

$$A = e^{\ln A}$$

therefore:

$$A^B = \left(e^{\ln A}\right)^B = e^{B \ln A}$$

Subroutine	Names	Other Subroutines Required
Called by LIBF (monitor)		
Printer-Keyboard Input/Output	TYPEZ	GETAD, EBCTB, HOLEZ
1132 Printer Output	PRNTZ	None
Paper Tape Input/Output	PAPTZ	None
Card Code-EBCDIC Conversion	HOLEZ	GETAD, EBCTB, HOLTB
Console Printer Code Table	EBCTB	None
Card-Keyboard Code Table	HOLTB	None
Address Calculation	GETAD	None
ARITHMETIC AND FUNCTIONAL		
Called by CALL		
Real Hyperbolic Tangent (E)	ETNH, ETANH	EEXP, ELD/ESTO, EADD, EDIV, EGETP
Real Hyperbolic Tangent (S)	FTNH, FTANH	FEXP, FLD/FSTO, FADD, FDIV, FGETP
Real Base to Real Exponent (E)	EAXB, EAXBX	EEXP, ELN, EMPY
Real Base to Real Exponent (S)	FAXB, FAXBX	FEXP, FLN, FMPY
Real Natural Logarithm (E)	ELN, EALOG	XMD, EADD, EMPY, EDIV, NORM, EGETP
Real Natural Logarithm (S)	FLN, FALOG	FSTO,XMDS,FADD,FMPY,FDIV,NORM,FGETP
Real Exponential (E)	EXPN, EEXP	XMD, FARC, EGETP
Real Exponential (S)	FXPN, FEXP	XMDS, FARC, FGETP
Real Square Root (E)	ESQR, ESQRT	ELD, ESTO, EADD, EMPY, EDIV, EGETP
Real Square Root (S)	FSQR, FSQRT	FLD/FSTO, FADD, FMPY, FDIV, FGETP
Real Trigonometric Sine/Cosine (E)	ESIN, ESINE, ECOS, ECOSN	EADD, EMPY, NORM, XMD, EGETP
Real Trigonometric Sine/Cosine (S)	FSIN, FSINE, FCOS, FCOSN	FADD, FMPY, NORM, XMDS, FSTO, FGETP
Real Trigonometric Arctangent (E)	EATN, EATAN	EADD, EMPY, EDIV, XMD, EGETP, NORM
Real Trigonometric Arctangent (S)	FATN, FATAN	FADD, FMPY, FDIV, XMDS, FSTO, FGETP
Fixed-Point Square Root	XSQR	None
Real Absolute Value (E)	EAVL, EABS	EGETP
Real Absolute Value (S)	FAVL, FABS	FGETP
Integer Absolute Value	IABS	None
Real Binary to Decimal/Real Decimal to Binary	FBTD, FDTB	None
Called by LIBF		
Get Parameters (E)	EGETP	ELD
Get Parameters (S)	FGETP	FLD
Real Base to Integer Exponent (E)	EAXI, EAXIX	ELD/ESTO, EMPY, EDVR
Real Base to Integer Exponent (S)	FAXI, FAXIX	FLD/FSTO, FMPY, FDVR
Real Reverse Divide (E)	EDVR, EDVRX	ELD/ESTO, EDIV
Real Reverse Divide (S)	FDVR, FDVRX	FLD/FSTO, FDIV
Real Divide (E)	EDIV, EDIVX	XDD, FARC
Real Divide (S)	FDIV, FDIVX	FARC
Real Multiply (E)	EMPY, EMPYX	XMD, FARC
Real Multiply (S)	FMPY, FMPYX	XMDS, FARC
Real Reverse Subtract (E)	ESBR, ESBRX	EADD
Real Reverse Subtract (S)	FSBR, FSBRX	FADD
Real Add/Subtract (E)	EADD, EADDX, ESUB, ESUBX	FARC, NORM
Real Add/Subtract (S)	FADD, FADDX, FSUB, FSUBX	NORM, FARC
Load/Store FAC (E)	ELD, ELDX, ESTO, ESTOX	None
Load/Store FAC (S)	FLD, FLDX, FSTO, FSTOX	None
Fixed Point Double Word Divide	XDD	XMD
Fixed Point Double Word Multiply	XMD	None
Fixed Point Fractional Multiply (short)	XMDS	None
Real Reverse Sign	SNR	None
Integer to Real	FLOAT	NORM
Real to Integer	IFIX	None
Fixed Integer Base to an Integer Exponent	FIXI, FIXIX	None
Normalize	NORM	None
Real Arithmetic Range Check	FARC	None
DUMP		
Called by CALL		
Dump Status Area	DMP80	None
Selective Dump on Console Printer	DMTX0, DMTD0	WRTY0
Selective Dump on Printer	DMPX1, DMPD1	PRNT1

