File Number 1130-30
Form C26-5929-4

Systems Reference Library

IBM 1130 Subroutine Library

This publication describes the libraries provided with the following programming systems:

- 1130 Card/Paper Tape
- 1130 Disk Monitor Version 1
- 1130 Disk Monitor Version 2

These libraries may be referenced within this manual as C/PT, DM1, and DM2, respectively. The library for DM2 is known as the System Library. DM2 core addresses are given in symbolic form. Figure 4 lists the absolute equivalents of these symbolic addresses.

The programming system libraries consist of input/output, conversion, arithmetic and functional, and utility subroutines. Included in the descriptions are calling sequences for the subroutines and explanations of the parameters involved.

The section on conversion subroutines describes the codes used to communicate with the 1130 system I/O devices. An appendix lists these codes and shows their relationship to each other.

PREFACE

The publication describes how the programmer can use the IBM 1130 library subroutines to increase the efficiency of his programs and decrease his writing and testing time. The libraries include the following programs.

- Interrupt Service Subroutines
- Interrupt Level Subroutines
- FORTRAN I/O Subroutines
- Data Code Conversion Subroutines
- Arithmetic and Functional Subroutines
- Selective Dump Subroutines
- Utility Programs

The subroutines are available for use with both the 1130 Assembler and the 1130 FORTRAN Compiler. The Utility Programs are executable under Monitor control.

In Assembler language, the user calls the subroutines via a calling sequence. The appropriate subroutine calls are generated by the FORTRAN compiler whenever a read, write, arithmetic, or CALL statement is encountered. This publication describes each subroutine and the required calling sequence. All subroutines in the 1130 libraries are included in the lists that appear in Appendix A.

It is assumed that the reader is familiar with the methods of data handling and the functions of instructions in the IBM 1130 Computing System. He must also be familiar with the assembler or compiler used in conjunction with the subroutines. The following IBM publications provide the prerequisite information.

IBM 1130 Functional Characteristics (Form A26-5881)

IBM 1130 Computing System Input/Output Units (Form A26-5890)

IBM 1130 Assembler Language (Form C26-5927)

IBM 1130/1800 Basic FORTRAN IV Language (Form C26-3715)

The operating procedures manuals for the programming systems also provide information on subroutine usage. These manuals are:

IBM 1130 Card/Paper Tape Programming System Operators Guide (Form C26-3629)

IBM 1130 Disk Monitor System Reference Manual (Form C26-3750)

IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide (Form C26-3717)

MACHINE CONFIGURATION

The use of the library subroutines requires the following minimum machine configuration:

IBM 1131 Central Processing Unit with 4096 words of core storage

IBM 1442 Card Read Punch, or IBM 1134 Paper Tape Reader with IBM 1055 Paper Tape Punch

In addition, the following input/output units and features can be controlled by the input/output subroutines. (The 1403, 2310, 2501 and 1231 are supported by DM2 only.)

Console Printer/Keyboard
Single Disk Storage
1132 Printer
1627 Plotter
1403 Printer
2310 Disk Storage
2501 Card Reader
1231 Optical Mark Page Reader
Synchronous Communications Adapter

Plotter subroutines are described in IBM 1130/1800 Plotter Subroutines (Form C26-3755).

SCA subroutines are described in IBM 1130 Synchronous Communications Adapter Subroutines (Form C26-3706).

Fourth Edition

This edition (C26-5929-4) is a major revision of the previous edition (C26-5929-3) which is now obsolete. Information has been added that enables the user to program either version of the 1130 Disk Monitor System. Changes are indicated by a vertical line at the left of affected text and a bullet (●) at the left of a title of a changed illustration.

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications Department 232, San Jose, California 95114.

CONTENTS

INTRODUCTION

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 the reading of 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 subroutines in the 1130 system consists of a CALL or LIBF statement (whichever is required to call the specific subroutine), followed by the DC statements that make up the parameter list. The calling sequences for the various subroutines in the libraries are presented later in the manual. Each subroutine is self-contained, so that only those subroutines required by the current job are in core storage during execution.

The interrupt service subroutines (ISSs) transfer data from and to the various input/output devices attached to the computer. These subroutines handle all the details peculiar to each device, including the usually complex interrupt functions, and can control many input/output devices at the same time by overlapping their operations.

ISS CHARACTERISTICS

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

- Methods of data transfer
- Interrupt processing
- ILS (interrupt level subroutine) operation
- ISS (interrupt service subroutine) 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

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 following system I/O devices: 1442 Card Read Punch, 1442 Card Punch, 1134 Paper Tape Reader and 1055 Paper Tape Punch, Console Printer, Keyboard, 1132 Printer, and 1627 Plotter.

Data Channel

Other system I/O devices operate via a data channel, which requires an I/O operation only to initiate data transfer. These devices are provided with control information, word-counts, and data from the user's I/O area. Once initiated, data transfer proceeds concurrently with program execution. An operation-complete interrupt signals the end of an I/O operation when all data has been transferred. All disk drives, the 1403 Printer, and the 2501 Card Reader operate via a data channel.

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 subroutine, performing certain housekeeping functions, and clearing the level by a BOSC instruction when interrupt processing is complete.

Level processing is done by the ILSs (interrupt level subroutines). Entered by interrupts, ILSs give temporary control to device processing subroutines (ISSs) and eventually return control to the user program. The interrupt entrance address is stored during the loading of a core load or program, in the appropriate interrupt branch address; location 8 for interrupt level zero (ILS00), location 9 for interrupt level one (ILS01),..., location 12 for interrupt level four (ILS04). The device processing entrance address is computed during the loading of a core load from identifying information that is a part of the ILS.

In the card/paper tape system, the device processing entrance address is stored during the loading of a program 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 ISSs (interrupt service subroutines). The ISSs 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 an ILS as part of the interrupt processing. The calling entry point is specified in the ISS statement. The interrupt entry points are set up in the ISS and identified in the ILS. They are entered indirectly through a branch address table.

ILS OPERATION

The ISS/ILS package services all input/output interrupts.

Description

There is one ILS for each interrupt level used. Each subroutine 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 (Disk Monitor ILSs also save Index Register Two (XR2)).

Interrupt service subroutines are loaded first so that the loader loads only the ILSs that are required. For example, if a main program does not call the 1132 Printer subroutine, the subroutine for interrupt level 1 (ILS01) need not be loaded because no interrupts will occur on that level. An ILS cannot be called; it is included in a core load or program only if requested by an ISS (see ISS-Define Interrupt Service Entry Point in IBM 1130 Assembler Language, Form C26-5927).

When the ILSs are loaded, the core addresses assigned to them are incorporated into the appropriate locations in the Interrupt Transfer Vector (words 8-13). Interrupts occurring during execution of a user program cause a Branch Indirect, via the interrupt branch address, to the correct ILS.

Recurrent Subroutine Entries

Recurrent entries to a subroutine can result from 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 such 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 ISSs were written with the assumption that all LIBFs would be executed from the mainline level of interrupt priority. There are no provisions in any ISSs to handle recurrent entries.

ISS OPERATION

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

The disk subroutines are included here as ISSs even though in the Disk Monitor System they are not truly ISSs. They do however, have most of the characteristics of an ISS.

ISS Subdivision

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

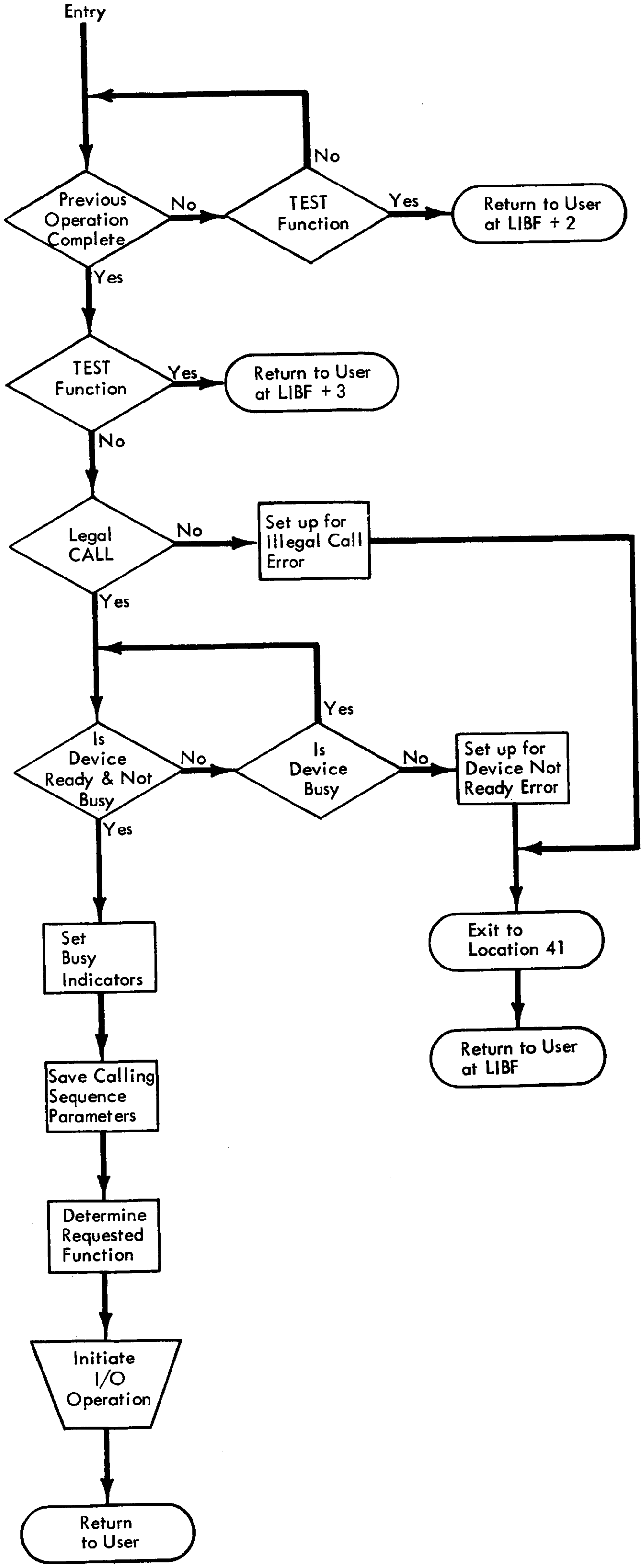
Call Processing

Each ISS saves and restores the contents of the Accumulator and Extension, Index Registers; and the Carry and Overflow indicators. The call portion, 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 the call portion of an ISS.

Determine Status of Previous Operation. This function can be performed by using a programmed subroutine-busy indicator to determine if a previous operation is complete. The CARD1 subroutine is a



Entry
Previous Operation Complete
No
TEST Function
No
Yes
Return to User at LIBF + 2
Yes
TEST Function
Yes
Return to User at LIBF + 3
No
Legal CALL
No
Set up for Illegal Call Error
Yes
Yes
Is Device Ready & Not Busy
No
Is Device Busy
No
Set up for Device Not Ready Error
Yes
Set Busy Indicators
Exit to Location 41
Return to User at LIBF
Save Calling Sequence Parameters
Determine Requested Function
Initiate I/O Operation
Return to User

Figure 1. Call Portion of an ISS

good example. When an operation is started on the 1442, a subsequent LIBF CARD1 for the 1442 is not honored until the subroutine-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 subroutine-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, a 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 paper tape reader is busy.

Check Legality of Calling Sequence. Calling sequences are checked for such items as illegal function character, illegal device identification code, zero or negative word count, etc.

Save Calling Sequence. The call portion saves, within itself, all of the calling sequence information needed to perform an I/O operation. The user can modify a calling sequence, even though an I/O operation is not yet complete.

NOTE: The I/O data area should be left intact during an operation because the ISS is continually accessing and modifying that area.

Initiate I/O Operation. The call portion only initiates an I/O operation. Subsequent character interrupts or operation complete interrupts are handled by the interrupt response routine.

Interrupt Response Processing

The I/O interrupt response portion of an ISS is illustrated in Figure 2.

Operation. An I/O interrupt causes a user program to exit to an interrupt level subroutine, which in turn exits to the I/O interrupt response portion of an ISS. The interrupt response portion checks for errors, does any necessary data manipulation, initiates character operations, and initiates retry operations in case of errors. It then returns control to the interrupt level subroutine, which returns control to the user.

Character Interrupts. These interrupts occur for devices under direct program control whenever data

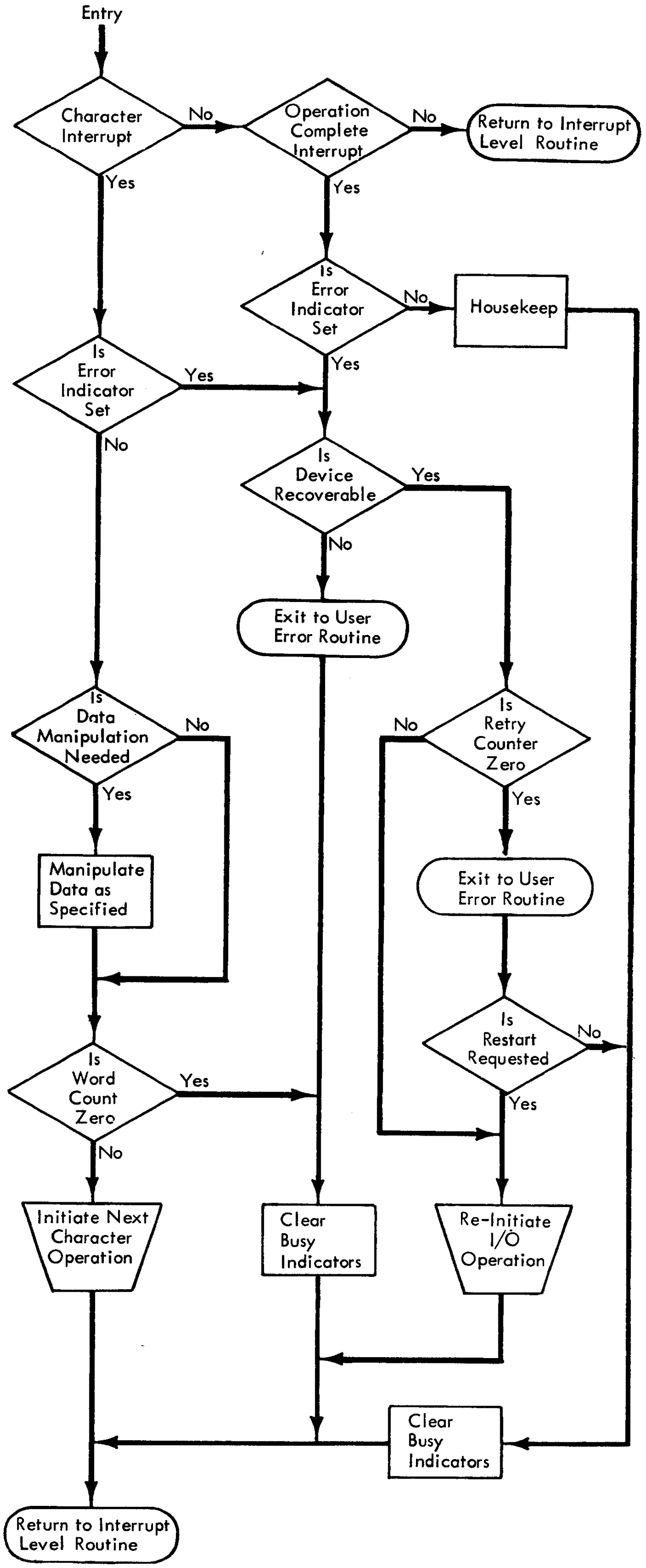

Entry
Character Interrupt
No
Yes
Operation Complete Interrupt
No
Yes
Return to Interrupt Level Routine
Is Error Indicator Set
No
Yes
Housekeep
Is Error Indicator Set
Yes
No
Is Device Recoverable
Yes
No
Exit to User Error Routine
Is Data Manipulation Needed
No
Yes
Is Retry Counter Zero
No
Yes
Manipulate Data as Specified
Exit to User Error Routine
Is Restart Requested
No
Yes
Is Word Count Zero
Yes
No
Initiate Next Character Operation
Clear Busy Indicators
Re-Initiate I/O Operation
Clear Busy Indicators
Return to Interrupt Level Routine

Figure 2. Interrupt Response Portion of an ISS

can be read or written, e.g., a card column punched or a paper tape character read.

Operation Complete Interrupts. These interrupts occur in disk and card operations when a specified block of data has been read or written, e.g., a disk record read.

Error Detection and Recovery Procedures. These procedures are an important part of an ISS. However, little can be done about reinitiating an operation until a character interrupt or operation complete interrupt occurs. Therefore, error indicators are not examined until one of these interrupts occurs.

Recoverable Device. This is an I/O device that can be easily repositioned by a subroutine or by an operator and an I/O operation reinitiated. If a device is not recoverable, or if an error cannot be corrected after a specified number of retries, the user is informed of the error condition. If a device is recoverable, the user may request, via his error subroutine, that the operation be reinitiated.

GENERAL ERROR-HANDLING PROCEDURES

Each ISS has its own error detecting portion, which determines the type of error and chooses an error procedure. (In this context, the term, error, includes such conditions as last card, channel 9, channel 12, etc.) Errors fall into one of two categories: those that are detected before an I/O operation is initiated, and those that are detected after an I/O operation has been initiated. Appendix B contains a list of the errors detected by the ISSs; Appendix C contains descriptions of the actions taken by each ISS after the return from user-written error subroutines.

Pre-Operative Error Detection

Before an ISS initiates an I/O operation, it checks the device status and the legality of calling sequence parameters. If a device is not ready, or a parameter is in error, the subroutine stores the address of the LIBF statement in core location $PRET(28) and exits to core location $PRET+1 (29), which is a WAIT instruction. The Accumulator contains an error indicator that defines the error (see Appendix E). This error indicator consists of four hexadecimal digits that are defined as follows.

Digit 1 identifies the ISS subroutine called:

DM1 and C/PT System

1 - CARD0 or CARD1

2 - TYPE0 or WRTY0

3 - PAPT1 or PAPTN

5 - DISK0, DISK1, or DISKN

6 - PRNT1

7 - PLOT1

8 - SCAT1, SCAT2, or SCAT3

DM2 System

1 - CARD0, CARD1, or CARDZ; PNCH0, PNCH1, or PNCHZ

2 - TYPE0 or TYPEZ, WRTY0 or WRTYZ

3 - PAPT1, PAPTN, PAPTX, or PAPTZ

4 - READ0, READ1, or READZ

5 - DISKZ, DISK1, or DISKN

6 - PRNT1 or PRNTZ

7 - PLOT1 or PLOTX

8 - SCAT1, SCAT2, or SCAT3

9 - PRNT3 or PRNZ

A - OMPR1

Digits 2 and 3 are reserved.

Digit 4 identifies the error:

0 - device not ready

1 - illegal LIBF parameter or illegal specification in the I/O area.

There is a WAIT instruction in core location $PRET+1 and a branch instruction (BSC I $PRET) in the next location. Therefore, the LIBF may be executed again (after the error condition has been corrected) by pressing PROGRAM START on the console. The user can, if he chooses, replace these two instructions with an exit to his own error subroutine.

Post-Operative Error Detection

After an I/O operation has been started, certain conditions may be detected about which the user should be informed. The conditions might be card jams for which manual intervention is needed before the operation can continue; read checks that have not been corrected after a specified number of retries; or indications of equipment readiness, such as last card or channel 12 indicators. All these conditions are detected by the interrupt response portion (see ISS Operation).

No Error Parameter. If no error parameter is included in the calling sequence that initiated the I/O operation and a post-operative error condition is detected, the card/paper tape system subroutine initiates a Wait procedure (programmed loop), which continues until an operator corrects the detected condition.

The DM2 system does not use a programmed loop, but rather branches to a post-operative error trap that is similar to the pre-operative error trap. Each interrupt level (1-4) has its own post-operative error trap with accompanying WAIT address.

Level 1 -	$PST1
Level 2 -	$PST2
Level 3 -	$PST3
Level 4 -	$PST4

Processing resumes after the operator corrects the detected condition and presses PROGRAM START.

Error Parameter Included. If an error parameter is included in the calling sequence, a Branch and Store Instruction Counter instruction (BSI) to the user's error subroutine specified in the calling sequence is executed. Identifying information is placed in the accumulator and extension (see Appendix B). When the user's error subroutine returns control to the ISS using the return link (see Basic ISS Calling Sequence), the subroutine examines the Accumulator. If the user has cleared the Accumulator before returning to the subroutine, he is requesting that the error condition be ignored and the operation terminated. If the user has not cleared the Accumulator, he is requesting that the operation be restarted, in which case the subroutine reinitiates the operation before returning to the user's main program.

Implications of the User's Error Subroutine. It is important to note that a user's error subroutine (entered via the LIBF error parameter address) is executed as part of the interrupt processing. The interrupt level is still on, preventing recognition of other interrupts of the same or lower priority. This has the following implications:

1. Return must be made to the ISS subroutine via the return link (set up by the BSI instruction executed by the ISS subroutine). Otherwise, normal processing cannot be continued because the ISS must return to the ILS to restore the contents of the Accumulator and Extension, Status Indicators, and Index Registers.

2. Return must be made with a BSC instruction, not a BOSC instruction. Otherwise, the interrupt level is turned off, setting up the possibility that another interrupt could occur on the same level thus destroying the return address to the user from the ILS.
3. A LIBF or CALL to another subroutine from the user's error subroutine can cause a recurrent-entry problem. If that subroutine is already in use when the interrupt occurs, the user's new LIBF or CALL destroys the original return address and disrupts operation of the called subroutine.
4. A LIBF or CALL to another ISS can cause an endless loop if the new I/O device operates on the same or lower priority interrupt level than the device that caused the error.

NOTE: A call to WRTY0 to type an error message can be made only if the user does not wait for the completion of typing or for operator intervention before returning to the ISS. A test loop on level 4 (typewriter) or a WAIT loop will both block the clearing of the level that caused the interrupt to the user's error subroutine.

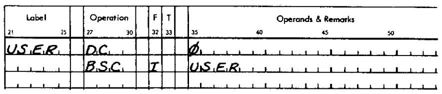
5. The user should have a separate error subroutine for each device to prevent errors on several devices (on different levels) from causing recurrent-entry problems in the user's error subroutine.

NOTE: The error codes in the Accumulator may not distinguish between ISSs, as the preoperative error codes do.

Since the ILS saves Index Register 1 as part of its interrupt processing, the user's error subroutine can also use this index register without saving and restoring it. However, the user cannot depend on the contents of Index Register 1 unless he initializes it as part of his error subroutine. The DM2 ILSs also save Index Register 2.

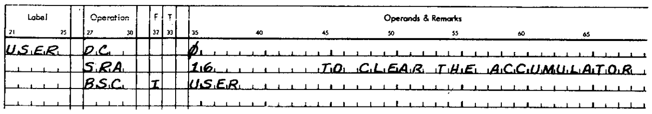
Programming Techniques - User's Error Subroutine Exits. Some programming techniques that can be used in conjunction with the ISS error exit are as follows:

1. To try the operation again:



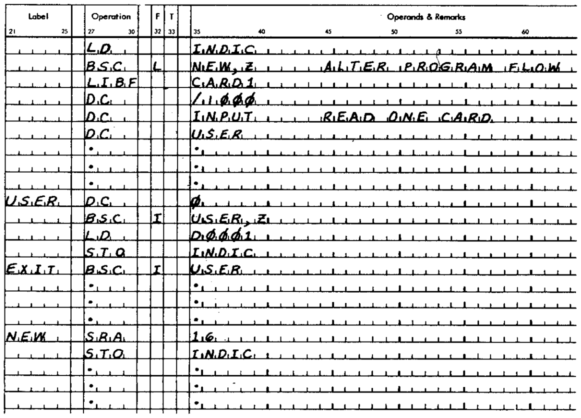
```
Label    Operation  F  T  Operands & Remarks
21   25  27    30  32 33  35   40   45   50
USER     DC               Ø
         BSC        I     USER
```

2. To terminate the operation:



```
Label    Operation  F  T  Operands & Remarks
21   25  27    30  32 33  35   40   45   50   55   60   65
USER     DC               Ø
         SRA              16        TO CLEAR THE ACCUMULATOR
         BSC        I     USER
```

3. To indicate that a condition ("last card" or "channel 9") was detected and that the normal program flow should be altered:



```
Label    Operation  F  T  Operands & Remarks
21   25  27    30  32 33  35   40   45   50   55   60
         LD               INDIC
         BSC        L     NEW,Z      ALTER PROGRAM FLOW
         LIBF             CARD1
         DC               /1ØØØ
         DC               INPUT      READ ONE CARD
         DC               USER
         .                .
         .                .
         .                .
USER     DC               Ø
         BSC        I     USER,Z
         LD               DØØØ1
         STO              INDIC
EXIT     BSC        I     USER
         .                .
         .                .
         .                .
NEW      SRA              16
         STO              INDIC
         .                .
         .                .
         .                .
```

BASIC ISS CALLING SEQUENCE

Each ISS described in this manual is entered via a calling sequence. These calling sequences follow a basic pattern. In order not to burden the reader with redundant descriptions, this section presents the basic calling sequences and describes those parameters that are common to most of the subroutines.

Basic Calling Sequence	
LIBF	Name
DC	Control parameter
DC	I/O area
DC	Error subroutine

The above calling sequence, with the parameters shown, is basic to most of the ISSs. Detailed descriptions of the above four parameters are omitted when the subroutines are described later in the manual. Unless otherwise specified, the subroutine returns control to the instruction immediately following the last parameter.

Name Parameter

Each subroutine has a symbolic name, that must be written in the LIBF statement exactly as listed in Tables 1 and 2.

Table 1. DM1 and C/PT System ISS Names

Device	Subroutine
1442 Card Read Punch	CARD0 or CARD1
Disk	DISK0, DISK1, or DISKN
1132 Printer	PRNT1
Keyboard/Console Printer	TYPE0
Console Printer	WRTY0
1134/1055 Paper Tape	PAPT1 or PAPTN
1627 Plotter	PLOT1
Synchr. Comm. Adapter	SCAT1, SCAT2, or SCAT3

For some devices more than one subroutine is available, although only one can be selected for use in any one program (including called subroutines).

NAME0. The NAME0 subroutine is the shortest and least complicated. The NAME0 version is the standard subroutine for the 1442, 2501, and Console Printer/Keyboard. The NAME0 version of the Disk routine (DISK0) can be used if transfer of data is 320 words or less (DM1 or C/PT system only).

NAME1. The NAME1 version is the standard subroutine for the disk, 1132, 1403, 2501, 1134/1055, 1231, and 1627. It may be used if a user error exit is needed rather than the internal looping and retries by the NAME0 subroutine.

Table 2. DM1 and/or DM2 System ISS Names

Device	Subroutine
1442 Card Read Punch	CARDZ, CARD0, or CARD1
2501 Card Reader (DM2 only)	READZ, READ0, or READ1
1442 Card Punch	PNCHZ, PNCH0, or PNCH1
Disk	DISKZ, DISK1, or DISKN
1132 Printer	PRNTZ, PRNT1, or PRNT2
1403 Printer (DM2 only)	PRNTZ or PRNT3
Keyboard/Console Printer	TYPEZ or TYPE0
Console Printer	WRTYZ or WRTY0
1134/1055 Paper Tape Reader Punch	PAPTZ, PAPT1, PAPTN, or PAPTX (DM2 only)
1627 Plotter	PLOT1 or PLOTX
1231 Optical Mark Page Reader (DM2 only)	OMPR1
Synchr. Comm. Adapter	SCAT1, SCAT2, or SCAT3

NAMEN. The NAMEN version is available to operate the 1134/1055 Paper Tape Reader and Punch simultaneously and to minimize extra disk revolutions when transferring more than 320 words to or from the disk. The NAMEN subroutine offers more options than the NAME1 subroutine. In DM2, it also provides the ability to operate all five disk drives simultaneously.

NAMEZ. The NAMEZ version is designed for use in an error free environment. It provides no pre-operative parameter checking. The FORTRAN formatting subroutines use these ISSs but they do not use the calling sequence listed below (see Subroutines Used by FORTRAN).

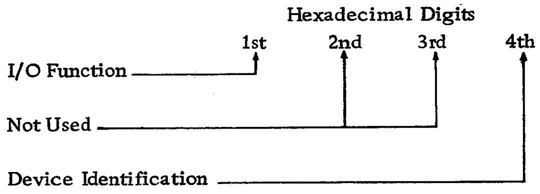
PRNT2. The PRNT2 version is used when the 1132 is used with the SCA.

PRNT3. The PRNT3 version is used with the 1403.

Control Parameter

The control parameter, in the form of four hexadecimal digits, conveys necessary control data to the ISSs by specifying the desired function (read, write, etc.), the device identification, and similar control information. Most subroutines do not use all four digits.

A typical control parameter is illustrated below.



Since the I/O function and device identification digits are used in most subroutines, a description of the purpose of each is given here.

I/O Function

The function digit in the calling sequence specifies which I/O operation the user is requesting. Three of these functions, read, write, and test are used in most subroutines.

Read. The read function causes a specified amount of data to be read from an input device and placed in a specified input area. Depending upon the device, an interrupt signals the subroutine either when the next character is ready or when all requested data has been read. When the specified number of characters has been read, the subroutine becomes available for another call to that device.

Write. The write function causes a specified amount of data from the user's output area to be written, i.e., printed or punched, by an output device. As with the read function, an interrupt signals the subroutine when the device can accept another character, or when all characters have been written. When the specified number of characters has been written, the subroutine becomes available for another call to that device.

Test. The test function causes a check to be made as to the status of a previous operation initiated on an I/O device. If the previous operation has been completed, the subroutine branches to the LIBF +3 core location; if the previous operation has not been completed, the subroutine branches to the LIBF +2 core location. The test function is illustrated below:

```
          LIBF      Name

LIBF +1   DC        Control Parameter
                      (specifying Test function)

LIBF +2   OP Code   xxxx....

LIBF +3   OP Code   xxxx....
```

NOTE: Specifying the test function requires two statements (one LIBF and one DC), except in Disk subroutines, where three statements are required.

The test function is useful in situations in which input data has been requested, but no processing can be done until the data is available.

Device Identification

This digit should be zero except for the Test function with the PAPTN (paper tape) subroutine.

NOTE: For all disk subroutines, this digit appears in the I/O area rather than in the control parameter.

I/O Area Parameter

The I/O area for a particular operation consists of one table of control information and data. This table is composed of a data area preceded by a control word (two control words for disk operations) that specifies how much data is to be transferred. The area parameter in the calling sequence is the address (symbolic or actual) of the first control word that precedes the data area.

The control word contains a word count referring to the number of data words in the table. It is important to remember that the number of words in the table is not always the number of characters to be read or written, because some codes pack two characters per word. The disk subroutines require a second control word, which is described along with those subroutines.

Error Parameter

The error parameter is the means by which an ISS can give temporary control to the user in the event of conditions such as error, last card, etc. This parameter is not required for the NAME0 subroutines for the 2501, 1442, Console Printer, or Keyboard. The instruction sequence for setting up the error subroutine is shown below.

```
        LIBF    NAME
         .       .
        DC      ERROR (error parameter)
         .       .
ERROR   DC      0        (return link)
         .       .
         .       .       (error routine)
        BSC I   ERROR (branch to return link)
```

The return link is the address in the related ISS to which control must be returned upon completion of the error subroutine. The link is inserted in location ERROR by a BSI from the ISS when the subroutine branches to the error subroutine.

The types of errors that cause a branch to the error address are listed in Appendix B.

NOTE: The user's error subroutine is executed as part of the interrupt response handling. The interrupt level is still on and remains on until control is returned to the ISS (see General Error Handling Procedures).

ASSIGNMENT OF CORE STORAGE LOCATIONS (DM1 AND C/PT SYSTEM)

The portion of core storage used by the ISS and ILS subroutines is defined below. Care should be used in altering any of these locations (see Figure 3).

The areas illustrated in Figure 3 are described below.

Interrupt Branch Addresses

ILS Subroutines. When required, the address of ILS00 is always stored in location 8, ILS01 in location 9,..., ILS05 in location 13.

Interrupt Trap. The address of the interrupt trap is stored in any location for which no ILS is loaded.

1132 Printer

This area is used by 1132 Printer.

