CONTROL DATA CORPORATION • COMPUTER EQUIPMENT GROUP
__________ DIVISION

DOCUMENT CLASS IMS CHANGE NO. ______ DATE ______ PAGE NO. ______

PRODUCT NAME RUN 2.3 OBJECT LIBRARY

PRODUCT MODEL NO. C010 MACHINE SERIES 64/65/6600

DEPT. NO. 6231 PROJECT NO. 7MS05

SUBMITTED

____
PROJ. MGR. DATE

REVIEWED

DRYoung 3/4/69
DEPT. MGR. DATE

____
DIRECTOR DATE

____
DRB CHAIRMAN DATE

FINAL APPROVAL

____
□ PPB DATE
□ GEN MGR
□ OTHER

M626

INDEX

1. Introduction

This IMS is organized into three basic sections. Section II deals with routines that are involved with the I/O of the system, Section III deals with the utility routines available in the system, while Section IV covers some of the mathematical routines.

The IMS assumes that the reader has a working knowledge of FORTRAN and knows the general form of the statements involved. For instance, in considering a statement such as PRINT n, L, it is assumed that the reader is familiar with the meanings normally attached to "n" and "L".

A. Definition of Terms

The following table contains terms and abbreviations freely used in the IMS that the reader may not be familiar with:

ABBREVIATIONS

ADDR	=	address
A/N	=	alphanumeric
BA	=	buffer argument (first word address of FET)
BS	=	buffer status
C/R	=	character/record
CTR	=	counter
C/W	=	character/word
EOF	=	end-of-file
EOI	=	end-of-information
EOR	=	end-of-record
FET	=	File Environment Table
FN	=	file number
FWA	=	first word address
I/O	=	Input/Output

LOG	=	logical
LWA	=	last word address
O/L	=	overlay
OP	=	operation
PRU	=	physical record unit
PTR	=	pointer
RA	=	reference address
RC	=	record count
RCL	=	recall
RJ	=	return

B. Concepts

1. Location one - RA+1 is the way of communicating with the operating system.

2. Periodic recall - normal recall in which the central processor is relinquished only until the next time around the monitor loop.

3. Automatic recall - recall in which the central processor is relinquished until bit 0 in a specified location is set to 1, indicating the completion of some peripheral processor activity (eg. an input/output operation on a file).

4. FET - the FET area, especially words 5 and 14 which contain the record count and end-of-file flag, are widely used.

5. Read ahead - the FORTRAN I/O system will read ahead on the binary and coded files in an attempt to make the I/O more efficient. This read ahead makes the backspacing of files involved.

6. Execution File Name Handling - SYSTEM (Q8NTRY) places in RA+2, and the locations immediately following, the file names from the FORTRAN PROGRAM card. The name of the file is left justified and the file's FET address is right justified in the word. (Thus the declared file names replace any actual file names at execution time in the RA area.)

   The logical file name (LFN) which appears in the first word of the FET is determined in one of the three following ways:

   CASE 1: If no actual parameters are specified, the LFN will be the file name from the PROGRAM card.

   Example

```
    :
    RUN(S)
    LGO.
    :
    PROGRAM TEST1 (INPUT, OUTPUT, TAPE1, TAPE2)
```

```
Before    SYSTEM (Q8NTRY)

  RA+2    000 ——————— 000
          000 ——————— 000

After
                                        LFN in FET
  RA+2    INPUT———————FET address       INPUT
          OUTPUT——————FET address       OUTPUT
          TAPE1———————FET address       TAPE1
          TAPE2———————FET address       TAPE2
```

CASE 2: If actual parameters are specified, the LFN will be that specified by the corresponding actual parameter, or the file name from the PROGRAM card if no actual parameter was specified.

Note: It must be recognized that a one-to-one correspondence exists between the actual parameters and the file names found on the PROGRAM card.

Example:

```
:
RUN(S)
LGO(,,DATA,ANSW)
:
PROGRAM TEST2(INPUT, OUTPUT, TAPE1, TAPE2,
TAPE3=TAPE1)
```

```
Before

  RA+2    000—————————000
          000—————————000
          DATA————————000
          ANSW————————000

After
                                          LFN in FET
  RA+2    INPUT———————FET address         INPUT
          OUTPUT——————FET address         OUTPUT
          TAPE1———————FET address         DATA
          TAPE2———————FET address         ANSW
          TAPE3       FET address of      Uses TAPE1 FET
                             TAPE1
```

CASE 3: An equivalenced file name from the PROGRAM card will ignore an actual parameter. The LFN will be that of the file to the right of the equivalence and no new FET will be created.

Example:

```
 :
RUN(S)
LGO(,,DATA,ANSW)
 :
PROGRAM TEST3(INPUT,OUTPUT,TAPE1=OUTPUT,
TAPE2,TAPE3)
```

Before

```
RA+2    000---------000
        000---------000
        DATA--------000
        ANSW--------000
```

After

```
                                          LFN in FET
RA+2    INPUT-------FET address           INPUT
        OUTPUT------FET address           OUTPUT
        TAPE1-------FET address of OUTPUT Uses OUTPUT
                                          FET
        TAPE2-------FET address           ANSW
        TAPE3-------FET address           TAPE3
```

C. Routine SYSTEM

1.0 General Information

System is a multiple entry routine which handles program initialization, error tracing, diagnostic printing, termination of output buffers, and transfers to specified non-standard error procedures.

1.1 Approximate length: 1055B

2.0 Entry Points

2.1 Q8NTRY - Initializes I/O buffer parameters

2.1.1 Calling Sequence and Returns

Entered by doing an RJ to Q8NTRY with the following conditions:

A0 = execution field length
B1 = address of first word of program
B2 = address of word containing length of main program, less buffers
B2+1= address of USASI switch
RA+2= L00---0 if a line count had been specified on the users RUN card or the first file name.
RA+3= Line count if a line limit has been specified on the next file name.
RA+64=Number of file names.

Upon exit, the I/O buffers will have been initialized. See pages 2 and 3 of the General Concepts section for a thorough explanation and examples.

2.2 END - terminates all output buffers, prints the error summary, transfers control to the calling overlay if in overlay mode and not in (0,0) overlay, or, in any other case, exits to monitor.

2.2.1 Calling Sequence and Returns

Entered by doing an RJ to END.

2.3 EXIT - enters EXIT in the dayfile and begins END processing.

2.3.1 Calling Sequence and Returns

Entered by doing an RJ to EXIT.

2.4 STOP - enters STÓP in the dayfile and begins END processing.

2.4.1 Calling Sequence and Returns

Entered by doing an RJ to STOP, where

X7 = the message following STOP on the FORTRAN source statement, or blanks if none.

2.5 ABNORML - Recovery from fatal error changed to non-fatal.

2.5.1 Calling Sequence and Returns

This routine gains control from an execution time routine when an error has been assembled as fatal and during the processing of the job was changed to non-fatal with non-standard error recovery. An abonormal termination is given.

An RJ to SYSTEM must have been done just prior to executing an RJ to ABNORML.

2.6 SYSTEMC - changes entry in SYSTEM'S error table according to the arguments passed.

2.6.1 Calling Sequence and Returns

Entered by doing an RJ to SYSTEMC or by calling SYSTEM from a FORTRAN program, via a special entry point, SYSTEMP.

Entry	B1	= Address of error number
	B2	= Beginning address of parameter list

First = Fatal (non-zero)/non-fatal (0)
Second = Print frequency (8 bits)
Third = Print frequency increment (8 bits)
Fourth = Print limit (12 bits)
Fifth = Non-standard recovery address
Sixth = Maximum traceback for any error

If any parameter in list is negative, then its value is not altered.

2.7 SYSTEMP - Adjusts arguments for use in non-standard recovery, then transfers to SYSTEM for error processing.

2.7.1 Calling Sequence and Returns

Entered by doing an RJ to SYSTEMP, or CALL SYSTEMP with eight parameters from a FORTRAN program. (For a description of FORTRAN utilitzation of SYSTEMP, see Appendix J, FORTRAN Reference Manual.)

Entry B1-B6 = either dummy parameters, or arguments of calling routine
X1 = address of error number
X2 = message address

2.8 SYSTEM - Error tracing, diagnostic printing, termination of output buffers.

2.8.1 Calling Sequence and Returns

Entered by doing an RJ to SYSTEM.

Entry X1 = error number
X2 = address of diagnostic message

Special Entry Conditions

X1 = -1 SYSTEM call from routine END; all normal buffer closeout procedures are to be performed.

X1 = 0 SYSTEM call from routine ABNORML; perform minimal buffer closeout procedures.

When through processing, SYSTEM either transfers to a specified non-standard error recovery address, aborts the job, or returns to the calling routine, depending on the type of error being processed.

3.0 Diagnostics

3.1 Fatal to execution

3.1.1 "Output line limit exceeded," error number 84, will be given and execution of the program halted if the line limit is exceeded.

3.1.2 Job will be aborted by Q8NTRY if the SCOPE 2 common block is too small to hold the files assigned to it.

4.0 External Routines

4.1 ADVIN.

4.1.1 Calling Sequence and Returns

A RJ to ADVIN is made with

X1 = address of first word of FET of the file

ADVIN advances the IN pointer by 1.

4.2 INITL.

4.2.1 Calling Sequence and Returns

A RJ to INITL. is made with the following registers set:

Entry B2 = address of first word of FET of the file, or the complement of either the file name or logical tape number.

X6 = open parameter
X7 = read/write parameter

Exit B2 = address of first word of FET of the file, or unchanged if file not found.
X5 = code and status
X6 = CIO control word
B5 = 1
B6 = -1 if an uncleared EOF on a read request
B6 = -2 if an attempted read after write

4.3 CIO1.

4.3.1 Calling Sequence and Returns

Entry B6 = return address
X1 = address of first word of FET
X2 = function code for CIO

4.4 OPEN. - opens files

4.4.1 Calling Sequence and Returns

This routine is entered by doing an RJ to OPEN with X1 set to the address of the first word of the FET of the associated file and X2 set to the function code for the desired call to OPE.

Upon exit, the file will have been opened in the manner determined by the function code.

4.5 SIO.

4.5.1 Calling Sequence and Returns

This is the entry point for read/write processing. It is entered by doing an RJ to SIO with X1 set to the address of the first word of the FET associated with the file, SIO.CTL control word with bit 2 =0 for a read request and =1 for a write request, and B registers set according to the following criteria:

B7=0 This is a formatted I/O request

For a formatted read request, the 150 character DAT. buffer will be utilized, with blank fill, replacing any zero bytes with blanks.

For a formatted write request B1 contains the number of characters to be written from the DAT. buffer, starting at DAT., through DAT. + B1-1. Characters are expected in R1 format; i.e. right-adjusted, one character per word, with zero fill.

B7≠0 This is an unformatted I/O request

B1 =0 INPUTB/OUTPUTB initialization
B1 > 0 unformatted read/write request, B1 contains the number of words to be transferred
B1 < 0 INPUTB/OUTPUTB termination

Upon exit data will have been transferred between the area defined and the buffer, operating system calls will have been made as required, and IN and OUT will have been updated. Also X4 will have been set as follows:

X4 = 0 EOR
X4 < 0 EOF
X4 > 0 else

4.6 SIO.END - Write an end-of-file indication on a file

4.6.1 Calling Sequence and Returns

This routine is entered by executing an RJ to SIO.END with XI set to the address of the first word of the FET of the file to be accessed. Any remaining data in the buffer is written to the file and then an end-of-file indication is appended to the file.

5.0 Structure

5.1 Q8NTRY - the initialization routine for program buffers. If initialization is not necessary, control is returned to the calling routine, otherwise RA+2 is checked to see whether or not it contains a line count flag. If it does, the line count is converted from BCD to octal and stored, the number of files decremented by two and the remaining files are moved up two cells in the users RA area. The buffers and their associated FET'S are then set up. If the FET is in the SCOPE2 common block, a buffer area is set aside at the LWA+1 of the program. The FET is kept in the common block and not intimately associated with the buffer. If the FET does not reside in the SCOPE2 Common block, the FWA of the FET is set to the FWA of the buffer and the FWA of the buffer changed to LWA+1 of the FET. Files are checked to see whether they have been equivalenced or whether an execution time file name has been denoted. (See section 1-B General Concepts for explanation of equivalenced file handling). Q8NTRY expects to find the file name in the upper 42 bits of the RA word and either the buffer length or the ordinal of the File to which the present File is to be equivalenced. The next FET is set up at the FWA+17 of the last FET. If the FET is not in the SCOPE2 Common block, the FWA of the next FET is moved to the LWA-17 of the previous buffer. When all Files have been initialized, control is returned to the calling program.

5.2 END - traces back until it finds the main program, then fetches the program name and stores it with the format END NAME and calls MSG to write this message in the dayfile. It next makes a special call to SYSTEM to close all output buffers. On return, it issues an ENDRUN call to stop the program.

5.3 EXIT - places EXIT message in the dayfile then transfers control to the SYSTEM call routine on END.

5.4 STOP - places STOP + message into the dayfile then transfers control to the SYSTEM call routine of END.

5.5 ABNORML - Fetches and stores last encountered error number and stores in display code "DETECTED BY (Program Name)". Jumps to SYSTEM to abort job.

5.6 SYSTEMC - changes the structure of the error table entry to fit user specifications. The actual structure of the error table entry is revised and the new structure placed back into the error table.

5.7 SYSTEMP - user callable routine to change the number and message of an error diagnostic.

5.8 SYSTEM - handles all diagnostic printing and traceback for FORTRAN object time routines. Upon entry B1, B2, B3, B7 and A0 are saved. Checks to see that output buffer has been specified and whether or not a special call has been issued from END or INPUTC. For special call from END, the error summary is printed output buffers closed and the job terminated. On special call from INPUTC, the line of error occurance is printed out prior to error diagnostic and traceback information. Otherwise, the specified error table entry is fetched and the diagnostic printed. If a non-standard recovery address has been specified, control is transferred to the specified user SUBPROGRAM. If an error number which is out of range has been specified it is given the number of the last entry on the error table.

II. Input/Output

A. Introduction to I/O

Upon encountering an input or output statement, the compiler generates a calling sequence for use by the execution time subroutines. There is no format cracking done during compilation, so all format diagnostics are produced during execution. Each particular set of I/O statement, i.e., READ, WRITE, ENCODE, BUFFER IN, etc., use an individual execution time subroutine. These subroutines do their own processing within themselves and depend only on the generalized routine SIO$ for the I/O. All information necessary for the completion of the task is generated by the compiler and passed to the execution time routine with successive calls.

In order for a central memory program to communicate with an external file, most information entering or leaving the program must pass through a buffer. For every I/O file, whether it be standard input or output, scratch tape, or data tape, used by the FORTRAN program, a declaration of the file name must be made on the PROGRAM card. Each file name causes a buffer with a minimum length of 1022 words or normally 2022 words to be reserved for its use. The execution time subroutines use the SIO$ routine to communicate with the system CIO (Circular Input/Output) for the physical transfer of data.

The compiler has I/O statement processors which decide from the form of statement which execution time routines are to be called. If a format statement is required, then the address of it must be available during execution. Since most I/O has to pass through a buffer, the address of this buffer must also be known. This information is compiled and sent to the subroutine in one entry. The I/O list is processed and one entry is made for each array or data item. It is during these entries that the format statement is cracked. A final entry is made to signal the end of the list.

The coded input statements (READ n, L; READ (i, n) L; READ INPUT TAPE i, n, L) call INPUTC. The file specified by "i" is read and the data "L" returns to the program according to the format "n". During compilation, the address of the format statement is set into B3 to be passed to the subroutine. The address of a variable format is retrieved by assigning a variable tag to the format statement; thereby fetching the proper address during execution.

Binary data may be read by READ (i) L or READ TAPE i, L. During execution, INPUTB is referenced to read file "i" and insert the data in "L". No special word count is reserved in the data itself. Binary data may be written on a file by WRITE (i) L or WRITE TAPE i, L. Either of these statements requests OUTPTB to transfer the information from "L" to file "i". The number of words written by these statements must be greater or equal to the number of words read by the corresponding READ statement.

OUTPTC is the execution time subroutine called to write coded data on a file. The statements PRINT n, L; PUNCH n, L; WRITE (i, n) L; or WRITE OUTPUT TAPE i, n, L will all cause OUTPTC to be referenced. As with coded input, the format is cracked during execution. There is little difference between the procedure of format cracking used by OUTPTC and INPUTC.

ENCODE and DECODE statements are also available. Storage manipulation to transfer data under a specific FORMAT statement is all that is involved so no physical data file is referenced. Therefore, the list processor used by READ/WRITE compiles a calling sequence to the execution time subroutines OUTPTS and INPUTS. These subroutines work on the same format cracking scheme as OUTPTC and INPUTC.

All the aforementioned statements result in the I/O being accomplished by the execution time subroutines before control is returned to the central program. Therefore, the data is immediately available to the programmer after an I/O statement has been processed. However, the user may choose to buffer his own I/O in which case the BUFFER IN and BUFFER OUT statements are available. BUFFEI and BUFFEO (execution time subroutines) are called, respectively, to initiate the transfer of data via CIO. Control is returned to the central program as soon as SIO$ has requested CIO to initiate the file action. Any block of data, up to normal central memory restrictions, will be handled by these statements. Before using the data the user must check the status of the buffered unit by an IF (UNIT, i). This statement compiles a calling sequence to IOCHEK which is the execution time routine used for checking the status and actually completing the buffer in process.

The execution time subroutines receive all addresses from the program via index registers. A calling sequence is contructed by the compiler for each statement. Listed on the following pages are the calling sequences used during execution.

RUN I/O Calling Sequences

READ, WRITE, PRINT, PUNCH B1 = 0

First entry

B2 = address of FET or complimented address of variable tape number.

B3 = address of format statement or complemented address of variable format number (for coded only, B3 is not set for binary).

RJ INPUTB/INPUTC/OUTPTB/OUTPTC

Intermediate entries

B1 = address of data item or beginning address of array.

B2 = array length (# of words) or 1.

RJ INPUTB/INPUTC/OUTPTB/OUTPTC

Final entry

B1 = -1

RJ INPUTB/ INPUTC/ OUTPTB/ OUTPTC

ENCODE, DECODE

First entry

B1 = address of packed data.

B3 = address of format statement or complemented address of variable format.

B4 = character length or complemented address of variable character length.

RJ INPUTS/OUTPTS

Intermediate entries

B1 = address of data item or beginning address of array.

B2 = array length (# of words) or 1.

RJ INPUTS/OUTPTS

Final entry	B1 = -1
	RJ INPUTS/OUTPTS

BUFFER IN, BUFFER OUT

First entry	B1 = mode constant
	B2 = address of buffer parameter list or complemented address of variable tape number.
Second entry	B7 = address of first word of data block
Third entry	B7 = address of last word of data block
	RJ BUFFEI/BUFFEO

NAMELIST

(single entry)	B1 = address of NAMELIST information
	B2 = address of FET or complemented address of variable tape number.
	RJ INPUTN/OUTPTN

Final entry B1 = -1

RJ INPUTS/OUTPTS

BUFFER IN, BUFFER OUT

(single entry) B1 = mode, or complemented address if variable mode.

B2 = address of FET or complemented address of variable tape number.

B3 = Fwa of data block.

B4 = lwa of data block, or complemented lwa of data block if type double or complex.

RJ BUFFEI/BUFFEO

NAMELIST

(single entry) B1 = address of NAMELIST information

B2 = address of FET or complemented address of variable tape number.

RJ INPUTN/OUTPTN

The following table lists the routine called for each type of source statement:

FORTRAN SOURCE STATEMENT	NAME OF OBJECT TIME ROUTINE REFERENCED
1. READ (u) I	INPUTB
2. WRITE (u) I	OUTPTB
3. READ (u,n) I READ n,l	INPUTC
4. WRITE (u,n) I PRINT n,l PUNCH n,l	OUTPTC
5. DECODE (c,n,v) I	INPUTS
6. ENCODE (c,n,v) I	OUTPTS
7. BUFFER IN	BUFFEI
8. BUFFER OUT	BUFFEO
9. IF (UNIT,i)	IOCHEK
10. IF (IOCHECK,i)	IOCHEC
11. ENDFILE i	ENDFIL
12. IF (ENDFILE,i) IF (EOF,i)	IFENDF IFENDF
13. BACKSPACE i	BACKSP
14. REWIND i	REWINM

B. Use of File Environment Table (FET) and associated symbols by FORTRAN

Symbol	Word	Use
FET.BA	1	As described for SCOPE system, i.e., no special fields.
FET.FIR	2	FIRST pointer and Device Type Indication (user error processing bits, and length of FET are also contained in this word.)
FET.IN	3	IN pointer
FET.OUT	4	OUT pointer
FET.LIM	5	LIMIT pointer (PRU size and the number of PRUS per record block are also contained in this word.)
FET.LCNT	6	The number of user logical records from the beginning of the file to the current position of the file.
FET.MLRS	7	Maximum size logical record information (applicable to S- and L-style tapes only.)
FET.BINB	8	Minus zero, if this file is to be BINARY BLOCKED: it may also contain the line limit for the file OUTPUT.
FET.PARI	9	Negative if a parity error has been encountered while reading.
	10-13	Used as described for SCOPE system.
FET.WDS	14	a) BUFFER IN - word 14 contains

```
                  35          17          0
    -  |          |    FWA    |   LWA+1   |
```

Symbol Word Use

where FWA and LWA are the parameters specified in the BUFFER IN statement.

FET.WDS 14 b) BUFFER OUT - word 14 contains

		35	17	0
+		FWA	LIMIT	

where FIRST and LIMIT are the original FIRST and LIMIT for this file. These are the values restored in word 2 and word 5 by IOCHEK when the BUFFER OUT is completed.

c) Lower 18 bits is set equal to the number of words read by a BUFFER IN statement. This is set by SIO$ (through IOCHEK) upon completing a BUFFER IN request.

FET.EOF 15 a) The file is a buffered file if bit 0 and bit 59 are not the same. The "BUFFER I/O" flag is set by BUFFEI and BUFFEO.

b) Independently of a), if word 15<0, the last operation on this file read an EOF. If word 15≥0, no EOF was read. This flag, not the buffer status, is used to detect trying to read past an uncleared EOF. BUFFER I/O makes no use of this flag.

FET.LMAX 16 Output file line limit. This is present by the compiler from a parameter on the RUN card. At execution time OUTPTC decrements this value; if it reaches zero an

Symbol	Word	Use
		error is indicated and the job is aborted. (RUN only).
FET.TRIG	17	Bits 29-0 (right justified) contains the initiate I/O TRIGGER value. Bits 59-30 (right justified) contains the buffer length - TRIGGER value. The TRIGGER value is used by SIO$ to determine the point at which to initiate another I/O call. The TRIGGER value is the maximum of (1 PRU, "TRIGGER" percent of buffer size) "TRIGGER" is an assembly parameter in SIO$ with initial definition of: TRIGGER MICRO 1,0,/25/

C. Description of Routine SIO$

1.0 General Information

The function of this routine is to perform various I/O operations for the other FORTRAN object time routines. It contains a collection of I/O functions that are used by the FORTRAN object I/O library. It was written to provide:

a) centralized I/O communications with the operating system

b) centralized responsibility for the management of buffers

c) a centralized location for file positioning

1.1 Approzimate length: 1100B

2.0 Entry Points

2.1 DAT. - character string buffer - a data entry

DAT. is a 150 word character string buffer used by the routines which do coded I/O. It holds one, right-justified character per word.

2.2 INITL. - this routine accomplishes file related initialization for FORTRAN input/output routines.

2.2.1 Calling Sequence and Returns

RJ INITL. with the following registers set:

B2 - address of the first word of the file environment table for the file to be referenced
or
the complement of the address of the word containing the logical unit designation for the file to be referenced.

X6 - open parameter to be used if the file must be opened.

X7 - nominal CIO request (i.e. READ or WRITE with the correct mode included).

Upon exit, file related initialization for FORTRAN I/O routines will have been accomplished if B2 and B6 are positive. The file will be correctly positioned, the control word SIO.CTL will be correctly set up for the read/write request, and the file will be opened, if necessary. Error conditions are as follows:

B2 < 0 -- FET not found
B6 = -1 uncleared EOF
B6 = -2 read request follows write on file

2.3 SIO.

2.3.1 Calling Sequence and Returns

This is the entry point for read/write processing. It is entered by doing an RJ to SIP. with X1 set to the address of the first word of the FET associated with the file, SIO.CTL control word with bit 2 =0 for a read request and =1 for a write request, and B registers set according to the following criteria:

B7 =0 This is a formatted I/O request

For a formatted read request, the 150 character DAT. buffer will be utilized, with blank fill, replacing any zero bytes with blanks.

For a formatted write request B1 contains the number of characters to be written from the DAT. buffer, starting at DAT., through DAT. + B1-1. Characters are expected in R1 format; i.e. right-adjusted, one character per word, with zero fill.

B7 ≠0 This is an unformatted I/O request

B1=0 INPUTB/OUTPUTB initialization
B1 > 0 unformatted read/write request, B1 contains the number of words to be transferred
B1 < 0 INPUTB/OUTPUTB termination

Upon exit data will have been transferred between the area defined and the buffer; operating system calls will have been made as required, and IN and OUT will have been updated. Also X4 will have been set as follows:

X4 = 0 EOR (end of SCOPE - logical - record)
X4 > 0 EOF (end of file)
X4 < 0 else

2.4 SIO.END - Write an end-of-file indication on a file

2.4.1 Calling Sequence and Returns

This routine is entered by executing an RJ to SIO.END with X1 set to the address of the first word of the FET of the file to be accessed. Any remaining data in the buffer is written to the file and then an end-of-file indication is appended to the file.

2.5 OPEN. - opens files

2.5.1 Calling Sequence and Returns

This routine is entered by doing an RJ to OPEN. with X1 set to the address of the first word of the FET of the associated file and X2 set to the function code for the desired call to open the file.

Upon exit, the file will have been opened in the manner determined by the function code.

2.6 FIZBAK. - positions file when backspacing

2.6.1 Calling Sequence and Returns

This area is entered by doing an RJ to FIZBAK. with X1 set to the address of the first word of the FET associated with the file. This routine is used when a backspace or a write after a read is to be performed. It backspaces the file to the current PRU, reads the PRU into the buffer starting at FIRST, leaves OUT pointing to the logically next coded record and IN pointing to the last word +1 of the PRU, and leaves the file positioned physically following the PRU. All operations are done with recall so they will be done on return. Upon exit B6 will have been set to one if the current PRU is an end-of-file, otherwise B6 will have been set to zero.

2.7 POSFIL. - positions file to before PRU

2.7.1 Calling Sequence and Returns

This area is entered by doing an RJ to POSFIL. with X1 set to the first word of the FET associated with the file. POSFIL. repositions the file like FIZBAK., except upon exit the file is positioned physically before the current PRU, and IN and OUT are interchanged.

2.8 BKSPRU. - backspaces one PRU

2.8.1 Calling Sequence and Returns

This area is entered by doing an RJ to BKSPRU. with X1 set to the address of the first word of the FET associated with the file. Upon exit the file will have been backspaced one PRU (and recall will have been used).

2.9 RDPRU. - reads one PRU

2.9.1 Calling Sequence and Returns

RJ RDPRU. with the following registers set:

X1 - address of the first word of FET

This routine will read the next PHYSICAL RECORD UNIT on the file specified by the contents of register The read is accomplished by setting the buffer empty and with room only for one PRU (determined by the PRU size placed in the FET in word FET.LIM).

The following registers are destroyed:

B6,
X0, X2, X3, X4, X5, X6, X7
A2, A3, A5, and A6

2.10 FIZBA. - reposition file after current PRU.

2.10.1 Calling Sequence and Returns

```
(B5) = 1
(B7) = additional number of words to backspace
RJ     FIZBA
(X1)   FET address
```

Return

```
(B7) = 0 if current PRU is not EOF
(B7) ≠ 0 if current PRU is EOF
```

Function: reposition file and backspace the number of words specified in B7.

If B7 = 0 upon entry, this routine is identical to FIZBAK. except that the file must be recorded as one scope logical record.

2.11 POSFI.

2.11.1 Calling Sequence and Returns

```
B5 = 1  (X1) = FET address
RJ  POSFI.
```

Return

```
B7 = 0  current PRU is not EOF
B7 ≠ 0  current PRU is EOF
```

Function: identical to POSFIL.

2.12 CIO1. - this is the CIO communication routine.

2.12.1 Calling Sequence and Returns

Jump to CIO1. with the following registers set:

X1 - address of first word of FET
X2 - CIO function (referred to as F)
B6 - return address after calling CIO

If F is zero then the routine RCL will be called. In this case if F is negative (i.e. F =-0), then an atuomatic recall on the address in register X1 will be issued.

When F is non-zero and positive, a normal CIO call will be issued and control will be returned to the user as soon as the call has been placed. If F is non-zero and negative then an automatic recall call of CIO will be placed and control will be returned to the caller when the function (the complement of F) has been completed.

Registers destroyed: X5, X6, X7
X5, A6, A7

2.13 RCL1.- This is a central processor recall routine.

2.13.1 Calling Sequence and Returns

Jump to RCL1. with the following registers set:

X1 - (necessary only if X2 is negative) address for which to issue an automatic recall
X2 - recall type indicator (if positive then normal recall, else automatic recall)
B6 - address to return control to after the recall has been satisfied.

Registers destroyed: X5, X6
X5, A6

2.14 MVWDS. - this is a general move-words routine

2.14.1 Calling Sequence and Returns

RJ MVWDS. with the following registers set:

B2 - count of words to be moved
B5 - 1
B7 - destination address for block
X2 - origin address of block

Return will be made after the block of COUNT words has been moved. Upon exit the following registers will be defined:

A5 - address of last location loaded from
A7 - address of last location stored into.
X6 - contents of register A3

Registers destroyed: B4
A4, A5, A6, A7
A4, X5, X6, X7

2.15 ADVIN. - advance IN pointer

2.15.1 Calling Sequence and Returns

This routine is entered by doing an RJ to ADVIN. with X1 set to the address of the first word of the FET of the associated file. ADVIN. advances IN by 1. Registers destroyed are: A2, A5, A6, X2, X5, X6, B1 and B2.

2.16 S10.CTL - SIO. control word

This is a data entry whose format is set up by INITL.

The input/output control word SIO.CTL is set up:

a. If the file does not reside on a one-half inch magnetic tape then SIO.CTL is -
```
   VFD 6/0,36/0,18/CIO.CODE
```

b. If the file does reside on one-half inch magnetic tape then SIO.CTL is -
```
   VFD 6/DEVICE,36/0,18/CIO.CODE
```

Where DEVICE is taken from bits 48-53 of word 2 of the file environment table. The value of DEVICE carries the following information:

DEVICE = SSLLDD (Base 2)

STYLE SS =	00	Scope Internal Tape
	01	X-tape
	10	X-tape
	11	L-tape
LABELS LL =	00	No Labels
	01	Scope standard labels
	10	Scope option label
	11	(reserved)
DENSITY DD =	00	HI (556)
	01	LO (200)
	10	HY (800)
	11	(reserved)

If and only if the file resides on an S- or L-tape the CIO.CODE will be the READN or WRITEN commands rather than the nominal READ or WRITE commands.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 BUFFER SIZE TOO SMALL ON XXXXXXX RECOMPILE WITH BUFFER SIZE GE PRU SIZE will be printed from OPEN. if the file's buffer is less than one PRU in size.

4.0 External Routines

4.1 GETBA

4.1.1 Calling Sequence and Returns

Entry B2 = complement of address of either the file name or logical tape number

Exit B2 = FET address
X3 = file name

5.0 Structure

5.1 Local Routines

5.1.1 RDSPACE - wait for sufficient read space

Calling procedure -

RJ RDSPACE with the following registers set:

B5 - 1
X1 - address of first word of FET

Return will be made when there is at least one word in the buffer, or if an end condition is encountered. If FILE. upon a non-end-condition exit the following information is available:

X0 - first
X3 - limit
X4 - in
X5 - out
X6 - number of contiguous words available
X7 - space (total number of words read into the buffer)

Upon an end-condition exit the following information is pertinent:

X7 - 0 (this will serve as a signal of an end condition)

X2 - 0 (end-of-file encountered)
1 (end-of-scope-logical-record encountered)

Registers destroyed: X0, X2, X3, X4, X5, X6, X7
A2, A3, A4, A5, A6, and A7
B6

5.1.2 WRSPACE - wait for sufficient write space

Calling procedure -

RJ WRISPACE with the following registers set:

B4 - size
B5 - 1
X1 - address of first word of FET

Return will be made when available space in the buffer exceeds the value SIZE. If necessary CIO will be called to write out a partion of the buffer. Upon exit the following information is available:

X0 - first
X3 - limit
X4 - in
X5 - out
X6 - number of contiguous words available
X7 - space (total number of unfilled words in the buffer)

Registers destroyed: X0, X2, X3, X4, X5, X6, X7
A2, A3, A4, A5, A6, and A7
B6

Note: Filling SPACE words would cause the buffer to appear to be empty since there must be at least one empty word at all times in a buffer if there is any data in the buffer which has not been transmitted.

5.1.3 GETLIM - find out read space available

Calling procedure -

RJ GETLIM with the following registers set:

B5 = 1
X1 = address of first word of FET

Return will be made with information as follows:

X4 = FIRST
X3 = IN
X2 = OUT
X5 = LIMIT
B2 = number of contiguous words available
X7 = total space available

5.1.4 CHKPAR = check parity on last read

Calling procedure -

RJ CHKPAR with the following registers set:

B5 = 1
X1 = address of first word of FET
X3 = status word

Return will be made with information as follows:

X2 = CIO command for next read
Parity bit on in FET word for parity indication (FET.PARI)

5.2 (INITL.)

Upon normal return INITL. will have accomplished the following:

1. If register B2 does not contain the address of the first word of the file environment table then GETBA will be called. If the file cannot be found then an error return will be made (see below).

2. If this is the first access on the file specified then OPEN. will be called to open the file (using the parameter supplied by the caller.)

3. The input/output control word SIO.CTL is set up:

   a. If the file does not reside on a one-half inch magnetic tape then SIO.CTL is -
      VFD 6/0,36/0,18/CIO.CODE

   b. If the file does reside on one-half inch magnetic tape then SIO.CTL is -
      VFD 6/DEVICE,36/0,18/CIO.CODE

   Where DEVICE is taken from bits 48-53 of word 2 of the file environment table. the value of DEVICE carries the following information:

   DEVICE = SSLLDD (Base 2)

STYLE	SS =	00	SCOPE internal table
		01	X-tape
		10	S-tape
		11	L-tape
LABELS	LL =	00	No Lables
		01	SCOPE standard labels
		10	SCOPE option label
		11	(reserved)

```
DENSITY DD = 00  HI (556)
             01  LO (200)
             10  HY (800)
             11  (reserved)
```

If and only if the file resides on an S- or L-tape the CIO.COPE will be the READN or WRITEN commands rather than the nominal READ or WRITE commands.

4. If the present operation is a write then POSFIL. will be called to position the file if the last operation was a read or a backspace. In either case the end-of-file flag will set to the off state.

5.3 (SIO.) - The SIO.CTL word is examined to determine if the request is a read or a write request. B7 is then examined to determine if the request was for formatted or unformatted I/O.

5.3.1 Formatted read request. The maximum number of words in a formatted input record is set. If the file to be referenced resides on S- or L-style magnetic tape, a call to RDSPACE is made to wait until there is some information in the buffer, or until an empty buffer with EOR or EOF status is found. The minimum of max input words and the number of words in the tape block are moved to an area beginning at BURSTFWA. The OUT pointer is updated to reflect the removal of the tape block from the buffer. Also, any unused bits in the last word moved are masked out. (If the record wraps around the circular buffer, the move is done in two segments.) The record is then reformatted into one-character-per-word form, right justified with zero fill.

If the file to be referenced does not reside on S- or L-style magnetic tape, the request is considered as a formatted read from an internal file. The requested words will be transferred to an areas beginning at BURSTFWA. A call to RDSPACE is made to wait until there is at least one word in the buffer, or an end condition (EOR or EOF) occurs.

The value of LIMIT is saved, and the first word is fetched from the buffer. The number of contiguous words which may be moved from the buffer to satisfy the request is determined. The data is transferred from the buffer to the designated area until either the number of words requested is reached or a line terminator is encountered. Transfer of words continues until all words in the buffer are transferred. The OUT pointer is updated.

The record is then burst into one character-per-word format.

5.3.2 Unformatted read request

If the file to be referenced resides on S- or L-style magnetic tape, the control word is examined to determine record size, and from that, whether or not an end of record has been encountered. Full records are moved in contiguous sections only; the last record of a move requiring special casing, since the number of words and unused bit count will differ in the control word. Control is returned through SIO.

If the file to be referenced does not reside on S- or L-style magnetic tape, information is transferred in groups of two PRU at a time, the last request being a short request, IN, OUT, and FIRST are all appropriately updated, and if the buffer is full, control is transferred to entry CIO1.

5.3.3 Formatted write request

Information to be written is presumed to reside in the DAT. buffer one character per word, right justified with zero fill. The number of characters to be packed is in B1. The output line is first packed 10 characters per word. The beginning address of the packed line and its length are computed.

If the file to be referenced resides on S- or L-style magnetic tape, the control word for the line is formed and inserted in front of the packed line. The beginning address of the packed line is moved back one word, and the number of words to be transferred to the file buffer is incremented by one. A call to WRSPACE is made to wait until there is room enough in the buffer to append this line. The count of words to be moved by MVWDS is set to the smaller of the number of words which comprise the line and the number of contiguous words available. Also the total amount of space available is reduced by the number of words in the line. If all the words in the line were not moved, then MVWDS. is called again to move the remaining words into the buffer. IN is updated along with TRIGGER information and exit procedures performed.

If the file to be referenced does not reside on S- or L-style magnetic tape, it is considered to be a formatted write on an internal (or X-tape) file. Formatted writes are limited to 14 word records (136 characters + 4 bytes of binary zeros.) As many words as possible are moved into the contiguously available space in the buffer; the IN pointer is up-dated and when there are no more words to move, exit procedures are performed.

5.3.4 Unformatted write request

Initialization and termination procedures are performed where necessary. If the file resides on S or L-style magnetic tape, unfinished 512 word blocks are filled out, the control word for each block is formed, and an RJ to WRSPACE is made to wait until there is space in the buffer to append the fill-words. Also, the number of words which remain to be transferred to the buffer is decremented. When there is enough space, the words are moved to the buffer. If the block is not full, control is returned to the caller. If the move completely fills the block, IN is updated and the new IN address is zeroed out, in preparation for the next block's control word. The above processing is repeated until all words are transferred, and exit procedures are then performed.

If the file does not reside on S- or L-style magnetic tape, it is considered to be an unformatted write on an internal (or X-tape) file. An RJ is made to WRSPACE to wait until there is space for more than one word in the buffer. As many words as possible are moved into the contiguously available space in the buffer. The IN pointer is up-dated, and, if there are no more words to store, exit procedures are performed.

5.4 (SIO.END) This routine writes an end-of-file indication on a file. If there is any activity currently in progress on the file, SIO.END waits for it to cease.

If the file is unformatted and resides on an S- or L-style magnetic tape, CIO1. flushes out what remains in the buffer. A control word is inserted along with a data word to simulate a Scope internal EOF. The IN and OUT pointers are updated and a write nonstop command is set up.

If the file does not reside on an S- or L-style magnetic tape, or the file is being used for Buffered I/O, a standard write command is set up. CIO1. is called, and control is returned to the caller through the entry point.

5.5 (OPEN.) This routine will open the specified file using the supplied open parameter. Using the assembly parameter TRIGGER an input/output trigger value will be set up in the file environment table (in word FET.TRIG). If the buffer is too small to accept a single physical record unit of the device on which the file resides then the job will be aborted and an error message issued to the dayfile.

Return will be made to the caller after the open activity has been accomplished.

Registers destroyed: X0, X2, X3, X4, X5, X6, X7
A2, A3, A4, A5, A6, and A7

5.6 (FIZBAK.) This area is called when a backspace or a write after a read is to be performed. It backspaces the file to the current PRU, reads the PRU into the buffer starting at FIRST, leaves OUT pointing to the logically next coded record and IN pointing to the last word + 1 of the PRU, and leaves the file positioned physically following the PRU. Unless the PRU was an end-of-file, then it leaves the file positioned physically after the end-of-file. All operations are done with automatic recall so they will be completed on return.

5.7 (POSFIL.) This areas uses FIZBAK. to reposition the file. Then the IN and OUT pointers are reversed to account for a call to BKSPRU. which follows. The buffer status is set to a write completed and a return is made to the calling routine.

5.8 (BKSPRU.) This area backspaces one PRU. The buffer status is obtained by a call to CKSTAT.. If the buffer is busy, an address to return to, BKA, is set up and a call is made to CIOL. to go into automatic recall.

5.9 (RDPRU.) A call is made to CKSTAT. to obtain the buffer status. If the buffer is busy, an address to return to, RDA, is set up and a call is made to CIO1. to go into automatic recall.

5.10 (FIZBA.) This routine is called when a backspace or a write after a read is to be performed on a blocked binary file. It backspaces to the current PRU (or "n" words back from the current PRU), rereads the PRU in question starting at FIRST, leaves OUT pointing to the last word backspaced over and IN pointing to the last word + 1 of the PRU. If the previous record was an end-of-file then the file will be positioned physically after the end-of-file. Upon return the positioning operation will have been completed.

(POSFI.) This routine uses FIZBA. to reposition the file for a write after a read. The file is repositioned, the IN and OUT pointers are reversed and the file physically positioned before the PRU in the buffer.

D. Detailed Documentation of I/O Routines

D1. BACKSP

1.0 General Information

The function of this routine is to backspace one user logical record on unit i in response to the FORTRAN statement BACKSPACE i.

1.1 Approximate Length: 160B

2.0 Entry Points

2.1 BACKSP

2.1.1 Calling Sequence and Returns

The routine is entered by doing an RJ to BACKSP with registers set as follows:

X6 = (1) the FET address or (2) the complement of the address of either the file name or tape number

Upon exit the tape will have been backspaced one user logical record.

3.0 Diagnostic

3.1 Fatal to Execution

3.1.1 "Unassigned medium,file XXXXXXX," error number 53, will be given if no file was found for the tape unit.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM must be made prior to an RJ to ABNORML

4.2 INITL.

4.2.1 Calling Sequence and Returns

An RJ to INITL is made with the following registers set:

Entry B2 = address of first word of FET of the file, or the complement of either the file name or logical tape number.
X6 = open parameter
X7 = read/write parameter

Exit B2 = address of first word of FET of the file, or unchanged if file not found.
X5 = code and status
X6 = CIO control word
X5 = 1
X6 = -1 if an uncleared EOF on a read request
B6 = -2 if an attempted read after write

4.3 CIO1

4.3.1 Calling Sequence and Returns

Entry B6 = return address
X1 = address of first word of FET
X2 = function code for CIO

4.4 GETBA

4.4.1 Calling Sequence and Returns

Entry B2 = complement of address of either file name or logical tape number

Exit B2 = FET address
X3 = file name

4.5 ADVIN

4.5.1 Calling Sequence and Returns

An RJ to ADVIN is made with

X1 = address of first word of FET of the file

ADVIN advances the IN pointer by 1.

4.6 BKSPRU

4.6.1 Calling Sequence and Returns

Entry X1 = address of the first word of FET

4.8 FIZBA

4.8.1 Calling Sequence and Returns

Entry B5 = 1
B7 = additional number of words to backspace
X1 = address of first word of FET

Exit B7 = 0 if current PRU is not EOF
B7 ≠ 0 if current PRU is EOF

4.9 S10.END - flushes a buffer

This routine is entered by executing an RJ to SIO.END with X1 set to the address of the first word of the FET of the file to be accessed. Any remaining data in the buffer is written to the file and then an end-of-file indication is appended to the file.

4.10 SYSTEM

4.10.1 Calling Sequence and Returns

Entry X1 = error number
X2 = address of diagnostic message

D2. BUFFEI

1.0 General Information

This routine is called in response to a FORTRAN BUFFER IN statement. It issues a call to CIO1 to transfer information from the unit specified in the BUFFER IN statement to (1) the user array for S and L tapes or (2) the program buffer for the file for X and I tapes. Each FORTRAN BUFFER IN statement produces three calls to BUFFEI.

1.1 Length - 113B

2.0 Entry Points

2.1 BUFFEI

2.1.1 Calling Sequence and Returns

The routine is entered by doing an RJ to BUFFEI. It is entered three times in order to pass parameters. The register conditions for the entries should be:

Entry 1

B2 contains (1) the address of the FET associated with the file or (2) the complement of the address of the logical file name or tape number.
B1 contains the mode indicator
Zero - even parity (coded)
Non-zero - odd parity (binary)

Entry 2

B7 contains the first word address (FWA) of the area to which the data is to be transferred.

Entry 3

B7 contains the last word address (LWA) of the area to which the data is to be transferred.

Upon exit from entry 3, the request for transfer of the data will have been issued, there is no guarantee that the transfer has actually taken place.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXXXX," error number 54, will be given if no file was found for a tape unit.

3.1.2 "*Buf In * Endfile XXXXXXX," error number 54 will be given if an attempt is made to read past an uncleared EOF.

3.1.3 "*Buf In * Last Op Write," error number 56, will be given if the first word address of the storage area is greater than or equal to the last word address.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM must be made prior to an RJ to ABNORML.

4.2 INITL

4.2.1 Calling Sequence and Returns

An RJ to INITL. is made with the following registers set:

Entry B2 = address of first word of FET of the file, or the complement of either the file name or logical tape number.
X6 = open parameter
X7 = read/write parameter

Exit = B2 = address of first word of FET of the file, or unchanged if file not found.
X5 = code and status
X6 = CIO control word
B5 = 1
B6 = -1 if an uncleared EOF on a read request
B6 = -2 if an attempted read after write

4.3 CIO1

4.3.1 Calling Sequence and Returns

```
Entry     B6 = return address
          X1 = address of first word of FET
          X2 = function code for CIO
```

4.4 RCL1

4.4.1 Calling Sequence and Returns

```
Entry     B6 = return address
          X2 = recall type indicator (if
               positive, then normal recall,
               else automatic recall.)
          X1 = (necessary only if X2 is
               negative) address for which
               to issue an automatic recall.
```

4.5 SYSTEM

4.5.1 Calling Sequence and Returns

```
Entry     X1 = error number
          X2 = address of diagnostic message
```

5.0 Structure

5.1 No registers are saved.

5.2 If this is the last entry, a branch is taken to LAST. If this is the first entry, a branch is taken to FIRST. Otherwise, the block start location is saved. The flag for the third entry is set up and a branch is taken to exit.

5.3 (FIRST) The I/O mode and file indicator are saved. The flag for the second entry is set up and the trace-back information is stored in NAME +1. A branch is taken to exit.

5.4 (LAST) The flag for a subsequent first entry is set up.

5.5 INITL. is called to initialize the file. If the file has not been found, an error exit is taken. If there has been an attempt to read past an EOF, one error exit is taken. Otherwise buffering of input takes place. Processing for I or X tapes differs from processing for S- or L-style tapes since a control word is involved for S- or L-style tapes. A read code is set, a return point of BUFFEI is set in B6, and control is given to CIO1.

D3. BUFFEO

1.0 General information

This routine is called in response to a FORTRAN BUFFER OUT statement. It issues a call to CIO1. to initiate a transfer of data directly from the user array to the external unit specified in the BUFFER OUT statement. For each FORTRAN BUFFER OUT statement BUFFEO is entered three times.

1.1 Approximate length - 100B

2.0 Entrv Points

2.1 BUFFEO

2.1.1 Calling Sequence and Returns

This routine is entered by doing an RJ to BUFFEO. It is entered three times in order to pass parameters. The register conditions for the entries should be:

Entry 1

B2 contains (1) the address of the FET associated with the file or (2) the complement of the address
B1 contains the mode indicator
Zero - even parity (coded)
Non-zero - odd parity (binary)

Entry 2

B7 contains the first word address (FWA) of the data to be transferred.

Entry 3

B7 contains the last word address (LWA) of the data to be transferred.

Upon exit from entry 3, the request for transfer of the data will have been issued. There is no guarantee that the transfer has actually taken place.

3.0 Diagnostics

3.1.1 "Unassigned medium, file XXXXXXX," error number 58, will be given if no file was found for the tape unit.

3.1.2 "*Buf Out * FWA .GT. LWA," error number 59, will be given if the first word address of the storage area is greater than or equal to the last word address.

3.1.3 "Buff OUT," array too large, error number 114, will be given if the array given is larger than 14 words for a coded write.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM must be made prior to an RJ to ABNORML.

4.2 INITL

4.2.1 Calling Sequence and Returns

An RJ to INITL. is made with the following registers set:

```
Entry      B2 = address of first word of FET
                of the file, or the complement
                of either the file name or
                logical tape number.
           X6 = open parameter
           X7 = read/write parameter

Exit       B2 = address of first word of FET
                of the file, or unchanged if
                file not found.
           X5 = code and status
           X6 = CIO control word
           B5 = 1
           B6 = -1 if an uncleared EOF on a
                read request
           B6 = -2 if an attempted read after
                write
```

4.3 CIO1

4.3.1 Calling Sequence and Returns

```
Entry     B6 - return address
          X1 = FET address
          X2 = function code for CIO
```

4.4 RCL1

4.4.1 Calling Sequence and Returns

```
Entry     B6 - return address
          X2 = recall type indicator (if
               positive, then normal re-
               call, else automatic recall.)
          X1 = (necessary only if X2 is
               negative) address for which to
               issue an automatic recall.
```

4.5 SYSTEM

4.5.1 Calling Sequence and Returns

```
Entry     X1 = error number
          X2 = address of diagnostic message
```

5.0 Structure

5.1 No registers are saved.

5.2 If this is the last entry, a branch is taken to LAST. If this is the first entry, a branch is taken to FIRST. Otherwise, the block start location is saved. The flag for the third entry is set up and a branch is taken to exit.

5.3 (FIRST) The I/O mode and file indicator are saved. The flag for the second entry is set up and the trace-back information is stored in NAME +1. A branch is taken to exit.

5.4 (LAST) The flag for a subsequent first entry is set up.

5.5 INITL is called to initialize the file. If the file has not been found, an error exit is taken. Otherwise buffering of output takes place. Processing for I or X tapes differs from processing S- or L-style tapes since, for the latter, a control word is involved. A write code is set, a return point of BUFFEO is set in B6, and control is given to CIO1.

D4. ENDFIL

1.0 General Information

The function of this routine is to write an end-of-file on magnetic tape unit i in response to the FORTRAN statement ENDFILE i.

1.1 Length: 41_8

2.0 Entry Points

2.1 ENDFIL

2.1.1 Calling Sequence and Returns

The routine is entered by doing an RJ to ENDFIL. X6 should contain the complement of the address of either the file name or the logical tape number associated with the tape unit; or X6 should contain the address of the first word of the FET associated with the file. Upon exit, an end-of-file will have been written on the tape.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXXXX," error number 60, will be given if no file was found for the tape unit.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM must be made prior to an RJ to ABNORML.

4.2 GETBA

4.2.1 Calling Sequence and Returns

```
Entry B2 = complement of address of either
           file name or logical tape number

Exit       B2 = FET address
           X3 = file name
```

4.3 INITL.

4.3.1 Calling Sequence and Returns

An RJ to INITL. is made with the following registers set:

Entry B2 = address of first word of FET of file or the complement of either the file name or logical tape number.
X6 = open parameter
X7 = read/write parameter

Exit B2 = address of first word of FET of the file, or unchanged if file not found.
X5 = code and status
X6 - CIO control word
B5 = 1
B6 = -1 if an uncleared EOF on a read request

4.4 ADVIN

4.4.1 Calling Sequence and Returns

This routine is entered by doing an RJ to ADVIN. with: X1 = address of first word of FET of the file FET of the file.

4.5 POSFI

4.5.1 Calling Sequence and Returns

Entry B5 = 1 (X1) = FET address
RJ POSFI.

Exit B7 = 0 current PRU is not EOF
B7 ≠ 0 current PRU is EOF

4.6 SIO.END

4.6.1 Calling Sequence and Returns

This routine is entered by executing an RJ to SIO.END with X1 set to the address of the first word of the FET of the file to be accessed. Any remaining data in the buffer is written to the file and then an end-of-file indication is appended to the file.

4.7 SYSTEM

4.7.1 Calling Sequence and Returns

Entry X1 = error number
X2 = address of diagnostic message

5.0 Structure

5.1 The contents of registers B2 and B6 are saved. If necessary, GETBA is called to obtain the address of the FET. If the file cannot be referenced, an error exit is taken. If the file is blocked, binary file positioning is as follows: If the last operation was a read, POSFI is called to position the file as necessary. ADVIN. is called if it is necessary to advance the IN pointer. If the file is a buffered file, no positioning is done. In all other cases, INITL. is called to position the file. When the file has been positioned, the record count is incremented and SIO.END is called to write an end of file. The saved B registers are then restored, and control is returned to the caller via the entry point.

D5. GETBA

1.0 General Information

The function of this routine is to search the list of file names to find the address of the first word of the FET of the associated file. GETBA is called by various I/O routines when the logical file is a variable.

1.1 Approximate length: 20B

2.0 Entry Points

2.1 GETBA

2.1.1 Calling Sequence and Returns

The routine is called by an RJ to GETBA. B2 should contain the complemented address of the word containing a file name or a logical unit number. Upon exit, B2 will still contain the negative address if the file is not found. If the file is found, B2 will contain the address of the first word of the FET of the associated file.

3.0 Diagnostics Produced: None

4.0 External Routines: None

5.0 Structure

5.1 The specified location is examined. If it contains a logical unit number, (N) or (NN), the number is converted to "TAPEN" or "TAPENN". If the location does not contain a number, the upper 42 bits are extracted and used as a file name.

5.2 (SEARCH) The low core locations starting at RA +2 are examined to find a match for the file name. If it is not found before the file list is exhausted, GETBA exits with B2 unmodified (still negative).

5.3 (HIT) If the file name is found, B2 is set to the address of the first word of the FET of the associated file, and the routine exits.

D6. IFENDF

1.0 General Information

The function of this routine is to check the previous read operation to determine if an end-of-file has been encountered on unit i where i is a non-buffered unit.

1.1 Length: 41^{B}

2.0 Entry Points

2.1 IFENDF

2.1.1 Calling Sequence and Returns

The routine is entered by doing an RJ to IFENDF. X6 should contain the complement of the address of either the file name or the logical tape number associated with the tape unit; or X6 should contain the address of the first word of the FET associated with the file. Upon exit X6=1 if an end-of-file was encountered by the previous read operation, otherwise X6=0.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXXXX," error number 61, will be given if no file was found for the tape unit.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Return

An RJ to SYSTEM must be made prior to an RJ to ABNORML

4.2 GETBA

4.2.1 Calling Sequence and Returns

Entry B2 = compement of address of either
file name or logical tape number

Exit B2 - FET address
B3 = file name

4.3 SYSTEM

4.3.1 Calling Sequence and Returns

Entry X1 - error number
X2 = address of diagnostic message

5.0 Structure

5.1 Register B2 is saved. If necessary, GETBA is called to obtain the FET address. If the file cannot be referenced, an error exit is taken. If the end-of-file flag is not set in the FET, B2 is restored and the routine exits. If the end-of-file flag is set, it is cleared; and if the file is a BUFFER file, B2 is restored and the routine exits. If the file is a binary file and resides on an S or L tape, the end-of-record is incremented over. On all S or L tape files, the end-of-file bit is cleared from word 1 of the FET. If the file does not reside on an S or L tape and is not blocked binary, the end-of-record and end-of-file bits are cleared from word 1 of the FET. B2 is then restored and the routine exits.

D7. INPUTB

1.0 General Information

The function of INPUTB is to transfer one logical record of binary information from a file unit to storage locations, as specified by FORTRAN READ statements:

```
READ(i)L
READ TAPE i,L
```

If L is omitted, the routine skips over one logical record.

1.1 Approximate Length: 254B

2.0 Entry Points

2.1 INPUTB

2.1.1 Calling Sequence and Returns

There are three entries to this routine. A first entry to initialize input, one intermediate entry for each array or data item, and a final entry to signal the end of the list and terminate input.

```
First Entry:  B2 = FET address or comple-
                   mented address of
                   variable tape number.
              RJ   INPUTB1.

Intermediate Entries:  B1 = Address of data
                            item or begin-
                            ning address of
                            array

                       B2 = number of words

                       RJ   INPUTB.

Final Entry:  B1 = 1

              RJ   INPUTB.
```

Upon exit, one logical record will have been read from the file unit specified.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXXXX," error number 62, will be given if no file was found for the tape unit.

3.1.2 "Read-Write sequence error XXXXXXX," error number 90, will be given if the previous operation on the file was a write.

3.1.3 "Bin Input *Endfile *XXXXXXX," error number 63, will be given if an attempt is made to read past an uncleared EOF.

3.1.4 "List Exceed Data on File XXXXXXX," error number 89, will be given if the number of list items exceeds the number of items on the record.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM just be made prior to an RJ to ABNORML

4.2 CIO1

4.2.1 Calling Sequence and Returns

Entry B6 = return address
X1 = FET address
X2 = function code for CIO

4.3 GETBA

4.3.1 Calling Sequence and Returns

Entry B2 = FET address
X3 = file name

4.4 OPEN

4.4.1 Calling Sequence and Returns

Entry X1 = FET address
X2 = OPEN parameter

4.5 SIO.

4.5.1 Calling Sequence and Returns

Entry B1 = number of words to be moved

B7 = address of data item or beginning address of array

X1 = FET address

X6 = function code for CIO

Exit X4 = 1 if EOF was encountered

= 0 if EOR was encountered during the move operation

=<0 if the requisite number words were moved

4.6 SYSTEM

4.6.1 Calling Sequence and Returns

Entry X1 = error number

X2 = address of diagnostic message

4.7 INITL

4.7.1 Calling Sequence and Returns

An RJ to INITL is made with the following registers set:

Entry B2 = address of first word of FET of the file, or the complement of either the file name or logical tape number.

X6 = open parameter

X7 - read/write parameter

Exit B2 = address of first word of FET of the file, or unchaged if file not found.

X5 = code and ststus

X6 = CIO control word

B5 = 1

B6 = -1 is an uncleared EOF on a read request

B6 = -2 if an attempted read after write

5.0 Structure

5.1 No registers are saved.

5.2 INPUTB is called at least 2 times for any binary write. The initial call sets up trace back information and calls GETBA to obtain the address of the FET. If the file cannot be referenced an error exit is taken. The FET address is saved. If the file is not blocked binary INITL. is called to initialize the file. If the last operation was a read or an end-of-file was read, appropriate error terminations are taken. An initialization call is then made to SIO.

As many intermediate calls are made as are necessary to transfer the required data. Unless an end-of-file is read or the file is blocked the parameters with which INPUTB was called are passed in a call to SIO. The terminal call to INPUTB simply issues a terminal call to SIO. unless the file is blocked.

The first call to INPUTB on a blocked file opens the file if it is not yet opened. Reading past EOF is checked for, the first read is initiated, local calls are initialized and GETWDS is called to provide a number of sequential words.

An intermediate call to INPUTB on a blocked file calls GETWDS if necessary to provide a number of sequential words, sets up a call to MVWDS. based upon the contents of the control word to transfer the user's logical record, and calls GETWDS again if necessary.

The last call to INPUTB on a blocked file skips the unread portion of the user's logical it presents and updates the record count.

D8. INPUTC

1.0 General Information

The function of INPUTC is to transfer formatted input from a file unit to storage locations, as specified by a FORTRAN formatted READ statement. INPUTC actually functions as a linkage to KRAKER, which performs the actual transfer and cracks the information for internal storage. Although KRAKER is a part of the INPUTC deck, it functions as a separate entity, and will be described separately.

1.1 Approximate length (including KRAKER): 1130B

2.0 Entry Points

2.1 INPUTC

2.1.1 Calling Sequence and Returns

There are three calls to INPUTC. The first entry to initialize input, one intermediate entry for each array or data item, and a final entry to signal the end of the list and terminate input.

```
First entry:  B2 = address of FET or
                   complemented
                   address of variable tape
                   number.
              B3 = address of format statement
                   or complemented address of
                   variable format statement
```

INPUTC then calls KRAKER, see below, for KRAKER parameters.

```
Intermediate entries:  B1 = address of data
                            item or beginning
                            address of array
                       B2 = array length
                       RJ   INPUTC
```

INPUTC then calls KRAKER, see below, for KRAKER parameters.

```
Final entry:  B1 = -1
              RJ   INPUTC
```

INPUTC then calls KRAKER.

KRAKER linkage description:

```
First entry:   X3 = 0 not decode
               B1 = 0 initial entry
               B2 =   FWA of data array
                      (here = DAT.)
               B4 =   address of return jump
                      instruction with which
                      to read the next line.
               B6 =   line count
               B7 =   maximum number of
                      characters per line
               CALL KRAKER

Intermediate entries: B1 = FWA data
                      B2 = number of words
                      CALL KRAKER

Final entry:   B1   0
               CALL KRAKER
```

When INPUTC returns KRAKER the next line, it preserves registers: B2, B3, B6 and sets B7 = 1.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXXXX," error number 64, will be given if no file was found for a tape unit.

3.1.2 "BCD Input * ENDFILE *," error number 65, will be given if an attempt is made to read past an uncleared EOF.

3.1.3 "BCD Input *Last Op Write *", error number 88, will be given if an attempt is made to read after a write operation on the file.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM just be made prior to an RJ to ABNORML.