Subroutine	Names	Other Subroutines Required
INTERRUPT LEVEL		
Level 0	ILS00*	None
Level 1	ILS01*	None
Level 2	ILS02*	None
Level 3	ILS03*	None
Level 4	ILS04*	None
*These subroutines are not identified by name in the card and paper tape systems		
CONVERSION		
Called by LIBF		
Binary to Decimal	BINDC	None
Binary to Hexodecimal	BINHX	None
Decimal to Binary	DCBIN	None
EBCDIC to Console Printer Code	EBPRT	EBPA, PRTY
IBM Card Code to or From EBCDIC	HOLEB	EBPA, HOLL
IBM Card Code to Console Printer Code	HOLPR	HOLL, PRTY
Hexadecimal to Binary	HXBIN	None
EBCDIC to or from PTTC/8	PAPEB	EBPA
IBM Card Code to or from PTTC/8	PAPHL	EBPA, HOLL
PTTC/8 to Console Printer Code	PAPPR	None
IBM Card Code to or from EBCDIC	SPEED	None
EBCDIC and PTTC/8 Table	EBPA	None
IBM Card Code Table	HOLL	None
Console Printer Code Table	PRTY	None
DISK SUBROUTINE INITIALIZE (cord/paper tape only)		
Called by CALL		
Set Pack Initialize Routine	SPIR0, SPIR1, SPIRN	DISK0, DISK1, DISKN
OVERLAY (monitor only)		
Called by LIBF		
Local Read-in	FLIP0, FLIP1	DISKZ or DISK0, DISK1 or DISKN
INTERRUPT SERVICE		
Called by LIBF		
Card	CARD0, CARD1	ILS00, ILS04
Disk (part of supervisor in monitor system)	DISK0, DISK1, DISKN	ILS02
Paper Tape	PAPT1, PAPTN	ILS04
Plotter	PLOT1	ILS04
1132 Printer	PRNT1	ILS01
Console Printer-Keyboard	TYPE0, WRTY0	HOLL, PRTY, ILS04

APPENDIX A. 1130 SUBROUTINE LIBRARY

Subroutine	Names	Other Subroutines Required
FORTRAN		
Called by CALL		
Loader Reinitialization (card only)	LOAD	None
Data Switch	DATSW	None
Sense Light On	SLITE, SLITT	None
Overflow Test	OVERF	None
Divide Check Test	DVCHK	None
Function Test	FCTST	None
Trace Start	TSTRT	TSET
Trace Stop	TSTOP	TSET
Integer Transfer of Sign	ISIGN	None
Real Transfer of Sign (E)	ESIGN	ESUB, ELD
Real Transfer of Sign (S)	FSIGN	FSUB, FLD
Called by LIBF (card/paper tape)		
Real IF Trace (E)	VIF	TTEST, VWRT, VIOF, VCOMP
Real IF Trace (S)	WIF	FSTO, TTEST, WWRT, WIOF, WCOMP
Integer IF Trace (E)	VIIF	TTEST, VWRT, VIOF, VCOMP
Integer IF Trace (S)	WIIF	TTEST, WWRT, WIOI, WCOMP
Integer Arithmetic Trace (E)	VIAR, VIARX	TTEST, VWRT, VIOI, VCOMP
Integer Arithmetic Trace (S)	WIAR, WIARX	TTEST, WWRT, WIOI, WCOMP
Real Arithmetic Trace (E)	VARI, VARIX	ESTO, TTEST, VWRT, VIOF, VCOMP
Real Arithmetic Trace (S)	WARI, WARIX	FSTO, TTEST, WWRT, WIOF, WCOMP
Computed GO TO Trace (E)	VGOTO	TTEST, VWRT, VIOI, VCOMP
Computed GO TO Trace (S)	WGOTO	TTEST, WWRT, WIOI, WCOMP
Trace Test-Set Indicator	TTEST, TSET	None
Pause	PAUSE	None
Stop	STOP	None
Subscript Calculation	SUBSC	None
Store Argument Address	SUBIN	None
I/O Linkage (E)	VFIO, VRED, VWRT, VCOMP, VIOAI, VIOAF, VIOFX, VIOIX, VIOF, VIOI	FLOAT, ELD/ESTO, IFIX
I/O Linkage (S)	WFIO, WRED, WWRT, WCOMP, WIOAI, WIOAF, WIOFX, WIOIX, WIOF, WIOI	FLOAT, FLD/FSTO, IFIX
Card Input/Output	CARDZ	HOLEZ
Printer-Keyboard Output	WRTYZ	GETAD, EBCTB
Printer-Keyboard Input/Output	TYPEZ	GETAD, EBCTB, HOLEZ
1132 Printer Output	PRNTZ	None
Paper Tape Input/Output	PAPTZ	None
Card Code-EBCDIC Conversion	HOLEZ	GETAD, EBCTB, HOLTB
Console Printer Code Table	EBCTB	None
Card-Keyboard Code Table	HOLTB	None
Address Calculation	GETAD	None
Called by LIBF (monitor)		
Real IF Trace (E)	SEIF	FSTO, TTEST, SWRT, SIOF, SCOMP
Real IF Trace (S)	SFIF	FSTO, TTEST, SWRT, SIOF, SCOMP
Integer IF Trace	SIIF	TTEST, SWRT, SIOI, SCOMP
Integer Arithmetic Trace	SIAR, SIARX	TTEST, SWRT, SIOI, SCOMP
Real Arithmetic Trace (E)	SEAR, SEARX	ESTO, TTEST, SWRT, SIOF, SCOMP
Real Arithmetic Trace (S)	SFAR, SFARX	FSTO, TTEST, SWRT, SIOF, SCOMP
Computed GO TO Trace	SGOTO	TTEST, SWRT, SIOI, SCOMP
Trace Test-Set Indicator	TTEST, TSET	None
Pause	PAUSE	None
Stop	STOP	None
Subscript Calculation	SUBSC	None
Store Argument Address	SUBIN	None
I/O Linkage (non-disk)	SFIO, SRED, SWRT, SCOMP, SIOAF, SIOAI, SIOF, SIOI, SIOFX, SIOIX	FLOAT, ELD/ESTO or FLD/FSTO, IFIX
Disk-I/O Linkage	SDFIO, SDRED, SDWRT, SDCOM, SDAF, SDAI, SDF, SDI, SDFX, SDIX	DISKZ
Disk Find	SDFND	DISKZ
Card Input/Output	CARDZ	HOLEZ
Disk Input/Output (part of supervisor)	DISKZ	None
Printer-Keyboard Output	WRTYZ	GETAD, EBCTB