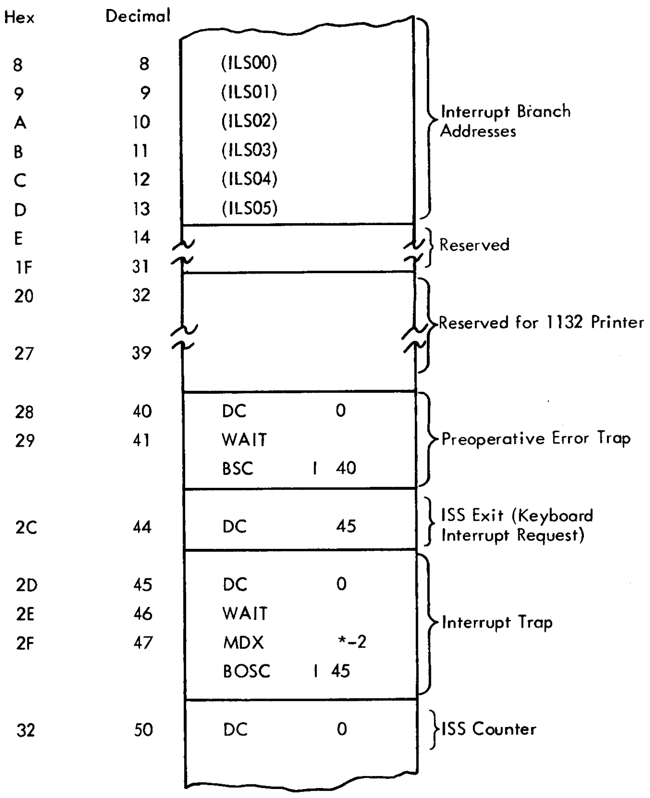
Pre-operative Error Trap

This exit is used whenever a pre-operative error (illegal LIBF or device not ready) is detected by an ISS.

To retry the call, press START.

ISS Exit

The ISS exit results from pressing the Keyboard Interrupt Request key. The TYPE0 and WRTY0 subroutines execute a BSI I 2C whenever a keyboard operator request is detected. Note that interrupt level 4 is still on.



Hex	Decimal			
8	8	(ILS00)		Interrupt Branch Addresses
9	9	(ILS01)		
A	10	(ILS02)		
B	11	(ILS03)		
C	12	(ILS04)		
D	13	(ILS05)		
E	14			Reserved
1F	31			
20	32			Reserved for 1132 Printer
27	39			
28	40	DC	0	Preoperative Error Trap
29	41	WAIT		
		BSC	I 40	
2C	44	DC	45	ISS Exit (Keyboard Interrupt Request)
2D	45	DC	0	Interrupt Trap
2E	46	WAIT		
2F	47	MDX	*-2	
		BOSC	I 45	
32	50	DC	0	ISS Counter

● Figure 3. ISS and ILS Core Locations for the DM1 and C/PT System

The user-written subroutine must return to the TYPE0 or WRTY0 subroutine in order to allow interrupts of equal or lower priority to occur. Also a call executed to any subroutine might cause a recurrent-entry problem unless the user can guarantee that the subroutine was not in use when the keyboard interrupt occurred.

Location 2C is initialized with the address of the interrupt trap in case the user fails to store an address in the interrupt trap to process Keyboard operator requests.

Interrupt Trap

This trap is entered when an interrupt occurs for which there is no ILS and/or no ISS assigned to the pertinent bit in the Interrupt Level Status Word (ILSW).

- Interrupts of higher priority will be processed before the system finally halts with the IAR displaying /002F.

ISS Counter

The ISS counter is incremented by +1 every time an ISS initiates an interrupt-causing I/O operation and is decremented by +1 when the operation is complete. A positive value in this location indicates the number of interrupt(s) pending. This counter should never be negative.

ASSIGNMENT OF CORE STORAGE LOCATIONS (DM2 SYSTEM)

The portion of core storage used by the ISS and ILS subroutines is defined below. Care should be used in altering any of these locations (see Figure 4).

The areas illustrated in Figure 4 are described below.

Interrupt Branch Addresses

ILS Subroutines. The address of ILS00 is always stored in location 8, ILS01 in location 9,. . . , ILS05 in location 13.

Interrupt Trap. The address of the Program Stop Key trap ($STOP) is stored in any location for which no ILS is loaded.

Reserved Areas

These locations are reserved for the DM2 system.

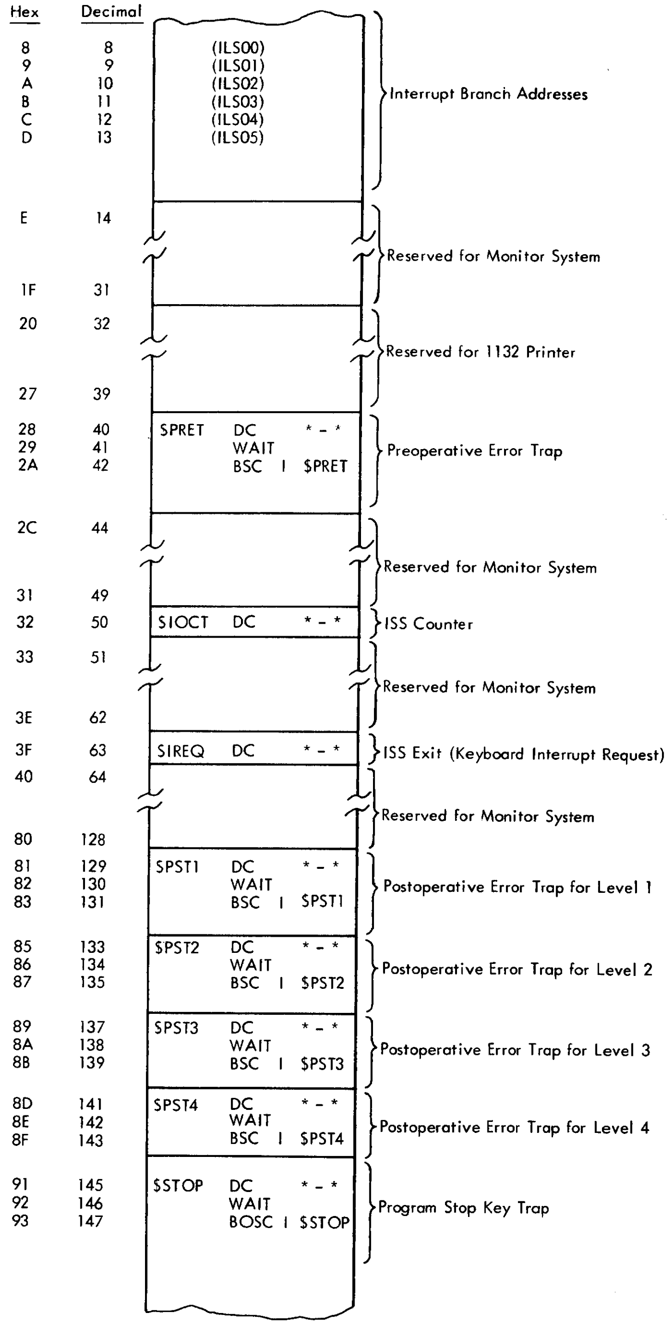
1132 Printer

This area is used by 1132 Printer.

Pre-operative Error Trap

This exit is used whenever a pre-operative error (illegal LIBF or device not ready) is detected by an ISS.

To retry the call, press START.



● Figure 4. ISS and ILS Core Locations for the DM2 System

ISS Counter

The ISS counter is incremented by +1 every time an ISS initiates an interrupt-causing I/O operation and is decremented by +1 when the operation is complete.

A positive value in this location indicates the number of interrupt(s) pending. This counter should never be negative.

ISS Exit

The ISS exit results from pressing the Keyboard Interrupt Request key. The TYPE0 and WRTY0 subroutines execute a BSI I $IREQ whenever a Keyboard operator request is detected. Note that interrupt level 4 is still on.

The user-written subroutine must return to the TYPE0 or WRTY0 subroutine in order to allow interrupts of equal or lower priority to occur. Also a call executed to any subroutine might cause a recurrent-entry problem unless the user can guarantee that the subroutine was not in use when the keyboard interrupt occurred.

$IREQ is initialized with the address of the system dump entry point during JOB processing. This allows the user to terminate the job by pressing the Interrupt Request key (INT REQ).

Post-operative Error Traps

These traps are entered when a device not ready condition is detected prior to the initiation of an I/O operation in the interrupt response portion of an ISS subroutine. Each interrupt level (1-4) has its own post-operative error trap. The system will WAIT with the IAR displaying the address of $PST1, $PST2, $PST3, or $PST4, depending on the interrupt level of the device.

DESCRIPTIONS OF INTERRUPT SERVICE SUBROUTINES

Note that the subroutine READ0, READ1, PNCH0, PNCH1, PRNT3, and OMPR1 are available only with the DM2 system.

1442 CARD READ PUNCH SUBROUTINE (CARD0 AND CARD1)

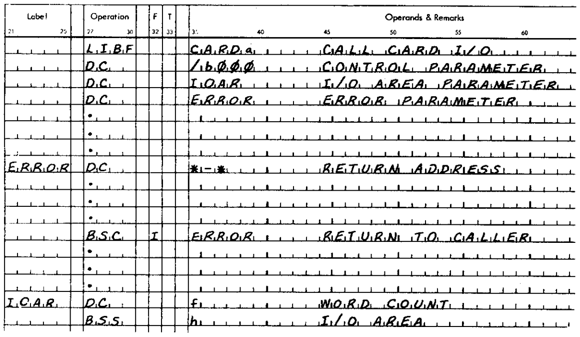
The card subroutines perform all I/O functions relative to the IBM 1442 Card Read Punch: read, punch, feed, and stacker select.

CARD0 Subroutine. The CARD0 subroutine is shorter and less complicated than CARD1 and is the standard subroutine for the 1442.

CARD0 can be used if the error parameter is not needed. When an error occurs, the subroutine loops (DM1 and C/PT system) or WAITs at $PST4+1 (DM2 system), until the operator takes corrective action. Last card conditions cause pre-operative not-ready exits.

CARD1 Subroutine. The CARD1 subroutine can be used for the Card Read Punch if a user error exit is needed, rather than the error procedures of the CARD0 subroutine.

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
21  25 27   30    32 33 35   40   45   50   55   60
       LIBF             CARDa     CALL CARD I/O
       DC               /bØØØ     CONTROL PARAMETER
       DC               IOAR      I/O AREA PARAMETER
       DC               ERROR     ERROR PARAMETER
       .
       .
       .
ERROR  DC               *-*       RETURN ADDRESS
       .
       .
       .
       BSC        I     ERROR     RETURN TO CALLER
       .
       .
       .
IOAR   DC               f         WORD COUNT
       BSS              h         I/O AREA
```

where

a is 0 or 1,

b is the I/O function digit,

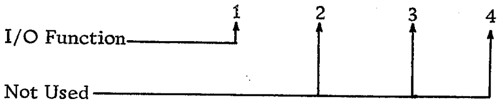
f is the number of columns to be read from or punched into the card,

h is the length of the I/O area. h must be equal to or greater than f.

The calling sequence parameters 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 particular operation to be performed on the 1442 Card Read Punch. The functions, associated digital values, and required parameters are listed and described below.

Function	Digital Value	Required Parameters*
Test	0	Control
Read	1	Control, I/O Area, Error**
Punch	2	Control, I/O Area, Error**
Feed	3	Control, Error**
Stacker Select	4	Control

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

**Error parameter not required for CARD0.

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

Read. Reads one card and transfers a specified number of columns of data to the user's input area. The number of columns read (1-80) is specified by the user in the first location of the I/O area. The subroutine clears the remainder of the I/O area and stores a 1 in bit position 15 of each word, initiates the card operation, and returns control to the user's program. When each column is ready to be read, a column interrupt occurs. This permits the card subroutine to read the data from that column into the user's input area (clearing bit 15), after which the user's program is again resumed. This sequence of events is repeated until the requested number of columns has been read, after which the remaining column interrupts are cleared (no data read).

When an operation complete interrupt occurs, the card subroutine checks for errors, informs the user if an error occurred (CARD1 only), and sets up to terminate (CARD1 only) or retry the operation.

The data in the user's input area is in a code similar to IBM Card Code format; that is, each 12-bit column image is left-justified in one 16-bit word.

Punch. Punches into one card the number of columns of data specified by the word count found at the beginning of the user's output area. The punch operation is similar to the read operation. As each column comes under the punch dies, a column interrupt occurs; the card subroutine transfers a word from the user's output area to the punch and then returns control to the user's program.

This sequence is repeated until the requested number of columns has been punched, after which an Operation Complete interrupt occurs. At this time the card subroutine checks for errors, informs the user if an error occurred (CARD1 only), and sets up to terminate (CARD1 only) or retry the operation. The character punched is the image of the leftmost 12 bits in the word.

Feed. Initiates a card feed cycle. This advances all cards in the machine to the next station, i.e., a card at the punch station advances to the stacker, a card at the read station advances to the punch station, and a card in the hopper advances to the read station. No data is read or punched as a result of a feed operation and no column interrupts occur.

When the card advance is complete, an Operation Complete interrupt occurs. At this time the card subroutine checks for errors, informs the user if an error occurred (CARD1 only), and sets up to terminate (CARD1 only) or retry the operation.

Stacker Select. Selects stacker 2 for the card currently at the punch station. After the card passes the punch station, it is directed to stacker 2.

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 columns of data to be read or punched, always starting the count at column 1.

Error Parameter

CARD0. CARD0 has no error parameter. If an error is detected while an operation complete interrupt is being processed, the subroutine loops (DM1 or C/PT system) or WAITs at $PST4 (DM2) with interrupt level 4 on, waiting for operator intervention. When the condition has been corrected, the 1442 made ready, and PROGRAM START pressed, the subroutine retries the operation.

CARD1. CARD1 has an error parameter. If an error is detected, the user can request the subroutine to terminate (clear subroutine-busy indicator and the interrupt level) or to loop (DM1 or C/PT system) or WAIT at $PST4 (DM2) waiting for operator intervention (interrupt level 4 on). (See Basic Calling Sequence.)

Protection of Input Data

Since the CARD subroutines read data directly into the user's I/O area, the user can manipulate the data before the entire card has been processed. This procedure is inherently dangerous because, if an error occurs, the data may be in error and error recovery procedures will cause the operation to be tried again. The exit via the error parameter is the only method of informing the user that an error has occurred.

Therefore, do not manipulate data before the entire card has been processed when using CARD0.

When using CARD1, the following precautions should be taken:

- Do not store converted data back into the read-in area.
- Do not take any irretrievable action based on the data until the card has been read correctly; i.e., be prepared to convert the data or perform the calculations a second time.
- When data manipulation is complete, check the user-assigned error indicator that is set when a branch to the user-written error subroutine occurs. The data conversion or calculations can then be reinitiated, if necessary.

Last Card

A read or feed function requested after the last card has been detected will eject that card and cause a branch to the pre-operative error exit (location 29). A punch function will punch and then eject that card with a normal exit. Therefore,to eject the last card without causing a pre-operative error exit,request a punch function with a word count of one and a blank in the data field.

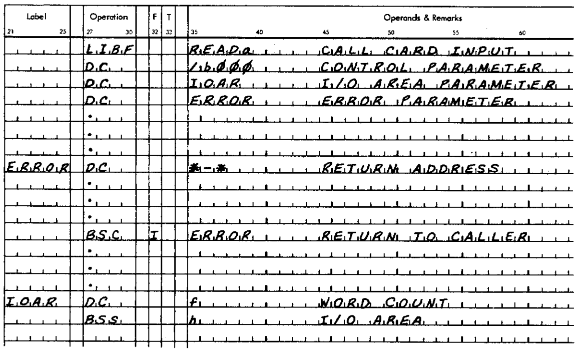
2501 CARD READER SUBROUTINES (READ0 AND READ1)

These card subroutines, available only with the Monitor system, perform read and test functions relative to the IBM 2501 card reader.

READ0 Subroutine. READ0 is shorter than READ1, provides no error parameter, and is the standard subroutine for operation of the 2501 card reader. On an error, READ0 branches to $PST4, and the system WAITs for operator intervention. The last card condition causes a branch to $PRET.

READ1 Subroutine. READ1 is used for operation of the 2501 card reader if a user error exit is required.

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
21  25 27    30  32 32 35   40     45   50   55   60
       LIBF            READa       CALL CARD INPUT
       DC              /b000       CONTROL PARAMETER
       DC              IOAR        I/O AREA PARAMETER
       DC              ERROR       ERROR PARAMETER
       .
       .
       .
ERROR  DC              *-*         RETURN ADDRESS
       .
       .
       .
       BSC        I    ERROR       RETURN TO CALLER
       .
       .
       .
IOAR   DC              f           WORD COUNT
       BSS             h           I/O AREA
```

where

a is 0 or 1,

b is the I/O function digit,

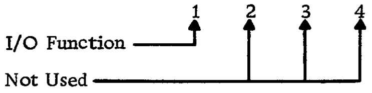
f is the number of columns to be read from the card,

h is the length of the I/O area. h must be equal to or greater than f.

The calling sequence parameters 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 particular operation to be performed on the 2501 Card Reader. The functions, associated digital values, and required parameters are listed and described below.

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

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

**The error parameter is not required for READ0.

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

Read. Reads one card and transfers a specified number of columns of data to the user's input area. The number of columns read (1-80) is specified by the user in the first location of the input area. The subroutine initiates the read function and returns control to the user's program.

When an operation complete interrupt occurs, the card subroutine checks for errors. If an error occurred, READ0 exits to $PST4; READ1 informs the user of the error and sets up to terminate or retry the operation.

The data in the user's input area is in IBM Card Code format; that is, each 12-bit column image is left-justified in one 16-bit word.

There is no separate feed function. However, a feed can be obtained by a read function with a word count of zero.

I/O Area Parameter

The I/O area parameter is the label on the control word that precedes the user's input area. The control word consists of a word count that specifies the number of columns of data to be read, always starting with column 1.

Error Parameter

READ0. READ0 has no error parameter. If an error is detected while an operation complete interrupt is being processed, the subroutine branches to $PST4, with interrupt level 4 on, waiting for operator intervention. When the condition has been corrected, the 2501 made ready, and PROGRAM START pressed, the subroutine attempts the operation again.

READ1. READ1 has an error parameter. If an error is detected, the user can request the subroutine to terminate (that is, to clear the subroutine's busy indicator and turn off the interrupt level) or retry. Prior to a retry, the subroutine checks to see if the unit is ready. If the unit is not ready, the subroutine branches to $PST4 with interrupt level 4 on, waiting for operator intervention.

Last Card

A read function requested after the last card has been fed from the hopper causes an exit to $PRET. When the reader is made ready and the PROGRAM START key pressed, the last card is read and fed into the stacker.

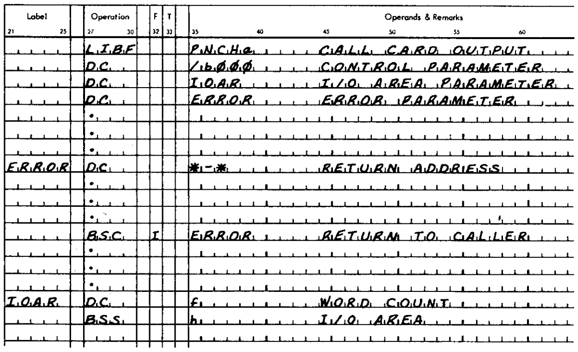
1442 CARD PUNCH SUBROUTINES (PNCH0 AND PNCH1)

These card subroutines, available only with the Monitor system, perform all I/O functions relative to the IBM 1442-5 Card Punch, that is, punch and feed. These subroutines may also be used with the 1442-6 or 1442-7 Card Read Punch for punch and feed functions.

PNCH0. The PNCH0 subroutine is shorter than PNCH1, provides no error parameter, and is the standard subroutine for operation of the 1442 card punch. On an error, PNCH0 branches to $PST4, and the system WAITs for operator intervention. The last card condition causes a branch to $PRET.

PNCH1. PNCH1 can be used for operation of the 1442 card punch if a user error exit is desired.

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
21  25 27  30    32 33 35   40   45   50   55   60
       LIBF            PNCHa     CALL CARD OUTPUT
       DC              /b000     CONTROL PARAMETER
       DC              IOAR      I/O AREA PARAMETER
       DC              ERROR     ERROR PARAMETER
        .
        .
        .
ERROR  DC              *-*       RETURN ADDRESS
        .
        .
        .
       BSC        I    ERROR     RETURN TO CALLER
        .
        .
        .
IOAR   DC              f         WORD COUNT
       BSS             h         I/O AREA
```

where

a is 0 or 1,

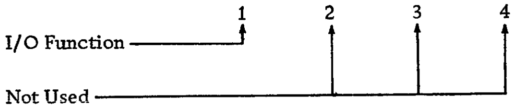
b is the I/O function digit,

f is the number of columns to be punched into the card, h is the length of the I/O area. h must be equal to or greater than f.

The calling sequence parameters 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 particular operation to be performed on the 1442 Card Punch. The functions, associated digital values, and required parameters are listed and described below.

Function	Digital Value	Required Parameters*
Test	0	Control
Punch	2	Control I/O Area, Error**
Feed	3	Control, Error**

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

**The error parameter is not required for PNCH0.

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

Punch. Punches into one card the number of columns of data specified by the word count found at the beginning of the user's output area. As each column comes under the punch dies, a column interrupt occurs, the subroutine transfers a word from the user's output area to the punch, and then returns control to the user's program. The character punched is the image of the leftmost 12 bits in the word.

This sequence is repeated until the requested number of columns has been punched, after which an Operation Complete interrupt occurs. At this time the card subroutine checks for errors. If an error occurred, PNCH0 exits to $PST4; PNCH1 informs the user of the error and sets up to terminate or retry the operation.

Feed. Initiates a card feed cycle. This function advances all cards in the machine to the next station; that is, a card at the punch station advances to the stacker, a card at the read station advances to the punch station, and a card in the hopper advances to the read station. No data is punched as a result of a feed function and no column interrupts occur.

When the card advance is complete, an operation complete interrupt occurs. At this time the card subroutine checks for errors. If an error occurred, PNCH0 exits to $PST4; PNCH1 informs the user of the error and sets up to terminate or retry the operation.

I/O Area Parameter

The I/O area parameter is the label of the control word that precedes the user's output area. The control word consists of a word count that specifies the number of columns of data to be punched, always starting with column 1.

Error Parameter

PNCH0. PNCH0 has no error parameter. If an error is detected while an operation complete interrupt is being processed, the subroutine branches to $PST4 with interrupt level 4 on, waiting for operator intervention. When the condition has been corrected, the 1442 made ready, and PROGRAM START pressed, the subroutine retries the operation.

PNCH1. PNCH1 has an error parameter. If an error is detected, the user can request the subroutine to terminate (that is, to clear the subroutine busy indicator and turn off the interrupt level) or retry. Prior to a retry, the subroutine checks to see if the unit is ready. If the unit is not ready, the subroutine branches to $PST4, with interrupt level 4 on, waiting for operator intervention.

DISK SUBROUTINES (DM1 AND C/PT SYSTEM)

The disk subroutines perform all reading and writing of data relative to disk storage. This includes the major functions: seek, read, and write, in conjunction with readback check, file protection, and defective cylinder handling.

DISK0. The DISK0 subroutine is the shortest version of the disk subroutine and can be used if not more than 320 words are to be read or written at one time.

DISK1. The DISK1 version is the standard subroutine for the disk and allows more than 320 words to be read or written; however, a full disk revolution might occur between sectors. DISK1 requires more core storage than DISK0.

DISKN. The DISKN subroutine minimizes extra disk revolutions in transferring more than 320 words. The DISKN subroutine requires more core storage than DISK1.

The major difference between DISK1 and DISKN is the ability of DISKN to read or write consecutive sectors on the disk without taking an extra revolution. If a full sector is written, the time in which the I/O command must be given varies. DISKN is programmed so that it can "make" the sector gap the majority of the time; DISK1 approximately 50 percent of the time.

All three disk subroutines have the same error handling procedures. In DM1, the disk subroutines are part of the Supervisor and as such are not stored in the Subroutine Library. Consequently, these subroutines have no LET entries.

Sector Numbering and File Protection

In the interest of providing disk features permitting versatile and orderly control of disk operations, programming conventions have been adopted concerning sector numbering, file protection, and defective cylinder handling. Successful use of the disk subroutines can be expected only if user programs are built within the framework of these conventions.

The primary concern behind these conventions is the safety of data recorded on the disk. To this end, the file-protection scheme plays a major role, but does so in a manner that is dependent upon the sector-numbering technique. The latter contributes to data safety by allowing the disk subroutine to verify the correct positioning of the access arm before it actually performs a write operation. This verification requires that sector identification be prerecorded on each sector and that subsequent writing to the disk be done in a manner that preserves the existing identification. The disk subroutines have been organized to comply with these requirements.

Sector Numbering

The details of the numbering scheme are as follows: each disk sector is assigned an address from the sequence 0, 1,...,1623, corresponding to the sector position in the ascending sequence of cylinder and sector numbers from cylinder 0 (outermost) sector 0, through cylinder 202 (innermost) sector 7. The user can address cylinders 0 through 199. The remaining three cylinders are reserved for defective-cylinder handling.

Each cylinder contains eight sectors and each sector contains 321 words. The sector address is recorded in the first word of each sector and occupies the rightmost eleven bit positions. Of these eleven positions, the three low-order positions identify the sector (0-7) within the cylinder. Utilization of this first word for identification purposes reduces the per sector availability of data words to 320; therefore, transmission of full sectors of data is performed in units of this amount. The sector addresses must be initially recorded on the disk by the user and are thereafter rewritten by the disk subroutines as each sector is written (see Disk Initialization).

File Protection

File protection is provided to guard against the inadvertent destruction of previously recorded data. By having the write functions (except write immediate) uniformly test for the file-protect status of sectors that they are about to write, this control can be achieved.

This convention is implemented by assigning a file-protected area to each disk. The address of the first unprotected sector (0000-1623) on each disk is stored within the disk subroutine (C/PT) or in COMMA (DM1). Every sector below this one is file-protected, i.e., no writing is permitted below this address which in DM1 is the address of the first sector of Working Storage.

Defective Cylinder Handling

A defective sector is one in which, after ten retries, a successful writing operation cannot be completed. A cylinder having one or more defective sectors is defined as a defective cylinder. The disk subroutines can operate when as many as three cylinders are defective.

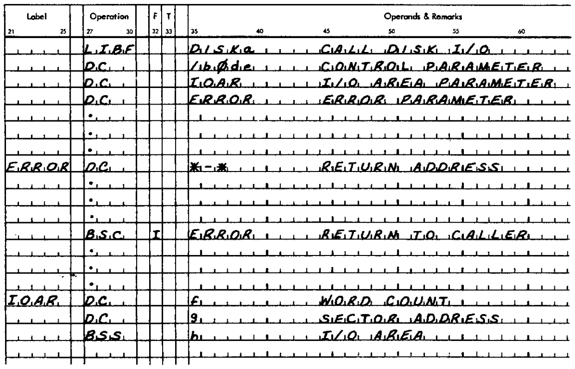
Since there are 203 cylinders on each disk, the subroutine can "overflow" the normally used 200 cylinders when defective cylinders are encountered (see Effective Address Calculation).

The address of each defective cylinder is stored within the disk subroutines by the user (C/PT) or both on sector 0 and in COMMA (DM1). (See Disk Initialization .)

In the C/PT system, if a cylinder becomes defective during an operation, the user can move the data in that cylinder and each higher-addressed cylinder into the next higher-addressed cylinders. Then the address of the new defective cylinder can be stored in DISKx +16, +17, or +18 and normal operation continued. Thus the user should not store the new defective cylinder address in DISKx and then continue normally because the effective sector address computation then yields a sector address eight higher than is desired (see Effective Address Calculation).

If there are no defective cylinders, all three words in the defective cylinder table contain /0658. If, for example, only sector 0009 is defective, the table would contain /0008 (cylinder 1), /0658, and /0658.

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
       LIBF            DISKa        CALL DISK I/O
       DC              /bØde        CONTROL PARAMETER
       DC              IOAR         I/O AREA PARAMETER
       DC              ERROR        ERROR PARAMETER
       .
       .
       .
ERROR  DC              *-*          RETURN ADDRESS
       .
       .
       .
       BSC        I    ERROR        RETURN TO CALLER
       .
       .
       .
IOAR   DC              f            WORD COUNT
       DC              g            SECTOR ADDRESS
       BSS             h            I/O AREA
```

where

a is 0, 1, or N.

b is the I/O function digit,

d is the Seek option digit,

e is the Displacement option digit,

f is the number of words to be transferred to or from the disk,

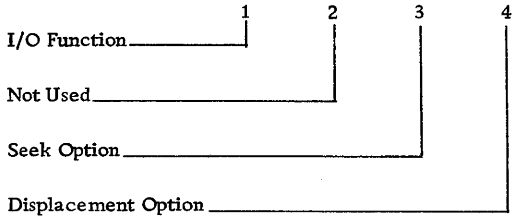
g is the sector address at which the transfer is to begin,

h is the length of the I/O area. h must be equal to or greater than f.

The calling sequence parameters 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 Disk Storage. The functions, their associated digital value, and the required parameters are listed and described below.

Function	Digital Value	Required Parameters*
Test	0	Control, I/O Area
Read	1	Control, I/O Area, Error
Write without RBC	2	Control, I/O Area, Error
Write with RBC	3	Control, I/O Area, Error
Write Immediate	4	Control, I/O Area
Seek	5	Control, I/O Area, Error

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

Test. Branches to LIBF +3 if the previous operation has not been completed, to LIBF +4 if the previous operation has been completed.

NOTE: This function requires two parameters.

Read. Positions the access arm and reads data into the user's I/O area until the specified number of words has been transmitted. Although sector identification words are read and checked for agreement with expected values, they are neither transmitted to the I/O data area nor counted in the number of words transferred.

If, during the reading of a sector, a read check occurs, up to ten retries are attempted. If the error persists, the function is temporarily discontinued, an error code is placed in the Accumulator, the address of the faulty sector is placed in the Extension, and an exit is made to the error subroutine specified by the error parameter.

Upon return from the error subroutine, that sector operation is reinitiated or the function is terminated, depending on whether the Accumulator is non-zero or zero.

Write With Readback Check. This function first checks whether or not the specified sector address is in a file-protected area. If it is, the subroutine places the appropriate error code in the Accumulator and exits to location 28.

If the specified sector address is not in a file-protected area, the subroutine positions the access arm and writes the contents of the indicated I/O data area into consecutive disk sectors. Writing

begins at the designated sector and continues until the specified number of words has been transmitted.
A readback check is performed on the data written.

If any errors are detected, the operation is retried up to ten times. If the function still cannot be accomplished, an appropriate error code is placed in the Accumulator, the address of the faulty sector is placed in the extension, and an exit is made to the error subroutine designated in the Error parameter.

Upon return from this error subroutine, the same sector operation is reinitiated or the function is terminated depending upon whether the contents of the Accumulator is non-zero or zero.

As each sector is written, the subroutine supplies the sector identification word. The identification word for the first sector is obtained from the I/O area, although it and subsequently generated identification words are not included in the word count. Writing less than 320 words clears the remainder of the sector to zero.

Write Without Readback Check. This function is the same as the function described above except that no readback check is performed.

Write Immediate. Writes data with no attempt to position the access arm, check for file-protect status, or check for errors. Writing begins at the sector number specified by the rightmost three bits of the sector address. This function is provided to fulfill the need for more rapid writing to the disk than is provided in the previously described write functions. Primary application will be found in the "streaming" of data to the disk for temporary bulk storage.

As each sector is written, the subroutine supplies the sector identification word. The identification word for the first sector is obtained from the I/O area, although it and subsequently generated identification words are not included in the word count. Writing less than 320 words clears the remainder of the sector to zero.

Seek. Initiates a seek as specified by the seek option digit. If any errors are detected, the operation is retried up to ten times.

Seek Option

If zero, a seek is executed to the cylinder whose sector address is in the disk I/O area control word; if non-zero, a seek is executed to the next cylinder toward the center of the disk, regardless of the sector address in the disk I/O area control word. This option is valid only when the seek function is specified.

The seek function requires that the user set up the normal I/O area parameter (see I/O Area Parameter) even though only the sector address in the I/O area is used. The I/O area control (first) word is ignored.

Displacement Option

If zero, the sector address word contains the absolute sector identification; if non-zero, the file protect address for the specified disk is added to bits 4-15 of the sector address word to generate the effective sector identification. The file-protect address is the sector identification of the first unprotected sector.

I/O Area Parameter

The I/O area parameter is the label of the first of two control words which precede the user's I/O area.