4.2 INITL

4.2.1 Calling Sequence and Returns

An RJ to INITL. is made with the following registers set:

Entry B2 = address of first word of FET of the file, or the complement of either the file name or logical tape number.
X6 = open parameter
X7 = read/write parameter

Exit B2 = address of first word of FET of the file, or unchanged if file not found.
X5 = code and status
X6 = CIO control word
B5 = 1
B6 = -1 if an uncleared EOF on a read request
B6 = -2 if an attempted read after write

4.3 GETBA

4.3.1 Calling Sequence and Returns

Entry B2 = complement of address of either file name or logical tape number

Exit B2 = FET address
X3 = file name

4.4 OPEN

4.4.1 Calling Sequence and Returns

This routine is entered by doing an RJ to OPEN. with X1 set to the address of the first word of the FET of the associated file and X2 set to the function code for the desired call to OPE.

Upon exit, the file will have been opened in the manner determined by the function code.

4.5 SIO.

4.5.1 Calling Sequence and Returns

This is the entry point for read/write processing. It is entered by doing an RJ to SIO. with X1 set to the address of the first word of the FET associated with the file, SIO.CTL control word with bit 2 =0 for a read request and =1 for a write reqeest, and B registers set according to the following criteria:

B7 =0 This is a formatted I/O request

For a formatted read request, the 150 character DAT. buffer will be utilized, with blank fill, replacing any zero bytes with blanks.

For a formatted write request B1 contains the number of characters to be written from the DAT. buffer, starting at DAT,, through DAT. + B1-1. Characters are expected in R1 format; i.e. right-adjusted, one character per word, with zero fill.

B7 ≠0 This is an unformatted I/O request

B1 = 0 INPUTB/OUTPUTB initialization
B1 0 unformatted read/write request, B1 contains the number of words to be transferred
B1 0 INPUTB/OUTPUTB termination

Upon exit data will have been transferred between the area defined and the buffer, operating system calls will have been made as required, and IN and OUT will have been updated. Also X4 will have been set as follows:

X4 = 0 EOR
X4 > 0 EOF
X4 < 0 else

4.6 SYSTEM

4.6.1 Calling Sequence and Returns

Entry X1 = error number
X2 = address of diagnostic message

5.0 Structure

D9. INPUTN

1.0 General Information

This routine is called to handle FORTRAN statements of the following form:

READ n
READ (u,n)
where n has been specified as NAMELIST.

INPUTN will read from the specified file converting data as directed, and place it is the memory locations specified by the NAMELIST group.

1.1 Approximate Length: 1226B

2.0 Entry Points

2.1 INPUTN

This is the only entry point to the routine and performs all NAMELIST input.

2.1.1 Calling Sequence and Returns

Upon entry, the following B registers are set:

B1 = fwa of the NAMELIST information
B2 = address of first word of FET of the file or complemented address of variable tape number.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Namelist name not found," error number 66, will be given if there is no Namelist name corresponding to the one requested.

3.1.2 "No I/O medium assigned," error number 66, will be given if no file was found for the tape unit.

3.1.3 "Wrong type constant," error numblr 66, will be given if the form of input data does not correspond in type to the variable specified.

3.1.4 "Incorrect subscript," error number 66, will be given if the subscript form does not correspond with that of the program variable.

3.1.5 "Too many constants," error number 66, will be given if the number of pieces of input data specified for an array is more than the number of items in the array.

3.1.6 "(,$,or = expected, missing," error number 66, will be given if a syntax error is encountered in NAMELIST input.

3.1.7 "Variable name not found," error number 66, will be given if the data contains a name not in the NAMELIST group list.

3.1.8 "Bad numeric constant," error number 66, will be given if NAMELIST input data entry is illegally formed.

3.1.9 "Missing constant after *," error number 66, will be given if NAMELIST input data is incompletely formed.

3.1.10 "Uncleared EOF on Read," error number 66, will be given if an attempt is made to read past an uncleared EOF.

3.1.11 "Attempted Read after Write," error number 66, will be given if an attempt is made to read after a write operation on the file.

3.2 Informative

3.2.1 "Precision lost in floating integer constant," error number 49, will be given if an attempt is made to read an integer constant with more than 48 bits of precision into a single precision floating point word.

3.2.2 "Namelist data terminated by EOF, not $," error number 49, will be given if an end of file indicator is found in Namelist input data before the terminating $.

3.2.3 "Too few constants for unsubscripted array," error number 49, will be given if the number of values listed for an array with no indexing is less than the size of an array.

4.2 INITL

4.2.1 Calling Sequence and Returns

An RJ to INITL. is made with the following registers set:

Entry B2 = address of first word of FET of the file, or the complement of either the file name or logical tape number.
X6 = open parameter
X7 = read/write parameter

Exit B2 = address of first word of FET of the file, or unchanged if file not found.
X5 = code and status
X6 = CIO control word
B5 = 1
B6 = -1 if an uncleared EOF on a read request
B6 = -2 if an attempted read after write

4.3 SIO.

4.3.1 Calling Sequence and Returns

This is the entry point for read/write processing. It is entered by doing an RJ to SIO. with X1 set to the address of the first word of the FET associated with the file, SIO.CTL control word with bit 2 =0 for a read request and =1 for a write request, and B registers set according to the following criteria:

B7 =0 This is a formatted I/O request

For a formatted read request, the 150 character DAT. buffer will be utilized, with blank fill, replacing any zero bytes with blanks.

For a formatted write request B1 contains the number of characters to be written from the DAT. buffer, starting at DAT., through DAT. +B1-1. Characters are expected in R1 format; i.e. right-adjusted, one character per word, with zero fill.

B7 ≠0 This is an unformatted I/O request

B1=0 INPUTB/OUTPUTB initialization
B1>0 unformatted read/write request, B1 contains the number of words to be transferred
B1<0 INPUTB/OUTPUTB termination

Upon exit data will have been transferred between the area defined and the buffer, operating system calls will have been made as required, and IN and OUT will have been updated. Also X4 will have been set as follows:

X4 = 0 EOR
X4 > Ø EOF
X4 > 0 else

4.4 SYSTEM

4.4.1 Calling Sequence and Returns

Entry X1 = error number
X2 = address of diagnostic message

5.0 Structure

5.1 INPUTN first saves registers, then calls INITL. to initialize the file. If the file is not found, or an attempt to read past an end of file is made, error exits are taken. A scan of the input file for a valid Namelist group name in the appropriate syntax is made. When a valid group name is encountered, valid variable names within the group are searched for, and syntax checked. Subscript evaluation takes place where necessary. Data constants are picked up, multiple groups recognized and checked for ize and data is converted according to type (integer, real, double precision, complex, and logical). Each record of validated information is placed in the DAT. buffer and a call is made to SIO. to transfer the information. Processing continues until a $, which terminates a Namelist group, or an end-of-file is encountered.

6.1 The NAMELIST information area:

Word 0: The NAMELIST name in display code, zero filled, left-adjusted in the lower 42 bits.

.
.
.

Word 3K-2: the name of the Kth associated variable, in display code, zero-filled, left-adjusted in the lower 42 bits.

Word 3K-1: 1 in bits 59-54 (indicates FWA_k is the address of a variable's first memory location)

Zero in bits 53-48

FWA_k in 47-30

T_k in 29-0 (all right justified within the allotted bits)

FWA_k is the address of the variables first memory location

T_k is the type of the variable:

1 = logical
2 = integer
4 = real
5 = double
6 = complex

Word 3K: Zero in 59-54 if not dimensioned

2 in 59-54 if dimensioned

$M2_k$ in 53-36

$M1_k$ in 35-18

LNG in 17-0

$M2_k$ is: the first dimension of a three-dimensional array.

0 otherwise.

LNG is: the number of elements (not necessarily the number of computer words) of an array.

1 for a variable

.
.
.

Word 3N+1: 0 (where N is the number of variables associated with the NAMELIST name.)

D10. INPUTS

1.0 General Information

The function of this routine is to transfer c consecutive BCD characters from starting address v, to the address(es) of the specified list variables L, according to the FORTRAN FORMAT specified by n, these parameters having been specified in the FORTRAN statement DECODE (c,n,v)L. This is a core to core transfer and therefore does not use peripheral equipment.

1.1 Length: 302_8

2.0 Entry Points

2.1 INPUTS

2.1.1 Calling Sequence and Returns

The routine is entered by doing a RJ to INPUTS. It is entered four or more times. The first two entries are initialization phases, an intermediate entry for each data item, and a final entry to signal the end of data. The register conditions of the entry points are as follows:

First Entry	B1 = 0,
	B2 = 0,
	B3 = the address of the format statement,
	B4 = the character length,

Second Entry	B1 = the beginning address of the picked data, B2 = 0,
Intermediate Entries	B1 = the address of data item or beginning address of array, B2 = the array length or zero,
Final Entry	B1 = -1.

Upon exit the character transfer will have been affected.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 *DECODE CHAR/RECORD .GT.150* error number 66, will be given if the number of characters to be transferred is greater than 150.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Return

A RJ to SYSTEM must be made prior to a RJ to ABNORML.

4.2 KRAKER

4.2.1 Calling Sequence and Return

Entered three or more times, the first is for initialization purposes, the intermediates for processing, and the final to signal the end of data. At entry times the following register conditions should exist:

First Entry	B1 = 0, B2 = the address of data character buffer, B3 = the address of the FORMAT statement, B4 = the address of data fetch sequence, B5 = the address of calling routine, B6 = 0, B7 = the number of characters,
Intermediate Entries	B1 = the address of data item or beginning address of array, B2 = the array length of zero,
Final Entry	B1 = -1.

4.3 SYSTEM

4.3.1 Calling Sequence and Return

Entry X1 = the error number,
X2 = the address of the diagnostic message.

5.0 Structure

5.1 Initialization

5.1.1 (FIRST) Initialization phase I. Stores traceback information and checks to see that the char/record count does not exceed 150. If it does, a jump to ERRS occurs. If not, the second entry flag is set to zero, the address of the format statement is saved, and the number of characters in the record is saved.

5.1.2 (SECND) Initialization phase II. Saves the beginning address of the packed data and sets the registers for the initial entry into KRAKER, then calls KRAKER.

5.1.3 (ERRS) Calls SYSTEM (with an error number of 66 and a message of *DECODE CHAR/RECORD .GT. 150*) to process the error and then calls ABNORML to abort the job since the error is fatal to execution.

5.2 RJDAT. Used by KRAKER to get a data line.

5.2.1 (RDNX) Fetches a data word, sets up a character counter, and sets up an address pointer equal to the first word address of the data character buffer.

5.2.2 (RDA) Sets up a counter which indicates the number of characters remaining in the word being processed, and a lower six bit mask register.

5.2.3 (RDB) Left shifts the data word six bits then picks up the lower six bits; if the result is zero it creates a blank character.

5.2.4 (NONE) Stores the character into the data buffer then decrements the character counter and the char/word indicator by one and increments the address pointer by one. Checks to see if all characters have been stored and if so executes a jump to RDC. Checks to see whether or not all the characters in a word have been processed and if not executes a jump to RDB. If true, it increments the data word address by one then jumps to RDA to fetch a new word.

5.2.5 (RDC) Saves data word pointer, sets a pointer to the address of the last word of the character buffer.

5.2.6 (RDD) Checks to see if block is full and if not stores a blank character and increments the address pointer by one. If full, a jump to RDE is executed.

5.2.7 (RDE) Checks to see if block is full and if not executes a jump to RDD. If full, it sets a pointer to the first word address of the data character buffer and exits.

D11. IOCHEC

1.0 General Information

This routine is called in response to the statement IF(IOCHECK,i)... It sets the return true (0) and exits. (There is no parity checking.)

1.1 Length: 3_8

2.0 Entry Points

2.1 IOCHEC

2.1.1 Calling Sequence and Returns

The routine exits after setting X6 to zero.

3.0 Diagnostics: none.

4.0 External routines: none.

5.0 Structure

5.1 X6 is set to zero and the routine exits.

D12. IOCHEK

1.0 General Information

The function of this routine is to check the status of a buffered operation on logical unit i, to restore the buffer parameters affected by that operation, and if the last operation was a BUFFER IN, to ensure that the data has actually been transferred in response to the FORTRAN statement IF(UNIT,i) m_1,m_2,m_3. Control is transferred to m_1 if the unit is busy. Upon completion of the routine, control is transferred to m_2 if the last operation was a BUFFER OUT or if the BUFFER IN terminated normally, or if the last operation was a BUFFER IN and an EOR was read. Control is transferred to m_3 if the last operation was a BUFFER IN and an EOF was read. If i is a non-buffered unit, no buffer parameters are changed and control is transferred to m_2.

1.1 Approximate length: 113B

2.0 Entry Points

2.1 IOCHEK

2.1.1 Calling Sequence and Returns

This routine is called in response to a FORTRAN IF (unit,i) statement. IOCHEK is entered once for each FORTRAN statement. All B-registers are saved.

Entry conditions

X6 contains (1) the FET address or (2) the complement of the address of the logical file name or tape number

Exit conditions

X6 = 1	if a read operation detected an end of file
X6 = 0	for all other conditions

Upon exit the pending buffer operation will have been completed and the FET entries will have been reset.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXXXX," error number 67, will be given if no file was found for the tape unit.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM must be made prior to an RJ to ABNORML

4.2 CIO1

4.2.1 Calling Sequence and Returns

Entry B6 = return address
X1 = FET address
X2 = function code for CIO

4.3 GETBA

4.3.1 Calling Sequence and Returns

Entry B2 = complement of address of either file name or logical tape number

Exit B2 = FET address
X3 = file name

4.4 SIO.

4.4.1 Calling Sequence and Returns

Entry B1 - number of words to be moved
B7 - address of data item or beginning address of array.
X1 = FET address
X6 = function code for CIO

Exit X3 = 0 if EOR was encountered during the move operation
X3 ≠ 0 otherwise

SYSTEM

4.5.1 Calling Sequence and Returns

Entry X1 = error number
X2 = address of diagnostic message

5.0 Structure

5.1 If the file being referenced is not a BUFFER file, no action is taken by IOCHEK. If necessary, GETBA is called to obtain the address of the FET. If the file cannot be referenced, an error exit is taken. If the file is busy, normal recall is initiated. If the previous operation on the file was a BUFFER OUT, the FET pointers are reset (FIRST=IN=OUT), the B registers are restored, and the routine exits. If the previous operation was a BUFFER IN, SIO. is called to complete any pending information transfer on an I or X tape file. On an S or L tape, recall is issued until the operation in progress is completed. The length of the block of information transferred is then placed in word 14 of the FET, the B registers are restored, and the routine exits.

D13. LENGTH

1.0 General Information

The function of this routine is to return the number of words read on logical unit i by the last BUFFER IN operation in response to the FORTRAN function LENGTH(i).

1.1 Length: 21_8

2.0 Entry Points

2.1 LENGTH

2.1.1 Calling Sequence and Returns

This routine is entered by doing a RJ to LENGTH. B1 should contain the address of either the file name or the logical tape number associated with the tape unit. Upon exit, X6 will contain the number of words read on the file by the last BUFFER IN operation.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXXXX", error number 81, will be given if no file was found for the tape unit.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

A RJ to SYSTEM must be made prior to a RJ to ABNORML.

4.2 GETBA

4.2.1 Calling Sequence and Returns

```
Entry     B2 = complement of address of either file
               name or logical tape number

Exit      B2 = FET address
          X3 = file name
```

4.3 SYSTEM

4.3.1 Calling Sequence and Returns

```
Entry     X1 = error number
          X2 = address of diagnostic message
```

5.0 Structure

5.1 B2, the only B register used, is saved.

5.2 The entrance parameter is complemented and the FET address of the file for unit i is obtained by calling GETBA. If GETBA found the file in the list of FORTRAN files, a branch is taken to LEA.

5.3 The file name is stored into the error message. SYSTEM (with an error number of 81 and an error message of "Unassigned medium, file XXXXXXX") is called to process the error and ABNORML is called to abort the job because the error is fatal.

5.4 (LEA) X6 is set equal to the number of words read on unit i by the last BUFFER IN operation.

5.5 B2 is restored and a branch is taken to exit.

D14. OUTPTB

1.0 General Information

The function of this routine is to transfer one logical record of binary information from storage location (L) to a file unit (i) as specified by one of the following FORTRAN statements:

```
WRITE (i)L
WRITE TAPE i,L
```

If L is omitted, the routine writes a zero length logical record onto the file unit (i).

1.1 Approximate length: 211B

2.0 Entry Points

2.1 OUTPTB

2.1.1 Calling Sequence and Returns

This routine is entered by doing an RJ to OUTPTB. It is entered three or more times; a first entry to initialize output, one intermediate entry for each array or data item, and a final entry to signal the list and terminate output. The register conditions for the entries should be:

First entry	B1 = 0 B2 = the address of the buffer parameter list or the complement of the address of the file name or logical tape number
Intermediate	B1 = the address of the data item or the beginning address of the array B2 = the array length or 0
Final Entry	B1 = -1

Upon exit one binary logical record will have been written.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXXXX," error number 82, will be given if no file was found for the tape unit.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM must be made prior to an RJ to ABNORML

4.2 CIO1

4.2.1 Calling Sequence and Returns

Entry B6 - return address
XI = FET address
X2 = function code for CIO

4.3 GETBA

4.3.1 Calling Sequence and Returns

Entry B2 = complement of address of either file name or logical tape number

Exit B2 = FET address
X3 = file name

4.4 OPEN

4.4.1 Calling Sequence and Returns

Entry X1 = FET address
X2 = OPEN parameter

4.5 SIO.

4.5.1 Calling Sequence and Returns

Entry B1 = number of words to be transferred
B7 = address of data item or beginning address of array
X1 = FET address
X6 - function code for CIO

4.6 SYSTEM

4.6.1 Calling Sequence and Returns

Entry X1 = error number
X2 = address of diagnostic message

4.7 INITL

4.7.1 Calling Sequence and Returns

An RJ to INITL is made with the wollowing registers set:

Entry B2 = address of first word of FET of the file, or the complement of either the file name or logical tape number
X6 = open parameter
X7 = read/write parameter

Exit B2 = address of first word of FET of the file, or unchanged if file not found.
X5 = code and status
X6 = CIO control word
B5 = 1
B6 = -1 if an uncleared EOF on a read request
B6 = -2 if an attempted read after write

5.0 Structure

5.1 At least two entries are made to OUTPTB in response to a binary write. The initial entry calls GETBA if necessary to obtain the address of the FET. If the file cannot be referenced an error exit is taken. If the file is not blocked INITL is called to initialize the file, and an initialization call is made to SIO.

As many intermediate entries are made as are necessary to transfer data. The parameters with which OUTPTB was called are passed in a call to SIO unless the file is blocked.

The terminal entry makes a terminal call to SIO unless the file is blocked.

The first call to OUTPB on a blocked file opens the file if it is not yet opened, positions the file if the last operation was read, and initializes local cells.

An intermediate call to OUTPB on a blocked file calls GETWDS if necessary to provide a number of sequential words in the buffer, sets up a call to MVWDS. to transfer the binary blocking control word and the user's logical record, and calls GETWDS again if necessary.

The last call to OUTPTB on a blocked file calls GETWDS if necessary, prepares the next binary blocking control word, and updates the record count.

D15. OUTPTC

1.0 General Information

This routine serves as the interface between the FORTRAN formatted output statements and the SCOPE system input/ output processing. In conjunction with the routine KODER (which converts information to the format specification), data items may be written onto a file in lines of MAXCHAR length. Although KODER is a part of the OUTPTC deck, it functions as a separate entity, and will be described separately.

1.1 Approximate length (including KODER): 1274B

2.0 Entry Points

2.1 OUTPTC

2.1.1 Calling Sequence and Returns

The procedure for utilizing OUTPTC involves an initialization call (accomplished by a return jump to the entry point OUTPTC with parameters designating the file to be referenced and the format to be used), intermediate calls to convert an item or a sequence of items, and a final call to terminate the processing of the last line.

Calling Procedure:

Initialization Call

```
        RJ OUTPTC
```

Where Register B2 has been set to a value ALPHA. If ALPHA>0 then ALPHA is the address of the first word of the file environment table of the file to be referenced.
If ALPHA<0 then the contents of the word at location -ALPHA contains the file designation, which may be in either of two forms:

1. VFD 60/U $1 \leq U \leq 99$
2. VFD 42/FNAME,18/0 Where FNAME is the display code file-name to be referenced.

Register B3 has been set to a value BETA. If BETA>0 then it is the address of the format to be used (the word at BETA will contain the display code of the format number in the source code and the format strong begins at location BETA+1. If BETA<0 then it is the complement of the first word of a variable format strong.

Intermediate Calls

RJ OUTPTC

Where Register B1 has been set to the address of a sequence of items (possibly of length 1) to be converted.

Register B2 contains the number of words which comprise the items to be converted (the items may be single or double word entities).

Final Call

RJ OUTPTC

Where Register B1 has been set to a negative quantity.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXX," error number 83, will be given if no file was found for a tape unit.

3.1.2 "Output file line limit exceeded", error number 84, will be given if the line limit, as specified on the RUN card is exceeded.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM just be made prior to an RJ to ABNORML

4.2 INITL

4.2.1 Calling Sequence and Returns

An RJ to INITL is made with the following registers set:

Entry B2 = address of first word of FET of the file, or the complement of either the file name or logical tape number.

X6 = open parameter
X7 = read/write parameter

Exit B2 = address of first word of FET of file, or unchanged if file not found.
X5 = code and status
X6 = CIO control word
B5 = 1
B6 = -1 if an uncleared EOF on a read request.
B6 = -2 if an attempted read after write

4.3 SIO.

4.3.1 Calling Sequence and Returns

This is the entry point for read/write processing. It is entered by doing an RJ to SIO. with X1 set to the address of the first word of the FET associated with the file, SIO.CTL control word with bit 2 =0 for a read request and =1 for a write request, and B registers set according to the following criteria:

B7 =0 This is a formatted I/O request

For a formatted read request, the 150 character DAT. buffer will be utilized, with blank fill, replacing any zero bytes with blanks.

For a formatted write request, B1 contains the number of characters to be written from the DAT. buffer, starting at DAT., through DAT. +B1-1. Characters are expected in R1 format; i.e. right-adjusted, one character per word, with zero fill.

B7 ≠0 This is an unformatted I/O request

B1=0 INPUTB/OUTPUTB initialization
B1 0 unformatted read/write request, B1 contains the number of words to be transferred
B1 0 INPUTB/OUTPUTB termination

Upon exit data will have been transferred between the area defined and the buffer, operating system calls will have been made as required, and IN and OUT will have been updated. Also X4 will have been set as follows:

X4 = 0 EOR
X4 0 EOF
X4 0 else

4.4 SYSTEM

4.4.1 Calling Sequence and Returns

Entry X1 = error number
X2 = address of diagnostic message

D16. OUTPTN

1.0 General Information

This routine is called to handle FORTRAN statements of the form:

PRINT n
WRITE (u,n)

where n has been specified as NAMELIST.

OUTPTN will convert the data associated with the NAMELIST group according to the mode of the variables in the group. The information will be placed on the specified file.

1.1 Length: 452B

2.0 Entry Points

2.1 OUTPTN. This is the only entry point to the routine and performs all NAMELIST output.

2.1.1 Calling Sequence and Returns

Upon entry the following B registers are set:

B1 - fwa of the NAMELIST group area
B2 - address of the first word of FET of the file or complemented address of variable tape number.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXXXX," error number 64, will be given if no file was found for the tape unit.

3.1.2 "Output file line limit exceeded," error number 84, will be given if the line limit, as specified on the RUN card, is exceeded.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM must be made prior to an to ABNORML

4.2 INITL

4.2.1 Calling Sequence and Returns

An RJ to INITL is made with the following registers set:

Entry	B2 =	address of first word of FET of the file, or the complement of either the file name or logical tape number,
	X6 =	open parameter
	X7 =	read/write parameter
Exit	B2 =	address of first word of FET of the file, or unchanged if file not found.
	X5 =	code and status
	X6 =	CIO control word
	B5 =	1
	B6 =	-1 if an uncleared EOF on a read request
	B6 =	-2 if an attempted read after write

4.3 SIO.

4.3.1 Calling Sequence and Returns

This is the entry point for read/write processing. It is entered by doing an RJ to SIO. with X1 set to the address of the first word of the FET associated with the file, SIO.CTL control word with bit 2 =0 for a read request and =1 for a write request, and B registers set according to the following criteria:

B7 =0 This is a formatted I/O request

For a formatted read request, the 150 character DAT. buffer will be utilized, with blank fill, replacing any zero bytes with blanks.

For a formatted write request, BI contains the number of characters to be written from the DAT, buffer, starting at DAT., through DAT. = B1-1. Characters are expected in R1 format; i.e. right-adjusted, one character per word, with zero fill.

B7 ≠0 This is an unformatted I/O request

B1=0 INPUTB/OUTPUTB initialization
B1 0 unformatted read/write request, B1 contains the number of words to be transferred
B1 0 INPUTB/OUTPTB termination

Upon exit data will have been transferred between the area defined and the buffer, operating system calls will have been made as required, and IN and OUT will have been updated. Also X4 will have been set as follows:

X4 = 0 EOR
X4 0 EOF
X4 0 else

4.4 SYSTEM

4.4.1 Calling Sequence and Returns

Entry X1 = error
X2 = address of diagnostic message

5.0 Structure

5.1 OUTPTN first save registers, then calls INITL. to initialize the file. If the file is not found, error exit is taken. OUTPTN then forms the first line of Namelist output, i.e., $ "Namelist-group-name", and calls SlO. to transfer the line to the output buffer. The Namelist information area, as described in Section 6 of INPUTN, is then utilized in printing out the variables and their values which are associated with the Namelist group. Information is stored in the DAT. buffer and again SIO. is called to transfer the information, a record at a time, to the output buffer. Processing continues until the zeroword at the end of the Namelist information area is encountered. At that point, "END" is sent to the output buffer via SIO. and control is returned to the caller via the entry point.

D17. OUTPUTS

1.0 General Information

The function of this routine is to transfer the information in the list variables, L, according to the FORTRAN FORMAT specified by n, into the locations(s) starting at v, c BCD characters per record. These parameters having been specified in the FORTRAN statement ENCODE (c,n,v)L. This is a core to core transfer and therefore does not use peripheral equipment.

1.1 Length: 305_8

2.0 Entry Points

2.1 OUTPUTS

2.1.1 Calling Sequence and Returns

The routine is entered by doing a RJ to OUTPUTS. It is entered four or more times. The first two entries are initialization phases, an intermediate entry for each data item, and a final entry to signal the end of data. The register conditions at the entry points are as follows:

First Entry	B1 = 0,
	B2 = 0,
	B3 = the address of the format statement,
	B4 = the character length,

Second Entry	B1 = the beginning address of the packed data, B2 = 0,
Intermediate Entries	B1 = the address of data item or beginning address of array, B2 = the array length or zero,
Final Entry	B1 = -1.

Upon exit the character transfer will have been affected.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 ENCODE * CHAR/REC .GT. 150* error number 85, will be given if the number of characters to be transferred is greater than 150.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Return

A RJ to SYSTEM must be made prior to a RJ to ABNORML.

4.2 KODER

4.2.1 Calling Sequence and Return

Entered three or more times, the first is for initialization purposes, the intermediates for processing, and the final to signal the end of data. At entry times the following register conditions should exist:

First Entry	B1 = 0, B2 = the address of data character buffer B3 = the address of the FORMAT statement, B4 = the address of data fetch sequence, B5 = the address of calling routine, B6 = 0, B7 = the number of characters,
Intermediate Entries	B1 = the address of data item or beginning address of array, B2 = the array length or zero,
Final Entry	B1 = -1.

4.3 SYSTEM

4.3.1 Calling Sequence and Return

Entry	X1 = the error number X2 = the address of the diagnostic message

5.0 Structure

5.1 Initialization

5.1.1 (FIRST) Initialization phase I. Stores traceback information and checks to see that the char/record count does not exceed 150. If it does, a jump to ERRS occurs. If not, the second entry flag is set to zero, the address of the format statement is saved, and the number of characters in the record is saved.

5.1.2 (SECND) Initialization phase II. Saves the beginning address of the packed data and sets the registers for the initial entry into KODER, then calls KODER.

5.1.3 (ERRS) Calls SYSTEM (with an error number of 85 and a message of *ENCODE CHAR/REC .GT. 150*) to process the error and then calls ABNORML to abort the job since the error is fatal to execution.

5.2 RJDAT. Used by KODER to store a data line.

5.2.1 (WTNX) Sets up a character pointer, a CHAR/RECORD counter, and a WORD pointer.

5.2.2 (WTA) Clears a word to zeros and sets up the CHAR/WORD counter.

5.2.3 (WTB) Fetches a character and checks to see whether or not it is zero. If it is, a jump to WTC is executed; otherwise, the CHAR/RECORD and the CHAR/WORD counters are decremented by one, and the character stored in the data word. Checks to see if the CHAR/RECORD counter is less than one. If it is, executes a jump to WTC; otherwise, checks to see whether or not CHAR/WORD counter is zero. If it is not, jumps back to WTB; otherwise, it stores WORD and increments the WORD pointer by one and jumps back to WTA.

5.2.4 (WTC) Asks whether or not CHAR/WORD counter is zero, and if it is executes a jump to WTE, otherwise a blank character is set.

5.2.5 (WTD) Decrements the CHAR/RECORD and CHAR/WORD counters by one and stores the blank character.

5.2.6 (WTE) Asks if CHAR/WORD counter is not equal to zero, and if so executes a jump to WTD, otherwise it sets up a word of blanks.

5.2.7 (WTF) Stores a packed word and increments the WORD pointer by one, and then asks if the CHAR/RECORD counter is less than one. If true, a jump to WTG is executed, otherwise the CHAR/RECORD counter is decremented by one and a jump to WTF is executed.

5.2.8 (WTG) Saves word address and sets the character buffer address, then exits.

D18. REWINM

1.0 General Information

The function of this routine is to rewind to load point tape unit i in response to the FORTRAN statement REWIND i.

1.1 Ength: 52_8

2.0 Entry Points

2.1 REWINM

2.1.1 Calling Sequence and Returns

The routine is entered by doing an RJ to REWINM, X6 should contain the complement of the address either the file name or the logical tape number associated with the tape unit; or X6 should contain the address of the first word of the FET associated with the file. Upon exit the tape will have been rewound to load point.

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Unassigned medium, file XXXXXXX," error number 86, will be given if no file was found for the tape unit.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

An RJ to SYSTEM must be made prior to an RJ to ABNORML

4.2 CIO1.

4.2.1 Calling Sequence and Returns

```
Entry   B6 - return address
        X1 = FET address
        X2 = function code for CIO
```

4.3 GETBA

4.3.1 Calling Sequence and Returns

Entry B2 = complement of address of either the file name, or logical tape number.

Exit B2 = address of first word of FET of the file.

X8 = file name

4.4 INITL.

4.4.1 Calling Sequence and Returns

An RJ to INITL. is made with the following registers set:

Entry B2 = address of first word of FET of file, or the complement of either the file name or logical tape number.

X6 = open parameter

X7 = read/write parameter

Exit B2 = address of first word of FET of the file, or unchanged if file not found.

X5 = code and status

X6 = CIO control word

B5 = 1

B6 = -1 if an uncleared EOF on a read request

B6 = -2 if an attempted read after write

4.5 ADVIN

4.5.1 Calling Sequence and Returns

An RJ to ADVIN is made with the following register set:

X1 = address of first word of FET of the file

ADVIN advances the IN pointer by 1.

4.6 SYSTEM

4.6.1 Calling Sequence and Returns

Entry X1 = error number
X2 = address of diagnostic message

5.0 Structure

5.1 Registers B1, B2, B6 and B7 are saved. If necessary, GETBA is called to obtain the address of the FET. If the file cannot be referenced, an error exit is taken. If the file has not been opened, the B registers are restored and the routine exits. INITL is called to determine the status of the file. If the file is blocked binary and the last operation was a write, ADVIN. is called to write an end-of-file if necessary. The end-of-file flag is then cleared and CIO1. is called to rewind the file. The B registers are restored and the routine exits.

D19. XRCL

1.0 General Information

XRCL is called to enter the program into recall status.

1.1 Length: 5

2.0 Entry Points

2.1 XRCL

2.1.1 Calling Sequence and Returns: none

3.0 Diagnostics Produced: none

4.0 External Routines: none

5.0 Structure

5.1 Wait until RA+1 is clear. Put "RCL" in the upper 18 bits of RA+1. Wait until RA+1 is clear, then EXIT.

D20 KRAKER

1.0 General Information

The function of this routine is to perform formatted conversion of data from display code form into machine internal form in response to calls from INPUTC or INPUTS.

1.1 Approximate length: (see section D8 on INPUTC)

2.0 Entry Points

2.1 KRAKER

2.1.1 Calling Sequence and Returns

The procedure for utilizing KRAKER involves an initialization call providing various state setting information (location of the format to use, length of the input line, etc.), intermediate calls to perform the conversion of the data items, and a terminal call to conclude processing of the last line.

Calling Procedure:

Initialization Call

```
     RJ KRAKER
```

where the following registers have been set

B1 -- 0
B2 -- the address of the first word of the data buffer which will hold an input line, burst into one character per word, right-justified with binary zero fill
B3 -- address of the format id for a compiled format or the address minus one of the first word of a variable format. (For a compiled format, the format id is a word containing the format number in display code, left-justified with binary zero fill; the format proper follows the format id word.)
B4 -- the address of the word containing a RJ instruction to the routine which will "read" the next line into the data buffer

B5 -- the maximum length of the input line (in characters)
B6 -- the address of the word which will contain the record count for the file being read from
X3 -- 0 if KRAKER is to handle calls resulting from a READ statement;
≠0 if KRAKER is to handle calls resulting from a DECODE statement

Intermediate Calls

RJ KRAKER

where the following registers have been set

B1 -- the address of the first word of the items in which to place the converted data
B2 -- the number of words which comprise the items. (If the contents of B2 are zero, then one item will be converted.)

Final Call

RJ KRAKER

where B1 has been set to a negative value

3.0 Diagnostics

3.1 Fatal to Execution

3.1.1 "Illegal functional letter", error number 74, will be issued if an unrecognizable format specification is encountered.

3.1.2 "Paren group not closed", error number 75, will be issued if the end-of-format indication (a 00_8 character) is encountered before the closing right parenthesis of the format.

3.1.3 "Field width zero", error number 76, will be issued if a format specification which specifies a field width of zero is encountered

3.1.4 "Exceeded record size", error number 77, will be issued if an attempt to read beyond the length specified in the initialization entry to KRAKER is encountered.

3.1.5 "Illegal data in field", error number 78, will be issued if a non-digit character appears out of place in a numeric field.

3.1.6 "Data overflow", error number 79, will be issued if the data to be converted has a value which is too large to be expressed internally in the machine.

3.1.7 "Hollerith format with list", error number 80, will be issued if no format conversion specification is encountered in a format and there is a request for an item to be converted.

4.0 External Routines

4.1 ABNORML

4.1.1 Calling Sequence and Returns

A RJ to SYSTEM must be made prior to a RJ to ABNORML (which will abort the job)

4.2 SYSTEM

4.2.1 Calling Sequence and Returns

```
Entry     X1 = error number
          X2 = address of diagnostic message
```

5.0 Structure

5.1 The main cycle of conversion for each format specification consists of determining the appropriate processor for the format specification and jumping to the location for the processor. The processor selected then converts the data, increments the format and data pointers and then returns to a common point where the converted value is stored and, if the intermediate entry was a "short-list" call, the cycle is entered again. If only a single item was requested then the routine returns to the caller.

5.2 Format Specification Processors

5.2.1 (RPARN) Right Parenthesis Processor

If the repeat count is not exhausted for the group then the format pointer is reset to the beginning of the group and the scanning cycle is re-entered (unless this is a DECODE call and this is the terminal call, in which case an exit is made from KRAKER). If the end of the format has been reached then the format pointer is reset (the manner of resetting depends on whether the USASI mode of format scanning has been selected or not). If no item conversion specification has been encountered and there is a request for an item outstanding then a diagnostic message is issued and the job aborted.

5.2.2 (LPARN) Left Parenthesis Processor

Unless too many levels of parentheses are encountered (three are allowed in USASI formats, two otherwise) this processor serves to push-down the level and information associated with group repeating.

5.2.3 (SLASH) Slash Processor

The routine RDNX is called to read the next line (or lines) to satisfy the "n/" format specification.

5.2.4 (XCODE) X-specification Processor

The data pointer is incremented by the count of the "X" specification and the scanning cycle re-entered.

5.2.5 (HCODE) H-specification Processor

If there is no overflow of data requested then the next "n" characters are transferred from the data string to the format to appear after the "nH".

5.2.6 (STAR) Asterisk-specification Processor

Data is transferred from the data string to the format until the matching asterisk is encountered in the format (anu asterisks encountered in the data stream will be transmitted as blanks).

5.2.7 (PCODE,PLUS,MINUS) Scale Factor Processor

The value of the scale factor designated is set into the location SCA.

5.2.8 (ICODE,FCODE,ECODE,DCODE) D, E, F, G, and I Specification Processors

These processors (using several common segments of code) convert decimal numeric data into internal form. The floating point processor portion uses triple precision multiplication in the scaling of the numbers which it then rounds to either double or single precision for storing into the data item.

5.2.9 (ACODE,RCODE) A- and R-specification Processors

These processors transfer display code data from the data string into either left-justified, blank fill form (A) or right-justified, binary zero fill form (R) for storing.

5.2.10 (OCODE) O-specification Processor

This processor converts octal numeric data from display code form into internal form.

5.2.11 (LCODE) L-specification processor

The value of the logical specification is set to false (-0) as an initial condition and if the first non-blank character in the defined field is a "T" then the value is set to true (+0).

5.3 Local Routines

5.3.1 (XOV)

This routine checks for the record length requested being larger than the specified length in the initial call.

5.3.2 (LDNX)

This routine loads and bursts the next word of the format.

5.3.3 (FWD)

This routine converts a string of format characters from digits into binary form until a non-blank, non-digit character is encountered. (Blanks are ignored.)

5.3.4 (RDNX)

This routine reads the next line from the data stream.

III. Utility Routines

A. Description of Each Routine

Al. ALGOER

1.0 General Information

ALGOER is called when an error is detected in an assigned or computed GO TO. It gives a diagnostic, traceback, and aborts the job.

1.1 Length: 12_8

2.0 Entry Points

2.1 ALGOER

2.1.1 Calling Sequence and Returns: none

3.0 Diagnostics

3.1 Informative; none

3.2 Fatal

3.2.1 ERROR, COMPUTED OR ASSIGNED GO TO UST

4.0 External Routines

4.1 SYSTEM

4.1.1 Calling Sequence and Returns

```
Entry     X1 = error number
          X2 = address of diagnostic message
```

4.2 ABNORML

4.2.1 Calling Sequence and Returns

Must be preceeded by a RJ to SYSTEM.

A2. DISPLA

1.0 General Information

The function of this routine is to display a variable name according to the Hollerith specifications H and a numerical value k in the dayfile in response to the FORTRAN statement CALL DISPLA(h,k). The value k is displayed as an integer if it is not normalized and in floating point format if it is normalized. If k is infinite or out of range, it will be displayed as a message to that effect.

1.1 Length: 247_8

2.0 Entry Points

2.1 DISPLA

2.1.1 Calling Sequence and Returns

This routine is entered by doing a RJ to DISPLA. B1 should contain the beginning address of the array containing the Hollerith data and B2 should contain the address of the numerical data. Upon exit, the Hollerith message and its numerical value will have been displayed on the dayfile.

3.0 Diagnostics

3.1 No diagnostics are given.

4.0 External Routines

4.1 No external routines are used.

5.0 Structure

5.1 No registers are saved.

5.2 The numerical data is fetched.

5.3 (DLAY1) If a request cannot be issued at this time, a branch is taken to DLAY1.

5.4 (DBA) The next word of the Hollerith message is fetched and stored for a MSG request.

5.4.1 If there are more words in the Hollerith message, a branch is taken to DBA.

5.4.2 If the numerical value is positive, a branch is taken to DIA. Otherwise, the numerical value is complemented.

5.5 (DIA) The sign of the numerical value is stored for the MSG request (a positive value is given a blank rather than a plus sign).

5.5.1 If the numerical value is zero, a branch is taken to INT.

5.6 (DIB) If the numerical value is infinite, a branch is taken to ERR with the message INFINITE.

5.6.1 If the numerical value is out of range, a branch is taken to ERR with the message RANGE. Otherwise, the numerical value is normalized.

5.6.2 If the numerical value was already normalized, a branch is taken to FPT.

5.6.3 If the numerical value is not a floating point zero, a branch is taken to INT.

5.7 (FPT) The numerical value is converted to a floating point number and stored one character per word in the array DAT.

5.8 (PCK) The pointer to the array DAT is initialized.

5.8.1 (DPG) The next character in the number is packed into a word.

5.8.2 (DPJ) If there is a character in the high order position of the word, a branch is taken to DPH. Otherwise, the characters are shifted left one place.

5.8.3 If there are no more characters to be packed into the word, a branch is taken to DPJ. Otherwise, a branch is taken to DPG.

5.8.4 (DPH) The packed word is stored for the MSG request.

5.8.5 If there are no more characters to be packed into words, a branch is taken to DPI. Otherwise, a branch to DPG is taken.

5.9 (DPI) The MSG request is set up.

5.10 (WRT) If the MSG request cannot be issued, a branch is taken to WRT.

5.10.1 The MSG request is issued.

5.11 (WRU) If the MSG request has not been honored, a branch is taken to WRU.

5.11.1 A RCL request is issued and a branch is taken to exit.

5.12 (INT) The numerical value is converted to an integer and stored one character per word in the array DAT.

5.13 (ERR) The error message is stored for the MSG request and a branch is taken to DPI.

A3. DUMP

1.0 General Information

The function of this routine is to dump from one to twenty areas of storage according to a specified format on the OUTPUT file in response to one of the following FORTRAN statements:

CALL DUMP($a_1,b_1,f_1,\ldots,a_n,b_n,f_n$)

CALL PDUMP($a_1,b_1,f_1,\ldots,a_n,b_n,f_n$)

where the a_i's and b_i's are respectively the first and last words of the storage area to be dumped and the f_i's are the format specifications. If no parameters are provided, an octal dump of all storage occurs. If PDUMP was called, control is returned to the calling program. If DUMP was called, the calling program is terminated and control is returned to the monitor.

1.1 Length: 213_8

2.0 Entry Points

2.1 DUMP

2.1.1 Calling Sequence and Returns

The routine is entered by doing a RJ to DUMP. It may be entered with as many as sixty parameters. The order of the parameters should be $a_1,b_1,f_1,\ldots,a_n,b_n,f_n$, where $n \leq 20$. The first six parameters should be contained in order in registers B1-B6. The remaining parameters should be stored in order beginning at location ST+6, where ST is the address of the beginning of the parameter region for the routine as described in the FORTRAN reference manual Appendix H. B7 should contain the number of parameters. Upon exit, a dump of the designated storage areas according to their specified formats will have been performed on the OUTPUT file. The calling program will have been terminated and control returned to the monitor.

2.2 PDUMP

2.2.1 Calling Sequence and Returns

The routine is entered by doing a RJ to PDUMP. It may be entered with as many as sixty parameters. The order of the parameters should be $a_1,b_1,f_1,\ldots,a_n,b_n,f_n$, where n is less than or equal to 20. The first six parameters should be contained in order in registers B1-B6. The remaining parameters should be stored in order beginning at location ST+6, where ST is the address of the beginning of the parameter region for the routine as described in the FORTRAN reference manual Appendix H. B7 should contain the number of parameters. Upon exit, a dump of the designated storage areas according to their specified formats will have been performed on the OUTPUT file.

3.0 Diagnostics

3.1 No diagnostics are given.

4.0 External Routines

4.1 OUTPTC

4.1.1 Calling Sequence and Returns

First Entry
B1 = 0
B2 = the address of the buffer parameter list or the complemented address of the variable tape number
B2 = the address of the format statement

Intermediate Entries
B1 = the address of the data item or the beginning address of the array
B2 = the array length or zero

Final Entry
B1 = -1

4.2 STOP

4.2.1 Calling Sequence and Returns

Upon exit from this routine, the calling program will have been terminated and control is returned to the monitor.

5.0 Structure

5.1 No registers are saved.

5.2 (PDUMP) FLG is set to zero to indicate that the routine was entered at PDUMP and then a branch is taken to PDUMP1.

5.3 (DUMP) FLG is set to one to indicate that the routine was entered at DUMP. A RJ is made to DMP to perform the dump and then a RJ is made to STOP to terminate execution of the calling program.

5.4 (PDUMP1) A RJ is made to DMP to perform the dump and then a branch is taken to exit.

5.5 (DMP) If a dump of all storage is indicated, a branch is taken to DMA. Otherwise, the parameters contained in B1 through B6 are stored in the parameter list.

5.5.1 The end of the parameter list is calculated.

5.6 (DML) If there are no more storage areas left to be dumped, a branch is taken to exit. Otherwise, the start of the parameter list is updated and the first and last word address of the area to be dumped are stored for a call to LST.

5.6.1 If the last word address of the area to be dumped is greater than the first word address of that area, a branch is taken to DMG. Otherwise, the two addresses are reversed and stored.

5.7 (DMG) The dump is given an E format specification. If the dump parameter indicated E format, a branch is taken to DUF.

5.7.1 The dump is given an I format specification. If the dump parameter indicated I format, a branch is taken to DUF. Otherwise, the dump is given an O format specification.

5.8 (DUF) The format specification is stored for a call to OUTPTC. A RJ is made to LST to write the dump on the OUTPUT file and then a branch is taken to DML.

5.9 (DMA) The first word address of the area to be dumped is set equal to zero and the last word address, to the execution field length. These addresses are stored for a call to LST.

5.9.1 An O format specification is stored for a call to OUTPTC.

5.9.2 A RJ is made to LST to write the dump on the OUTPUT file and then a branch is taken to exit.

5.10 (LSS) If an intermediate entry to OUTPTC has to be made, a branch is taken to LSM. Otherwise, B1 is set equal to minus one and a RJ is made to OUTPTC. to signal the end of the dump. A branch is then taken to exit.

5.11 (LST) The parameters for the first entry to OUTPTC are set up.

5.11.1 If the routine was entered at PDUMP, a branch is taken to LSTA. Otherwise, a RJ is made to OUTPTC to signal the beginning of a dump from DUMP.

5.11.2 (LSTA) A RJ is made to OUTPTC to signal the beginning of a dump from PDUMP.

5.12 (LSA) B1 is set to the beginning address of the area to be dumped and this address is stored in the array to be written on the OUTPUT file by OUTPTC.

5.12.1 The beginning address of the area to be dumped is updated and B2 is set equal to zero.

5.13 (LSP) If no words remain to be transferred to the array to be written on the OUTPUT file by OUTPTC, a branch is taken to LSS. Otherwise, B2 is incremented by one and the next word to be dumped is stored into the array to be written on the OUTPUT file by OUTPTC.

5.13.1 If less than four of the words to be dumped have been stored into the array to be written on the OUTPUT file by OUTPTC, a branch is taken to LSP.

5.14 (LSM) B2 is increased by one. A RJ is made to OUTPTC to write out one line of the dump and then a branch is taken to LSA.

A4. DVCHK

1.0 General Information

Routine provides for compatibility. It is called in response to the FORTRAN statement IF DIVIDE CHECK n_1, n_2.

1.1 Length: 6

2.0 Entry Points

2.1 DVCHK

2.1.1 Calling Sequence and Returns

Entry B1 - address to return value in

Exit X6 - indefinite, a 1 is returned or out of range

otherwise a 2

3.0 Diagnostics: none

4.0 External Routine: none

A5. LEGVAR

1.0 General Information

Called in response to a function reference LEGVAR (a) where a is a variable. It checks the legitimacy of the specified library and returns the following values:

-1 - variable indefinite
 0 - variable legitimate
 1 - variable out of range

1.1 Length: 5

2.0 Entry points

2.1 LEGVAR

2.1.1 Calling Sequence and Returns

Entry B1 - contains address of variable in question

Exit X6 = 1 or 0 or -1 (see 1.0)

3.0 Diagnostics: none

4.0 External Routines: none

A6. LOCF

1.0 General Information

LOCF is called as a function to find the address of a variable.

1.1 Length: 5

2.0 Entry Points

2.1 LOCF

2.1.1 Calling Sequence and Returns

Entry B1 = contains address of variable
X6 = contains address of variable

2.2 XLOCF - synonymous entry with LOCF

2.2.1 Calling Sequence and Returns

Entry B1 contains address of variable
X6 contains address of variable

3.0 Diagnostics: none

4.0 External Routines: none

A7. OVERLAY

1.0 General Information

FORTRAN statement routine is called for CALL OVERLAY (fn,l,l_2r).

OVERLAY will translate the information. It is sent into a call to the loader, which will cause the requested overlay to be loaded. After loading, control is returned to this routine, which then sets up the entry exit/line of the overlay and transfers control to the overlay.

1.1 Length: 45_8

2.0 Entry Points

2.1 OVERLAY

2.1.1 A return jump is made to this entry point after the following registers are set:

B1 - address of fn
B2 - address of l_1
B3 - address of l_2
B4 - address of r

This will cause the requested overlay to be loaded and control transferred to it.

3.0 Diagnostics

3.1 Informative: None

3.2 Fatal to execution

3.2.1 FATAL ERROR IN LOADER

This is given when the loader sets the fatal error bit in the loader control words.

4.0 External Routines

4.1 SYSTEM

4.1.1 Called to list the error message and give traceback.

4.2 ABNORML

4.2.1 Call to close out and abort the job.

5.0 Structure

5.1 Check fourth parameter. If it wasn't "RECALL" go to 5.2 (OVA). Otherwise check to see if the requested overlay was the last one loaded. If not, go to 5.2 (OVA). If it was, initialize the entry/exit line of the overlay and enter the overlay.

5.2 (OVA) Delete trailing blanks from the file name and set up the parameters for the loader. Call the loader to have the overlay loaded. If fatal errors were detected, go to 5.3 (OVE). Otherwise, save the entry address of the overlay, initialize the entry/exit line of the overlay, and enter the overlay.

5.3 (OVE) Call SYSTEM to print the error message and give traceback. Then call ABNORML to abort the job.

6.0 FORMATS

6.1 OVX - bits 54-59 l_1
48-53 l_2
30-42 Entry Address

This call is set each time an overlay has been loaded and checked when the RECALL parameter is specified.

6.2 OVR - contains "RECALLbbbb" in display code (b means a blank).

6.3 OVP - First of three words that are used for communication with the loader:

#1 file name with trailing blanks eliminated
#2 bits 54-59 l_1
48-53 l_2
41 =1 (indicates an overlay load)
#3 0

A8. OVERFL

1.0 General Information

This routine, provided for compatibility, is call in response to the following FORTRAN statements:

IF QUOTIENT OVERFLOW n_1,n_2

IF ACCUMULATOR OVERFLOW n_1,n_2

1.1 Length: 5

2.0 Entry Points

2.1 OVERFL

2.1.1 Calling Sequence and Returns

Entry B1 - contains address to return flag in

Exit flag = 1 if X6 is out of range
flag = 2 otherwise

3.0 Diagnostics: None

4.0 External Routines: None

A9. PAUSE

1.0 General Information

The function of this routine is to display the words PAUSE n as a dayfile message and stop program execution until the console operator makes an entry to either continue or terminate the program in response to the FORTRAN statement PAUSE n where n ≤ 5 octal digits.

1.1 Length: 22_8

2.0 Entry Points

2.1 PAUSE

2.1.1 Calling Sequence and Returns

The routine is entered by doing a RJ to PAUSE. X7 should contain the number n.

3.0 Diagnostics

3.1 No diagnostics are given

4.0 External Routines

4.1 No external routines are used.

5.0 Structure

5.1 No B registers are used.

5.2 An MSG request is set up with the message PAUSE n.

5.3 (PAV) If the MSG request cannot be issued at this time, a branch is taken to PAV. Otherwise, the MSG request is issued.

5.3.1 The PAUSE bit is set in RA.

5.4 (PAX) If the PAUSE bit is cleared, a branch is taken to exit. Otherwise, a RCL request is set up.

5.5 (PAY) If the RCL request cannot be issued at this time, a branch is taken to PAY. Otherwise, the RCL request is issued and a branch is taken to PAX.

A10. REMARK

1.0 General Information

The function of this routine is to place a message of not more than 40 characters in the dayfile in response to the FORTRAN statement CALL REMARK (H) where H is a Hollerith specification of not more than 40 characters.

1.1 Length: 22_8

2.0 Entry Points

2.1 REMARK

2.1.1 Calling Sequence and Returns

The routine is entered by doing a RJ to REMARK. B1 should contain the beginning address of the four word array containing the message to be placed in the dayfile.

3.0 Diagnostics

3.1 No diagnostics are given.

4.0 External Routines

4.1 CPC

4.1.1 Calling Sequence and Returns

The RJ to CPC must be in the lower half of a word. The word following the RJ to CPC must contain the display-coded name of the called PP program in the high order 18 bits and the parameters for that program in the low order 36 bits. Bit 40 of that word is the recall bit and bit 41 signifies whether it is a file or system action request. Upon exit, control is transferred to the second word following the RJ to CPC.

5.0 Structure

5.1 No B registers are used.

5.2 The four words containing the message to be placed in the dayfile are fetched and stored for the MSG request.

5.3 The MSG request is set up for CPC.

5.4 (NOP) A RJ is made to CPC (with the following word designating a MSG request with recall) to place the message on the dayfile and then a branch is taken to exit.

A11. SCOPE2B

1.0 General Information

This routine is called from a control card SCOPE2B. It expects one or two parameters. SCOPE2B (n,efn) where n is the number of files and efn is the logical file name upon which information is to be written. Its purpose is to write loader information on the efn which will create a labeled common block, named SCOPE2B, large enough to contain FETs for all files in a FORTRAN program. This provides compatibility with prior to RUN 2.3 binaries because of the expanded length of the 2.3 FET.

1.1 Length: 53_8

2.0 Entry Points

2.1 SCOPE2B

2.1.1 Calling Sequence and Returns: None

3.0 Diagnostics produced.

3.1 BAD PARAMETER - placed in dayfile if the number of parameters is not 1 or 2.

4.0 External Routines: None

A12. SECOND

1.0 General Information

The function of this routine is to return the CP time used in floating point seconds.

1.1 Length: 24_8

2.0 Entry Point

2.1 SECOND

2.1.1 Calling Sequence and Returns

Routine is entered by a RJ to SECOND. There are no arguments.

2.1.3 SECOND requests action of PP routines TIM and RECALL.

4.0 External Routines

4.1 TIM

4.1.1 Calling Sequence and Returns

A request for action by PP routine TIM is initiated by placing in location RA+1 absolute

59	41	17	0
TIM	0 —— 0	pointer	

The pointer (bits 0 through 17) points at the relative address to which TIM is to return the CP time used. The value is returned in the form

59	35	11	0
zeros	seconds	milliseconds	

4.2 RECALL

4.2.1 Calling Sequence and Returns

A request for recall is issued by placing in location RA+1 absolute

59	41	0
RCL	—— zeros ——	

5.0 Structure

5.1 AO and all B registers are undisturbed.

A13. SEGMENT

1.0 General Information

The function of this routine is to form a request to the LOADER to load and link/delink program segments in response to the FORTRAN call

```
CALL SEGMENT (FN,LEVEL,SEG,LIB,MAP)
```

where

FN = variable name of location containing file name (in left justified display code) from which loading should take place.

LEVEL = level of segment load

SEG = Simple or subscripted variable name of array containing a list of segments, sections or subprograms (in left justified display code) to be loaded with this call. The list must be terminated by a zero word. If the first entry is a zero word, all subprograms remaining on the file FN are to be loaded.

The remaining parameters are optional

LIB = if zero or left out, an attempt will be made to satisfy unsatisfied externals from the system library.

MAP = if zero or left out, a map of the segment load will be produced.

1.1 Length: 131_8

2.0 Entry Points

2.1 SEGMENT

2.1.1 Calling Sequence and Returns

The routine is entered by a standard FORTRAN call, an RJ in the upper 30 bits of a word followed by the traceback information in the lower 30 bits.

RJ SEGMENT	07	0X	CALLER

30 18 0

On entry the addresses of the arguments should be in the B registers beginning with B1. Since the number of arguments is variable, SEGMENT looks back into the traceback information to determine the number of arguments passed to it.

3.0 Diagnostics

3.2 Fatal to Execution

3.2.1 "ARGUMENTS ILLEGAL," ERROR NUMBER 51 will be given if illegal names appear in the segment list. Note that the failure to terminate the segment list with a zero word should cause an illegal name to be found.

3.2.2 "FATAL ERROR IN LOADER," ERROR NUMBER 51 will be given if the loader returns with its fatal error flag set.

3.3 Warnings

3.3.1 "NON-FATAL ERROR IN LOADER," ERROR NUMBER 52 will be given if the loader returns with its non-fatal error flag set and if a map has not been suppressed.

4.0 External Routines

4.1 LOADER

4.1.1 Calling Sequence and Returns

The loader is called by an RJ to loader. The word following the RJ must contain the address of the load sequence parameter list. Entries in this list are two words long, the last of which must be followed by a zero word. SEGMENT forms only one entry; this entry is in PARM and PARM1 and is followed by a zero word ZERO.

```
              +--------------+------------------+
              | RJ  LOADER   |                  |
              +--------------+------------------+
                            30                  0
              +--------------+------------------+
              |              | address of PARM  |
              +--------------+------------------+
                                   17           0
              +-------------------+-------------+
PARM          | File Name         |     S1      |
              +-------------------+-------------+
              +---------------------------------+
PARM1         |                                 |
              +---------------------------------+
              +---------------------------------+
ZERO          | 0 ----------------------------0 |
              +---------------------------------+
```

Bits 0 to 17 of the first word of the entry should point to the location of the list of segments or subprograms to be loaded from the file. If the remainder of the file is to be loaded, these bits should be zero.

The loader returns a reply in the load segment parameter list. Bit 37 of the second word is the fatal error flag and bit 36 of the second word is the non-fatal error flag.

4.2 CIO

4.2.1 Calling Sequence and Returns

If necessary a request of CIO is made to empty the output buffer and thus correctly position the output file so that the loader written map will appear in the correct position in the listing. The request by setting the code and status bits of the OUTPUT file FET to 24_8 and placing in location RA+1 (absolute) the word

CIO	FET address
18	0

6.0 Formats

The list of segments or subprograms to be loaded (third input parameter) must have all trailing blanks from legal alphanumeric identifiers at the first word of the list and continues until a zero word is found (list terminator) or until an illegal alphanumeric identifier is found.

A14. SLITE

1.0 General Information

This routine is called in response to the FORTRAN statement SENSE LIGHTi to turn on the specified sense lite. If the lite number is zero, all the sense lites are turned off.

1.1 Length: 15_8

2.0 Entry Points

2.1 SLITE

2.1.1 Calling Sequence and Returns

The routine is entered by doing a RJ to SLITE. B1 is expected to contain the address of the lite number. Upon exit no registers are expected set.

3.0 Diagnostics

3.1 Informative

3.1.1 45_8 - BAD SENSE LITE NUMBER is given if the lite number is not 0 through 6.

3.2 Fatal: none

4.0 External Routines

4.1 SYSTEM

4.1.1 Calling Sequence and Returns

Entry X1 = error number
X2 = address of diagnostic message

5.0 Structure

5.1 The lite number is checked to see that it is between 0 and 6. If not an error message is given.

5.2 The specified lite is turned on. If the number is zero all lites are turned off.

5.3 The routine then exits.

6.0 Formats

6.1 Lites: They are kept in RA. Bit 0 is lite 1, bit 1 is lite 2, etc.

A15. SLITET

1.0 General Information

This routine is called in response to the FORTRAN statement IF (SENSE LIGHT i)n_1,n_2 or when SLITET(i) is used as a function. Its purpose is to turn off the specified sense lite. It returns in X6 and in the memory location a 1 if the specified lite was on or a 2 if it was not.

1.1 Length: 20_8

2.0 Entry Points

2.1 SLITET

2.1.1 Calling Sequence and Returns

The routine is entered by doing a RJ to SLITET. B1 is expected to be set to the memory location containing the lite number and B2 to the address in which the flag is to be returned. If the lite is on, it is turned off and a 1 is returned. If it was off a 2 is returned. Their value return is both through memory and X6.

3.0 Diagnostics produced.

3.1 Informative

3.1.1 46_8 - BAD SENSE LITE NUMBER is given if the lite number is not between 1-6.

4.0 External Routines

4.1 SYSTEM

4.1.1 Calling Sequence and Returns

Entry X1 = error number
X2 = address of diagnostic message

5.0 Structure

5.1 The lite number is checked to see that it is between 1 and 6. An informative error is given if it is not.

5.2 The specified lite is turned off.

5.3 1 is returned if the lite was on, 2 if it was off.

A16. SSWTCH

1.0 General Information

Routine called in response to the FORTRAN statement IF (SENSE SWITCH i)n_1,n_2.

1.1 Length: 17_8

2.0 Entry Points

2.1 SSWTCH

2.1.1 Calling Sequence and Returns

Entry	B1 - points to switch number B2 - address of variable to return flag in

Exit	flag = 1 if SWITCH i is on
	flag = 2 if SWITCH i is off

3.0 Diagnostics

3.1 Informative: none

3.1.1 BAD SWITCH NUMBER

4.0 External Routines

4.1 SYSTEM

4.1.1 Calling Sequence and Returns

X1 = error number
X2 = address of error message

A17. START

1.0 General Information

This routine is called by the statement CALL START. It places the word START in the dayfile.