APPENDIX B. ERRORS DETECTED BY THE ISS SUBROUTINES

ERROR	CONTENTS OF ACCUMULATOR		Contents of Extension (if any)
	Binary	Hexadecimal	
Card			
*Last card	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	0 0 0 0	
*Feed check *Read check *Punch check	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 1	
Device not ready Last card indicator on for Read	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0	1 0 0 0	
Illegal device (not 0 version) Device not in system Illegal function Word count over +80 Word count zero or negative	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1	1 0 0 1	
Printer-Keyboard			
Device not ready	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	2 0 0 0	
Device not in system Illegal function Word count zero or negative	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1	2 0 0 1	
Paper Tape			
*Punch not ready	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0	0 0 0 4	
*Reader not ready	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1	0 0 0 5	
Device not ready	0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0	3 0 0 0	
Illegal device Illegal function Word count zero or negative Illegal check digit	0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 1	3 0 0 1	
Disk			
*Disk overflow	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0	0 0 0 4	
*Seek failure remaining after ten attempts	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	0 0 0 3	Effective Sector Id
*Read check remaining after ten attempts Data Error Data overrun	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 1	Effective Sector Id
*Write check remaining after ten attempts Write select Data error Data Overrun	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 2	Effective Sector Id
Device not ready	0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0	5 0 0 0	
Illegal device (not 0 version) Device not in system Illegal function Attempt to write in file protected area Word count zero or negative Word count over +320 (0 version only) Starting sector identification over + 1599	0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 1	5 0 0 1	
1132 Printer			
*Channel 9 detected	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	0 0 0 3	
*Channel 12 detected	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0	0 0 0 4	
Device not ready or end of forms	0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0	6 0 0 0	
Illegal function Illegal word count	0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1	6 0 0 1	
Plotter			
Plotter not ready	0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0	7 0 0 0	
Illegal device Device not in system Illegal function Word count zero or negative	0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1	7 0 0 1	

NOTE: The errors marked with an asterisk cause a branch via the error parameter. These errors are detected during the processing of interrupts; as a consequence, the user error routine is an interrupt routine, executed at the priority level of the I/O device.

All other errors cause a branch to location 41. The address of the LIBF in error is in location 40.

APPENDIX C. SUBROUTINE ACTION AFTER RETURN FROM A USER'S ERROR ROUTINE

Error Code	Condition	Subroutine Action
Card		
0000	If function is PUNCH	Eject card and terminate
	Otherwise	Terminate Immediately
0001*	If Accumulator is 0	Terminate immediately
	Otherwise	Loop until 1442 is ready, then reinitiate operation
Paper Tape		
0004, 0005	If Accumulator is 0	Terminate immediately
	Otherwise	Check again for device ready
Disk		
0001, 0002, and 0003	If A Reg. is 0	Terminate immediately
	Otherwise	Retry 10 more times
1132 Printer		
0003, and 0004	If Accumulator is 0	Terminate immediately
	Otherwise	Skip to channel 1 and then terminate

*Assumes operator intervention.

APPENDIX D. CHARACTER CODE CHART

Ref No.	EBCDIC Binary 0123 4567	EBCDIC Hex	IBM Card Code Rows 12 11 0 9 8 7-1	IBM Card Code Hex	Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex Notes
0	0000 0000	00	12 0 9 8 1	B030	NUL			
1	0001	01	12 9 1	9010				
2	0010	02	12 9 2	8810				
3	0011	03	12 9 3	8410				
4	0100	04	12 9 4	8210	PF Punch Off		6D (U/L)	41 ①
5*	0101	05	12 9 5	8110	HT Horiz.Tab		6E (U/L)	
6*	0110	06	12 9 6	8090	LC Lower Case		7F (U/L)	
7*	0111	07	12 9 7	8050	DEL Delete			
8	1000	08	12 9 8	8030				
9	1001	09	12 9 8 1	9030				
10	1010	0A	12 9 8 2	8830				
11	1011	0B	12 9 8 3	8430				
12	1100	0C	12 9 8 4	8230				
13	1101	0D	12 9 8 5	8130				
14	1110	0E	12 9 8 6	80B0				
15	1111	0F	12 9 8 7	8070				
16	0001 0000	10	12 11 9 8 1	D030				
17	0001	11	11 9 1	5010				
18	0010	12	11 9 2	4810				
19	0011	13	11 9 3	4410				
20*	0100	14	11 9 4	4210	RES Restore		4C (U/L)	05 ②
21*	0101	15	11 9 5	4110	NL New Line		DD (U/L)	81 ③
22*	0110	16	11 9 6	4090	BS Backspace		5E (U/L)	11
23	0111	17	11 9 7	4050	IDL Idle			
24	1000	18	11 9 8	4030				
25	1001	19	11 9 8 1	5030				
26	1010	1A	11 9 8 2	4830				
27	1011	1B	11 9 8 3	4430				
28	1100	1C	11 9 8 4	4230				
29	1101	1D	11 9 8 5	4130				
30	1110	1E	11 9 8 6	40B0				
31	1111	1F	11 9 8 7	4070				
32	0010 0000	20	11 0 9 8 1	7030				
33	0001	21	0 9 1	3010				
34	0010	22	0 9 2	2810				
35	0011	23	0 9 3	2410				
36	0100	24	0 9 4	2210	BYP Bypass		3D (U/L)	03
37*	0101	25	0 9 5	2110	LF Line Feed		3E (U/L)	
38*	0110	26	0 9 6	2090	EOB End of Block			
39	0111	27	0 9 7	2050	PRE Prefix			
40	1000	28	0 9 8	2030				
41	1001	29	0 9 8 1	3030				
42	1010	2A	0 9 8 2	2830				
43	1011	2B	0 9 8 3	2430				
44	1100	2C	0 9 8 4	2230				
45	1101	2D	0 9 8 5	2130				
46	1110	2E	0 9 8 6	20B0				
47	1111	2F	0 9 8 7	2070				
48	0011 0000	30	12 11 0 9 8 1	F030				
49	0001	31	9 1	1010				
50	0010	32	9 2	0810				
51	0011	33	9 3	0410				
52	0100	34	9 4	0210	PN Punch On		0D (U/L)	09 ④
53*	0101	35	9 5	0110	RS Reader Stop		0E (U/L)	
54*	0110	36	9 6	0090	UC Upper Case			
55	0111	37	9 7	0050	EOT End of Trans.			
56	1000	38	9 8	0030				
57	1001	39	9 8 1	1030				
58	1010	3A	9 8 2	0830				
59	1011	3B	9 8 3	0430				
60	1100	3C	9 8 4	0230				
61	1101	3D	9 8 5	0130				
62	1110	3E	9 8 6	00B0				
63	1111	3F	9 8 7	0070				