The first word contains a count of the number of data words that are to be transmitted during the disk operation. If the DISK1 or DISKN subroutine is used, this count need not be limited by sector or cylinder size, since these subroutines cross sector and cylinder boundaries, if necessary, in order to process the specified number of words. However, if the DISK0 subroutine is used, the count is limited to 320.

The second word contains the sector address where reading or writing is to begin. Bits 0-3 are used for device identification and must be zero. Bits 4-15 specify the sector address. Following the two control words is the user's data area.

Error Parameter

Refer to the section, Basic Calling Sequence.

Important Locations

The relative locations within the DISK0, DISK1, and DISKN subroutines are defined as follows:

DISKx +0 - entry point from calling transfer vector when LIBF DISKx is executed.
+2 - loader stores address of first location (in the calling transfer vector) assigned to DISKx
+4 - entry point from ILS handling Disk Storage interrupts.
+7 - area code for disk storage.
+8 - zero
+9 - zero
+10 - cylinder identification (bits 4-12) of the cylinder currently under the disk read/write heads (loaded as +202)
+11 - unused

+12 - reserved
+13 - sector address (bits 4-15) of the first non-file-protected sector for disk storage (loaded as 0)
+14 - reserved
+15 - reserved
+16 - sector address of the first defective cylinder for disk storage (loaded as +1624)
+17 - sector address of the second defective cylinder for disk storage (loaded as +1624)
+18 - sector address of the third defective cylinder for disk storage (loaded as +1624)

In the DM1 system, words DISKx +10 through DISKx +18 are stored in COMMA,not in DISKx.

Effective Address Calculation

An effective disk address is calculated as follows:

1. Start with the user-requested sector address (found in the sector address word of the I/O area).
2. If the displacement option (found in the control parameter) is non-zero, add the sector address of the first non-file-protected sector (found in DISKx +13 in C/PT System).

   NOTE: This starting address will cause a pre-operative error exit to location 41 if over 1599.

3. If the resulting address is equal to or greater than the sector address of the first defective cylinder (found in DISKx +16 in the C/PT System), add 8.
4. If the resulting address is equal to or greater than that of the second defective cylinder (found in DISKx +17 in the C/PT System), add 8 more.
5. If the resulting address is equal to or greater than that of the third defective cylinder (found in DISKx +18 in the C/PT System), add 8 more.

The address obtained from steps 1-5 is the effective sector address.

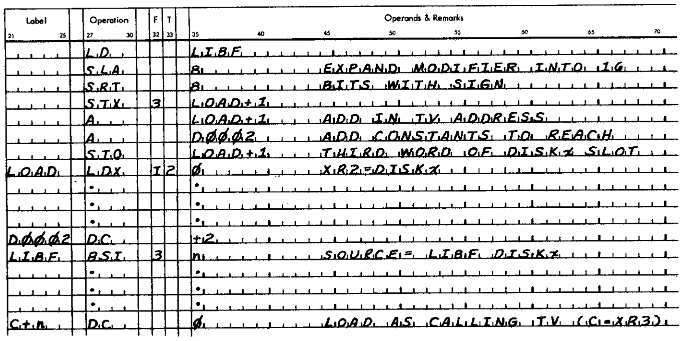
Disk Initialization

In the C/PT System, it is the user's responsibility to correctly load DISKx +13, +16, +17, and +18 at execution time and whenever a new disk is initialized. The following programs can be used to perform these functions.

Disk Pack Initialization Routine (DPIR). The functions of this program are to write sector addresses on a disk, to detect any defective cylinders, and to store defective cylinder information, file protect addresses, and a disk label in sector 0 of the disk. The operating procedures for DPIR are located in the publication IBM 1130 Card/Paper Tape Programming System Operator's Guide (Form C26-3629) and the IBM 1130 Disk Monitor System Reference Manual (Form C26-3750).

Set Pack Initialization Routine for C/PT System (SPIR0, SPIR1, and SPIRN). The function of these subroutines is to store defective cylinder information and the file protect address from sector 0 of the disk into the appropriate DISKx subroutine.

If the above subroutines are not used, the starting address of the DISKx routine can be loaded into an index register for easy use in reaching the specified locations:


Label
Operation
F
T
Operands & Remarks

```
      LD          LIBF
      SLA         8          EXPAND MODIFIER INTO 16
      SRT         8          BITS WITH SIGN
      STX    3    LOAD+1
      A           LOAD+1     ADD IN TV ADDRESS
      A           DØØØ2      ADD CONSTANTS TO REACH
      STO         LOAD+1     THIRD WORD OF DISKZ SLOT
LOAD  LDX    I2   Ø          XR2=DISKZ
      .           .
      .           .
      .           .
DØØØ2 DC          +2
LIBF  BSI    3    n          SOURCE= LIBF DISKZ
      .           .
      .           .
      .           .
C+n   DC          Ø          LOAD AS CALLING TV (C=XR3)
```

The SPIR is a special-purpose utility subroutine. It is not called by LIBF as are the other disk subroutines described in this section. SPIR0 must be used if DISK0 is called, SPIR1 if DISK1 is called, or SPIRN if DISKN is called.

NOTE: In no case should SPIR be used with the DM1 or DM2 System.

The SPIR reads sector 0000 from the disk and stores the first four words into the disk ISS that is in core. Therefore, the SPIR subroutine should be called before any calls are made to the disk ISS.

The calling sequence for SPIR is as follows:

```
CALL      SPIRx
DC        /0000
```

The four words read from sector 0000 are described under Disk Pack Initialization Routine in the publication IBM 1130 Card/Paper Tape Programming System Operator's Guide (Form C26-3629) and in the IBM 1130 Disk Monitor System Reference Manual (Form C26-3750).

DISK SUBROUTINES (DM2 SYSTEM)

All disk subroutines used by the Monitor system (including DISKZ) reside in the IBM System area on the monitor disk. The disk subroutines are stored in a special core image format in this area rather than in the System Library, since the Monitor system always requires a disk I/O subroutine. The required version is fetched by the Core Image Loader just prior to execution.

The disk subroutines used with the Monitor system are DISKZ, DISK1, and DISKN.

DISKZ. DISKZ is intended for use in a FORTRAN environment in which FORTRAN I/O is used. DISKZ makes no pre-operative parameter checks and offers no file protection. It is the shortest of the three disk I/O subroutines and requires a special calling sequence. (See DISKZ-Disk I/O Subroutine.)

DISK1. DISK1 is intended for use by Assembler language programs in which the core storage requirement is of more importance than the execution time. DISK1 is longer than DISKZ but is the shorter of the two subroutines intended for use in Assembler language programs (DISK1 and DISKN). However, DISK1 does not minimize extra disk revolutions when transferring more than 320 words.

DISKN. DISKN minimizes extra disk revolutions in transferring more than 320 words. It provides all the functions provided by DISK1 as well as the ability to operate all five drives simultaneously.

NOTE: Both DISK1 and DISKN can be specified on the Monitor XEQ record for use with FORTRAN programs. However, they offer no real advantage over DISKZ if they are called by the disk FORTRAN I/O subroutine.

One of the major differences among the disk subroutines is the ability to read or write consecutive sectors on the disk without taking extra revolutions. If full sectors are written, the time in which the I/O command must be given varies. DISKN is programmed so that transfers of more than 320 words are made with a minimum number of extra revolutions occurring between sectors.

DISK1 and DISKN have the same error handling procedures.

NOTE: In the DM2 system, the disk I/O subroutines are not stored in the System Library; consequently, they do not have LET entries.

Sector Numbering and File Protection

In the interest of providing disk features permitting versatile and orderly control of disk operations, programming conventions have been adopted concerning sector numbering, file protection, and defective sector handling. Successful use of disk I/O subroutines can be expected only if user programs are built within the framework of these conventions. The primary concern behind the conventions is the safety of data recorded on the disk. To this end, the file protection scheme plays a major role, but does so in a manner that is dependent upon the sector-numbering technique. The latter contributes to data safety by allowing the disk I/O subroutine to verify the correct positioning of the access arm before it actually performs a write operation. This verification requires that sector identification be prerecorded on each sector and that subsequent writing on the disk be done in a manner that preserves the existing identification. The disk I/O subroutines support these requirements.

Sector Numbering

Each disk sector is assigned an address from the sequence 0, 1, ..., 1623, corresponding to the sector position in the ascending sequence of cylinder and sector numbers from cylinder 0, sector 0 (outermost), through cylinder 202, sector 7 (innermost). The user can address cylinders 0 through 199. The remaining three cylinders are reserved for defective cylinder handling.

Each cylinder contains eight sectors and each sector contains 321 words, counting the sector address. The sector address is recorded in the first word of each sector and occupies the rightmost eleven bit positions. Of these eleven positions, the three low-order positions identify the sector (0-7) within the cylinder. Utilization of this first word for identification purposes reduces the per sector availability of data words to 320; therefore, transmission of full sectors of data is performed in increments of 320 words.

Sector addresses must be initially recorded on the disk by the user (via DISC or DCIP: see 1130 Monitor Programming and Operator's Guide (Form C26-3717)) and are thereafter rewritten by the disk I/O subroutines as each sector is written.

NOTE: Although not actually written on the disk, the logical drive code must be part of the sector address parameter (bits 1-3) which is stored in the second word of the I/O area. Bit 0 must always be zero.

File Protection

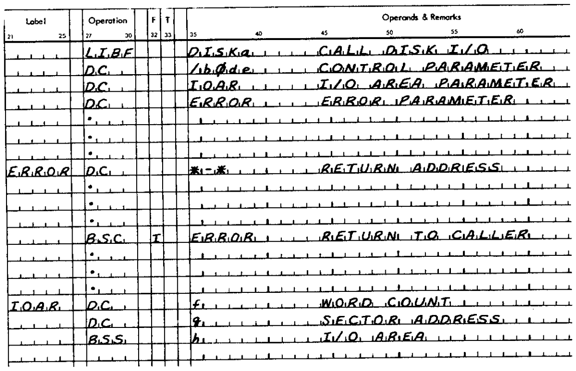
File protection is provided to prohibit the inadvertent destruction of previously recorded data. This control is achieved by having all write functions (except write immediate) test for the file-protection status of sectors they are about to write.

Each cartridge has a file-protect address in COMMA. This address is the address of the first unprotected sector, i.e., the address of the beginning of Working Storage. Every sector, from sector 0 up to the sector address maintained in COMMA, is file-protected. The initial assignment of the file-protect address is performed by the disk initialization program DCIP or DISC (see 1130 Monitor Programming and Operator's Guide (Form C26-3717)). Subsequent updating of the file-protect address is performed by the Monitor programs.

Defective Sector Handling

A defective sector is a sector on which a read or write function cannot be successfully completed during initialization of the cartridge. A cylinder having one or more defective sectors is defined as a defective cylinder. The disk I/O subroutines can accommodate as many as three defective cylinders per cartridge. Since there are 203 cylinders on each disk, the disk I/O subroutines can "overflow" the 200 cylinders normally used when defective cylinders are encountered (see Effective Address Calculation).

Calling Sequence



```
Label   Operation   F T   Operands & Remarks
21   25 27   30    32 33  35    40    45    50    55    60
        LIBF              DISKa           CALL DISK I/O
        DC                /bØde           CONTROL PARAMETER
        DC                IOAR            I/O AREA PARAMETER
        DC                ERROR           ERROR PARAMETER
        .
        .
        .
ERROR   DC                *-*             RETURN ADDRESS
        .
        .
        .
        BSC         I     ERROR           RETURN TO CALLER
        .
        .
        .
IOAR    DC                f               WORD COUNT
        DC                g               SECTOR ADDRESS
        BSS               h               I/O AREA
```

where

a is 1 or N. Note that LIBF DISK0 is equivalent to LIBF DISK1.

b is the I/O function digit,

d is the Seek option digit,

e is the Displacement option digit,

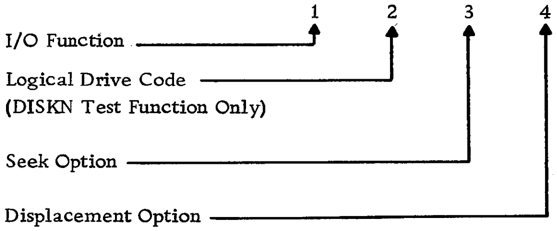
f is the number of words to be transferred to or from the disk,

g is the sector address, including the logical drive code, at which the transfer is to begin,

h is the length of the I/O area. h must be equal to or greater than f.

Control Parameter

This parameter consists of four hexadecimal digits, shown below:



I/O Function

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

Function	Digital Value	Required Parameters*
Test	0	Control, I/O Area
Read	1	Control, I/O Area, Error
Write without RBC	2	Control, I/O Area, Error
Write with RBC	3	Control, I/O Area, Error
Write Immediate	4	Control, I/O Area
Seek	5	Control, I/O Area, Error

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

Test. Branches to LIBF+3 if the previous operation on the drive has not been completed, to LIBF+4 if the previous operation has been completed.

NOTE: This function requires the I/O area parameter even though it is not used.

Read. Positions the access arm and reads data into the user's I/O area until the specified number of words has been transmitted. Although sector identification words are read and checked for agreement with expected values, they are neither transmitted to the I/O area nor counted in the number of words transferred.

If, during the reading of a sector, a read check occurs, up to 16 retries are attempted. If the error persists, the function is temporarily discontinued, an error code is placed in the Accumulator, the address of the faulty sector is placed in the Extension, and an exit is made to the error subroutine specified by the error parameter.

Upon return from the error subroutine, the operation is either reinitiated or terminated, depending on whether the Accumulator is non-zero or zero, respectively.

Write With Readback Check. Checks whether or not the specified sector address is in a file-protected area. If it is, the subroutine places the appropriate error code in the Accumulator and exits to $PRET.

If the specified sector address is not in a file-protected area, the subroutine positions the access arm and writes the contents of the indicated I/O area onto the disk. Writing begins at the designated sector and continues until the specified number of words have been transmitted. A readback check is performed on the data written.

If a partial sector (less than 320 words) is written, the remaining words of the sector are automatically set to zero.

If any errors are detected, the operation is retried up to 16 times. If the function cannot be accomplished, an appropriate error code is placed in the Accumulator, the address of the faulty sector is placed in the Extension, and an exit is made to the error subroutine designated by the error parameter.

Upon return from this error subroutine, the operation is either reinitiated or terminated, depending upon whether the Accumulator is non-zero, or zero, respectively.

As each sector is written, the subroutine supplies the sector identification word. The identification word for the first sector is obtained from the I/O area, although it and subsequently generated identification words are not included in the word count.

Write Without Readback Check. Functions the same as Write With Readback Check except that no readback check is performed.

Write Immediate. Writes data with no attempt to position the access arm, check for file-protect status, or check for errors. Writing begins at the sector number specified in the user's I/O area. This function provides more rapid writing to the disk than is provided in the previously described Write functions; it provides, for example, the ability to "stream" data to the disk for temporary bulk storage or to write addresses in Working Storage (see ADRWS).

If a partial sector (less than 320 words) is written, the remaining words of the sector are automatically set to zero.

As each sector is written, the subroutine supplies the sector identification word. The identification word for the first sector is obtained from the I/O area, although it and subsequently generated identification words are not included in the word count.

Seek. Initiates a seek as specified by the seek option digit. If any errors are detected, the operation is retried up to 16 times.

The seek function requires that the user set up the normal I/O area parameters (see I/O Area Parameter) even though only the sector address in the I/O area is used.

Logical Drive Code

Digit 2 defines the logical drive code (0, 1, 2, 3, or 4). This digit is used only with the DISKN test function.

Seek Option

If digit 3 of the control parameter is zero, a seek is executed to the cylinder whose sector address is in the I/O area; if non-zero, a seek is executed to the next non-defective cylinder toward the center, regardless of the sector address in the I/O area. This seek to the next non-defective cylinder must be taken into consideration when planning for the "streaming" of data.

This option is valid only when the seek function is specified.

Displacement Option

If digit 4 of the control parameter is zero, the sector address word contains the absolute sector identification; if non-zero, the file-protect address for the specified cartridge is added to bits 4-15 of the sector address word to generate the effective sector identification. The file-protect address is the sector identification of the first unprotected sector, i.e., the address of the first sector of Working Storage.

I/O Area Parameter

The I/O area parameter is the label of the first of two control words which precede the user's I/O area. The first word contains the number of data words that are to be transferred during the disk operation. This number need not be limited by sector or cylinder size, since the subroutines cross sector and cylinder boundaries, if necessary, in order to transmit the specified number of words.

The second word contains the sector address at which reading or writing is to begin. Bit 0 must be zero. Bits 1-3 are the device identification (drive code) and must be 0, 1, 2, 3, or 4. Bits 4-15 specify the sector address. The user's I/O area follows the two control words.

Error Parameter

If an error is detected, the user can request the subroutine to terminate (that is, to clear the subroutine's busy indicator and turn off interrupt level 2) or to branch to $PST2, with interrupt level 2 on, waiting for operator intervention.

Effective Address Calculation

An effective disk address is calculated as follows:

1. Obtain the sector address found in the sector address word of the I/O area.
2. If the displacement option digit in the control parameter is non-zero, add the sector address of the first sector that is not file-protected.

   NOTE: This address causes an exit to $PRET if it exceeds 1599.

3. If the resultant address is equal to or greater than the sector address of the first defective cylinder, add 8.
4. If the resultant address is equal to or greater than that of the second defective cylinder, add 8 more.
5. If the resultant address is equal to or greater than that of the third defective cylinder, add 8 more.

The address obtained from steps 1-5 is the effective sector address. Defective cylinders are handled in this manner for all operations, including seek and write immediate.

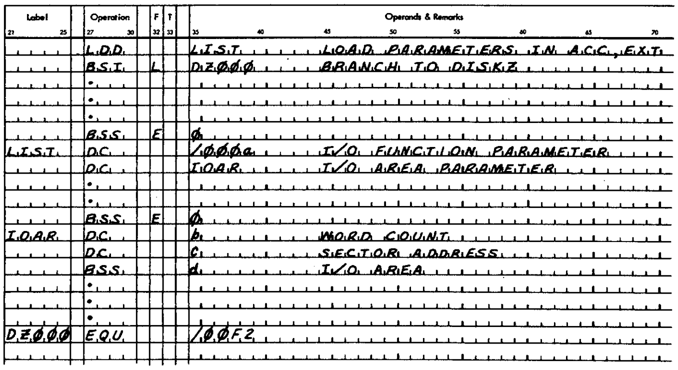
Disk Initialization

Before the Monitor system is stored on a cartridge, the Disk Cartridge Initialization Program (DCIP) must be executed. This program writes sector addresses on the disk cartridge, detects any defective cylinders, stores defective cylinder information and a cartridge ID in sector 0 of cylinder 0, and initializes DCOM. The operating procedure for DCIP is listed in the publication IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide (Form C26-3717).

DISKZ – DISK INPUT/OUTPUT SUBROUTINE

The DISKZ subroutine offers no file protection, no pre-operative parameter checks, no write immediate function, and no write without readback check function. It is intended for use by the Monitor programs and by FORTRAN programs in which disk FORTRAN I/O is used. Although DISKZ has many of the characteristics of an ISS, it is assembled as though it was a mainline and is stored in a special Core Image format in the System Device Subroutine area.

Calling Sequence



```
      LDD       LIST      LOAD PARAMETERS IN ACC, EXT
      BSI  L    DZØØØ     BRANCH TO DISKZ
      .
      .
      .
      BSS  E    Ø
LIST  DC        /ØØØa     I/O FUNCTION PARAMETER
      DC        IOAR      I/O AREA PARAMETER
      .
      .
      BSS  E    Ø
IOAR  DC        b         WORD COUNT
      DC        c         SECTOR ADDRESS
      BSS       d         I/O AREA
      .
      .
      .
DZØØØ EQU       /ØØF2
```

where

a is the I/O function digit: 0 indicates a read, 1 a write.

b is the number of words to be transferred to or from the disk

c is the sector address at which the transfer is to begin,

d is the length of the I/O area. d must be equal to or greater than b.

The word count (first word of the buffer) must be non-negative and must be on an even core boundary. The sector address must be the second word of the buffer. The drive code (0, 1, 2, 3, or 4) is in bits 1-3 of the sector address. Bit zero is always zero. A word count of zero indicates a seek to the cylinder denoted in the sector address. File protection is not provided. If the access arm is not positioned at the cylinder addressed, DISKZ seeks to that cylinder before performing the requested function. A read follows each seek to verify that the seek was successful. No buffer is required for this read.

Buffer Size. Maximum of 320 words.

Operation. DISKZ performs read, seek, and write with readback check functions. Each function returns control to the user after it has been initiated. To determine the completion of a disk operation, the user may test $DBSY (in COMMA) until it is cleared to zero. DISKZ itself tests this word before initiating an operation. Following a write, this subroutine performs a readback check on the data just written. If it detects an error, it reexecutes the write. Similarly, if a sector is not located or if an error is detected during a read, DISKZ repeats the operation. All operations are attempted 16 times before DISKZ indicates an unrecoverable error.

If a partial sector (less than 320 words) is written, the remaining words of the sector are set to zero.

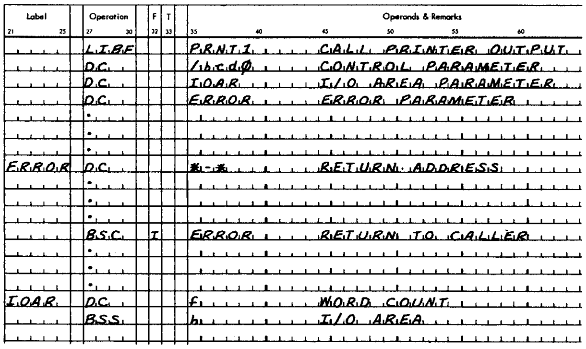
Subroutines Required. No other subroutines are required by DISKZ.

NOTE: It is important to realize that the DISKZ subroutine is designed to operate in an error-free environment; it is not recommended for general usage. The user should therefore use DISK1 or DISKN whenever possible.

1132 PRINTER SUBROUTINE (PRNT1)

The printer subroutine PRNT1 handles all print and carriage control functions relative to the IBM 1132 Printer (see also PRNT2). 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



```
      LIBF      PRNT1     CALL PRINTER OUTPUT
      DC        /bcdØ     CONTROL PARAMETER
      DC        IOAR      I/O AREA PARAMETER
      DC        ERROR     ERROR PARAMETER
      .
      .
      .
ERROR DC        *-*       RETURN ADDRESS
      .
      .
      .
      BSC  I    ERROR     RETURN TO CALLER
      .
      .
      .
IOAR  DC        f         WORD COUNT
      BSS       h         I/O AREA
```

where

b is the I/O function digit,

c is the "immediate" carriage operation digit,

d is the "after-print" carriage operation digit,

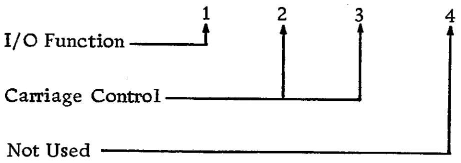
f is the number of words to be printed on the 1132 Printer,

h is the length of the I/O area. h must be equal to or greater than f.

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 Numeric	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, 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 subroutine after the line of data has been printed (see Appendix B for error codes). Upon return from this error subroutine, 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 3.

Table 3. 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

Print Numeric. 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 3. 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.

Carriage control functions do not check for channel 9 or channel 12 indications.

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.

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.

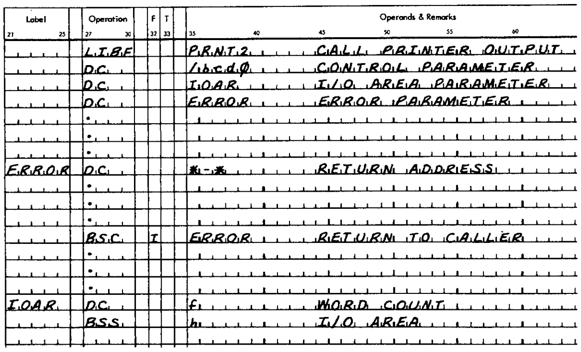
1132 PRINTER/SYNCHRONOUS COMMUNICATIONS ADAPTOR SUBROUTINE (PRNT2)

The printer subroutine PRNT2 is an additional printer subroutine for the IBM 1132 Printer, specifically provided to permit concurrent operation of the 1132 and the Synchronous Communications Adapter. PRNT2 handles all print and carriage control functions related to the 1132.

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 word.

Restriction. The PRNT1 and PRNT2 subroutines are mutually exclusive; i.e., both subroutines may not be in core at the same time. Thus, if the Synchronous Communications Adapter is in operation, the PRNT2 subroutine must be used for concurrent operation of the 1132 Printer. If the PRNT2 subroutine is required in a core load for the concurrent operation of the 1132 Printer and the Adapter, all IBM- and user-written programs in that core load using the PRNT1 subroutine must be modified to use the PRNT2 subroutine.

Calling Sequence



```
Label   Operation  F T  Operands & Remarks
21   25 27    30  32 33 35   40   45   50   55   60
        LIBF            PRNT2     CALL PRINTER OUTPUT
        DC              /bcd0     CONTROL PARAMETER
        DC              IOAR      I/O AREA PARAMETER
        DC              ERROR     ERROR PARAMETER
        .
        .
        .
ERROR   DC              *-*       RETURN ADDRESS
        .
        .
        .
        BSC        I    ERROR     RETURN TO CALLER
        .
        .
        .
IOAR    DC              f         WORD COUNT
        BSS             h         I/O AREA
```

where

b is the I/O function digit,

c is the "immediate" carriage operation digit,

d is the "after-print" carriage operation digit,

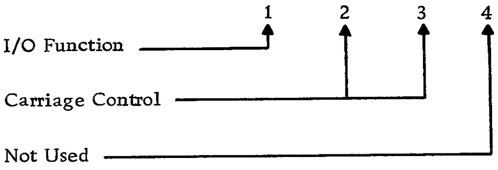
f is the number of words to be printed on the 1132 Printer,

h is the length of the I/O area. h must be equal to or greater than f.

The calling sequence parameters are described in the following paragraphs.

Control Parameter

The control 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 the 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 Numeric	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, to LIBF+3 if the previous operation has been completed.

Print. Prints characters from the user's I/O area; checks 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 (see Appendix B for error codes). Upon return from this error routine, a skip to channel 1 is initiated or the operation 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 3.

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

Carriage Control

Digits 2 and 3 specify the carriage control operations listed in Table 3. 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 Carriage, and digits 2 and 3 both specify carriage operations, only digit 2 is used.

Carriage control functions do not check for channel 9 and channel 12 indications.

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 of data to be printed. The data must be in EBCDIC format, packed two characters per word.

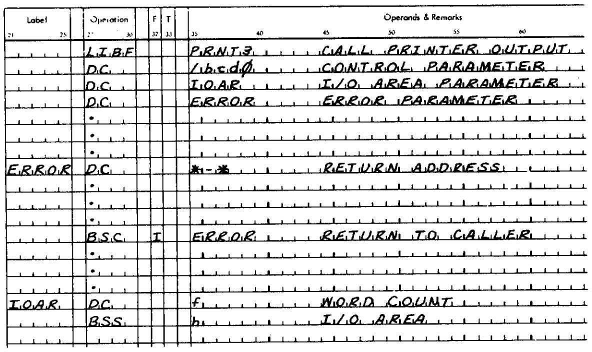
Error Parameter

See Basic Calling Sequence.

1403 PRINTER SUBROUTINE (PRNT3)

The printer subroutine PRNT3, available only with the Monitor system, handles all print and carriage control functions relative to the 1403 Printer. Only one line of data can be printed and/or one carriage operation executed with each call to the printer subroutine.

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
21  25 27     30  32 33 35   40   45   50   55   60
       LIBF          PRNT3     CALL PRINTER OUTPUT
       DC            /bcdØ     CONTROL PARAMETER
       DC            IOAR      I/O AREA PARAMETER
       DC            ERROR     ERROR PARAMETER
       .
       .
       .
ERROR  DC            *-*       RETURN ADDRESS
       .
       .
       .
       BSC        I  ERROR     RETURN TO CALLER
       .
       .
       .
IOAR   DC            f         WORD COUNT
       BSS           h         I/O AREA
```

where

b is the I/O function digit,

c is the "immediate" carriage operation digit,

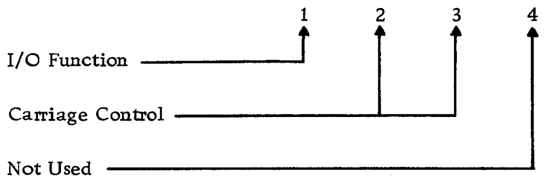
d is the "after-print" carriage operation digit,

f is the number of words to be printed on the 1403 Printer,

h is the length of the I/O area. h must be equal to or greater than f.

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 the 1403 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

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

Test. Branches to LIBF+2 if the previous operation has not been completed, 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 and error indications. If any of these conditions are detected, the subroutine branches to the user's error subroutine after the line of data has been printed with an error code in the Accumulator (see Appendix B). Upon return from this error subroutine, a skip to channel 1 is initiated and the function is reinitiated or terminated, depending upon the error code and whether the Accumulator is non-zero or zero.

Control Carriage. Controls the carriage as specified by the carriage control digits listed in Table 4.

Carriage Control

Digits 2 and 3 specify the carriage control functions listed in Table 4. 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 function is print, only digit 3 is examined, if the function is control, and digits 2 and 3 both specify carriage operations, only digit 2 is used.

Carriage control functions do not check for channel 9 or channel 12 indications.

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

Table 4. 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
7	- Immediate Skip To Channel 7
8	- Immediate Skip To Channel 8
9	- Immediate Skip To Channel 9
A	- Immediate Skip To Channel 10
B	- Immediate Skip To Channel 11
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
7	- Skip After Print To Channel 7
8	- Skip After Print To Channel 8
9	- Skip After Print To Channel 9
A	- Skip After Print To Channel 10
B	- Skip After Print To Channel 11
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 words of data to be printed. The data must be in 1403 Printer code, packed two characters per word.

Error Parameter

See Basic ISS Calling Sequence.

KEYBOARD/CONSOLE PRINTER

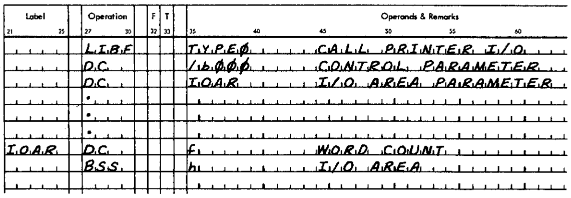
There are two ISSs for the transfer of data to and from the Console Printer and the Keyboard.

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



Label 21 25		Operation 27 30		F 32	T 33		Operands & Remarks 35 40 45 50 55 60
		LIBF					TYPEØ CALL PRINTER I/O
		DC					/bØØØ CONTROL PARAMETER
		DC					IOAR I/O AREA PARAMETER
		•					
		•					
		•					
IOAR		DC					f WORD COUNT
		BSS					h I/O AREA

where

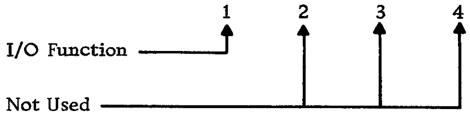
b is the I/O function digit,

f is the number of characters to be printed on the console printer,

h is the length of the I/O area. h must be equal to or greater than f.

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 Keyboard and/or Console Printer. The function, their associated digital values, and the required parameters are listed and 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, 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 portion of the subroutine, which then unlocks the Keyboard.
2. When a key is pressed, a character interrupt signals the interrupt response portion that a character is ready to be read into core storage.
3. The interrupt response portion converts the keyboard data to Console Printer 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 subroutine checks to determine if the final character has been read and printed. If so, the operation is considered complete. In the DM1 and C/PT system, if the Console Printer becomes not-

ready during printing, the subroutines loop, waiting for the Console Printer to become ready. In the DM2 system they trap to $PRET or $PST4.

5. Steps 2 through 4 are repeated until the specified number of characters have been read and printed. The characters read into the I/O area are in a code similar to 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.

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

Erase Field. When the interrupt response portion recognizes the erase field control character, it assumes that the entire message is in error and is to be entered again. The subroutine 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 a 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 (new line) character terminating the new message.

End-of-Message. When the interrupt response portion 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 interrupt 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 subroutine to execute an indirect BSI instruction to core location 2C ($IREQ in DM2), where the user must have previously stored the address of an interrupt request subroutine. Bit 1 of the Accumulator contains the Keyboard/Console Printer identification bit, that is, the device status word, shifted left two bits.

The user's interrupt request subroutine must return to the ISS subroutine via the return link. The user's subroutine 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-operative 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 and is equal to one-half the number of characters if the print function is requested.

PAPER TAPE SUBROUTINES (DM1 AND C/PT SYSTEM)

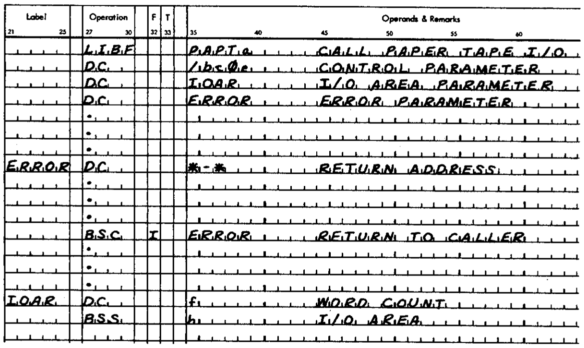
The paper tape subroutines, PAPT1 and PAPTN, handle the transfer of data from the IBM 1134 Paper Tape Reader to core storage and from core storage to the IBM 1055 Paper Tape Punch. Any even number of characters can be transferred via one calling sequence.

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

The PAPT1 subroutine can operate 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 already be in the packed format when the subroutine is called for a punch function.

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
21  25 27     30  32 33 35   40   45   50   55   60
       LIBF            PAPTa       CALL PAPER TAPE I/O
       DC              /bcØe       CONTROL PARAMETER
       DC              IOAR        I/O AREA PARAMETER
       DC              ERROR       ERROR PARAMETER
       .
       .
       .
ERROR  DC              *-*         RETURN ADDRESS
       .
       .
       .
       BSC      I      ERROR       RETURN TO CALLER
       .
       .
       .
IOAR   DC              f           WORD COUNT
       BSS             h           I/O AREA
```

where

a is a 1 or N,

b is the I/O function digit,

c is a check digit,

e is a device identification digit,

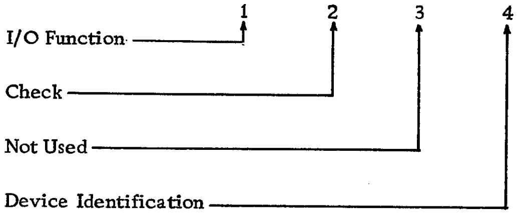
f is the number of words to be read from or punched into paper tape,

h is the length of the I/O area. h must be equal to or greater than f.

