JAN 2 9 1960

SDC FIELD NOTE A WORKING PAPER | FN-LO-201

M Blauer.

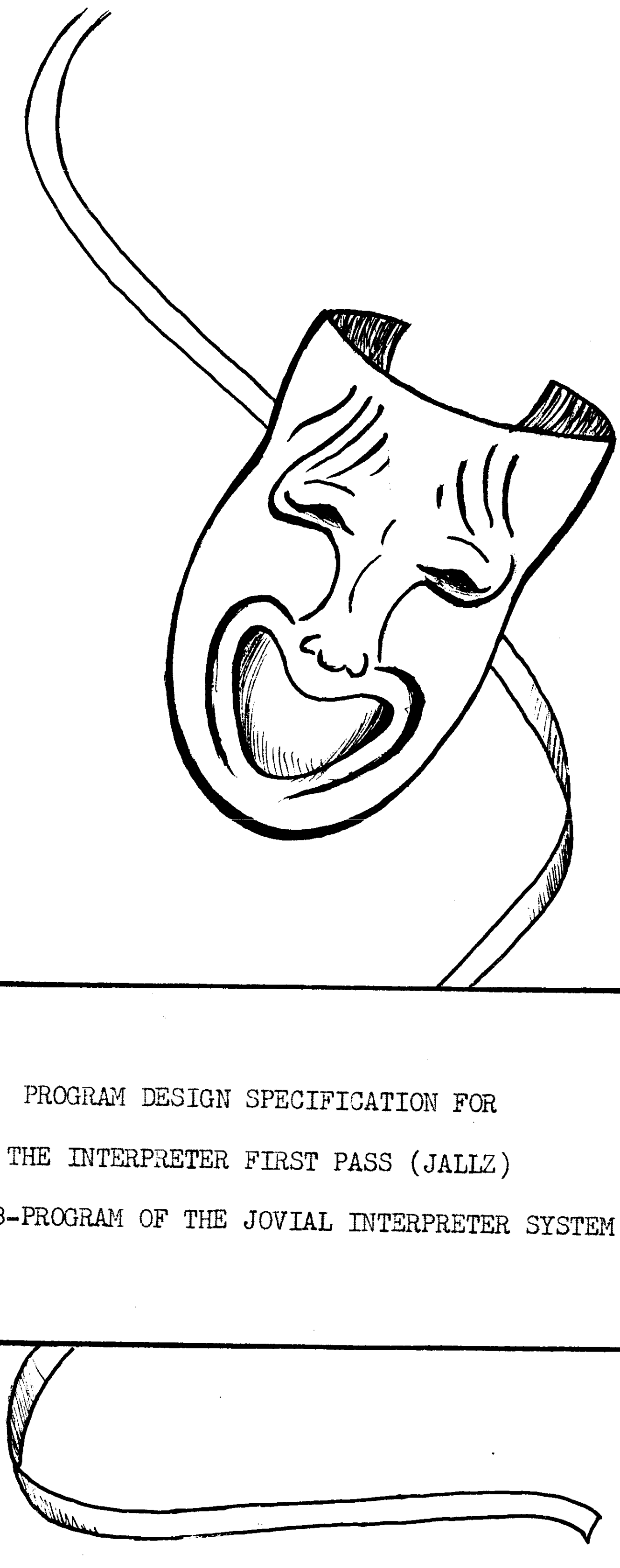
AUTHOR [illegible]
CUSS Project

APPROVED [illegible]
M. Blauer

(Produced under System Development Corporation subcontract No 202 issued by International Electric Corporation in performance of contract AF-30(635)-11583)