1.1 Length: 7

2.0 Entry points

2.1 START

2.1.1 Calling Sequence and Returns

Entered by a RJ START. No registers are expected to be set.

3.0 Diagnostics produced: none

4.0 External routines: none

5.0 Structure

5.1 A delay is taken until RA+1 is zero.

5.2 A "MSG" request is sent to the operating system specifying "START" to be placed in the dayfile.

5.3 The routine exits.

A18. TIME

1.0 General Information

The function of this routine is to write a message of up to 50 characters in the dayfile.

1.1 Length: 23_8

2.0 Entry Point

2.1 TIME

2.1.1 Calling Sequence and Returns

Routine is entered by an RJ to TIME. B1 contains the address of the first word of the message, or is zero if there is no message.

2.1.3 TIME requests action by PP routine MSG.

4.0 External Routines

4.1 Calling Sequence and Returns

A request for action by PP routine MSG is initiated by placing in location RA+1 absolute.

```
59        41          17          0
 ________________________________
|   MSG   |  0 ---- 0 |  pointer  |
|_________|___________|___________|
```

The pointer (bits 0 through 17) points at the relative address of the first word of the message to be printed. This word always contains "TIMEbbbbbb," and is followed by the input message or, if none, a zero word.

6.0 Formats

The input message must always end on a word boundary, and if less than 5 words must be terminated by a zero word.

DOCUMENT CLASS IMS PAGE NO. IV-A1
PRODUCT NAME RUN 2.3 - OBJECT LIBRARY
PRODUCT MODEL NO. C010 * 2.3 MACHINE SERIES 64/65/6600

IV. Mathematical Library

A. Library Functions

A1. ALNLOG

ENTRY POINTS: ALOG, ALOG10

PURPOSE:
ALOG : Evaluate the natural logarithm of a real number.
ALOG10 : Evaluate the logarithm to base 10.0 of a real number.

USAGE:
A = ALOG(X)

A = ALOG10(X)
where X is the single precision floating point argument and A is the result in single precision floating point.

METHOD:

ALOG10(X) = $\log_{10}$(e)*ALOG(X)

let $X = 2^K * W$, $.5^{1/2} \leq W < 2^{1/2}$

$\log_e (X) = K*\log_e(2.0) + \log_e(W)$

$\log_e (W) = \log_e\left(\frac{1+t}{1-t}\right) = 2t - Q*t,$

$Q = \frac{n_1 Z + n_2 Z^2 + n_3 Z^3}{d_0 + d_1 Z + d_2 Z^2 + d_3 Z^3}$, $Z = t^2$

LANGUAGE: COMPASS

TIME:
28 μs for ALOG
32 μs for ALOG10

STORAGE:
55_{10} words. 35 words of program.
20 words of constants.

ACCURACY:
ALOG: In the range of $.5 \leq X \leq 2$ the maximum observed absolute error was 3.2E-15 for values of X outside this range the maximum observed relative error was 5.8E-15

RESTRICTIONS:
If $X \leq 0$, the result is set to - infinity and the normal exit is taken.

A3. ATAN

PURPOSE: Evaluate the arctangent of a real number

USAGE: A = ATAN(X)

where X is the single precision floating point input argument and A is the result in single precision floating point.

METHOD: Let A = ATAN(X), then $-PI/2 < A < +PI/2$

Let P = tan (PI/16), T = tan (3PI/16)

ATAN(W) = sign (W)*ATAN(V), V = ABS(W)

ATAN(V) = ATAN(R) + C, R, and C defined below.

$0 \leq V < P$, R = V, C = 0.0

$P \leq V < 2^{1/2}-1$, R = (V-P)/(1 + V*P), C = PI/16

$2^{1/2}-1 \leq V < 1$, R = (V-T)/(1 + V*T), C = 3PI/16

$1 \leq V < 2^{1/2}+1$, R = (1-V*T)/(V+T), C = 5PI/16

$2^{1/2}+1 \leq V$, R = (1-V*P)/(V + P), C = 7PI/16

ATAN(R) = R - R*Q, $Z = R^2$,

$$Q = \frac{n_0 + n_1Z + n_2Z^2 + n_3Z^3}{d_0 + d_1Z + d_2Z^2 + d_3Z^3}$$

LANGUAGE: COMPASS

TIME: 32 µs

STORAGE: 60_{10} words, 26_{10} words of program

ACCURACY: In ATAN when $|X| \leq .196$ the maximum observed absolute error was 4.6E-16, and for all other values of X the maximum observed relative error was 8.OE-15.

RESTRICTIONS: In ATAN if X is indefinite or out-of-range the result is set to indefinite and the normal return taken.

EXTERNAL REFERENCES: SYSTEM

A4. ATAN2

PURPOSE: Evaluate the arctangent of the ratio of two real numbers.

USAGE: A = ATAN2(Y,X)

where X and Y are the single precision floating point input arguments and A is the single precision floating point result.

METHOD: Let B = ATAN2(Y,X), then B is the argument of the complex number X+iY and $-PI \le B \le +PI$

$$B = \begin{cases} sign(Y)*PI/2 & ,X = 0 \\ ATAN(Y/X) & ,X > 0 \\ ATAN(Y/X)+sign(Y)*PI & ,X < 0 \end{cases}$$

LANGUAGE: COMPASS

TIME: 43 µs

STORAGE: 74_{10} words, 40_{10} words of program

ACCURACY: When $|Y/X| \le .196$ the maximum observed absolute error was 1.6E-15 and for all other values of Y and X the maximum observed relative error was 3.7E-14.

RESTRICTIONS: If X or Y is indefinite or out-of-range the result is set to indefinite. If X=Y=0 the result is set to indefinite.

EXTERNAL REFERENCES: SYSTEM

A5. CABS

PURPOSE:	Compute the magnitude of a complex number .
USAGE:	R = CABS(Z) where R and Z represent the complex numbers
METHOD:	$Z = X + iY$, $R = U + iV$ $U = (X^2 + Y^2)^{1/2}$, $V = 0$ The square root function is evaluated as described in its procedure write up.
LANGUAGE:	COMPASS
STORAGE:	30_{10} words, 15_{10} words of program.
TIME:	21 μs
ACCURACY AND RESTRICTIONS:	See square root description

A7. CCOS

PURPOSE: Evaluate the complex valued sine of a complex number.

USAGE: R = CCOS(Z)

where Z is the complex valued input argument and R is the complex valued result

METHOD: Let Z = X + iY, R = U + iV, then

$U = \text{COS } (X)*(e^{Y} + e^{-Y})/2$

$V = \sin\ (X)*(e^{-Y}-e^{Y})/2$

The evaluation of the real valued sine, cosine, and exponential functions are described in their respective routines.

LANGUAGE: COMPASS

STORAGE: 37_{10} words, 21_{10} words of program

TIME: 20+EXP+SIN+COS

ACCURACY AND RESTRICTIONS: See individual descriptions

EXTERNAL REFERENCES: EXP, SIN, COS, SYSTEM

A8. CEXP

PURPOSE:	Evaluate the complex valued exponential of a complex number.
USAGE:	R = CEXP(Z) where Z is the complex valued input argument and R is the complex valued result.
METHOD:	Let Z = X +iY, R = U +iV then $U = \cos(Y)*e^X$, $V = \sin(Y)*e^X$ The real valued sine, cosine, and exponential functions are evaluated as described in their respective procedures.
LANGUAGE:	COMPASS
STORAGE:	27_{10} words, 16_{10} words of program
TIME:	10 μs +SIN+COS+EXP
ACCURACY AND RESTRICTIONS:	See individual descriptions
EXTERNAL REFERENCES:	SYSTEM, SIN, COS, EXP

A9. CLOG

PURPOSE: Evaluate the complex logarithm of a complex number.

USAGE: W = CLOG(Z)

where W and Z are the complex valued arguments.

METHOD: Let Z = X + iY and W = U + iV,

then U + iV = CLOG(Z) and

$U = .5 \log_e(X^2 + Y^2)$,

$V = \text{arctangent } (Y/X), -PI \leq V \leq PI$.

The methods used in evaluating the log and arctangent are found in the description of ALOG and ATAN2.

LANGUAGE: COMPASS

TIME: 10 μs + ALOG + CABS + ATAN2

STORAGE: 24_{10} words, 15_{10} words of program

ACCURACY: See descriptions of ALOG, ATAN2

RESTRICTIONS: If Z = 0, U is set to -infinity, V is set to indefinite and a normal return taken.

EXTERNAL REFERENCES: SYSTEM, ALOG, CABS, ATAN2

A10. CSIN

PURPOSE: Evaluate the complex valued sine of a complex number.

USAGE: R = CSIN(Z)

where Z is the complex valued input argument and R is the complex valued result.

METHOD: Let Z = X +iY, R = U +iV then

$U = \sin (X)*(e^Y + e^{-Y})/2$

$V = \cos (X)*(e^Y - e^{-Y})/2$

The evaluation of the real valued sine, cosine, and exponential functions are described in their respective routine procedures.

LANGUAGE: COMPASS

STORAGE: 37_{10} words, 21_{10} words of program

TIME: 2-+EXP+COS+SIN

ACCURACY AND RESTRICTIONS: See individual descriptions

EXTERNAL REFERENCES: EXP, COS, SIN, SYSTEM

A11. CSQRT

PURPOSE: Compute the square root of a complex number

USAGE: R = CSQRT(Z)

where R and Z represent the complex numbers

METHOD: $Z = X + iY$, $R = U + iV$

$$a = (X^2 + Y^2)^{1/2}$$

$$b = \left[(a + \left|X\right|)/2\right]^{1/2}$$

$$c = Y/2b$$

If $X \geq 0$ $U = b$ and $V = c$

$X < 0$ $U = c*\mathrm{sign}(Y)$

$V = b*\mathrm{sign}(Y)$

The square root function is evaluated as described in its procedure write-up.

LANGUAGE: COMPASS

STORAGE: 29_{10} words, 17_{10} words of program

TIME: 16+SQRT+CABS

ACCURACY AND RESTRICTIONS: See square root description.

EXTERNAL REFERENCES: SQRT, SYSTEM, CABS

A13. DATAN

ENTRY POINTS: DATAN, DATAN2

PURPOSE: DATAN evaluates the arctangent of a double precision number.

DATAN2, evaluates the arctangent of the ratio of two double precision numbers.

USAGE: A = DATAN (X)

A = DATAN2 (Y, X), where X and Y are the double precision input arguments and A is the double precision result.

METHOD: Let A = DATAN (X), then - PI/2 A PI/2

Let B = DATAN2 (Y, X), then B is the argument of the complex number X + iY and - PI $\leq$ B $\leq$ PI.

$$B = \begin{cases} \text{Sign (Y)*PI/2} & , X = 0 \\ \text{DATAN (Y,X)} & , X \quad 0 \\ \text{DATAN (Y,X)* Sign (Y)*PI,} & X < 0 \end{cases}$$

Let P = Tan (PI/16), T = Tan (3PI/16)

Let W = A or B as appropriate

Atan (W) = Sign (W)* Atan (V), V = $|W|$

Atan (V) = Atan (R) + C, R and C

DATAN (continued)

Defined Below

$0 \leq V < P,$ $R = V,$ $C = 0$

$P \leq V < -\sqrt{2},$ $R = (V - P)/(1 + V*P),$ $C = PI/16$

$-\sqrt{2} \leq V < 1,$ $R = (V - T)/(1 + V*T),$ $C = 3PI/16$

$1 \leq V < \sqrt{2},$ $R = (1 - V*T)/(V + T),$ $C = 5PI/16$

$\sqrt{2} \leq V,$ $R = (1 - V*P)/(V + P),$ $C = 7PI/16$

Where DATAN(R) is computed from a Taylor-Maclaurin polynomial of degree 27. This polynomial was telescoped from the Taylor-Maclaurin power series of degree 39.

TIMING: 96 µs

STORAGE: 201_8 CM words for both DATAN and the entry point

DATAN2. 114_8 CM words of program.

DATAN (continued)

ACCURACY: DATAN; For 1000 values of X uniformly distributed $0 \leq X < .196$, the maximum observed absolute error was 4.3×10^{-29}.

For 2400 values of X randomly distributed $.196 \leq X < 1000$. The maximum observed relative error was 2.2×10^{-28}.

DATAN2: Results were the same as for DATAN when the denominator was positive. For a similar test, but with a negative denominator, the maximum observed relative error was 5.8×10^{-28}.

RESTRICTIONS: If X = 0 and Y = 0, the result is set to indefinite and appropriate error messages are generated.

EXTERNAL REFERENCES: SYSTEM.

A17. DEXP

PURPOSE: Evaluate the exponential of a double precision number.

USAGE A = DEXP (X), where X is the double precision argument and A is the double precision result.

METHOD: Let $N = \left[X/\log_e (2.) + .5 \right]$, and

$R = R1 + R2 = X - N*\log_e (2.), \left| R \right| \leq \log_e (2.)/2.$

R1 and R2 represent the more significant and less significant parts of R. $e^X = 2^{N}*e^{R2}$.

e^{R1} is evaluated from a polynomial of Degree 17. This polynomial was telescoped from a truncated MacLaurin power series of degree 20.

$e^{R2} = 1 + R2.$

TIMING: 81 µs.

STORAGE: 136_8 CM words. 53_8 words of program.

ACCURACY: For 5000 values of X uniformly distributed in the range $\left| X \right| \leq \log_e (2.)/2.$ The maximum observed relative error was 3.6×10^{-29}.

For 5000 values of X uniformly distributed in the range $\left| X \right| \leq 600.$, the maximum observed relative error was $8. \times 10^{-29}$.

RESTRICTIONS: If $X \geq 743$, the result A is set to infinity. If $X \leq -675$. the result is set to zero. In either case, the appropriate error message is generated.

EXTERNAL REFERENCES: SYSTEM.

A18. DLNLOG

ENTRY POINTS: DLOG, DLOG10

PURPOSE: Evaluate the natural logarithm or logarithm to base 10.0 of a double precision number.

USAGE: A = DLOG(X)

A = DLOG10(X)

where X is the double precision input argument and A is the double precision result.

METHOD: DLOG

$X = 2^K*W, \quad .5^{1/2} \leq W < 2^{1/2}$

$\log_e(X) = K*\log_e(2.0) + \log_e(W)$

$\log_e(W)$ is approximated by a_o by the .equation $c_1*t + c_3*t^3 + c_5*t^5 + c_7*t^7$, $t = (W-1)/(W+1)$, (Hastings, Approximations for Digital Computers, Princeton, University Press, 1955), which yields about 32 binary digits of accuracy. Two Newtons iterations are done to yield the double precision value of $\log_e(W)$. The iteration formula for $f(a) = e^a - X = 0$ is

$$a_{n+1}=a_n - (1 - X*e^{-a}n).$$

The term e^{-ao} is calculated in double precision using the same coding as e^{R1} in DEXP. Let $R = Xe^{-ao}$ and $T = 1.0 - R$, R1, T1, R2, T2 denote the 2 significant parts of R and T. Led $a_1 = a_0 - T1$

$$a_2 = a_1 - (1 - Xe^{-a_o+T_1})$$

$$= a_o - T1 - (1 - R*(1 + T1 + T1^{2/2} + T1^{3}/6))$$

$$= a_o - T1 - (1 - R - R*T1 - R*T1^{2/2} - R*T1^{3}/6)$$

$$1 - R = T1 + T2$$

$$R*T1 = T1 - T1^{2} - \underline{T1*T2} = T1 - T1^{2}$$

$$R*T1^{2/2} = T1^{2/2} - T1^{3/3} - \underline{T1^{2}* T2/2}$$

$$\approx T1^{2/2} - T1^{3/2}$$

$$R*T1^{3/6} = T1^{3/6} - \underline{T1^{4/6}} - \underline{T1^{3}*T2/6} = T1^{3/6}$$

where the underlined terms are ignored because they are insignificant with respect to the desired accuracy of the final result.

$$a_2 = a_o - T1 - T2 - T1^{2} (1/2 + T1/3)$$

which is the actual computing formula used.

DLOG10. $\log_{10}(X) = \log_{10}(e)*\log_e(X)$

LANGUAGE: COMPASS

TIME:
96 μs DLOG
100 μs DLOG10

STORAGE: 126 words

ACCURACY: DLOG and DLOG10

For 2000 values of X uniformly distributed in the range $.5 \leq X \leq 2$, the maximum observed absolute error was 2.4E-29. For 2000 values of X, $X < .5$ and $X > 2$, the maximum observed relative error was 2.1E-29.

RESTRICTIONS: DLOG and DLOG10

If X = 0, the result is set to -infinity, if $X \leq 0$ the result is set to indefinite. For both cases a normal return is taken.

EXTERNAL REFERENCE: SYSTEM

A21. DSINCOS

PURPOSE: Evaluate the sine or cosine function of a double precision quantity.

USAGE: A DSIN (X), A = DCOS (X), where

X represents the double precision input argument and A represents the double precision result.

METHOD: Let N = $\left[X/(PI/2) + .5 \right]$, and

R = X-N*PI/2, then $|R| \leq PI/4$.

Let K = N Modulo 4, K = 0, 1, 2, 3.

Then Sin (X) = Sin (R + N*PI/2) =

Sin (R + K*PI/2) = Sin (R) * Cos (K*PI/2) +

Cos (R) * Sin (K*PI/2) and a similar

formula for Cosine (X). Depending upon

whether Sine (X) or Cosine (X) was wanted

and upon the value of K, either the Sine

of Cosine of R is evaluated and complemented

if necessary

The Sine and Cosine of R are evaluated by

polynomials of degree 21 and 20 respectively.

These polynomials were telescoped from a

truncated Taylor - MacLaurin power series

of degree 25 and 24.

DSINCOS (continued)

TIMING: 73 µs

STORAGE: 172_8 CM words, 77_8 words of program. The maximum observed absolute error of DSIN and DCOS for 10,000 values of X, $| X | \leq$ PI/4 was 2.7×10^{-29}.

For 2000 values of X, PI/4 $<$ $|X|$ $\leq$ 10PI, The maximum absolute error was 6.21×10^{29}. The absolute error for other values of X will be $\lesssim 6.9 \times 10^{-29} + |X| *2^{-50} *10^{-29}$.

RESTRICTIONS: If $|X| \geq$ PI$*2^{94}$ the result is set indefinite and appropriate error messages are generated.

EXTERNAL REFERENCES: SYSTEM.

A22. DSQRT

PURPOSE: Evaluate the square root of a double precision number.

USAGE: A = DSQRT(X) where X is the double precision input argument and A is the double precision result.

METHOD: Let 2^N*W = more significant half of X,

$.5 \leq W < 1$, $N = 2K + r$, $r*N \geq 0$,

$r = -1, 0, +1$, then

$X^{1/2} = 2^K*2^{r/2}*W^{1/2}$.

Let $B = .585786W + .420495$ be the initial guess to $W^{1/2}$ and C be the result of two Newton's iterations done in single precision. Then $S = 2^K * 2^{r/2} * C$ differs from $X^{1/2}$ by approximately $10^{-9} * 2^K$. To get the double precision value of $X^{1/2}$ two Newton's iterations are done in double precision using the equation

$A_{n+1}=A_n-(A_n^2 - X) / 2A_n$ with $A_o = S$.

TIMING: 32.5 µs

STORAGE: 57_8 CM words 32_8 words of program

ACCURACY: The maximum observed relative error for 100,000 arguments uniformly distributed $.25 < x < 4.$ was 1.2×10^{-29}

RESTRICTIONS: If $X < 0$, the result is set to indefinite and appropriate error messages are generated

EXTERNAL REFERENCES: SYSTEM.

A23. EXP

PURPOSE: Evaluate the exponential of a real number

USAGE: Function call, A = EXP(X), where X is the single precision floating point argument and A is the result in single precision floating point.

METHOD: Let $N = \left[X/\log_e(2.) + .5 \right]$, and

$R = X - N*\log_e(2.)$, then $|R| \leq \log_e(2.)/2$

and $e^X = 2N*e^R$.

$e^R = 1 + R + Q$, where

$Q = R*(R*B - Z*T) / (2*B + Z*T - R*B)$

$Z = R^2$, $T = 28.* Z + 2520.$, and

$B = Z^2 + 420.*Z + 15120.$

TIMING: .30 μs

STORAGE: 57_8 CM words, 34_8 words of program.

ACCURACY: The maximum observed relative error for 70,000 values of X uniformly distributed in the range $|X| < .347$ was 3.8×10^{-15}. The maximum observed relative error for 435,000 values of X uniformly distributed over the range $-675 \leq x \leq 741$ was $5. \times 10^{-15}$

RESTRICTIONS: The result A is set to zero if $X \leq -675.82$, and is set to + infinity if $X \geq 741.67$.

EXTERNAL REFERENCES: SYSTEM.

A29. SINCOS

ENTRY POINTS: SIN, COS

PURPOSE: Evaluate the SINE or COSINE function of a real number.

USAGE: A = SIN(X).

A = COS(X),

where X is the single precision floating point input argument expressed in radians and A is the result in single precision floating point.

METHOD: Let N = $\left[X/(PI/2 + .5 \right]$, and

R = X - N*PI/2, then $|R| \leq$ PI/4.

Let K = Nmod4, K = 0, 1, 2, 3, then

sin(X) = sin(R + N*PI/2) = sin(R + K*PI/2)

= sin(R)*cos(K*PI/2 + cos(R)*sinK*PI/2)

and a similar formual for the cosine(X).

Depending upon K, either the sine (K = 0, 2) or cosine (K = 1,3) of R is evaluated and complemented if necessary.

The sine and cosine of R are evaluated by polynomials of degree 11 and 12, respectively. These polynomials were telescoped from a truncated Taylor-MacLaurin power series of degree 15 and 14.

LANGUAGE: COMPASS

TIME: 27.3 μs

STORAGE: 63 words, 39 words of program, 24 words of constants and error messages.

ACCURACY: The maximum observed absolute error for 10,000 values of X uniformly distributed in the basic range $|X| \leq$ PI/4 was 5.3E - 15 for both the sine and cosine. The maximum observed absolute error for 20,000 values of X uniformly distributed in the range $|X| \leq$ 31.4 was 6.1E - 15 for both the sine and cosine.

SINCOS (continued)

RESTRICTIONS: If X $\geq$1.1E14 the result is set to indefinite and an error message generated. The program tests for an infinite or indefinite argument and if either condition exists, the appropriate message is generated.

EXTERNAL REFERENCES: SYSTEM

A31. SQRT

PURPOSE: Evaluate the square root of a real number.

USAGE: A = SQRT(X)

where X is the single precision floating point input argument and A is the result in single precision floating point.

METHOD: Let $X = 2^N*W, \quad .5 \leq W < 1,$

$N = 2K + r, \ r*N \quad 0,$

$r = -1, 0, +1,$ then $X^{1/2} = 2^K*2^{r/2}*W1/2$

Let $B = .585786437W + .4204951288$ be the initial approximation to W1/2,

Let C = result of two Newton's iterations using B as the initial guess to W1/2, then

$$\frac{4C = (B2 + W)^2 + 4B^2W}{B(B^2 + W)}$$

$$D \simeq 2X1/2 \simeq 2^{K-1}2^{r}/2 \ (4C)$$

$$A = X^{1/2} = .25*D + X/D$$

LANGUAGE: COMPASS

TIME: 21.0 μs

STORAGE: 35_{10} words. 28 words of program. 13 words of constants and error messages.

ACCURACY: The maximum observed relative error for 200,000 arguments uniformly distributed between .25 and 4.0 was 3.5E-15.

In addition 199,999 of the 200,000 results gave exact agreement with the true value when it was rounded to single precision.

RESTRICTIONS: If the argument is negative, out-of-range, or indefinite, the result is set to indefinite and a normal return is taken.

A33. TANH

PURPOSE: Evaluate the hyperbolic tangent of X.

USAGE: A = TANH (X) where X is the single precision floating point argument and A is the result in single precision floating point.

METHOD: For X = 0, TANH (X) = 0.

For $0 < |X| < 17.61$, TANH (X) = $\left[e^{2X}-1\right] / \left[e^{2X}+1\right]$

For $|X| \geq 17.61$, TANH (X) is set to ± 1.

TIMING: 44 µs.

STORAGE: 45_8 CM words. 20_8 words of instructions.

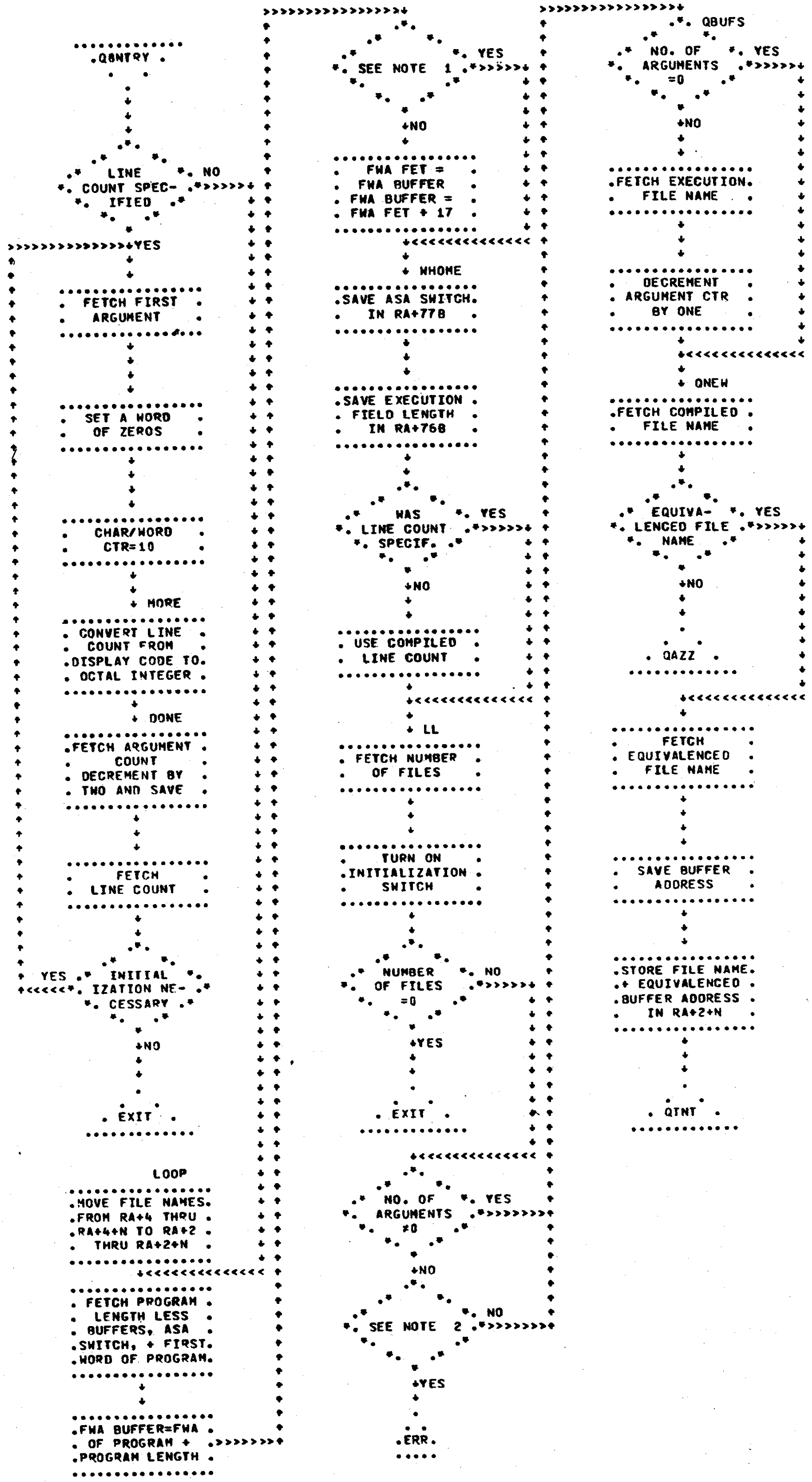
ACCURACY: $\approx 1.4 \times 10^{-14}$ on the average.

RESTRICTIONS: If $|X|$ is indefinite or out of range, the result is set indefinite and appropriate error messages are generated.

EXTERNAL REFERENCES: EXP, SYSTEM.

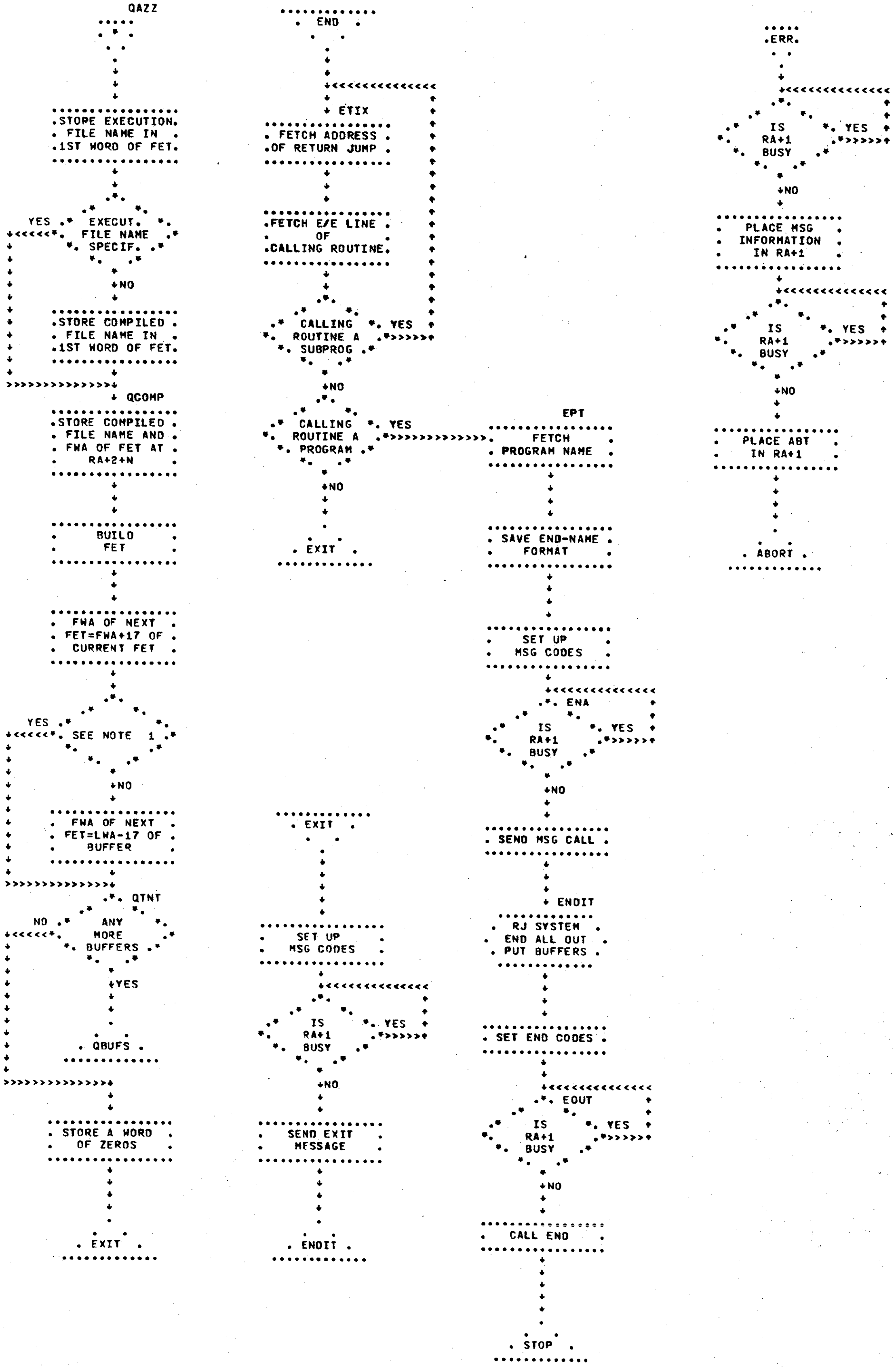
>SYSTEM<

NOTES
1. FET IN SCOPE2 COMMON BLOCK
2. SCOPE2 COMMON BLOCK TOO SMALL

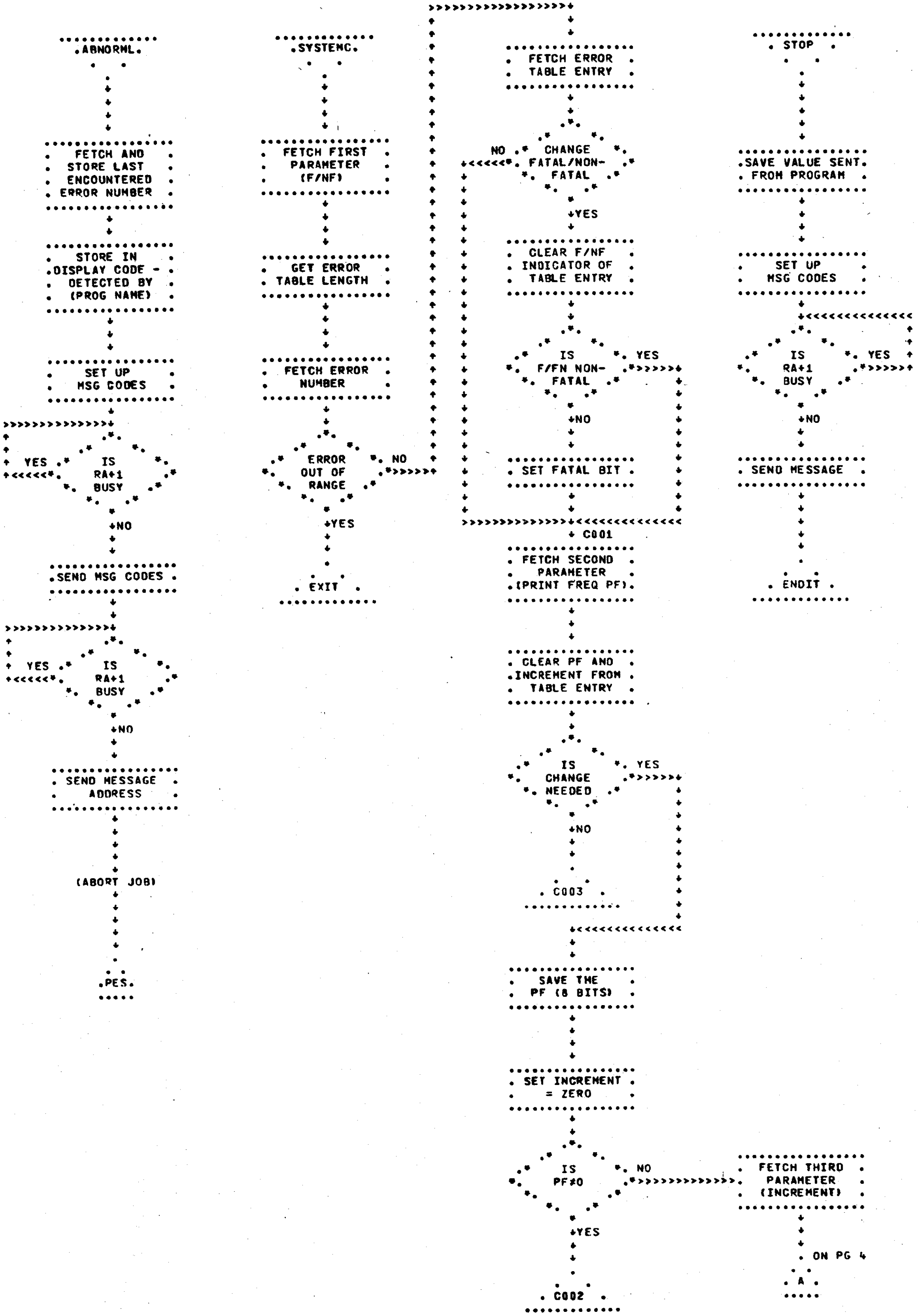

QBNTRY
LINE COUNT SPEC- IFIED
NO
YES
FETCH FIRST ARGUMENT
SET A WORD OF ZEROS
CHAR/WORD CTR=10
MORE
CONVERT LINE COUNT FROM DISPLAY CODE TO. OCTAL INTEGER
DONE
FETCH ARGUMENT COUNT DECREMENT BY TWO AND SAVE
FETCH LINE COUNT
YES
INITIAL IZATION NE- CESSARY
NO
EXIT
LOOP
MOVE FILE NAMES. FROM RA+4 THRU RA+4+N TO RA+2 THRU RA+2+N
FETCH PROGRAM LENGTH LESS BUFFERS, ASA SWITCH, + FIRST. WORD OF PROGRAM.
FWA BUFFER=FWA OF PROGRAM + PROGRAM LENGTH
SEE NOTE 1
YES
NO
FWA FET = FWA BUFFER FWA BUFFER = FWA FET + 17
WHOME
SAVE ASA SWITCH. IN RA+77B
SAVE EXECUTION FIELD LENGTH IN RA+76B
HAS LINE COUNT SPECIF.
YES
NO
USE COMPILED LINE COUNT
LL
FETCH NUMBER OF FILES
TURN ON INITIALIZATION SWITCH
NUMBER OF FILES =0
NO
YES
EXIT
NO. OF ARGUMENTS ≠0
YES
NO
SEE NOTE 2
NO
YES
ERR.
QBUFS
NO. OF ARGUMENTS =0
YES
NO
FETCH EXECUTION. FILE NAME
DECREMENT ARGUMENT CTR BY ONE
QNEW
FETCH COMPILED FILE NAME
EQUIVA- LENCED FILE NAME
YES
NO
QAZZ
FETCH EQUIVALENCED FILE NAME
SAVE BUFFER ADDRESS
STORE FILE NAME. + EQUIVALENCED BUFFER ADDRESS IN RA+2+N
QTNT

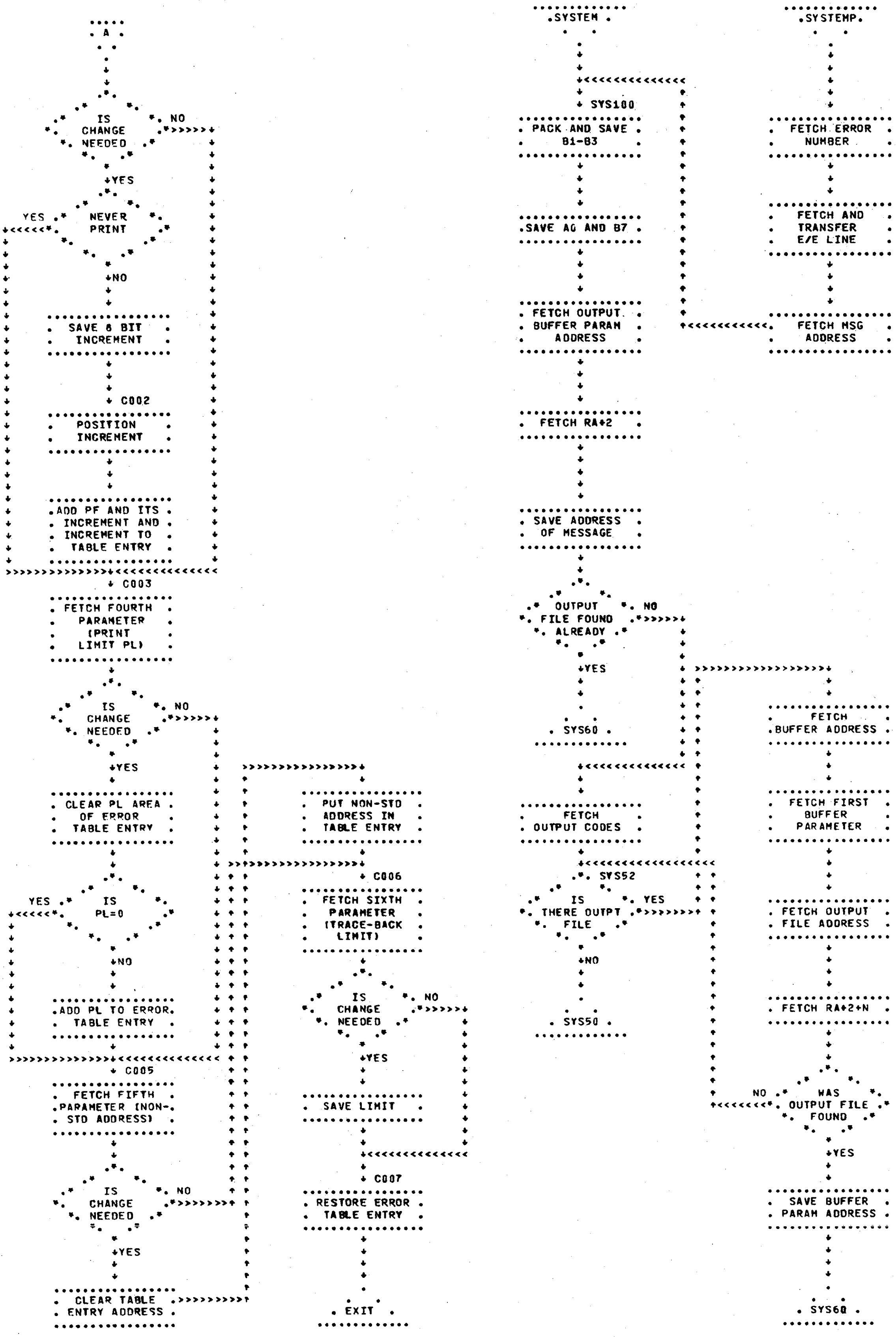
>SYSTEM<

NOTES
1. CURRENT FET IN SCOPE2 COMMON BLOCK

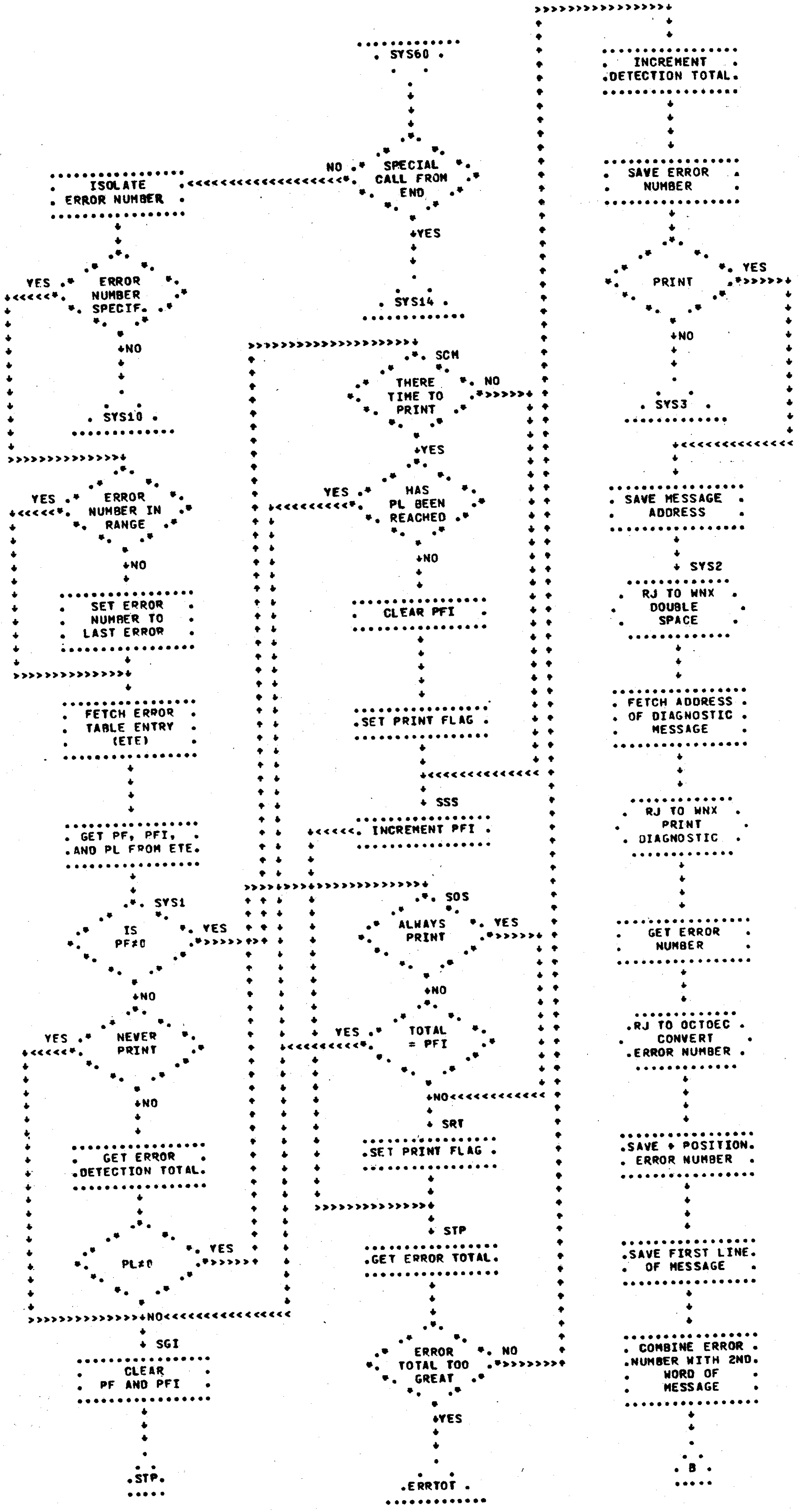

QAZZ
STORE EXECUTION. FILE NAME IN 1ST WORD OF FET.
EXECUT. FILE NAME SPECIF.
YES
NO
STORE COMPILED FILE NAME IN 1ST WORD OF FET.
QCOMP
STORE COMPILED FILE NAME AND FWA OF FET AT RA+2+N
BUILD FET
FWA OF NEXT FET=FWA+17 OF CURRENT FET
SEE NOTE 1
YES
NO
FWA OF NEXT FET=LWA-17 OF BUFFER
QTNT
ANY MORE BUFFERS
NO
YES
QBUFS
STORE A WORD OF ZEROS
EXIT
END
ETIX
FETCH ADDRESS OF RETURN JUMP
FETCH E/E LINE OF CALLING ROUTINE.
CALLING ROUTINE A SUBPROG
YES
NO
CALLING ROUTINE A PROGRAM
YES
NO
EXIT
EXIT
SET UP MSG CODES
IS RA+1 BUSY
YES
NO
SEND EXIT MESSAGE
ENDIT
EPT
FETCH PROGRAM NAME
SAVE END-NAME FORMAT
SET UP MSG CODES
ENA
IS RA+1 BUSY
YES
NO
SEND MSG CALL
ENDIT
RJ SYSTEM END ALL OUT PUT BUFFERS
SET END CODES
EOUT
IS RA+1 BUSY
YES
NO
CALL END
STOP
ERR.
IS RA+1 BUSY
YES
NO
PLACE MSG INFORMATION IN RA+1
IS RA+1 BUSY
YES
NO
PLACE ABT IN RA+1
ABORT

>SYSTEM<


ABNORML.
FETCH AND STORE LAST ENCOUNTERED ERROR NUMBER
STORE IN DISPLAY CODE - DETECTED BY (PROG NAME)
SET UP MSG CODES
IS RA+1 BUSY
YES
NO
SEND MSG CODES
IS RA+1 BUSY
YES
NO
SEND MESSAGE ADDRESS
(ABORT JOB)
PES.
SYSTEMC.
FETCH FIRST PARAMETER (F/NF)
GET ERROR TABLE LENGTH
FETCH ERROR NUMBER
ERROR OUT OF RANGE
NO
YES
EXIT
FETCH ERROR TABLE ENTRY
CHANGE FATAL/NON-FATAL
NO
YES
CLEAR F/NF INDICATOR OF TABLE ENTRY
IS F/FN NON-FATAL
YES
NO
SET FATAL BIT
C001
FETCH SECOND PARAMETER (PRINT FREQ PF)
CLEAR PF AND INCREMENT FROM TABLE ENTRY
IS CHANGE NEEDED
YES
NO
C003
SAVE THE PF (8 BITS)
SET INCREMENT = ZERO
IS PF≠0
NO
YES
C002
FETCH THIRD PARAMETER (INCREMENT)
ON PG 4
A
STOP
SAVE VALUE SENT FROM PROGRAM
SET UP MSG CODES
IS RA+1 BUSY
YES
NO
SEND MESSAGE
ENDIT

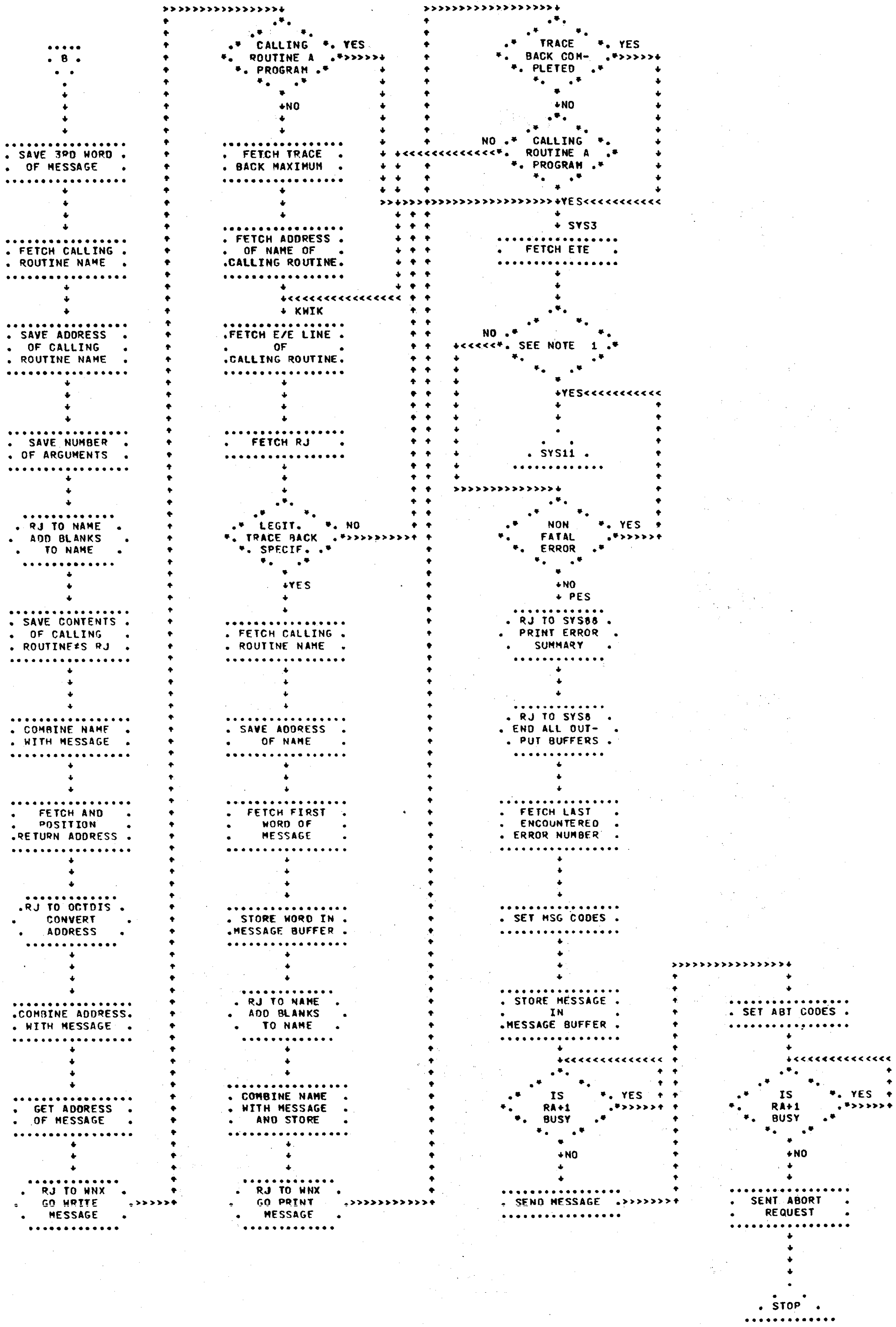

A
IS CHANGE NEEDED
NO
YES
NEVER PRINT
YES
NO
SAVE 8 BIT INCREMENT
C002
POSITION INCREMENT
ADD PF AND ITS INCREMENT AND INCREMENT TO TABLE ENTRY
C003
FETCH FOURTH PARAMETER (PRINT LIMIT PL)
IS CHANGE NEEDED
NO
YES
CLEAR PL AREA OF ERROR TABLE ENTRY
IS PL=0
YES
NO
ADD PL TO ERROR TABLE ENTRY
C005
FETCH FIFTH PARAMETER (NON-STD ADDRESS)
IS CHANGE NEEDED
NO
YES
CLEAR TABLE ENTRY ADDRESS
PUT NON-STD ADDRESS IN TABLE ENTRY
C006
FETCH SIXTH PARAMETER (TRACE-BACK LIMIT)
IS CHANGE NEEDED
NO
YES
SAVE LIMIT
C007
RESTORE ERROR TABLE ENTRY
EXIT
SYSTEM
SYS100
PACK AND SAVE B1-B3
SAVE A0 AND B7
FETCH OUTPUT BUFFER PARAM ADDRESS
FETCH RA+2
SAVE ADDRESS OF MESSAGE
OUTPUT FILE FOUND ALREADY
NO
YES
SYS60
FETCH OUTPUT CODES
SYS52
IS THERE OUTPT FILE
YES
NO
SYS50
SYSTEMP
FETCH ERROR NUMBER
FETCH AND TRANSFER E/E LINE
FETCH MSG ADDRESS
FETCH BUFFER ADDRESS
FETCH FIRST BUFFER PARAMETER
FETCH OUTPUT FILE ADDRESS
FETCH RA+2+N
WAS OUTPUT FILE FOUND
NO
YES
SAVE BUFFER PARAM ADDRESS
SYS60

>SYSTEM<


SYS60
SPECIAL CALL FROM END
NO
YES
SYS14
ISOLATE ERROR NUMBER
ERROR NUMBER SPECIF.
YES
NO
SYS10
ERROR NUMBER IN RANGE
YES
NO
SET ERROR NUMBER TO LAST ERROR
FETCH ERROR TABLE ENTRY (ETE)
GET PF, PFI, AND PL FROM ETE.
SYS1
IS PF≠0
YES
NO
NEVER PRINT
YES
NO
GET ERROR DETECTION TOTAL.
PL≠0
YES
NO
SGI
CLEAR PF AND PFI
STP
SCM
THERE TIME TO PRINT
NO
YES
HAS PL BEEN REACHED
YES
NO
CLEAR PFI
SET PRINT FLAG
SSS
INCREMENT PFI
SOS
ALWAYS PRINT
YES
NO
TOTAL = PFI
YES
NO
SRT
SET PRINT FLAG
STP
GET ERROR TOTAL.
ERROR TOTAL TOO GREAT
NO
YES
ERRTOT
INCREMENT DETECTION TOTAL.
SAVE ERROR NUMBER
PRINT
YES
NO
SYS3
SAVE MESSAGE ADDRESS
SYS2
RJ TO WNX DOUBLE SPACE
FETCH ADDRESS OF DIAGNOSTIC MESSAGE
RJ TO WNX PRINT DIAGNOSTIC
GET ERROR NUMBER
RJ TO OCTDEC CONVERT ERROR NUMBER
SAVE + POSITION. ERROR NUMBER
SAVE FIRST LINE. OF MESSAGE
COMBINE ERROR NUMBER WITH 2ND. WORD OF MESSAGE
B

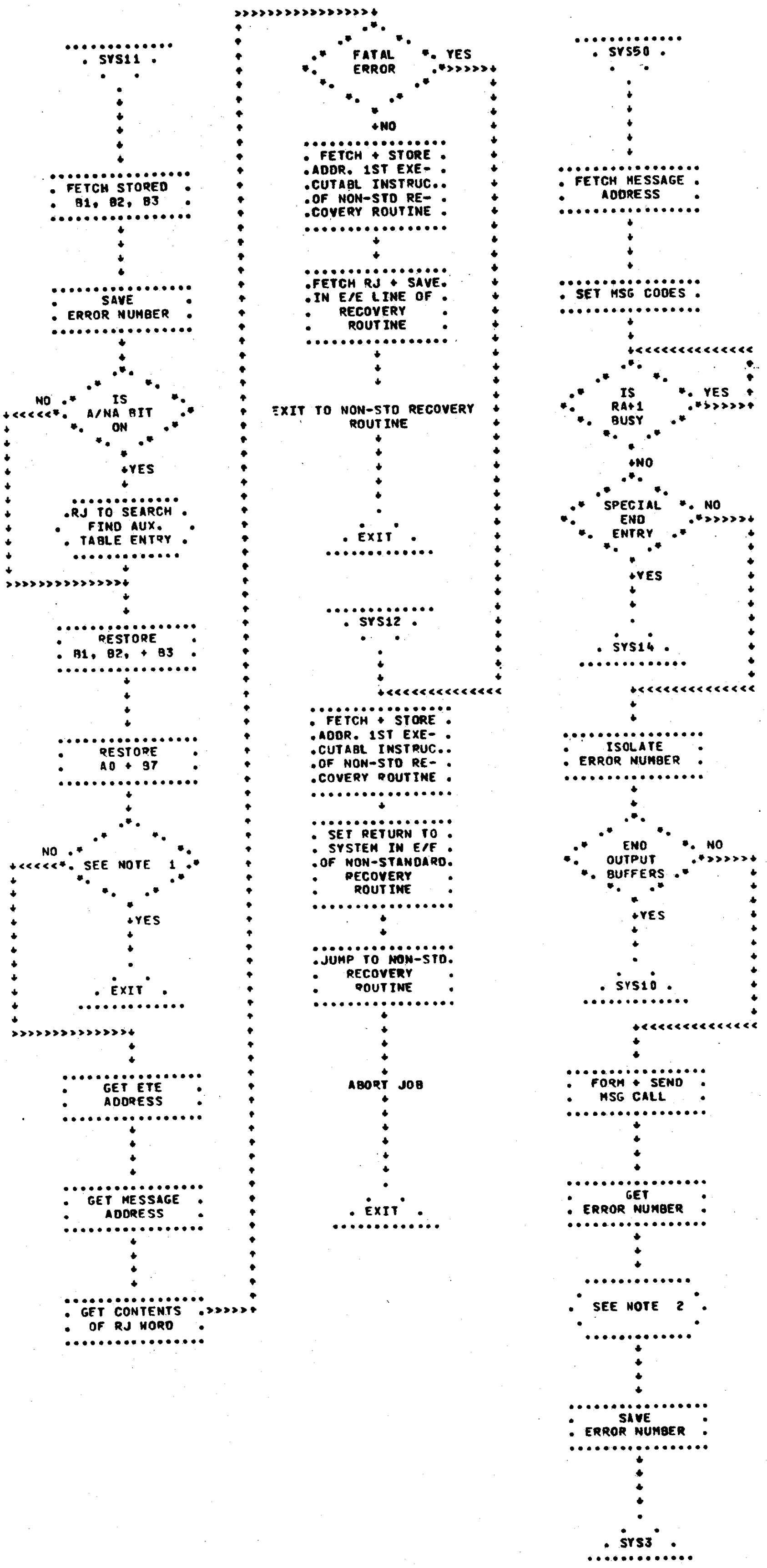
>SYSTEM<

NOTES
  1. NON- STANDARD RECOVERY

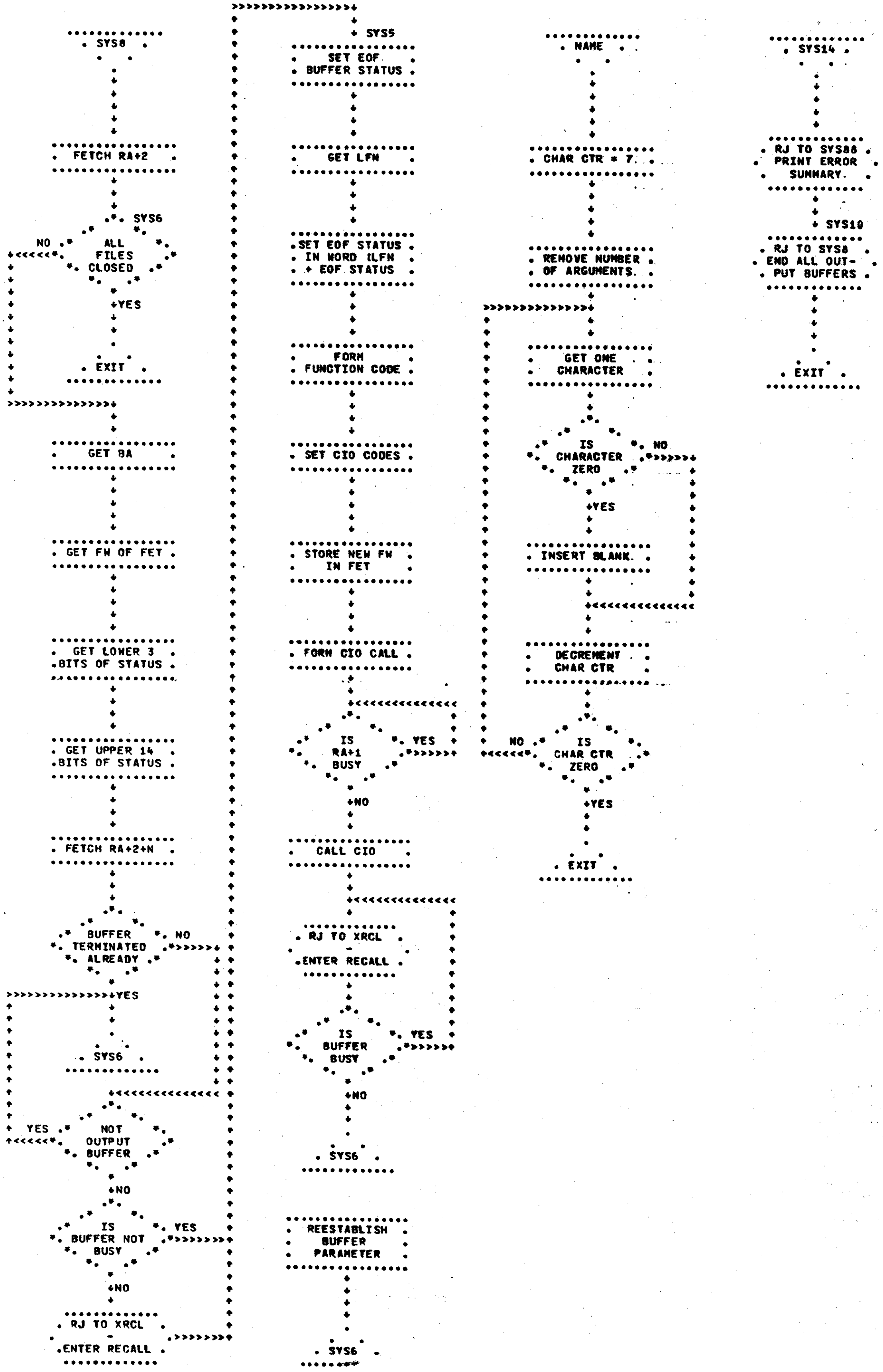

B
SAVE 3RD WORD OF MESSAGE
FETCH CALLING ROUTINE NAME
SAVE ADDRESS OF CALLING ROUTINE NAME
SAVE NUMBER OF ARGUMENTS
RJ TO NAME ADD BLANKS TO NAME
SAVE CONTENTS OF CALLING ROUTINE#S RJ
COMBINE NAME WITH MESSAGE
FETCH AND POSITION RETURN ADDRESS
RJ TO OCTDIS CONVERT ADDRESS
COMBINE ADDRESS WITH MESSAGE
GET ADDRESS OF MESSAGE
RJ TO WNX GO WRITE MESSAGE
CALLING ROUTINE A PROGRAM
YES
NO
FETCH TRACE BACK MAXIMUM
FETCH ADDRESS OF NAME OF CALLING ROUTINE
KWIK
FETCH E/E LINE OF CALLING ROUTINE
FETCH RJ
LEGIT. TRACE BACK SPECIF.
NO
YES
FETCH CALLING ROUTINE NAME
SAVE ADDRESS OF NAME
FETCH FIRST WORD OF MESSAGE
STORE WORD IN MESSAGE BUFFER
RJ TO NAME ADD BLANKS TO NAME
COMBINE NAME WITH MESSAGE AND STORE
RJ TO WNX GO PRINT MESSAGE
TRACE BACK COM- PLETED
YES
NO
NO
CALLING ROUTINE A PROGRAM
YES
SYS3
FETCH ETE
NO
SEE NOTE 1
YES
SYS11
NON FATAL ERROR
YES
NO
PES
RJ TO SYS88 PRINT ERROR SUMMARY
RJ TO SYS8 END ALL OUT- PUT BUFFERS
FETCH LAST ENCOUNTERED ERROR NUMBER
SET MSG CODES
STORE MESSAGE IN MESSAGE BUFFER
IS RA+1 BUSY
YES
NO
SEND MESSAGE
SET ABT CODES
IS RA+1 BUSY
YES
NO
SENT ABORT REQUEST
STOP