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.

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

Read. Reads paper tape characters into the specified number of words in the I/O area. Initiating reader motion causes an interrupt to occur when a character can be read into core. If the specified number of words has not been read, or the stop character has not been read (see Check), reader motion is again initiated.

Punch. Punches paper tape characters into the tape from the words in the I/O area. Each character punched causes an interrupt which indicates that the next character can be accepted. The operation is terminated by transferring either a stop character or the specified number of words.

Check Digit

The check digit specifies whether or not word-count checking is desired while completing a read or punch operation as shown below:

0 Check
1 No check

Check. This function should be used with the Perforated Tape and Transmission Code (PTTC/8) only (see Data Codes). The PTTC/8 code for DEL is used as the delete character when reading. The delete character is not placed in the I/O area and therefore does not enter into the count of the total number of words to be read.

The PTTC/8 code for NL is used as the stop character when doing a read or punch.. On a read operation, the NL character is transferred into the I/O area. On a punch operation, the NL character is punched into the paper tape.

When the NL character is encountered before the specified number of words has been read or punched, the operation is terminated. When the specified number of words has been read or punched, the operation is terminated, even though a NL character has not been encountered.

No Check. The read or punch function is terminated when the specified number of words has been read or punched. No checking is done for a delete or stop character.

Device Identification

When the test function is specified, the PAPTN subroutine must be told which device (reader or punch) is to be tested for an operation complete indication. (Remember that both the reader and the punch can operate simultaneously.) Therefore, the device identification is used only for the test function in the PAPTN subroutine. If the device identification digit is a 0, the subroutine tests for a reader complete indication; if it is a 1, the subroutine tests for a punch complete indication.

I/O Area Parameter

The I/O area parameter is the label of the control word that precedes the user's I/O area. It consists of a word count that specifies the number of words to be read into or punched from core. Since characters are packed two per word in the I/O area, this count is one-half the maximum number of characters transferred. Because an entire eight-bit channel image is transferred by the subroutine, any combination of channel punches is acceptable. The data can be a binary value or a character code. The code most often used is the PTTC/8 code. (See Data Codes.)

Error Parameter

See Basic ISS Calling Sequence.

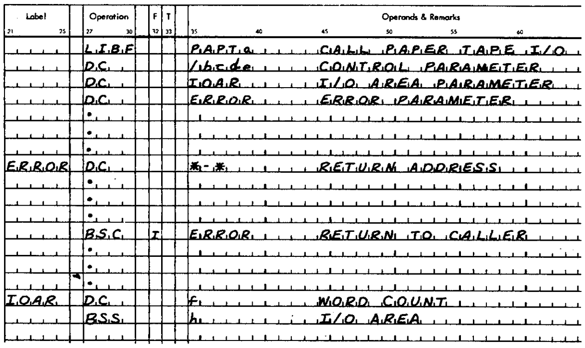
PAPER TAPE SUBROUTINES (DM2 SYSTEM)

The paper tape subroutines, PAPT1, PAPTN, and PAPTX, handle the transfer of data from the IBM 1134 Paper Tape Reader to core storage and from core storage to the IBM 1055 Paper Tape Punch. Any even number of characters may be transferred via one calling sequence (PAPTX also allows an odd character count).

The PAPTN or PAPTX subroutine must be used if simultaneous reading and punching are desired. The PAPT1 subroutine will operate both devices but only one at a time. The PAPT1 and PAPTN subroutines use only a word count, reading and punching an even number of characters; PAPTX can use a word count or character count, permitting an odd number of characters to be read or punched. PAPTX allows the user to start punching from or reading into the left or right half of a word. One-frame records can be written on tape.

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. The data is packed two characters per computer word by the subroutine when reading, and must be in that form when the subroutine is called for a punch function.

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
21  25 27    30   32 33 35   40   45   50   55   60
       LIBF             PAPTa     CALL PAPER TAPE I/O
       DC               /bcde     CONTROL PARAMETER
       DC               IOAR      I/O AREA PARAMETER
       DC               ERROR     ERROR PARAMETER
       .
       .
       .
ERROR  DC               *-*       RETURN ADDRESS
       .
       .
       .
       BSC        I     ERROR     RETURN TO CALLER
       .
       .
       .
IOAR   DC               f         WORD COUNT
       BSS              h         I/O AREA
```

where

a is 1, N, or X.

b is the I/O function digit,

c is a check digit,

d is the character mode digit,

e is a device identification digit,

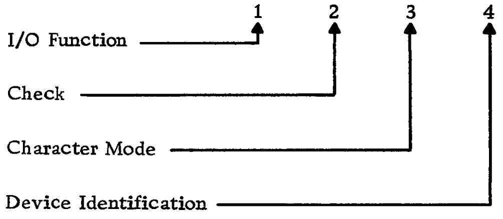
f is the number of words to be read from or punched into paper tape,

h is the length of the I/O area. h must be equal to or greater than f.

The parameters used in the above calling sequence 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 a particular operation performed on the 1134/1055 Paper Tape attachment. The functions, associated digital values and required parameters are listed and described below.

Function	Digital Value	Required Parameters*
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.

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

Read. Reads paper tape characters into the specified number of words in the I/O area. Initiating reader motion causes an interrupt to occur when a character can be read into core. If the specified number of words has not been read or the stop character has not been read (see Check), reader motion is again initiated.

Punch. Punches paper tape characters into the tape from the words in the I/O area. Each character punched causes an interrupt which indicates that the next character can be accepted. The operation is terminated either by encountering a stop character (see Check) or by transferring the requested number of words.

Check Digit

The check digit specifies whether or not checking is desired while doing a read or punch operation.

0 - Check
1 - No check

No Check. The read or punch function is terminated when the specified number of words or characters has been read or punched. No check is made for a delete or stop character.

Check. This function should be used with the perforated tape and transmission (PTTC/8) code only (see Data Codes). The PTTC/8 code for DEL will be used as the delete character when doing a read. The delete character is not placed in the I/O area and therefore is not included in the word or character count.

The PTTC/8 code for NL will be used as the stop character when doing a read or punch. On a read operation, the NL character is transferred into the I/O area and causes the operation to be terminated. On a punch operation, the NL character is punched in the paper tape and causes the operation to be terminated.

When the NL character is encountered before the specified number of words has been read or punched, the operation is terminated. When the specified number of words has been read or punched, the operation is terminated even though an NL character has not been encountered.

Character Mode

This digit is examined by the PAPTX subroutine

- If it is zero, the first word of this I/O area is interpreted as a word count.
- If it is non-zero, the first word of the I/O area is interpreted as a character count:

  If the character mode digit is non-zero and even, the first character will be read into or punched from bits 0-7 of the first data word. Bits 8-15 of the last data word will not be altered if the character count is odd.

  If the character mode digit is non-zero and odd, the first character will be read into or punched from bits 8-15 of the first data word. Bits 0-7 of the first data word will not be altered. If the character count is even, bits 8-15 of the last data word will not be altered.

Device Identification

When the test function is specified, the PAPTN and PAPTX subroutines must be told which device (reader or punch) is to be tested for an "operation complete" indication. (Remember that both the reader and the punch can operate simultaneously.) Therefore, the device identification digit is used for the test function in the PAPTN and PAPTX subroutines only; if it is a 0, the subroutine tests for a "reader complete" indication; if it is a 1, the subroutine tests for a "punch complete" indication.

I/O Area Parameter

The I/O area parameter is the label of the control word that precedes the user's I/O area. Chaining is not permitted with the paper tape subroutine; therefore, the control word consists of a word count or character count only. The word count specifies the number of words to be read into or punched from the user's I/O area. Since characters are packed two per word in the I/O area, this count is one-half the maximum number of characters transferred. The character count, used only by the PAPTX subroutine if the character mode is non-zero, is the maximum number of characters to be read or punched.

Because an entire 8-bit channel image is transferred by the subroutine, any combination of channel punches is acceptable. The data may be a binary value or a character code. The code most often used is the PTTC/8 code (see Data Codes).

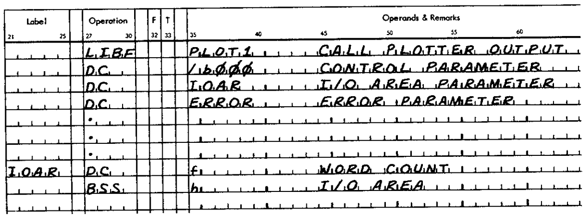
Error Parameter

(See Basic Calling Sequence.)

PLOTTER SUBROUTINE (PLOT1)

The plotter subroutine converts hexadecimal digits in the user's output area into actuating signals that control the movement of the plotter recording pen. Each hexadecimal digit in the output area is translated into a plotter operation that draws a line segment or raises or lowers the recording pen. The amount of data that can be recorded with one calling sequence is limited only by the size of the corresponding output area.

Calling Sequence



```
Label     Operation   F  T   Operands & Remarks
21    25  27    30   32 33   35   40   45   50   55   60
          LIBF               PLOT1          CALL PLOTTER OUTPUT
          DC                 /b000          CONTROL PARAMETER
          DC                 IOAR           I/O AREA PARAMETER
          DC                 ERROR          ERROR PARAMETER
          .
          .
          .
IOAR      DC                 f              WORD COUNT
          BSS                h              I/O AREA
```

where

b is the I/O function digit,

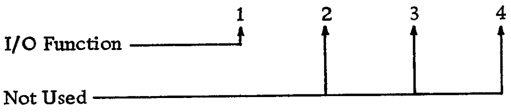
f is the number of words of plotter data,

h is the length of the I/O area. h must be equal to or greater than f.

The calling sequence parameters 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 1627 Plotter. The functions, their associated digital value, and the required parameters are listed and described below.

Function	Digital Value	Required Parameter*
Test	0	Control
Write	1	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, to LIBF+3 if the previous operation has been completed.

Write. Changes hexadecimal digits in the output area into signals that actuate the plotter. Table 5 lists the hexadecimal digits and the plotting actions they represent. Figure 5 shows the binary and hexadecimal configurations for drawing the letter E.

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 used.

Error Parameter

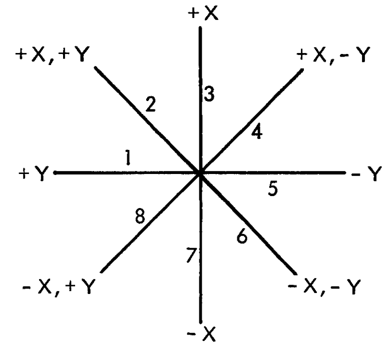
This parameter is not used but must be included because the subroutine will return to LIBF+4. (See Basic Calling Sequence.)

PLOTTER SUBROUTINE (PLOTX)

The PLOTX subroutine converts the hexadecimal digit in the parameter into a control word. The control word is stored in a buffer inside the PLOTX subroutine. One digit is transferred with each calling sequence. When the plotter is ready to accept control, the movement of the plotter recording pen is controlled by the words in the PLOTX buffer.

Table 5. Plotter Control Digits

Hexadecimal Digit	Plotter Action (See Diagram Below)
0	Pen Down
1	Line Segment = +Y
2	Line Segment = +X, +Y
3	Line Segment = +X
4	Line Segment = +X, - Y
5	Line Segment = - Y
6	Line Segment = - X, - Y
7	Line Segment = - X
8	Line Segment = - X, + Y
9	Pen Up
A	Repeat the previous pen motion the number of times specified by the next digit (Maximum-15 times)
B	Repeat the previous pen motion the number of times specified by the next two digits (Maximum-255 times)
C	Repeat the previous pen motion the number of times specified by the nexi three digits (Maximum-4095 times)
D	Not Used
E	Not Used
F	Not Used



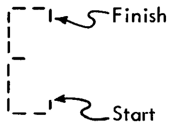
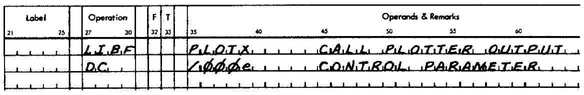
Binary	Hexadecimal	Figure
0000011100010001	0711	
0011101000100101	3A25	
1001000100000011	9103	
1010001001010101	A255	
0111100111111111	79FF	

Figure 5. Plotter Example

Calling Sequence


Label
21
25
Operation
27
30
F
T
32
33
Operands & Remarks
35
40
45
50
55
60

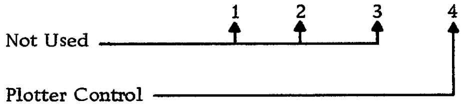
```
      LIBF  PLOTX     CALL PLOTTER OUTPUT
      DC    /ØØØe     CONTROL PARAMETER
```

where

e is the plotter control digit.

Control Parameter

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



Plotter Control

The plotter control digit specifies the recording pen action to be taken. This digit is expressed in hexadecimal.

Hexadecimal Digit	Plotter Action
0	Pen down
1	Line segment = +Y
2	Line segment = +X, +Y
3	Line segment = +X
4	Line segment = +X, -Y
5	Line segment = -Y
6	Line segment = -X, -Y
7	Line segment = -X
8	Line segment = -X, +Y
9	Pen up
A-F	Not used

If there is no room in the buffer for the control digit, the subroutine will loop until there is room.

If the plotter is in a not-ready, not-busy condition, the subroutine traps to $PRET.

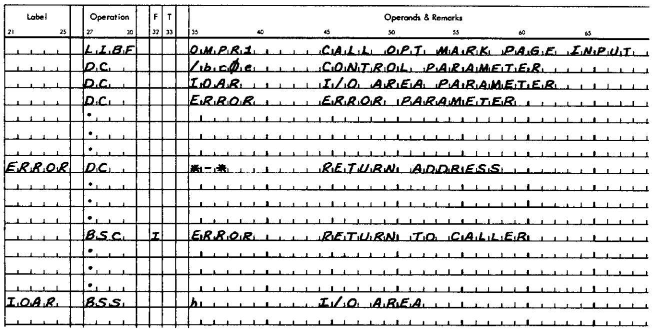
The PLOTX subroutine has no error handling capabilities.

1231 OPTICAL MARK PAGE READER SUBROUTINE (OMPR1)

The Optical Mark Page Reader subroutine OMPR1 handles the reading of paper documents eight and one-half inches wide by eleven inches deep by the 1231 Optical Mark Page Reader. A maximum of 100 words from one page can be read with one call to the subroutine.

When called to perform a read function, OMPR1 performs a feed function and reads a page into core storage according to the Master Control Sheet (see the publication IBM 1231, 1232 Optical Mark Page Readers, Form A21-9012), and the setting of the switches on the reader. Other functions performed by OMPR1 are feed, stacker select, and disconnect.

Calling Sequence