PAGE 1 OF 69 PAGES

SYSTEM DEVELOPMENT CORPORATION • 2500 COLORADO AVENUE • SANTA MONICA, CALIFORNIA



PROGRAM DESIGN SPECIFICATION FOR THE INTERPRETER FIRST PASS (JALLZ) A SUB-PROGRAM OF THE JOVIAL INTERPRETER SYSTEM

TABLE OF CONTENTS

PROGRAM DESIGN SPECIFICATION FOR THE INTERPRETER FIRST PASS (JALLZ) A SUB PROGRAM OF THE JOVIAL INTERPRETER SYSTEM

INTRODUCTION

This document defines the workings of a rather large and complex program which represents the efforts of several programmers*. Anyone wishing to get a general knowledge of the internal functioning of such a program as the Interpreter would do well to study the description contained. A knowledge of the JOVIAL language is essential to a critical understanding of this document. Such knowledge may be obtained by reading FN-LO-34-2. Of particular importance to the user of the Interpreter is the descriptive analysis of error message printouts contained herein.

The First Pass of the Interpreter (hereinafter referred to as JALLZ) has as its main purpose the preparation of the internal tables and their core allocation for future use by other system sub-programs. This is accomplished by several major program functions. The first involves reading in the JOVIAL-coded Object Program (including system data characteristic description cards) and breaking each JOVIAL statement down into a form which lends itself to further analysis, this results in the formation of the necessary internal tables such as Intermediate Language table, Statement Label table, Variable table, Subscript table, Constant table, Status table, and Status-Switch table. At this time all cross references from table to table are relative to the starting location of the respective table. Once all statements have been analyzed, the core allocation and address assignment begins. This Pass computes the amount of storage required for all variables and allocates a specific place in core memory for each. It also allocates space and addresses for all internal tables to be used by JOLLZ, the Second Pass of the Interpreter. Finally, it makes all table cross references absolute and writes the internal tables onto tape prior to returning to the Test Control Program (JTCPZ). Throughout JALLZ any program errors detected are logged, and in most cases, the occurrence of an error will cause an indicator (Sense Bit 12) to be set, meaning discontinuance of further operation after return to the Test Control Program.

* The people who contributed to the programming of JALLZ and the writing of this document are Paul McIsaac, H. L. Howell, John Rafferty and Patricia Weaver.

ENVIRONMENT

Sense Switch

Sense switch 1 must be set prior to operating JALIZ. When it is off, the program will assume tape unit A2 contains the program deck input. When it is on, card reader input is assumed.

Master Compool

The Interpreter First Pass (JALIZ) utilizes data definitions included in the Master Compool, specifies by the test designer, through the Test Control card. This compool is read in by the Test Control program prior to the operation of JALIZ.

INPUT

The JOVIAL program deck must be headed with a START card. The format of the card may vary, provided that the first word of the card reads START. The remainder of the card may contain anything the programmer wishes (e.g., name, date, etc.). It is essential that no other JOVIAL statement be on the START card (i.e., the START card is a separate card in itself). If JALIZ fails to find a START card, it will continue searching for one, skipping all other cards until it has found the START card.

The final statement of the Input deck must be a TERM card. This is a statement headed by the word TERM. It may or may not be followed by a statement label. If it is followed by a label, the label indicates the first JOVIAL statement to be operated. If no label follows, the first statement of the program will be initially operated. A dollar sign ($) must follow the TERM statement.
Note: A more complete description of card and deck format will be found in FN-LO-34-2.

FUNCTIONS

The main purpose of JALIZ is the conversion of the JOVIAL object program to a form in which it can be interpreted by JOLIZ. JALIZ is composed of three major sections and several subsections.

The sections are:

A) Read and translate cards. Uses subsections A1 and A2.

B) Analyze statements and produce first form of intermediate language for interpretation. Uses subsections B1 and B2.

FUNCTIONS cont.

C) Assign absolute addresses for all references to variables, constants, and other statements in all tables used by JOLIZ.

Each of these individual sections has certain functions to perform which contribute to the overall functions of JALIZ. These will be described below.

Functions of Section A

1. Searches for START card, skipping all other tape records or cards until START card is found.

2. Upon finding START card, transfers to section (Section A1) which reads in and converts Item table declaration cards.

3. After processing item table definition cards, determines type of statement on following cards by examing first significant word.