NOTES: Typewriter Output
① Tabulate
② Shift to black
③ Carrier Return
④ Shift to red

* Recognized by all Conversion subroutines
Codes that are not asterisked are recognized only by the SPEED subroutine

Ref No.	EBCDIC Binary 0123	EBCDIC Binary 4567	EBCDIC Hex	IBM Card Code Rows 12 11 0 9 8 7-1	IBM Card Code Hex	Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex
64*	0100	0000	40	no punches	0000	(space)	ǂ	10 (U/L)	21
65		0001	41	12 0 9 1	B010				
66		0010	42	12 0 9 2	A810				
67		0011	43	12 0 9 3	A410				
68		0100	44	12 0 9 4	A210				
69		0101	45	12 0 9 5	A110				
70		0110	46	12 0 9 6	A090				
71		0111	47	12 0 9 7	A050				
72		1000	48	12 0 9 8	A030				
73		1001	49	12 8 1	9020				
74*		1010	4A	12 8 2	8820	¢		20 (U)	02
75*		1011	4B	12 8 3	8420	. (period)	4B	6B (L)	00
76*		1100	4C	12 8 4	8220	<		02 (U)	DE
77*		1101	4D	12 8 5	8120	(	4D	19 (U)	FE
78*		1110	4E	12 8 6	80A0	+	4E	70 (U)	DA
79*		1111	4F	12 8 7	8060	| (logical OR)		3B (U)	C6
80*	0101	0000	50	12	8000	&	50	70 (L)	44
81		0001	51	12 11 9 1	D010				
82		0010	52	12 11 9 2	C810				
83		0011	53	12 11 9 3	C410				
84		0100	54	12 11 9 4	C210				
85		0101	55	12 11 9 5	C110				
86		0110	56	12 11 9 6	C090				
87		0111	57	12 11 9 7	C050				
88		1000	58	12 11 9 8	C030				
89		1001	59	11 8 1	5020				
90*		1010	5A	11 8 2	4820	!		5B (U)	42
91*		1011	5B	11 8 3	4420	$	5B	5B (L)	40
92*		1100	5C	11 8 4	4220	*	5C	08 (U)	D6
93*		1101	5D	11 8 5	4120	)	5D	1A (U)	F6
94*		1110	5E	11 8 6	40A0	;		13 (U)	D2
95*		1111	5F	11 8 7	4060	¬ (logical NOT)		6B (U)	F2
96*	0110	0000	60	11	4000	- (dash)	60	40 (L)	84
97*		0001	61	0 1	3000	/	61	31 (L)	BC
98		0010	62	11 0 9 2	6810				
99		0011	63	11 0 9 3	6410				
100		0100	64	11 0 9 4	6210				
101		0101	65	11 0 9 5	6110				
102		0110	66	11 0 9 6	6090				
103		0111	67	11 0 9 7	6050				
104		1000	68	11 0 9 8	6030				
105		1001	69	0 8 1	3020				
106		1010	6A	12 11	C000				
107*		1011	6B	0 8 3	2420	, (comma)	6B	3B (L)	80
108*		1100	6C	0 8 4	2220	%		15 (U)	06
109*		1101	6D	0 8 5	2120	_ (underscore)		40 (U)	BE
110*		1110	6E	0 8 6	20A0	>		07 (U)	46
111*		1111	6F	0 8 7	2060	?		31 (U)	86
112	0111	0000	70	12 11 0	E000				
113		0001	71	12 11 0 9 1	F010				
114		0010	72	12 11 0 9 2	E810				
115		0011	73	12 11 0 9 3	E410				
116		0100	74	12 11 0 9 4	E210				
117		0101	75	12 11 0 9 5	E110				
118		0110	76	12 11 0 9 6	E090				
119		0111	77	12 11 0 9 7	E050				
120		1000	78	12 11 0 9 8	E030				
121		1001	79	8 1	1020				
122*		1010	7A	8 2	0820	:		04 (U)	82
123*		1011	7B	8 3	0420	#		0B (L)	C0
124*		1100	7C	8 4	0220	@		20 (L)	04
125*		1101	7D	8 5	0120	' (apostrophe)	7D	16 (U)	E6
126*		1110	7E	8 6	00A0	=	7E	01 (U)	C2
127*		1111	7F	8 7	0060	"		0B (U)	E2

ǂ Any code other than those defined will be interpreted by PRNT1 as a space.