>SYSTEM<

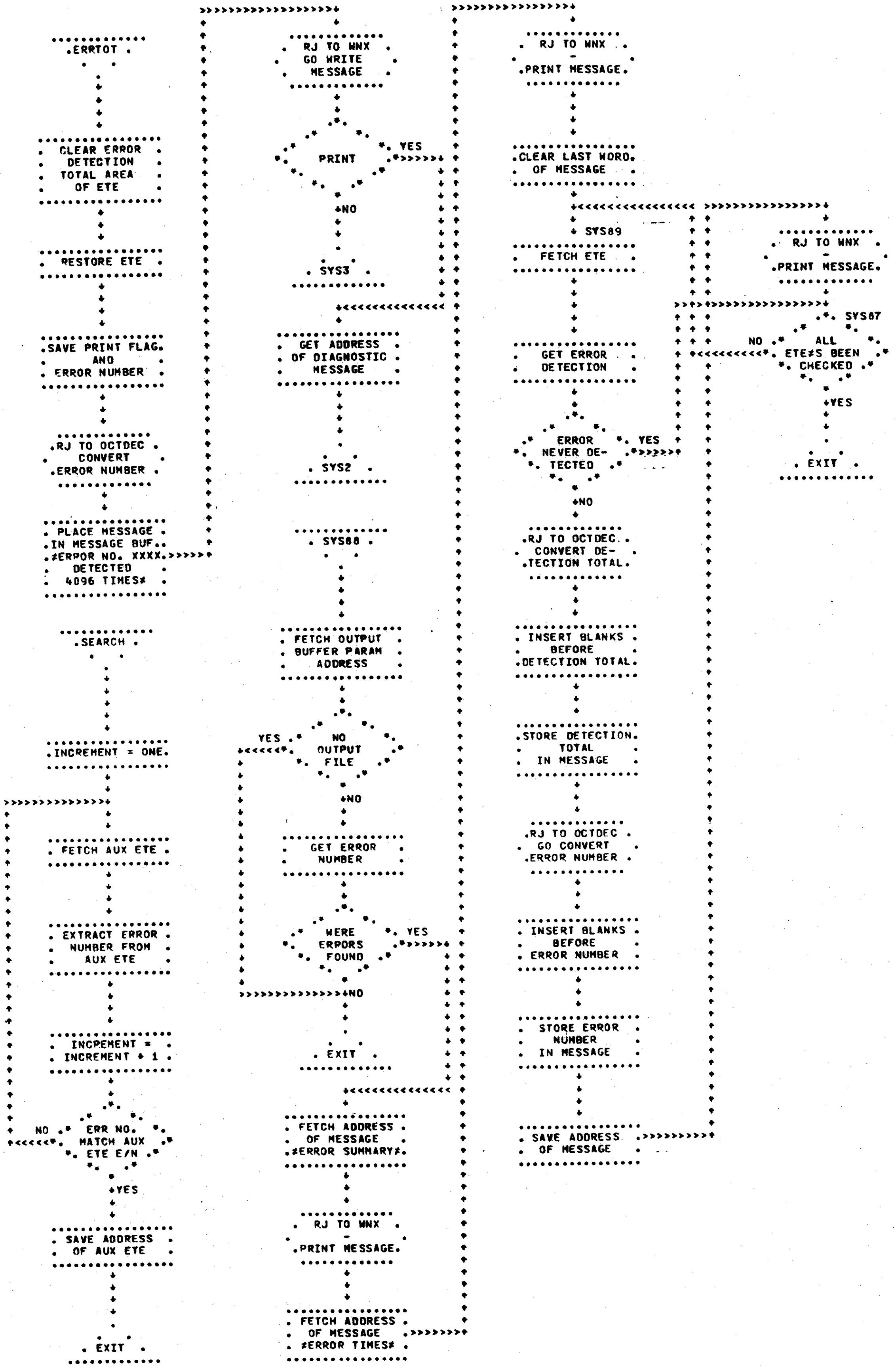
NOTES
1. NON FATAL ERROR W/O NON-STD RECOVERY
2. RJ TO OCTDEC. CONVERT ERROR NUMBER TO DISPLAY CODE

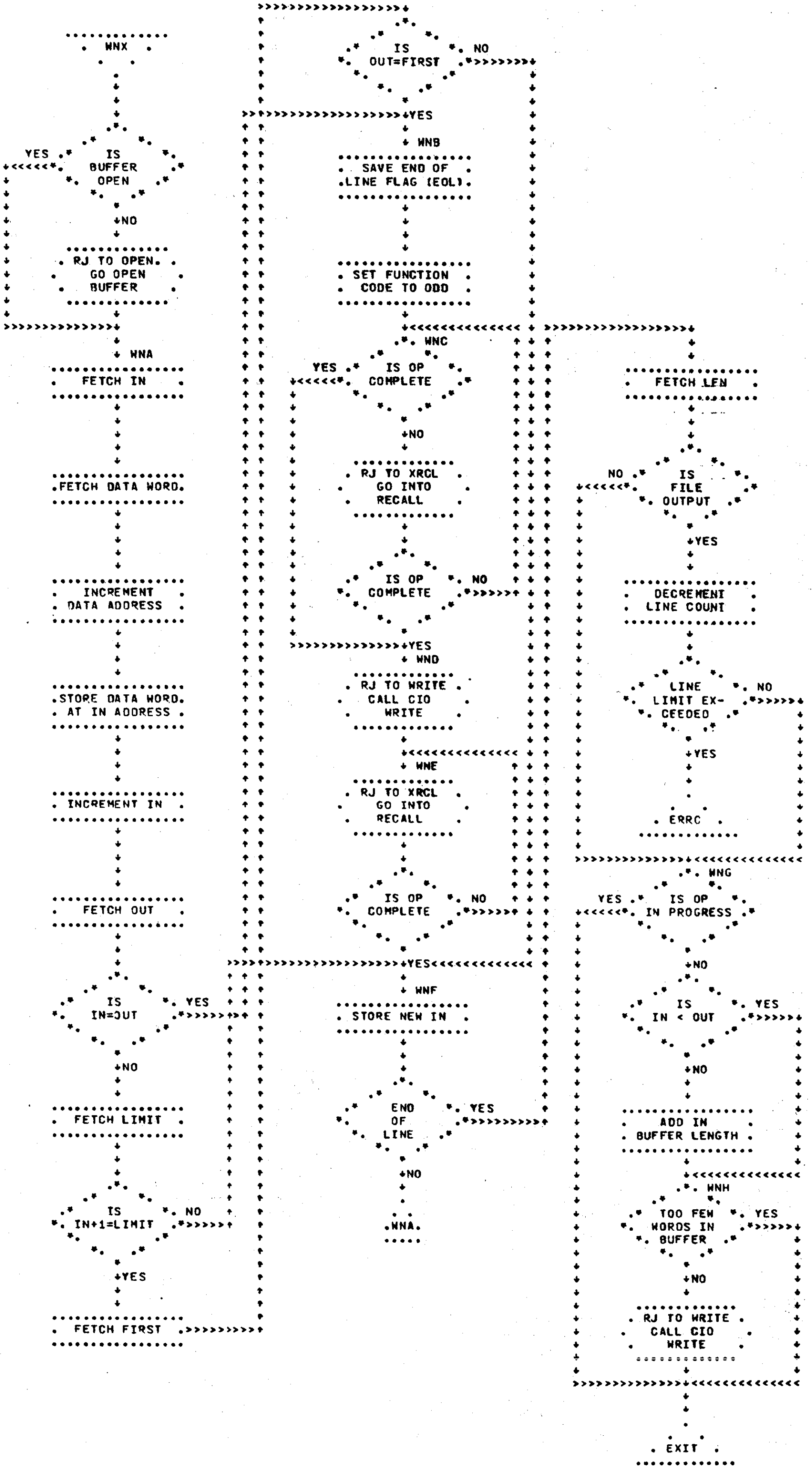


>SYSTEM<

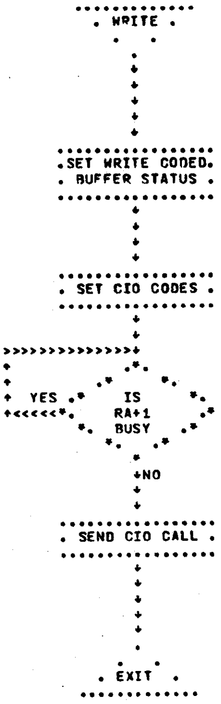

SYS8
FETCH RA+2
SYS6
ALL FILES CLOSED
NO
YES
EXIT
GET BA
GET FW OF FET
GET LOWER 3 BITS OF STATUS
GET UPPER 14 BITS OF STATUS
FETCH RA+2+N
BUFFER TERMINATED ALREADY
NO
YES
SYS6
NOT OUTPUT BUFFER
YES
NO
IS BUFFER NOT BUSY
YES
NO
RJ TO XRCL - ENTER RECALL
SYS5
SET EOF BUFFER STATUS
GET LFN
SET EOF STATUS IN WORD ILFN + EOF STATUS
FORM FUNCTION CODE
SET CIO CODES
STORE NEW FW IN FET
FORM CIO CALL
IS RA+1 BUSY
YES
NO
CALL CIO
RJ TO XRCL - ENTER RECALL
IS BUFFER BUSY
YES
NO
SYS6
REESTABLISH BUFFER PARAMETER
SYS6
NAME
CHAR CTR = 7.
REMOVE NUMBER OF ARGUMENTS.
GET ONE CHARACTER
IS CHARACTER ZERO
NO
YES
INSERT BLANK.
DECREMENT CHAR CTR
IS CHAR CTR ZERO
NO
YES
EXIT
SYS14
RJ TO SYS88 PRINT ERROR SUMMARY.
SYS10
RJ TO SYS8 END ALL OUT-PUT BUFFERS
EXIT

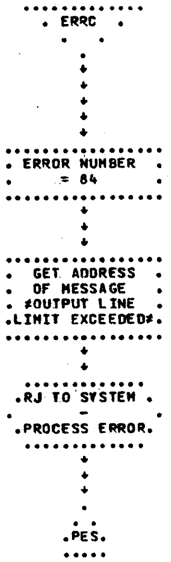
>SYSTEM<

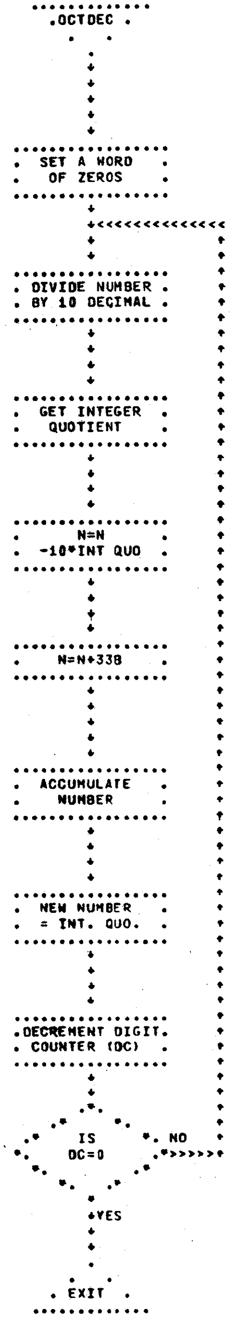

ERRTOT
CLEAR ERROR DETECTION TOTAL AREA OF ETE
RESTORE ETE
SAVE PRINT FLAG AND ERROR NUMBER
RJ TO OCTDEC CONVERT ERROR NUMBER
PLACE MESSAGE IN MESSAGE BUF. ≠ERROR NO. XXXX DETECTED 4096 TIMES≠
SEARCH
INCREMENT = ONE.
FETCH AUX ETE
EXTRACT ERROR NUMBER FROM AUX ETE
INCREMENT = INCREMENT + 1
NO
ERR NO. MATCH AUX ETE E/N
YES
SAVE ADDRESS OF AUX ETE
EXIT
RJ TO WNX GO WRITE MESSAGE
PRINT
YES
NO
SYS3
GET ADDRESS OF DIAGNOSTIC MESSAGE
SYS2
SYS88
FETCH OUTPUT BUFFER PARAM ADDRESS
YES
NO OUTPUT FILE
NO
GET ERROR NUMBER
WERE ERRORS FOUND
YES
NO
EXIT
FETCH ADDRESS OF MESSAGE ≠ERROR SUMMARY≠
RJ TO WNX - PRINT MESSAGE.
FETCH ADDRESS OF MESSAGE ≠ERROR TIMES≠
RJ TO WNX - PRINT MESSAGE.
CLEAR LAST WORD OF MESSAGE
SYS89
FETCH ETE
GET ERROR DETECTION
ERROR NEVER DE- TECTED
YES
NO
RJ TO OCTDEC CONVERT DE- TECTION TOTAL.
INSERT BLANKS BEFORE DETECTION TOTAL.
STORE DETECTION TOTAL IN MESSAGE
RJ TO OCTDEC GO CONVERT ERROR NUMBER
INSERT BLANKS BEFORE ERROR NUMBER
STORE ERROR NUMBER IN MESSAGE
SAVE ADDRESS OF MESSAGE
RJ TO WNX - PRINT MESSAGE.
SYS87
NO
ALL ETE≠S BEEN CHECKED
YES
EXIT

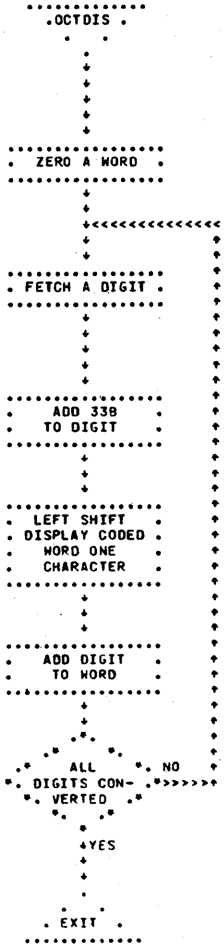

WNX
IS BUFFER OPEN
YES
NO
RJ TO OPEN. GO OPEN BUFFER
WNA
FETCH IN
FETCH DATA WORD.
INCREMENT DATA ADDRESS
STORE DATA WORD. AT IN ADDRESS
INCREMENT IN
FETCH OUT
IS IN=OUT
YES
NO
FETCH LIMIT
IS IN+1=LIMIT
NO
YES
FETCH FIRST
IS OUT=FIRST
NO
YES
WNB
SAVE END OF LINE FLAG (EOL).
SET FUNCTION CODE TO ODD
WNC
IS OP COMPLETE
YES
NO
RJ TO XRCL GO INTO RECALL
IS OP COMPLETE
NO
YES
WND
RJ TO WRITE CALL CIO WRITE
WNE
RJ TO XRCL GO INTO RECALL
IS OP COMPLETE
NO
YES
WNF
STORE NEW IN
END OF LINE
YES
NO
.WNA.
FETCH LEN
IS FILE OUTPUT
NO
YES
DECREMENT LINE COUNT
LINE LIMIT EX- CEEDED
NO
YES
ERRC
WNG
IS OP IN PROGRESS
YES
NO
IS IN < OUT
YES
NO
ADD IN BUFFER LENGTH
WNH
TOO FEW WORDS IN BUFFER
YES
NO
RJ TO WRITE CALL CIO WRITE
EXIT

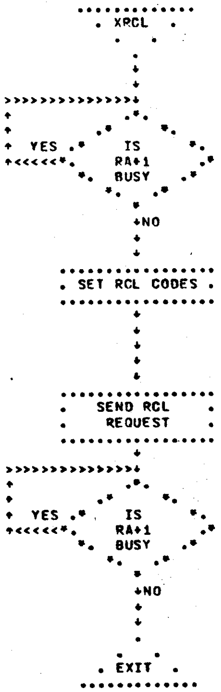
>SYSTEM<


WRITE
SET WRITE CODED BUFFER STATUS
SET CIO CODES
IS RA+1 BUSY
YES
NO
SEND CIO CALL
EXIT


ERRC
ERROR NUMBER = 84
GET ADDRESS OF MESSAGE ≠OUTPUT LINE LIMIT EXCEEDED≠
RJ TO SYSTEM - PROCESS ERROR
PES.

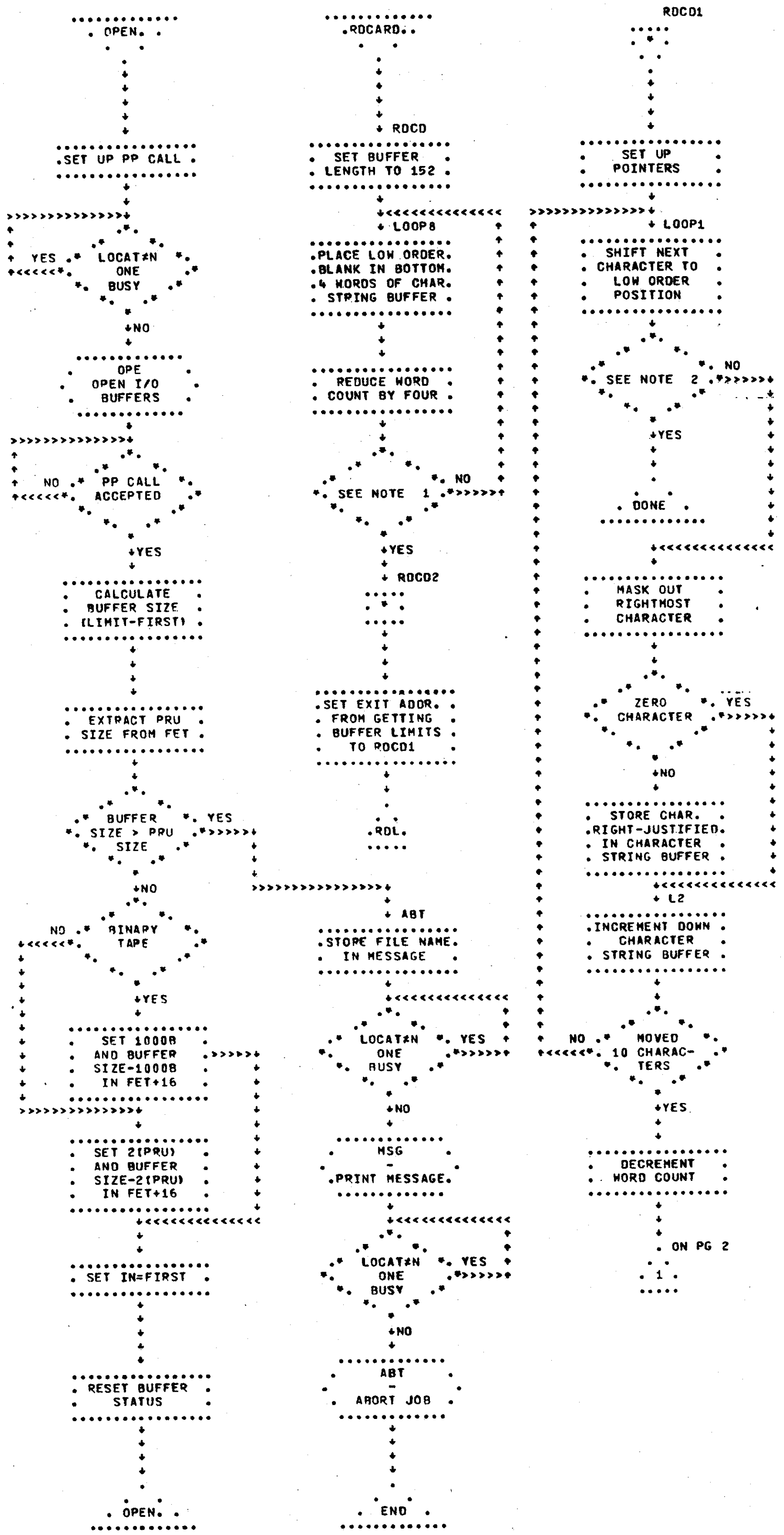

OCTDEC
SET A WORD OF ZEROS
DIVIDE NUMBER BY 10 DECIMAL
GET INTEGER QUOTIENT
N=N -10*INT QUO
N=N+33B
ACCUMULATE NUMBER
NEW NUMBER = INT. QUO.
DECREMENT DIGIT COUNTER (DC)
IS DC=0
NO
YES
EXIT

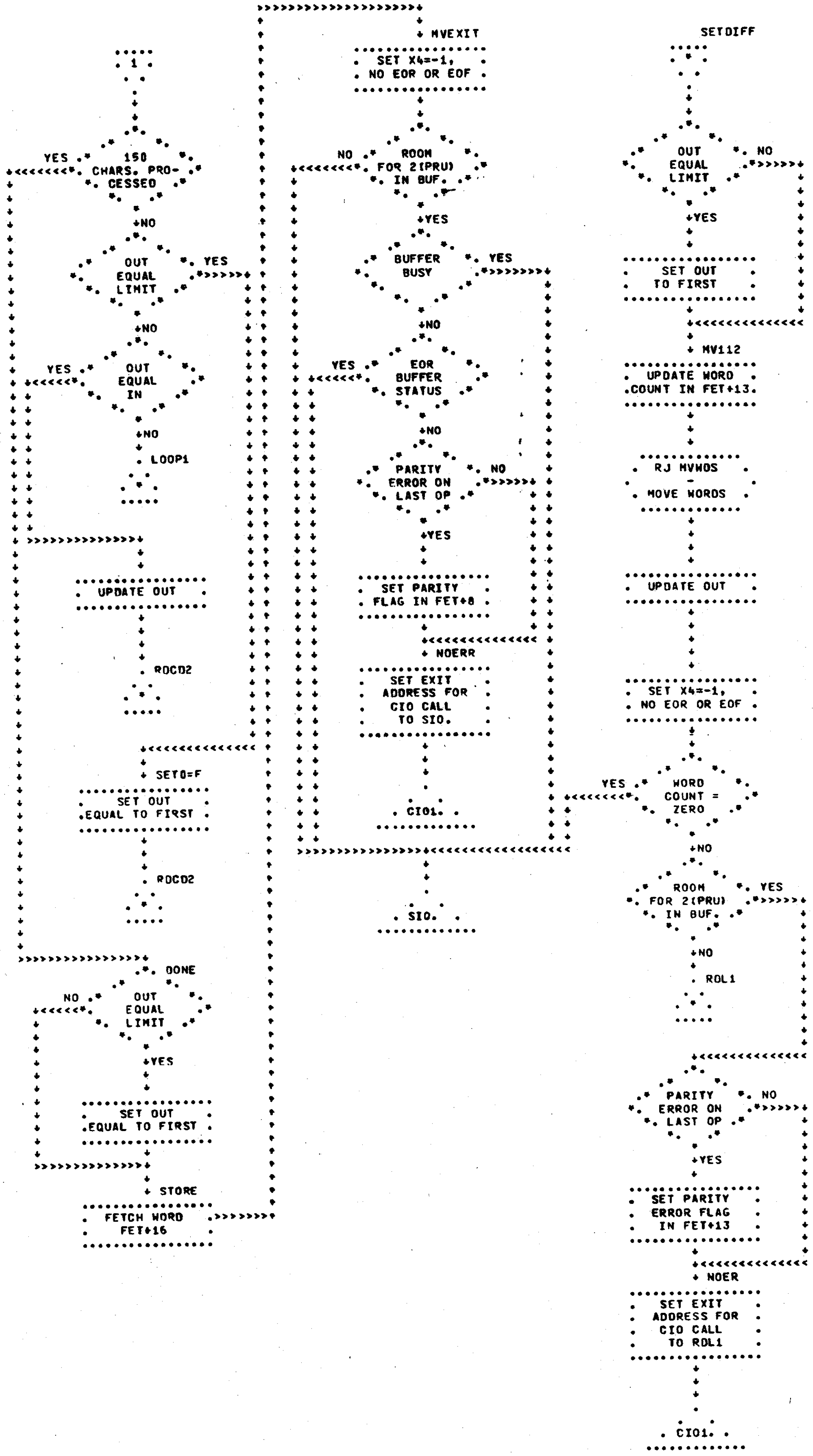

OCTDIS
ZERO A WORD
FETCH A DIGIT
ADD 33B TO DIGIT
LEFT SHIFT DISPLAY CODED WORD ONE CHARACTER
ADD DIGIT TO WORD
ALL DIGITS CON- VERTED
NO
YES
EXIT


XRCL
IS RA+1 BUSY
YES
NO
SET RCL CODES
SEND RCL REQUEST
IS RA+1 BUSY
YES
NO
EXIT

>SIO$<

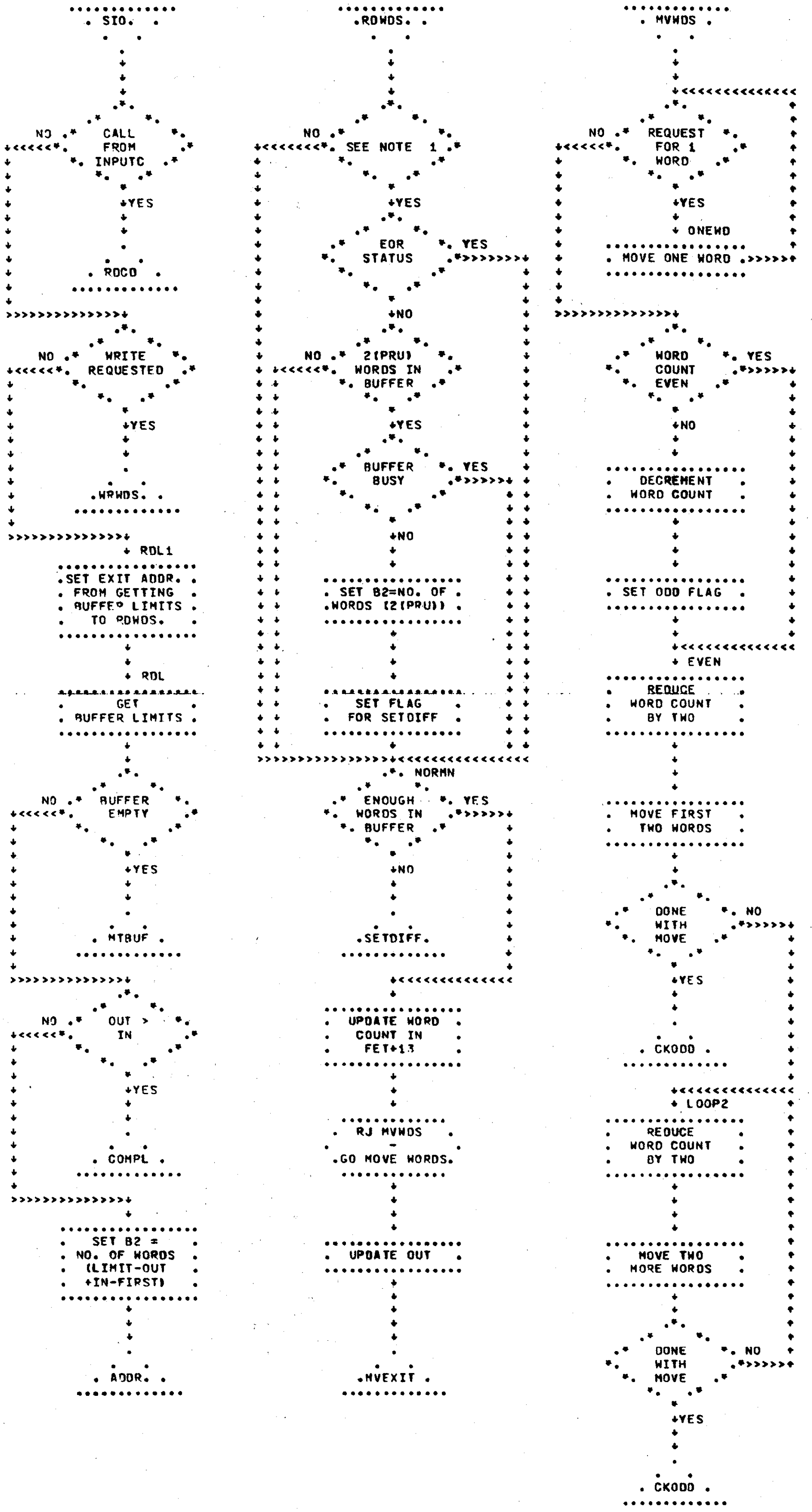
NOTES
  1. FILLED CHAR. STRING BUF. WITH BLANKS
  2. FOUND RIGHT-JUSTIFIED LINE TERMINATOR


OPEN.
SET UP PP CALL
YES
LOCAT≠N ONE BUSY
NO
OPE
OPEN I/O BUFFERS
NO
PP CALL ACCEPTED
YES
CALCULATE BUFFER SIZE (LIMIT-FIRST)
EXTRACT PRU SIZE FROM FET
BUFFER SIZE > PRU SIZE
YES
NO
NO
BINARY TAPE
YES
SET 1000B AND BUFFER SIZE-1000B IN FET+16
SET 2(PRU) AND BUFFER SIZE-2(PRU) IN FET+16
SET IN=FIRST
RESET BUFFER STATUS
OPEN.
RDCARD.
RDCD
SET BUFFER LENGTH TO 152
LOOP8
PLACE LOW ORDER. BLANK IN BOTTOM. 4 WORDS OF CHAR. STRING BUFFER
REDUCE WORD COUNT BY FOUR
SEE NOTE 1
NO
YES
RDCD2
SET EXIT ADDR. FROM GETTING BUFFER LIMITS TO RDCD1
RDL.
ABT
STORE FILE NAME. IN MESSAGE
LOCAT≠N ONE BUSY
YES
NO
MSG
-
PRINT MESSAGE.
LOCAT≠N ONE BUSY
YES
NO
ABT
-
ABORT JOB
END
RDCD1
SET UP POINTERS
LOOP1
SHIFT NEXT CHARACTER TO LOW ORDER POSITION
SEE NOTE 2
NO
YES
DONE
MASK OUT RIGHTMOST CHARACTER
ZERO CHARACTER
YES
NO
STORE CHAR. RIGHT-JUSTIFIED. IN CHARACTER STRING BUFFER
L2
INCREMENT DOWN CHARACTER STRING BUFFER
NO
MOVED 10 CHARAC-TERS
YES
DECREMENT WORD COUNT
ON PG 2
1

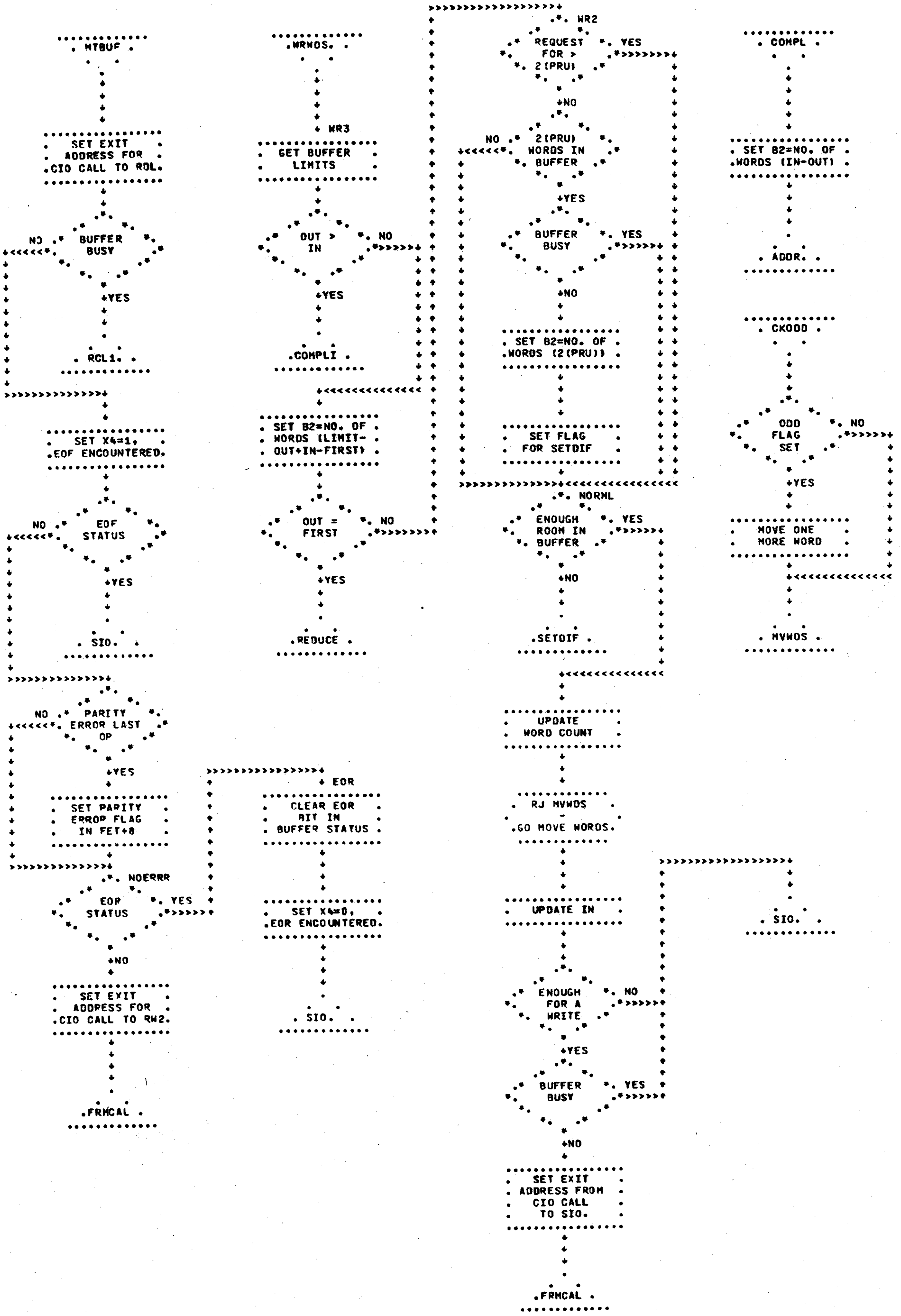

1
150 CHARS. PROCESSED
YES
NO
OUT EQUAL LIMIT
YES
NO
OUT EQUAL IN
YES
NO
LOOP1
UPDATE OUT
RDCD2
SETO=F
SET OUT EQUAL TO FIRST
RDCD2
DONE
OUT EQUAL LIMIT
NO
YES
SET OUT EQUAL TO FIRST
STORE
FETCH WORD FET+16
MVEXIT
SET X4=-1, NO EOR OR EOF
ROOM FOR 2(PRU) IN BUF.
NO
YES
BUFFER BUSY
YES
NO
EOR BUFFER STATUS
YES
NO
PARITY ERROR ON LAST OP
NO
YES
SET PARITY FLAG IN FET+8
NOERR
SET EXIT ADDRESS FOR CIO CALL TO SIO.
CIO1.
SIO.
SETDIFF
OUT EQUAL LIMIT
NO
YES
SET OUT TO FIRST
MV112
UPDATE WORD COUNT IN FET+13.
RJ MVWDS
MOVE WORDS
UPDATE OUT
SET X4=-1, NO EOR OR EOF
WORD COUNT = ZERO
YES
NO
ROOM FOR 2(PRU) IN BUF.
YES
NO
RDL1
PARITY ERROR ON LAST OP
NO
YES
SET PARITY ERROR FLAG IN FET+13
NOER
SET EXIT ADDRESS FOR CIO CALL TO RDL1
CIO1.

>SIOS<

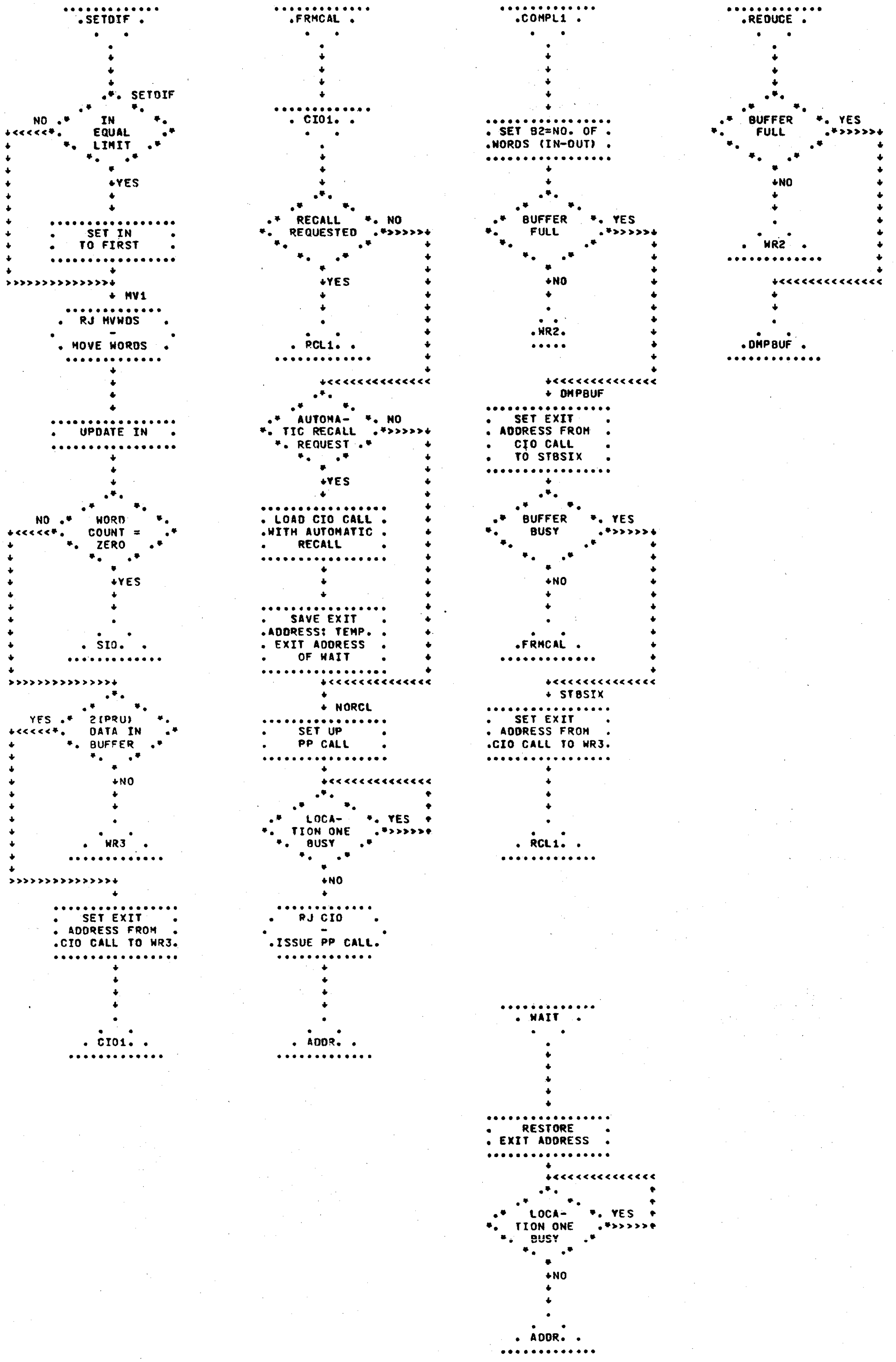
NOTES
   1. WANT TO MOVE MORE THAN 2(PRU) WORDS



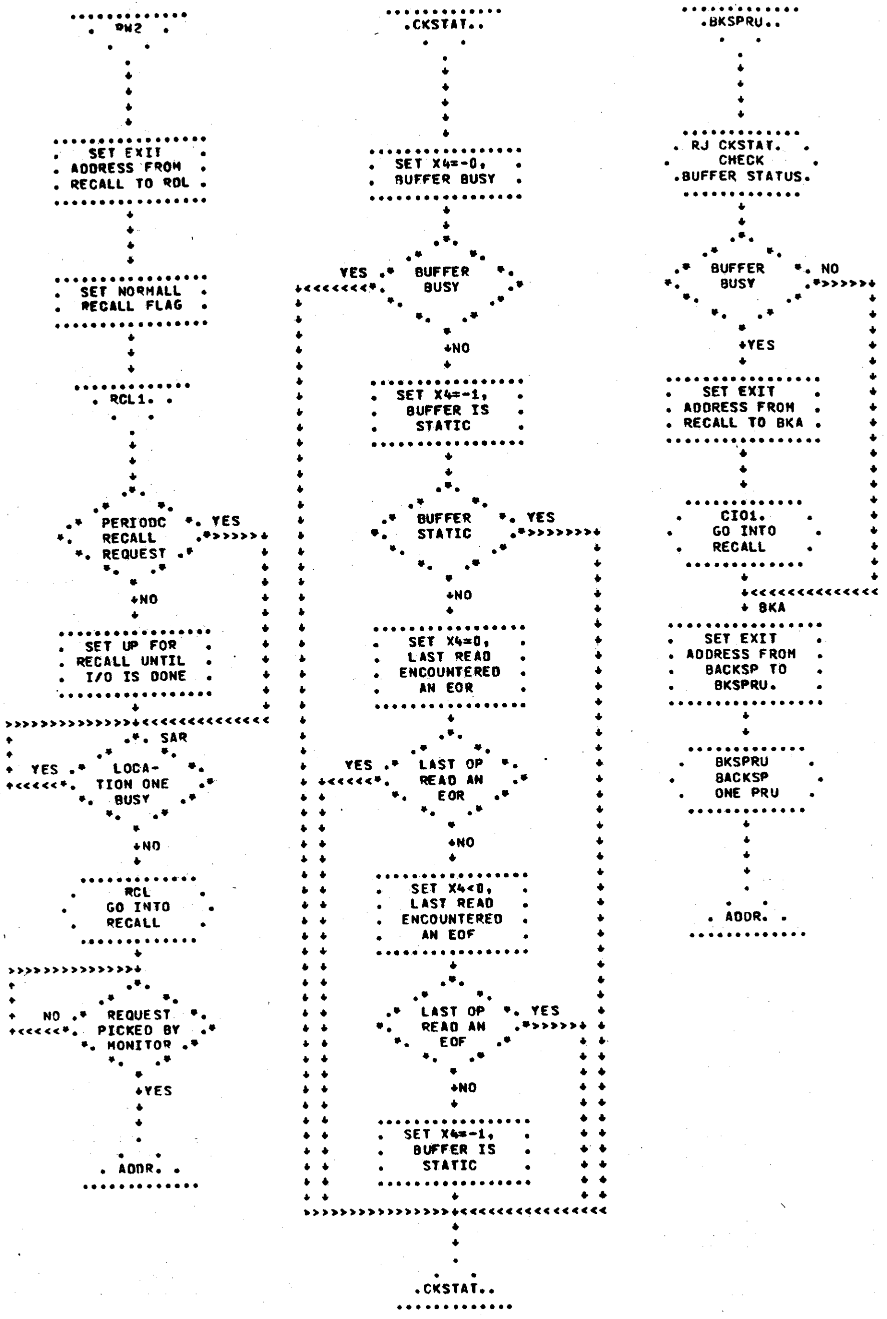
>SIO$<

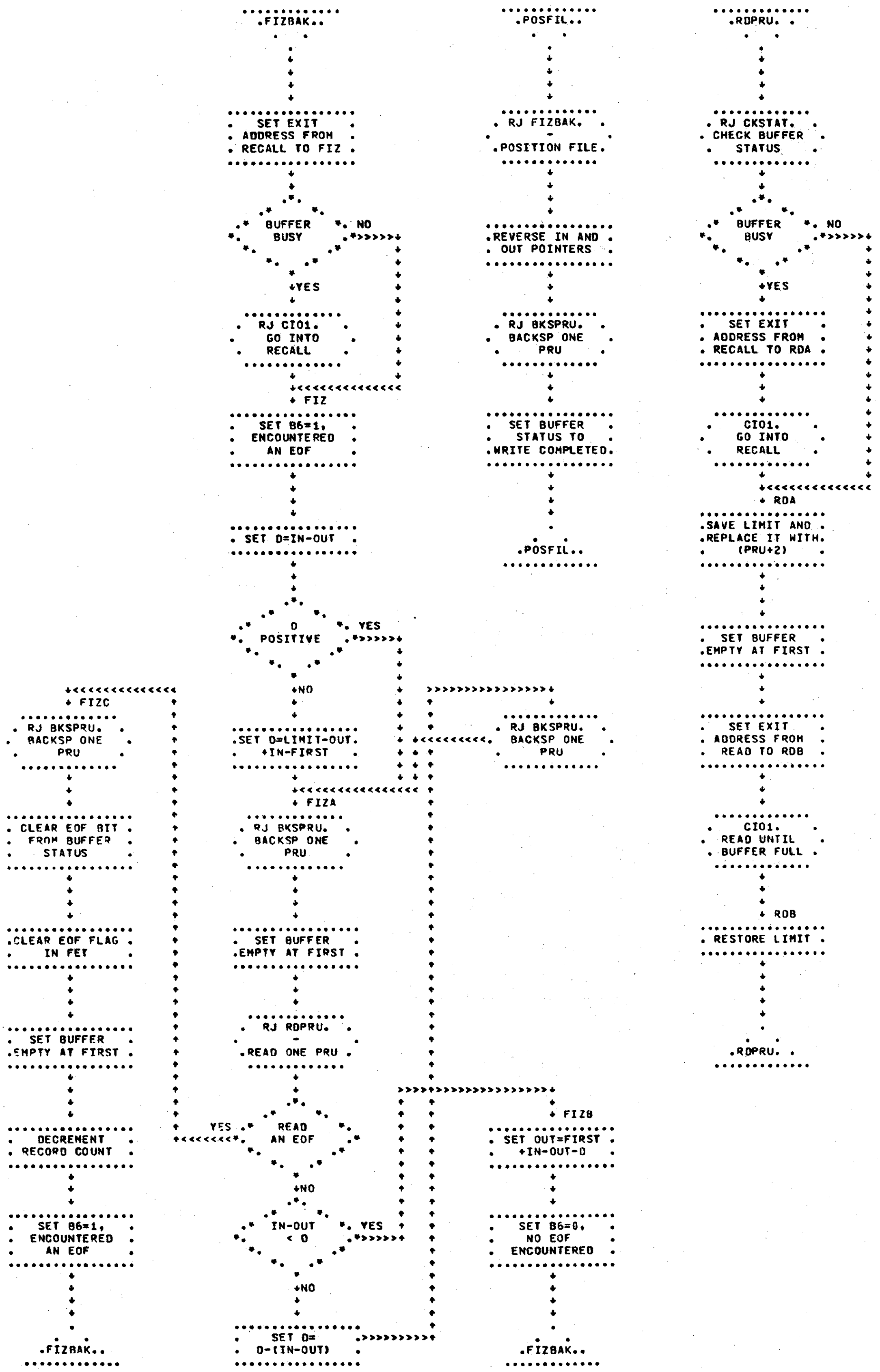

WTBUF
SET EXIT ADDRESS FOR CIO CALL TO RCL.
BUFFER BUSY
NO
YES
RCL1.
SET X4=1. EOF ENCOUNTERED.
EOF STATUS
NO
YES
SIO.
PARITY ERROR LAST OP
NO
YES
SET PARITY ERROR FLAG IN FET+8
NOERRR
EOR STATUS
YES
NO
SET EXIT ADDRESS FOR CIO CALL TO RW2.
FRMCAL
WRWDS.
WR3
GET BUFFER LIMITS
OUT > IN
NO
YES
COMPLI
SET B2=NO. OF WORDS (LIMIT-OUT+IN-FIRST)
OUT = FIRST
NO
YES
REDUCE
EOR
CLEAR EOR BIT IN BUFFER STATUS
SET X4=0. EOR ENCOUNTERED.
SIO.
WR2
REQUEST FOR > 2(PRU)
YES
NO
2(PRU) WORDS IN BUFFER
NO
YES
BUFFER BUSY
YES
NO
SET B2=NO. OF WORDS (2(PRU))
SET FLAG FOR SETDIF
NORML
ENOUGH ROOM IN BUFFER
YES
NO
SETDIF
UPDATE WORD COUNT
RJ MVWDS - GO MOVE WORDS.
UPDATE IN
ENOUGH FOR A WRITE
NO
YES
BUFFER BUSY
YES
NO
SET EXIT ADDRESS FROM CIO CALL TO SIO.
FRMCAL
SIO.
COMPL
SET B2=NO. OF WORDS (IN-OUT)
ADDR.
CKODD
ODD FLAG SET
NO
YES
MOVE ONE MORE WORD
MVWDS

>SIO3<


SETDIF
SETDIF
IN EQUAL LIMIT
NO
YES
SET IN TO FIRST
MV1
RJ MVWDS - MOVE WORDS
UPDATE IN
WORD COUNT = ZERO
NO
YES
SIO.
2(PRU) DATA IN BUFFER
YES
NO
WR3
SET EXIT ADDRESS FROM CIO CALL TO WR3.
CIO1.
FRMCAL
CIO1.
RECALL REQUESTED
NO
YES
RCL1.
AUTOMA- TIC RECALL REQUEST
NO
YES
LOAD CIO CALL WITH AUTOMATIC RECALL
SAVE EXIT ADDRESS: TEMP. EXIT ADDRESS OF WAIT
NORCL
SET UP PP CALL
LOCA- TION ONE BUSY
YES
NO
RJ CIO - ISSUE PP CALL.
ADDR.
COMPL1
SET B2=NO. OF WORDS (IN-OUT)
BUFFER FULL
YES
NO
WR2.
DMPBUF
SET EXIT ADDRESS FROM CIO CALL TO STBSIX
BUFFER BUSY
YES
NO
FRMCAL
STBSIX
SET EXIT ADDRESS FROM CIO CALL TO WR3.
RCL1.
WAIT
RESTORE EXIT ADDRESS
LOCA- TION ONE BUSY
YES
NO
ADDR.
REDUCE
BUFFER FULL
YES
NO
WR2
DMPBUF

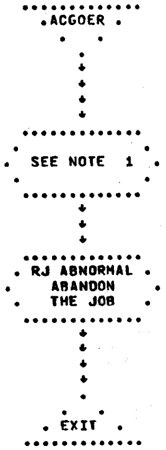
>CIO1<


RW2
SET EXIT ADDRESS FROM RECALL TO RDL
SET NORMALL RECALL FLAG
RCL1.
PERIODC RECALL REQUEST
YES
NO
SET UP FOR RECALL UNTIL I/O IS DONE
SAR
LOCA-TION ONE BUSY
YES
NO
RCL GO INTO RECALL
REQUEST PICKED BY MONITOR
NO
YES
ADDR.
CKSTAT.
SET X4=-0, BUFFER BUSY
BUFFER BUSY
YES
NO
SET X4=-1, BUFFER IS STATIC
BUFFER STATIC
YES
NO
SET X4=0, LAST READ ENCOUNTERED AN EOR
LAST OP READ AN EOR
YES
NO
SET X4<0, LAST READ ENCOUNTERED AN EOF
LAST OP READ AN EOF
YES
NO
SET X4=-1, BUFFER IS STATIC
CKSTAT.
BKSPRU.
RJ CKSTAT. CHECK BUFFER STATUS.
BUFFER BUSY
NO
YES
SET EXIT ADDRESS FROM RECALL TO BKA
CIO1. GO INTO RECALL
BKA
SET EXIT ADDRESS FROM BACKSP TO BKSPRU.
BKSPRU BACKSP ONE PRU
ADDR.

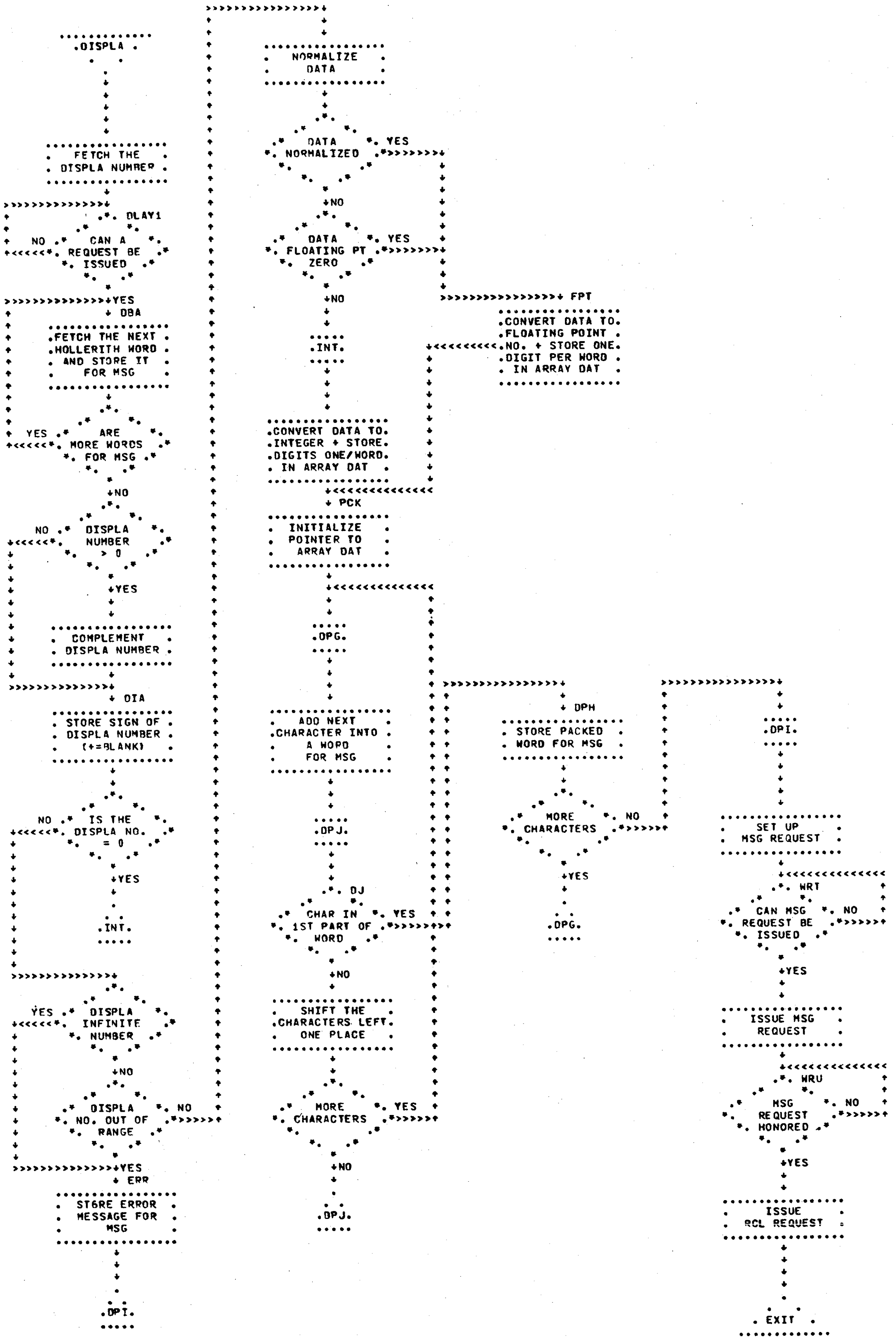

FIZBAK.
SET EXIT ADDRESS FROM RECALL TO FIZ
BUFFER BUSY
NO
YES
RJ CIO1. GO INTO RECALL
FIZ
SET B6=1, ENCOUNTERED AN EOF
SET D=IN-OUT
D POSITIVE
YES
NO
SET D=LIMIT-OUT +IN-FIRST
FIZA
RJ BKSPRU. BACKSP ONE PRU
SET BUFFER EMPTY AT FIRST
RJ RDPRU. - READ ONE PRU
READ AN EOF
YES
NO
IN-OUT < 0
YES
NO
SET D= D-(IN-OUT)
FIZC
RJ BKSPRU. BACKSP ONE PRU
CLEAR EOF BIT FROM BUFFER STATUS
CLEAR EOF FLAG IN FET
SET BUFFER EMPTY AT FIRST
DECREMENT RECORD COUNT
SET B6=1, ENCOUNTERED AN EOF
FIZBAK.
RJ BKSPRU. BACKSP ONE PRU
FIZB
SET OUT=FIRST +IN-OUT-D
SET B6=0, NO EOF ENCOUNTERED
FIZBAK.
POSFIL.
RJ FIZBAK. - POSITION FILE.
REVERSE IN AND OUT POINTERS
RJ BKSPRU. BACKSP ONE PRU
SET BUFFER STATUS TO WRITE COMPLETED.
POSFIL.
RDPRU.
RJ CKSTAT. CHECK BUFFER STATUS
BUFFER BUSY
NO
YES
SET EXIT ADDRESS FROM RECALL TO RDA
CIO1. GO INTO RECALL
RDA
SAVE LIMIT AND REPLACE IT WITH (PRU+2)
SET BUFFER EMPTY AT FIRST
SET EXIT ADDRESS FROM READ TO RDB
CIO1. READ UNTIL BUFFER FULL
RDB
RESTORE LIMIT
RDPRU.