```
Label Operation F T  Operands & Remarks
21    27        32 33 35
      LIBF          OMPR1     CALL OPT MARK PAGE INPUT
      DC            /bcØe     CONTROL PARAMETER
      DC            IOAR      I/O AREA PARAMETER
      DC            ERROR     ERROR PARAMETER
      .
      .
      .
ERROR DC            *-*       RETURN ADDRESS
      .
      .
      .
      BSC   I       ERROR     RETURN TO CALLER
      .
      .
      .
IOAR  BSS           h         I/O AREA
```

where

b is the I/O function digit,

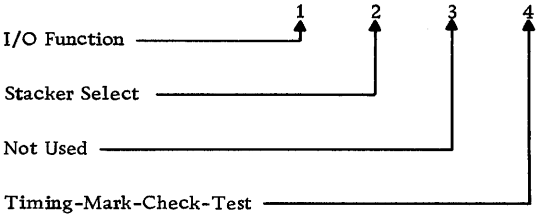
c is the stacker select digit,

e is the timing-mark-check test digit,

h is the length of the I/O area. h must be equal to or greater than the number of words designated to be read on the Master Control Sheet.

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 1231 reader. The functions, their associated digital values, and the required parameters are:

Function	Digital Value	Required Parameters*
Test	0	Control
Read	1	Control, I/O Area, Error
Feed	3	Control
Disconnect	4	Control
Stacker Select	5	Control

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

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

The operation to be tested is specified by the fourth digit of the control parameter. A zero value in digit 4 specifies a normal device-busy test; that is, a test to determine if there is an operation in progress for which no operation complete interrupt has occurred. The subroutine is "not busy" once the operation complete interrupt takes place. A value of one for digit 4 specifies a Timing-Mark-Check-Busy test. This test indicates a "busy" condition as long as the Test-Timing-Mark-Check indicator in the Device Status Word is on. If the user wishes to run with the Timing Mark Switch set on, it is recommended that digit 4 be set to one when performing a test function.

A test function must not directly follow a feed function.

Read. Reads words or segments (response positions 1-5 or 6-10 of any word) from a document page into core storage starting at the I/O area address. It is not necessary for the user to perform a feed function prior to a read. In the absence of a feed, the read feeds the document before reading. When a read function follows a feed, the read begins with the document started by the feed. The number of bits per word read and the number of words per document read depends upon the way in which the Master Control Sheet is programmed (see the publication IBM 1231 Optical Mark Page Readers, Form A21-9012). OMPR1 reads a maximum of 100 words. Any word not programmed to be read (mark positions 8 or 18 not penciled on the Master Control Sheet) is skipped. Digit 2 of the control parameter specifies whether or not the document being read is to be stacker-selected. If digit 2 is set to one, the document is stacker-selected; if digit 2 is set to zero, it is not.

NOTE: On a feed, or feed as the result of a read, the document is fed from the hopper, the selected data is read into a delay line (and read out on a read), and the document continues through the machine to the stacker.

Feed. Initiates a feed cycle. This function advances a document from the hopper through the read station and into the stacker. Selected information from the document is stored in a delay line. A read function following a feed causes this data to be read. If a feed function is followed by another feed function without an intervening read function, the data read from the document corresponding to the first feed is overlaid in the delay line by the data read from the second document.

A feed function must not be followed directly by a test function.

Disconnect. Terminates the read function on the data currently being read from the delay line. The subroutine busy indicator is cleared.

Stacker Select. Performs a stacker select on the sheet currently being read (and fed), providing the stacker select function is requested while the "OK to select" bit is on in the Device Status Word (DSW). This bit remains on until 50 milliseconds after the read operation is completed. If the request to select arrives too late, the sheet falls in the normal stacker.

I/O Area Parameter

The I/O area parameter is the label of the user's I/O area.

Error Parameter

There is an error parameter for the read function only. Exits are made to the user's error subroutine when the following conditions are detected:

Master Control Sheet Error
Timing Mark Error
Read Error
Hopper Empty
Document Selected.

(See Basic Calling Sequence and Appendices B and C).

Many of the functions and capabilities available within the general I/O and conversion subroutines described in this manual are beyond specification by the FORTRAN language. For example, the feed function of the 1442 cannot be specified in FORTRAN. Therefore, a set of limited-function I/O and conversion subroutines is included in the subroutine library for use by FORTRAN-compiled programs. Any subroutines written in Assembler language that execute I/O operations, and that are intended to be used in conjunction with FORTRAN-compiled programs must employ these special I/O subroutines for any I/O device specified in a mainline *IOCS record or for any device on the same interrupt level.

These subroutines are intended to operate in an error-free environment and thus provide no pre-operative parameter checking.

The subroutine library contains the following special routines:

DISKZ	-	Disk Input/Output Subroutine (DM1 only)
CARDZ	-	1442 Input/Output Subroutine
TYPEZ	-	Keyboard/Console Printer Input/Output Subroutine
WRTYZ	-	Console Printer Subroutine
PRNTZ	-	1132 Printer Subroutine
PAPTZ	-	Paper Tape Input/Output Subroutine
PLOTX	-	1627 Plotter Subroutine (see PLOTX)
HOLEZ	-	IBM Card Code/EBCDIC Conversion Subroutine
EBCTB	-	EBCDIC/Console Printer Code Table
HOLTB	-	IBM Card Code Table
GETAD	-	Subroutine Used to Locate Start Address of EBCTB/HOLTB

GENERAL SPECIFICATIONS

Except for PLOTX, the FORTRAN I/O device subroutines operate in a non-overlapped mode. Thus, the device subroutines do not return control to the calling program until the operation is completed.

The input/output buffer for the subroutines is a 121-word buffer starting at location /003C. The maximum amount of data transferable is listed in the description of each subroutine. Output data must be stored in unpacked (one character per word) EBCDIC format, /00XX. Data entered from an input device is converted to unpacked (one character per word) EBCDIC format,/00XX.

The EBCDIC character set recognized by the subroutine comprises digits 0-9, alphabetic characters A-Z, blank, and special characters $-+.&=(),'/*<%#@. Any other character is recognized as a blank.

The Accumulator, Extension, and Index Registers 1 and 2 are used by the FORTRAN device subroutines and must be saved, if required, before entry into any given FORTRAN subroutine.

The Accumulator must be set to zero for input operations. For output operations, the Accumulator must be set to /0002, except for PRNTZ and WRTYZ, in which output is the only valid operation. Index Registers 1 and 2 are set to the number of characters transmitted, except for PRNTZ (1132 Printer) in which Index Register 2 contains the number of characters printed plus an additional character for forms control.

ERROR HANDLING

Device errors, e.g., not-ready and read check, cause a WAIT in the subroutine itself. After the appropriate corrective action is taken by the operator, PROGRAM START is pressed to execute or reinitiate the operation.

DESCRIPTIONS OF I/O SUBROUTINES

The subroutines described in the sections that follow do not provide a check to determine validity of parameters (contents of Accumulator and Index Register 2). Invalid parameters cause indeterminate operation of the subroutines.

TYPEZ-KEYBOARD/CONSOLE PRINTER I/O SUBROUTINE

Buffer Size. Maximum of 80 words input, 120 words output.

Keyboard Input. The subroutine returns the carrier, reads up to 80 characters from the Keyboard, and stores them in the I/O buffer in EBCDIC format. Upon recognition of the end-of-field character or reception of the 80th character, the subroutine returns control to the user (the remainder of the buffer is unchanged). Upon recognition of the erase field character or the backspace character, the carrier is returned and the subroutine is reinitialized for the re-entry of the entire message. Characters are printed by the Console Printer during Keyboard input.

Console Printer Output. The subroutine returns the carrier and prints the number of characters indicated by Index Register 2 from the I/O buffer.

Subroutines Required. The following subroutines are required with TYPEZ:

HOLEZ, GETAD, EBCTB, HOLTB

WRTYZ – CONSOLE PRINTER OUTPUT SUBROUTINE

Buffer Size. Maximum of 120 words.

Operation. This subroutine returns the carrier and prints the number of characters indicated by Index Register 2 from the I/O buffer.

Subroutines Required. The following subroutines are required with WRTYZ:

GETAD, EBCTB

CARDZ - 1442 CARD READ PUNCH INPUT/OUTPUT SUBROUTINE

Buffer Size. Maximum of 80 words.

Card Input. This subroutine reads 80 columns from a card and stores the information in the I/O buffer in EBCDIC format.

Card Output. This subroutine punches the number of characters indicated by Index Register 2 from the I/O buffer. Punching is done in IBM card code format.

Subroutines Required. The following subroutines are required with CARDZ:

HOLEZ, GETAD, EBCTB, HOLTB

PAPTZ – 1134/1055 PAPER TAPE READER PUNCH I/O SUBROUTINE

Buffer Size. Maximum of 80 characters.

1134 Paper Tape Input. This subroutine reads paper tape punched in PTTC/8 format. Paper tape is read until 80 characters have been stored or until a new-line character is read. If 80 characters have been stored and a new-line character has not been read, one more character, assumed to be a new line character, is read from tape. (Delete and case shift characters cause nothing to be stored.) If the first character read is not a case shift character, it is assumed to be a lower case character. The input is converted to EBCDIC format.

1055 Paper Tape Output. The contents of the I/O buffer is converted from EBCDIC to PTTC/8, and the number of characters indicated by Index Register 2 is punched, in addition to the required case shift characters.

PRNTZ – 1132 PRINTER OUTPUT SUBROUTINE

Buffer Size. Maximum of 121 characters.

Index Register 2. The value stored in Index Register 2 must be the number of characters to be printed plus an additional character for carriage control. Up to 120 characters can be printed in any one operation. The first character to be printed is stored in location /003D.

The carriage of the 1132 Printer is controlled prior to the printing of a line. The following is a list of the carriage control characters and their related functions:

/00F1	Skip to channel 1 prior to printing
/00F0	Double space prior to printing
/004E	No skip or space prior to printing
Any other character	Single space prior to printing.

Channel 12 Control. If a punch in channel 12 is encountered while a line is being printed, a skip-to-channel-1 is taken prior to the printing of the next line.

DISKZ – DISK INPUT/OUTPUT SUBROUTINE (DM1 ONLY)

Operation. This subroutine reads or writes disk storage. Data is transferred to or from the disk, one sector (320 words) at a time.

Following a write operation, the subroutine performs a read back check on the data just written. If an error is detected, a rewrite occurs. Similarly, if a sector is not located or an error is detected during a read, the subroutine repeats the operation. A read is attempted ten times before the computer halts with an error display.

Subroutines Loaded. The following subroutine is required with DISKZ.

ILS02

Many of the I/O and conversion subroutines cannot be specified in FORTRAN. Therefore, the System Library includes a set of limited-function I/O and conversion subroutines for FORTRAN programs. Any Assembler language I/O subroutines used by FORTRAN programs must employ these special subroutines for any I/O device specified in a mainline IOCS control record.

Of all the FORTRAN device subroutines, only DISKZ, PRNZ, and PLOTX return control to the caller after initiating an operation (PLOTX is described with the basic ISSs).

These subroutines are intended for use in an error-free environment and thus provide no pre-operative parameter checking.

The System Library contains the following ISS subroutines for FORTRAN programs:

CARDZ	-	1442 Input/Output Subroutine
PNCHZ	-	1442 Output Subroutine
READZ	-	2501 Input Subroutine
TYPEZ	-	Keyboard/Console Printer I/O Subroutine
WRTYZ	-	Console Printer Subroutine
PRNTZ	-	1132 Printer Subroutine
PRNZ	-	1403 Printer Subroutine
PAPTZ	-	Paper Tape Input/Output Subroutine
PLOTX	-	1627 Plotter Subroutine
DISKZ	-	Disk Input/Output Subroutine
HOLEZ	-	IBM Card Code/EBCDIC Conversion Subroutine
EBCTB	-	EBCDIC/Console Printer Code Table
HOLTB	-	IBM Card Code Table
GETAD	-	Subroutine to Locate Start Address of EBCTB/HOLTB

GENERAL SPECIFICATIONS (EXCEPT DISKZ)

The "Z" device subroutines are ISS subroutines. They use a 121-word input/output buffer, contained in the non-disk FORTRAN I/O subroutine SFIO. The maximum amount of data transferable is listed in the description of each subroutine. Output data must be stored in unpacked (one character per word) EBCDIC format. Input data is converted to unpacked EBCDIC format.

The EBCDIC character set recognized by the subroutines comprises digits 0-9, alphabetic characters A-Z, blank, and special characters $-+.&=(),'/*<%#@. Any other character is recognized as a blank.

The Accumulator, Extension, and Index Registers 1 and 2 are used by the FORTRAN device subroutines and must be saved, if required, before entry into the subroutines. The Accumulator must be set to zero for input operations.

For output operations, the Accumulator must be set to /0002, except for PRNZ, PRNTZ, PNCHZ, and WRTYZ, in which output is the only valid operation. Index Register 2 must be set to the number of characters to be transferred, except for PRNZ and PRNTZ. For these two subroutines, Index Register 2 must contain the number of characters to be printed plus an additional character for carriage control. Index Register 1 must contain the starting address of the input buffer.

ERROR HANDLING

Device errors, e.g., not ready and read check, result in a branch to $PST1, $PST2, $PST3, and $PST4 depending on the level to which the device is assigned. After the appropriate corrective action is taken by the operator, PROGRAM START is pressed to execute or reinitiate the operation.

If a monitor control record is encountered by CARDZ, READZ, or PAPTZ, the subroutine initiates a CALL EXIT. The control record itself will not be processed.

DESCRIPTIONS OF I/O SUBROUTINES

The subroutines described in the sections that follow do not provide a check to determine validity of parameters (contents of Accumulator and Index Register 2). Invalid parameters cause indeterminate operation of the subroutines.

TYPEZ-KEYBOARD/CONSOLE PRINTER I/O SUBROUTINE

Buffer Size. Maximum of 80 words input, 120 words output.

Keyboard Input. The subroutine returns the carrier and reads up to 80 characters from the Keyboard and stores them in the I/O buffer in EBCDIC format. Upon recognition of the end-of-field character or reception of the 80th character, the subroutine returns control to the user (the remainder of the buffer is unchanged). Upon recognition of the erase field character or the backspace character, the carrier is returned and the subroutine is reinitialized for the re-entry of the entire message. Characters are printed by the Console Printer during Keyboard input.

Console Printer Output. The subroutine returns the carrier and prints the number of characters indicated by Index Register 2 from the I/O buffer.

Subroutines Required. The following subroutines are required with TYPEZ:

HOLEZ, GETAD, EBCTB, HOLTB

WRTYZ – CONSOLE PRINTER OUTPUT SUBROUTINE

Buffer Size. Maximum of 120 words.

Operation. This subroutine returns the carrier and prints the number of characters indicated by Index Register 2 from the I/O buffer.

Subroutines Required. The following subroutines are required with WRTYZ:

GETAD, EBCTB

CARDZ – 1442 CARD READ PUNCH INPUT/OUTPUT SUBROUTINE

Buffer Size. Maximum of 80 words.

Card Input. This subroutine reads 80 columns from a card and stores the information in the I/O buffer in EBCDIC format.

Card Output. This subroutine punches the number of characters indicated by Index Register 2 from the I/O buffer. Punching is done in IBM Card Code.

Subroutines Required. The following subroutines are required with CARDZ:

HOLEZ, GETAD, EBCTB, HOLTB

PAPTZ – 1134/1055 PAPER TAPE READER PUNCH I/O SUBROUTINE

Buffer Size. Maximum of 120 characters.

1134 Paper Tape Input. This subroutine reads paper tape punched in PTTC/8 format. The subroutine reads paper tape until 120 characters have been stored or until a new-line character is read. If 120 characters have been stored and a new-line character has not been read, one more character, assumed to be a new line character, is read from tape. (Delete and case shift characters cause nothing to be stored.) If the first character read is not a case shift character, it is assumed to be a lower case character. The input is converted to EBCDIC format.

1055 Paper Tape Output. The contents of the I/O buffer is converted from EBCDIC to PTTC/8, and the number of characters indicated by Index Register 2 is punched, in addition to the required case shift characters.

PRNTZ – 1132 PRINTER OUTPUT SUBROUTINE

Buffer Size. Maximum of 121 characters.

Index Register 2. The value stored in Index Register 2 must be the number of characters to be printed, plus an additional character for carriage control. Up to 120 characters can be printed in any one operation.

The carriage of the 1132 Printer is controlled prior to the printing of a line. The following is a list of the carriage control characters and their related functions:

/00F1	Skip to channel 1 prior to printing
/00F0	Double space prior to printing
/004E	No skip or space prior to printing

Any other character - Single space prior to printing.

Channel 12 Control. If a punch in channel 12 is encountered while a line is being printed, a skip-to-channel-1 is taken prior to the printing of the next line provided the next function is not /004E (no skip or space prior to printing).

PNCHZ – 1442 OUTPUT SUBROUTINE

Buffer Size. Maximum of 80 words.

Card Output. This subroutine punches from the I/O buffer the number of characters indicated in the location preceding the buffer. Punching is done in IBM Card Code.

Subroutines Required. The following subroutines are required with PNCHZ:

HOLEZ, GETAD, EBCTB, HOLTB

READZ – 2501 INPUT SUBROUTINE

Buffer Size. Maximum of 80 words.

Card Input. This subroutine reads 80 columns from a card and stores the information in the I/O buffer in EBCDIC format.

Subroutines Required. The following subroutines are required with READZ:

HOLEZ, GETAD, EBCTB, HOLTB

PRNZ – 1403 PRINTER SUBROUTINE

Buffer Size. Maximum of 121 characters.

Index Register 2. The first character in the I/O buffer is the carriage control character, followed by up to 120 characters to be printed. Therefore, Index Register 2 must contain the number of characters to be printed plus one.

The carriage is controlled prior to the printing of a line; no "after-print" carriage control is performed. The following is a list of the carriage control characters and their related functions:

/00F1	Skip to channel 1 prior to printing
/00F0	Double space prior to printing
/004E	No skip or space prior to printing

Any other character - Single space prior to printing.

Channel 12 Control. If a punch in channel 12 is encountered while a line is being printed, a skip to channel 1 is executed prior to printing the next line provided the next function is not /004E (no skip or space prior to printing).

DATA CODE CONVERSION SUBROUTINES

The basic unit of information within the 1130 Computing System is the 16-bit binary word. This information can be interpreted in a variety of ways, depending on the circumstances. For example, in internal computer operations, words may be interpreted as instructions, as addresses, as binary integers, or as real (floating point) numbers (see Arithmetic and Functional Subroutines).

A variety of data codes exists for the following reasons.

1. The programmer needs a compact notation to represent externally the bit configuration of each computer word. This representation is provided in the hexadecimal notation.
2. A code is required for representing alphameric (mixed alphabetic and numeric) data within the computer. This code is provided by the Extended Binary Coded Decimal Interchange Code (EBCDIC).
3. The design and operation of the input/output devices is such that many of them impose a unique correspondence between character representations in the external medium and the associated bit configurations within the computer. Subroutines are needed to convert input data from these devices to a form on which the computer can operate and to prepare computed results for output on the devices.

This and following sections of the manual describe the data codes used and the subroutines provided for converting data representations among these codes.

A detailed description of the binary, octal, hexadecimal, and decimal number systems is contained in the publication, IBM 1130 Functional Characteristics (Form A26-5881).

DESCRIPTIONS OF DATA CODES

In addition to the internal 16-bit binary representation, the conversion subroutines handle the following codes:

- Hexadecimal Notation
- IBM Card Code
- Perforated Tape and Transmission Code (PTTC/8)
- Console Printer (1053) Code
- 1403 Printer Code (Monitor System only)
- Extended Binary Coded Decimal Interchange Code (EBCDIC)

A list of these codes is contained in Appendix D.

HEXADECIMAL NOTATION

Although binary numbers facilitate the operations of computers, they are awkward for the programmer to handle. A long string of 1's and 0's cannot be effectively transmitted from one individual to another. For this reason, the hexadecimal number system is often used as a shorthand method of communicating binary numbers. Because of the simple relationship of hexadecimal to binary, numbers can easily be converted from one system to another.

In hexadecimal notation a single digit is used to represent a 4-bit binary value as shown in Figure 6. Thus, a 16-bit word in the 1130 System can be expressed as four hexadecimal digits. For example, the binary value

1101001110111011

can be separated into four sections as follows:

Binary	1101	0011	1011	1011
Hexadecimal	D	3	B	B

Another advantage of hexadecimal notation is that fewer positions are required for output data printed, punched in cards, or punched in paper tape. In the example above, only four card columns are required to represent a 16-bit binary word.

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

Figure 6. Hexadecimal Notation

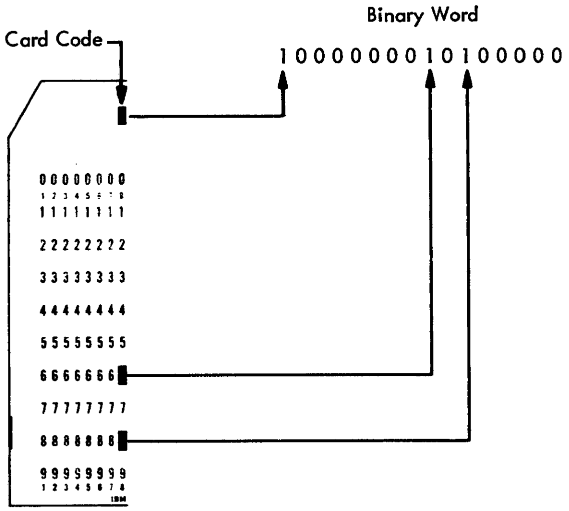
IBM CARD CODE

IBM Card Code can be used as an input/output code with the 1442 Card Read Punch, 1442 Card Punch, and 2501 Card Reader, and as an input code on the Keyboard.

This code defines a character by a combination of punches in a card column. Card-code data is taken from or placed into the leftmost twelve bits of a computer word as shown below:

Card Row	12	11	0	1	2	3	4	5	6	7	8	9	-	-	-	-
Computer Word	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

For example, a plus sign, which has a card code of 12, 6, 8, is placed into core storage in the binary configuration illustrated in the following diagram.



PERFORATED TAPE AND TRANSMISSION CODE (PTTC/8)

The PTTC/8 code is an 8-bit code used with IBM 1134/1055 Paper Tape units. This code represents a character by a stop position, a check position, and six positions representing the 6-bit code, BA8421. PTTC/8 characters can be packed two per computer word as shown below.

	1st								2nd							
PTTC/8 Characters	S	B	A	C	8	4	2	1	S	B	A	C	8	4	2	1
Computer Word	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The graphic character is defined by a combination of binary code and case; a control character is defined by a binary code and has the same meaning in upper or lower case. This implies that upper and lower case characters must appear in a PTTC/8 message wherever necessary to establish or change the case.

The binary and PTTC/8 codes for 1/(lower case) and =? (upper case) are shown in Figure 7.

The delete and stop characters have a special meaning (in check mode only) when encountered by the paper tape subroutines.

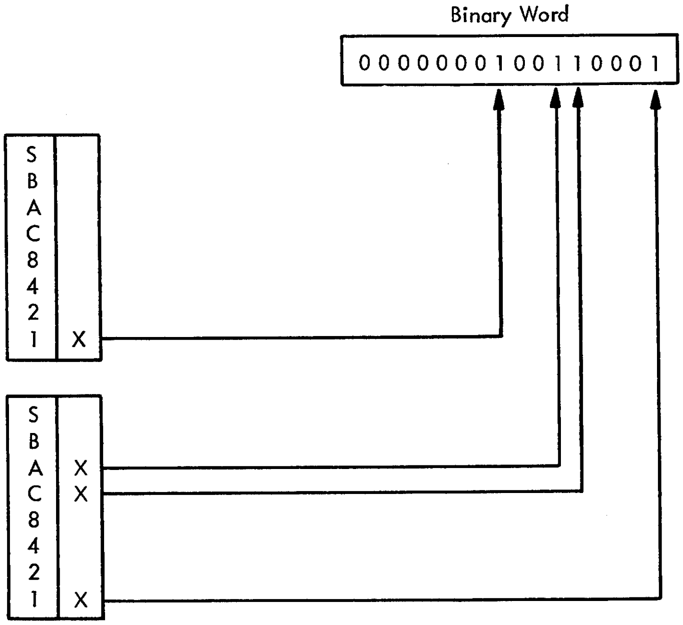


Figure 7. PTTC/8 Code for 1/(Lower Case) or = ? (Upper Case)

CONSOLE PRINTER CODE

The Console Printer uses an 8-bit code that can be packed two characters per 16-bit word.

The following control characters have special meanings when used with the Console Printer.

Character	Control Operation
HT	Tabulate
RES	Shift to black ribbon
NL	Carrier return to new line
BS	Backspace
LF	Line feed without carrier return
RS	Shift to red ribbon

EXTENDED BINARY CODED DECIMAL INTERCHANGE CODE (EBCDIC)

EBCDIC is the standard code for internal representation of alphameric and special characters and for the 1132 Printer. This code uses eight binary bits for each character, thus making it possible to store either one or two characters in a 16-bit word. Combinations of the eight bits allow 256 possible codes. (At present, not all of these combinations represent characters.) The complete EBCDIC code is listed in Appendix D.

For reasons of efficiency, most of the conversion subroutines do not recognize all 256 codes. The asterisked codes in Appendix D constitute the subset recognized by most of the conversion subroutines.

1403 PRINTER CODE

The 1403 Printer uses a 6-bit binary code with one parity bit. Data format is two 7-bit characters per word, as follows:

Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Value	*	P	32	16	8	4	2	1	*	P	32	16	8	4	2	1
	1st data character								2nd data character							
* = Not Used P = Parity Bit																

Parity bits are not assigned by the hardware. The conversion subroutine must assign the parity bits and arrange the characters in the form in which they are to be printed.

CONVERSION SUBROUTINES

These subroutines convert data to and from 16-bit binary words and I/O device codes.

BINDC	Binary value to IBM Card Code decimal value.
DCBIN	IBM Card Code decimal value to binary value.
BINHX	Binary value to IBM Card Code hexadecimal value.
HXBIN	IBM Card Code hexadecimal value to binary value
HOLEB	IBM Card Code subset to EBCDIC subset; EBCDIC subset to IBM Card Code subset.
SPEED	IBM Card Code characters to EBCDIC; EBCDIC to IBM Card Code characters.
PAPEB	PTTC/8 subset to EBCDIC subset; EBCDIC subset to PTTC/8 subset.
PAPHL	PTTC/8 subset to IBM Card Code subset; IBM Card Code subset to PTTC/8 subset.
PAPPR	PTTC/8 subset to Console Printer or 1403 Printer code.
HOLPR	IBM Card Code subset to Console Printer or 1403 Printer code.
EBPRT	EBCDIC subset to Console Printer or 1403 Printer code.

The following conversion tables are used by some of the conversion subroutines.

PRTY	Console Printer and 1403 Printer code.
EBPA	EBCDIC and PTTC/8 subsets.
HOLL	IBM Card Code subset.

The following conversion subroutines are used by the DM2 system only.

BIDEC 32-bit binary value to IBM Card Code decimal value.

DECBI IBM Card Code decimal value to 32-bit binary value.

ZIPCO Supplement to all standard conversions except those involving PTTC/8.

The first four listed subroutines and the DM2 subroutines BIDEC and DECBI change numeric data from its input form to a binary form, or from a binary form to an appropriate output data code. The last eight (including ZIPCO) convert entire messages, one character at a time, from one input/output code to another. The types of conversions accomplished by these subroutines are illustrated in Figure 8.

Except where specified, these subroutines do not alter the Accumulator, Extension, Carry and Overflow indicators, or any index register.

NOTES: 1. All mention of 1403 Printer Code applies to the DM2 system only.

2. The conversion subroutines and conversion tables for the Communications Adapter are described in the publication IBM 1130 Synchronous Communications Adapter Subroutines (Form C26-3706). The subroutines are EBC48, HOL48, and HXCV. The adapter subroutine conversion table is STRTB.

Error Checking

All code conversion subroutines (except SPEED and ZIPCO) accept only the codes marked with an asterisk in Appendix D. An input character that does not conform to a specified code is an error.

BINHX and BINDC subroutines do not detect errors. HXBIN and DCBIN terminate conversion at the point of error detection; they do not replace the character in error. The contents of the Accumulator are meaningless when conversion is terminated because of an error.

The remainder of the conversion subroutines replace the character in error with a space character, stored in the output area in output code. Conversion is not terminated when an error is detected.

When a conversion subroutine detects an error it turns the Carry indicator off and turns the Overflow indicator on before returning control to the

CONVERTED FROM	CONVERTED TO									
	Binary	IBM Card Code (256)	IBM Card Code (Subset)	PTTC/8 (Subset)	EBCDIC (256)	EBCDIC (Subset) 1132 Printer	Console Printer	Hex Equivalent (Card Code)	Decimal Equivalent (Card Code)	1403 Printer Code
Binary								BINHX	BINDC BIDEC	
IBM Card Code (256)					SPEED ZIPCO*		ZIPCO*			ZIPCO*
IBM Card Code (Subset)				PAPHL		HOLEB	HOLPR			HOLPR
PTTC/8 (Subset)			PAPHL			PAPEB	PAPPR			PAPPR
EBCDIC (256)		SPEED					ZIPCO*			ZIPCO*
EBCDIC (Subset) 1132 Printer			HOLEB	PAPEB			EBPRT			EBPRT
Hex Equivalent (Card Code)	HXBIN			PAPHL		HOLEB	HOLPR			HOLPR
Decimal Equivalent (Card Code)	DCBIN DECBI			PAPHL		HOLEB	HOLPR			HOLPR
1403 Printer Code		ZIPCO*			ZIPCO*		ZIPCO*			
Console Printer Code		ZIPCO*			ZIPCO*					ZIPCO*

* In conjunction with appropriate conversion table.

Figure 8. Types of Conversion

user. Otherwise, the settings of the Carry and Overflow indicators are not changed by the conversion subroutines.

BINDC

This subroutine converts a 16-bit binary value to its decimal equivalent in five IBM Card Code numeric characters and one sign character. The five characters and the sign are placed in six computer words as illustrated below.

I/O Locations	Conversion Data	Bits in Core Storage 0 ◄──► 15
Accumulator	+01538	0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0
OUTPT	+	1 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0
OUTPT + 1	0	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
OUTPT + 2	1	0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
OUTPT + 3	5	0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
OUTPT + 4	3	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
OUTPT + 5	8	0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

Calling Sequence

Label	Operation	F	T	Operands & Remarks
21 25	27 30	32	33	35 40 45 50
	LIBF			BINDC
	DC			OUTPT
	•			•
OUTPT	BSS			6

Input

Input is a 16-bit binary value in the Accumulator.

Output

Output is an IBM Card Code sign character (plus or minus) in location OUTPT, and five IBM Card Code numeric characters in OUTPT +1 through OUTPT +5.

Errors Detected

The BINDC subroutine does not detect errors.

DCBIN

This subroutine converts a decimal value in five IBM Card Code numeric characters and a sign character to a 16-bit binary word. The conversion is the opposite of the BINDC subroutine conversion.

Calling Sequence

Label	Operation	F	T	Operands & Remarks
21 25	27 30	32	33	35 40 45 50
	LIBF			DCBIN
	DC			INPUT
	•			•
INPUT	BSS			6

Input

Input is an IBM Card Code sign character in location INPUT and five IBM Card Code decimal characters in INPUT +1 through INPUT +5.

Output

Output is a 16-bit binary value displayed in the Accumulator.

Errors Detected

Any sign other than an IBM Card Code plus, ampersand, space, or minus, or any decimal digits other than a space or 0 through 9 is an error. Any converted value greater than +32767 or less than -32768 is an error.

BINHX

This subroutine converts a 16-bit binary word into hexadecimal notation in four IBM Card Code characters as illustrated below.

I/O Locations	Conversion Data	Bits in Core Storage 0 ◄──► 15
Accumulator	A59E	1010 0101 1001 1110
OUTPT	A	1001 0000 0000 0000
OUTPT + 1	5	0000 0001 0000 0000
OUTPT + 2	9	0000 0000 0001 0000
OUTPT + 3	E	1000 0001 0000 0000

Calling Sequence

Label	Operation	F	T	Operands & Remarks
21 25	27 30	32	33	35 40 45 50
	LIBF			BINHX
	DC			OUTPT
	•			•
OUTPT	BSS			4

Input

Input is a 16-bit binary word in the Accumulator.

Output

Output is four IBM Card Code hexadecimal digits in location OUTPT through OUTPT +3.

Errors Detected

The BINHX subroutine does not detect errors.

HXBIN

This subroutine converts four IBM Card Code hexadecimal characters into one 16-bit binary word. The conversion is the opposite of the BINHX subroutine conversion illustrated above.

Calling Sequence

```
Label  Operation  F T  Operands & Remarks
21  25 27     30  32 33 35
       LIBF            HXBIN
       DC              INPUT
       .               .
INPUT  BSS             4
```

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

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 below.

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

Calling Sequence

```
Label  Operation  F T  Operands & Remarks
21  25 27     30  32 33 35   40   45   50   55   60   65
       LIBF            HOLEB       CALL SUBSET CONVERSION
       DC              /ØØØe       CONTROL PARAMETER
       DC              INPUT       INPUT AREA ADDRESS
       DC              OUTPUT      OUTPUT AREA ADDRESS
       DC              f           CHARACTER COUNT
       .
       .
       .
INPUT  BSS             g           INPUT AREA
       .
       .
       .
OUTPT  BSS             h           OUTPUT AREA
```

where

e indicates the direction of conversion,

f is the number of characters to be converted,

g is the length of the input area. g must be equal to or greater than f if e is 0. If e is 1, g must be equal to f/2, or (f+1)/2 if f is odd.

h is the length of the output area. h must be equal to or greater than f/2 ((f+1)/2 if f is odd) if e is 0. If e is 1, h must be equal to or greater than f.

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.

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

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 CARD0 or CARD1 subroutine sets the bit to a zero.

NOTE: SPEED should not be used with READ0 or READ1 since the 2501 subroutines do not pre-store 1 bits in each word of the I/O area. Use HOLEB or ZIPCO for 2501 operations.

Calling Sequence

```
Label   Operation  F  T   Operands & Remarks
21  25  27   30   32 33   35   40   45   50   55   60   65
      LIBF            SPEED     CALL EBCDIC CONVERSION
      DC              /1ØØde    CONTROL PARAMETER
      DC              INPUT     INPUT AREA ADDRESS
      DC              OUTPUT    OUTPUT AREA ADDRESS
      DC              f         CHARACTER COUNT
      .
      .
      .
INPUT BSS             g         INPUT AREA
      .
      .
      .
OUTPT BSS             h         OUTPUT AREA
```

where

d indicates whether the EBCDIC characters are packed or unpacked,

e indicates the direction of conversion,

f indicates the character count,

g is the length of the input area,

h is the length of the output area,

g and h are defined as follows:

IBM Card Code to packed EBCDIC

$g \geq f$
$h \geq f/2, (f+1)/2$

IBM Card Code to unpacked EBCDIC

$g \geq f$
$h \geq f$

Packed EBCDIC to IBM Card Code

$g \geq f/2, (f+1)/2$
$h \geq f$

Unpacked EBCDIC to IBM Card Code

$g \geq f$
$h \geq f$

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

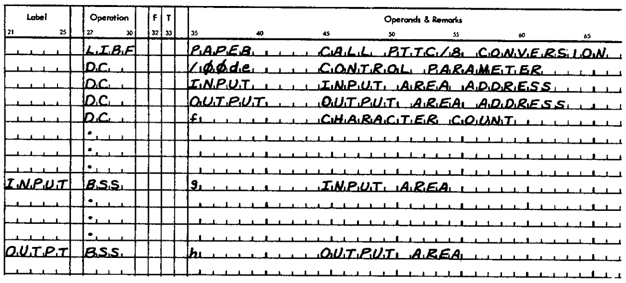
Any input character code not listed in Appendix D is an error. All IBM Card Code punch combinations, except multiple punches in rows 1-7, are legal.

PAPEB

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 below.

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

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
21  25 27    30  32 33 35   40   45   50   55   60   65
      LIBF          PAPEB     CALL PTTC/8 CONVERSION
      DC            /ØØde     CONTROL PARAMETER
      DC            INPUT     INPUT AREA ADDRESS
      DC            OUTPUT    OUTPUT AREA ADDRESS
      DC            f         CHARACTER COUNT
      .
      .
      .
INPUT BSS           g         INPUT AREA
      .
      .
      .
OUTPT BSS           h         OUTPUT AREA
```

where

d is the case initialization digit,

e indicates the direction of conversion,

f indicates the character count,

g is the length of the input area. g must be equal to or greater than f/2 or (f+1)/2.

h is the length of the output area. h must be equal to or greater than f/2 or (f+1)/2.

Control Parameter

This parameter consists of four hexadecimal digits. Digits 1 and 2 are not used. The third digit indicates whether or not the case is to be initialized before conversion begins:

0 — Initialize case
1 — Do not alter case

The fourth digit indicates the direction of conversion:

0 — PTTC/8 to EBCDIC
1 — EBCDIC to PTTC/8

Input

Input (either PTTC/8 or EBCDIC characters, as specified by the control parameter) starts in location INPUT. Characters are packed two per 16-bit computer word in both codes.

Output

Output is either EBCDIC or PTTC/8 characters starting in OUTPT. Characters in either code are in packed format. The subroutine starts processing at location INPUT.

If the output is in EBCDIC, overlap of the input and output areas is possible if the address INPUT is equal to or greater than the address OUTPT.

If the output is in PTTC/8, overlap of the input and output areas is not recommended because the number of output characters might be greater than the number of input characters.

Character Count

This parameter specifies the number of PTTC/8 or EBCDIC characters in the input area. The count must include case shift characters even though they will not appear in the output. Because the input is packed, the character count will not be equal to the number of binary words in the input area. If an odd number of output characters is produced, bits 8-15 of the last word used in the output area are set to a space character if the output is EBCDIC, or to a delete character if the output is PTTC/8.

There is no danger of overflowing the output area if the number of words in a PTTC/8 output area is equal to the number of characters in the input area.

Errors Detected

Any input character that is not marked with an asterisk in Appendix D is an error.

Subroutine Operation

If the input is in PTTC/8 code, all control characters (except case shift (LC or UC) characters) are converted to output. Case shift characters only define the case mode of the graphic characters that follow.

If the initialize option is selected, the case is set to lower. All characters are interpreted as lower case characters until an upper case shift (UC) character is encountered. If the do-not-alter option is selected, the case remains set according to the last case shift character encountered in the previous LIBF message.

If the input is in EBCDIC, all data and control characters are converted to output. The user should not specify case shifting in his input message; this is handled automatically by the PAPEB subroutine.

Case shift characters are inserted in a PTTC/8 output message where needed to define certain graphic characters that have the same binary value and are differentiated only by a case mode character. For example, the binary value 0101 1011 (5B), is interpreted as a $ in lower case and an ! in upper case (see Appendix D).

If the initialize option is selected, the case shift character needed to interpret the first graphic character is inserted in the output message and the case mode is initialized for that mode. If the do-not-alter option is selected, the case mode remains set according to the last case shift character required in the previous LIBF message, i.e., no case shift is forced.

If a case shift character appears in the input message, it is output but does not affect the case mode. If it is an upper case shift (UC) and the next input character requires an upper case shift, the subroutine still inserts an upper case shift into the message, i.e., two UC characters will appear in the output message.

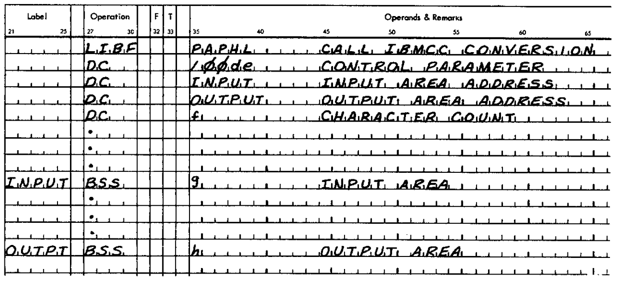
The conversion is halted when the character count is decremented to zero or when a new line (NL) control character is read.

PAPHL

This subroutine converts PTTC/8 subset to IBM Card Code subset or IBM Card Code subset to PTTC/8 subset. The relationship of the two codes for converting PTTC/8 to IBM Card Code is illustrated below.

I/O Locations	Conversion Data	Bits in Core Storage 0 ←——→ 15			
INPUT	UC J	0000	1110	0101	0001
	S T	0011	0010	0010	0011
OUTPT	J	0101	0000	0000	0000
OUTPT +1	S	0010	1000	0000	0000
OUTPT +2	T	0010	0100	0000	0000

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
21  25 27  30     32 33 35   40   45   50   55   60   65
      LIBF         PAPHL     CALL IBMCC CONVERSION
      DC           /ØØde     CONTROL PARAMETER
      DC           INPUT     INPUT AREA ADDRESS
      DC           OUTPUT    OUTPUT AREA ADDRESS
      DC           f         CHARACTER COUNT
      .
      .
      .
INPUT BSS          g         INPUT AREA
      .
      .
      .
OUTPT BSS          h         OUTPUT AREA
```

where

d is the case initialization digit,

e indicates the direction of conversion,

f indicates the character count,

g is the length of the input area. g must be equal to or greater than f if e is 0. If e is 1, g must be equal to f/2, or (f+1)/2 if f is odd.

h is the length of the output area. h must be equal to or greater than f/2 ((f+1)/2 if f is odd) if e is 0. If e is 1, h must be equal to or greater than f.

Control Parameter

This parameter consists of four hexadecimal digits. Digits 1 and 2 are not used. The third digit indicates whether or not the case is to be initialized before conversion begins:

0 - Initialize case
1 - Do not alter case

The fourth digit indicates the type of conversion:

0 - PTTC/8 to IBM Card Code
1 - IBM Card Code to PTTC/8

Input

Input is either PTTC/8 or IBM Card Code characters (as specified by the control parameter starting in location INPUT. PTTC/8 characters are packed two per binary word; IBM Card Code characters are not packed.

Output

Output is either IBM Card Code or PTTC/8 code characters starting in location OUTPT. PTTC/8 codes are packed two per binary word; IBM Card Code characters are not packed.

If the conversion is IBM Card Code input to PTTC/8 output, the input area may overlap the output area if the address INPUT is equal to or greater than the address OUTPT. Case shift characters are inserted in the output message where needed to define certain graphic characters (see PAPEB).

If the conversion is PTTC/8 input to IBM Card Code output, the input area may 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. The subroutine starts processing at location INPUT.

Character Count

This parameter specifies the number of PTTC/8 or EBCDIC characters in the input area. The count must include case shift characters, even though they will not appear in the output. Because the input may be packed, the character count may not be equal to the number of binary words in the input area.

There is no danger of overflowing the output area if the number of words in the output area is equal to the number of characters in the input area.

Errors Detected

Any input character not marked by an asterisk in Appendix D is an error.

Subroutine Operation

Case and shift character handling is described under PAPEB.

If an odd number of PTTC/8 output characters is produced, bits 8-15 of the last used word in the output area are set to a delete character.

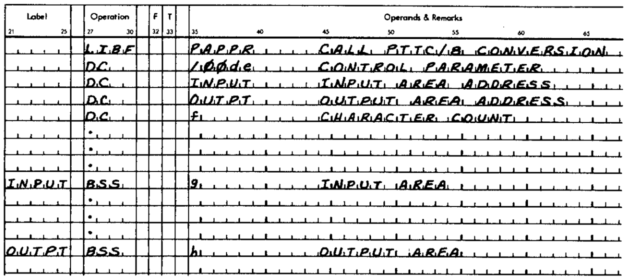
The conversion is halted when the character count is decremented to zero or when a new line (NL) control character is read.

PAPPR

This subroutine converts PTTC/8 subset to either Console Printer or 1403 Printer code. The conversion to 1403 Printer code is illustrated below.

I/O Locations	Conversion Data		Bits in Core Storage 0◄──────►15
INPUT	UC	J	0000 1110 0101 0001
INPUT + 1	LC	$	0110 1110 0101 1011
OUTPT	J	$	0101 1000 0110 0010

Calling Sequence



```
Label   Operation  F T  Operands & Remarks
21   25 27    30  32 33 35   40   45   50   55   60   65
        LIBF            PAPPR      CALL PTTC/8 CONVERSION
        DC              /ØØde      CONTROL PARAMETER
        DC              INPUT      INPUT AREA ADDRESS
        DC              OUTPT      OUTPUT AREA ADDRESS
        DC              f          CHARACTER COUNT
        .
        .
        .
INPUT   BSS             g          INPUT AREA
        .
        .
        .
OUTPT   BSS             h          OUTPUT AREA
```

where

d is the case initialization digit,

e is the output printer code digit,

f is the number of characters in the input area to be converted,

g is the length of the input area. g must be equal to or greater than f/2 if the character count is even, (f + 1)/2 if the character count is odd.

h is the length of the output area. h must be equal to or greater than f/2, minus the number of paper tape control characters in the input area, plus 1 if the result is odd.

Control Parameter

This parameter consists of four hexadecimal digits. Digits 1 and 2 are not used. The third digit indicates whether or not the case is to be initialized before conversion begins:

0 - Initialize case
1 - Do not alter case

The fourth digit determines the output printer code.

0 - Console Printer code
1 - 1403 Printer code

Input

Input consists of PTTC/8 characters starting in location INPUT. PTTC/8 characters are packed two per binary word. All control characters except case shift (LC or UC) characters are converted to output. Case shift characters are used only to define the case mode of the graphic characters that follow.

Output

Output consists of either Console Printer or 1403 Printer characters starting in location OUTPT. This code is packed two characters per binary word. If overlap of the input and output areas is desired, the address INPUT must be equal to or greater than the address OUTPT. This is necessary because the subroutine starts processing at location INPUT.

Character Count

This parameter specifies the number of PTTC/8 characters in the input area. The count must include case shift characters, even though they do not appear in the output. Because the input is packed, the character count is not equal to the number of binary words in the input area.

If an odd number of output characters is produced, bits 8-15 of the last used word in the output area are set to a space character.

The conversion is halted when the character count is decremented to zero or when a new line (NL) control character is read.

Errors Detected

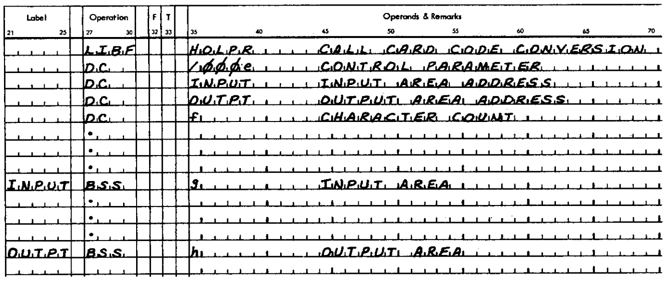
Any input character not marked by an asterisk in Appendix D is an error.

HOLPR

This subroutine converts IBM Card Code subset to either Console Printer or 1403 Printer code. The conversion to 1403 Printer code is illustrated below.

I/O Locations	Conversion Data	Bits in Core Storage 0 ◄──────► 15
INPUT	J	0101 0000 0000 0000
INPUT + 1	,	0010 0100 0010 0000
OUTPT	J,	0101 1000 0001 0110

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
21  25 27     30  32 33 35          45
       LIBF            HOLPR       CALL CARD CODE CONVERSION
       DC              1ØØØe       CONTROL PARAMETER
       DC              INPUT       INPUT AREA ADDRESS
       DC              OUTPT       OUTPUT AREA ADDRESS
       DC              f           CHARACTER COUNT
       •
       •
       •
INPUT  BSS             g           INPUT AREA
       •
       •
       •
OUTPT  BSS             h           OUTPUT AREA
```

where

e is the output printer code digit,

f is the number of characters in the input area to be converted,

g is the length of the input area. g must be equal to or greater than f.

h is the length of the output area. h must be equal to or greater than f/2.

Control Parameter

This parameter consists of four hexadecimal digits. Digits 1-3 are not used. The fourth digit determines the output printer code.

0 - Console Printer code
1 - 1403 Printer code

Input

Input consists of IBM Card Code characters, starting in location INPUT. The characters are not packed.

Output

Output consists of either Console Printer or 1403 Printer characters, starting in location OUTPT. The code is packed two characters per binary word.

The input area may overlap the output area if the address INPUT is equal to or greater than the address OUTPT. The subroutine starts processing at location INPUT.