Ref No.	EBCDIC Binary 0123	4567	EBCDIC Hex	IBM Card Code Rows 12	11	0	9	8	7-1	IBM Card Code Hex	Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex
128	1000	0000	80	12		0		8	1	B020				
129		0001	81	12		0			1	B000	a			
130		0010	82	12		0			2	A800	b			
131		0011	83	12		0			3	A400	c			
132		0100	84	12		0			4	A200	d			
133		0101	85	12		0			5	A100	e			
134		0110	86	12		0			6	A080	f			
135		0111	87	12		0			7	A040	g			
136		1000	88	12		0		8		A020	h			
137		1001	89	12		0	9			A010	i			
138		1010	8A	12		0		8	2	A820				
139		1011	8B	12		0		8	3	A420				
140		1100	8C	12		0		8	4	A220				
141		1101	8D	12		0		8	5	A120				
142		1110	8E	12		0		8	6	A0A0				
143		1111	8F	12		0		8	7	A060				
144	1001	0000	90	12	11			8	1	D020				
145		0001	91	12	11				1	D000	j			
146		0010	92	12	11				2	C800	k			
147		0011	93	12	11				3	C400	l			
148		0100	94	12	11				4	C200	m			
149		0101	95	12	11				5	C100	n			
150		0110	96	12	11				6	C080	o			
151		0111	97	12	11				7	C040	p			
152		1000	98	12	11			8		C020	q			
153		1001	99	12	11		9			C010	r			
154		1010	9A	12	11			8	2	C820				
155		1011	9B	12	11			8	3	C420				
156		1100	9C	12	11			8	4	C220				
157		1101	9D	12	11			8	5	C120				
158		1110	9E	12	11			8	6	C0A0				
159		1111	9F	12	11			8	7	C060				
160	1010	0000	A0		11	0		8	1	7020				
161		0001	A1		11	0			1	7000				
162		0010	A2		11	0			2	6800	s			
163		0011	A3		11	0			3	6400	t			
164		0100	A4		11	0			4	6200	u			
165		0101	A5		11	0			5	6100	v			
166		0110	A6		11	0			6	6080	w			
167		0111	A7		11	0			7	6040	x			
168		1000	A8		11	0		8		6020	y			
169		1001	A9		11	0	9			6010	z			
170		1010	AA		11	0		8	2	6820				
171		1011	AB		11	0		8	3	6420				
172		1100	AC		11	0		8	4	6220				
173		1101	AD		11	0		8	5	6120				
174		1110	AE		11	0		8	6	60A0				
175		1111	AF		11	0		8	7	6060				
176	1011	0000	B0	12	11	0		8	1	F020				
177		0001	B1	12	11	0			1	F000				
178		0010	B2	12	11	0			2	E800				
179		0011	B3	12	11	0			3	E400				
180		0100	B4	12	11	0			4	E200				
181		0101	B5	12	11	0			5	E100				
182		0110	B6	12	11	0			6	E080				
183		0111	B7	12	11	0			7	E040				
184		1000	B8	12	11	0		8		E020				
185		1001	B9	12	11	0	9			E010				
186		1010	BA	12	11	0		8	2	E820				
187		1011	BB	12	11	0		8	3	E420				
188		1100	BC	12	11	0		8	4	E220				
189		1101	BD	12	11	0		8	5	E120				
190		1110	BE	12	11	0		8	6	E0A0				
191		1111	BF	12	11	0		8	7	E060				