>ACGOER<

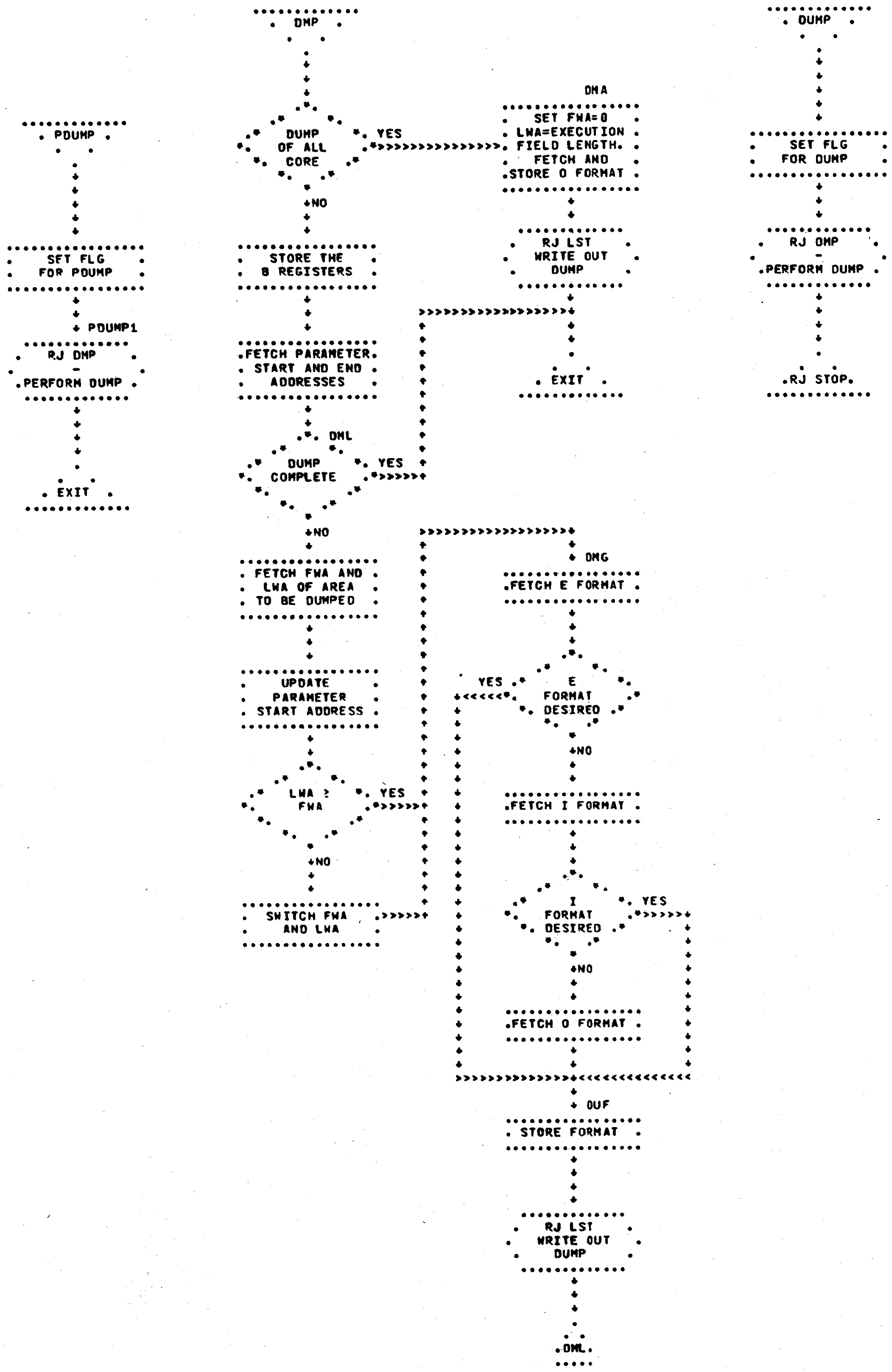
NOTES
   1. RJ SYSTEM ≠ERROR, COMPUTED OR ASSIGNED GO TO STATEMENT≠



>DISPLA<

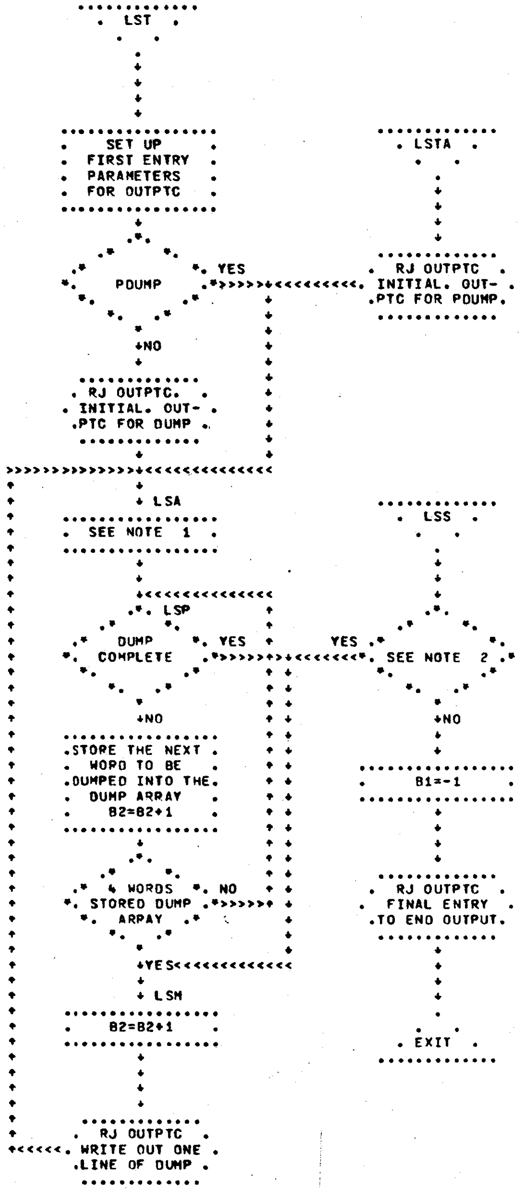

DISPLA
FETCH THE DISPLA NUMBER
DLAY1
CAN A REQUEST BE ISSUED
NO
YES
DBA
FETCH THE NEXT HOLLERITH WORD AND STORE IT FOR MSG
ARE MORE WORDS FOR MSG
YES
NO
DISPLA NUMBER > 0
NO
YES
COMPLEMENT DISPLA NUMBER
DIA
STORE SIGN OF DISPLA NUMBER (+=BLANK)
IS THE DISPLA NO. = 0
NO
YES
INT.
DISPLA INFINITE NUMBER
YES
NO
DISPLA NO. OUT OF RANGE
NO
YES
ERR
ST6RE ERROR MESSAGE FOR MSG
DPI.
NORMALIZE DATA
DATA NORMALIZED
YES
NO
DATA FLOATING PT ZERO
YES
NO
INT.
CONVERT DATA TO. INTEGER + STORE. DIGITS ONE/WORD. IN ARRAY DAT
FPT
CONVERT DATA TO. FLOATING POINT NO. + STORE ONE. DIGIT PER WORD IN ARRAY DAT
PCK
INITIALIZE POINTER TO ARRAY DAT
DPG.
ADD NEXT CHARACTER INTO A WORD FOR MSG
DPJ.
DJ
CHAR IN 1ST PART OF WORD
YES
NO
SHIFT THE CHARACTERS LEFT ONE PLACE
MORE CHARACTERS
YES
NO
DPJ.
DPH
STORE PACKED WORD FOR MSG
MORE CHARACTERS
NO
YES
DPG.
DPI.
SET UP MSG REQUEST
WRT
CAN MSG REQUEST BE ISSUED
NO
YES
ISSUE MSG REQUEST
WRU
MSG REQUEST HONORED
NO
YES
ISSUE RCL REQUEST
EXIT

>DUMP<

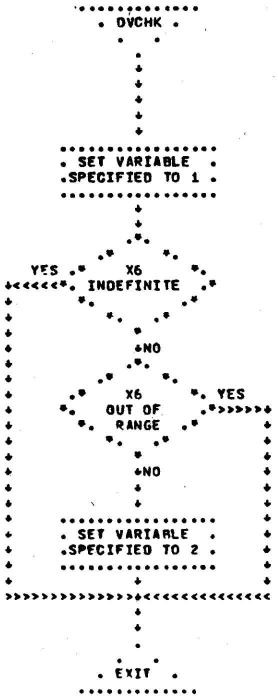

PDUMP
SET FLG FOR PDUMP
PDUMP1
RJ DMP - PERFORM DUMP
EXIT
DMP
DUMP OF ALL CORE
YES
NO
STORE THE B REGISTERS
FETCH PARAMETER. START AND END ADDRESSES
DML
DUMP COMPLETE
YES
NO
FETCH FWA AND LWA OF AREA TO BE DUMPED
UPDATE PARAMETER START ADDRESS
LWA ≥ FWA
YES
NO
SWITCH FWA AND LWA
DMA
SET FWA=0 LWA=EXECUTION FIELD LENGTH. FETCH AND STORE O FORMAT
RJ LST WRITE OUT DUMP
EXIT
DMG
FETCH E FORMAT
E FORMAT DESIRED
YES
NO
FETCH I FORMAT
I FORMAT DESIRED
YES
NO
FETCH O FORMAT
DUF
STORE FORMAT
RJ LST WRITE OUT DUMP
DML.
DUMP
SET FLG FOR DUMP
RJ DMP - PERFORM DUMP
RJ STOP.

>DUMP<

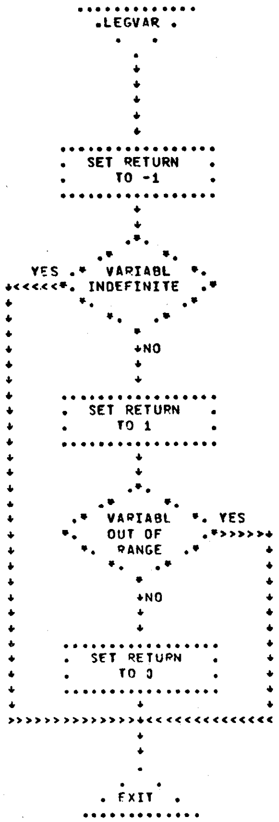
NOTES
1. SET B1=BEGIN ADDR DUMP ARRAY.STORE FWA OF DUMP IN ARRAY + UPDATE.SET B2=0
2. DOES INTERMEDIATE ENTRY TO OUTPTC HAVE TO BE MADE


LST
SET UP FIRST ENTRY PARAMETERS FOR OUTPTC
LSTA
PDUMP
YES
RJ OUTPTC INITIAL. OUT-PTC FOR PDUMP.
NO
RJ OUTPTC. INITIAL. OUT-PTC FOR DUMP .
LSA
SEE NOTE 1
LSS
LSP
DUMP COMPLETE
YES
YES
SEE NOTE 2
NO
NO
STORE THE NEXT WORD TO BE DUMPED INTO THE DUMP ARRAY B2=B2+1
B1=-1
4 WORDS STORED DUMP ARRAY
NO
RJ OUTPTC FINAL ENTRY TO END OUTPUT.
YES
LSM
B2=B2+1
EXIT
RJ OUTPTC WRITE OUT ONE LINE OF DUMP

>DVCHK<


DVCHK
SET VARIABLE SPECIFIED TO 1
YES
X6 INDEFINITE
NO
X6 OUT OF RANGE
YES
NO
SET VARIABLE SPECIFIED TO 2
EXIT

>LEGVAR<

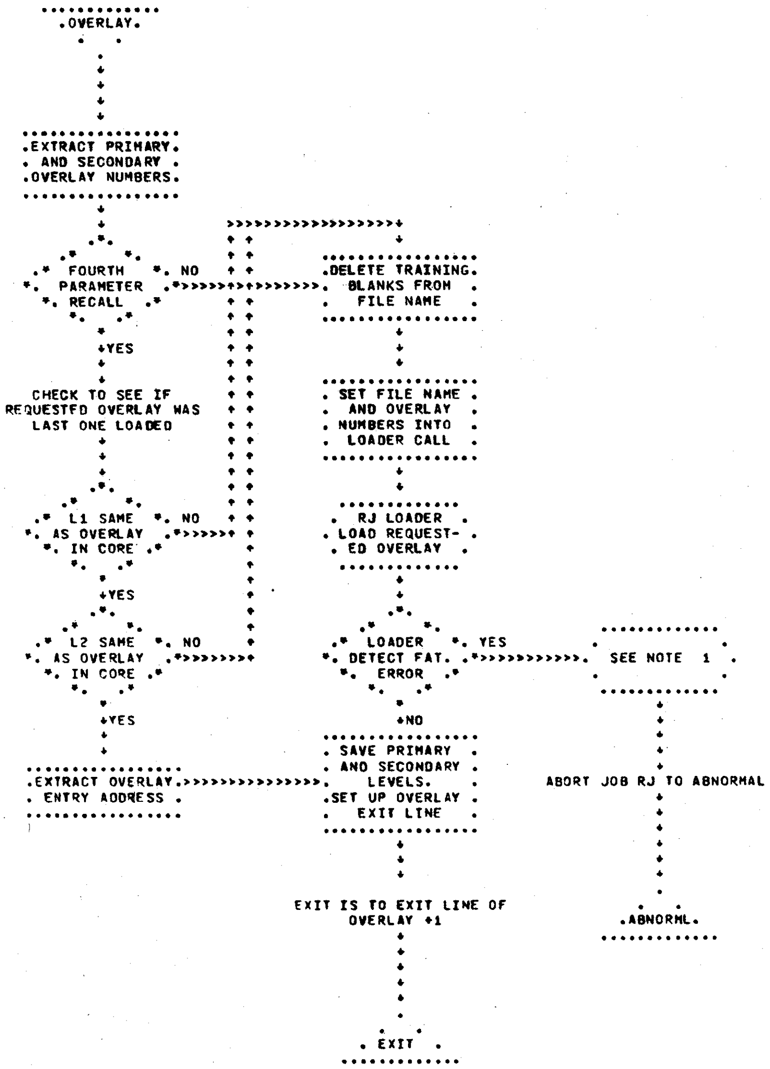

LEGVAR
SET RETURN TO -1
VARIABL INDEFINITE
YES
NO
SET RETURN TO 1
VARIABL OUT OF RANGE
YES
NO
SET RETURN TO 0
EXIT

>LOCF<

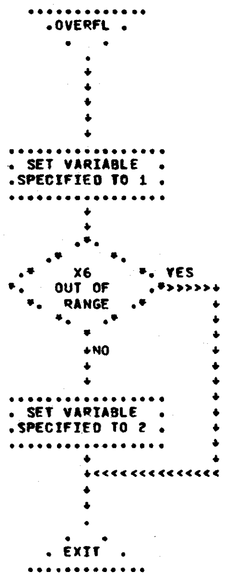
```
             .............
             .  LOCF   .
                .   .
                  .
                  +
                  +
                  +
                  +
           ...............
           .  SET ADDRESS  .
           .  OF VARIABLE  .
           .     IN X6     .
           ...............
                  +
                  +
                  +
                  +
                  .
                .   .
             .  EXIT   .
             .............
```

>OVERLAY<

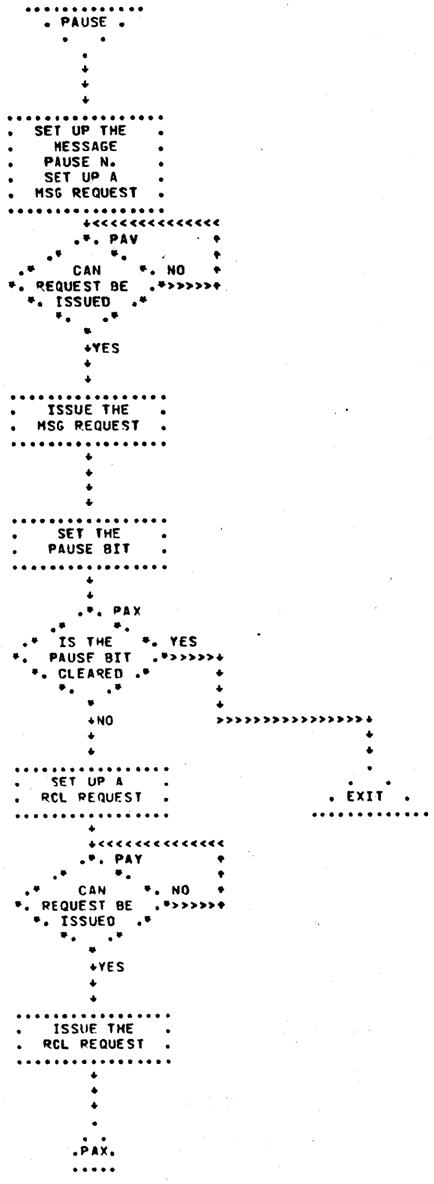
NOTES
1. SYSTEM. LIST ERROR MESSAGE AND GIVE TRACEBACK


OVERLAY
EXTRACT PRIMARY AND SECONDARY OVERLAY NUMBERS
FOURTH PARAMETER RECALL
NO
YES
CHECK TO SEE IF REQUESTED OVERLAY WAS LAST ONE LOADED
L1 SAME AS OVERLAY IN CORE
NO
YES
L2 SAME AS OVERLAY IN CORE
NO
YES
EXTRACT OVERLAY ENTRY ADDRESS
DELETE TRAINING BLANKS FROM FILE NAME
SET FILE NAME AND OVERLAY NUMBERS INTO LOADER CALL
RJ LOADER LOAD REQUEST-ED OVERLAY
LOADER DETECT FAT. ERROR
YES
NO
SEE NOTE 1
ABORT JOB RJ TO ABNORMAL
ABNORML
SAVE PRIMARY AND SECONDARY LEVELS. SET UP OVERLAY EXIT LINE
EXIT IS TO EXIT LINE OF OVERLAY +1
EXIT

>OVERFL<


OVERFL
SET VARIABLE SPECIFIED TO 1
X6 OUT OF RANGE
YES
NO
SET VARIABLE SPECIFIED TO 2
EXIT

>PAUSE<

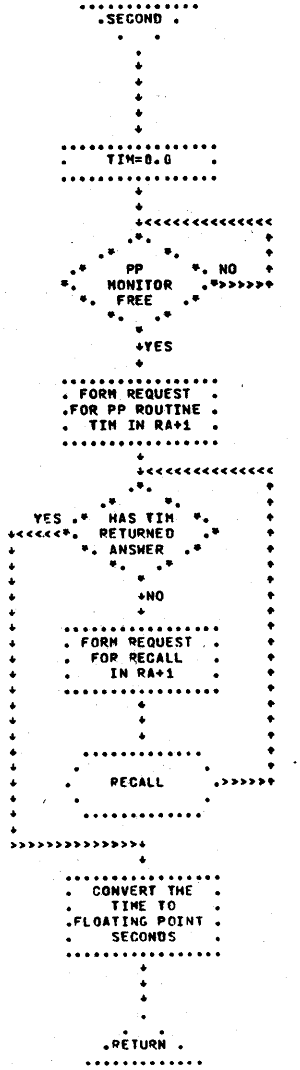

PAUSE
SET UP THE MESSAGE PAUSE N. SET UP A MSG REQUEST
PAV
CAN REQUEST BE ISSUED
NO
YES
ISSUE THE MSG REQUEST
SET THE PAUSE BIT
PAX
IS THE PAUSE BIT CLEARED
YES
NO
SET UP A RCL REQUEST
EXIT
PAY
CAN REQUEST BE ISSUED
NO
YES
ISSUE THE RCL REQUEST
PAX

>REMARK<

```
          .............
          .REMARK .
             .   .
               .
               +
               +
               +
               +
       .................
       .   FETCH THE   .
       .  MESSAGE AND  .
       . STORE IT FOR  .
       . A MSG REQUEST .
       .................
               +
               +
               +
       .................
       .  SET UP THE   .
       .  MSG REQUEST  .
       .................
               +
               +
               +
               +
             .....
             .NOP.
             .....
               +
               +
               +
               +
         .............
        .   RJ CPC    .
       .   ISSUE MSG   .
        .REQUEST W/RCL.
         ............
               +
               +
               +
               +
               .
             .   .
          . EXIT  .
          ............
```

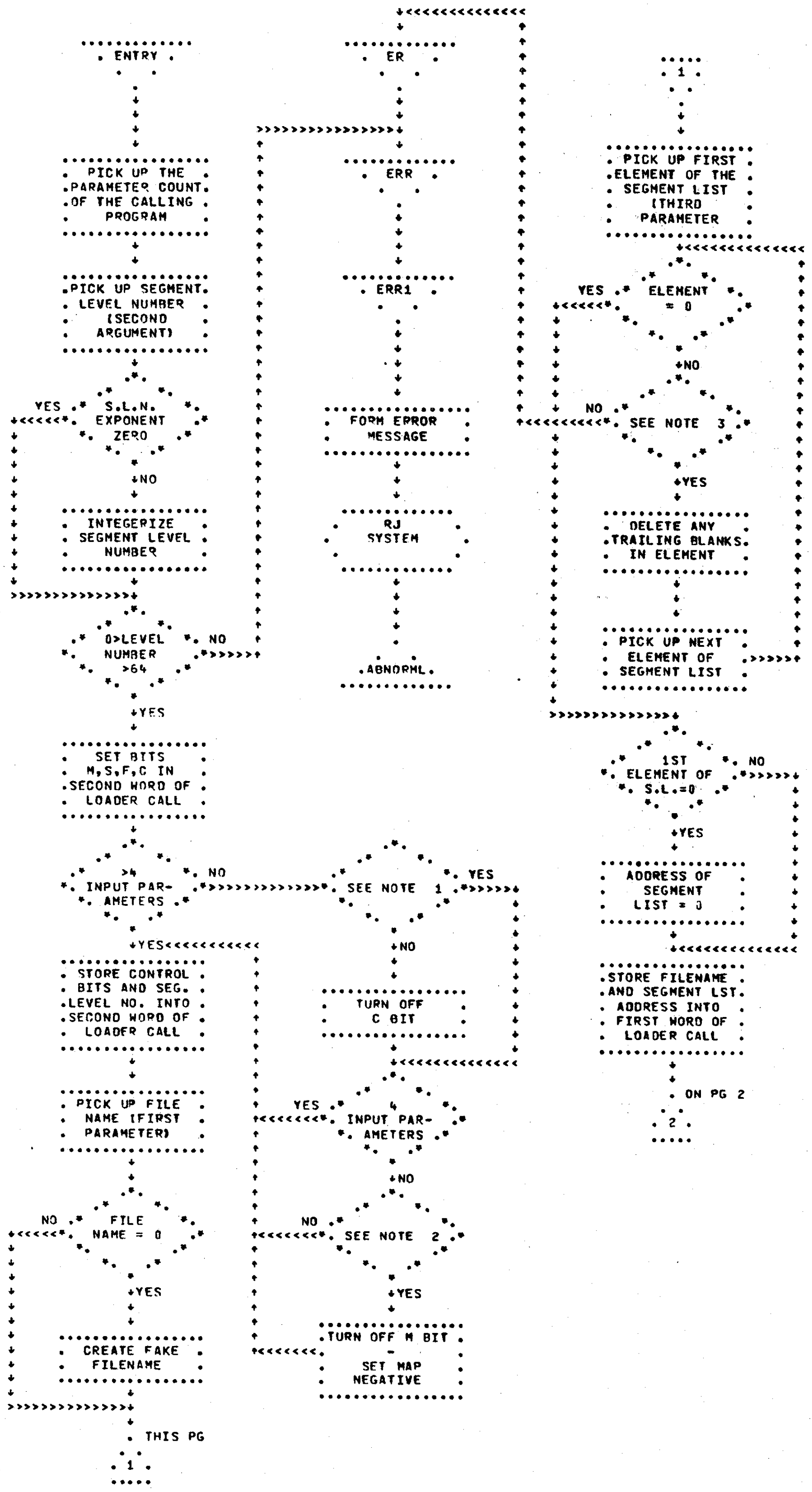
```
  .............
    .SCOPE28.
      .   .
        .
        ↓
        ↓
        ↓             >>>>>>>>>>>>>>>>>↓
       .*.            ↑                ↓
     .*   *.          ↑        .................
   .*   ANY   *. NO   ↑        .  WRITE ≠BAD    .
*.  SPECIFIED  .*>>>>>↑>>>>>>>>. PARAMETER≠ IN .
  *. PARAMS. .*       ↑        .    DAYFILE    .
     *.   .*          ↑        .................
        *             ↑                ↓
        ↓YES          ↑                ↓
       .*.            ↑                ↓
     .*   *.          ↑                ↓
   .*  MORE   *. YES  ↑                .
*.  SPECIFIED  .*>>>>>↑             .    .
  *. THAN 2  .*                   . EXIT   .
     *.   .*                    .............
        *
        ↓NO
       .*.
     .*   *.
   .* LOGICAL  *. NO
*.  FILE NAME  .*>>>>>↓
  *. SPECIF. .*       ↓
     *.   .*          ↓
        *             ↓
        ↓YES          ↓
        ↓             ↓
.................     ↓
.CHANGE NAME OF .     ↓
.    FILE TO    .     ↓
.   WRITE ON    .     ↓
.................     ↓
        ↓             ↓
        ↓<<<<<<<<<<<<<<
        ↓
.................
. CHANGE NUMBER .
.   OF FILES    .
.   TO BINARY   .
.................
        ↓
        ↓
        ↓
        ↓
.................
.   PLACE IN    .
.  LOADEP CARD  .
.................
        ↓
        ↓
        ↓
.................
. WRITE LOADER  .
.INFORMATION ON .
. LOGICAL FILE  .
.................
        ↓
        ↓
        ↓
        ↓
        .
      .   .
    . EXIT  .
  .............
```

>SECOND<


SECOND
TIM=0.0
PP MONITOR FREE
NO
YES
FORM REQUEST FOR PP ROUTINE TIM IN RA+1
HAS TIM RETURNED ANSWER
YES
NO
FORM REQUEST FOR RECALL IN RA+1
RECALL
CONVERT THE TIME TO FLOATING POINT SECONDS
RETURN

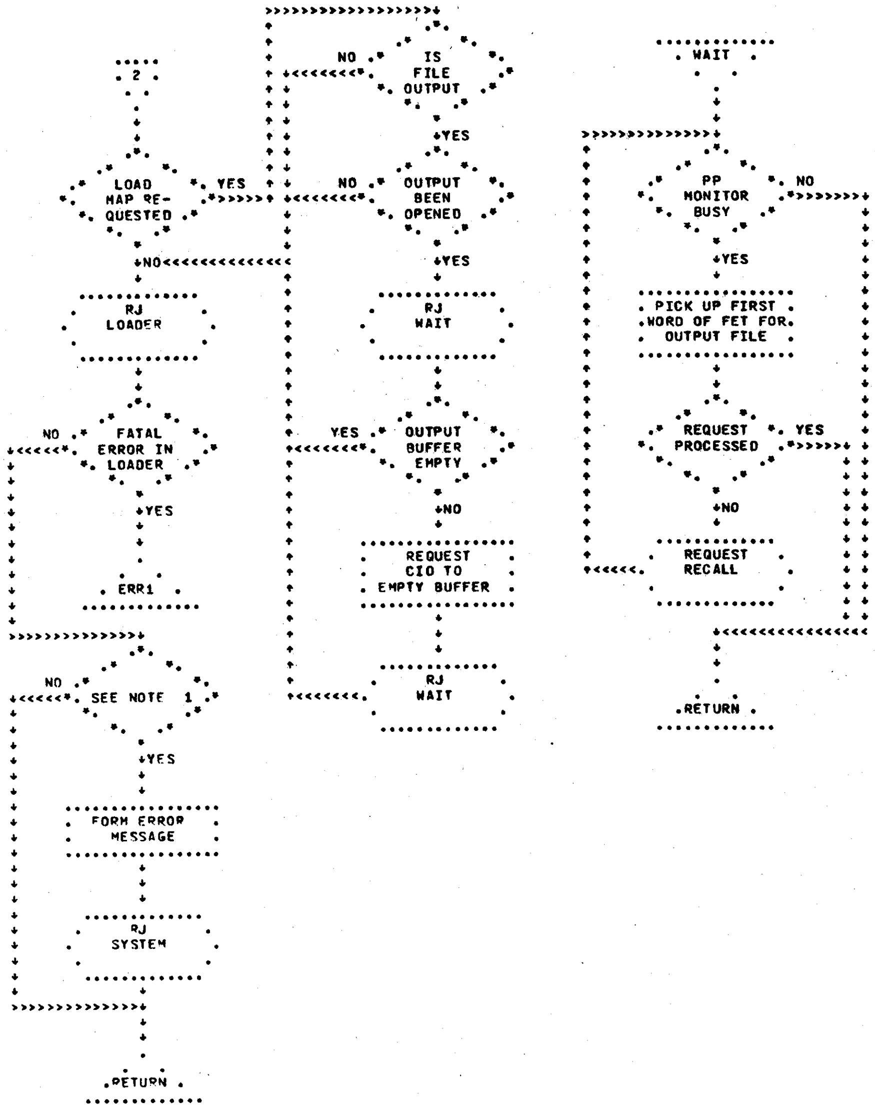
>SEGMENT<

NOTES
1. LIBRARY CALLS DFSIRED (PARAMETER 4 = 0)
2. IS MAP DESIRED (PARAMETER 5 ≠ 0)
3. ELEMENT LEGAL ALPHANUMERIC IDENTIFIER

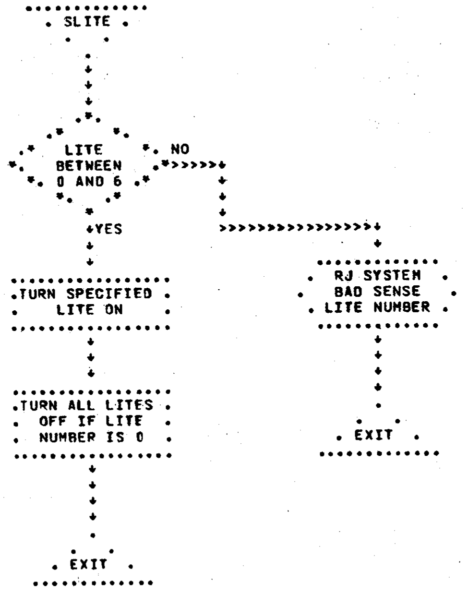

ENTRY
PICK UP THE PARAMETER COUNT OF THE CALLING PROGRAM
PICK UP SEGMENT LEVEL NUMBER (SECOND ARGUMENT)
S.L.N. EXPONENT ZERO
YES
NO
INTEGERIZE SEGMENT LEVEL NUMBER
0>LEVEL NUMBER >64
NO
YES
SET BITS M,S,F,C IN SECOND WORD OF LOADER CALL
>4 INPUT PARAMETERS
NO
YES
STORE CONTROL BITS AND SEG. LEVEL NO. INTO SECOND WORD OF LOADER CALL
PICK UP FILE NAME (FIRST PARAMETER)
FILE NAME = 0
NO
YES
CREATE FAKE FILENAME
THIS PG
1
ER
ERR
ERR1
FORM EPROR MESSAGE
RJ SYSTEM
ABNORML
SEE NOTE 1
YES
NO
TURN OFF C BIT
4 INPUT PARAMETERS
YES
NO
SEE NOTE 2
NO
YES
TURN OFF M BIT - SET MAP NEGATIVE
1
PICK UP FIRST ELEMENT OF THE SEGMENT LIST (THIRD PARAMETER
ELEMENT = 0
YES
NO
SEE NOTE 3
NO
YES
DELETE ANY TRAILING BLANKS IN ELEMENT
PICK UP NEXT ELEMENT OF SEGMENT LIST
1ST ELEMENT OF S.L.=0
NO
YES
ADDRESS OF SEGMENT LIST = 0
STORE FILENAME AND SEGMENT LST. ADDRESS INTO FIRST WORD OF LOADER CALL
ON PG 2
2

>SEGMENT<

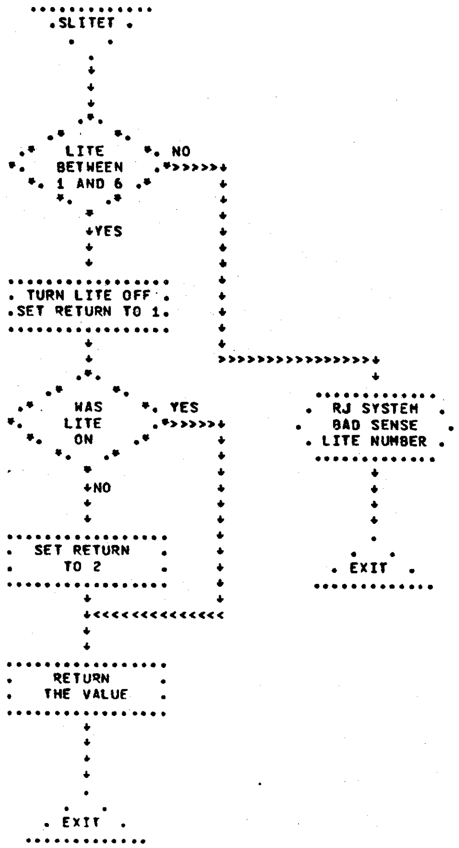
NOTES
   1. NON-FATAL ERROR IN LOADER AND MAP REQUESTED



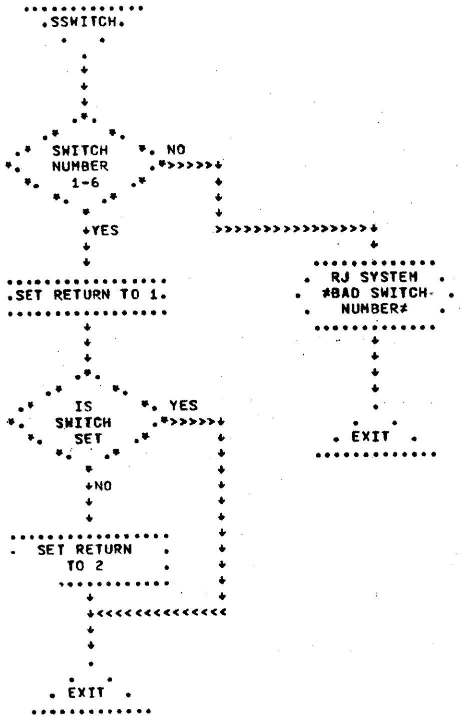
>SLITE<


SLITE
LITE BETWEEN 0 AND 6
NO
YES
TURN SPECIFIED LITE ON
TURN ALL LITES OFF IF LITE NUMBER IS 0
EXIT
RJ SYSTEM BAD SENSE LITE NUMBER
EXIT

>SLITET<

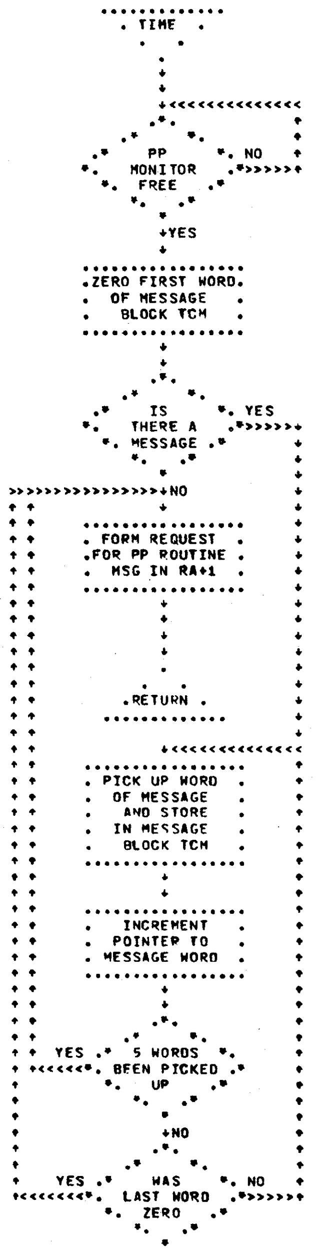

SLITET
LITE BETWEEN 1 AND 6
NO
YES
TURN LITE OFF
SET RETURN TO 1.
WAS LITE ON
YES
NO
SET RETURN TO 2
RJ SYSTEM
BAD SENSE
LITE NUMBER
EXIT
RETURN THE VALUE
EXIT

>SSWTCH<


SSWITCH
SWITCH NUMBER 1-6
NO
YES
SET RETURN TO 1
RJ SYSTEM
≠BAD SWITCH NUMBER≠
IS SWITCH SET
YES
NO
SET RETURN TO 2
EXIT
EXIT

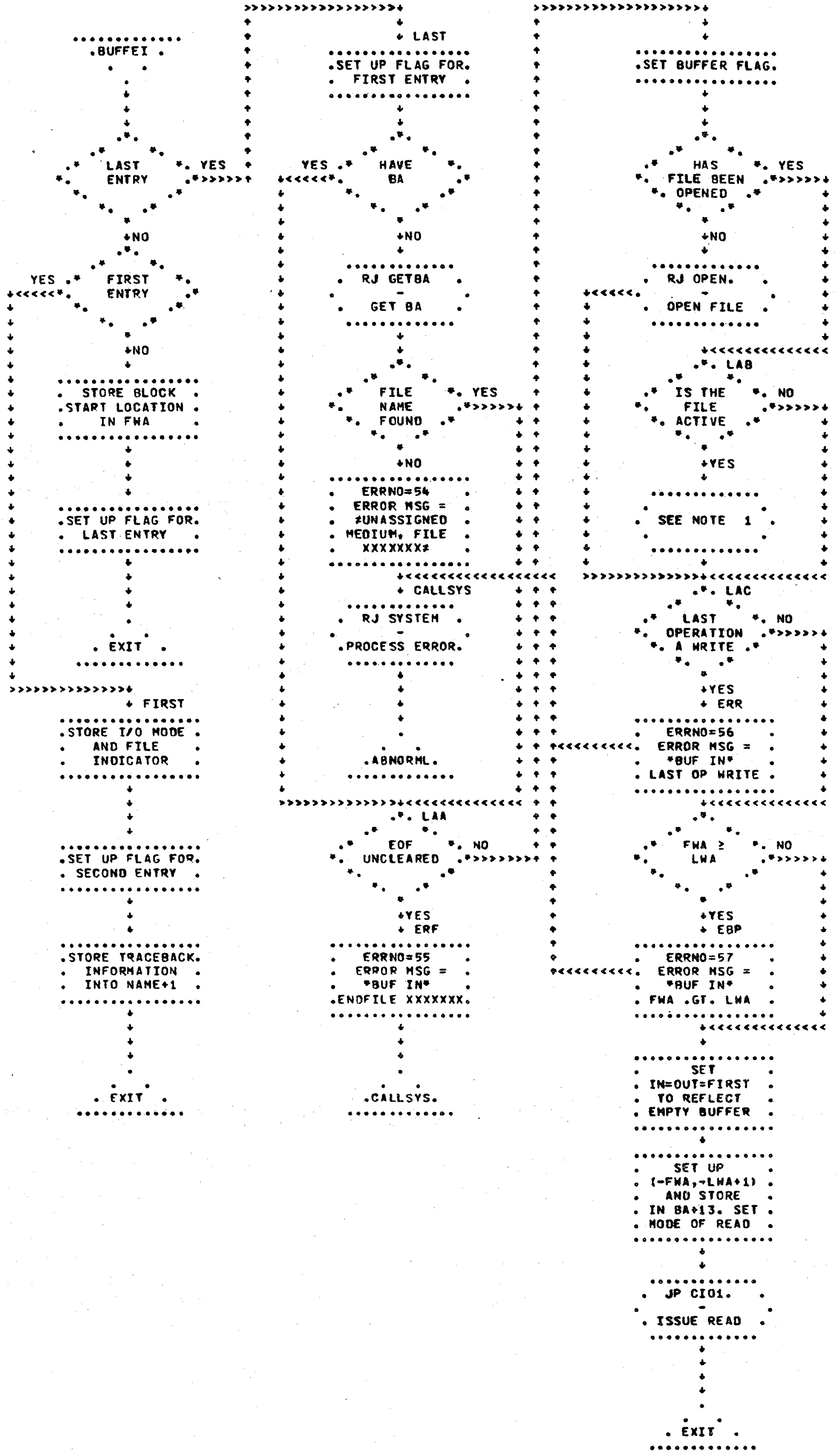
>START<

```
    .............
    .  START  .
       .   .
         .
         ↓
         ↓
         ↓
         ↓
         ↓
 .................
 .   WAIT UNTIL  .
 . RA+1 IS ZERO  .
 .................
         ↓
         ↓
 .................
 .SEND A MESSAGE .
 . REQUEST (MSG) .
 .    TO PLACE   .
 .     ≠START≠   .
 .   IN DAYFILE  .
 .................
         ↓
         ↓
         ↓
         .
       .   .
    .  EXIT   .
    .............
```


TIME
PP MONITOR FREE
NO
YES
ZERO FIRST WORD OF MESSAGE BLOCK TCM
IS THERE A MESSAGE
YES
NO
FORM REQUEST FOR PP ROUTINE MSG IN RA+1
RETURN
PICK UP WORD OF MESSAGE AND STORE IN MESSAGE BLOCK TCM
INCREMENT POINTER TO MESSAGE WORD
YES
5 WORDS BEEN PICKED UP
NO
YES
WAS LAST WORD ZERO
NO

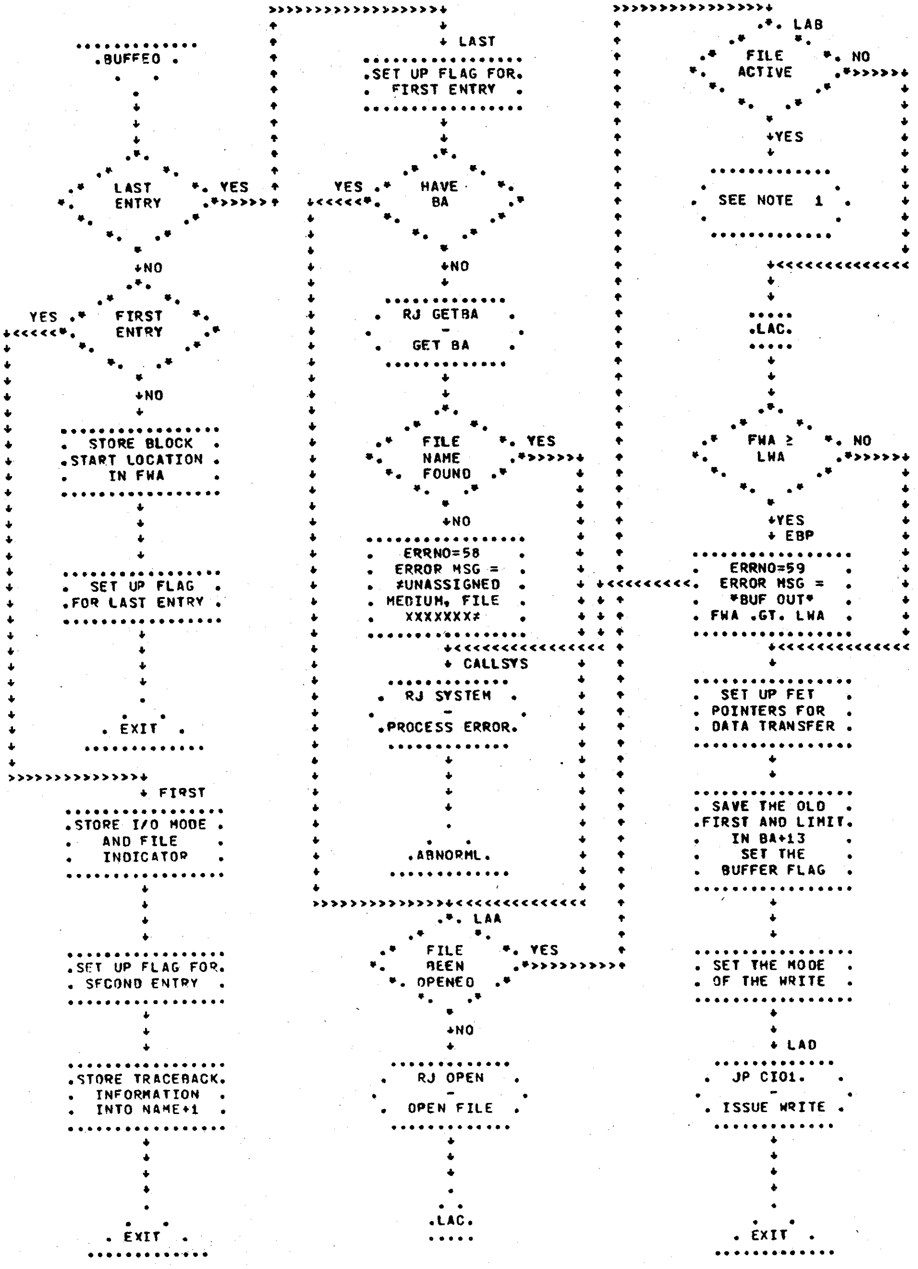
>BUFFEI<

NOTES
1. JP CIO1. GO INTO RCL UNTIL FILE INACTIVE


BUFFEI
LAST ENTRY
YES
NO
FIRST ENTRY
YES
NO
STORE BLOCK START LOCATION IN FWA
SET UP FLAG FOR. LAST ENTRY
EXIT
FIRST
STORE I/O MODE AND FILE INDICATOR
SET UP FLAG FOR. SECOND ENTRY
STORE TRACEBACK. INFORMATION INTO NAME+1
EXIT
LAST
SET UP FLAG FOR. FIRST ENTRY
HAVE BA
YES
NO
RJ GETBA - GET BA
FILE NAME FOUND
YES
NO
ERRNO=54 ERROR MSG = ≠UNASSIGNED MEDIUM, FILE XXXXXXX≠
CALLSYS
RJ SYSTEM - PROCESS ERROR.
ABNORML.
LAA
EOF UNCLEARED
NO
YES
ERF
ERRNO=55 ERROR MSG = *BUF IN* ENDFILE XXXXXXX.
CALLSYS.
SET BUFFER FLAG.
HAS FILE BEEN OPENED
YES
NO
RJ OPEN. - OPEN FILE
LAB
IS THE FILE ACTIVE
NO
YES
SEE NOTE 1
LAC
LAST OPERATION A WRITE
NO
YES
ERR
ERRNO=56 ERROR MSG = *BUF IN* LAST OP WRITE
FWA ≥ LWA
NO
YES
EBP
ERRNO=57 ERROR MSG = *BUF IN* FWA .GT. LWA
SET IN=OUT=FIRST TO REFLECT EMPTY BUFFER
SET UP (-FWA,-LWA+1) AND STORE IN BA+13. SET MODE OF READ
JP CIO1. - ISSUE READ
EXIT

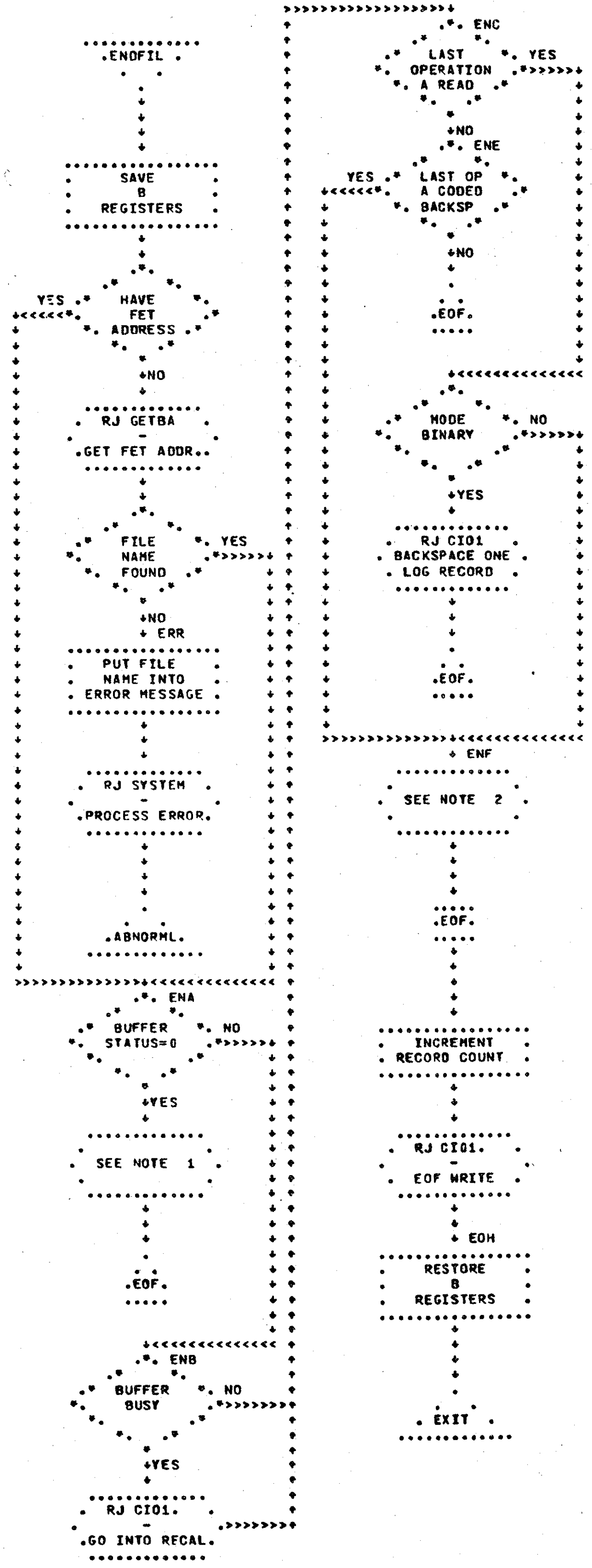
>BUFFEO<

NOTES
1. JP CIO1. GO INTO RCL UNTIL FILE INACTIVE

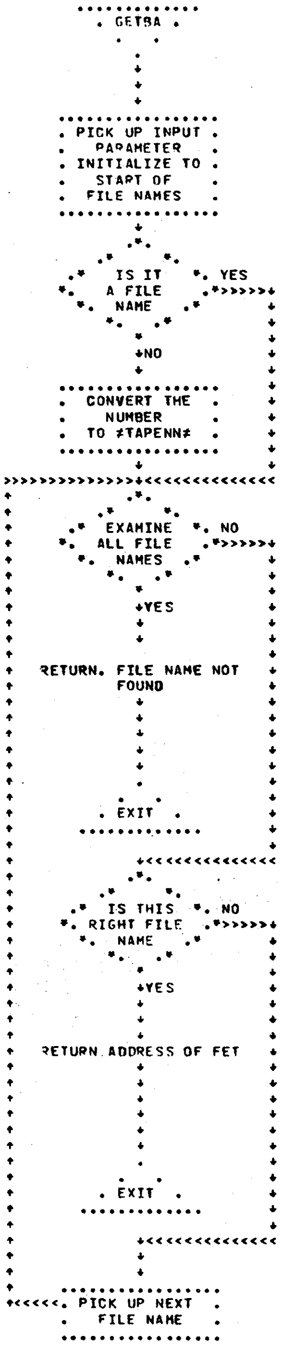

BUFFEO
LAST ENTRY
YES
NO
FIRST ENTRY
YES
NO
STORE BLOCK START LOCATION IN FWA
SET UP FLAG FOR LAST ENTRY
EXIT
FIRST
STORE I/O MODE AND FILE INDICATOR
SET UP FLAG FOR SECOND ENTRY
STORE TRACEBACK INFORMATION INTO NAME+1
EXIT
LAST
SET UP FLAG FOR FIRST ENTRY
HAVE BA
YES
NO
RJ GETBA - GET BA
FILE NAME FOUND
YES
NO
ERRNO=58 ERROR MSG = ≠UNASSIGNED MEDIUM, FILE XXXXXXX≠
CALLSYS
RJ SYSTEM - PROCESS ERROR.
ABNORML.
LAA
FILE BEEN OPENED
YES
NO
RJ OPEN - OPEN FILE
LAC.
LAB
FILE ACTIVE
NO
YES
SEE NOTE 1
LAC.
FWA ≥ LWA
NO
YES
EBP
ERRNO=59 ERROR MSG = *BUF OUT* FWA .GT. LWA
SET UP FET POINTERS FOR DATA TRANSFER
SAVE THE OLD FIRST AND LIMIT IN BA+13 SET THE BUFFER FLAG
SET THE MODE OF THE WRITE
LAD
JP CIO1. - ISSUE WRITE
EXIT

>ENDFIL<

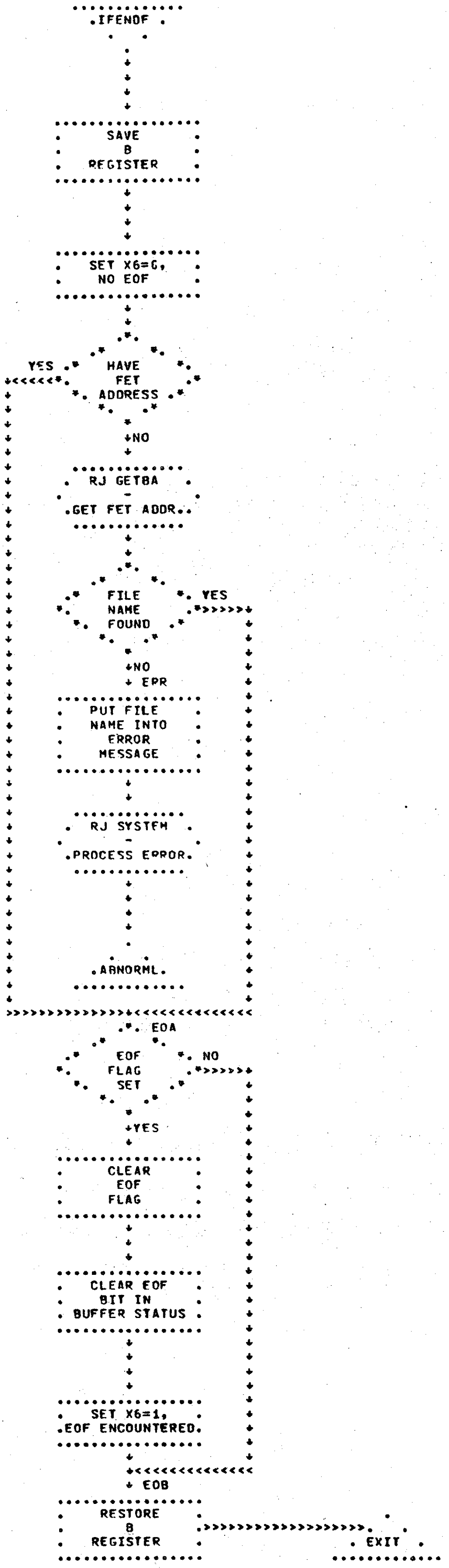
NOTES
   1. RJ OPEN. OPEN AS IN WRITE(0) NOT REWIND
   2. RJ POSFIL. POSITION FILE AFTER CURRENT PRU

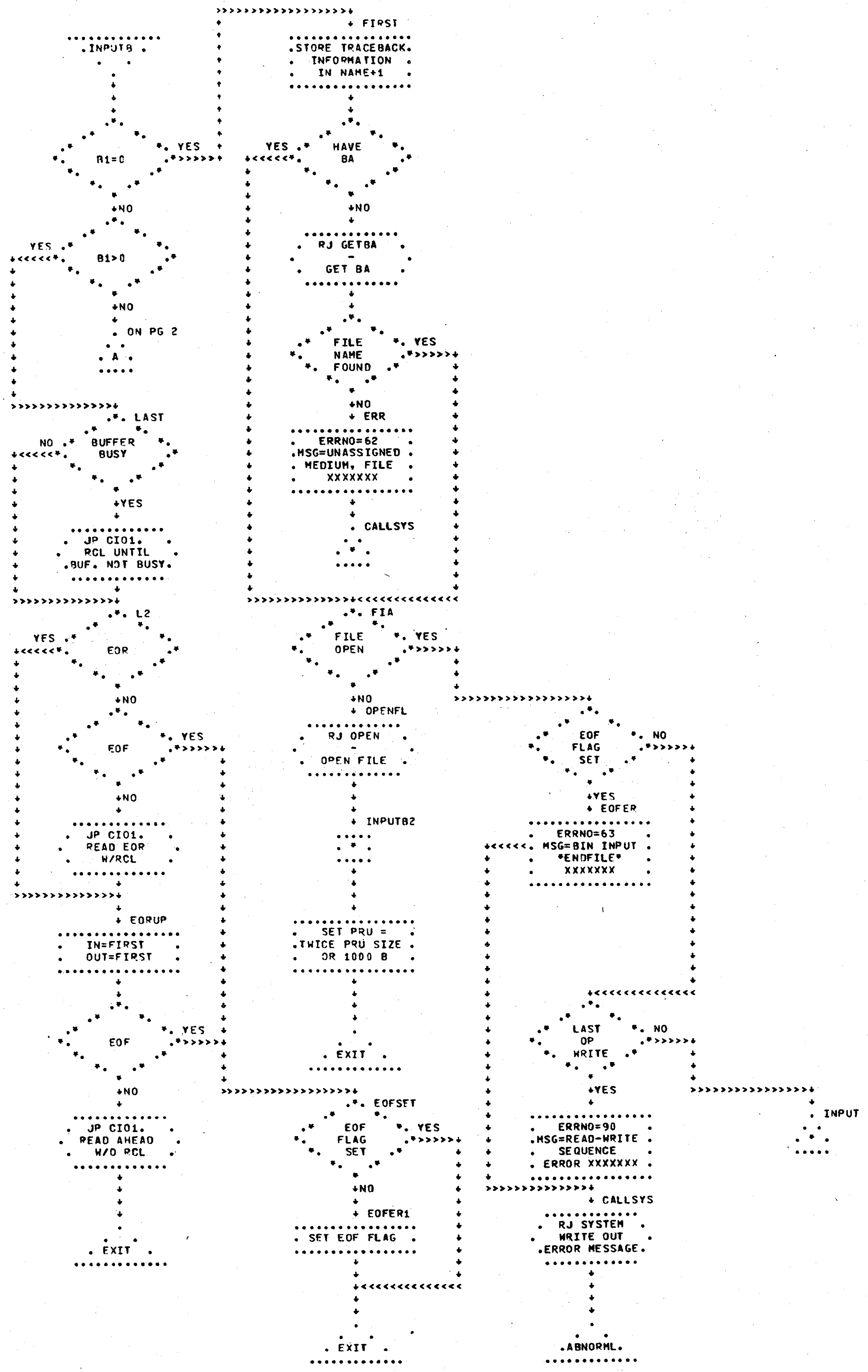


>GETBA<


GETBA
PICK UP INPUT PARAMETER INITIALIZE TO START OF FILE NAMES
IS IT A FILE NAME
YES
NO
CONVERT THE NUMBER TO ≠TAPENN≠
EXAMINE ALL FILE NAMES
NO
YES
RETURN. FILE NAME NOT FOUND
EXIT
IS THIS RIGHT FILE NAME
NO
YES
RETURN ADDRESS OF FET
EXIT
PICK UP NEXT FILE NAME

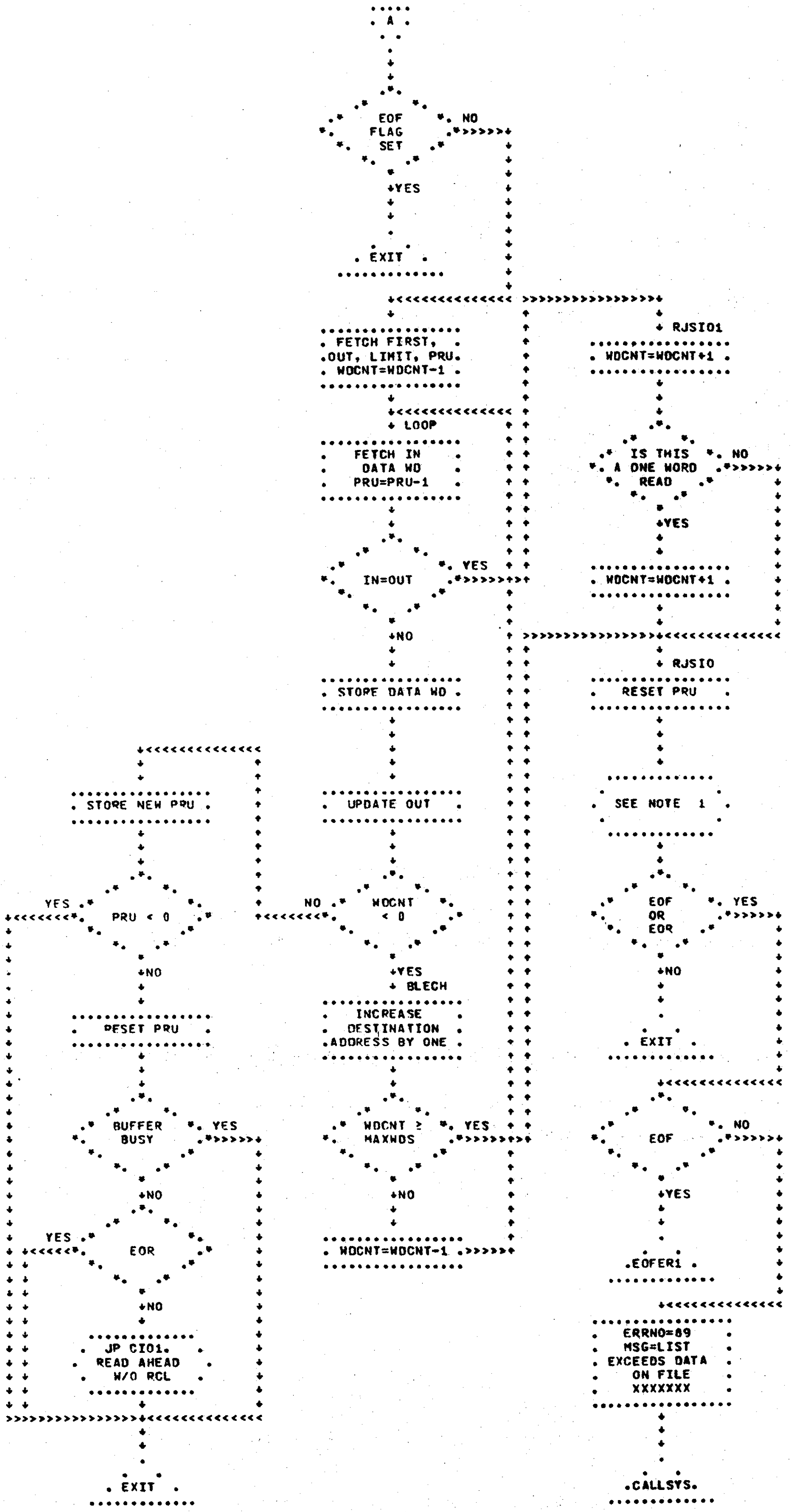
>IFENOF<


IFENOF
SAVE B REGISTER
SET X6=0, NO EOF
HAVE FET ADDRESS
YES
NO
RJ GETBA - GET FET ADDR.
FILE NAME FOUND
YES
NO
EPR
PUT FILE NAME INTO ERROR MESSAGE
RJ SYSTEM - PROCESS ERROR.
ABNORML.
EOA
EOF FLAG SET
NO
YES
CLEAR EOF FLAG
CLEAR EOF BIT IN BUFFER STATUS
SET X6=1, EOF ENCOUNTERED.
EOB
RESTORE B REGISTER
EXIT


INPUTB
FIRST
STORE TRACEBACK INFORMATION IN NAME+1
B1=0
YES
NO
HAVE BA
YES
NO
RJ GETBA - GET BA
B1>0
YES
NO
ON PG 2
A
FILE NAME FOUND
YES
NO
ERR
ERRNO=62 MSG=UNASSIGNED MEDIUM, FILE XXXXXXX
CALLSYS
LAST
BUFFER BUSY
NO
YES
JP CIO1. RCL UNTIL BUF. NOT BUSY.
L2
EOR
YES
NO
FIA
FILE OPEN
YES
NO
OPENFL
RJ OPEN - OPEN FILE
EOF
YES
NO
JP CIO1. READ EOR W/RCL
EOF FLAG SET
NO
YES
EOFER
ERRNO=63 MSG=BIN INPUT *ENDFILE* XXXXXXX
INPUTB2
EORUP
IN=FIRST OUT=FIRST
SET PRU = TWICE PRU SIZE OR 1000 B
EOF
YES
NO
EXIT
LAST OP WRITE
NO
YES
INPUT
JP CIO1. READ AHEAD W/O RCL
EOFSET
EOF FLAG SET
YES
NO
ERRNO=90 MSG=READ-WRITE SEQUENCE ERROR XXXXXXX
CALLSYS
EOFER1
SET EOF FLAG
RJ SYSTEM WRITE OUT ERROR MESSAGE.
EXIT
EXIT
ABNORML.