Character Count

This number specifies the number of IBM Card Code characters to be converted and is equal to the number of words in the input area. If an odd count is specified, bits 8-15 of the last word used in the output area are not altered.

Errors Detected

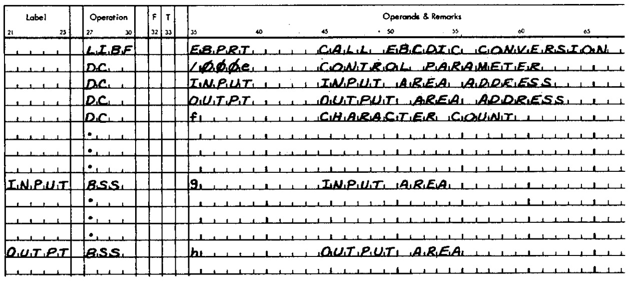
Any input character not marked with an asterisk in Appendix D is an error.

EBPRT

This subroutine converts EBCDIC subset to either Console Printer or 1403 Printer Code. The conversion to 1403 Printer code is shown below.

I/O Locations	Conversion Data	Core Storage Bits 0 ◄──────► 15
INPUT	LE	1101 0011 1100 0101
INPUT + 1	ES	1100 0101 1110 0010
OUTPUT	LE	0001 1010 0110 1000
OUTPT + 1	ES	0110 1000 0000 1101

Calling Sequence



```
Label  Operation  F T  Operands & Remarks
21  25 27     30  32 33 35          45
       LIBF            EBPRT       CALL EBCDIC CONVERSION
       DC              1ØØØe       CONTROL PARAMETER
       DC              INPUT       INPUT AREA ADDFESS
       DC              OUTPT       OUTPUT AREA ADDRESS
       DC              f           CHARACTER COUNT
       •
       •
       •
INPUT  BSS             g           INPUT AREA
       •
       •
       •
OUTPT  BSS             h           OUTPUT AREA
```

where

e is the output printer code digit,

f is the number of characters in the input area to be converted,

g is the length of the input area. g must be equal to or greater than f/2.

h is the length of the output area. h must be equal to or greater than f/2.

Control Parameter

This parameter consists of four hexadecimal digits. Digits 1-3 are not used. The fourth digit determines the output printer code.

0 — Console Printer code
1 — 1403 Printer code

Input

Input consists of EBCDIC characters starting in location INPUT. EBCDIC characters are packed two per word.

Output

Output consists of either Console Printer or 1403 Printer code starting in location OUTPT. The code is packed two characters per binary word.

The address INPUT must be equal to or greater than the address OUTPT if overlap of the input and output areas is desired. The subroutine starts processing at location INPUT.

Character Count

This parameter specifies the number of EBCDIC characters to be converted. This count is not equal to the number of words in the input area. If an odd count is specified, bits 8-15 of the last word used in the output area are not altered; however, these bits may cause print checks if they comprise an illegal character.

Errors Detected

Any input character not marked with an asterisk in Appendix D is an error.

BIDEC

This subroutine converts a 32-bit binary value to its decimal equivalent in ten IBM Card Code numeric characters and one sign character. The conversion is illustrated below.

I/O Locations	Conversion Data	Core Storage Bits 0 ◄──► 15
Accumulator	+0016777218	0000 0001 0000 0000
Extension		0000 0000 0000 0010

I/O Locations	Conversion Data	Core Storage Bits 0 ◄──► 15
OUTPT	+	1000 0000 1010 0000
OUTPT + 1	0	0010 0000 0000 0000
OUTPT + 2	0	0010 0000 0000 0000
OUTPT + 3	1	0001 0000 0000 0000
OUTPT + 4	6	0000 0000 1000 0000
OUTPT + 5	7	0000 0000 0100 0000
OUTPT + 6	7	0000 0000 0100 0000
OUTPT + 7	7	0000 0000 0100 0000
OUTPT + 8	2	0000 1000 0000 0000
OUTPT + 9	1	0001 0000 0000 0000
OUTPT + 10	8	0000 0000 0010 0000

Calling Sequence

Label 21 25	Operation 27 30	F 32	T 33	Operands & Remarks 35 40 45 50 55 60 65
	LIBF			BIDEC CALL BINARY CONVERSION
	DC			OUTPUT OUTPUT AREA ADDRESS
	•			
	•			
	•			
OUTPT	BSS			11 OUTPUT AREA

Input

Input is a 32-bit binary value in the Accumulator and Extension.

Output

Output is an IBM Card Code sign character (+ or -) in location OUTPT, and ten IBM Card Code numeric characters in OUTPT+1 through OUTPT+10.

Errors Detected

The BIDEC subroutine does not detect errors.

DECBI

This subroutine converts a decimal value consisting of ten IBM Card Code numeric characters and a sign character to a 32-binary word. This subroutine is the opposite of the BIDEC subroutine (see above) except that fewer than ten characters may be specified.

Calling Sequence

```
Label  Operation  F T  Operands & Remarks
       LIBF            DECBI     CALL DECIMAL CONVERSION
       DC              INPUT     INPUT AREA ADDRESS
       DC              WDCNT     WORD COUNT ADDRESS
       .
       .
       .
WDCNT  DC              a         WORD COUNT
       .
       .
       .
INPUT  BSS             b         INPUT AREA
```

where

a is the number of characters to be converted not including the sign character,

b is the length of the input area. b must be equal to at least a plus 1.

Input

Input is an IBM Card Code sign character in location INPUT, the address (WDCNT) of the number of characters (1 to 10) to be converted, and specified number of characters in IBM Card Code in locations INPUT+1 through INPUT+N (where N = 1, 2,...10).

Output

Output is a 32-bit binary word, containing the converted value, in the Accumulator and Extension.

Errors Detected

Any of the following conditions causes the Overflow indicator to be turned on, the Carry indicator to be turned off, and an immediate exit to be made back to the caller:

1. Any sign other than a plus, minus, blank, or ampersand.
2. Any character other than a space or 0 through 9.
3. Any converted value greater than +2,147,483,647 or less than -2,147,483,648.

ZIPCO

This subroutine supplements all standard conversions except those involving PTTC/8 code. It offers the user the option of supplying his own conversion tables and codes. ZIPCO uses direct table access and is considerably faster than the other conversion subroutines.

Calling Sequence

```
Label  Operation  F T  Operands & Remarks
       LIBF            ZIPCO     CALL SPECIAL CONVERSION
       DC              /bcde     CONTROL PARAMETER
       DC              INPUT     INPUT AREA ADDRESS
       DC              OUTPT     OUTPUT AREA ADDRESS
       DC              f         CHARACTER COUNT
       CALL            i
       .
       .
       .
INPUT  BSS             g         INPUT AREA
       .
       .
       .
OUTPT  BSS             h         OUTPUT AREA
```

where

b is the input code digit,

c is the packed input digit,

d is the output code digit,

e is the packed output digit,

f is the number of characters to be converted,

g is the length of the input area,

h is the length of the output area,

j is the name of the conversion table to be used. This CALL is not executed; however, it is required following the character count parameter to cause the loading of the desired conversion table, provide the address of that table to ZIPCO, and provide information required by ZIPCO for the return to the calling program.

Control Parameter

This parameter consists of four hexadecimal digits as follows.

Digit 1	1 for 12-bit IBM Card Code input 0 for all other types of input
Digit 2	1 for unpacked input 0 for packed input
Digit 3	1 for 12-bit IBM Card Code output 0 for 8-bit IBM Card Code and all other types of output
Digit 4	1 for unpacked output 0 for packed output

Input

Input consists of packed or unpacked characters in the code specified by the conversion table and starting at location INPUT.

Output

Output consists of packed or unpacked characters in the code specified by the conversion table and starting at location OUTPT.

Character Count

This parameter specifies the number of input characters to be converted. If an odd count is specified with packed input, bits 8-15 of the last word used in the output area are not altered.

Table

The type of conversion is determined by the table called with ZIPCO. The user may call one of the IBM-supplied conversion tables or he may supply his own.

The following IBM-supplied System Library tables may be called with ZIPCO.

EBCCP	-	EBCDIC to Console Printer Code.
EBHOL	-	EBCDIC to IBM Card Code.
EBPT3	-	EBCDIC to 1403 Printer code.
CPEBC	-	Console Printer code to EBCDIC.
CPHOL	-	Console Printer code to IBM Card Code.
CPPT3	-	Console Printer code to 1403 Printer code.
HLEBC	-	IBM Card Code to EBCDIC.
HOLCP	-	IBM Card Code to Console Printer code.
HLPT3	-	IBM Card Code to 1403 Printer code.
PT3EB	-	1403 Printer code to EBCDIC.
PT3CP	-	1403 Printer code to Console Printer Code.
PTHOL	-	1403 Printer code to IBM Card Code.

Each conversion table consists of 256 characters--128 words with two 8-bit characters per word. The seven low-order bits of the character to be converted (input character) are used as an address. The address designates the position in the table of the corresponding conversion character. The high-order bit (bit 0) of the input character designates which half of the table word is to be used. When bit 0 is 1, the left half of the word is used. When bit 0 is 0, the right half of the word is used. All dummy entries of the IBM-supplied tables contain the code for a blank.

The following is an example of the conversion performed by ZIPCO. The tables show (1) the input EBCDIC values, (2) the table EBPT3 used for the conversion, and (3) the output characters in 1403 Printer code.

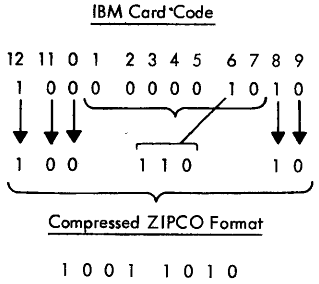
Input Location	Value
INPUT	1111 0010 0111 0010
INPUT + 1	0000 0000 1000 0000
INPUT + 2	0111 1111 1111 1111

Table Location	Value
EBPT3	0111 1111 0111 1111
EBPT3 + 1	0111 1111 0111 1111
.	.
.	.
.	.
EBPT3 + 113	0000 0001 0111 1111
.	.
.	.
.	.
EBPT3 + 127	0111 1111 0111 1111

Output Location	Value	1403 Print Character
OUTPT	0000 0001 0111 1111	2, b
OUTPT + 1	0111 1111 0111 1111	b, b
OUTPT + 2	0111 1111 0111 1111	b, b

When 12-bit IBM Card Code is specified as input (or output), ZIPCO performs a packing (or unpacking) of the character to 8-bits (or 12 bits). The 1-7 row punches on the card are expressed as a 3-bit hexadecimal number (there can never be more than one punch between the 1 and 7 row). In this format a 1 punch would be expressed as 001, a 7 punch as 111. The punches in the other card rows: 12, 11, 0, 8, and 9, are transferred directly.

For example, take the IBM Card Code character "+" which is a 12, 6, 8 punch.



Errors Detected

No errors are detected by ZIPCO.

The IBM 1130 Subroutine/System 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 6 lists the arithmetic and functional subroutines that are included in the Subroutine/System Library.

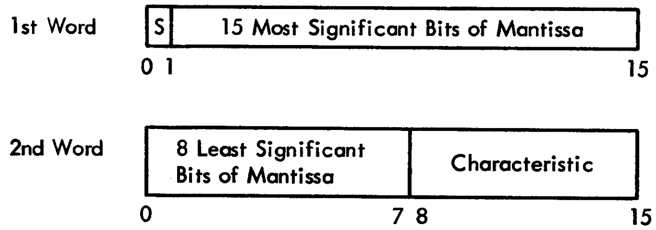
REAL DATA FORMATS

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:

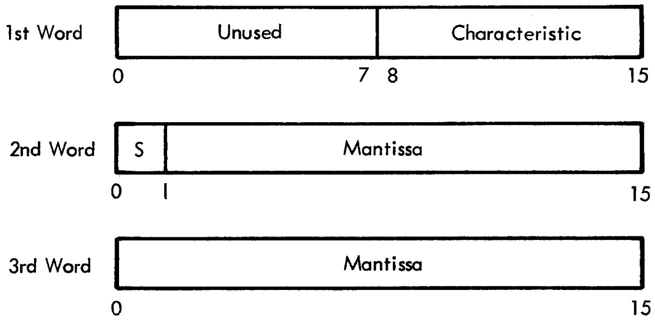

1st Word
S
15 Most Significant Bits of Mantissa
0 1
15
2nd Word
8 Least Significant Bits of Mantissa
Characteristic
0
7 8
15

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 if the number is negative) 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 consists of up to 31 significant bits with a binary exponent ranging from -128 to +127; however, normalization can, in some cases, cause the loss of 1 bit of significance.

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 if the number is negative).

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.

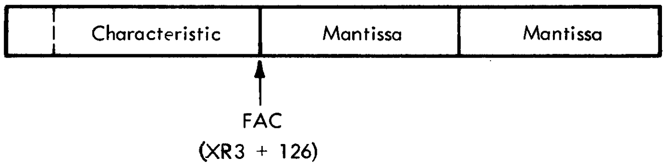
Table 6. Arithmetic and Functional Subroutines

SUBROUTINE	NAME	
Real (Floating Point)	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
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 an 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 Extension. The calling program should provide for this if the contents are significant. When execution begins, all three words of FAC contain zeros.

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 that 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 6 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, FMPYX, EMPY or EMPYX
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)

NOTE: On a divide by zero, the divide check indicator is turned on, the dividend is not changed, and the dividend remains in 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 Binary to Decimal

CALL	FBTD
DC	LDEC
Input	Real number in FAC
Result	A string of EBCDIC-coded data starting at location LDEC. Each EBCDIC character occupies the rightmost 8 bits of a word. The last character of the string is a blank.

The output format is exactly as follows:

sd. ddddddddEsddb

where:

s represents a sign (plus or minus)
d represents one of the decimal digits 0-9
b represents a blank

Real Decimal to Binary

CALL	FDTB
DC	LDEC
Input	A string of EBCDIC coded data at location LDEC. Each EBCDIC character occupies the rightmost 8 bits of a word. The first character of the input must be the sign (plus or minus). Following the sign, one to nine decimal digits (0-9) may be specified. The decimal point may appear before, within, or after the decimal digits. Immediately after the last decimal digit (or decimal point), the exponent is specified as follows.

Esddb

where:

s represents the sign of the exponent (plus or minus)
d represents one of the decimal digits (0-9)
b represents a blank (the blank is required to indicate the end of the string)

No embedded blanks may appear in the input string as the first blank is interpreted as the end of the data.

Result	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)

NOTE: On a divide by zero, the divide check indicator is turned on, the dividend is not changed, and the dividend remains in 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 calling 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. When execution begins, all three words contain zeros.

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).
2. 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 and remains in FAC.

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.

End of File (DM2 System Only). When the end-of-file record in the unformatted I/O area is read, a bit is ORed into position 2 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

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

for the range

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

EEXP

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

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

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

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

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

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

$$\frac{x}{2\pi} = n + y$$

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

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

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 subroutine for arctangent is built around a polynomial, F (z), that approximates Arctan (z) in the range $-.23 \leq z \leq .23$. The 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

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

and k is determined so that

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

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

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

where

$$a_1 = .333329573$$
$$a_2 = .199641035$$
$$a_3 = .131779888$$

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

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

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

Then for both extended 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

$$G(z) = b_0 + b_2z^2 + b_4z^4 + b_6z^6 + b_8z^8$$

where

$b_0 = 2.0$

$b_2 =\ .666666564181$

$b_4 =\ .400018840613$

$b_6 =\ .28453572660$

$b_8 =\ .125$

Standard Precision

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

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

$$F(z) = a_0 + a_1z + a_2z^2 + a_3z^3 + a_4z^4 + a_5z^5$$

where

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

Standard Precision

$$F(z) = a_0 + a_1z + a_2z^2 + a_3z^3 + a_4z^4$$

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

$$x \geq 32 \qquad \text{Tanh}(x) = 1$$
$$x \leq -32 \qquad \text{Tanh}(x) = -1$$

REAL BASE TO REAL EXPONENT

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

therefore:

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

The IBM 1130 Subroutine Library and the System Library include 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 a user's program.

DUMP SELECTED DATA ON CONSOLE PRINTER OR 1132 PRINTER

Two subroutines are available to select an area of core storage and dump it 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 format, using the WRTY0 subroutine

DMTD0 Dump on Console Printer in decimal format, using the WRTY0 subroutine

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

DMPD1 Dump on 1132 Printer in decimal format, 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 calling 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 indicators (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      HHHH  (±DDDDD)    HHHH (±DDDDD)
Overflow  Carry   Accumulator       Extension

HHHH  (±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 PROGRAM LOAD 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 calling program.

DM2 SYSTEM

The System Library contains a group of subroutines that perform various system utility functions. These subroutines, with the exception of SYSUP which can be called by the user, are intended for system use only. Under normal circumstances, they should not be deleted from the System Library.

The subroutines in the group are:

FLIPR	-	LOCAL/SOCAL overlay subroutine
RDREC	-	Read *ID record
CALPR	-	Call system print
FSLEN	-	Fetch phase IDs or,
FSYSU	-	Fetch system subroutine (FSYSU is an alternate entry point to FSLEN)
SYSUP	-	DCOM update

FLIPR (LOCAL/SOCAL OVERLAY)

The System Library contains a flipper subroutine (FLIPR) which is used to call LOCAL (load on call) and SOCAL (system load on call) subroutines into core storage. FLIPR is used with DISKZ, DISK1, or DISKN.

FLIPR passes the total word count to DISKZ, DISK1, or DISKN to fetch the LOCAL. When a LOCAL subroutine is called, control is passed to the flipper, which reads the LOCAL into core storage if it is not already in core and transfers control to it. All LOCALs in a given core load are executed from the same core storage locations; each LOCAL overlays the previous one. FLIPR fetches SOCALs in the same manner as LOCALs.

RDREC (READ *ID RECORD)

This subroutine is called by the Disk Maintenance Programs to read the *ID (disk label) record. This subroutine is intended for system use only.

CALPR (CALL SYSTEM PRINT)

This subroutine calls FSLEN to bring the system print subroutine into core storage for the purpose of printing one or more lines on the principal printer. This subroutine is intended for system use only.

FSLEN (FETCH PHASE IDs AND FETCH SYSTEM SUBROUTINE)

This subroutine has two entry points. They are FSLEN and FSYSU.

- FSLEN (Fetch Phase IDs from SLET)

This entry point obtains the requested phase ID headers from SLET.

- FSYSU (Fetch System Subroutine)

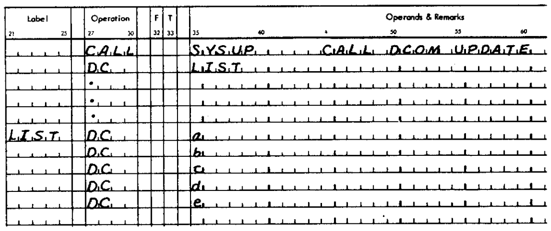
Fetches the requested system subroutine into core storage.

This subroutine is intended for system use only.

SYSUP (DCOM UPDATE)

Whenever a core load requires changing disk cartridges during the job, SYSUP must be called to update DCOM on the master cartridge (logical drive 0) with the IDs and DCOM information from all satellite cartridges mounted on the system. The cartridges are specified in the list or array in the SYSUP calling sequence. The list or array must be exactly five words long or be ended by a zero (not both).

The Assembler-Language calling sequence for SYSUP is:


Label
21
25
Operation
27
30
F
32
T
33
Operands & Remarks
35
40
45
50
55
60

```
      CALL SYSUP      CALL DCOM UPDATE
      DC   LIST
      .
      .
      .
LIST  DC   a
      DC   b
      DC   c
      DC   d
      DC   e
```

where

List is the address of the table of requested cartridge IDs,

a is the ID of the master cartridge on the system,

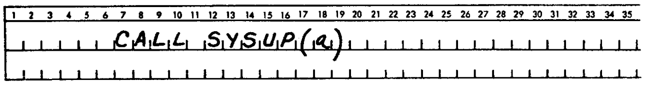
b is the ID of the first satellite cartridge on the system, c is the ID of the second satellite cartridge on the system,

d is the ID of the third satellite cartridge on the system,

e is the ID of the fourth satellite cartridge on the system.

If a is 0, the master cartridge is the same as before.

The FORTRAN calling sequence for SYSUP is:



```
      CALL SYSUP(a)
```

where

a is the name of the last item in an array containing the IDs of the satellite cartridges on the system. The last entry in the array may be 0, in which case the master cartridge remains unchanged.

For example:

CALL SYSUP A (5)

The array is stored in reverse order.

A(5) DC
A(4) DC
A(3) DC
A(2) DC
A(1) DC

Thus A(5) is the entry for logical 0, the master cartridge.

SYSUP messages are listed in the publication 1130 Monitor Programming and Operator's Guide (Form C26-3717). SYSUP execution is terminated if an error printout occurs.

DM1 SYSTEM

OVERLAY SUBROUTINES (FLIPPERS)

The Monitor subroutine library contains two flipper subroutines which are used to call LOCAL (load on call) subroutines into core storage. FLIP0 is used with DISK0 and DISKZ, and FLIP1 is used with DISK1 and DISKN. FLIP0 reads a LOCAL into storage one sector at a time, whereas FLIP1 passes the total word count to DISK1 or DISKN and that subroutine reads in the entire LOCAL. When a LOCAL subroutine is called, control is passed to the flippers which read the LOCAL into core storage if it is not already in core and transfer control to it. All LOCALs in a given core load are executed from the same core storage locations; each LOCAL overlays the previous one.

SYSTEM LIBRARY MAINLINE PROGRAMS (DM2 SYSTEM)

The 1130 System Library mainline programs provide the user with the ability to perform disk maintenance and paper tape utility functions by requesting execution of the appropriate program directly through the job stream.

The calling sequences for the System Library mainline programs are listed below. The operating procedures and error messages are contained in the IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide (Form C26-3717).

DISK MAINTENANCE PROGRAMS

The disk maintenance programs are mainline programs and subroutines that are a part of the System Library and that initialize and modify disk cartridge IDs, addresses, and tables required by the DM2 system. Normally, they should never be deleted from the System Library.

The disk maintenance programs are:

IDENT	-	Print Cartridge ID
DISC	-	Satellite Disk Initialization*
ID	-	Change Cartridge ID
COPY	-	Disk Copy
ADRWS	-	Write Sector Addresses in Working Storage
DLCIB	-	Delete CIB
DSLET	-	Dump System Location Equivalence Table
MODIF	-	System Maintenance Program

*All new cartridges are initialized using the stand-alone program DCIP (see IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide, Form C26-3717).

IDENT (Print Cartridge ID)

This program prints the ID and physical drive number of each cartridge mounted on the system.

IDENT prints all cartridge IDs regardless of validity (JOB card processing only recognizes valid IDs).

The calling sequence for IDENT is:

```
// XEQ IDENT
```

DISC (Satellite Disk Initialization)

This program re-initializes up to four satellite cartridges -- all but the master cartridge. It writes the sector addresses, defective cylinder addresses, a cartridge ID, a LET, a DCOM, and a CIB on each cartridge being re-initialized.

DISC overrides all cartridge IDs specified on the JOB card except the master cartridge ID.

The calling sequence for DISC is:

```
// XEQ DISC
*IDFID1,TID1,FID2,TID2,...FIDn,TIDn
```

where

FID1 through FIDn are the IDs currently on the satellite cartridges to be re-initialized (identified by IDENT or a JOB record),

TID1 through TIDn are the IDs to be written on the satellite cartridges by this program. A valid cartridge ID is a number between /0001 and /7FFF.

ID (Change Cartridge ID)

This program changes the ID on up to four satellite cartridges.

The calling sequence for ID is:

```
// XEQ ID
*IDFID1,TID1,FID2,TID2,...FIDn,TIDn
```

where

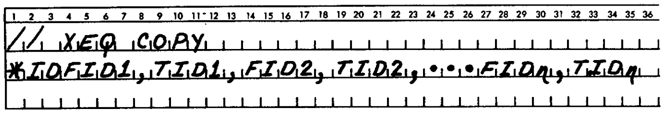
FID1 through FIDn are the IDs currently on the satellite cartridges being changed (these IDs must be in the same logical order as the entries on the JOB card),

TID1 through TIDn are the new IDs to be written on the selected satellite cartridges.

COPY (Disk Copy)

This program copies the contents (except the defective cylinder table and the cartridge ID) of one cartridge onto another. The copy ID (word 5 of sector 0) is incremented by one prior to being written on the new cartridge.

The calling sequence for COPY is:



```
// XEQ COPY
*IDFID1,TID1,FID2,TID2,•••FIDn,TIDn
```

where

FID1 through FIDn are the IDs of the cartridges to be copied.

TID1 through TIDn are the IDs of the cartridge onto which the copies are to be made.

If multiple copies are to be made from a single master, FID1 through FIDn will all contain the same ID.

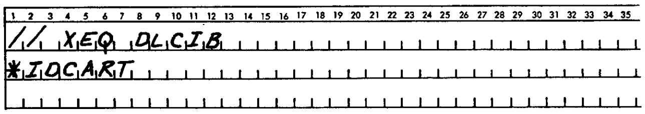
ADRWS (Write Sector Addresses in Working Storage)

This program, linked to from DUP on detection of the DUP control record DWADR, writes sector addresses on all sectors of Working Storage on the disk cartridge specified on the DWADR control record (see DUP in the 1130 Monitor Programming and Operator's Guide (Form C26-3717)).

DLCIB (Delete Core Image Buffer)

This program deletes the CIB from a non-system cartridge. If a user area is defined, the user area is moved two cylinders closer to cylinder 0. The new addresses of disk areas moved as the result of the deletion of the CIB are reflected in DCOM on the master cartridge, on the non-system cartridge from which the CIB is deleted, and in COMMA.

The calling sequence for DLCIB is:



```
// XEQ DLCIB
*IDCART
```

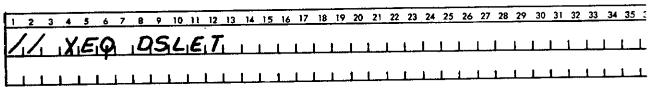
where

CART is the ID of the non-system cartridge from which the CIB is to be deleted.

DSLET (Dump System Location Equivalence Table)

This program dumps the contents of SLET on the principal printer. Each entry printed consists of a symbolic name, a phase ID, a core address, a word count, and a disk sector address. A SLET dump is listed in the publication 1130 Monitor Programming and Operator's Guide (Form C26-3717).

The calling sequence for DSLET is:

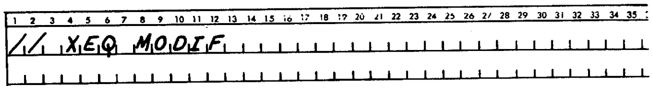


```
// XEQ DSLET
```

MODIF (System Maintenance Program)

Included in the system library is a system maintenance program, MODIF, that provides the user with the ability to update the monitor system on the master cartridge. This program makes changes to the version and modification level word in DCOM, the supervisor, DUP, FORTRAN compiler, assembler, and or system library. A card deck or paper tape containing corrections to update the monitor system to the latest version and modification level is supplied by IBM. Every modification must be run to update the version and modification level, even if the affected program has been deleted from the system.

The calling sequence for MODIF is:



```
// XEQ MODIF
```

PAPER TAPE UTILITY (PTUTL)

This program accepts input from the keyboard or the 1134 paper tape reader and provides output on the console printer and/or the 1055 paper tape punch.

PTUTL allows changes and/or additions to FORTRAN and assembler language source records as well as monitor control records.

The calling sequence for PTUTL is:

```
// XEQ PTUTL
```

The section on Writing ISSs and ILSs for the DM2 system will be found in the IBM 1130 Disk Monitor System, Version 2, Programming and Operator's Guide (Form C26-3717).

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 a LIBF rather than a 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 subroutines, are shown in Table 7.
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 BSI to the ISS at the ISS entry point plus n (see Table 7). The ISS must return to the ILS via a BSC instruction (not a BOSC).

INTERRUPT LEVEL SUBROUTINES

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

1. Precede the subroutine with an ILS statement.
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 subroutine 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 7. DM1 and C/PT System ISS/ILS Correspondence

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

$$\left.\begin{array}{l}\text{ILSW Bit 15 word}\\ \text{ILSW Bit 14 word}\\ \quad\cdot\\ \quad\cdot\\ \quad\cdot\\ \text{ILSW Bit 0 word}\end{array}\right\}\ \text{ISS Branch Table}$$

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.

During execution, the ISS Branch Table contains core addresses. It may be used with an indirect 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 equivalent to the bit position in the

ISS branch table. An indirect, indexed BSI may then be used to reach the appropriate ILS.

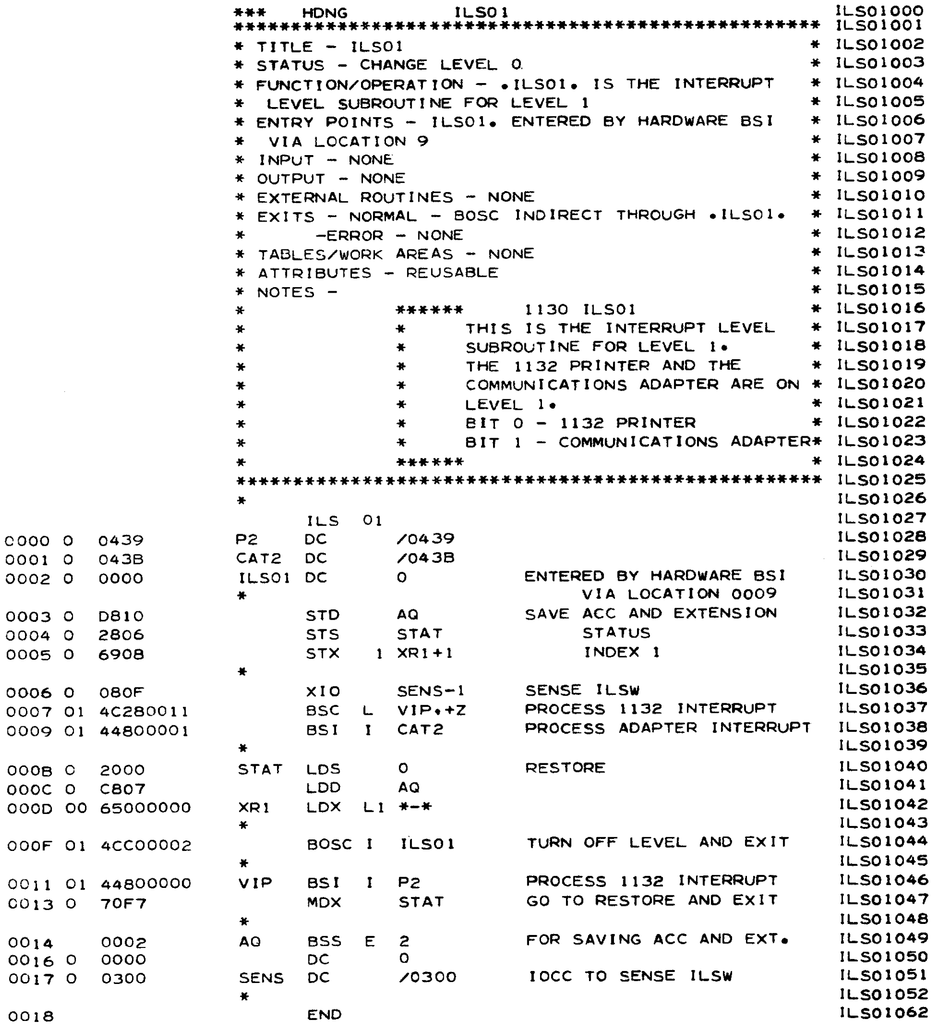
Each word in the ISS branch table has the following format:

Bits 0-7 — Increment added to the entry point named in the ISS statement to obtain the interrupt entry point in the ISS for this ILSW bit. (In IBM-written ISS subroutines, this increment is +4 for the primary interrupt level and +7 for the secondary interrupt level.)

Bits 8-15 — ISS number +51 for the ISS subroutine for this ILSW bit. This address should match word 13 of the compressed ISS header card.

3. The ILS entry point must immediately follow the ISS branch address table and must be a zero. The first zero word in the program is the end of the branch table and is also the entry point of the ILS. (The table must contain at least one entry.) The interrupt results in a BSI to the ILS entry point.
4. To clear the level, a user-written ILS, used with an IBM-supplied ISS, should exit via the return linkage with a BOSC instruction.