Ref No.	EBCDIC Binary 0123	EBCDIC Binary 4567	EBCDIC Hex	IBM Card Code Rows 12	11	0	9	8	7-1	IBM Card Code Hex	Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex
192	1100	0000	C0	12		0				A000	(+ zero)			
193*		0001	C1	12					1	9000	A	C1	61 (U)	3C or 3E
194*		0010	C2	12					2	8800	B	C2	62 (U)	18 or 1A
195*		0011	C3	12					3	8400	C	C3	73 (U)	1C or 1E
196*		0100	C4	12					4	8200	D	C4	64 (U)	30 or 32
197*		0101	C5	12					5	8100	E	C5	75 (U)	34 or 36
198*		0110	C6	12					6	8080	F	C6	76 (U)	10 or 12
199*		0111	C7	12					7	8040	G	C7	67 (U)	14 or 16
200*		1000	C8	12				8		8020	H	C8	68 (U)	24 or 26
201*		1001	C9	12			9			8010	I	C9	79 (U)	20 or 22
202		1010	CA	12		0	9	8	2	A830				
203		1011	CB	12		0	9	8	3	A430				
204		1100	CC	12		0	9	8	4	A230				
205		1101	CD	12		0	9	8	5	A130				
206		1110	CE	12		0	9	8	6	A0B0				
207		1111	CF	12		0	9	8	7	A070				
208	1101	0000	D0		11	0				6000	(- zero)			
209*		0001	D1		11				1	5000	J	D1	51 (U)	7C or 7 E
210*		0010	D2		11				2	4800	K	D2	52 (U)	58 or 5A
211*		0011	D3		11				3	4400	L	D3	43 (U)	5C or 5E
212*		0100	D4		11				4	4200	M	D4	54 (U)	70 or 72
213*		0101	D5		11				5	4100	N	D5	45 (U)	74 or 76
214*		0110	D6		11				6	4080	O	D6	46 (U)	50 or 52
215*		0111	D7		11				7	4040	P	D7	57 (U)	54 or 56
216*		1000	D8		11			8		4020	Q	D8	58 (U)	64 or 66
217*		1001	D9		11		9			4010	R	D9	49 (U)	60 or 62
218		1010	DA	12	11		9	8	2	C830				
219		1011	DB	12	11		9	8	3	C430				
220		1100	DC	12	11		9	8	4	C230				
221		1101	DD	12	11		9	8	5	C130				
222		1110	DE	12	11		9	8	6	C0B0				
223		1111	DF	12	11		9	8	7	C070				
224	1110	0000	E0			0		8	2	2820				
225		0001	E1		11	0	9		1	7010				
226*		0010	E2			0			2	2800	S	E2	32 (U)	98 or 9A
227*		0011	E3			0			3	2400	T	E3	23 (U)	9C or 9E
228*		0100	E4			0			4	2200	U	E4	34 (U)	B0 or B2
229*		0101	E5			0			5	2100	V	E5	25 (U)	B4 or B6
230*		0110	E6			0			6	2080	W	E6	26 (U)	90 or 92
231*		0111	E7			0			7	2040	X	E7	37 (U)	94 or 96
232*		1000	E8			0		8		2020	Y	E8	38 (U)	A4 or A6
233*		1001	E9			0	9			2010	Z	E9	29 (U)	A0 or A2
234		1010	EA		11	0	9	8	2	6830				
235		1011	EB		11	0	9	8	3	6430				
236		1100	EC		11	0	9	8	4	6230				
237		1101	ED		11	0	9	8	5	6130				
238		1110	EE		11	0	9	8	6	60B0				
239		1111	EF		11	0	9	8	7	6070				
240*	1111	0000	F0			0				2000	0	F0	1A (L)	C4
241*		0001	F1						1	1000	1	F1	01 (L)	FC
242*		0010	F2						2	0800	2	F2	02 (L)	D8
243*		0011	F3						3	0400	3	F3	13 (L)	DC
244*		0100	F4						4	0200	4	F4	04 (L)	F0
245*		0101	F5						5	0100	5	F5	15 (L)	F4
246*		0110	F6						6	0080	6	F6	16 (L)	D0
247*		0111	F7						7	0040	7	F7	07 (L)	D4
248*		1000	F8					8		0020	8	F8	08 (L)	E4
249*		1001	F9				9			0010	9	F9	19 (L)	E0
250		1010	FA	12	11	0	9	8	2	E830				
251		1011	FB	12	11	0	9	8	3	E430				
252		1100	FC	12	11	0	9	8	4	E230				
253		1101	FD	12	11	0	9	8	5	E130				
254		1110	FE	12	11	0	9	8	6	E0B0				
255		1111	FF	12	11	0	9	8	7	E070				

APPENDIX E. CORE REQUIREMENTS

Table 6. Arithmetic and Functional Subroutines

Standard		Extended	
FADD/FADDX FSUB/FSUBX	102	EADD/EADDX ESUB/ESUBX	98
FMPY/FMPYX	52	EMPY/EMPYX	46
FDIV/FDIVX	86	EDIV/EDIVX	78
FLD/FLDX FSTO/FSTOX	54	ELD/ELDX ESTO/ESTOX	46
FLOAT	10		10
IFIX	40		40
NORM	42		42
FSBR/FSBRX	24	ESBR/ESBRX	24
FDVR/FDVRX	28	EDVR/EDVRX	28
SNR	8		8
FABS/FAVL	12	EABS/EAVL	12
IABS	16		16
FGETP	22	EGETP	22
FARC	34		34
XMDS	28		--
FIXI/FIXIX	68		68
XSQR	52		52
XMD	66		66
XDD	74		74
FSIN/FSINE FCOS/FCOSN	108	ESIN/ESINE ECOS/ECOSN	138
FATN/FATAN	130	EATN/EATAN	150
FSQR/FSQRT	70	ESQR/ESQRT	76
FLN/FALOG	136	ELN/EALOG	148
FEXP/FEXPN	118	EEXP/EXPN	140
FAXI/FAXIX	78	EAXI/EAXIX	82
FAXB/FAXBX	54	EAXB/EAXBX	54
FTNH/FTANH	54	ETNH/ETANH	46
FBTD (bin. to dec.) FDTB (dec. to bin.)	420		420
DMTD0/DMTX0	412		412
DMPD1/DMPX1	520		520
DMP80	102		102
DATSW	34		34
DVCHK	16		16