SINPUTRK

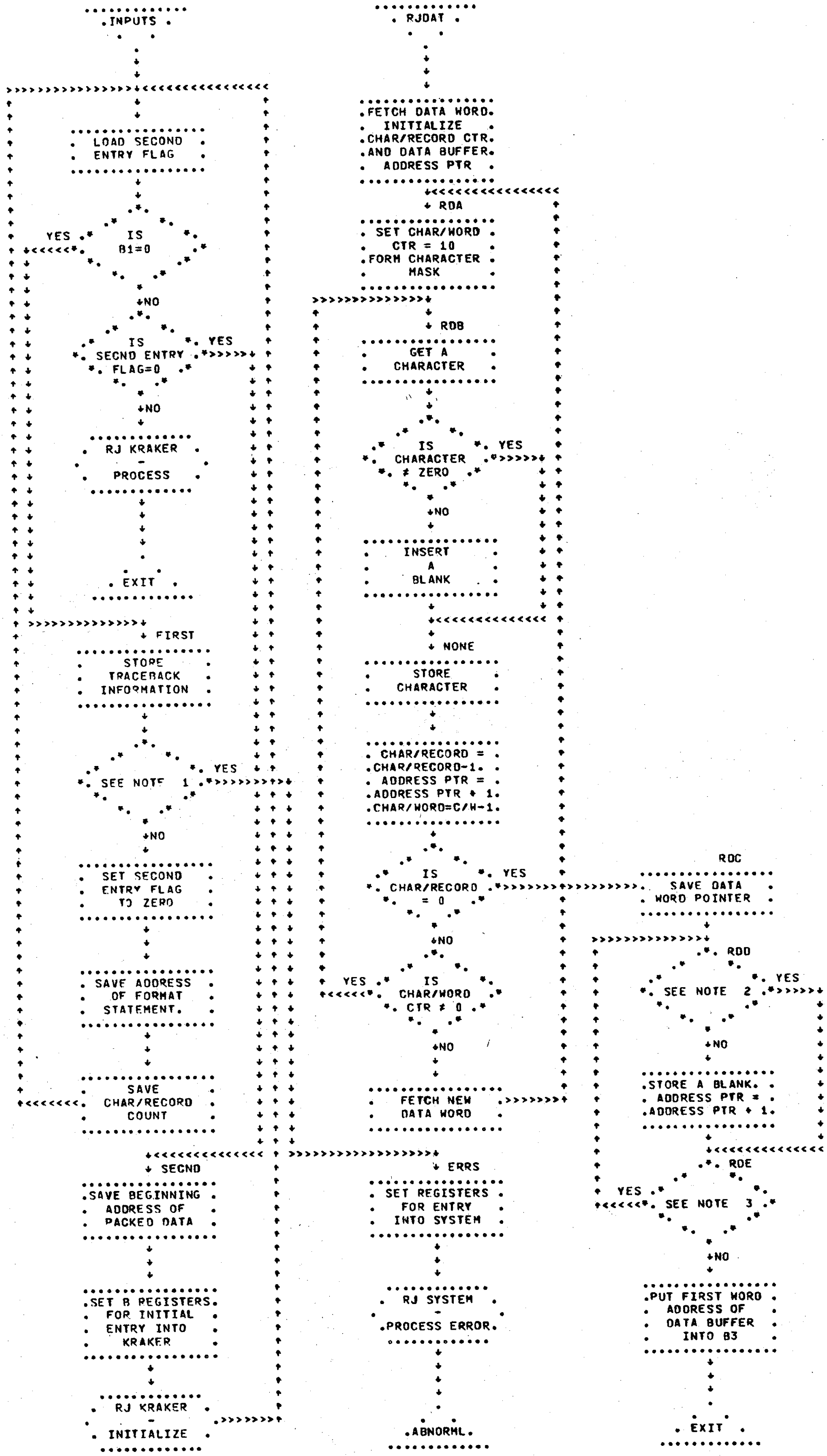
NOTES
1. RJ SIO. STORE DATA WDS. READ IF NECESSARY



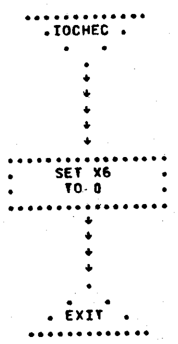
.INPUTS<

NOTES

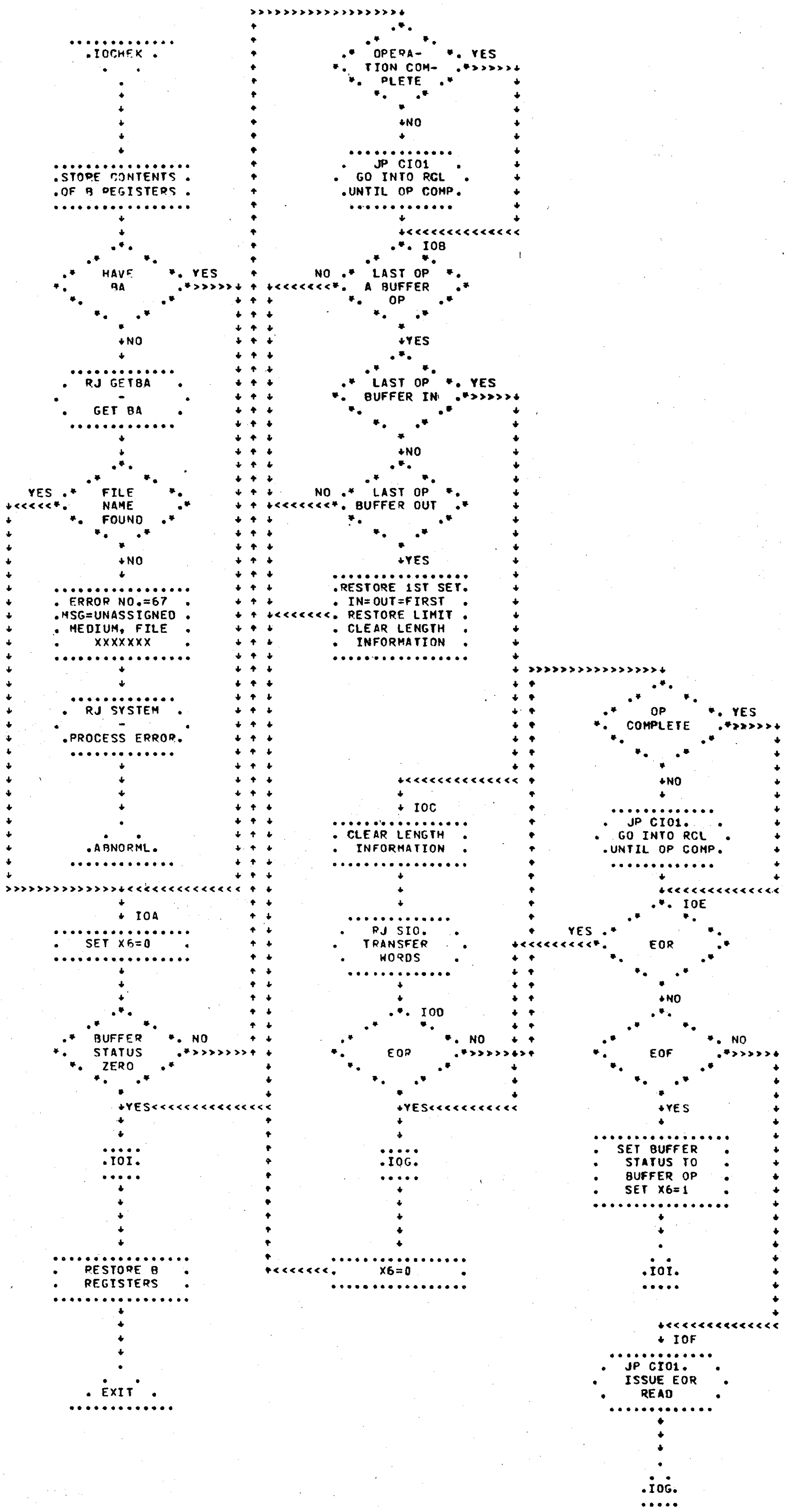
1. DOES CHAR/RECORD COUNT EXCEED 150
2. IS LAST WORD ADDRESS +1 OF DATA BUFFER = ADDRESS PTR
3. IS LAST WORD ADDRESS +1 OF DATA BUFFER ≠ ADDRESS PTR

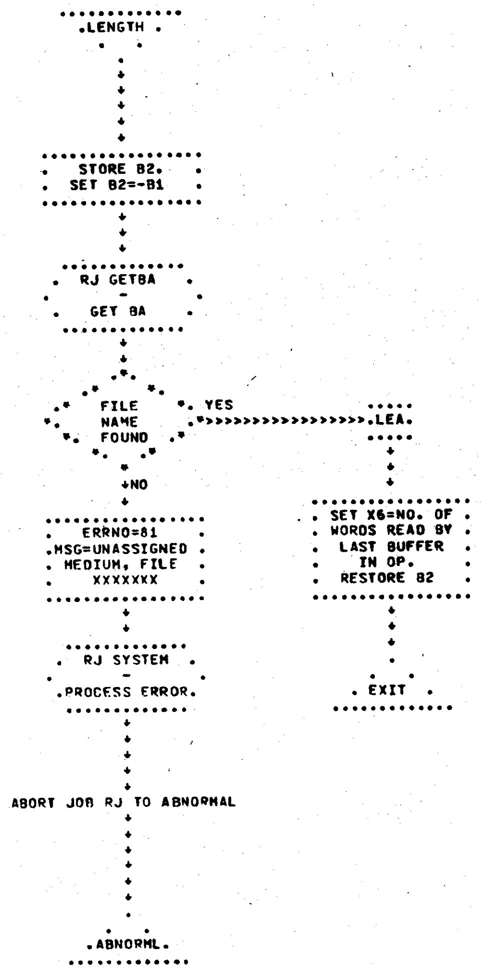

INPUTS
LOAD SECOND ENTRY FLAG
IS B1=0
YES
NO
IS SECND ENTRY FLAG=0
YES
NO
RJ KRAKER - PROCESS
EXIT
FIRST
STORE TRACEBACK INFORMATION
SEE NOTE 1
YES
NO
SET SECOND ENTRY FLAG TO ZERO
SAVE ADDRESS OF FORMAT STATEMENT.
SAVE CHAR/RECORD COUNT
SECND
SAVE BEGINNING ADDRESS OF PACKED DATA
SET B REGISTERS. FOR INITIAL ENTRY INTO KRAKER
RJ KRAKER - INITIALIZE
RJDAT
FETCH DATA WORD. INITIALIZE CHAR/RECORD CTR. AND DATA BUFFER. ADDRESS PTR
RDA
SET CHAR/WORD CTR = 10 FORM CHARACTER MASK
RDB
GET A CHARACTER
IS CHARACTER ≠ ZERO
YES
NO
INSERT A BLANK
NONE
STORE CHARACTER
CHAR/RECORD = CHAR/RECORD-1. ADDRESS PTR = ADDRESS PTR + 1. CHAR/WORD=C/W-1.
IS CHAR/RECORD = 0
YES
NO
IS CHAR/WORD CTR ≠ 0
YES
NO
FETCH NEW DATA WORD
ERRS
SET REGISTERS FOR ENTRY INTO SYSTEM
RJ SYSTEM - PROCESS ERROR.
ABNORML.
RDC
SAVE DATA WORD POINTER
RDD
SEE NOTE 2
YES
NO
STORE A BLANK. ADDRESS PTR = ADDRESS PTR + 1.
RDE
SEE NOTE 3
YES
NO
PUT FIRST WORD ADDRESS OF DATA BUFFER INTO B3
EXIT

>IOCHEC<

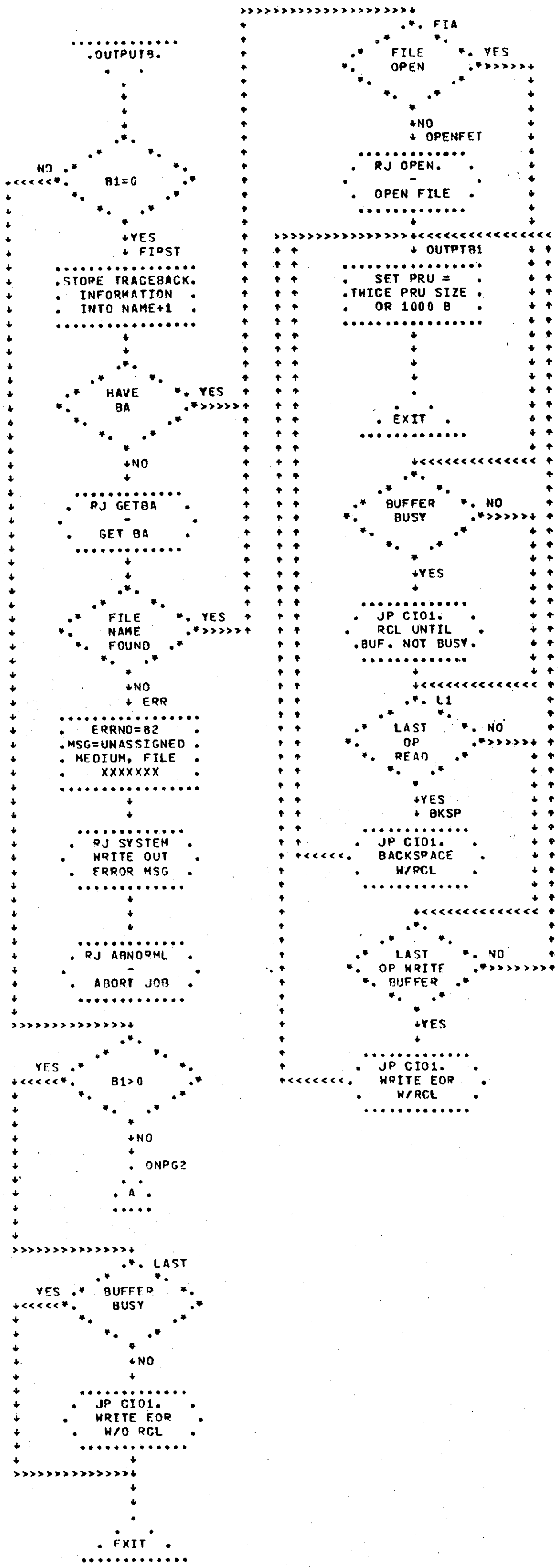


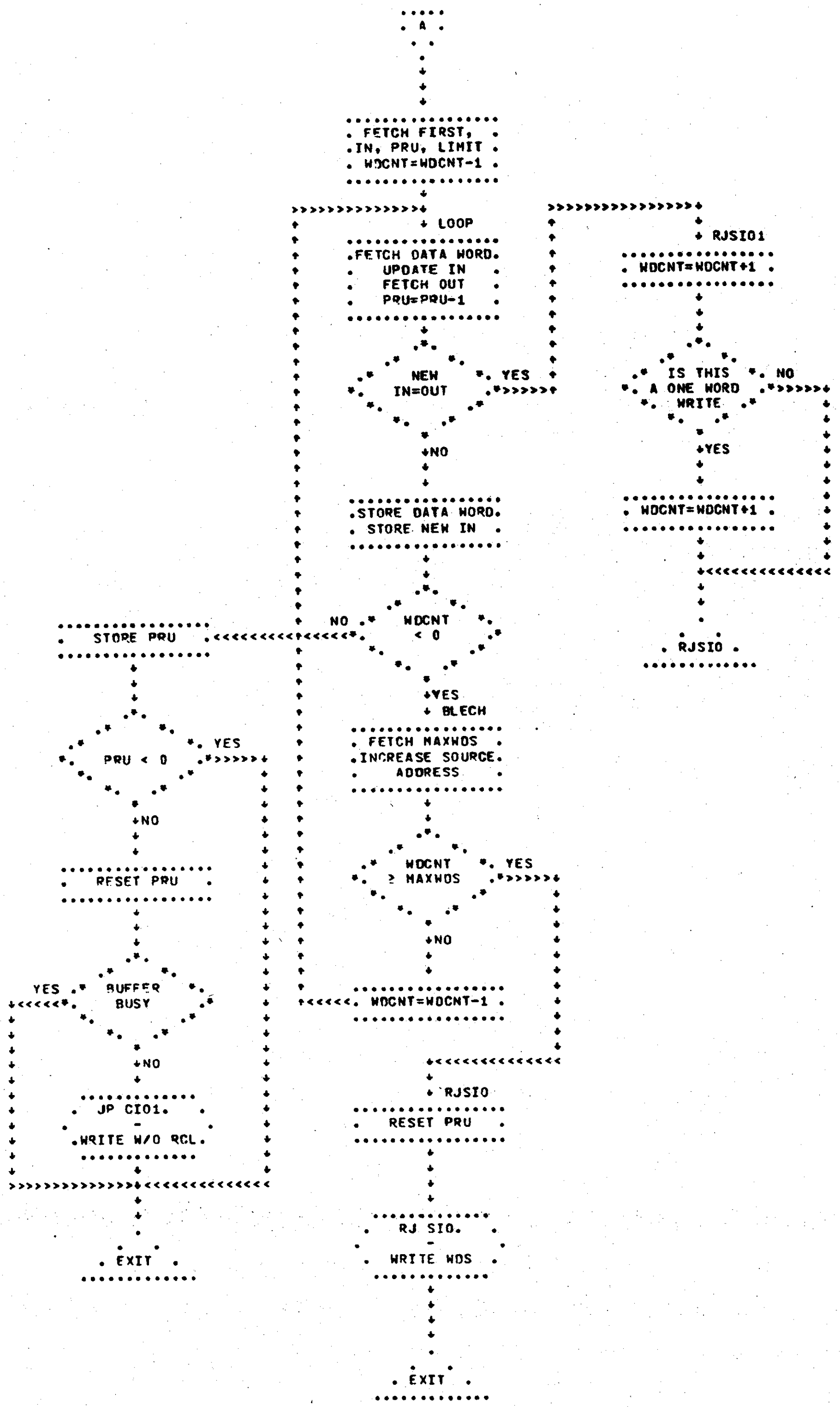
IOCHEK


IOCHEK
STORE CONTENTS OF B REGISTERS
HAVE BA
YES
NO
RJ GETBA - GET BA
FILE NAME FOUND
YES
NO
ERROR NO.=67 MSG=UNASSIGNED MEDIUM, FILE XXXXXXX
RJ SYSTEM - PROCESS ERROR.
ABNORML.
IOA
SET X6=0
BUFFER STATUS ZERO
NO
YES
IOI.
RESTORE B REGISTERS
EXIT
OPERA-TION COM-PLETE
YES
NO
JP CIO1 GO INTO RCL UNTIL OP COMP.
IOB
LAST OP A BUFFER OP
NO
YES
LAST OP BUFFER IN
YES
NO
LAST OP BUFFER OUT
NO
YES
RESTORE 1ST SET. IN=OUT=FIRST RESTORE LIMIT CLEAR LENGTH INFORMATION
IOC
CLEAR LENGTH INFORMATION
RJ SIO. TRANSFER WORDS
IOD
EOR
NO
YES
IOG.
X6=0
OP COMPLETE
YES
NO
JP CIO1. GO INTO RCL UNTIL OP COMP.
IOE
EOR
YES
NO
EOF
NO
YES
SET BUFFER STATUS TO BUFFER OP SET X6=1
IOI.
IOF
JP CIO1. ISSUE EOR READ
IOG.


LENGTH
STORE B2.
SET B2=-B1
RJ GETBA
-
GET BA
FILE
NAME
FOUND
YES
LEA
NO
ERRNO=81
MSG=UNASSIGNED
MEDIUM, FILE
XXXXXXX
SET X6=NO. OF
WORDS READ BY
LAST BUFFER
IN OP.
RESTORE B2
RJ SYSTEM
-
PROCESS ERROR
EXIT
ABORT JOB RJ TO ABNORMAL
ABNORML

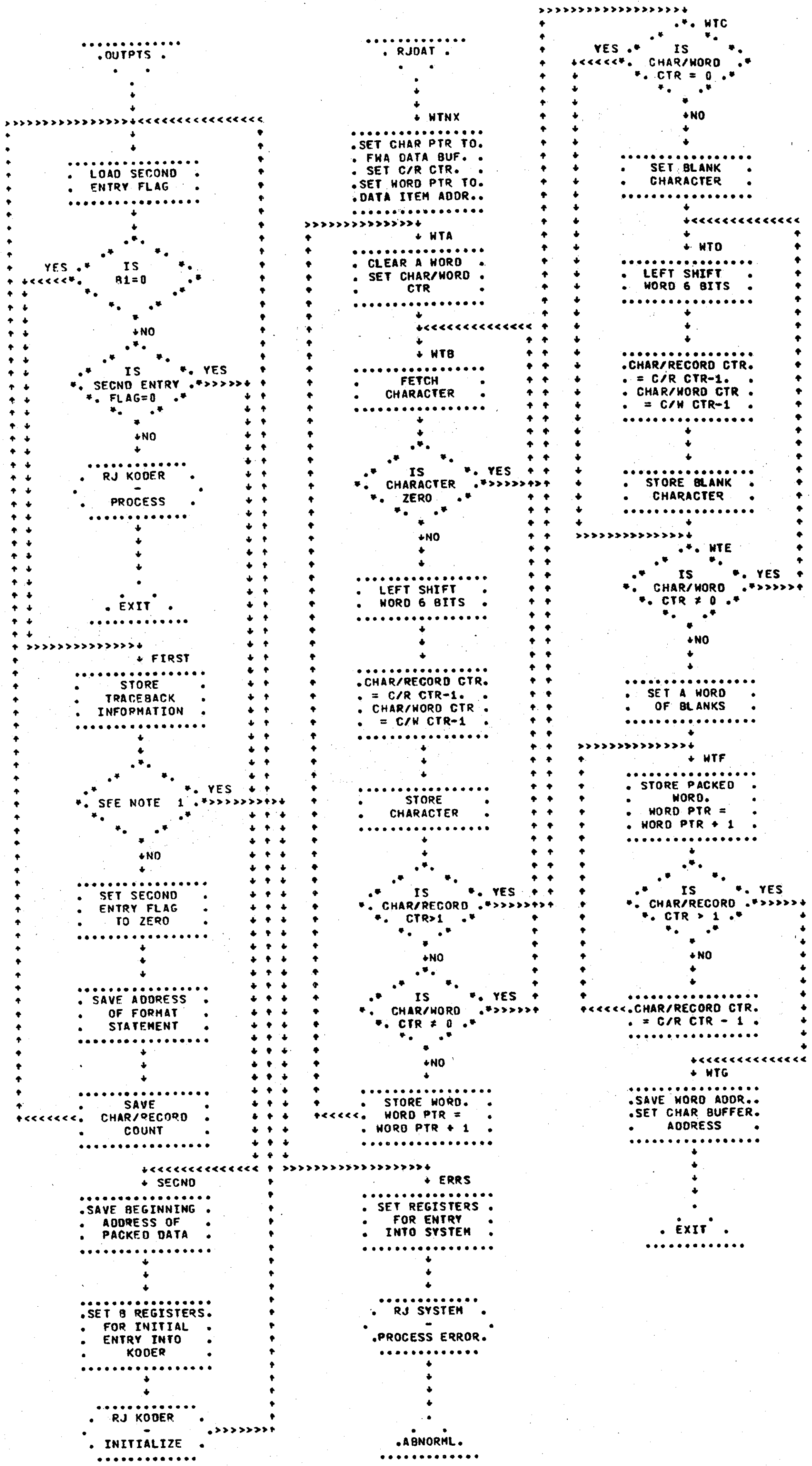
OUTPTB<


OUTPUTB.
NO
B1=0
YES
FIRST
STORE TRACEBACK. INFORMATION INTO NAME+1
HAVE BA
YES
NO
RJ GETBA
-
GET BA
FILE NAME FOUND
YES
NO
ERR
ERRNO=82
MSG=UNASSIGNED MEDIUM, FILE XXXXXXX
RJ SYSTEM WRITE OUT ERROR MSG
RJ ABNORML
-
ABORT JOB
YES
B1>0
NO
ONPG2
A
LAST
YES
BUFFER BUSY
NO
JP CIO1. WRITE EOR W/O RCL
EXIT
FIA
FILE OPEN
YES
NO
OPENFET
RJ OPEN.
-
OPEN FILE
OUTPTB1
SET PRU = TWICE PRU SIZE OR 1000 B
EXIT
BUFFER BUSY
NO
YES
JP CIO1. RCL UNTIL BUF. NOT BUSY.
L1
LAST OP READ
NO
YES
BKSP
JP CIO1. BACKSPACE W/RCL
LAST OP WRITE BUFFER
NO
YES
JP CIO1. WRITE EOR W/RCL


A
FETCH FIRST, IN, PRU, LIMIT WDCNT=WDCNT-1
LOOP
FETCH DATA WORD. UPDATE IN FETCH OUT PRU=PRU-1
NEW IN=OUT
YES
NO
RJSIO1
WDCNT=WDCNT+1
IS THIS A ONE WORD WRITE
NO
YES
WDCNT=WDCNT+1
RJSIO
STORE DATA WORD. STORE NEW IN
WDCNT < 0
NO
YES
BLECH
STORE PRU
PRU < 0
YES
NO
FETCH MAXWDS INCREASE SOURCE ADDRESS
WDCNT ≥ MAXWDS
YES
NO
RESET PRU
BUFFER BUSY
YES
NO
WDCNT=WDCNT-1
RJSIO
RESET PRU
JP CIO1. - WRITE W/O RCL.
EXIT
RJ SIO. - WRITE WDS
EXIT

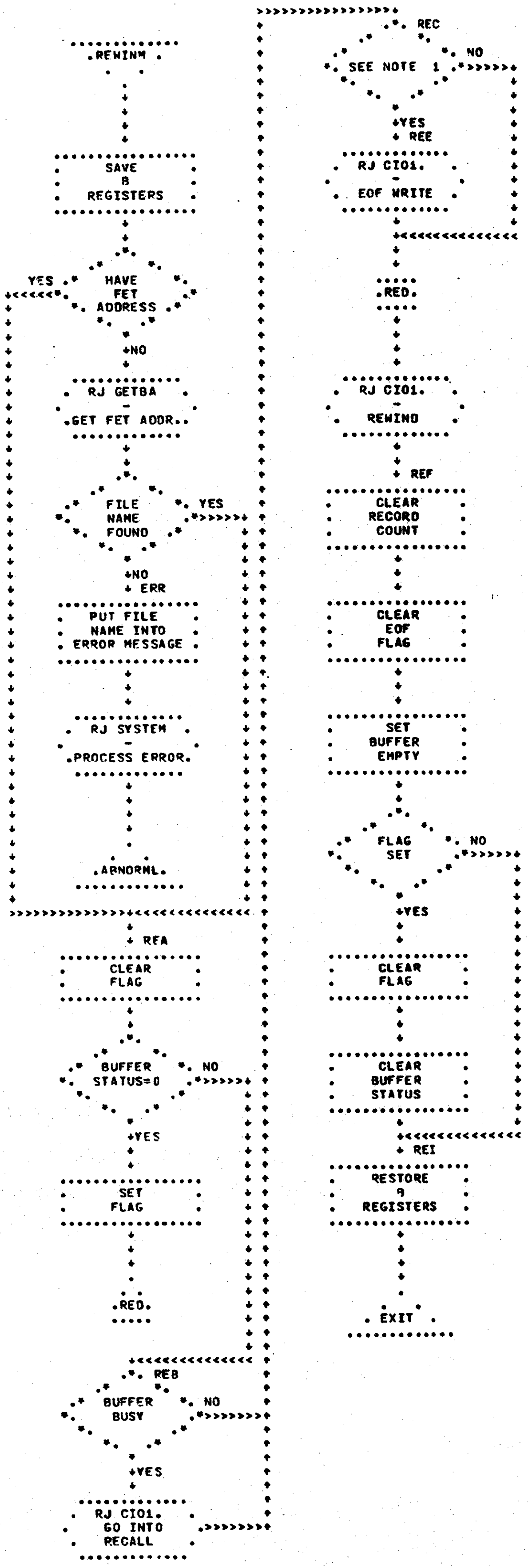
>OUTPTS<

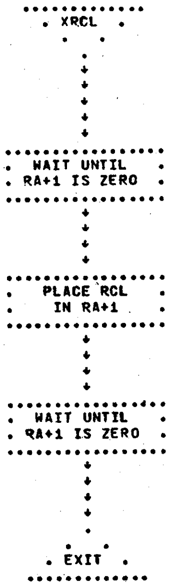
NOTES
1. DOES CHAR/RECORD COUNT EXCEED 150


OUTPTS
LOAD SECOND ENTRY FLAG
YES
IS B1=0
NO
IS SECND ENTRY FLAG=0
YES
NO
RJ KODER - PROCESS
EXIT
FIRST
STORE TRACEBACK INFORMATION
SEE NOTE 1
YES
NO
SET SECOND ENTRY FLAG TO ZERO
SAVE ADDRESS OF FORMAT STATEMENT
SAVE CHAR/RECORD COUNT
SECND
SAVE BEGINNING ADDRESS OF PACKED DATA
SET B REGISTERS. FOR INITIAL ENTRY INTO KODER
RJ KODER - INITIALIZE
RJDAT
WTNX
SET CHAR PTR TO. FWA DATA BUF. SET C/R CTR. SET WORD PTR TO. DATA ITEM ADDR.
WTA
CLEAR A WORD SET CHAR/WORD CTR
WTB
FETCH CHARACTER
IS CHARACTER ZERO
YES
NO
LEFT SHIFT WORD 6 BITS
CHAR/RECORD CTR. = C/R CTR-1. CHAR/WORD CTR = C/W CTR-1
STORE CHARACTER
IS CHAR/RECORD CTR>1
YES
NO
IS CHAR/WORD CTR ≠ 0
YES
NO
STORE WORD. WORD PTR = WORD PTR + 1
ERRS
SET REGISTERS FOR ENTRY INTO SYSTEM
RJ SYSTEM - PROCESS ERROR.
ABNORML.
WTC
YES
IS CHAR/WORD CTR = 0
NO
SET BLANK CHARACTER
WTD
LEFT SHIFT WORD 6 BITS
CHAR/RECORD CTR. = C/R CTR-1. CHAR/WORD CTR = C/W CTR-1
STORE BLANK CHARACTER
WTE
IS CHAR/WORD CTR ≠ 0
YES
NO
SET A WORD OF BLANKS
WTF
STORE PACKED WORD. WORD PTR = WORD PTR + 1
IS CHAR/RECORD CTR > 1
YES
NO
CHAR/RECORD CTR. = C/R CTR - 1.
WTG
SAVE WORD ADDR. SET CHAR BUFFER. ADDRESS
EXIT

>REWINM<

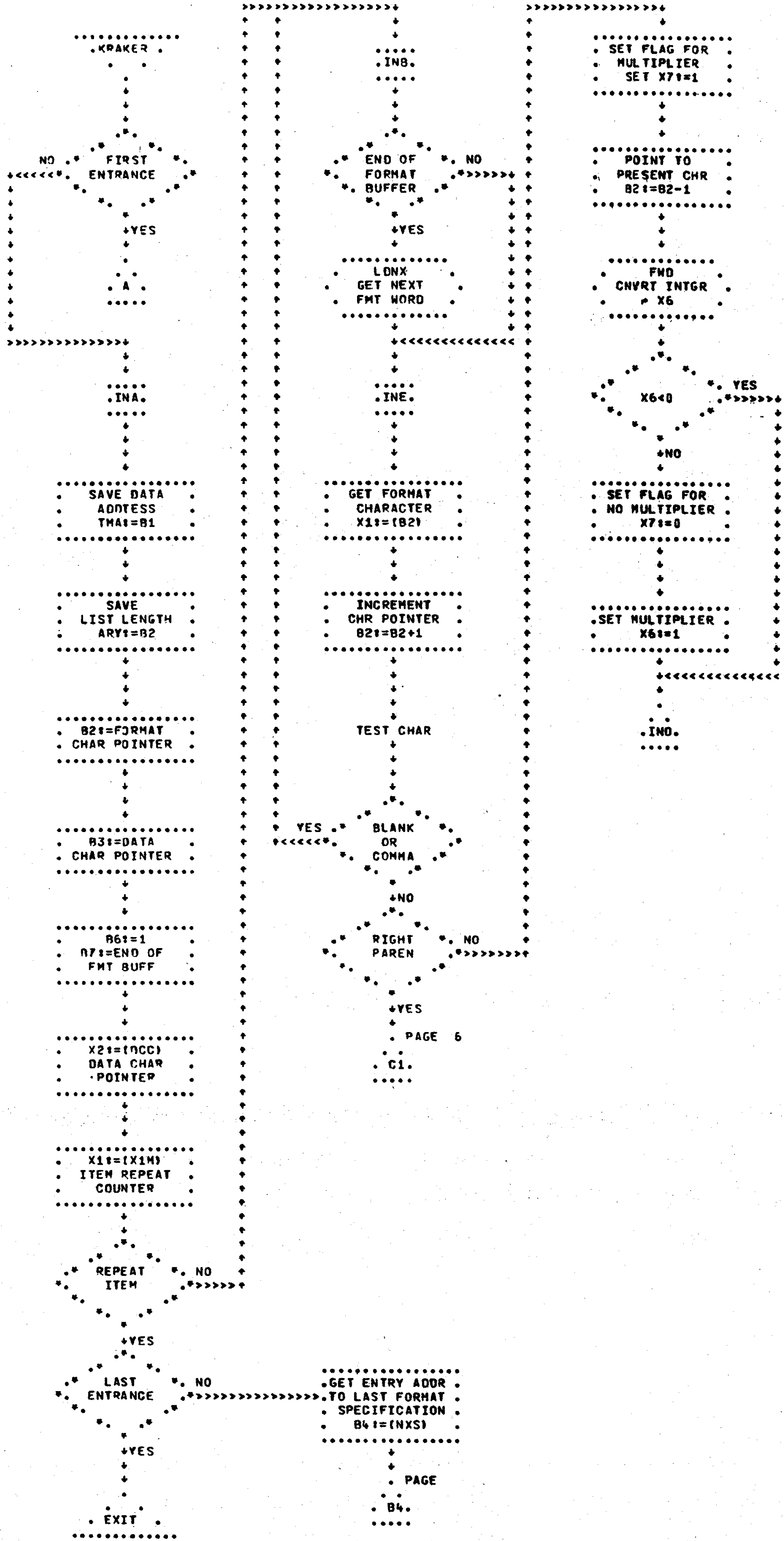
NOTES
1. LAST OP WRITE BUT NOT EOF WRITE



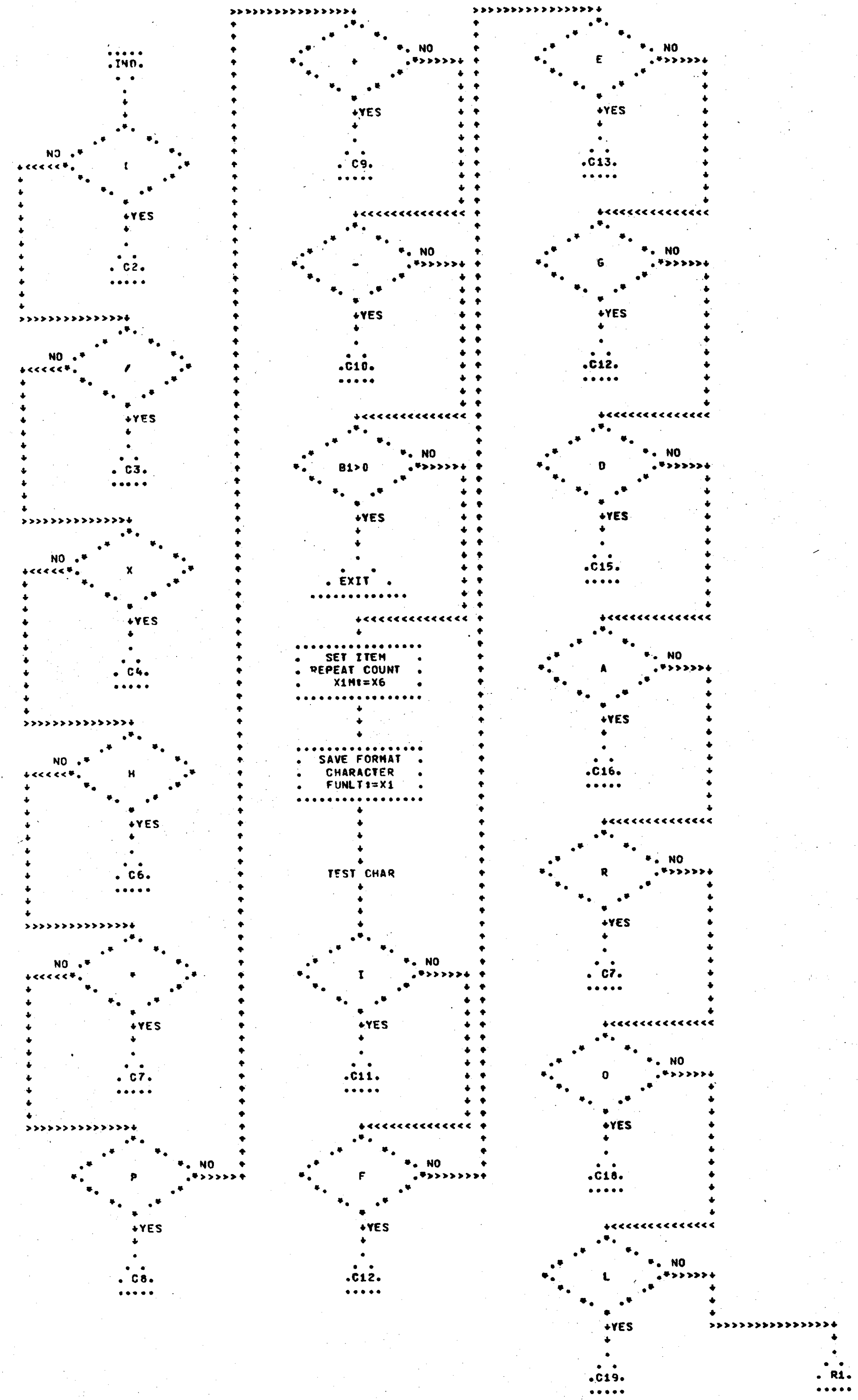

XRCL
WAIT UNTIL RA+1 IS ZERO
PLACE RCL IN RA+1
WAIT UNTIL RA+1 IS ZERO
EXIT

217

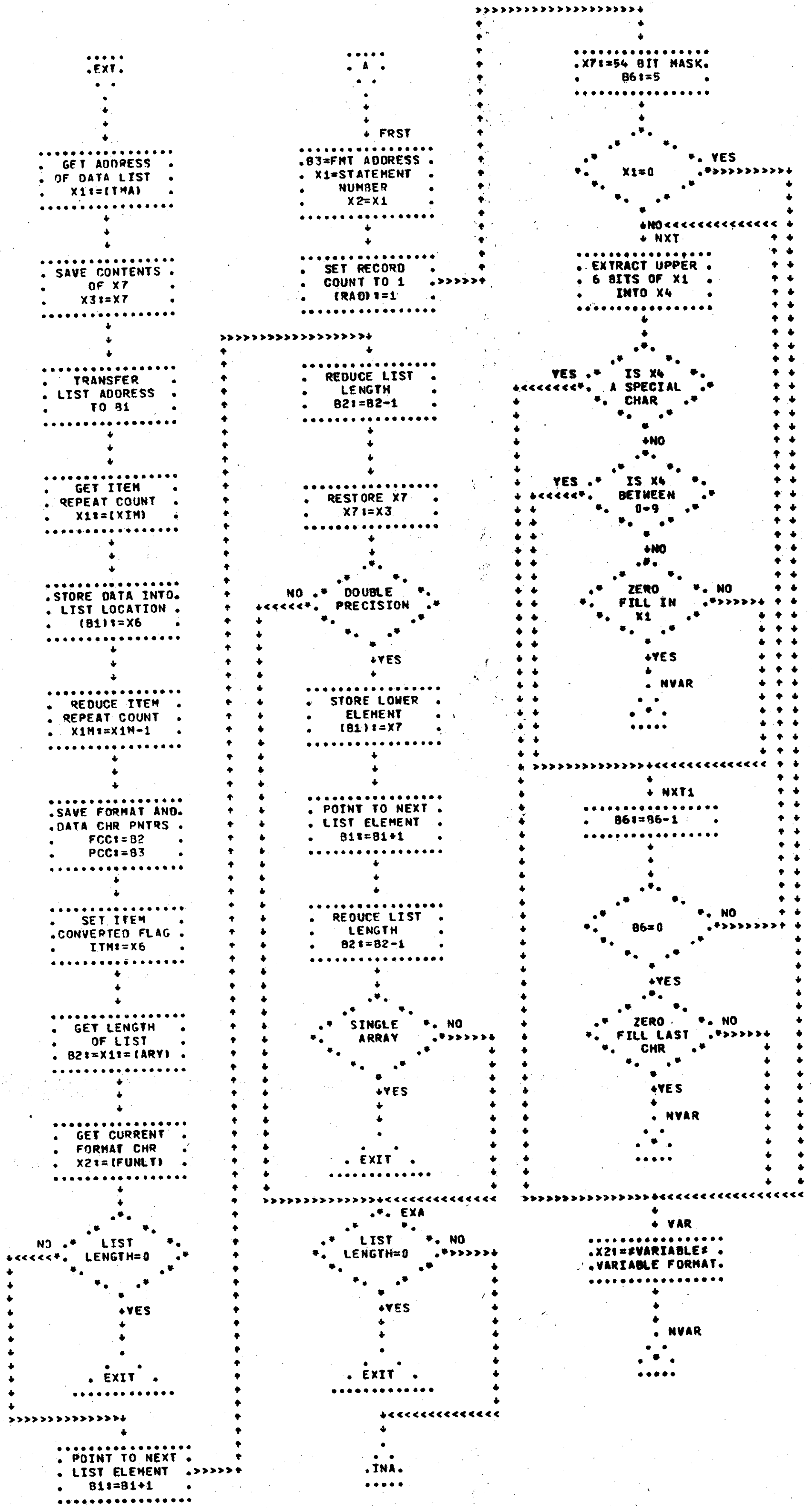
>KRAKER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL



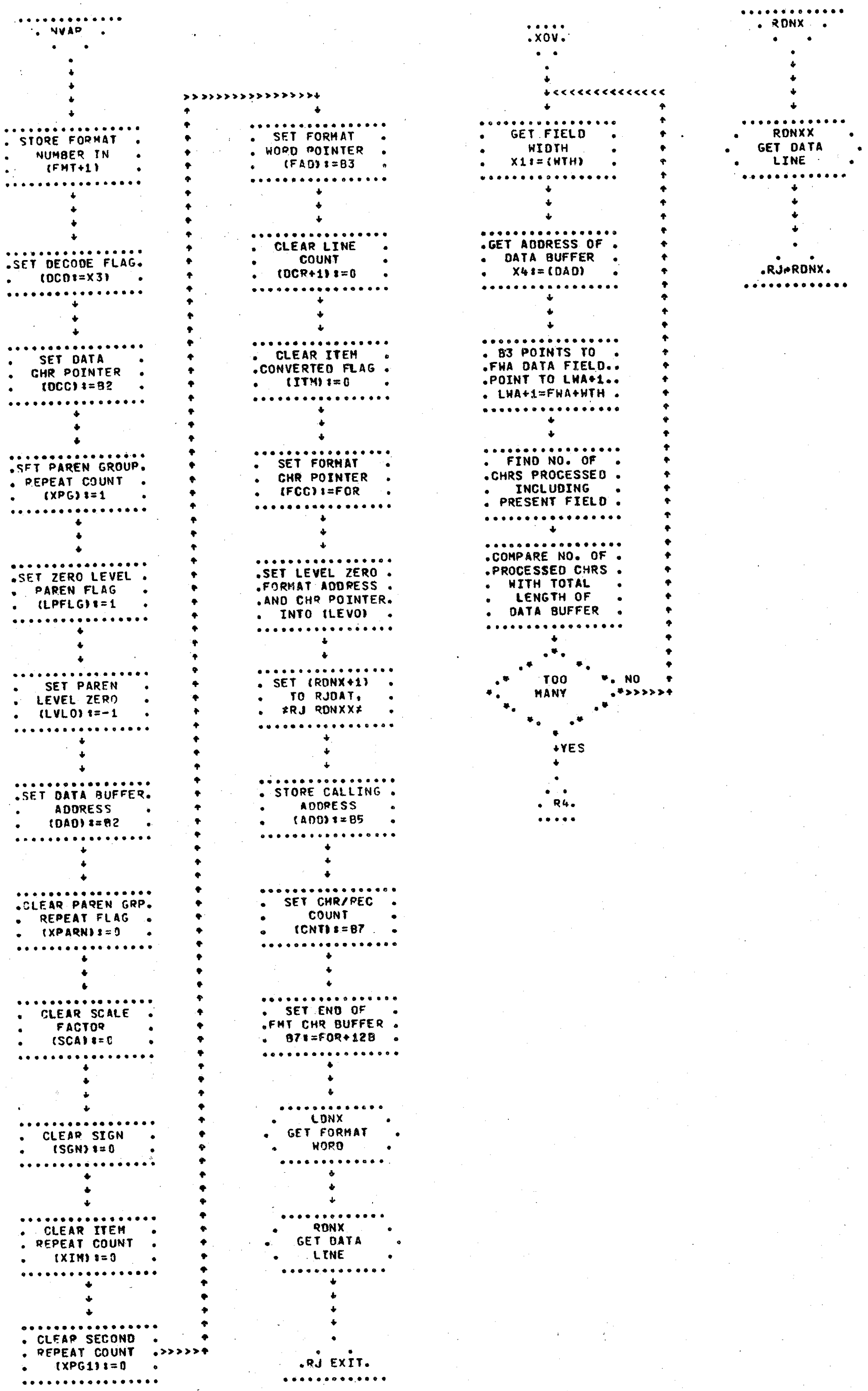
BRAKER
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL



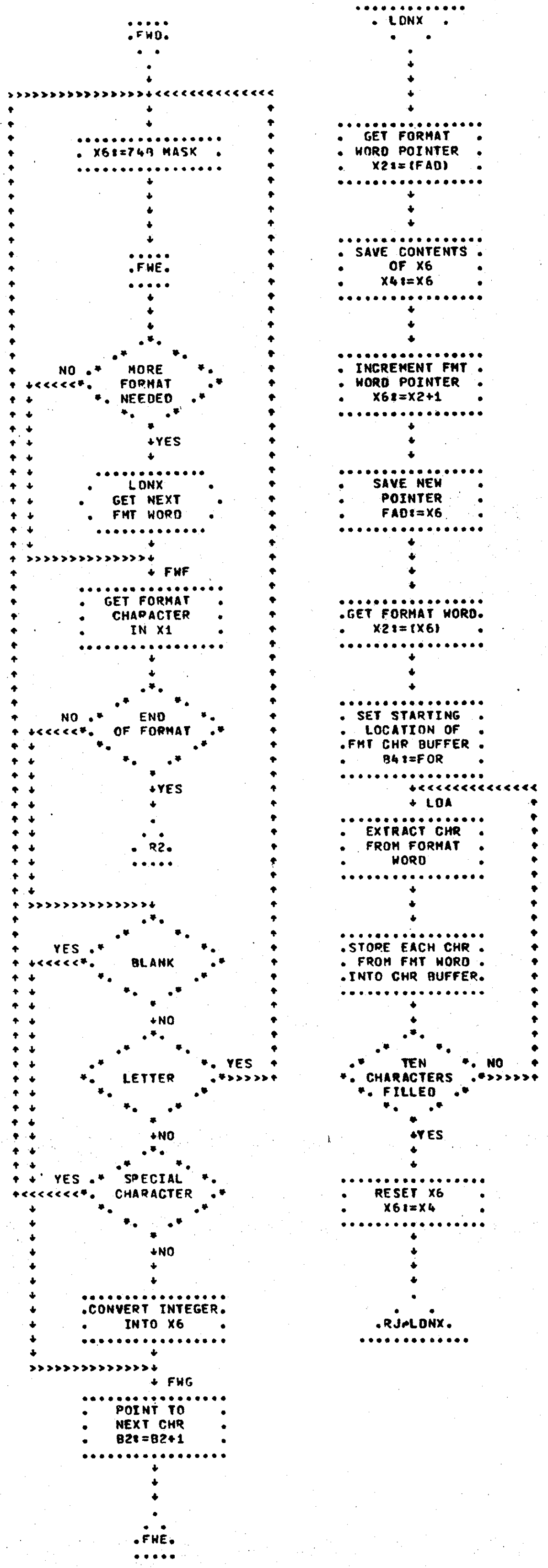
><AKER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


EXT
GET ADDRESS OF DATA LIST X1:=(TMA)
SAVE CONTENTS OF X7 X3:=X7
TRANSFER LIST ADDRESS TO B1
GET ITEM REPEAT COUNT X1:=(XIM)
STORE DATA INTO LIST LOCATION (B1):=X6
REDUCE ITEM REPEAT COUNT XIM:=XIM-1
SAVE FORMAT AND DATA CHR PNTRS FCC:=B2 PCC:=B3
SET ITEM CONVERTED FLAG ITM:=X6
GET LENGTH OF LIST B2:=X1:=(ARY)
GET CURRENT FORMAT CHR X2:=(FUNLT)
LIST LENGTH=0
NO
YES
EXIT
POINT TO NEXT LIST ELEMENT B1:=B1+1
A
FRST
B3=FMT ADDRESS X1=STATEMENT NUMBER X2=X1
SET RECORD COUNT TO 1 (RAD):=1
REDUCE LIST LENGTH B2:=B2-1
RESTORE X7 X7:=X3
DOUBLE PRECISION
NO
YES
STORE LOWER ELEMENT (B1):=X7
POINT TO NEXT LIST ELEMENT B1:=B1+1
REDUCE LIST LENGTH B2:=B2-1
SINGLE ARRAY
NO
YES
EXIT
EXA
LIST LENGTH=0
NO
YES
EXIT
INA
X7:=54 BIT MASK. B6:=5
X1=0
YES
NO
NXT
EXTRACT UPPER 6 BITS OF X1 INTO X4
IS X4 A SPECIAL CHAR
YES
NO
IS X4 BETWEEN 0-9
YES
NO
ZERO FILL IN X1
NO
YES
NVAR
NXT1
B6:=B6-1
B6=0
NO
YES
ZERO FILL LAST CHR
NO
YES
NVAR
VAR
X2:=*VARIABLE* VARIABLE FORMAT.
NVAR

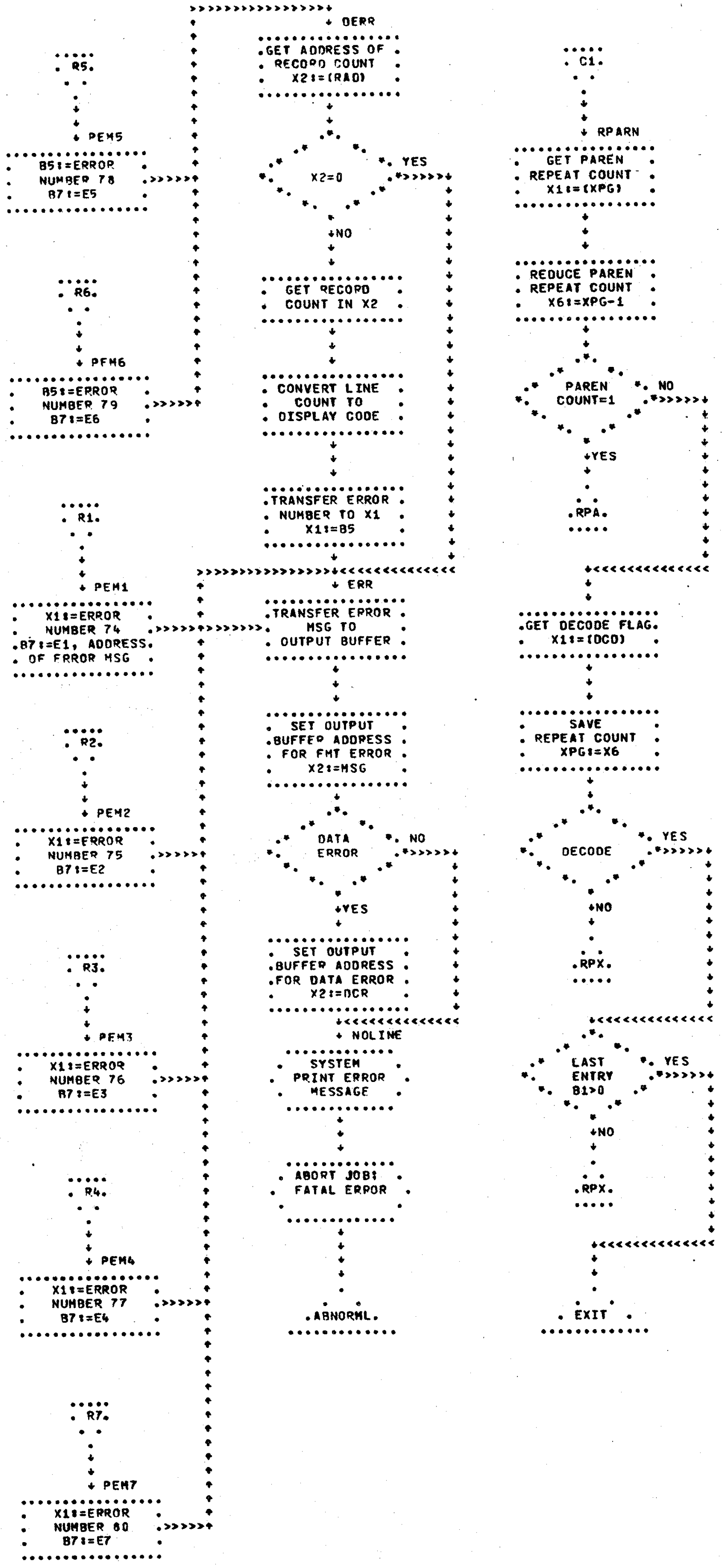
* INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


NVAP
STORE FORMAT NUMBER IN (FMT+1)
SET DECODE FLAG. (DCD)=X3)
SET DATA CHR POINTER (DCC)*=B2
SET PAREN GROUP. REPEAT COUNT (XPG)*=1
SET ZERO LEVEL PAREN FLAG (LPFLG)*=1
SET PAREN LEVEL ZERO (LVLO)*=-1
SET DATA BUFFER. ADDRESS (DAD)*=B2
CLEAR PAREN GRP. REPEAT FLAG (XPARN)*=0
CLEAR SCALE FACTOR (SCA)*=0
CLEAR SIGN (SGN)*=0
CLEAR ITEM REPEAT COUNT (XIM)*=0
CLEAR SECOND REPEAT COUNT (XPG1)*=0
SET FORMAT WORD POINTER (FAD)*=B3
CLEAR LINE COUNT (DCR+1)*=0
CLEAR ITEM CONVERTED FLAG (ITM)*=0
SET FORMAT CHR POINTER (FCC)*=FOR
SET LEVEL ZERO FORMAT ADDRESS AND CHR POINTER. INTO (LEVO)
SET (RDNX+1) TO RJDAT, ≠RJ RDNXX≠
STORE CALLING ADDRESS (ADD)*=B5
SET CHR/REC COUNT (CNT)*=B7
SET END OF FMT CHR BUFFER B7*=FOR+128
LDNX GET FORMAT WORD
RDNX GET DATA LINE
RJ EXIT.
XOV.
GET FIELD WIDTH X1*=(WTH)
GET ADDRESS OF DATA BUFFER X4*=(DAD)
B3 POINTS TO FWA DATA FIELD. POINT TO LWA+1. LWA+1=FWA+WTH
FIND NO. OF CHRS PROCESSED INCLUDING PRESENT FIELD
COMPARE NO. OF PROCESSED CHRS WITH TOTAL LENGTH OF DATA BUFFER
TOO MANY
NO
YES
R4.
RDNX
RDNXX GET DATA LINE
RJ*RDNX.

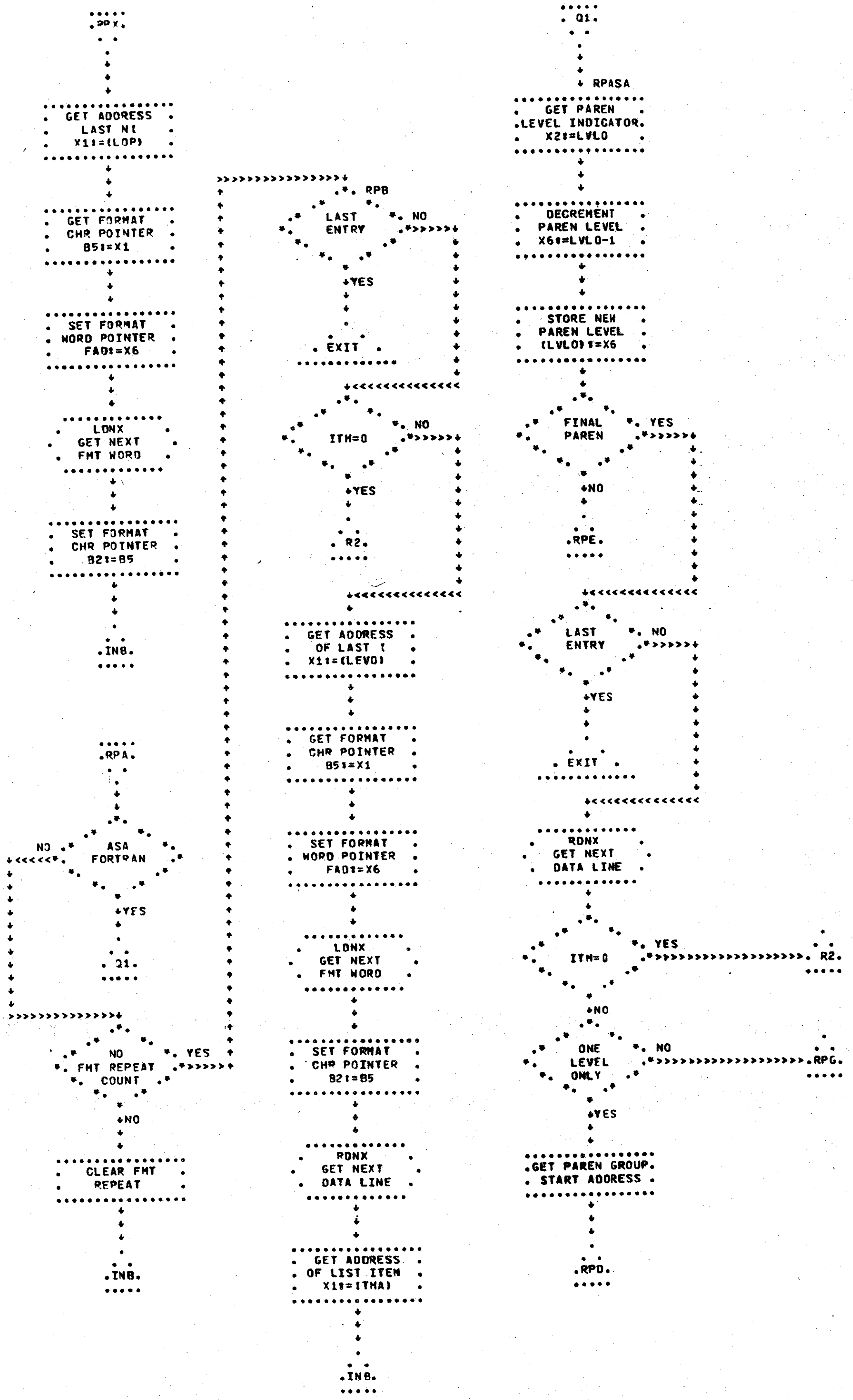
>KRAKER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


FWD.
X6‡=74B MASK
FWE.
MORE FORMAT NEEDED
NO
YES
LDNX GET NEXT FMT WORD
FWF
GET FORMAT CHARACTER IN X1
END OF FORMAT
NO
YES
R2.
BLANK
YES
NO
LETTER
YES
NO
SPECIAL CHARACTER
YES
NO
CONVERT INTEGER. INTO X6
FWG
POINT TO NEXT CHR B2‡=B2+1
FWE.
LDNX
GET FORMAT WORD POINTER X2‡=(FAD)
SAVE CONTENTS OF X6 X4‡=X6
INCREMENT FMT WORD POINTER X6‡=X2+1
SAVE NEW POINTER FAD‡=X6
GET FORMAT WORD. X2‡=(X6)
SET STARTING LOCATION OF FMT CHR BUFFER B4‡=FOR
LDA
EXTRACT CHR FROM FORMAT WORD
STORE EACH CHR FROM FMT WORD INTO CHR BUFFER.
TEN CHARACTERS FILLED
NO
YES
RESET X6 X6‡=X4
RJ+LDNX.