Sample ILS



```
                     ***   HDNG         ILS01                             ILS01000
                     **************************************************** ILS01001
                     * TITLE - ILS01                                    * ILS01002
                     * STATUS - CHANGE LEVEL 0                          * ILS01003
                     * FUNCTION/OPERATION - .ILS01. IS THE INTERRUPT    * ILS01004
                     *  LEVEL SUBROUTINE FOR LEVEL 1                    * ILS01005
                     * ENTRY POINTS - ILS01. ENTERED BY HARDWARE BSI    * ILS01006
                     *  VIA LOCATION 9                                  * ILS01007
                     * INPUT - NONE                                     * ILS01008
                     * OUTPUT - NONE                                    * ILS01009
                     * EXTERNAL ROUTINES - NONE                         * ILS01010
                     * EXITS - NORMAL - BOSC INDIRECT THROUGH .ILS01.   * ILS01011
                     *      -ERROR - NONE                               * ILS01012
                     * TABLES/WORK AREAS - NONE                         * ILS01013
                     * ATTRIBUTES - REUSABLE                            * ILS01014
                     * NOTES -                                          * ILS01015
                     *             ******      1130 ILS01               * ILS01016
                     *             *     THIS IS THE INTERRUPT LEVEL    * ILS01017
                     *             *     SUBROUTINE FOR LEVEL 1.        * ILS01018
                     *             *     THE 1132 PRINTER AND THE       * ILS01019
                     *             *     COMMUNICATIONS ADAPTER ARE ON  * ILS01020
                     *             *     LEVEL 1.                       * ILS01021
                     *             *     BIT 0 - 1132 PRINTER           * ILS01022
                     *             *     BIT 1 - COMMUNICATIONS ADAPTER* ILS01023
                     *             ******                               * ILS01024
                     **************************************************** ILS01025
                     *                                                    ILS01026
                           ILS  01                                        ILS01027
0000 0  0439         P2    DC       /0439                                 ILS01028
0001 0  043B         CAT2  DC       /043B                                 ILS01029
0002 0  0000         ILS01 DC       0          ENTERED BY HARDWARE BSI    ILS01030
                     *                               VIA LOCATION 0009    ILS01031
0003 0  D810               STD      AQ         SAVE ACC AND EXTENSION     ILS01032
0004 0  2806               STS      STAT            STATUS                ILS01033
0005 0  6908               STX    1 XR1+1           INDEX 1               ILS01034
                     *                                                    ILS01035
0006 0  080F               XIO      SENS-1     SENSE ILSW                 ILS01036
0007 01 4C280011           BSC  L   VIP.+Z     PROCESS 1132 INTERRUPT     ILS01037
0009 01 44800001           BSI  I   CAT2       PROCESS ADAPTER INTERRUPT  ILS01038
                     *                                                    ILS01039
000B 0  2000         STAT  LDS      0          RESTORE                    ILS01040
000C 0  C807               LDD      AQ                                    ILS01041
000D 00 65000000     XR1   LDX  L1 *-*                                    ILS01042
                     *                                                    ILS01043
000F 01 4CC00002           BOSC I   ILS01      TURN OFF LEVEL AND EXIT    ILS01044
                     *                                                    ILS01045
0011 01 44800000     VIP   BSI  I   P2         PROCESS 1132 INTERRUPT     ILS01046
0013 0  70F7               MDX      STAT       GO TO RESTORE AND EXIT     ILS01047
                     *                                                    ILS01048
0014    0002         AQ    BSS  E   2          FOR SAVING ACC AND EXT.    ILS01049
0016 0  0000               DC       0                                     ILS01050
0017 0  0300         SENS  DC       /0300      IOCC TO SENSE ILSW         ILS01051
                     *                                                    ILS01052
0018                       END                                            ILS01062
```

```
                    ***   HDNG    LIBF CARDO                             CRD00001
                          LIBR                                           CRD00002
0000      03059130  1130  ISS  01 CARDO    0     4                       CRD00003
                    **************************************************** CRD00004
                    *      THIS 1130 SUBROUTINE OPERATES THE 1442 CARD * CRD00005
                    *      READER PUNCH. IT INITIATES REQUESTED OPERA- * CRD00006
                    *      TIONS, PROCESSES ANY COLUMN OR OPERATION    * CRD00007
                    *      COMPLETE INTERRUPTS, AND AUTOMATICALLY      * CRD00008
                    *      INITIATES ERROR RECOVERY PROCEDURES.        * CRD00009
                    *                                                  * CRD00010
                    *      IDENTIFYING FEATURE - NO ERROR PARAMETER    * CRD00011
                    **************************************************** CRD00012
                    *             LOADER DEFINED LOCATIONS             * CRD00013
                    **************************************************** CRD00014
0000 0  695E        CARDO STX   1 CA30+1      LIBF ENTRANCE         (+0) CRD00015
0001 00 65800000    LINK  LDX  I1 0           LOADER STORES TV ADDR (+2) CRD00016
0003 0  7006              MDX     CA10                                   CRD00017
0004 0  0000        INT1  DC      0           COLUMN INTERRUPT      (+4) CRD00018
0005 01 4C0000DA          BSC  L  NT14                                   CRD00019
0007 0  0000        INT2  DC      0           OP CMPLTE INTERRUPT   (+7) CRD00020
0008 01 4C00009F          BSC  L  NT10                                   CRD00021
                    **************************************************** CRD00022
                    *             LIBF PROCESSING                      * CRD00023
                    **************************************************** CRD00024
                    *      THIS PORTION STORES CALLING SEQUENCE INFO   * CRD00025
                    *      AND CHECKS THE DEVICE STATUS BEFORE ANY I/O * CRD00026
                    *      OPERATION IS INITIATED. A CALLING ERROR OR  * CRD00027
                    *      NOT READY 1442 CAUSES AN ERROR EXIT TO      * CRD00028
                    *      LOCATION 41. IF THE OPERATION WILL CAUSE    * CRD00029
                    *      INTERRUPTS, THE ROUTINE IS SET BUSY AND THE * CRD00030
                    *      IOCS COUNTER IS INCREMENTED TO INDICATE     * CRD00031
                    *      INTERRUPT(S) PENDING.                       * CRD00032
                    **************************************************** CRD00033
000A 0  D07B        CA10  STO     TEMP        SAVE STATUS                CRD00034
000B 0  2856              STS     CA32                                   CRD00035
000C 0  6A54              STX   2 CA31+1                                 CRD00036
000D 0  C100              LD    1 0           X1= ADDR OF CALL+1         CRD00037
000E 0  180C              SRA     12          IS FUNCTION TEST           CRD00038
000F 01 4C200015          BSC  L  CA14,Z        NO                       CRD00039
0011 0  C078              LD      BUSY          YES, IS ROUTINE BUSY     CRD00040
0012 0  4818              BSC     +-                                     CRD00041
0013 0  7101              MDX   1 +1              NO, EXIT TO CALL+3     CRD00042
0014 0  7046              MDX     CA28            YES, EXIT TO CALL+2    CRD00043
0015 0  9077        CA14  S       D0004       IS FUNCTION LEGAL          CRD00044
0016 01 4C300070          BSC  L  CA40,Z-       NO, ERROR                CRD00045
0018 0  807A              A       H7003                                  CRD00046
0019 0  D00B              STO     CA20                                   CRD00047
001A 0  806D              A       CONST                                  CRD00048
001B 0  D0C7              STO     CA18                                   CRD00049
001C 0  C06D        CA15  LD      BUSY        IS ROUTINE BUSY            CRD00050
001D 01 4C20001C          BSC  L  CA15,Z        YES, WAIT TIL NOT        CRD00051
001F 0  0868        CA17  XIO     SENSE-1     IS DEVICE READY            CRD00052
0020 01 4C040072          BSC  L  CA42,E        NO, ERROR                CRD00053
0022 0  C066              LD      SENSE       SETUP CONTROL IOCC         CRD00054
0023 0  9075        CA18  S       SETUP                                  CRD00055
0024 0  D062              STO     INIT                                   CRD00056
0025 0  7000        CA20  MDX     CA20+1      WHAT IS FUNCTION           CRD00057
0026 0  7003              MDX     CA21          = GET                    CRD00058
0027 0  703D              MDX     CA36          = PUT                    CRD00059
0028 0  701B              MDX     CA25          = FEED                   CRD00060
0029 0  702B              MDX     CA26          = STK                    CRD00061
002A 0  9072        CA21  S       SETUP+4     GET FUNCTION               CRD00062
002B 0  D059              STO     COLM+1        SET UP READ I/O          CRD00063
002C 00 C5800001    CA21B LD   I1 1                                      CRD00064
002E 01 4C080070          BSC  L  CA40,+        = ERROR IF ZERO OR NEG   CRD00065
0030 0  D009              STO     CA22+1                                 CRD00066
0031 0  805A              A       D0001         SAVE WORD COUNT +1       CRD00067
0032 0  D061              STO     COUNT           BECAUSE DECREMENT IS   CRD00068
0033 0  D063              STO     RSTRT           BEFORE COLUMN READ     CRD00069
0034 0  905B              S       D0081                                  CRD00070
0035 01 4C300070          BSC  L  CA40,Z-       = ERROR IF OVER +81      CRD00071
0037 0  C101              LD    1 1                                      CRD00072
0038 0  D004              STO     CA23+1                                 CRD00073
0039 00 66000000    CA22  LDX  L2 0                                      CRD00074
003B 0  C050              LD      D0001                                  CRD00075
003C 00 D6000000    CA23  STO  L2 0             STORE +1 IN DATA AREA    CRD00076
003E 0  72FF              MDX   2 -1              (= NOT READ INDIC FOR  CRD00077
003F 0  70FC              MDX     CA23            SPEED CONVRT SBRT)     CRD00078
0040 0  C101        CA24  LD    1 1             SAVE DATA ADDRESS        CRD00079
0041 0  D042              STO     COLM                                   CRD00080
0042 0  D055              STO     RSTRT+1                                CRD00081
0043 0  7101              MDX   1 +1            SET X1 TO SKIP 2ND PARAM CRD00082
0044 0  0843        CA25  XIO     SENSE-1                                CRD00083
0045 0  1003              SLA     3             IS LAST CARD IND ON      CRD00084
0046 01 4C100050          BSC  L  CA25B,-         NO                     CRD00085
0048 0  C0DC              LD      CA20          IS FUNCTION GET OR FEED  CRD00086
0049 01 4C040050          BSC  L  CA25B,E         NO                     CRD00087
004B 0  1008              SLA     8             IS FUNCTION GET          CRD00088
```

```
004C 0  4808               BSC      +                                          CRD00089
004D 0  71FF               MDX   1 -1                  YES. SET XR1 = LIBF+1   CRD00090
004E 0  083B               XIO      FEED-1           EJECT CARD                CRD00091
004F 0  702C               MDX      CA43                                       CRD00092
0050 00 74010032     CA25B MDX   L  50.+1           INCREMENT IOCS COUNTER     CRD00093
0052 0  1000               NOP                                                 CRD00094
0053 0  C038               LD       D0001                                      CRD00095
0054 0  D035               STO      BUSY            SET ROUTINE BUSY           CRD00096
0055 0  C03F         CA26  LD       ERROR                                      CRD00097
0056 01 4C20005A           BSC   L  CA27.Z                                     CRD00098
0058 0  082D               XIO      INIT-1          INITIATE I/O               CRD00099
0059 0  7001               MDX      CA28                                       CRD00100
005A 0  082F         CA27  XIO      FEED-1                                     CRD00101
005B 0  7101         CA28  MDX   1 +1                                          CRD00102
005C 0  C029               LD       TEMP                                       CRD00103
005D 0  6906         CA29  STX   1 CA34+1           SET EXIT TO SKIP 1ST PARAM CRD00104
005E 00 65000000     CA30  LDX   L1 0               RESTORE STATUS             CRD00105
0060 00 66000000     CA31  LDX   L2 0                                          CRD00106
0062 0  2000         CA32  LDS      0                                          CRD00107
0063 00 4C000000     CA34  BSC   L  0               EXIT                       CRD00108
0065 0  9038         CA36  S        SETUP+5                                    CRD00109
0066 0  D01E               STO      COLM+1            SETUP PUNCH I/O          CRD00110
0067 00 C5800001           LD    I1 1                                          CRD00111
0069 01 4C080070           BSC   L  CA40.+            = ERROR IF ZERO OR NEG   CRD00112
006B 0  D028               STO      COUNT                                      CRD00113
006C 0  D02A               STO      RSTRT             SAVE WORD COUNT          CRD00114
006D 0  9021               S        D0080             DO NOT PUNCH OVER 80 COL CRD00115
006E 0  4808               BSC      +                                          CRD00116
006F 0  70D0               MDX      CA24                                       CRD00117
0070 0  C021         CA40  LD       H1001           ERROR CODE - ILLEGAL CALL  CRD00118
0071 0  700B               MDX      CA44                                       CRD00119
0072 0  1801         CA42  SRA      1               IS DEVICE BUSY             CRD00120
0073 01 4C04001F           BSC   L  CA17.E            YES. WAIT TIL NOT        CRD00121
0075 0  1003               SLA      3               IS DSW ERROR INDIC ON      CRD00122
0076 01 4C10007C           BSC   L  CA43.-            NO                       CRD00123
0078 0  C0AC               LD       CA20              YES. IS FUNCT GET/FEED   CRD00124
0079 01 4C04007C           BSC   L  CA43.E              NO                     CRD00125
007B 0  D019               STO      ERROR               YES. INDIC SKIP 1ST CD CRD00126
007C 0  C014         CA43  LD       H1000           ERROR CODE - DVCE NOT RDY  CRD00127
007D 0  71FF         CA44  MDX   1 -1                                          CRD00128
007E 00 6D000028           STX   L1 40                STORE CALL ADDR IN 40    CRD00129
0080 0  6129               LDX   1 41                 SET EXIT FOR 41          CRD00130
0081 0  70DB               MDX      CA29                                       CRD00131
                     ***************************************************** CRD00132
                     *                  CONSTANTS                        * CRD00133
                     ***************************************************** CRD00134
0082    0000               BSS   E  0                                          CRD00135
0082 1  0082         ADDR  DC       CHAR-1          ADDR TO REPLACE O/P AREA   CRD00136
0083 0  0000         CHAR  DC       0               TEMPORARY REGISTER       O CRD00137
0084 0  0000         COLM  DC       0               IOCC FOR COLUMN I/O      E CRD00138
0085 0  0000               DC       0                                        O CRD00139
0086 0  0000         TEMP  DC       0               TEMPORARY STORAGE          CRD00140
0087 0  0400         INIT  DC       /0400           IOCC TO INITIATE I/O     O CRD00141
0088 0  2075         CONST DC       SETUP-CA18-1+/2000                         CRD00142
0089 0  1700         SENSE DC       /1700           SENSE DSW WITHOUT RESET  O CRD00143
008A 0  0000         BUSY  DC       0               ROUTINE BUSY INDICATOR     CRD00144
008B 0  1402         FEED  DC       /1402           IOCC TO FEED 1 CARD      O CRD00145
008C 0  0001         D0001 DC       +1                                         CRD00146
008D 0  0004         D0004 DC       +4                                         CRD00147
008E 0  0008         D0008 DC       +8                                         CRD00148
008F 0  0050         D0080 DC       +80                                        CRD00149
0090 0  0051         D0081 DC       +81                                        CRD00150
0091 0  1000         H1000 DC       /1000                                      CRD00151
0092 0  1001         H1001 DC       /1001                                      CRD00152
0093 0  7003         H7003 DC       /7003           INSTRUCTIONS = MDX X +3    CRD00153
0094 0  0000         COUNT DC       0               NO. WORDS TO XFER          CRD00154
0095 0  0000         ERROR DC       0               SKIP ONE CARD INDIC        CRD00155
0096 0  0000         INDIC DC       0               FEED CHK (RD STATION) IND  CRD00156
0097 0  0000         RSTRT DC       0               RESTART INFO - WORD COUNT  CRD00157
0098 0  0000               DC       0                              DATA ADDR   CRD00158
0099 0  02FC         SETUP DC       /02FC           INITIATE IOCC SETUP - GET  CRD00159
009A 0  02FF               DC       /02FF                               - PUT  CRD00160
009B 0  02FE               DC       /02FE                               - FEED CRD00161
009C 0  0280               DC       /0280                               - STK  CRD00162
009D 0  0204               DC       /0204           COLUMN IOCC SETUP   - GET  CRD00163
009E 0  0301               DC       /0301                               - PUT  CRD00164
```

```
                           ************************************************ CRD00165
                           *            OP COMPLETE INTERRUPT PROCESSING   * CRD00166
                           ************************************************ CRD00167
                           *     THIS PORTION IS ENTERED FROM INTERR LEVEL * CRD00168
                           *     SROUTINE 04. IF NO ERROR HAS BEEN DETECTED. * CRD00169
                           *     THE ROUTINE IS SET NOT BUSY AND THE IOCS  * CRD00170
                           *     COUNTER IS DECREMENTED TO INDICATE        * CRD00171
                           *     INTERRUPT PROCESSING COMPLETED. OTHERWISE * CRD00172
                           *     THE SUBROUTINE LOOPS UNTIL THE OPERATOR HAS * CRD00173
                           *     INTERVENED AND THE 1442 BECOMES READY. AT * CRD00174
                           *     WHICH TIME THE CARDS ARE POSITIONED AND THE * CRD00175
                           *     I/O OPERATION IS RE-INITIATED.            * CRD00176
                           ************************************************ CRD00177
009F 0  80EC               NT10  A        D0001      OPER COMPLETE INTERRUPT    CRD00178
00A0 0  D0E2                     STO      CHAR                                  CRD00179
00A1 0  08E0                     XIO      CHAR-1       SENSE DSW WITH RESET     CRD00180
00A2 0  1003                     SLA      3            IS OPERATION OK          CRD00181
00A3 01 4C0200BA                 BSC  L   NT11.C         NO. ERROR              CRD00182
00A5 01 4C2800B4           NT10B BSC  L   NT10X.Z+       NO. LAST CARD          CRD00183
00A7 0  C0ED                     LD       ERROR                                 CRD00184
00A8 0  1002                     SLA      2              YES. WAS THIS SKIP OP  CRD00185
00A9 0  1810                     SRA      16                                    CRD00186
00AA 0  D0EA                     STO      ERROR                                 CRD00187
00AB 01 4C0200C5                 BSC  L   NT12.C           YES. INITIATE FUNCT  CRD00188
00AD 00 74FF0032           NT10E MDX  L   50.-1            NO. TERMINATE FUNCT  CRD00189
00AF 0  1000                     NOP                       DECREMENT IOCS COUNT CRD00190
00B0 0  1810                     SRA      16                                    CRD00191
00B1 0  D0D8                     STO      BUSY             CLEAR ROUTINE BUSY   CRD00192
00B2 01 4C800007                 BSC  I   INT2             EXIT                 CRD00193
00B4 0  C0CD               NT10X LD       ADDR                                  CRD00194
00B5 0  80D6                     A        D0001                                 CRD00195
00B6 0  F0CD                     EOR      COLM             IS FUNCTION PUT      CRD00196
00B7 0  4818                     BSC      +-                 NO                 CRD00197
00B8 0  08D1                     XIO      FEED-1             YES. EJECT LAST CD CRD00198
00B9 0  70F3                     MDX      NT10E                                 CRD00199
00BA 0  1005               NT11  SLA      5            SAVE FD CHK (RD STA) IND CRD00200
00BB 0  C0CB                     LD       INIT         IS FUNCT PUNCH           CRD00201
00BC 01 4C0400C5                 BSC  L   NT12.E         YES. DONT SKIP         CRD00202
00BE 0  1801                     SRA      1            IS FUNCT FEED            CRD00203
00BF 01 4C0400D5                 BSC  L   NT13B.E        YES                    CRD00204
00C1 0  C0D6                     LD       RSTRT+1      WAS ONE COL READ IN      CRD00205
00C2 0  F0C1                     EOR      COLM                                  CRD00206
00C3 01 4C1800D7                 BSC  L   NT13E.+-       NO. SKIP 1ST CARD      CRD00207
00C5 0  08BC               NT12  XIO      CHAR-1                                CRD00208
00C6 01 4C0400C5                 BSC  L   NT12.E       WAIT TIL READER READY    CRD00209
00C8 0  C0CC                     LD       ERROR        IS CARD SKIP NECESSARY   CRD00210
00C9 01 4C1800CE                 BSC  L   NT13.+-        NO                     CRD00211
00CB 0  08BE                     XIO      FEED-1       SKIP 1ST CARD            CRD00212
00CC 01 4C800007                 BSC  I   INT2                                  CRD00213
00CE 0  C0C8               NT13  LD       RSTRT                                 CRD00214
00CF 0  D0C4                     STO      COUNT            SETUP FOR RETRY      CRD00215
00D0 0  C0C7                     LD       RSTRT+1                               CRD00216
00D1 0  D0B2                     STO      COLM                                  CRD00217
00D2 0  08B3                     XIO      INIT-1       INITIATE I/O OPERATION   CRD00218
00D3 01 4C800007                 BSC  I   INT2         EXIT                     CRD00219
00D5 01 4C0200C5           NT13B BSC  L   NT12.C         NO SKIP IF FD CHK (RD) CRD00220
00D7 0  C0BB               NT13E LD       H7003                                 CRD00221
00D8 0  D0BC                     STO      ERROR          SET BIT 1 OF INDIC     CRD00222
00D9 0  70EB                     MDX      NT12                                  CRD00223
                           ************************************************ CRD00224
                           *            COLUMN INTERRUPT PROCESSING        * CRD00225
                           ************************************************ CRD00226
                           *     THIS PORTION IS ENTERED FROM INTERR LEVEL * CRD00227
                           *     SUBROUTINE 00. AFTER THE REQUESTED NO. OF * CRD00228
                           *     COLUMNS HAS BEEN READ. THE REMAINING COLUMN * CRD00229
                           *     INTERRUPTS ARE MERELY TURNED OFF AS THEY  * CRD00230
                           *     OCCUR. WHEN THE LAST COLUMN REQUESTED IS  * CRD00231
                           *     PUNCHED. AN INDICATION IS GIVEN TO THE 1442 * CRD00232
                           *     TO INITIATE AN OP COMPLETE INTERRUPT.     * CRD00233
                           ************************************************ CRD00234
00DA 0  D0A3               NT14  STO      CHAR       COLUMN REQUEST INTERRUPT   CRD00235
00DB 0  08A6                     XIO      CHAR-1       SENSE DSW WITH RESET     CRD00236
00DC 01 74FF0094                 MDX  L   COUNT.-1     ANY MORE COLS TO PROCESS CRD00237
00DE 0  700D                     MDX      NT18           YES                    CRD00238
00DF 01 4C1000E4                 BSC  L   NT16.-       IS THIS READ COL INTERR  CRD00239
00E1 01 74010094                 MDX  L   COUNT.+1       YES. SET TO SKIP       CRD00240
00E3 0  700B                     MDX      NT22             NEXT COL             CRD00241
00E4 01 74010084           NT16  MDX  L   COLM.+1        NO. STORE STOP PUNCH   CRD00242
00E6 01 C4800084                 LD   I   COLM             BIT (BIT 12) IN COL  CRD00243
00E8 0  E8A5                     OR       D0008            DATA                 CRD00244
00E9 0  D099                     STO      CHAR                                  CRD00245
00EA 0  C097                     LD       ADDR             PUNCH FROM TEMPORARY CRD00246
00EB 0  D098                     STO      COLM               LOCATION           CRD00247
00EC 01 74010084           NT18  MDX  L   COLM.+1      SET ADDR FOR NEXT COLUMN CRD00248
00EE 0  0895               NT20  XIO      COLM         EXECUTE COLUMN I/O       CRD00249
00EF 01 4C800004           NT22  BSC  I   INT1       EXIT                       CRD00250
00F2                             END                                            CRD00251
```

APPENDIX A. LISTING OF SUBROUTINES

Table 8 is a listing of the Disk Monitor 1 and Card/Paper Tape System Subroutine Library. The Disk Monitor 2 System Library is listed in Table 9.

● Table 8. DM1 and C/PT System Subroutine Library

Subroutine	Names	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 (DM1)		
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	ILS02
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

Table 8. DM1 and C/PT System Subroutine Library (Cont.)

Subroutine	Names	Subroutines Required
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
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 Hexadecimal	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

● **Table 8. DM1 and C/PT System Subroutine Library (Cont.)**

Subroutine	Names	Subroutines Required
Called by LIBF (Cont'd)		
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 (Card/Paper Tape)		
Called by CALL		
Set Pack Initialization Subroutine	SPIR0, SPIR1, SPIRN	DISK0, DISK1, DISKN
OVERLAY (DM1)		
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 DM1)	DISK0, DISK1, DISKN	ILS02
Paper Tape	PAPT1, PAPTN	ILS04
Plotter	PLOT1	ILS04
1132 Printer	PRNT1	ILS01
Keyboard/Console Printer	TYPE0, WRTY0	HOLL, PRTY, ILS04
PLOTTER SUBROUTINES		
Standard Plot Calls		
Standard Precision Character	FCHAR	FSIN, FCOS, FPLOT, FCHRX, FLD, FSTOX, FSTO
Standard Precision Scale	SCALF	FRULE
Standard Precision Grid	FGRID	FPLOT, POINT, FADD, FLD, FSTO, SNR
Standard Precision Plot	FPLOT	FMOVE, YPLT, PLOT1
Extended Plot Calls		
Extended Precision Character	ECHAR	ESIN, ECOS, EPLOT, ECHRX, ELD, ESTO, ESTOX
Extended Precision Scale	SCALE	ERULE
Extended Precision Grid	EGRID	EPLOT, POINT, EADD, ELD, ESTO, SNR
Extended Precision Plot	EPLOT	EMOVE, XYPLT, PLOT1
Common Plot Call		
Point Characters	POINT	PLOT1
Standard Plot LIBFs		
Standard Precision Annotation	FCHRX, FCHR1, WCHR1	FLOAT, FMPY, IFIX, FADD, FLDX, FINC, XYPLT, PLOT1, FSTOX, FLD
Standard Precision Plot Scaler	FRULE, FMOVE, FINC	FLDX, FSUBX, FMPYX, FLD, FSTOX, FMPY, IFIX, FADD
Extended Plot LIBFs		
Extended Precision Annotation	ECHRX, ECHR1, YCHR1	FLOAT, EMPY, IFIX, EADD, ELDX, EINC, XYPLT, PLOT1, ESTOX, ELD
Extended Precision Plot Scaler	ERULE, EMOVE, EINC	ELDX, ESUBX, EMPYX, ELD, ESTOX, EMPY, IFIX, EADD, ESTO
Common Plot LIBFs		
Pen Mover	XYPLT	PLOT1
Interface	PLOT1	PLOTX
Interrupt Service	PLOTX	
Synchronous Communications Adaptor Subroutines		
Synchronous Communications Adaptor (SCA) STR Mode	SCAT1	ILS01
SCA(BSC, Point-to-Point Mode)	SCAT2	ILS01
SCA(BSC, Multi-Point Mode	SCAT3	ILS01
1132-SCA Print with Overlap	PRNT2	ILS01
4 of 8 Code to EBCDIC, EBCDIC to 4 of 8 Code	EBC48	HXCV, STRTB
4 of 8 Code to IBM Card, IBM Card Code to 4 of 8 Code	HOL48	HXCV, HOLCA, STRTB
4 of 8 Code to Table of Displacements	HXCV	None
Table of IBM Card Codes	HOLCA	None
Table of 4 of 8 and EBCDIC Codes	STRTB	None

Table 9. 1130 Disk Monitor Version 2 System Library

System Library Programs	Names	Type and Sub-type	Subroutines Required
MAINLINES			
Disk Maintenance Programs			
Disk Initialization	DISC	2,0	SYSUP, RDREC, DISKZ
Print Cartridge ID	IDENT	2,0	CALPR, DISKZ
Change Cartridge ID	ID	2,0	RDREC, CALPR, DISKZ
Disk Copy	COPY	2,0	RDREC, DISKZ
Writer Sector Addresses in WS	ADRWS (cannot be called)	2	Linked From DUP DWADR
Delete CIB	DLCIB	2,0	RDREC, DISKZ
Dump System Location Equivalence Table	DSLET	2,0	FSLEN, DISKZ
System Maintenance	MODIF	-	DISKZ
Paper Tape Utility			
Keyboard or 1134 Input/Console Printer or 1055 Output	PTUTL	--	
SUBROUTINES			
Utility Calls			
Selective Dump on Console Printer	DMTD0, DMTX0	4,0	WRTY0
Selective Dump on 1132 Printer	DMPD1, DMPX1	4,0	PRNT1
Dump 80 Subroutine	DMP80	4,0	None
Update DCOM	SYSUP	4,0	FSLEN, FSYSU
Call System Print	CALPR	4,0	FSLEN
Read *ID Record	RDREC	4,0	FSLEN
Fetch Phase IDs or, Fetch System Subroutine	FSLEN, FSYSU	4,0	DISKZ
Common FORTRAN Calls			
Test Data Entry Switches	DATSW	4,8	None
Divide Check Test	DVCHK	4,8	None
Functional Error Test	FCTST	4,8	None
Overflow Test	OVERF	4,8	None
Sense Light Control and Test	SLITE, SLITT	4,8	None
FORTRAN Trace Stop	TSTOP	4,8	TSET
FORTRAN Trace Start	TSTRT	4,8	TSET
Integer Transfer of Sign	ISIGN	4,8	None
Extended Arith/Funct Calls			
Extended Precision Hyperbolic Tangent	ETANH, ETNH	4,8	EEXP, ELD/ESTO, EADD, EDIV, EGETP
Extended Precision A**B Function	EAXB, EAXBX	4,8	EEXP, ELN, EMPY
Extended Precision Natural Logarithm	ELN, EALOG	4,8	XMD, EADD, EMPY, EDIV, NORM, EGETP
Extended Precision Exponential	EEXP, EXPN	4,8	XMD, FARC, EGETP
Extended Precision Square Root	ESQR, ESQRT	4,8	ELD/ESTO, EADD, EMPY, EDIV, EGETP
Extended Precision Sine-Cosine	ESIN, ESINE, ECOS, ECOSN	4,8	EADD, EMPY, NORM, XMD, EGETP
Extended Precision Arctangent	EATN, EATAN	4,8	EADD, EMPY, EDIV, XMD, EGETP, NORM
Extended Precision Absolute Value Function	EABS, EAVL	4,8	EGETP
FORTRAN Sign Transfer Calls			
Extended Precision Transfer of Sign	ESIGN	4,8	ESUB, ELD
Standard Precision Transfer of Sign	FSIGN	4,8	FSUB, FLD
Standard Arith/Funct Calls			
Standard Precision Hyperbolic Tangent	FTANH, FTNH	4,8	FEXP, FLD/FSTO, FADD, FDIV, FGETP
Standard Precision A**B Function	FAXB, FAXBX	4,8	FEXP, FLN, FMPY
Standard Precision Natural Logarithm	FLN, FALOG	4,8	FSTO, XMDS, FADD, FMPY, FDIV, NORM, FGETP
Standard Precision Exponential	FEXP, FXPN	4,8	XMDS, FARC, FGETP
Standard Precision Square Root	FSQR, FSQRT	4,8	FLD/FSTO, FADD, FMPY, FDIV, FGETP
Standard Precision Sine-Cosine	FSIN, FSINE, FCOS, FCOSN	4,8	FADD, FMPY, NORM, XMDS, FSTO, FGETP
Standard Precision Arctangent	FATN, FATAN	4,8	FADD, FMPY, FDIV, XMDS, FSTO, FGETP
Standard Precision Absolute Value Function	FABS, FAVL	4,8	FGETP
Common Arith/Funct Calls			
Fixed Point (Fractional) Square Root	XSQR	4,8	None
Integer Absolute Function	IABS	4,8	None
Floating Binary/EBC Decimal Conversions	FBTD (BIN. TO DEC.) FDTB (DEC. TO BIN.)	4,0	None
Flipper for LOCAL SOCAL Subprograms	FLIPR	4,0	DISKZ, DISK1, or DISKN
FORTRAN Trace Subroutines			
Extended Floating Variable Trace	SEAR, SEARX	3,0	ESTO, TTEST, SWRT, SIOF, SCOMP
Fixed Variable Trace	SIAR, SIARX	3,0	TTEST, SWRT, SIOI, SCOMP
Standard Floating IF Trace	SFIF	3,0	FSTO, TTEST, SWRT, SIOF, SCOMP
Extended Floating IF Trace	SEIF	3,0	FSTO, TTEST, SWRT, SIOF, SCOMP
Fixed IF Trace	SIIF	3,0	TTEST, SWRT, SIOI, SCOMP
Standard Floating Variable Trace	SFAR, SFARX	3,0	FSTO, TTEST, SWRT, SIOF, SCOMP
GO TO Trace	SGOTO	3,0	TTEST, SWRT, SIOI, SCOMP

● Table 9. 1130 Disk Monitor Version 2 System Library (Cont.)