Standard		Extended	
FCTST	30		30
LOAD	138		138
OVERF	18		18
SLITE, SLITT	68		68
TSTOP	6		6
TSTRT	6		6
ISIGN	24		24
FSIGN	34	ESIGN	34
Card/Paper Tape Only		Card/Paper Tape Only	
WARI/WARIX	32	VARI/VARIX	32
WIAR/WIARX	36	VIAR/VIARX	36
WIF	26	VIF	26
WIIF	24	VIIF	24
WGOTO	22	VGOTO	22
WFIO/WIOI/WIOAI/ WIOF/WIOAF/ WIOFX/WCOMP/ WWRT/WRED/ WIOIX	854	VFIO/VIOI/VIOAI/ VIOF/VIOAF/ VIOFX/VCOMP/ VWRT/VRED/ VIOIX	864
Monitor Only		Monitor Only	
SDFIO/SDAF/SDAI/ SDCOM/SDF/SDFX/ SDI/SDIX/SDRED/ SDWRT	602		602
SDFND	60		60
SFAR/SFARX	32	SEAR/SEARX	32
SFIO/SIOI/SIOAI/ SIOF/SIOAF/SIOFX/ SCOMP/SWRT/SRED/ SIOIX	892		892
SFIF	26	SEIF	28
SGOTO	22		22
SIAR/SIARX	36		36
SIIF	24		24

Table 7. Miscellaneous and ISS Subroutines

Subroutines	No. Core Locations	Uses
CARD0	242	ILS00, ILS04
CARD1	246	ILS00, ILS04
PAPT1	254	ILS04
PAPTN	294	ILS04
DISK0	356 (1)	ILS02
DISK1	620 (1)	ILS02
DISKN	808 (1)	ILS02
WRTY0	124	ILS04
TYPE0	296	ILS04, PRTY, HOLL
PLOT1	216	ILS03
PRNT1	386	ILS01
ILS00	18	
ILS01	18	
ILS02	18	
ILS03	18	
ILS04	30	
SPIR0	48 (2)	
SPIR1	62 (2)	
SPIRN	62 (2)	
FLIP0	72 (3)	DISK0, DISKZ
FLIP1	48 (3)	DISK1, DISKN
PAUSE	12	
STOP	Card/P.T. 8	
	Monitor 12	
SUBSC	30	
SUBIN	32	
TTEST/TSET	16	
CARDZ	80	
PAPTZ	202	
PRNTZ	176	
TYPEZ	82	
WRYTZ	60	
HOLEZ	54	
GETAD	14	
EBCTB	54	
HOLTB	54	

(1) Card/Paper Tape only, part of supervisor in Monitor.
(2) Card/Paper Tape only
(3) Monitor only

Table 8. Conversion Subroutines

Conversion Subroutines	No. Core Locations	Uses
BINDC	72	
DCBIN	88	
BINHX	44	
HXBIN	66	
HOLEB	134	HOLL, EBPA
HOLPR	100	HOLL, PRTY
EBPRT	102	EBPA, PRTY
PAPEB	246	EBPA
PAPHL	244	EBPA, HOLL
PAPPR	192	EBPA, PRTY
SPEED	330	
HOLL	80	
EBPA	80	
PRTY	80	

APPENDIX F. EXECUTION TIMES

CONVERSION SUBROUTINES (see Table 9)

Basic Definitions

1. All times are based on 3.6 μsec memory.
2. The table ordering for codes is as follows (except SPEED)
   - Standard set: blank, +, &, -, 0-9, A-Z, other special
   - Extended set: standard, non-FORTRAN special, control
3. Maximum number of characters checked varies with the set.
   - Standard set
     - Except SPEED: 49
     - SPEED only: 16
   - Extended set
     - Except SPEED: 74
     - SPEED only: 45
4. Conversion times given are
   - Best time: Found as first character in set
   - Worst time, standard set: Found as last character in set
   - Worst time, extended set: Not found in set
5. Time per character is best time, plus table look-up time multiplied by the number of characters to be skipped.
   Example:
   - If best = 211, look-up = 45.5 and character is fourth in table (-)
   - Then, character time = 347.5 = 211 + (45.5)

1130 ISS TIMES (see Table 10)

Basic Definitions

1. Only CPU time used by ISS (including transfer vector BSC L) and ILS (including forced BSI I) is given. All the remaining time, minus cycle steals, is available to the user.
2. ILS time is included in ISS interrupt processing calculations
   - ILS00 - CARD0 (col), CARD1 (col)
   - ILS01 - PRNT1
   - ILS02 - DISK0, DISK1, DISKN
   - ILS03 - PLOT1
   - ILS04 - CARD0 (op complt), CARD1 (op complt) WRTY0, TYPE0, PAPT1, PAPTN
3. All times are based on a 3.6 sec memory.

Table 9. Conversion Subroutines

Subroutine	Initial-ization	Time, Per Character			Table Look-Up
		Best	Worst		
			Std. Set	Extd. Set	
BINDC	1130	-	-	-	-
DCBIN	1110	-	-	-	-
BINHX	620	-	-	-	-
HXBIN	760	-	-	-	-
HOLPR	430	211	2395	3533	45.5
EBPRT	420	207	2487	3675	47.5
HOLEB					
EBCDIC output	550	159	2343	3481	45.5
EBCDIC input	550	161	2441	3629	47.5
SPEED					
Packed EBCDIC output	250	270	-	-	-
Unpacked EBCDIC output	270	260	-	-	-
Packed EBCDIC input	240	394	1594	3914	80.0
Unpacked EBCDIC input	240	404	1604	3924	80.0
PAPPR	580				
Per shift char. input		180	-	-	-
Per graphic char. input		427	2707	3895	47.5
Per control char. input		407	2687	3875	47.5
PAPHL					
PTTC/8 input	490				
Per shift char. input		180	-	-	-
Per graphic char. input		306	2482	3870	49.5
Per control char. input		296	2472	3860	49.5
PTTC/8 output	490				
Per control char. output		266	-	3830	49.5
Per graphic char. output		316	2492	3880	49.5
Per shift/graphic char. output		446	2622	4010	49.5
PAPEB					
PTTC/8 input	440				
Per shift char. input		190	-	-	-
Per graphic char. input		366	2542	3930	49.5
Per control char. input		386	2562	3950	49.5
PTTC/8 output	440				
Control char. output		296	-	3860	49.5
Per graphic char. output		346	2522	3910	49.5
Per shift/graphic char. output		476	2652	4040	49.5