REMARKS
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL

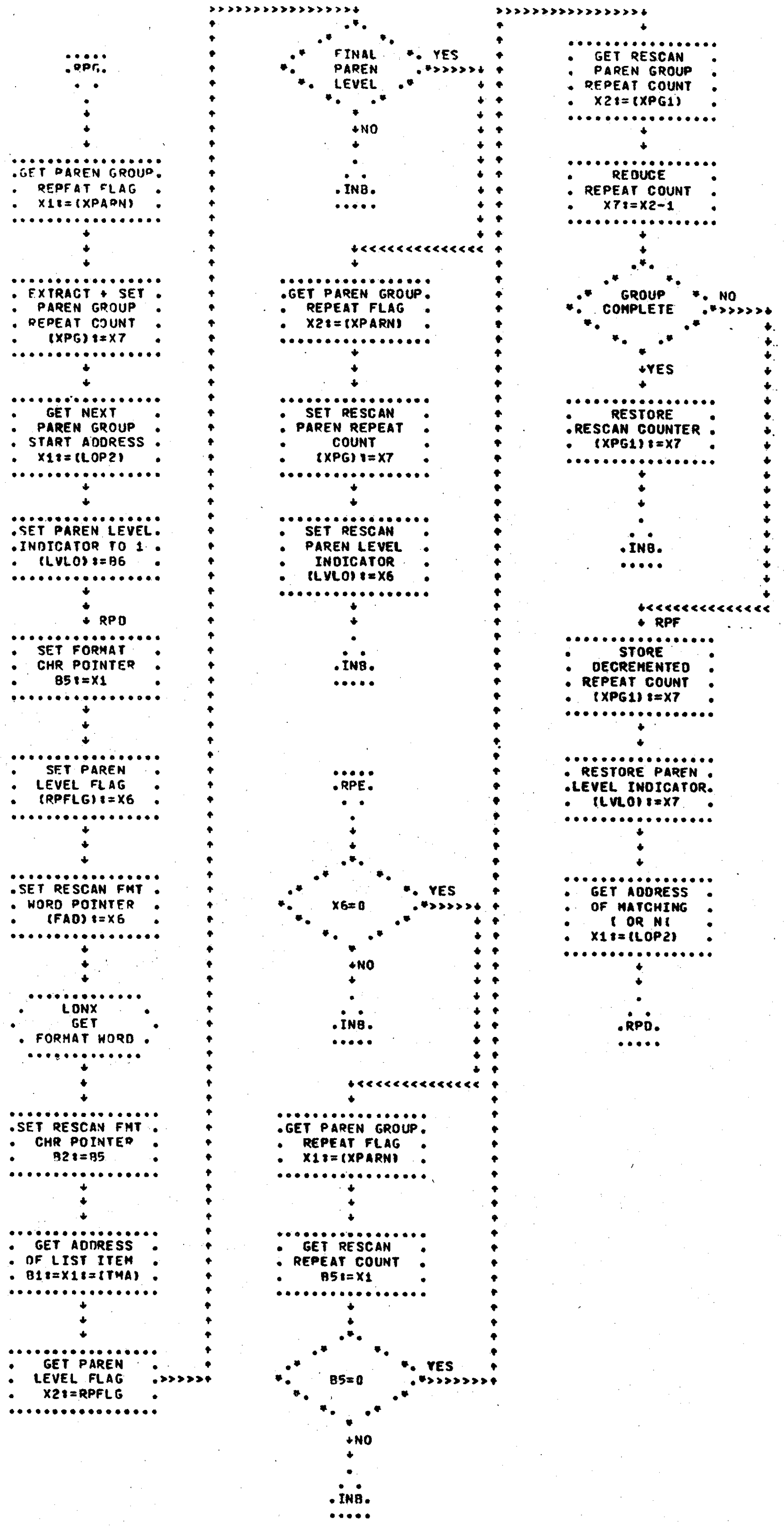

OERR
GET ADDRESS OF RECORD COUNT X2:=(RAD)
X2=0
YES
NO
GET RECORD COUNT IN X2
CONVERT LINE COUNT TO DISPLAY CODE
TRANSFER ERROR NUMBER TO X1 X1:=B5
R5.
PEM5
B5:=ERROR NUMBER 78 B7:=E5
R6.
PEM6
B5:=ERROR NUMBER 79 B7:=E6
R1.
PEM1
X1:=ERROR NUMBER 74 B7:=E1, ADDRESS OF ERROR MSG
ERR
TRANSFER ERROR MSG TO OUTPUT BUFFER
SET OUTPUT BUFFER ADDRESS FOR FMT ERROR X2:=MSG
DATA ERROR
NO
YES
SET OUTPUT BUFFER ADDRESS FOR DATA ERROR X2:=DCR
NOLINE
SYSTEM PRINT ERROR MESSAGE
ABORT JOB: FATAL ERROR
ABNORML.
R2.
PEM2
X1:=ERROR NUMBER 75 B7:=E2
R3.
PEM3
X1:=ERROR NUMBER 76 B7:=E3
R4.
PEM4
X1:=ERROR NUMBER 77 B7:=E4
R7.
PEM7
X1:=ERROR NUMBER 80 B7:=E7
C1.
RPARN
GET PAREN REPEAT COUNT X1:=(XPG)
REDUCE PAREN REPEAT COUNT X6:=XPG-1
PAREN COUNT=1
NO
YES
RPA.
GET DECODE FLAG. X1:=(DCD)
SAVE REPEAT COUNT XPG:=X6
DECODE
YES
NO
RPX.
LAST ENTRY B1>0
YES
NO
RPX.
EXIT

KRAKER
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL

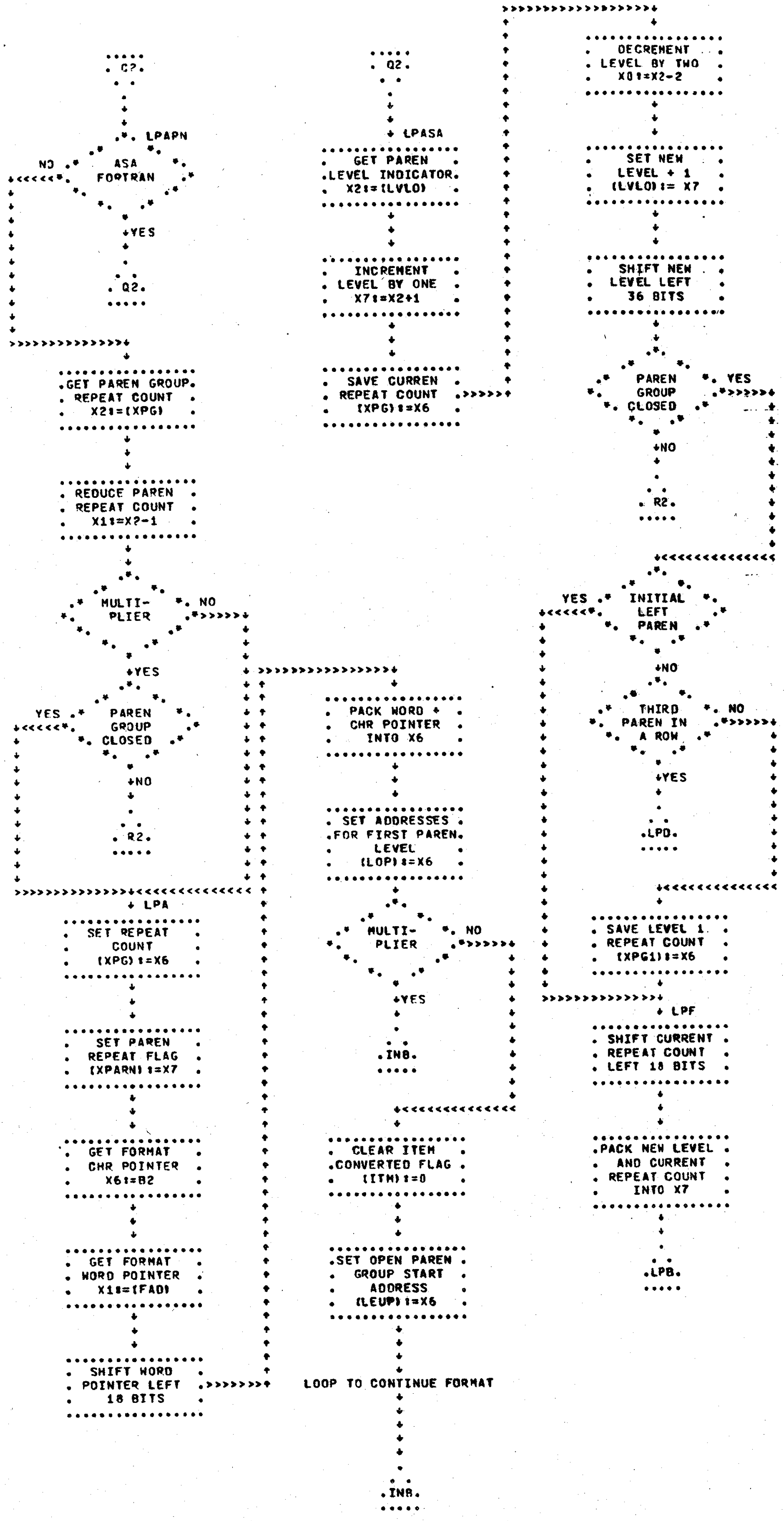

RPX
GET ADDRESS LAST NI X1:=(LOP)
GET FORMAT CHR POINTER B5:=X1
SET FORMAT WORD POINTER FAD:=X6
LDNX GET NEXT FMT WORD
SET FORMAT CHR POINTER B2:=B5
INB
RPA
ASA FORTRAN
NO
YES
Q1
FMT REPEAT COUNT
NO
YES
NO
CLEAR FMT REPEAT
INB
RPB
LAST ENTRY
NO
YES
EXIT
ITM=0
NO
YES
R2
GET ADDRESS OF LAST (X1:=(LEVO)
GET FORMAT CHR POINTER B5:=X1
SET FORMAT WORD POINTER FAD:=X6
LDNX GET NEXT FMT WORD
SET FORMAT CHR POINTER B2:=B5
RDNX GET NEXT DATA LINE
GET ADDRESS OF LIST ITEM X1:=(TMA)
INB
Q1
RPASA
GET PAREN LEVEL INDICATOR X2:=LVLO
DECREMENT PAREN LEVEL X6:=LVLO-1
STORE NEW PAREN LEVEL (LVLO):=X6
FINAL PAREN
YES
NO
RPE
LAST ENTRY
NO
YES
EXIT
RDNX GET NEXT DATA LINE
ITM=0
YES
R2
NO
ONE LEVEL ONLY
NO
RPG
YES
GET PAREN GROUP START ADDRESS
RPD

>KRAKER<

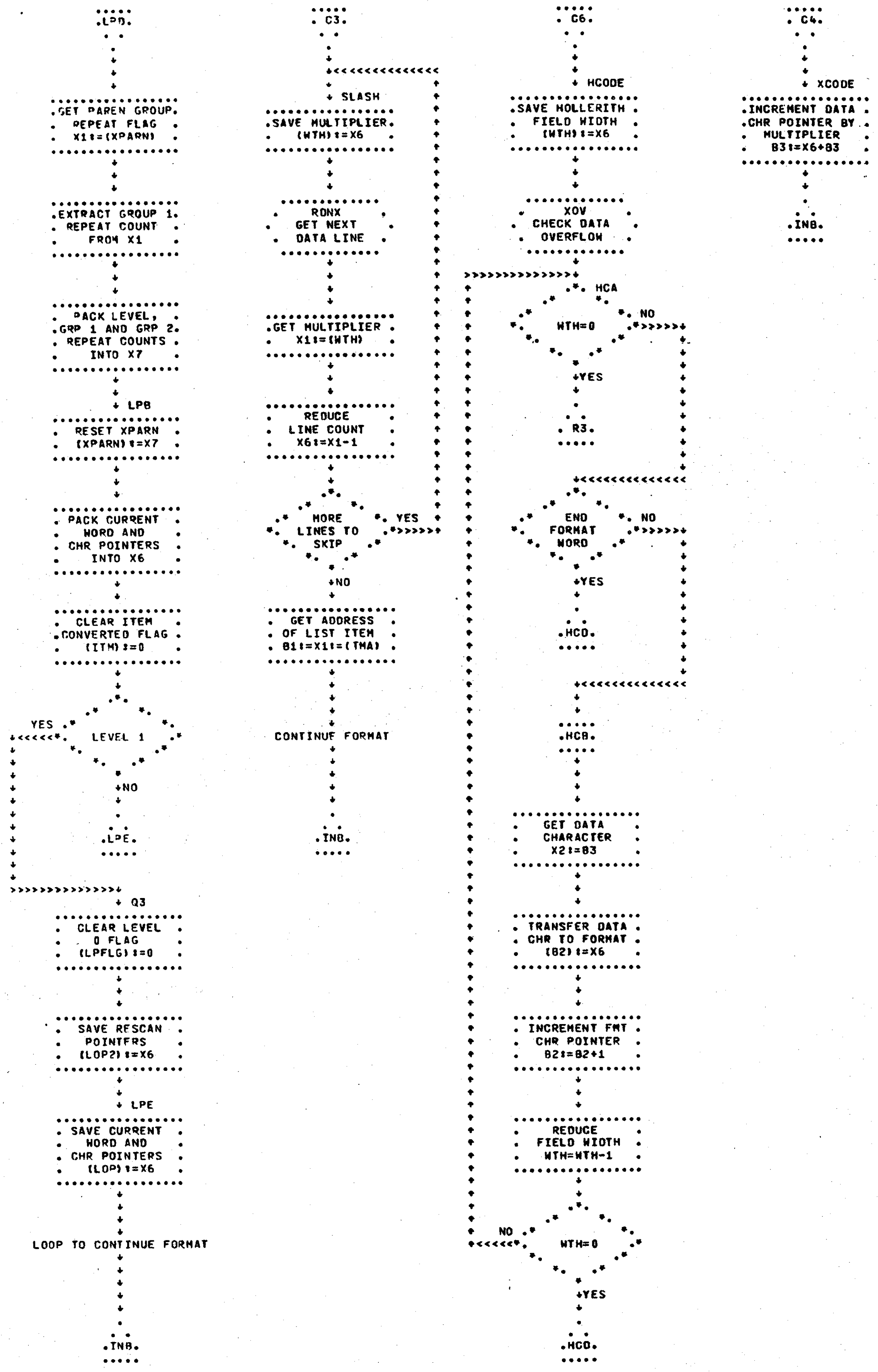
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


RPG.
GET PAREN GROUP REPEAT FLAG X1:=(XPARN)
EXTRACT + SET PAREN GROUP REPEAT COUNT (XPG):=X7
GET NEXT PAREN GROUP START ADDRESS X1:=(LOP2)
SET PAREN LEVEL INDICATOR TO 1 (LVLO):=B6
RPD
SET FORMAT CHR POINTER B5:=X1
SET PAREN LEVEL FLAG (RPFLG):=X6
SET RESCAN FMT WORD POINTER (FAD):=X6
LONX GET FORMAT WORD
SET RESCAN FMT CHR POINTER B2:=B5
GET ADDRESS OF LIST ITEM B1:=X1:=(TMA)
GET PAREN LEVEL FLAG X2:=RPFLG
FINAL PAREN LEVEL
YES
NO
INB.
GET PAREN GROUP REPEAT FLAG X2:=(XPARN)
SET RESCAN PAREN REPEAT COUNT (XPG):=X7
SET RESCAN PAREN LEVEL INDICATOR (LVLO):=X6
INB.
RPE.
X6=0
YES
NO
INB.
GET PAREN GROUP REPEAT FLAG X1:=(XPARN)
GET RESCAN REPEAT COUNT B5:=X1
B5=0
YES
NO
INB.
GET RESCAN PAREN GROUP REPEAT COUNT X2:=(XPG1)
REDUCE REPEAT COUNT X7:=X2-1
GROUP COMPLETE
NO
YES
RESTORE RESCAN COUNTER (XPG1):=X7
INB.
RPF
STORE DECREMENTED REPEAT COUNT (XPG1):=X7
RESTORE PAREN LEVEL INDICATOR (LVLO):=X7
GET ADDRESS OF MATCHING (OR N(X1:=(LOP2)
RPD.

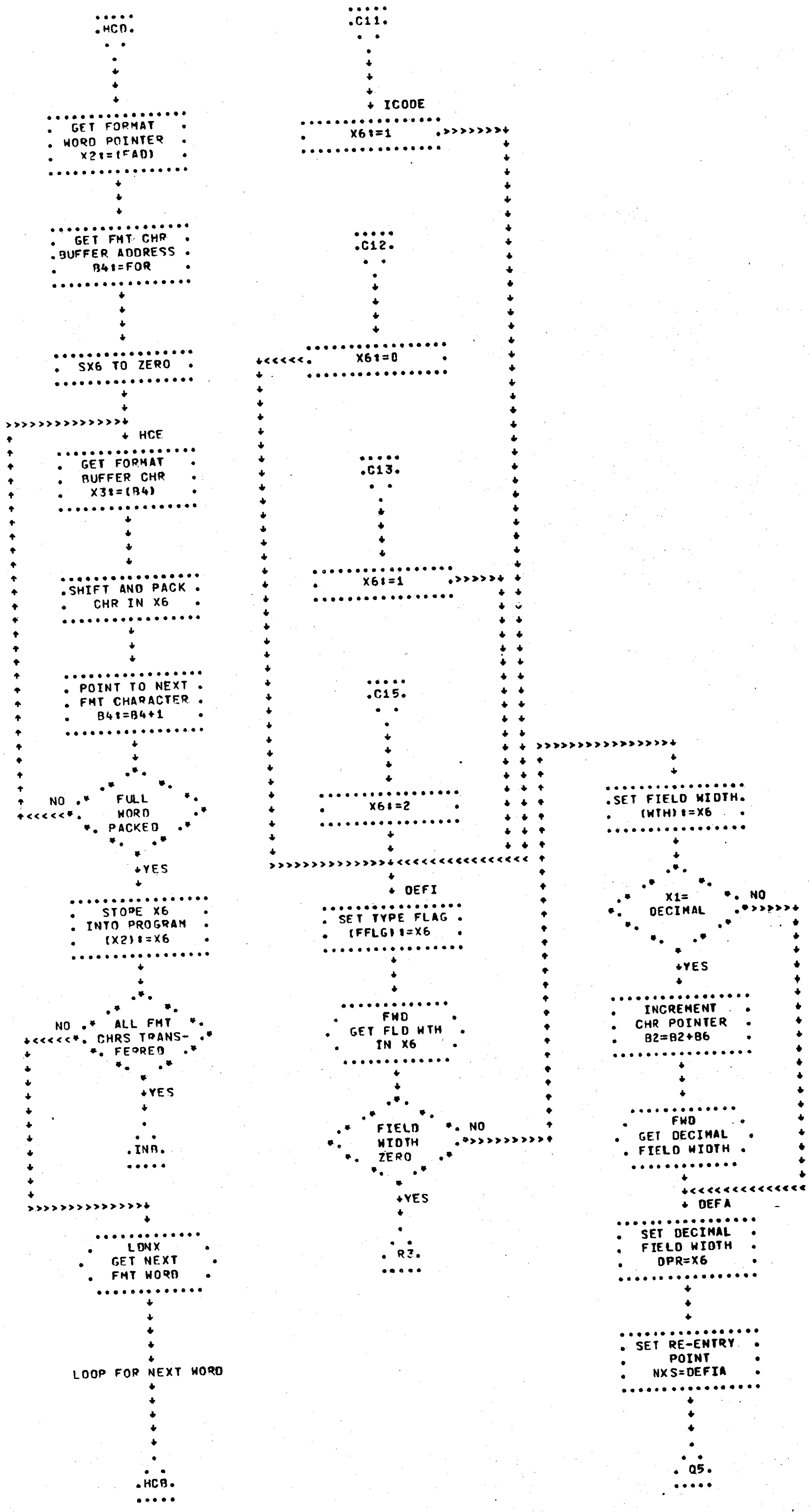
>VPAKEP<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


C2.
LPAPN
ASA FORTRAN
NO
YES
Q2.
GET PAREN GROUP REPEAT COUNT X2:=(XPG)
REDUCE PAREN REPEAT COUNT X1:=X2-1
MULTI- PLIER
NO
YES
PAREN GROUP CLOSED
YES
NO
R2.
LPA
SET REPEAT COUNT (XPG):=X6
SET PAREN REPEAT FLAG (XPARN):=X7
GET FORMAT CHR POINTER X6:=B2
GET FORMAT WORD POINTER X1:=(FAD)
SHIFT WORD POINTER LEFT 18 BITS
Q2.
LPASA
GET PAREN LEVEL INDICATOR. X2:=(LVLO)
INCREMENT LEVEL BY ONE X7:=X2+1
SAVE CURREN REPEAT COUNT (XPG):=X6
PACK WORD + CHR POINTER INTO X6
SET ADDRESSES FOR FIRST PAREN. LEVEL (LOP):=X6
MULTI- PLIER
NO
YES
INB.
CLEAR ITEM CONVERTED FLAG (ITM):=0
SET OPEN PAREN GROUP START ADDRESS (LEUP):=X6
LOOP TO CONTINUE FORMAT
INB.
DECREMENT LEVEL BY TWO X0:=X2-2
SET NEW LEVEL + 1 (LVLO):= X7
SHIFT NEW LEVEL LEFT 36 BITS
PAREN GROUP CLOSED
YES
NO
R2.
INITIAL LEFT PAREN
YES
NO
THIRD PAREN IN A ROW
NO
YES
LPD.
SAVE LEVEL 1 REPEAT COUNT (XPG1):=X6
LPF
SHIFT CURRENT REPEAT COUNT LEFT 18 BITS
PACK NEW LEVEL AND CURRENT REPEAT COUNT INTO X7
LPB.

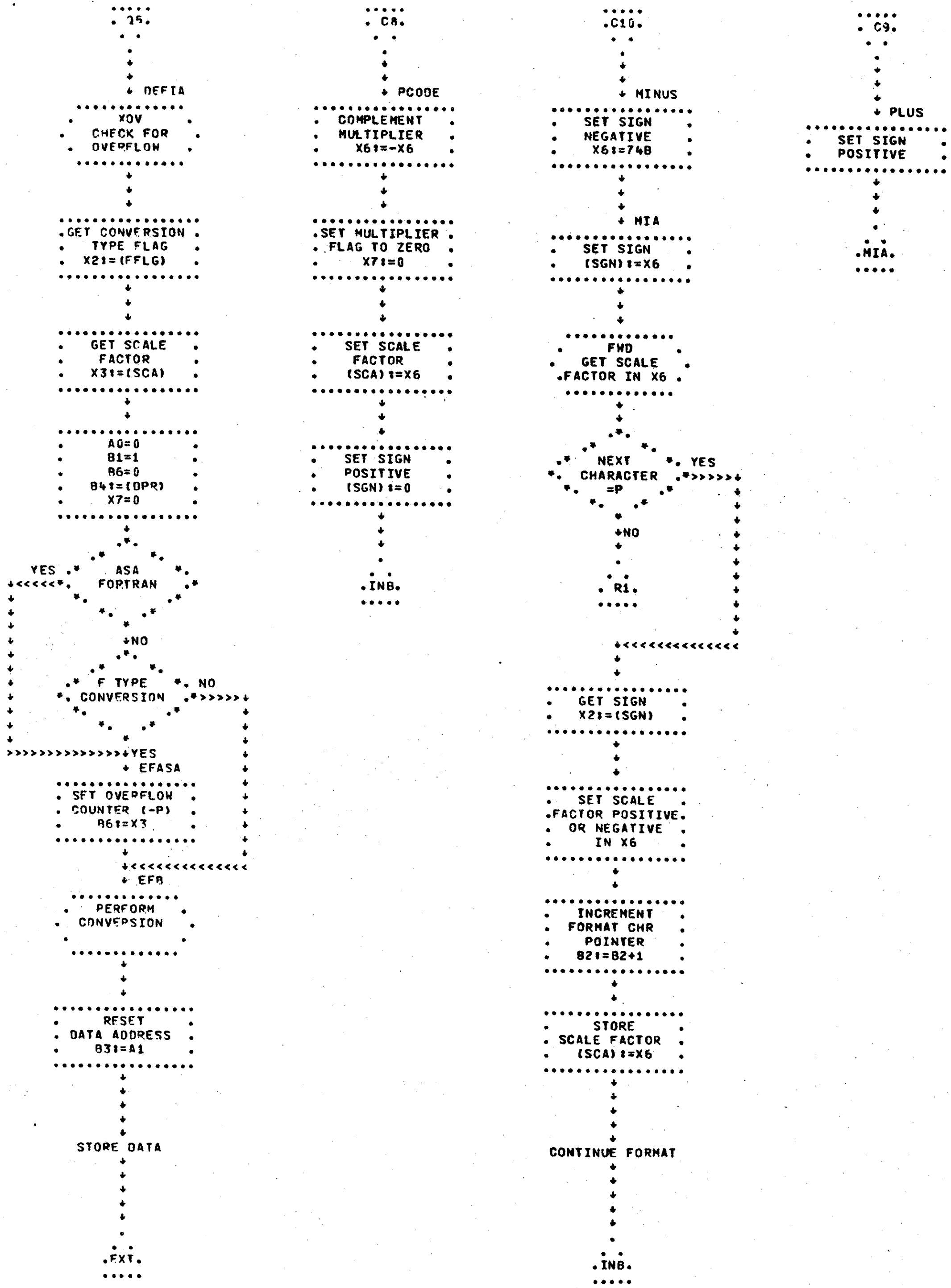
[illegible]
THE * [illegible] CONNECTORS STANDS FOR THE ASSOCIATED LABEL


LPD
GET PAREN GROUP REPEAT FLAG X1:=(XPARN)
EXTRACT GROUP 1 REPEAT COUNT FROM X1
PACK LEVEL, GRP 1 AND GRP 2 REPEAT COUNTS INTO X7
LP8
RESET XPARN (XPARN):=X7
PACK CURRENT WORD AND CHR POINTERS INTO X6
CLEAR ITEM CONVERTED FLAG (ITM):=0
LEVEL 1
YES
NO
LPE
Q3
CLEAR LEVEL 0 FLAG (LPFLG):=0
SAVE RESCAN POINTERS (LOP2):=X6
LPE
SAVE CURRENT WORD AND CHR POINTERS (LOP):=X6
LOOP TO CONTINUE FORMAT
INB
C3
SLASH
SAVE MULTIPLIER (WTH):=X6
RDNX GET NEXT DATA LINE
GET MULTIPLIER X1:=(WTH)
REDUCE LINE COUNT X6:=X1-1
MORE LINES TO SKIP
YES
NO
GET ADDRESS OF LIST ITEM B1:=X1:=(TMA)
CONTINUE FORMAT
INB
C6
HCODE
SAVE HOLLERITH FIELD WIDTH (WTH):=X6
XOV CHECK DATA OVERFLOW
HCA
WTH=0
NO
YES
R3
END FORMAT WORD
NO
YES
HCD
HCB
GET DATA CHARACTER X2:=B3
TRANSFER DATA CHR TO FORMAT (B2):=X6
INCREMENT FMT CHR POINTER B2:=B2+1
REDUCE FIELD WIDTH WTH=WTH-1
WTH=0
NO
YES
HCD
C4
XCODE
INCREMENT DATA CHR POINTER BY MULTIPLIER B3:=X6+B3
INB

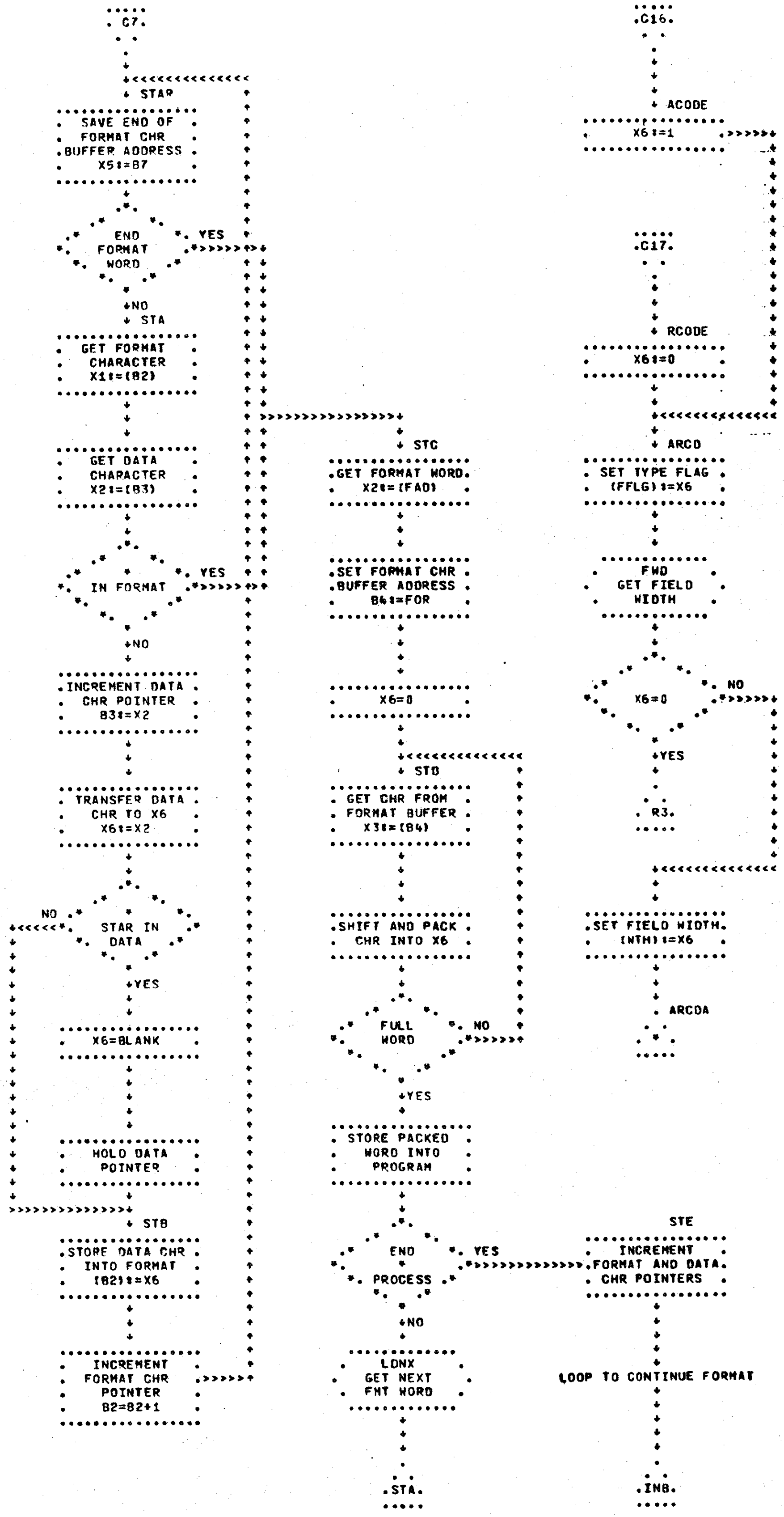
>KPAKER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


HCD
GET FORMAT WORD POINTER X2↑=(FAD)
GET FMT CHR BUFFER ADDRESS B4↑=FOR
SX6 TO ZERO
HCE
GET FORMAT BUFFER CHR X3↑=(B4)
SHIFT AND PACK CHR IN X6
POINT TO NEXT FMT CHARACTER B4↑=B4+1
FULL WORD PACKED
NO
YES
STOPE X6 INTO PROGRAM (X2)↑=X6
ALL FMT CHRS TRANS-FERRED
NO
YES
INB
LDNX GET NEXT FMT WORD
LOOP FOR NEXT WORD
HCB
C11
ICODE
X6↑=1
C12
X6↑=0
C13
X6↑=1
C15
X6↑=2
DEFI
SET TYPE FLAG (FFLG)↑=X6
FWD GET FLD WTH IN X6
FIELD WIDTH ZERO
NO
YES
R3
SET FIELD WIDTH. (WTH)↑=X6
X1= DECIMAL
NO
YES
INCREMENT CHR POINTER B2=B2+B6
FWD GET DECIMAL FIELD WIDTH
DEFA
SET DECIMAL FIELD WIDTH DPR=X6
SET RE-ENTRY POINT NXS=DEFIA
Q5

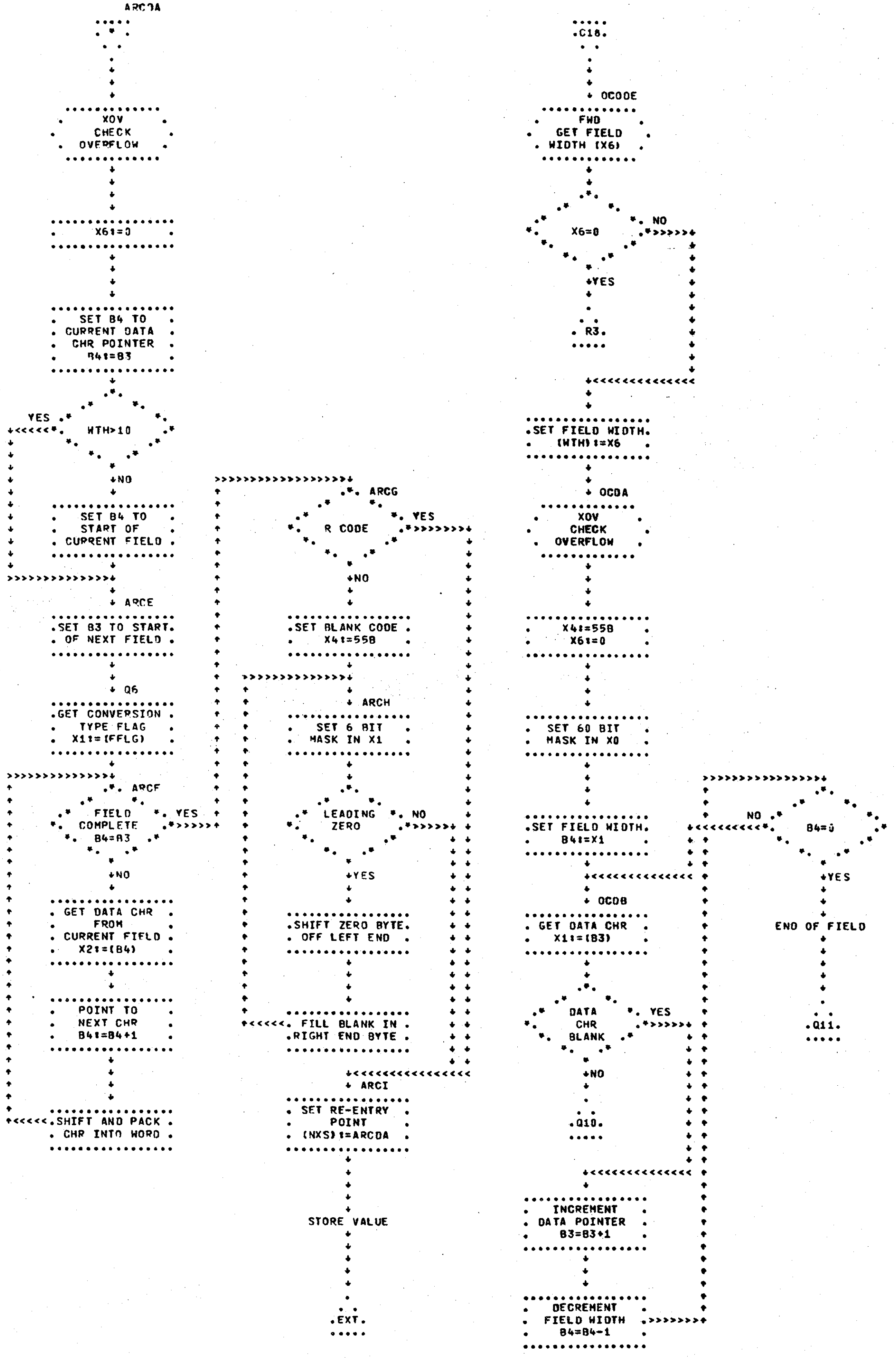
KRSVFR
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL



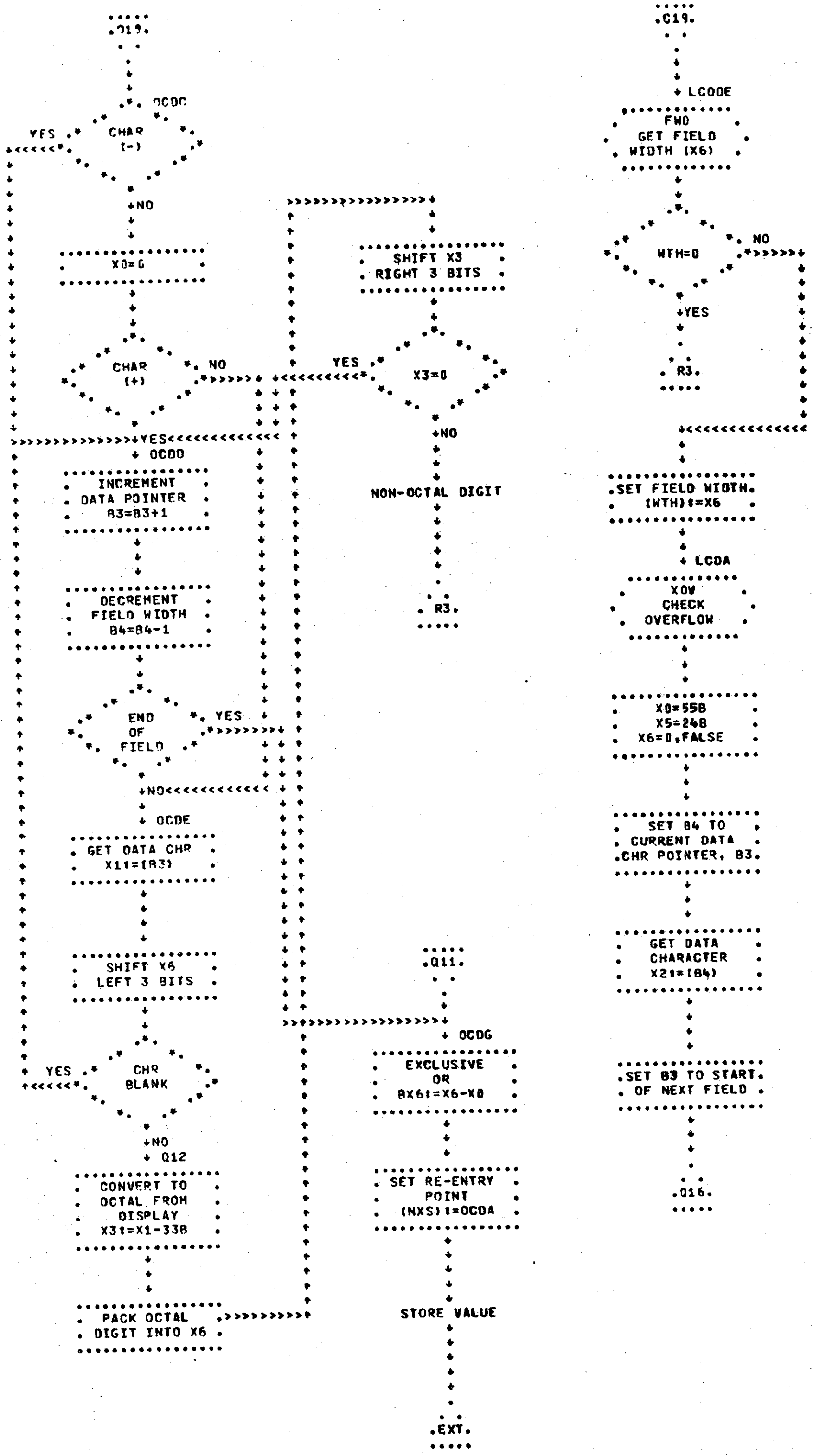
>KRAKER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


C7.
STAR
SAVE END OF FORMAT CHR BUFFER ADDRESS X5:=B7
END FORMAT WORD
YES
NO
STA
GET FORMAT CHARACTER X1:=(B2)
GET DATA CHARACTER X2:=(B3)
IN FORMAT
YES
NO
INCREMENT DATA CHR POINTER B3:=X2
TRANSFER DATA CHR TO X6 X6:=X2
STAR IN DATA
NO
YES
X6=BLANK
HOLD DATA POINTER
STB
STORE DATA CHR INTO FORMAT (B2):=X6
INCREMENT FORMAT CHR POINTER B2=B2+1
STC
GET FORMAT WORD. X2:=(FAD)
SET FORMAT CHR BUFFER ADDRESS B4:=FOR
X6=0
STD
GET CHR FROM FORMAT BUFFER X3:=(B4)
SHIFT AND PACK CHR INTO X6
FULL WORD
NO
YES
STORE PACKED WORD INTO PROGRAM
END PROCESS
YES
NO
LDNX GET NEXT FMT WORD
STA.
C16.
ACODE
X6:=1
C17.
RCODE
X6:=0
ARCD
SET TYPE FLAG (FFLG):=X6
FWD GET FIELD WIDTH
X6=0
NO
YES
R3.
SET FIELD WIDTH. (WTH):=X6
ARCDA
STE
INCREMENT FORMAT AND DATA. CHR POINTERS
LOOP TO CONTINUE FORMAT
INB.

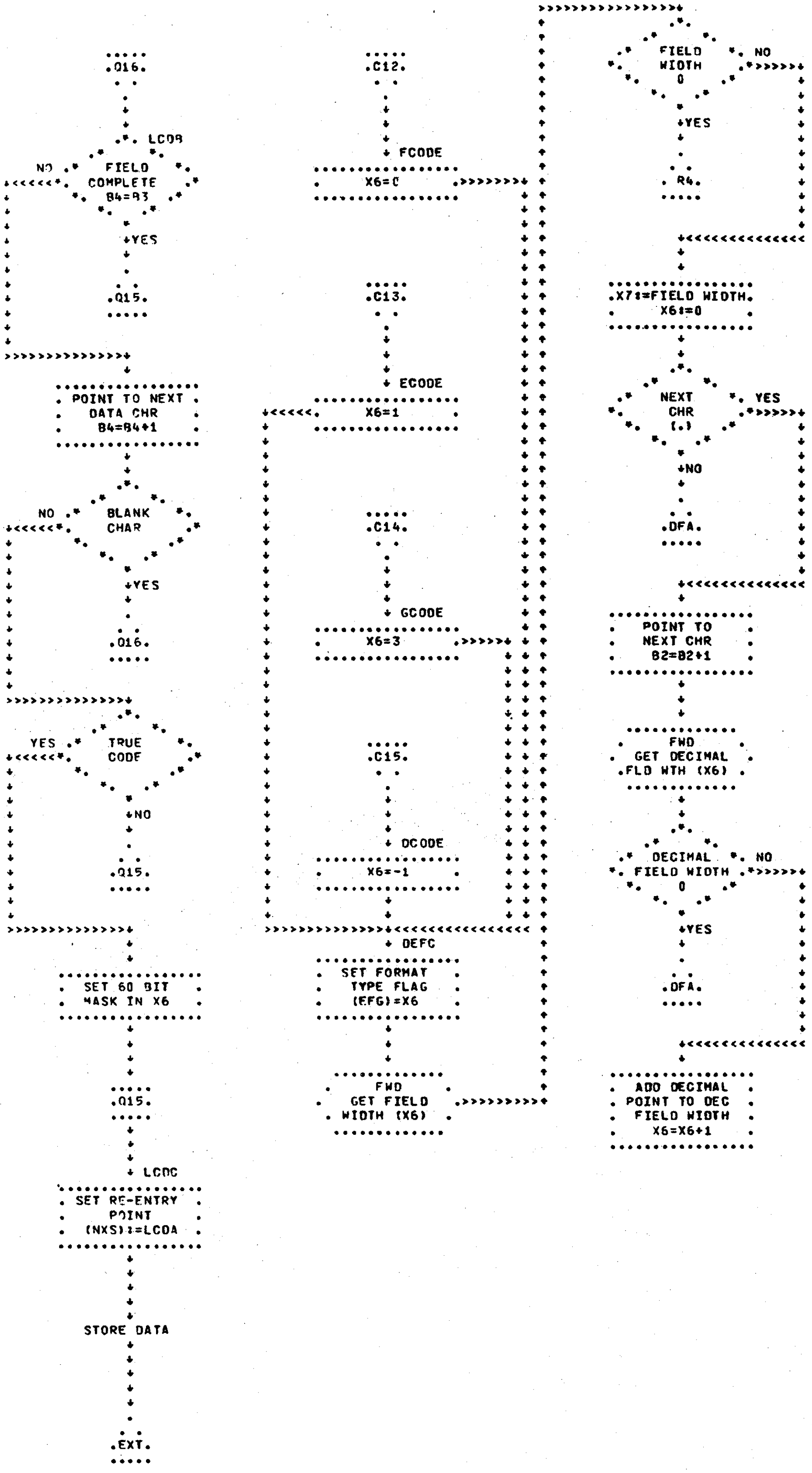
[illegible]
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


ARCDA
XOV CHECK OVERFLOW
X6:=0
SET B4 TO CURRENT DATA CHR POINTER B4:=B3
YES
WTH>10
NO
SET B4 TO START OF CURRENT FIELD
ARCE
SET B3 TO START OF NEXT FIELD
Q6
GET CONVERSION TYPE FLAG X1:=(FFLG)
ARCF
FIELD COMPLETE B4=B3
YES
NO
GET DATA CHR FROM CURRENT FIELD X2:=(B4)
POINT TO NEXT CHR B4:=B4+1
SHIFT AND PACK CHR INTO WORD
ARCG
R CODE
YES
NO
SET BLANK CODE X4:=55B
ARCH
SET 6 BIT MASK IN X1
LEADING ZERO
NO
YES
SHIFT ZERO BYTE OFF LEFT END
FILL BLANK IN RIGHT END BYTE
ARCI
SET RE-ENTRY POINT (NXS):=ARCDA
STORE VALUE
EXT
C16
OCODE
FWD GET FIELD WIDTH (X6)
X6=0
NO
YES
R3
SET FIELD WIDTH (WTH):=X6
OCDA
XOV CHECK OVERFLOW
X4:=55B X6:=0
SET 60 BIT MASK IN X0
SET FIELD WIDTH B4:=X1
NO
B4=0
YES
END OF FIELD
Q11
OCDB
GET DATA CHR X1:=(B3)
DATA CHR BLANK
YES
NO
Q10
INCREMENT DATA POINTER B3=B3+1
DECREMENT FIELD WIDTH B4=B4-1

>KRAKER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL

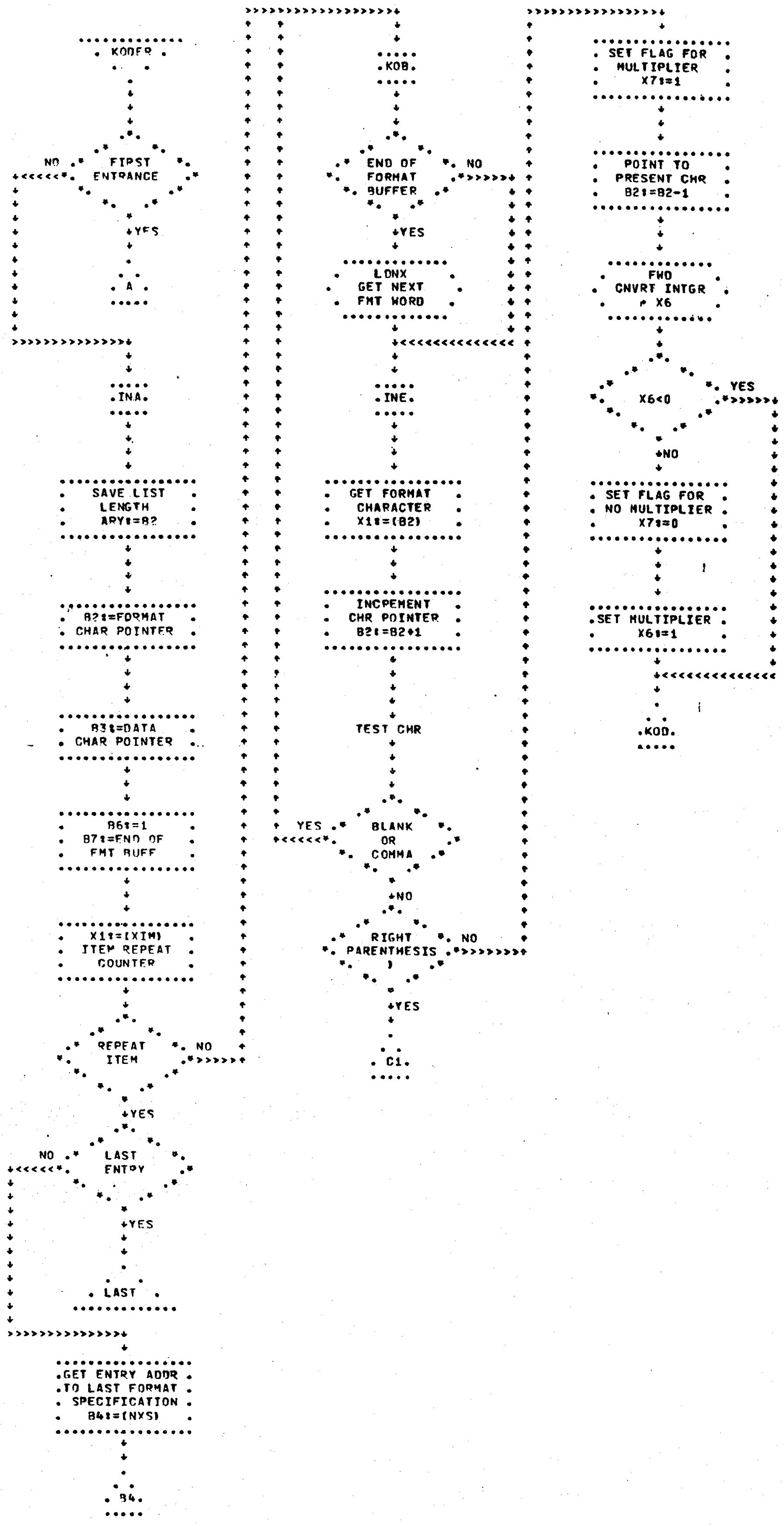


BRAKER
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL

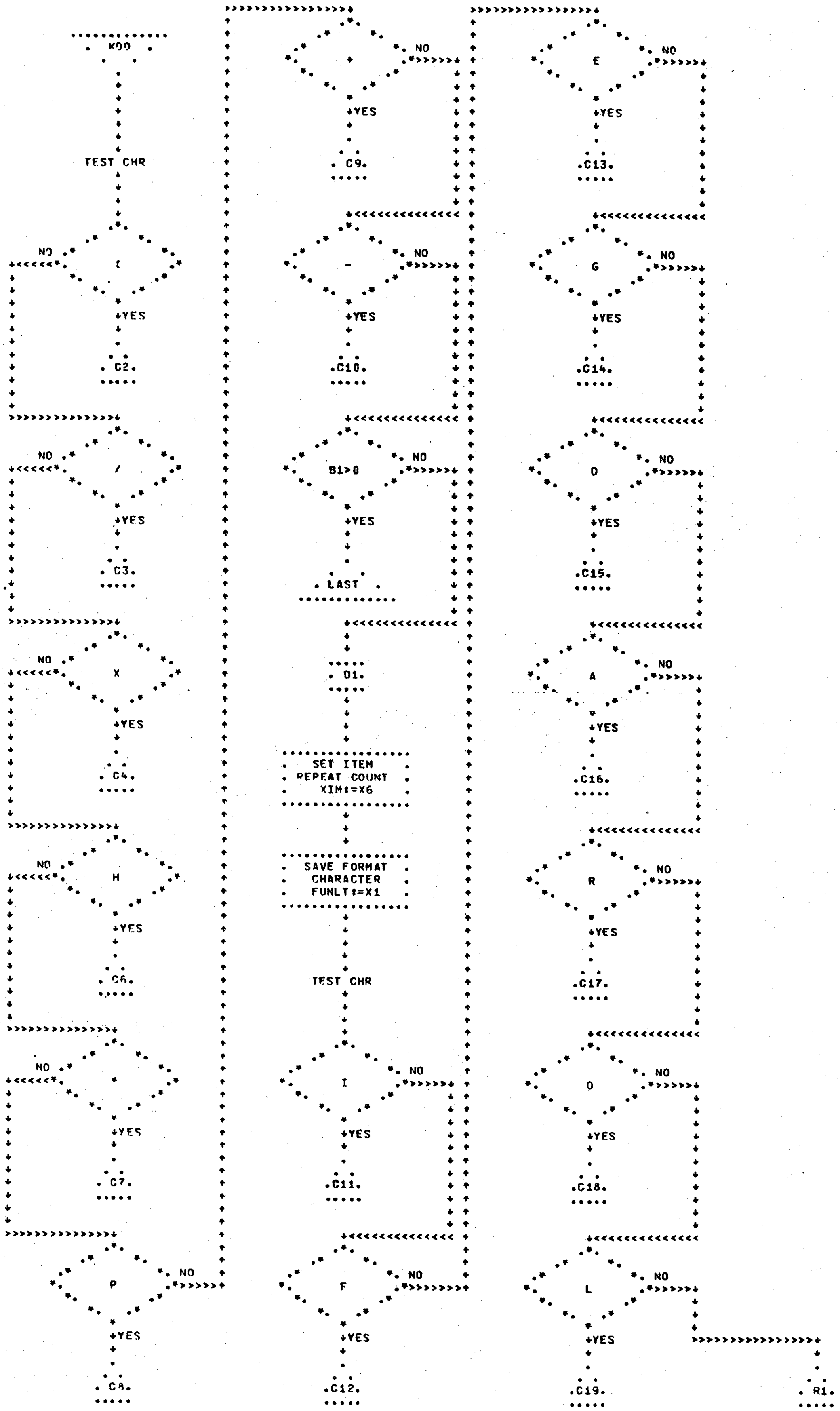


>KODER<

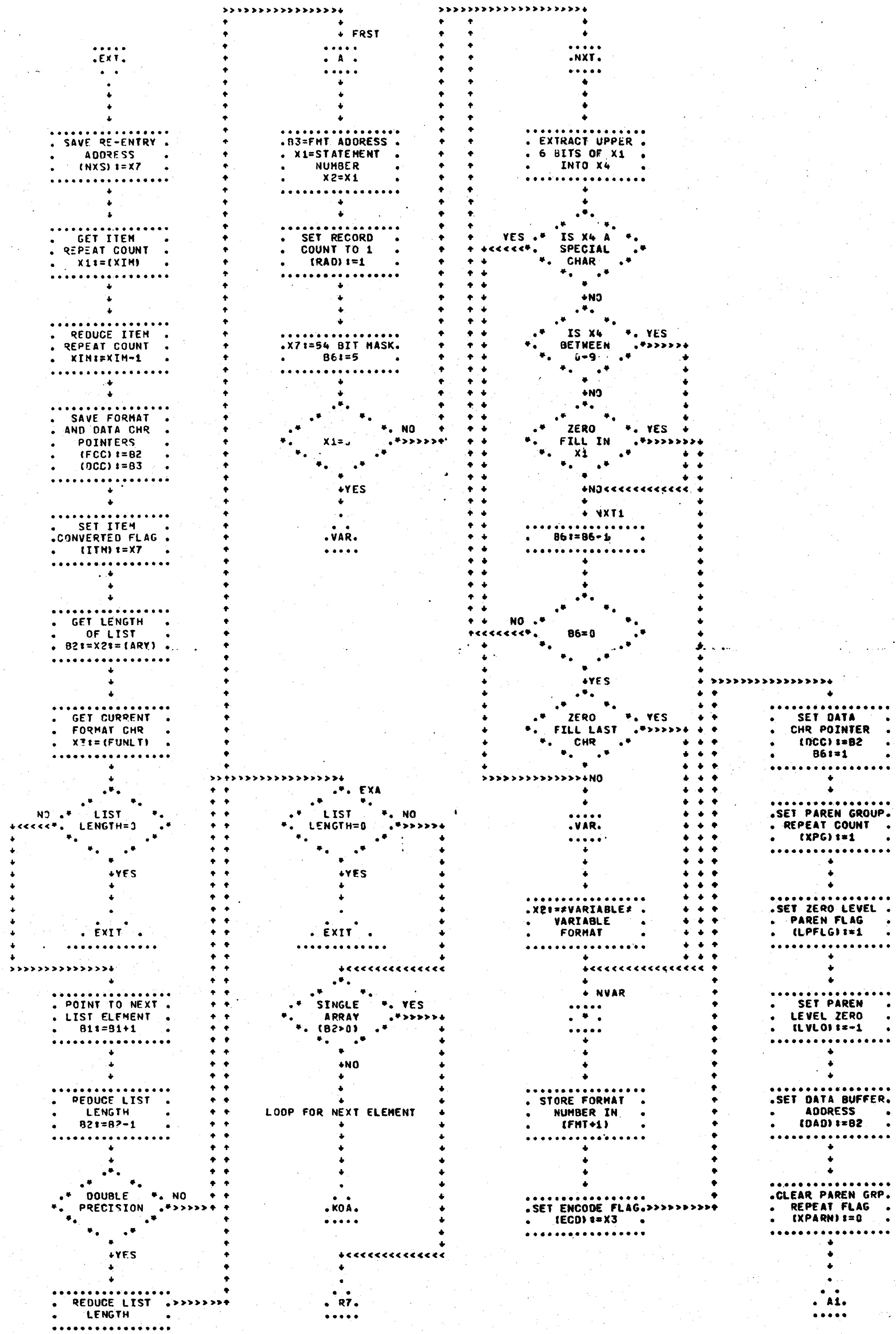
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL



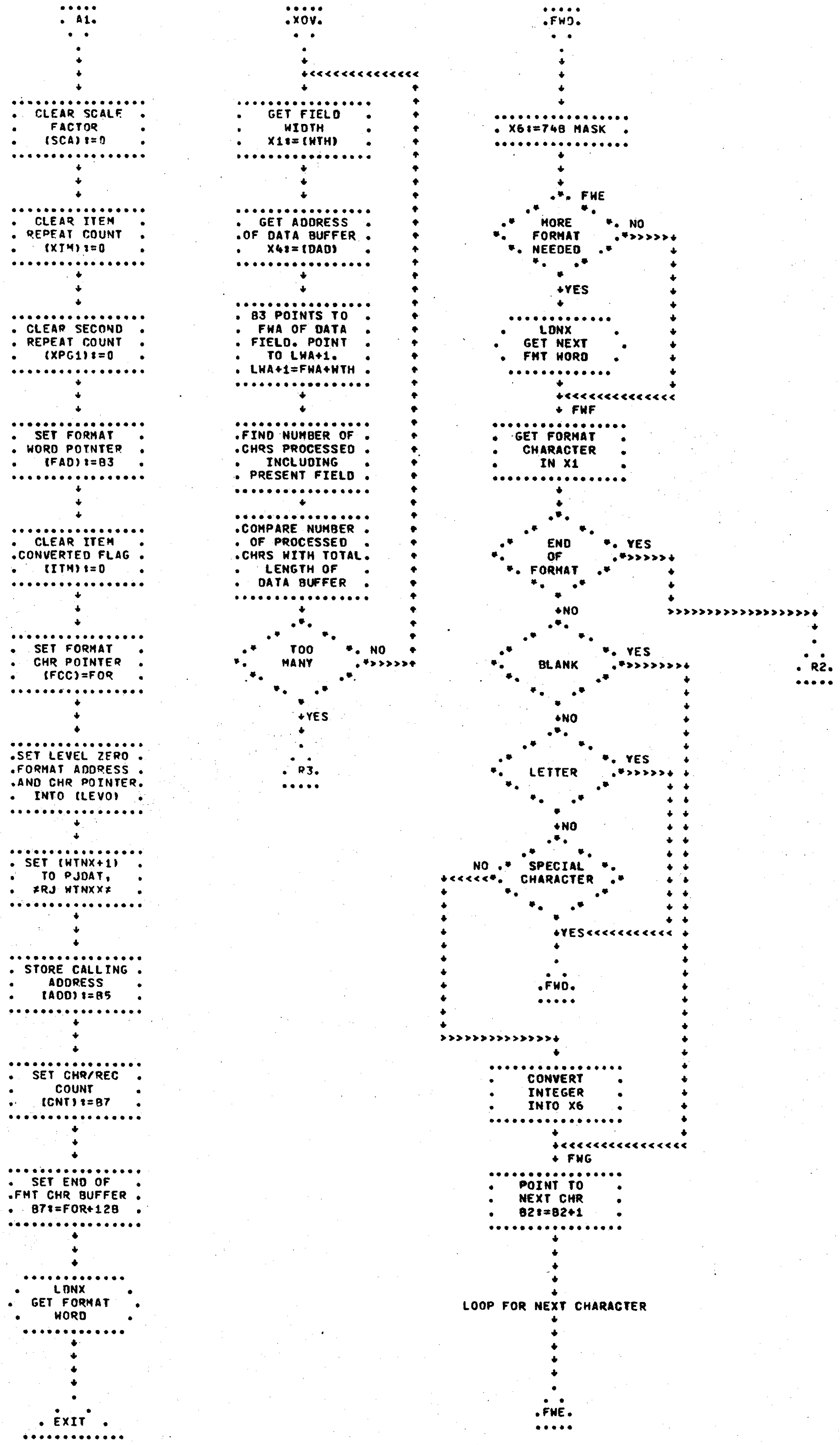
KODER
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL



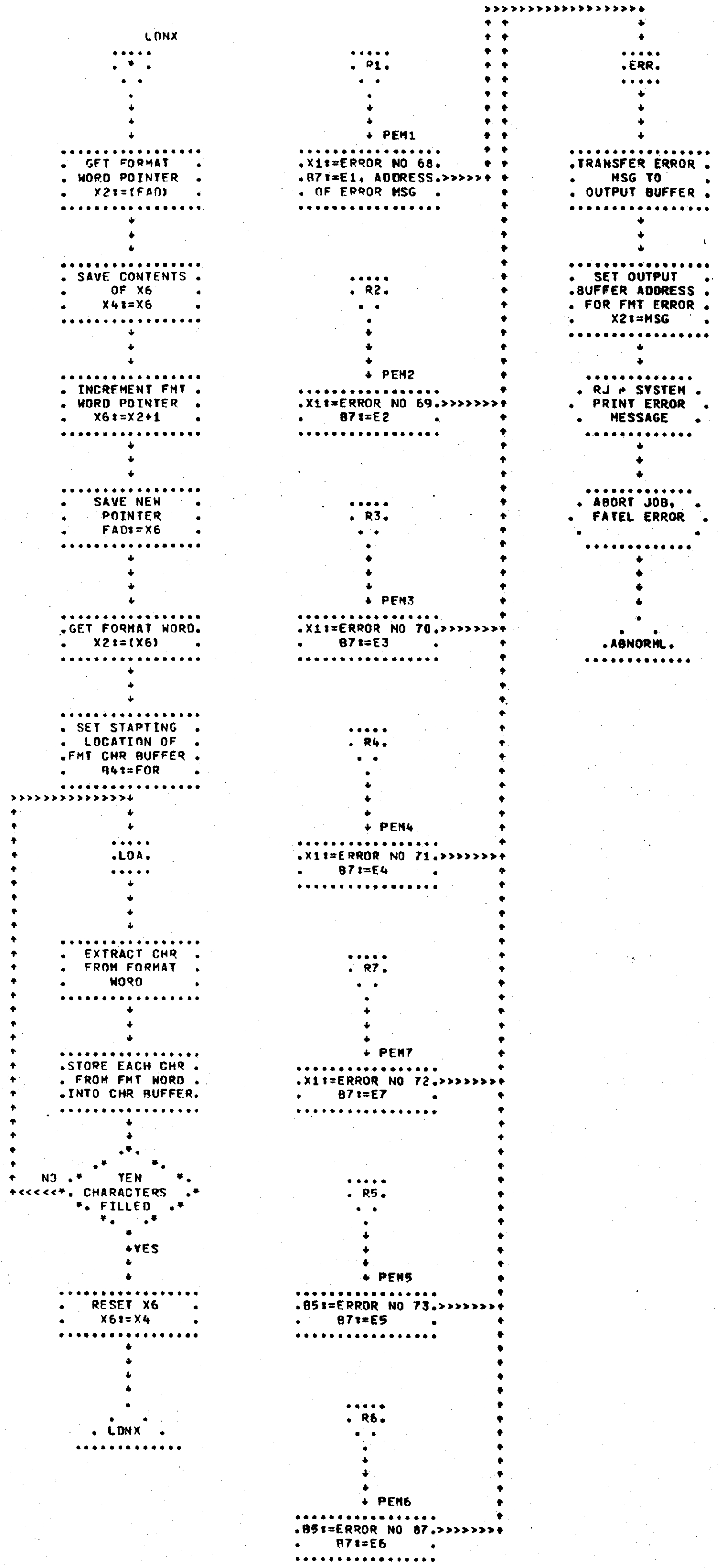
NOTE
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


EXT.
SAVE RE-ENTRY ADDRESS (NXS)=X7
GET ITEM REPEAT COUNT X1=(XIM)
REDUCE ITEM REPEAT COUNT XIM=XIM-1
SAVE FORMAT AND DATA CHR POINTERS (FCC)=B2 (DCC)=B3
SET ITEM CONVERTED FLAG (ITM)=X7
GET LENGTH OF LIST B2=X2=(ARY)
GET CURRENT FORMAT CHR X3=(FUNLT)
LIST LENGTH=0
NO
YES
EXIT
POINT TO NEXT LIST ELEMENT B1=B1+1
REDUCE LIST LENGTH B2=B2-1
DOUBLE PRECISION
NO
YES
REDUCE LIST LENGTH
FRST
A
B3=FMT ADDRESS X1=STATEMENT NUMBER X2=X1
SET RECORD COUNT TO 1 (RAD)=1
X7=54 BIT MASK. B6=5
X1=0
NO
YES
VAR.
EXA
LIST LENGTH=0
NO
YES
EXIT
SINGLE ARRAY (B2>0)
YES
NO
LOOP FOR NEXT ELEMENT
KOA.
R7.
NXT.
EXTRACT UPPER 6 BITS OF X1 INTO X4
IS X4 A SPECIAL CHAR
YES
NO
IS X4 BETWEEN 0-9
YES
NO
ZERO FILL IN X1
YES
NO
NXT1
B6=B6-1
B6=0
NO
YES
ZERO FILL LAST CHR
YES
NO
VAR.
X2=*VARIABLE* VARIABLE FORMAT
NVAR
*
STORE FORMAT NUMBER IN (FMT+1)
SET ENCODE FLAG (ECD)=X3
SET DATA CHR POINTER (DCC)=B2 B6=1
SET PAREN GROUP REPEAT COUNT (XPG)=1
SET ZERO LEVEL PAREN FLAG (LPFLG)=1
SET PAREN LEVEL ZERO (LVLO)=-1
SET DATA BUFFER ADDRESS (DAD)=B2
CLEAR PAREN GRP REPEAT FLAG (XPARN)=0
A1.