System Library Programs	Names	Type and Sub-type	Subroutines Required
Non-Disk FORTRAN Format I/O			
FORTRAN Format Subroutine	SFIO, SIOI, SIOAI, SIOF, SIOAF, SIOFX, SCOMP, SWRT, SRED, SIOIX	3,3	FLOAT, ELD/ESTO or FLD/FSTO, IFIX
FORTRAN Find Subroutine	SDFND	3,1	DISKZ, DISK1, or DISKN
Disk FORTRAN I/O	SDFIO, SDRED, SDWRT, SDCOM, SDAF, SDF, SDI, SDIX, SDFX, SDAI	3,1	DISKZ, DISK1, or DISKN
Unformatted FORTRAN Disk I/O	SUFIO	3,1	DISKZ, DISK1, or DISKN
FORTRAN Common LIBFs			
FORTRAN Pause	PAUSE	3,2	None
FORTRAN Stop	STOP	3,2	None
FORTRAN Subscript Displacement Calculation	SUBSC	3,0	None
FORTRAN Subroutine Initialization	SUBIN	3,0	None
FORTRAN Trace Test and Set	TTEST, TSET	3,0	None
FORTRAN I/O and Conversion Subroutines			
FORTRAN 1442 Input/Output Subroutine	CARDZ	5,3	HOLEZ, GETAD, EBCTB, HOLTB, ILS00, ILS04
FORTRAN 1442 Output Subroutine	PNCHZ	5,3	HOLEZ, GETAD, EBCTB, HOLTB, ILS00, ILS04
FORTRAN 2501 Input Subroutine	READZ	5,3	HOLEZ, GETAD, EBCTB, HOLTB, ILS04
Disk I/O Routine (Part of Supervisor)	DISKZ	-	ILS02
FORTRAN Paper Tape Subroutine	PAPTZ	5,3	ILS04
FORTRAN 1132 Printer Subroutine	PRNTZ	5,3	ILS01
FORTRAN 1403 Printer Subroutine	PRNZ	5,3	ILS04
FORTRAN Keyboard-Typewriter Subroutine	TYPEZ	5,3	GETAD, EBCTB, HOLEZ, ILS04
FORTRAN Typewriter Subroutine	WRTYZ	5,3	GETAD, EBCTB, ILS04
FORTRAN 1627 Plotter Subroutine	PLOTX	5,0	ILS03
FORTRAN Hollerith to EBCDIC Conversion	HOLEZ	3,3	GETAD, EBCTB, HOLTB
FORTRAN Get Address Routine	GETAD	3,3	None
FORTRAN EBCDIC Table	EBCTB	3,3	None
FORTRAN Hollerith Table	HOLTB	3,3	None
Extended Arith/Funct LIBFs			
Extended Precision Get Parameter Subroutine	EGETP	3,2	ELD
Extended Precision A**I Function	EAXI, EAXIX	3,2	ELD/ESTO, EMPY, EDVR
Extended Precision Divide Reverse	EDVR, EDVRX	3,2	ELD/ESTO, EDIV
Extended Precision Float Divide	EDIV, EDIVX	3,2	XDD, FARC
Extended Precision Float Multiply	EMPY, EMPYX	3,2	XMD, FARC
Extended Precision Subtract Reverse	ESBR, EXBRX	3,2	EADD
Extended Add-Subtract	EADD, ESUB, EADDX, ESUBX	3,2	FARC, NORM
Extended Load-Store	ELD, ELDX, ESTO, ESTOX	3,0	None
Standard Arith/Funct LIBFs			
Standard Precision Get Parameter Subroutine	FGETP	3,2	FLD
Standard Precision A**I Function	FAXI, FAXIX	3,2	FLD/FSTO, FMPY, FDVR
Standard Precision Divide Reverse	FDVR, FDVRX	3,2	FLD/FSTO, FDIV
Standard Precision Float Divide	FDIV, FDIVX	3,2	FARC
Standard Precision Float Multiply	FMPY, FMPYX	3,2	XMDS, FARC
Standard Precision Subtract Reverse	FSBR, FSBRX	3,2	FADD
Standard Add-Subtract	FADD, FSUB, FADDX, FSUBX	3,2	NORM, FARC
Standard Load-Store	FLD, FLDX, FSTO, FSTOX	3,0	None
Standard Precision Fractional Multiply	XMDS	3,2	None
Common Arith/Funct LIBFs			
Fixed Point (Fractional) Double Divide	XDD	3,2	XMD
Fixed Point (Fractional) Double Multiply	XMD	3,2	None
Sign Reversal Function	SNR	3,2	None
Integer to Floating Point Function	FLOAT	3,0	NORM
Floating Point to Integer Function	IFIX	3,0	None
I**J Integer Function	FIXI, FIXIX	3,2	None
Normalize Subroutine	NORM	3,0	None
Floating Accumulator Range Check Subroutine	FARC	3,2	None
Interrupt Service Subroutines			
1442 Card Read Punch Input/Output (No Error Parameter)	CARD0	5,0	ILS00, ILS04
1442 Card Read Punch Input/Output (Error Parameter)	CARD1	5,0	ILS00, ILS04
2501 Card Read Input (No Error Parameter)	READ0	5,0	ILS04
2501 Card Read Input (Error Parameter)	READ1	5,0	ILS04
1442 Card Punch Output (No Error Parameter)	PNCH0	5,0	ILS00, ILS04
1442 Card Punch Output (Error Parameter)	PNCH1	5,0	ILS00, ILS04
Multiple Sector Disk Input/Output (Part of Supervisor)	DISK1	-	ILS02
High Speed Multiple Sector Disk Input/Output (Part of Supervisor)	DISKN	-	ILS02
Synchronous Communications Adaptor (SCA) STR Mode	SCAT1	5,0	ILS01

Table 9. 1130 Disk Monitor Version 2 System Library (Cont.)

System Library Programs	Names	Type and Sub-type	Subroutines Required
Interrupt Service Subroutines (Cont'd)			
SCA(BSC, Point-to-Point Mode)	SCAT2	5,0	ILS01
SCA(BSC, Multi-Point Mode)	SCAT3	5,0	ILS01
Paper Tape Input/Output	PAPT1	5,0	ILS04
Simultaneous Paper Tape Input/Output	PAPTN	5,0	ILS04
Character/Word Count Paper Tape Input/ Output	PAPTX	5,0	ILS04
Plotter Output Subroutine	PLOT1	5,0	ILS03
1132 Printer Output Subroutine	PRNT1	5,0	ILS01
1132-SCA Print With Overlap	PRNT2	5,0	ILS01
1403 Printer Output Subroutine	PRNT3	5,0	ILS04
Keyboard/Console Printer Input/Output	TYPE0	5,0	HOL, PRTY, ILS04
Console Printer Output Subroutine	WRTY0	5,0	ILS04
1231 Optical Mark Page Reader Input Subroutine	OMPR1	5,0	ILS04
Conversion Subroutines			
Binary Word to 6 Decimal Characters (Card Code)	BINDC	3,0	None
Binary Word to 4 Hexadecimal Characters (Card Code)	BINHX	3,0	None
6 Decimal Characters (Card Code) to Binary Word	DCBIN	3,0	None
EBCDIC to Console Printer Output Code	EBPRT	3,0	EBPA, PRTY
Card Code to EBCDIC-EBCDIC to Card Code	HOLEB	3,0	EBPA, HOLL
Card Code to Console Printer Output Code	HOLPR	3,0	HOLL, PRTY
4 Hexadecimal Characters (Card Code) to Binary Word	HXBIN	3,0	None
PTTC/8 to EBCDIC-EBCDIC to PTTC/8	PAPEB	3,0	EBPA
PTTC/8 to Card Code-Card Code to PTTC/8	PAPHL	3,0	EBPA, HOLL
PTTC/8 to Console Printer Output Code	PAPPR	3,0	EBPA, PRTY
Card Code to EBCDIC-EBCDIC to Card Code	SPEED	3,0	None
4 of 8 Code to EBCDIC, EBCDIC to 4 of 8 Code	EBC48	3,0	HXCV, STRTB
4 of 8 Code to IBM Card Code, IBM Card Code to 4 of 8 Code	HOL48	3,0	HXCV, HOLCA, STRTB
4 of 8 Code to Table of Displacements	HXCV	3,0	None
32-Bit Binary Value to IBM Card Code Decimal Value	BIDEC	3,0	None
IBM Card Code Decimal Value to 32-Bit Binary Value	DECBI	3,0	None
Supplement to All Standard Conversions Except Those Involving PTTC/8	ZIPCO	3,0	Any ZIPCO Conversion Table
Conversion Tables			
EBCDIC and PTTC/8	EBPA	3,0	None
Card Code Table	HOLL	3,0	None
Console Printer Output Code Table	PRTY	3,0	None
Table of IBM Card Codes	HOLCA	3,0	None
Table of 4 of 8 and EBCDIC Codes	STRTB	3,0	None
ZIPCO Conversion Tables			
EBCDIC to Console Printer Code	EBCCP	3,0	None
EBCDIC to IBM Card Code	EBHOL	3,0	None
EBCDIC to 1403 Printer Code	EBPT3	3,0	None
Console Printer Code to EBCDIC	CPEBC	3,0	None
Console Printer Code to IBM Card Code	CPHOL	3,0	None
Console Printer Code to 1403 Printer Code	CPPT3	3,0	None
IBM Card Code to EBCDIC	HLEBC	3,0	None
IBM Card Code to Console Printer Code	HOLCP	3,0	None
IBM Card Code to 1403 Printer Code	HLPT3	3,0	None
1403 Printer Code to EBCDIC	PT3EB	3,0	None
1403 Printer Code to Console Printer Code	PT3CP	3,0	None
1403 Printer Code to IBM Card Code	PTHOL	3,0	None
Interrupt Level Subroutines			
Interrupt Level Zero Subroutine	ILS00	7,0	None
Interrupt Level One Subroutine	ILS01	7,0	None
Interrupt Level Two Subroutine (Part of Supervisor)	ILS02	7,1	None
Interrupt Level Three Subroutine	ILS03	7,0	None
Interrupt Level Four Subroutine (Part of Supervisor)	ILS04	7,1	None
Standard Plot Calls			
Standard Precision Character	FCHAR	4,0	FSIN, FCOS, FPLOT, FCHRX, FLD, FSTOX, FSTO
Standard Precision Scale	SCALF	4,0	FRULE
Standard Precision Grid	FGRID	4,0	FPLOT, POINT, FADD, FLD, FSTO, SNR
Standard Precision Plot	FPLOT	4,0	FMOVE, XYPLT, PLOT1

Table 9. 1130 Disk Monitor Version 2 System Library (Cont.)

System Library Programs	Names	Type and Sub-type	Subroutines Required
Extended Plot Calls			
Extended Precision Character	ECHAR	4,0	ESIN, ECOS, EPLOT, ECHRX, ELD, ESTO, ESTOX
Extended Precision Scale	SCALE	4,0	ERULE
Extended Precision Grid	EGRID	4,0	EPLOT, POINT, EADD, ELD, ESTO, SNR
Extended Precision Plot	EPLOT	4,0	EMOVE, XYPLT, PLOTI
Common Plot Call			
Point Characters	POINT	4,0	PLOTI
Standard Plot LIBFs			
Standard Precision Annotation	FCHRX, FCHRI, WCHRI	3,2	FLOAT, FMPY, IFIX, FADD, FLDX, FINC, XYPLT, PLOTI, FSTOX, FLD
Standard Precision Plot Scaler	FRULE, FMOVE, FINC	3,2	FLDX, FSUBX, FMPYX, FLD, FSTOX, FMPY, IFIX, FADD
Extended Plot LIBFs			
Extended Precision Annotation	ECHRX, ECHRI, YCHRI	3,2	FLOAT, EMPY, IFIX, EADD, ELDX, EINC, XYPLT, PLOTI, ESTOX, ELD
Extended Precision Plot Scaler	ERULE, EMOVE, EINC	3,2	ELDX, ESUBX, EMPYX, ELD, ESTOX, EMPY, IFIX, EADD, ESTO
Common Plot LIBFs			
Pen Mover	XYPLT	3,2	PLOTI
Interface	PLOTI	3,2	PLOTX
Interrupt Service	PLOTX	5,0	ILS03

APPENDIX B. ERRORS DETECTED BY THE ISS SUBROUTINES

ERROR	CONTENTS OF ACCUMULATOR		Contents of Extension (if any)
	Binary	Hexadecimal	
1442 Card Read Punch or 1442 Card Punch			
*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	
Keyboard/Console Printer			
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	
1134/1055 Paper Tape Reader/Punch			
*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	
2501 Card Reader			
*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	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 1	
Device not ready	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0	4 0 0 0	
Illegal function Word count over +80 Word count zero or negative	0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1	4 0 0 1	
Disk			
*Read check remaining after ten attempts (16 for DM2 system) 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 (16 for DM2 system) Write select Data error Data Overrun	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	0 0 0 2	Effective Sector Id
*Seek failure remaining after ten attempts (16 for DM2 system)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	0 0 0 3	Effective Sector Id
*Disk overflow	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0	0 0 0 4	
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 (DISK0 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	
Write select/power unsafe	0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0	5 0 0 2	Effective Sector Id
Identical to 5001 except error caused by a Monitor program (DISK1, DISKN only)	0 1 0 1 0 0 0 0 0 0 0 0 0 0 1 1	5 0 0 3	
Disk error (DISKZ only)	0 1 0 1 0 0 0 0 0 0 0 0 0 1 0 0	5 0 0 4	Effective Sector Id

NOTE: The errors marked with an asterisk cause a branch via the error parameter. These are postoperative errors and are detected during the processing of interrupts; as a consequence, the user error subroutine is an interrupt subroutine, executed at the priority level of the I/O device. All other errors are preoperative and cause a branch to location/0029 on the DM1 and C/PT system or $PRET+1 on the DM2 system. The address of the LIBF in error is in location/0028 or $PRET.

ERROR	CONTENTS OF ACCUMULATOR		Contents of Extension (if any)
	Binary	Hexadecimal	
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 Word count over +60 Word count zero or negative	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	
1403 Printer			
*Ring check *Sync check *Parity check	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 1	
*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	1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0	9 0 0 0	
Illegal function Word count over +60 Word count zero or negative	1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1	9 0 0 1	
Optical Mark Page Reader			
Master mark	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1	0 0 0 1	
Timing mark error Read error	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0	0 0 0 2	
Hopper empty	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1	0 0 0 3	
Document selected	0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0	0 0 0 4	
Device not ready	1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0	A 0 0 0	
Illegal function	1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 1	A 0 0 1	

NOTE: The errors marked with an asterisk cause a branch via the error parameter. These are postoperative errors and are detected during the processing of interrupts; as a consequence, the user error subroutine is an interrupt subroutine, executed at the priority level of the I/O device. All other errors are preoperative and cause a branch to location/0029 on the DM1 or C/PT system or $PRET+1 on the DM2 system. The address of the LIBF in error is in location/0028 or $PRET.

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

Error Code	Condition	Subroutine Action
1442 Card Read Punch or 1442 Card Punch		
0000	If function is PUNCH	Eject card and terminate
	Otherwise	Terminate immediately
0001*	If Accumulator is 0	Terminate immediately
	Otherwise	DM1 and C/PT System: Loop until 1442 is ready, then reinitiate operation
		DM2 System: WAIT at $PST4 until 1442 is readied and PROGRAM START pressed
2501 Card Reader		
0000		Terminate
0001*	If Accumulator is 0	Terminate immediately
	Otherwise	WAIT at $PST4 until 2501 is readied and PROGRAM START pressed
1134/1055 Paper Tape Reader/Punch		
0004, 0005	If Accumulator is 0	Terminate immediately
	Otherwise	Check again for device ready
Disk		
0001, 0002, and 0003	If Accumulator is 0	Terminate immediately
	Otherwise	Retry 10 times (DM1 and C/PT system), or 16 times (DM2 system)
1132 Printer		
0003, and 0004	If Accumulator is 0	Terminate immediately
	Otherwise	Skip to channel 1 and then terminate
1403 Printer		
0001	If Accumulator is 0	Terminate immediately
	Otherwise	Check for device ready and reinitiate the operation
0003, and 0004	If Accumulator is 0	Terminate immediately
	Otherwise	Skip to channel 1 and then terminate
1231 OMPR		
0001	If Accumulator is 0	Continue normal processing
	Otherwise	Use contents of Accumulator as new address of I/O area
0002*	If Accumulator is 0	Terminate immediately
	Otherwise	Check device ready, then reinitiate operation
0003		Terminate
0004*	If Accumulator is 0	Terminate immediately
	Otherwise**	Check device ready, then reinitiate operation

* Assumes operator intervention

** User must provide a WAIT in his error subroutine to allow him to remove the sheet from the select stacker, place the sheet back in the hopper, and make the 1231 ready.

Ref No.	EBCDIC			IBM Card Code							Graphics and Control Names		1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex	Notes	1403 Printer Hex
	Binary		Hex	Rows						Hex							
	0123	4567		12	11	0	9	8	7-1								
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					
5*		0101	05	12			9		5	8110	HT	Horiz.Tab		6D (U/L)	41	①	
6*		0110	06	12			9		6	8090	LC	Lower Case		6E (U/L)			
7*		0111	07	12			9		7	8050	DEL	Delete		7F (U/L)			
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					
37*		0101	25			0	9		5	2110	LF	Line Feed		3D (U/L)	03		
38*		0110	26			0	9		6	2090	EOB	End of Block		3E (U/L)			
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					
53*		0101	35				9		5	0110	RS	Reader Stop		0D(U/L)	09	④	
54*		0110	36				9		6	0090	UC	Upper Case		0E (U/L)			
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	③	Carrier Return
②	Shift to black	④	Shift to red

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

Ref No.	EBCDIC			IBM Card Code							Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex	1403 Printer Hex
	Binary		Hex	Rows						Hex					
	0123	4567		12	11	0	9	8	7-1						
64*	0100	0000	40	no punches						0000	blank	40	10 (U/L)	21	7F
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	6E
76*		1100	4C	12				8	4	8220	<		02 (U)	DE	
77*		1101	4D	12				8	5	8120	(	4D	19 (U)	FE	57
78*		1110	4E	12				8	6	80A0	+	4E	70 (U)	DA	6D
79*		1111	4F	12				8	7	8060	| (logical OR)		3B (U)	C6	
80*	0101	0000	50	12						8000	&	50	70 (L)	44	15
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	62
92*		1100	5C		11			8	4	4220	*	5C	08 (U)	D6	23
93*		1101	5D		11			8	5	4120	)	5D	1A (U)	F6	2F
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	61
97*		0001	61			0			1	3000	/	61	31 (L)	BC	4C
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	16
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	0B
126*		1110	7E					8	6	00A0	=	7E	01 (U)	C2	4A
127*		1111	7F					8	7	0060	"		0B (U)	E2	

Ref No.	EBCDIC			IBM Card Code							Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex	1403 Printer Hex
	Binary		Hex	Rows						Hex					
	0123	4567		12	11	0	9	8	7-1						
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			IBM Card Code							Graphics and Control Names	1132 Printer EBCDIC Subset Hex	PTTC/8 Hex U-Upper Case L-Lower Case	Console Printer Hex	1403 Printer Hex
	Binary		Hex	Rows						Hex					
	0123	4567		12	11	0	9	8	7-1						
192	1100	0000	C0	12		0				A000	(+ zero)				
193*		0001	C1	12					1	9000	A	C1	61 (U)	3C or 3E	64
194*		0010	C2	12					2	8800	B	C2	62 (U)	18 or 1A	25
195*		0011	C3	12					3	8400	C	C3	73 (U)	1C or 1E	26
196*		0100	C4	12					4	8200	D	C4	64 (U)	30 or 32	67
197*		0101	C5	12					5	8100	E	C5	75 (U)	34 or 36	68
198*		0110	C6	12					6	8080	F	C6	76 (U)	10 or 12	29
199*		0111	C7	12					7	8040	G	C7	67 (U)	14 or 16	2A
200*		1000	C8	12				8		8020	H	C8	68 (U)	24 or 26	6B
201*		1001	C9	12			9			8010	I	C9	79 (U)	20 or 22	2C
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	58
210*		0010	D2		11				2	4800	K	D2	52 (U)	58 or 5A	19
211*		0011	D3		11				3	4400	L	D3	43 (U)	5C or 5E	1A
212*		0100	D4		11				4	4200	M	D4	54 (U)	70 or 72	5B
213*		0101	D5		11				5	4100	N	D5	45 (U)	74 or 76	1C
214*		0110	D6		11				6	4080	O	D6	46 (U)	50 or 52	5D
215*		0111	D7		11				7	4040	P	D7	57 (U)	54 or 56	5E
216*		1000	D8		11			8		4020	Q	D8	58 (U)	64 or 66	1F
217*		1001	D9		11		9			4010	R	D9	49 (U)	60 or 62	20
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	0D
227*		0011	E3			0			3	2400	T	E3	23 (U)	9C or 9E	0E
228*		0100	E4			0			4	2200	U	E4	34 (U)	B0 or B2	4F
229*		0101	E5			0			5	2100	V	E5	25 (U)	B4 or B6	10
230*		0110	E6			0			6	2080	W	E6	26 (U)	90 or 92	51
231*		0111	E7			0			7	2040	X	E7	37 (U)	94 or 96	52
232*		1000	E8			0		8		2020	Y	E8	38 (U)	A4 or A6	13
233*		1001	E9			0	9			2010	Z	E9	29 (U)	A0 or A2	54
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	49
241*		0001	F1						1	1000	1	F1	01 (L)	FC	40
242*		0010	F2						2	0800	2	F2	02 (L)	D8	01
243*		0011	F3						3	0400	3	F3	13 (L)	DC	02
244*		0100	F4						4	0200	4	F4	04 (L)	F0	43
245*		0101	F5						5	0100	5	F5	15 (L)	F4	04
246*		0110	F6						6	0080	6	F6	16 (L)	D0	45
247*		0111	F7						7	0040	7	F7	07 (L)	D4	46
248*		1000	F8					8		0020	8	F8	08 (L)	E4	07
249*		1001	F9				9			0010	9	F9	19 (L)	E0	08
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					

Communications Adaptor subroutine core requirements are listed in the publication IBM 1130 Synchronous Communications Adaptor Subroutines (Form C26-3706). 1627 Plotter subroutine core requirements are included in the publication IBM 1130/1800 Plotter Subroutines (C26-3755).

● Table 10. Arithmetic and Functional Subroutines

Standard		Extended	
FADD/FADDX	102	EADD/EADDX	98
FSUB/FSUBX		ESUB/ESUBX	
FMPY/FMPYX	52	EMPY/EMPYX	46
FDIV/FDIVX	86	EDIV/EDIVX	78
FLD/FLDX	54	ELD/ELDX	46
FSTO/FSTOX		ESTO/ESTOX	
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	118	ESIN/ESINE	138
FCOS/FCOSN		ECOS/ECOSN	
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.)	420		420
FDTB (dec. to bin.)			
DMTD0/DMTX0	412		412
DMPD1/DMPX1	520		520
DMP80	102		102
DATSW	34		34
DVCHK	16		16
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

Standard		Extended	
C/PT System		C/PT System	
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/	854	VFIO/VIOI/VIOAI/	864
WIOF/WIOAF/		VIOF/VIOAF/	
WIOFX/WCOMP/		VIOFX/VCOMP/	
WWRT/WRED/		VWRT/VRED/	
WIOIX		VIOIX	
DMI System		DMI System	
SDFIO/SDAF/SDAI/	604		604
SDCOM/SDF/SDFX/			
SDI/SDIX/SDRED/			
SDWRT			
SDFND	60		60
SFAR/SFARX	32	SEAR/SEARX	32
SFIO/SIOI/SIOAI/	980		980
SIOF/SIOAF/SIOFX/			
SCOMP/SWRT/SRED/			
SIOIX			
SFIF	26	SEIF	28
SGOTO	22		22
SIAR/SIARX	36		36
SIIF	24		24
DM2 System		DM2 System	
SDFIO/SDAF/SDAI/	622		
SDCOM/SDF/SDFX/			
SDI/SDIX/SDRED/			
SDWRT			
SDFND	76		
SFAR/SFARX	32	SEAR/SEARX	32
SFIO/SIOI/SIOAI/	1148		
SIOF/SIOAF/SIOFX/			
SCOMP/SWRT/SRED/			
SIOIX			
SFIF	26	SEIF	28
SGOTO	22		
SIAR/SIARX	36		
SIIF	24		
SUFIO	732		

● Table 11. Miscellaneous and ISS Subroutines

Subroutines	No. Core Locations DM2 System	No. Core Locations DM1 and C/PT System	Uses
CARD0	244	242	ILS00, ILS04
CARD1	250	246	ILS00, ILS04
READ0	96	-	ILS04
READ1	110	-	ILS04
PNCH0	206	-	ILS04
PNCH1	218	-	ILS04
OMPR1	262	-	
PAPT1	260	254	ILS04
PAPTN	304	294	ILS04
DISK0	-	356	ILS02
DISK1	418	620	ILS02
DISKN	688	808	ILS02
WRTY0	126	124	ILS04
TYPE0	282	296	ILS04, PRTY, HOLL
PLOT1	222	216	ILS03
PRNT1	386	386	ILS01
PRNT3	262	-	
ILS00	22	18	
ILS01	28	18 (28)	
ILS02*	17	18	
ILS03	22	18 (24)	
ILS04*	32	30	
SPIR0	-	48 (-)	
SPIR1	-	62 (-)	
SPIRN	-	62 (-)	
FLIP0	-	- (72)	DISK0, DISKZ
FLIP1	-	- (48)	DISK1, DISKN
FLIPR	102	-	
PAUSE	12	12	
STOP	12	8 (12)	
SUBSC	30	30	
SUBIN	32	32	
TTEST/TSET	16	16	
DISKZ*	238	- (208)	ILS02
CARDZ	168	80 (136)	
PAPTZ	226	202 (222)	
PRNTZ	190	176 (200)	
TYPEZ	94	82 (94)	
WRYTZ	60	60	
READZ	58		ILS04
PNCHZ	72		ILS04
PRNZ	192		ILS04
HOLEZ	54	54	
GETAD	16	14	
EBCTB	60	54	
HOLTB	54	54	

*Part of Resident Monitor

( ) DM1 System core requirements are different.

● Table 12. Conversion Subroutines

Conversion Subroutines	No. Core Locations DM2 System	No. Core Locations DM1 and C/PT System	Uses
BINDC	72	72	
DCBIN	88	88	
BINHX	44	44	
HXBIN	66	66	
HOLEB	134	134	HOLL, EBPA
HOLPR	100	100	HOLL, PRTY
EBPRT	102	102	EBPA, PRTY
PAPEB	246	246	EBPA
PAPHL	244	244	EBPA, HOLL
PAPPR	192	192	EBPA, PRTY
ZIPCO	154	-	
SPEED	330	330	
HOLL	80	80	
EBPA	80	80	
PRTY	80	80	
EBCCP	128	-	
EBHOL	128	-	
EBPT3	128	-	
CPEBC	128	-	
CPHOL	128	-	
CPPT3	128	-	
HLEBC	128	-	
HOLCP	128	-	
HLPT3	128	-	
PT3EB	128	-	
PT3CP	128	-	
PTHOL	128	-	

APPENDIX F. EXECUTION TIMES

Execution times for the Communications Adaptor subroutines are listed in the adaptor subroutine manual, Form C26-3706.

CONVERSION SUBROUTINES (see Table 13).

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 + 3(45.5)

1130 ISS TIMES (see Tables 14 and 15)

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

DM1 and C/PT System

ILS00 - CARD0 (col), CARD1 (col)
ILS01 - PRNT1
ILS02 - DISK0, DISK1, DISKN
ILS03 - PLOT1
ILS04 - CARD0 (op complt), CARD1 (op complt) WRTY0, TYPE0, PAPT1, PAPTN

● Table 13. Conversion Subroutines

Subroutine	Initialization	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
ZIPCO (DM2 only)-					
All codes except IBM Card Code	270	270	-	-	-
IBM Card Code input	270	374	-	-	-
IBM Card Code output	270	435	-	-	-
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

DM2 System

ILS00 - CARD0 (col), CARD1 (col) PNCH0 (col), PNCH1 (col)
ILS01 - PRNT1
ILS02 - DISK1, DISKN
ILS03 - PLOT1, PLOTX
ILS04 - CARD0 (op complt), CARD1 (op complt), PNCH0 (op complt), PNCH1 (op complt), READ0, READ1, WRTY0, TYPE0, PAPT1, PAPTN, PAPTX, PRNT3, OMPR1

NOTE: In the DM2 system, the Z subroutines are considered to be ISSs and therefore use the appropriate ILSs, e.g., PRNTZ uses ILS01.

3. All times are based on a 3.6 μsec memory.

Table 14. 1130 ISS Times (DM1 and C/PT System)

Subroutine and Function	Times (μsec) (n = word count)
ILS00	112
ILS01	134
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
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
DISKN	
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

Table 15. 1130 ISS Times (DM2 System)

Subroutine and Function	Times (μ sec) (n = word count)
ILS00	112
ILS01	134
ILS02	102
ILS03	112
ILS04	163
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
READ0	
Test	173
Read	546
Feed	523
READ1	
Test	173
Read	576
Feed	553
PNCH0	
Test	165
Punch	763 + 185 (n)
Feed	605
PNCH1	
Test	165
Punch	800 + 190 (n)
Feed	640
WRTY0	
Test	165
Print	228 + 724 (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	$678 + \begin{cases} 418 = \text{if char is 0-9} \\ 472 = \text{if char is A} \\ 624 = \text{if char is B} \end{cases}$
PLOT1 (Cont'd)	$678 + \begin{cases} 752 = \text{if char is C} \\ 224 = \text{per dup. of previous pen motion} \end{cases}$
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
PRNT3	
Test	183
Print	3743 + 45 (n-1)
Control	
Single Space	785
Double Space	6746
Triple Space	12704
Skip to channel 12	817
Skip to channel 1	817
OMPR1	
Test	200
Feed	658
Read	737 + 262 x c
	c = no. of chars. programmed to be read
Disconnect	342
Sel. Stack.	324
DISK1	
Test	158
Read	1021 + 491 x + 1226 y

	x = no. sectors
	y = no. seeks after 1st sector

Write	
Without RBC	1035 + 491 x + 1226 y
Write	
With RBC	1829 + 982 x + 2452 y
Write Imm	689 + 491 x + 489 y
Seek	
1 to-center	1843
By addr	2056
DISKN	
Test	244
Read	1500 + 725 x + 1973 y

	x = no. sectors
	y = no. seeks after 1st sector

Write	
Without RBC	1500 + 725 x + 1973 y
Write	
With RBC	2599 + 1450 x + 3947 y
Write Imm	1085 + 725 x + 1707 y
Seek	
1 to-center	1871
By addr	2151

ARITHMETIC AND FUNCTION SUBROUTINES

The execution times of the arithmetic and function subroutines are shown in Table 16. All times are based on a 3.6 μsec memory; the times containing a decimal point are milliseconds, all other are microseconds.

SPIR (C/PT SYSTEM)

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

Table 16. Arithmetic and Function 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

C26-5929-4

IBM 1130 Printed in U.S.A. C26-5929-3

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENT FORM

IBM 1130 Subroutine Library

Form C26-5929-4

- Your comments, accompanied by answers to the following questions, help us produce better publications for your use. If your answer to a question is "No" or requires qualification, please explain in the space provided below. Comments and suggestions become the property of IBM.

	Yes	No
• Does this publication meet your needs?	□	□
• Did you find the material:		
Easy to read and understand?	□	□
Organized for convenient use?	□	□
Complete?	□	□
Well illustrated?	□	□
Written for your technical level?	□	□

- What is your occupation? ____________________
- How do you use this publication?

  As an introduction to the subject? □

  For advanced knowledge of the subject? □

  For information about operating procedures? □

  As an instructor in a class? □

  As a student in a class? □

  As a reference manual? □

  Other ____________________

- Please give specific page and line references with your comments when appropriate.

COMMENTS

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

YOUR COMMENTS, PLEASE...

This SRL bulletin is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

fold fold

FIRST CLASS
PERMIT NO. 2078
SAN JOSE, CALIF.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY . . .

IBM Corporation
Monterey & Cottle Rds.
San Jose, California
95114

Attention: Programming Publications, Dept. 232

fold fold

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]