Table 10. 1130 ISS Times

Subroutine and Function	Times (μsec) (n = word count)
ILS00	112
ILS01	112
ILS02	112
ILS03	112
ILS04	148
CARD0	
Test	165
Read	14930 + 38.5 (n)
Punch	763 + 185 (n)
Feed	605
Sel. Stack.	290
CARD1	
Test	165
Read	14972 + 38.5 (n)
Punch	800 + 190 (n)
Feed	640
Sel. Stack.	325
WRTY0	
Test	165
Print	228 + 734 (n)
TYPE0	
Test	165
Read print	685 + ϵ (825 + 48.5y) + 390 a + 1595 b + 1224 c ϵ = sum of char. times for each graphic y = no. char. skipped in table look-up a = EOM character b = re-entry character c = backspace character
Print	344 + 920 (n)
PAPT1	
Test	152
Read	432 + 808* (n) *add +112 if check
Punch	480 + 680* (n) *add +96 if check
PAPTN	
Test	176
Read	408 + 952* (n) *add +112 if check
Punch	464 + 840* (n) *add +64 if check
PLOT1	
Test	130
Print	698 + { 418 = if char is 0-9; 472 = if char is A; 624 = if char is B; 752 = if char is C; 224 = per dup. of previous pen motion }

Subroutine and Function	Times (μsec) (n = word count)
PRNT1	
Test	188
Print	44142 + 5971.2 (n-1)* *subtract 11.4 for each word where 1 char. does not match; 22.8 where both char. do not match.
Print Numeric	25950 + 2736.8 (n-1) +268 x x = no. idle cycles before 1st numeric char. on wheels is reached
Control	
Single space	708
Double space	998
Triple space	1288
Skip to channel 12	676*
Skip to channel 1	936* *add 208 for each channel crossed before correct one reached
DISK0	
Test	178
Read	1492
Write	
Without RBC	1778
With RBC	2050
Write Imm	1062
Seek	
1 to center	1076
By addr	1502
DISK1	
Test	178
Read	900 + 760 x + 478 y x = no. sectors y = no. seeks after 1st sector
Write	
Without RBC	1292 + 660 x + 822 y
Write	
With RBC	1562 + 1098 x + 908 y
Write Imm	660 + 622 x + 476 y
Seek	
1 to center	1072
By addr	1468
DISK N	
Test	178
Read	908 + 652 x + 1012 y x = no. sectors y = no. seeks after 1st sector
Write	
Without RBC	1516 + 610 x + 926 y
Write	
With RBC	1728 + 1022 x + 1178 y
Write Imm	820 + 606 x + 282 y
Seek	
1 to center	1076
By addr	1478

ARITHMETIC AND FUNCTIONAL SUBROUTINES

The execution times of the arithmetic and functional subroutines are shown in Table 11. All times are based on a 3.6 μsec memory; the times containing a decimal point are milliseconds, all other are microseconds.

SPIR

The SPIRx subroutines take 220 μsec (3.6 μsec memory) plus the DISKx time to read sector 0000.

Table 11. Arithmetic and Functional Subroutines

STANDARD		EXTENDED	
FADD/FADDX	460	EADD/EADDX	440
FSUB/FSUBX	560	ESUB/ESUBX	490
FMPY/FMPYX	560	EMPY/EMPYX	790
FDIV/FDIVX	766	EDIV/EDIVX	2060
FLD/FLDX	180	ELD/ELDX	160
FSTO/FSTOX	180	ESTO/ESTOX	170
FLOAT	330		330
IFIX	140		140
NORM	260		260
FSBR/FSBRX	650	ESBR/ESBRX	740
FDVR/FDVRX	1090	EDVR/EDVRX	2520
SNR	80		80
FABS/FAVL	50	EABS/EAVL	60
IABS	100		100
FGETP	330	EGETP	320
FARC	60		60
XMDS	260		--
FIXI/FIXIX	465		465
XSQR	550 av. (860 max.)		550 av. (860 max.)
XMD	520		520
XDD	1760		1760
FSIN/FSINE	3.0	ESIN/ESINE	5.4
FCOS/FCOSN	3.4	ECOS/ECOSN	5.9
FATAN/FATN	5.2	EATAN/EATN	8.9
FSQRT/FSQR	4.5	ESQRT/ESQR	10.4
FALOG/FLN	5.1	EALOG/ELN	8.0
FEXP/FXPN	2.0	EEXP/EXPN	4.4
FAXI/FAXIX	3.8	EAXI/EAXIX	4.7
FAXB/FAXBX	8.0	EAXB/EAXBX	13.3
FTANH/FTNH	4.3	ETANH/ETNH	8.1
FBTD (bin. to dec.)	40.0		40.0
FDTB (dec. to bin.)	20.0		20.0

INDEX