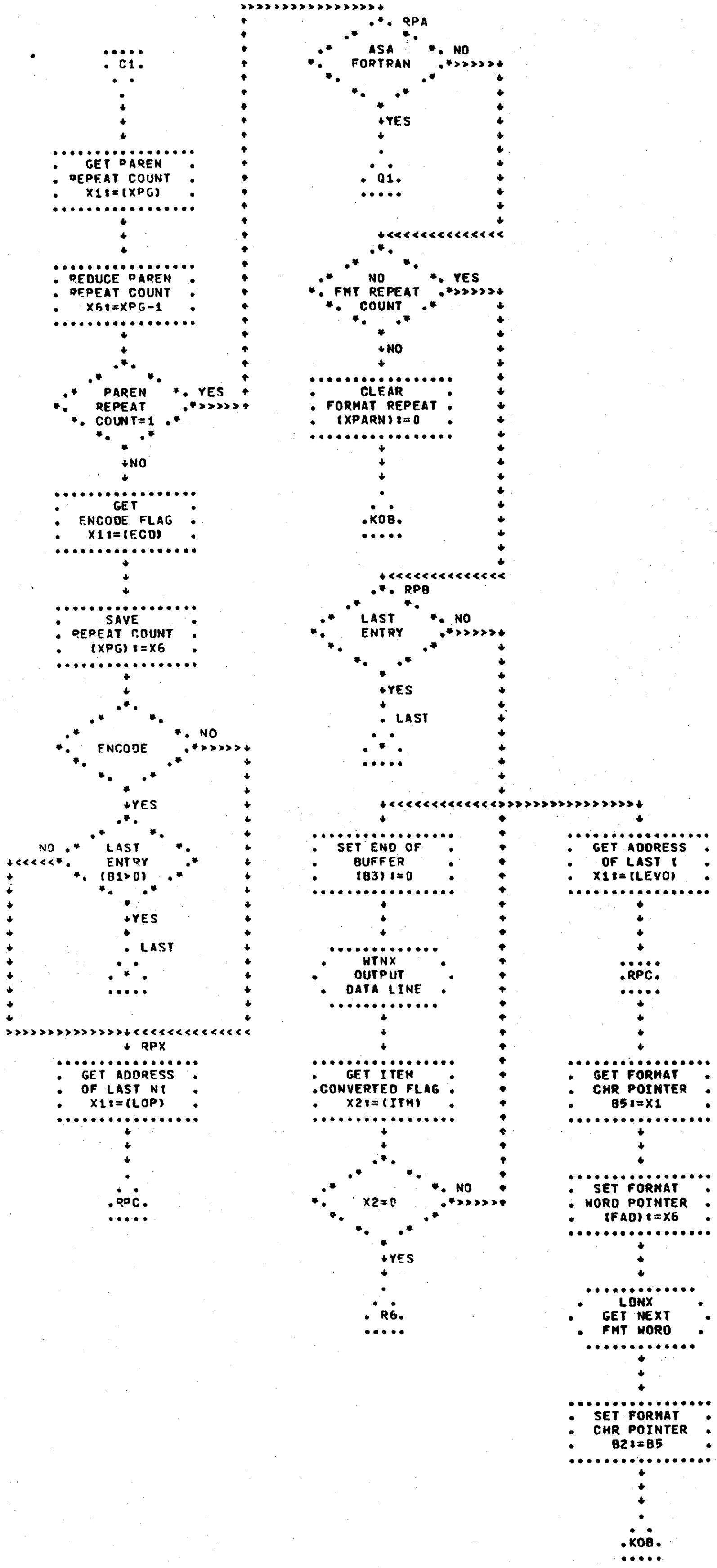
KODER
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


A1.
CLEAR SCALE FACTOR (SCA)↑=0
CLEAR ITEM REPEAT COUNT (XIM)↑=0
CLEAR SECOND REPEAT COUNT (XPG1)↑=0
SET FORMAT WORD POINTER (FAD)↑=B3
CLEAR ITEM CONVERTED FLAG (ITM)↑=0
SET FORMAT CHR POINTER (FCC)=FOR
SET LEVEL ZERO FORMAT ADDRESS AND CHR POINTER. INTO (LEVO)
SET (WTNX+1) TO PJDAT, ≠RJ WTNXX≠
STORE CALLING ADDRESS (ADD)↑=B5
SET CHR/REC COUNT (CNT)↑=B7
SET END OF FMT CHR BUFFER B7↑=FOR+128
LDNX GET FORMAT WORD
EXIT
XOV.
GET FIELD WIDTH X1↑=(WTH)
GET ADDRESS OF DATA BUFFER X4↑=(DAD)
B3 POINTS TO FWA OF DATA FIELD. POINT TO LWA+1. LWA+1=FWA+WTH
FIND NUMBER OF CHRS PROCESSED INCLUDING PRESENT FIELD
COMPARE NUMBER OF PROCESSED CHRS WITH TOTAL. LENGTH OF DATA BUFFER
TOO MANY
NO
YES
R3.
FWD.
X6↑=74B MASK
FWE
MORE FORMAT NEEDED
NO
YES
LDNX GET NEXT FMT WORD
FWF
GET FORMAT CHARACTER IN X1
END OF FORMAT
YES
NO
R2.
BLANK
YES
NO
LETTER
YES
NO
SPECIAL CHARACTER
NO
YES
FWD.
CONVERT INTEGER INTO X6
FWG
POINT TO NEXT CHR B2↑=B2+1
LOOP FOR NEXT CHARACTER
FWE.

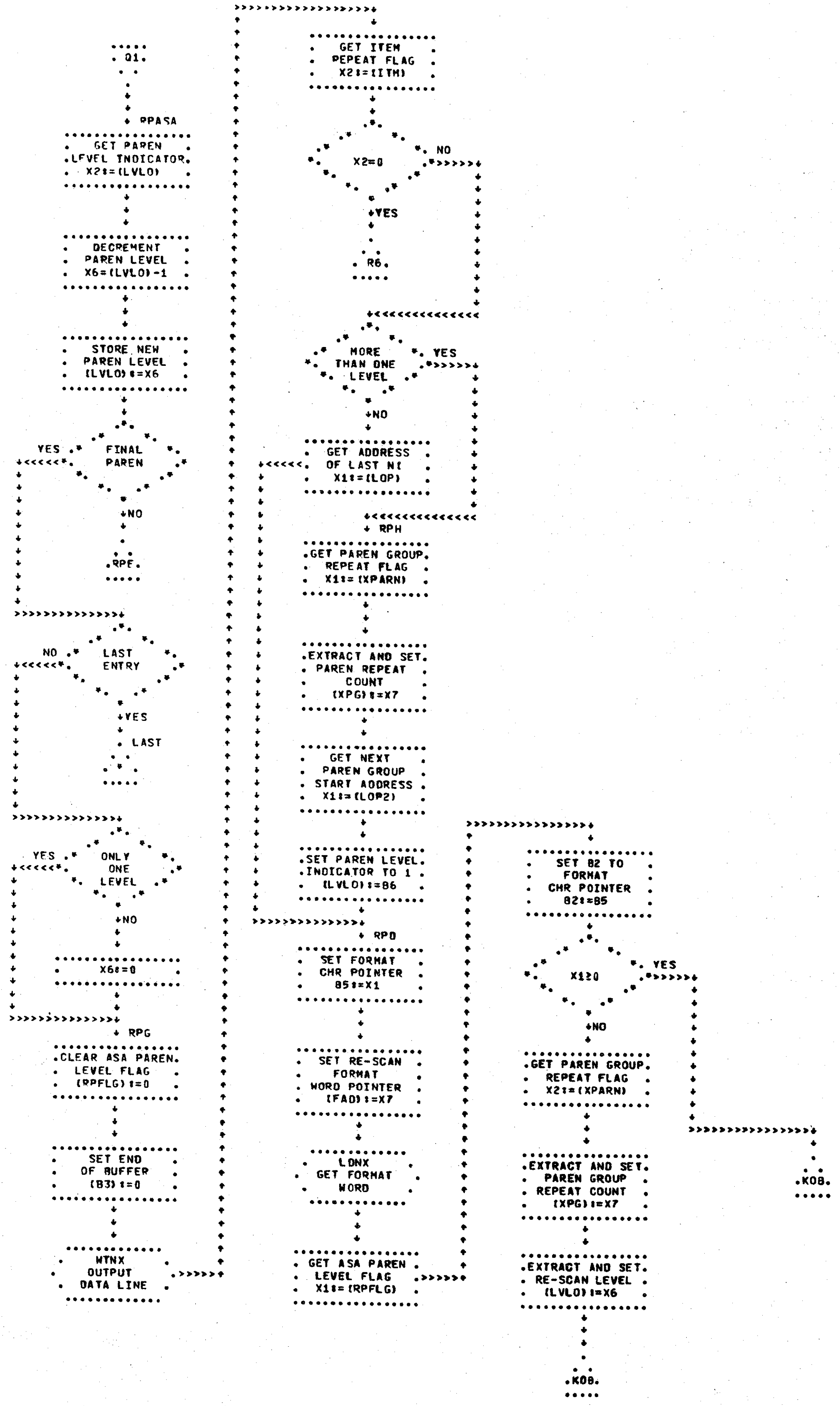
REMARKS
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


LDNX
*
GET FORMAT WORD POINTER X2:=(FAD)
SAVE CONTENTS OF X6 X4:=X6
INCREMENT FMT WORD POINTER X6:=X2+1
SAVE NEW POINTER FAD:=X6
GET FORMAT WORD. X2:=(X6)
SET STARTING LOCATION OF FMT CHR BUFFER B4:=FOR
LDA.
EXTRACT CHR FROM FORMAT WORD
STORE EACH CHR FROM FMT WORD INTO CHR BUFFER.
TEN CHARACTERS FILLED
NO
YES
RESET X6 X6:=X4
LDNX
R1.
PEM1
X1:=ERROR NO 68. B7:=E1. ADDRESS. OF ERROR MSG
R2.
PEM2
X1:=ERROR NO 69. B7:=E2
R3.
PEM3
X1:=ERROR NO 70. B7:=E3
R4.
PEM4
X1:=ERROR NO 71. B7:=E4
R7.
PEM7
X1:=ERROR NO 72. B7:=E7
R5.
PEM5
B5:=ERROR NO 73. B7:=E5
R6.
PEM6
B5:=ERROR NO 87. B7:=E6
ERR.
TRANSFER ERROR MSG TO OUTPUT BUFFER
SET OUTPUT BUFFER ADDRESS FOR FMT ERROR X2:=MSG
RJ - SYSTEM PRINT ERROR MESSAGE
ABORT JOB, FATEL ERROR
ABNORML.

>KOPER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL

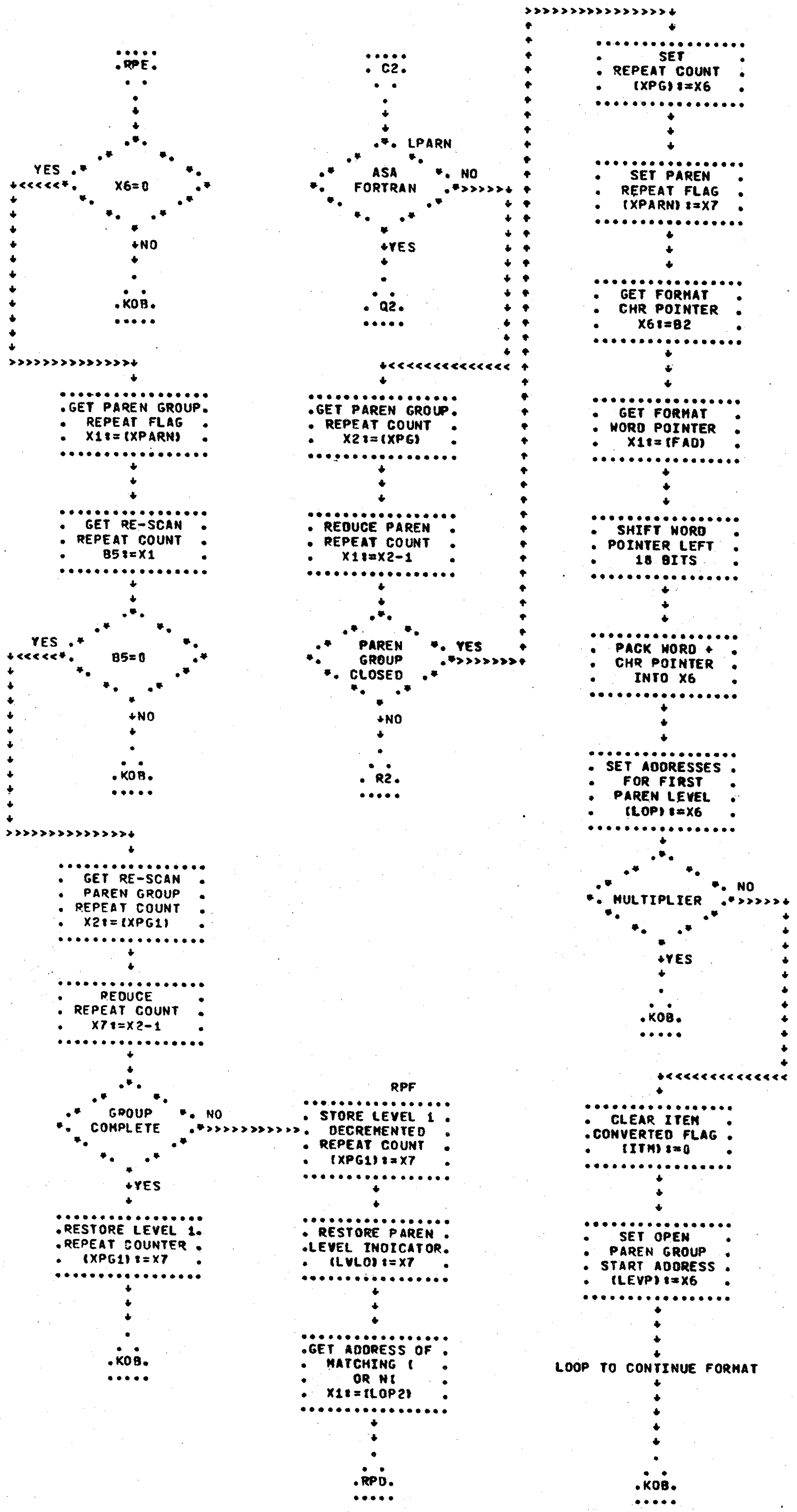

C1.
GET PAREN REPEAT COUNT X1:=(XPG)
REDUCE PAREN REPEAT COUNT X6:=XPG-1
PAREN REPEAT COUNT=1
YES
NO
GET ENCODE FLAG X1:=(ECD)
SAVE REPEAT COUNT (XPG):=X6
ENCODE
NO
YES
LAST ENTRY (B1>0)
NO
YES
LAST
RPX
GET ADDRESS OF LAST N(X1:=(LOP)
RPC.
RPA
ASA FORTRAN
NO
YES
Q1.
NO FMT REPEAT COUNT
YES
NO
CLEAR FORMAT REPEAT (XPARN):=0
KOB.
RPB
LAST ENTRY
NO
YES
LAST
SET END OF BUFFER (B3):=0
WTNX OUTPUT DATA LINE
GET ITEM CONVERTED FLAG X2:=(ITM)
X2=0
NO
YES
R6.
GET ADDRESS OF LAST (X1:=(LEVO)
RPC.
GET FORMAT CHR POINTER B5:=X1
SET FORMAT WORD POINTER (FAD):=X6
LDNX GET NEXT FMT WORD
SET FORMAT CHR POINTER B2:=B5
KOB.

>KODER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL

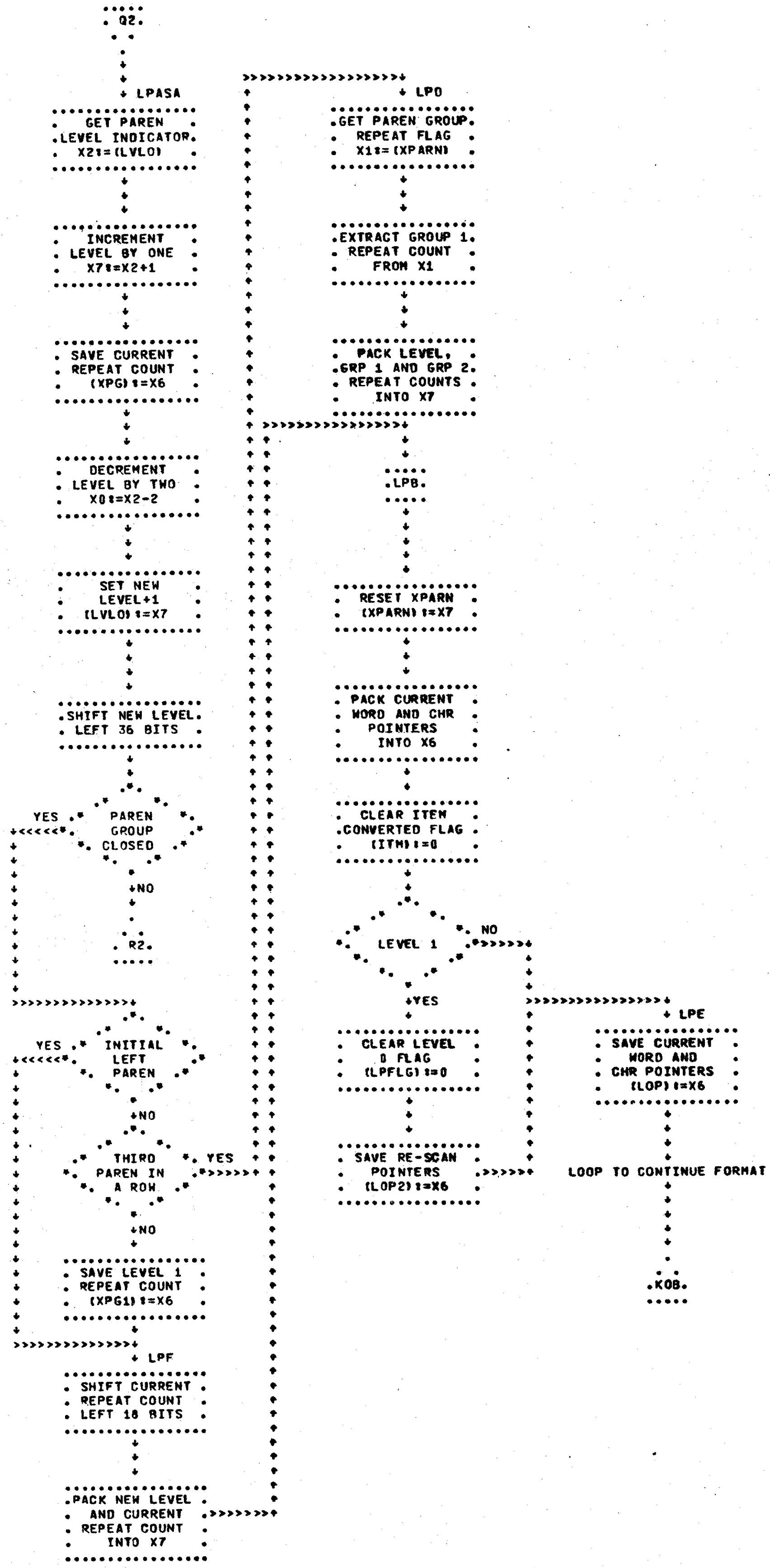

Q1.
RPASA
GET PAREN LEVEL INDICATOR X2:=(LVLO)
DECREMENT PAREN LEVEL X6=(LVLO)-1
STORE NEW PAREN LEVEL (LVLO):=X6
FINAL PAREN
YES
NO
RPE.
LAST ENTRY
NO
YES
LAST
ONLY ONE LEVEL
YES
NO
X6:=0
RPG
CLEAR ASA PAREN LEVEL FLAG (RPFLG):=0
SET END OF BUFFER (B3):=0
WTNX OUTPUT DATA LINE
GET ITEM REPEAT FLAG X2:=(ITM)
X2=0
NO
YES
R6.
MORE THAN ONE LEVEL
YES
NO
GET ADDRESS OF LAST NI X1:=(LOP)
RPH
GET PAREN GROUP REPEAT FLAG X1:=(XPARN)
EXTRACT AND SET PAREN REPEAT COUNT (XPG):=X7
GET NEXT PAREN GROUP START ADDRESS X1:=(LOP2)
SET PAREN LEVEL INDICATOR TO 1 (LVLO):=B6
RPD
SET FORMAT CHR POINTER B5:=X1
SET RE-SCAN FORMAT WORD POINTER (FAD):=X7
LDNX GET FORMAT WORD
GET ASA PAREN LEVEL FLAG X1:=(RPFLG)
SET B2 TO FORMAT CHR POINTER B2:=B5
X1≥0
YES
NO
GET PAREN GROUP REPEAT FLAG X2:=(XPARN)
EXTRACT AND SET PAREN GROUP REPEAT COUNT (XPG):=X7
EXTRACT AND SET RE-SCAN LEVEL (LVLO):=X6
K08.
K08.

>KODER<

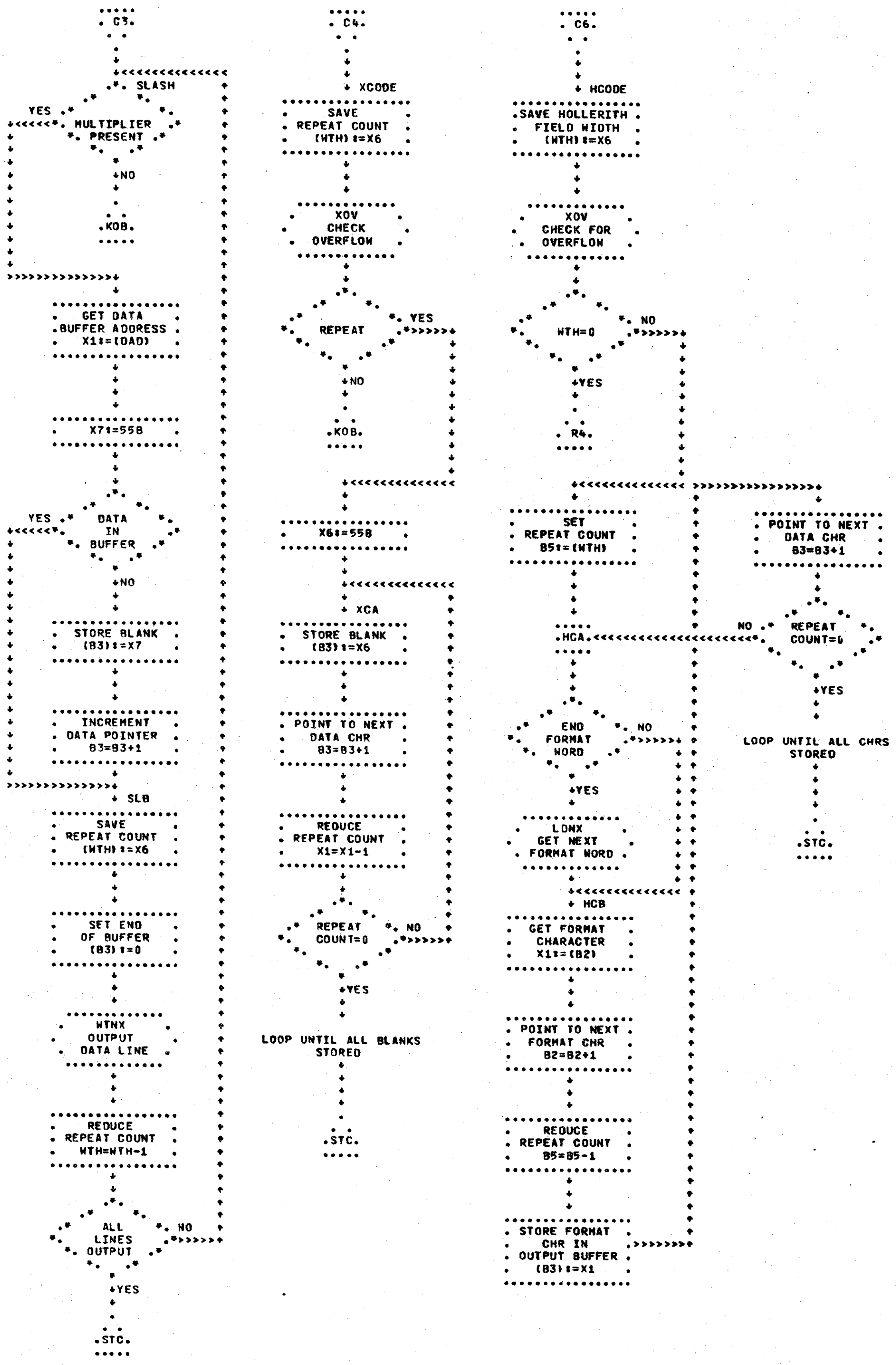
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


RPE.
X6=0
YES
NO
KOB.
GET PAREN GROUP. REPEAT FLAG X1:=(XPARN)
GET RE-SCAN REPEAT COUNT B5:=X1
B5=0
YES
NO
KOB.
GET RE-SCAN PAREN GROUP REPEAT COUNT X2:=(XPG1)
REDUCE REPEAT COUNT X7:=X2-1
GROUP COMPLETE
NO
YES
RESTORE LEVEL 1. REPEAT COUNTER (XPG1):=X7
KOB.
C2.
LPARN
ASA FORTRAN
NO
YES
Q2.
GET PAREN GROUP. REPEAT COUNT X2:=(XPG)
REDUCE PAREN REPEAT COUNT X1:=X2-1
PAREN GROUP CLOSED
YES
NO
R2.
RPF
STORE LEVEL 1 DECREMENTED REPEAT COUNT (XPG1):=X7
RESTORE PAREN LEVEL INDICATOR (LVLO):=X7
GET ADDRESS OF MATCHING (OR NI X1:=(LOP2)
RPD.
SET REPEAT COUNT (XPG):=X6
SET PAREN REPEAT FLAG (XPARN):=X7
GET FORMAT CHR POINTER X6:=B2
GET FORMAT WORD POINTER X1:=(FAD)
SHIFT WORD POINTER LEFT 18 BITS
PACK WORD + CHR POINTER INTO X6
SET ADDRESSES FOR FIRST PAREN LEVEL (LOP):=X6
MULTIPLIER
NO
YES
KOB.
CLEAR ITEM CONVERTED FLAG (ITM):=0
SET OPEN PAREN GROUP START ADDRESS (LEVP):=X6
LOOP TO CONTINUE FORMAT
KOB.

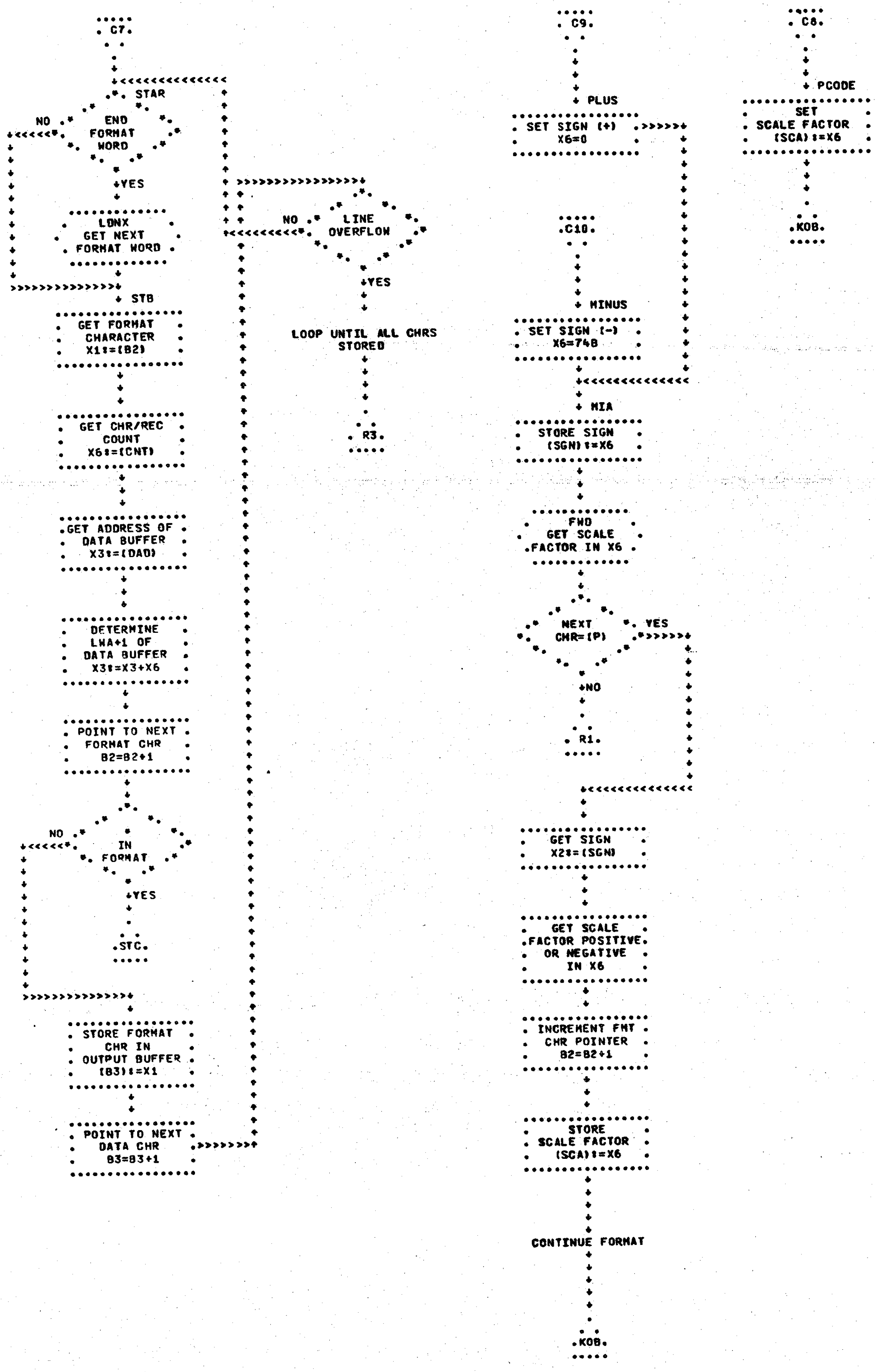
>KODER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


Q2.
LPASA
GET PAREN LEVEL INDICATOR X2:=(LVLO)
INCREMENT LEVEL BY ONE X7:=X2+1
SAVE CURRENT REPEAT COUNT (XPG):=X6
DECREMENT LEVEL BY TWO X0:=X2-2
SET NEW LEVEL+1 (LVLO):=X7
SHIFT NEW LEVEL LEFT 36 BITS
PAREN GROUP CLOSED
YES
NO
R2.
INITIAL LEFT PAREN
YES
NO
THIRD PAREN IN A ROW
YES
NO
SAVE LEVEL 1 REPEAT COUNT (XPG1):=X6
LPF
SHIFT CURRENT REPEAT COUNT LEFT 18 BITS
PACK NEW LEVEL AND CURRENT REPEAT COUNT INTO X7
LPD
GET PAREN GROUP REPEAT FLAG X1:=(XPARN)
EXTRACT GROUP 1 REPEAT COUNT FROM X1
PACK LEVEL, GRP 1 AND GRP 2 REPEAT COUNTS INTO X7
LPB.
RESET XPARN (XPARN):=X7
PACK CURRENT WORD AND CHR POINTERS INTO X6
CLEAR ITEM CONVERTED FLAG (ITM):=0
LEVEL 1
NO
YES
CLEAR LEVEL 0 FLAG (LPFLG):=0
SAVE RE-SCAN POINTERS (LOP2):=X6
LPE
SAVE CURRENT WORD AND CHR POINTERS (LOP):=X6
LOOP TO CONTINUE FORMAT
KOB.

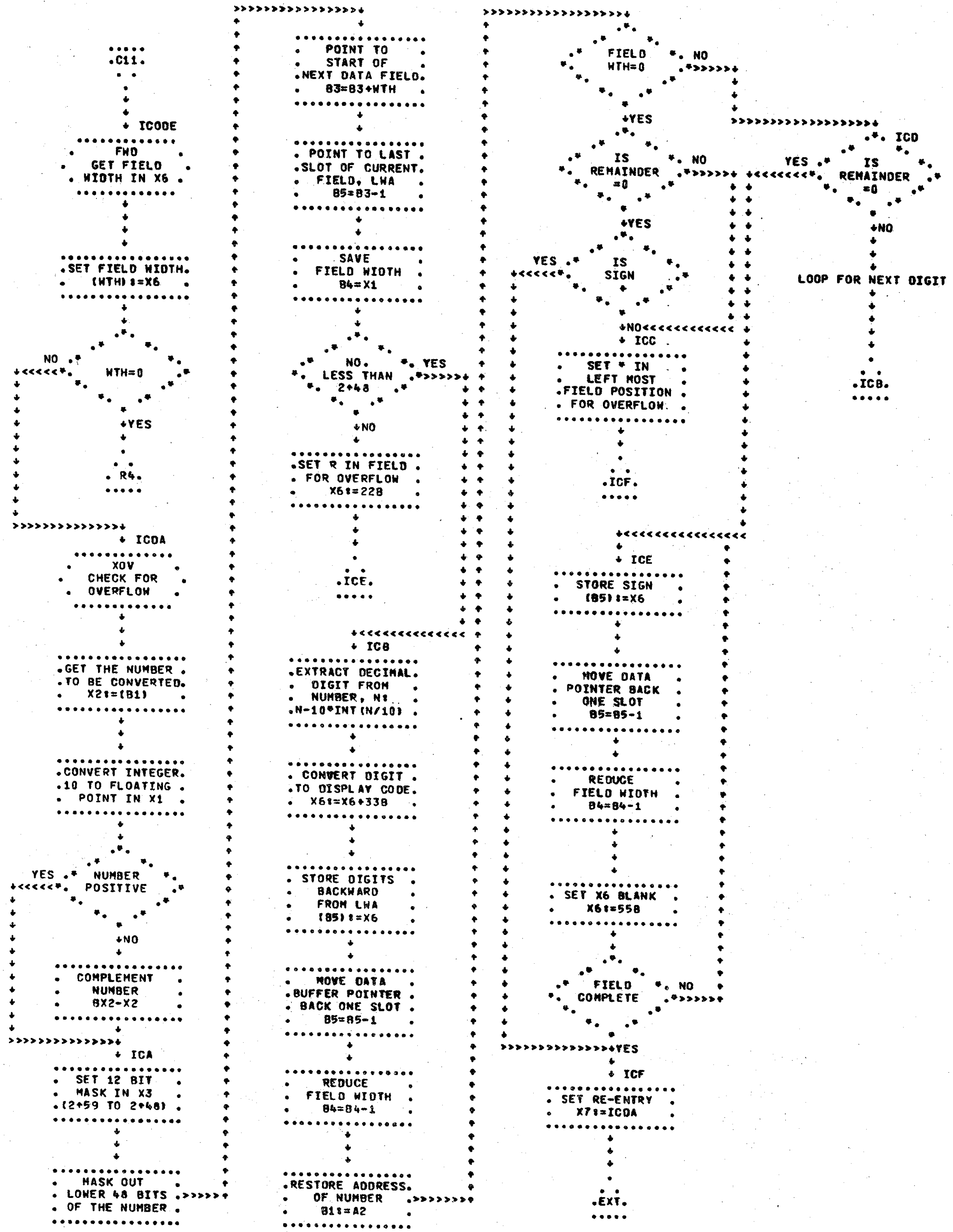
>KODEP<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


C3.
SLASH
YES
MULTIPLIER PRESENT
NO
KOB.
GET DATA BUFFER ADDRESS X1:=(DAD)
X7:=55B
YES
DATA IN BUFFER
NO
STORE BLANK (B3):=X7
INCREMENT DATA POINTER B3=B3+1
SLB
SAVE REPEAT COUNT (WTH):=X6
SET END OF BUFFER (B3):=0
WTNX OUTPUT DATA LINE
REDUCE REPEAT COUNT WTH=WTH-1
ALL LINES OUTPUT
NO
YES
STC.
C4.
XCODE
SAVE REPEAT COUNT (WTH):=X6
XOV CHECK OVERFLOW
REPEAT
YES
NO
KOB.
X6:=55B
XCA
STORE BLANK (B3):=X6
POINT TO NEXT DATA CHR B3=B3+1
REDUCE REPEAT COUNT X1=X1-1
REPEAT COUNT=0
NO
YES
LOOP UNTIL ALL BLANKS STORED
STC.
C6.
HCODE
SAVE HOLLERITH FIELD WIDTH (WTH):=X6
XOV CHECK FOR OVERFLOW
WTH=0
NO
YES
R4.
SET REPEAT COUNT B5:=(WTH)
POINT TO NEXT DATA CHR B3=B3+1
HCA.
NO
REPEAT COUNT=0
YES
LOOP UNTIL ALL CHRS STORED
STC.
END FORMAT WORD
NO
YES
LONX GET NEXT FORMAT WORD
HCB
GET FORMAT CHARACTER X1:=(B2)
POINT TO NEXT FORMAT CHR B2=B2+1
REDUCE REPEAT COUNT B5=B5-1
STORE FORMAT CHR IN OUTPUT BUFFER (B3):=X1

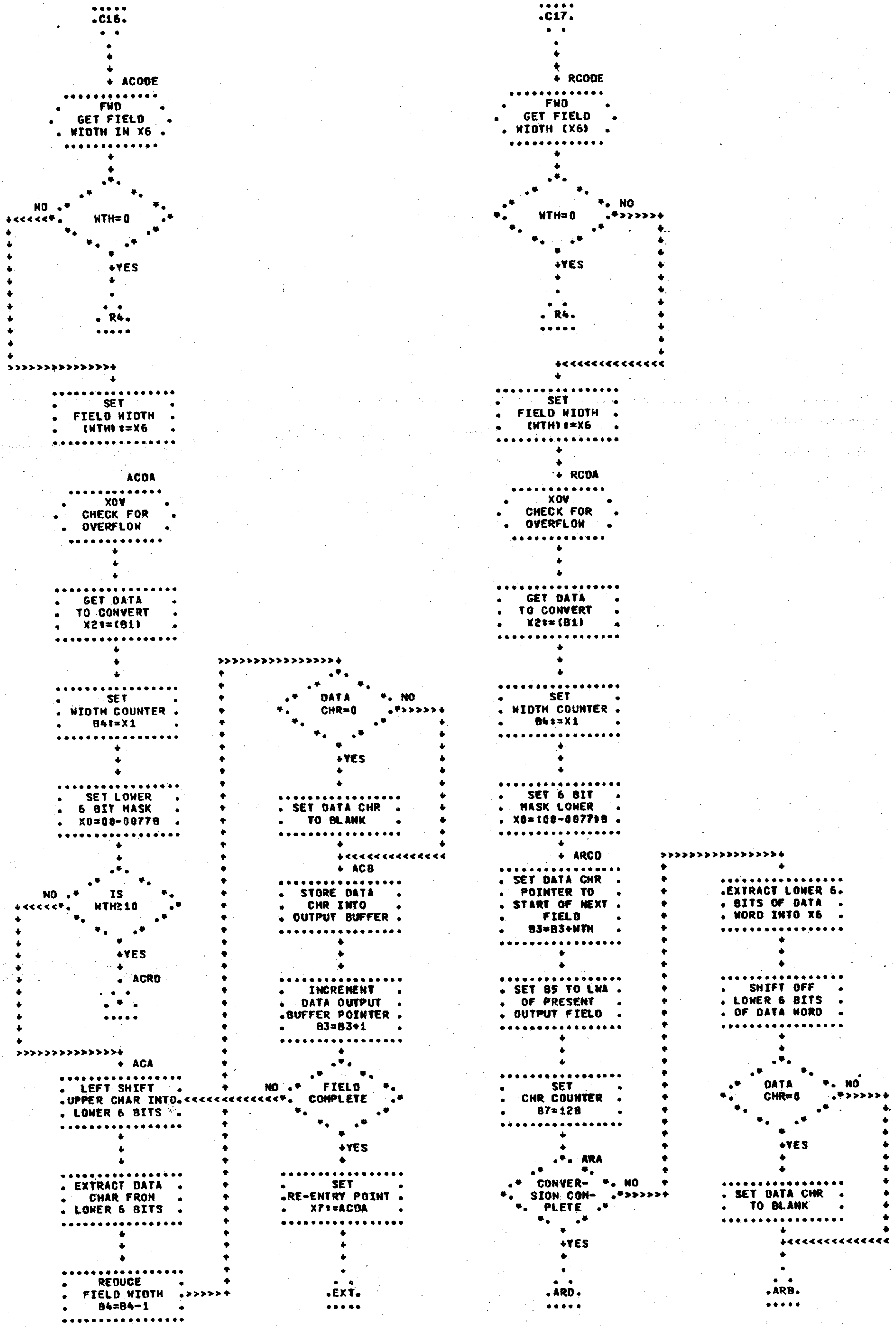
>KODER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


C7.
STAR
END FORMAT WORD
NO
YES
LDNX GET NEXT FORMAT WORD
STB
GET FORMAT CHARACTER X1:=(B2)
GET CHR/REC COUNT X6:=(CNT)
GET ADDRESS OF DATA BUFFER X3:=(DAD)
DETERMINE LWA+1 OF DATA BUFFER X3:=X3+X6
POINT TO NEXT FORMAT CHR B2=B2+1
IN FORMAT
NO
YES
STC.
STORE FORMAT CHR IN OUTPUT BUFFER (B3):=X1
POINT TO NEXT DATA CHR B3=B3+1
LINE OVERFLOW
NO
YES
LOOP UNTIL ALL CHRS STORED
R3.
C9.
PLUS
SET SIGN (+) X6=0
C10.
MINUS
SET SIGN (-) X6=74B
MIA
STORE SIGN (SGN):=X6
FWD GET SCALE FACTOR IN X6
NEXT CHR=(P)
YES
NO
R1.
GET SIGN X2:=(SGN)
GET SCALE FACTOR POSITIVE OR NEGATIVE IN X6
INCREMENT FMT CHR POINTER B2=B2+1
STORE SCALE FACTOR (SCA):=X6
CONTINUE FORMAT
KOB.
C8.
PCODE
SET SCALE FACTOR (SCA):=X6
KOB.

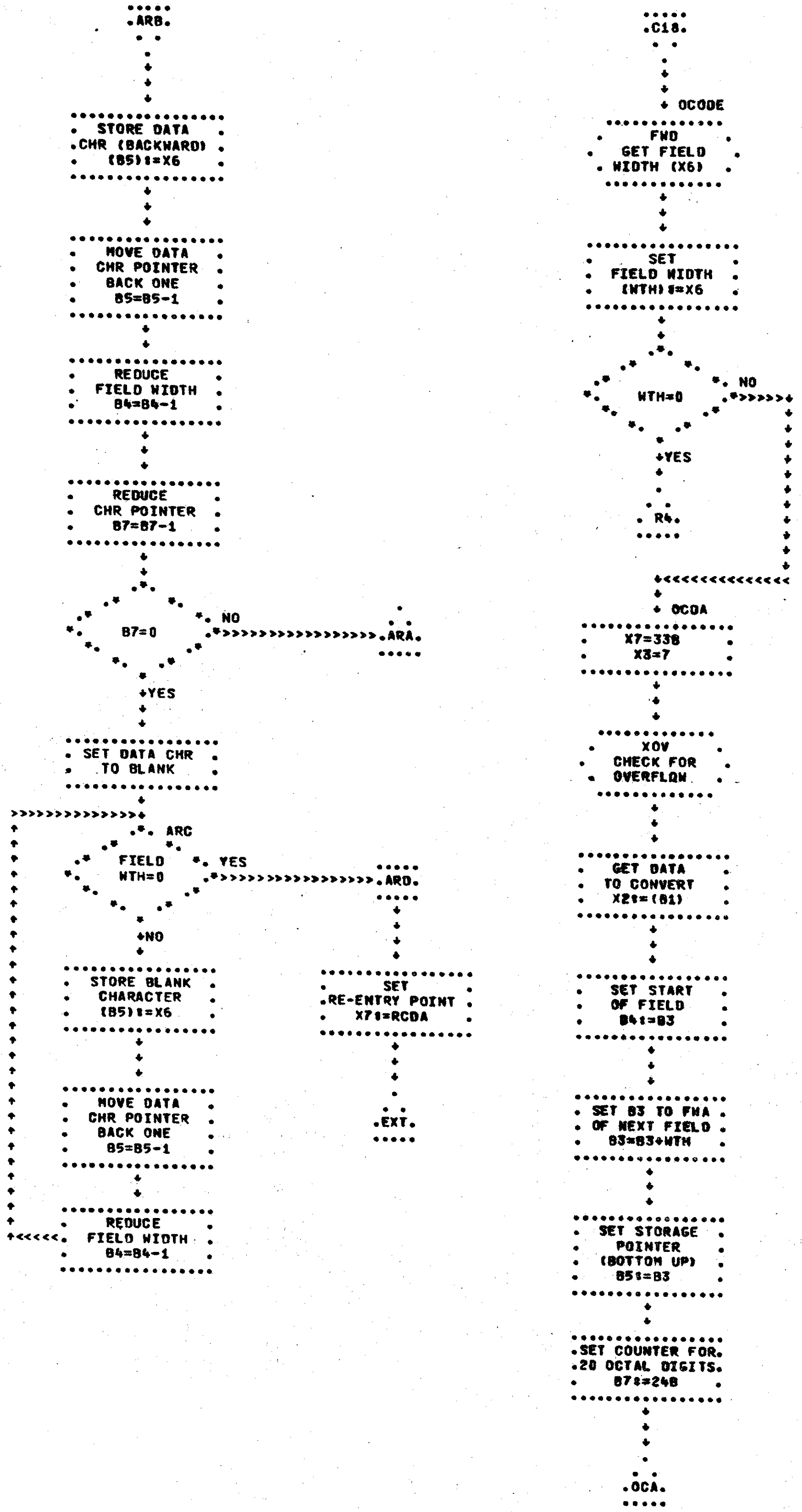
>KODE9<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL



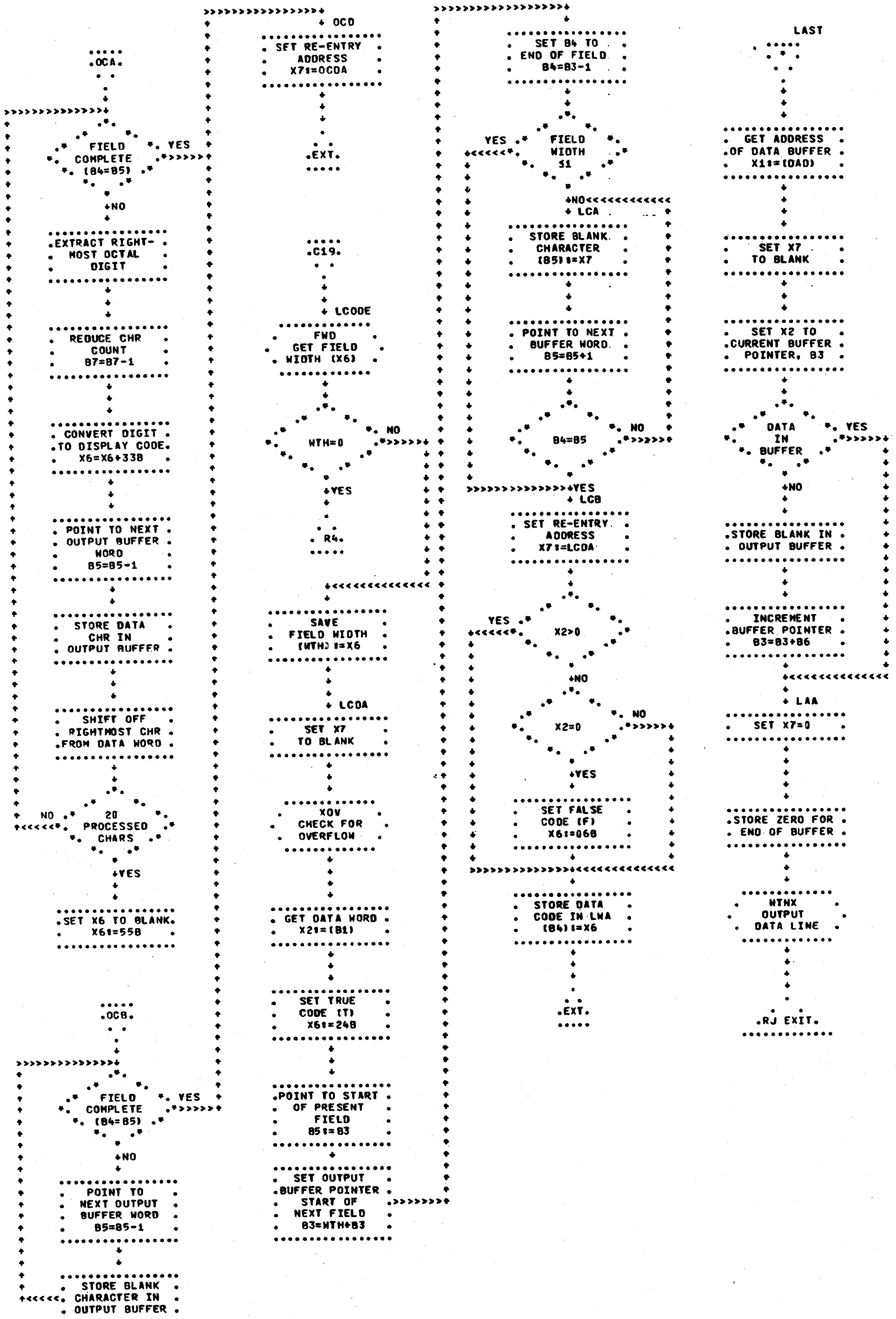
>KODER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


C16
ACODE
FWD GET FIELD WIDTH IN X6
WTH=0
NO
YES
R4
SET FIELD WIDTH (WTH)↑=X6
ACDA
XOV CHECK FOR OVERFLOW
GET DATA TO CONVERT X2↑=(B1)
SET WIDTH COUNTER B4↑=X1
SET LOWER 6 BIT MASK X0=00-00778
IS WTH≥10
NO
YES
ACRD
ACA
LEFT SHIFT UPPER CHAR INTO LOWER 6 BITS
EXTRACT DATA CHAR FROM LOWER 6 BITS
REDUCE FIELD WIDTH B4=B4-1
DATA CHR=0
NO
YES
SET DATA CHR TO BLANK
ACB
STORE DATA CHR INTO OUTPUT BUFFER
INCREMENT DATA OUTPUT BUFFER POINTER B3=B3+1
FIELD COMPLETE
NO
YES
SET RE-ENTRY POINT X7↑=ACDA
EXT
C17
RCODE
FWD GET FIELD WIDTH (X6)
WTH=0
NO
YES
R4
SET FIELD WIDTH (WTH)↑=X6
RCDA
XOV CHECK FOR OVERFLOW
GET DATA TO CONVERT X2↑=(B1)
SET WIDTH COUNTER B4↑=X1
SET 6 BIT MASK LOWER X0=(00-0077)8
ARCD
SET DATA CHR POINTER TO START OF NEXT FIELD B3=B3+WTH
SET B5 TO LWA OF PRESENT OUTPUT FIELD
SET CHR COUNTER B7=128
ARA
CONVER-SION COM-PLETE
NO
YES
ARD
EXTRACT LOWER 6 BITS OF DATA WORD INTO X6
SHIFT OFF LOWER 6 BITS OF DATA WORD
DATA CHR=0
NO
YES
SET DATA CHR TO BLANK
ARB

>KODER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


ARB
STORE DATA CHR (BACKWARD) (B5)$=X6
MOVE DATA CHR POINTER BACK ONE B5=B5-1
REDUCE FIELD WIDTH B4=B4-1
REDUCE CHR POINTER B7=B7-1
B7=0
NO
ARA
YES
SET DATA CHR TO BLANK
ARC
FIELD WTH=0
YES
ARD
NO
STORE BLANK CHARACTER (B5)$=X6
SET RE-ENTRY POINT X7$=RCDA
EXT
MOVE DATA CHR POINTER BACK ONE B5=B5-1
REDUCE FIELD WIDTH B4=B4-1
C18
OCODE
FWD GET FIELD WIDTH (X6)
SET FIELD WIDTH (WTH)$=X6
WTH=0
NO
YES
R4
OCOA
X7=33B X3=7
XOV CHECK FOR OVERFLOW
GET DATA TO CONVERT X2$=(B1)
SET START OF FIELD B4$=B3
SET B3 TO FWA OF NEXT FIELD B3=B3+WTH
SET STORAGE POINTER (BOTTOM UP) B5$=B3
SET COUNTER FOR 20 OCTAL DIGITS. B7$=24B
OCA

>KODER<
THE * INSIDE CONNECTORS STANDS FOR THE ASSOCIATED LABEL


OCA.
FIELD COMPLETE (B4=B5)
YES
NO
EXTRACT RIGHT-MOST OCTAL DIGIT
REDUCE CHR COUNT B7=B7-1
CONVERT DIGIT TO DISPLAY CODE. X6=X6+33B
POINT TO NEXT OUTPUT BUFFER WORD B5=B5-1
STORE DATA CHR IN OUTPUT BUFFER
SHIFT OFF RIGHTMOST CHR FROM DATA WORD
NO
20 PROCESSED CHARS
YES
SET X6 TO BLANK. X6:=55B
OCB.
FIELD COMPLETE (B4=B5)
YES
NO
POINT TO NEXT OUTPUT BUFFER WORD B5=B5-1
STORE BLANK CHARACTER IN OUTPUT BUFFER
OCD
SET RE-ENTRY ADDRESS X7:=OCDA
EXT.
C19.
LCODE
FWD GET FIELD WIDTH (X6)
WTH=0
NO
YES
R4.
SAVE FIELD WIDTH (WTH):=X6
LCDA
SET X7 TO BLANK
XOV CHECK FOR OVERFLOW
GET DATA WORD X2:=(B1)
SET TRUE CODE (T) X6:=24B
POINT TO START OF PRESENT FIELD B5:=B3
SET OUTPUT BUFFER POINTER START OF NEXT FIELD B3=WTH+B3
SET B4 TO END OF FIELD B4=B3-1
YES
FIELD WIDTH ≤1
NO
LCA
STORE BLANK CHARACTER (B5):=X7
POINT TO NEXT BUFFER WORD B5=B5+1
B4=B5
NO
YES
LCB
SET RE-ENTRY ADDRESS X7:=LCDA
YES
X2>0
NO
X2=0
NO
YES
SET FALSE CODE (F) X6:=06B
STORE DATA CODE IN LWA (B4):=X6
EXT.
LAST
GET ADDRESS OF DATA BUFFER X1:=(DAD)
SET X7 TO BLANK
SET X2 TO CURRENT BUFFER POINTER, B3
DATA IN BUFFER
YES
NO
STORE BLANK IN OUTPUT BUFFER
INCREMENT BUFFER POINTER B3=B3+B6
LAA
SET X7=0
STORE ZERO FOR END OF BUFFER
WTNX OUTPUT DATA LINE
RJ EXIT.

102684224 , OF 2