4. Processes each statement according to type of statement. Places class and form of each word found into Card Analysis Table (CAT).

   A) Converts all constants to binary, and stores into Constant Table (CON) with necessary information to interpret constant. Places relative position of this constant into CAT Table.

   B) Converts all subscripts to relative position in Subscript Table (SUB). Places relative position of this constant into CAT Table.

   C) Stores all variables into Variable Table (VAT) (Refer to Section A2 Functions.) Place relative position of this variable into CAT Table.

5. Upon finding each word in a statement and determining exactly what it is, checks the legality of this word as to its position in regard to preceding and following words. Also checks the legality in relation to this type of statement. Upon finding an error, logs out error message, location of error (statement + increment) and reference number to error causes. Sets flag indicating compiling "STOP". Continue input process to spot any possible additional format errors.

6. Upon processing each statement transfer control to location JOHN (section B) to convert CAT Table to Intermediate Language Table (ILT). Control returned from JOHN.

7. Upon finding TERM card, sets flag to indicate program end upon return from JOHN.

Functions of Section A1

This section is operated when Section A recognizes a START card or a Procedure Declaration.

1. Reads and interprets ITEM and TABLE declarations, lists of input and output parameters which appear in Procedure Declaration headings, and lists of values defined for tables of constants.

2. Checks the legality (both of format and constant) of all statements processed, and logs error messages for each illegal condition found.

3. Makes an entry for each table and item in VAT, enters lists of values for status items in the Status Table (STAT), and write lists of constant values for items within tables of constants on Tape D1.

4. For procedures, if a single item having the same name as the procedure is declared, and no output parameters were listed in the procedure heading, change the form and class of the CAT entry for the procedure declaration to one-output procedure.

5. Where a table is declared to be identical to a compool defined table, enter the latter in VAT also.

6. This section returns to Section A when it encounters a statement which is not an item or table declaration.

Functions of Section A2

This section is a closed subroutine. It is entered from Section A each time reference to a variable name is encountered in a program statement (other than item or table declarations). The name of the variable is the single entrance parameter.

1. Checks to see that the variable has been defined for this section of the object program. Accepts the first definition found in one of the following categories, checking the categories in the listed order:

    1. Definition by item or table declaration within the procedure (if the program statement is within a procedure).

    2. Definition by item or table declaration in the main program.

    3. Definition by the compool.

2. If no definition is found, logs an error message and exit with an error indication.

3. If a definition is found, exits with the relative position of the VAT entry which contains the definition.

Functions of Section A2 Cont.

4. Before exiting, adds new entries to VAT under the following circumstances:

   A) If the variable is an item within an identical table, and it is not yet in VAT, makes a VAT entry for it. If the item which is imaged is compool defined and not yet in VAT, enters it also.

   B) If this variable is compool defined and not yet in VAT, enters it. If it is an item and its table is not yet in VAT, enters it from the compool also.

Where the reference is encountered in a statement within a procedure the order of search is:

1. Variable defined for this procedure.

2. Variable defined by the main program.

3. Compool-defined variables.

Finding a variable in 1 or 2 above, may involve adding new entries to the variable table.

Functions of Section B

1. The main function of this section is to make all entries into the Intermediate Language Table. The section does this by processing one statement at a time after the statement has been placed by Section A, in the Card Analysis Table.

2. The section decides how to select the correct sequence of entries to be placed into the Intermediate Language Table to make the statement performed in a logically and arithmetically correct way. This is done with the aid of Sections B1 and B2.

3. It makes decisions as in a complete FOR statement as to what part of a statement may be entered into the Intermediate Language Table at this time and what part must be held aside until a later time when it will be placed in the ILT.

4. It decides which statement must be labeled if the object program has not already been given a statement label. In these situations it will generate a unique label for the object program. Unique labels always begin with numbers. It generates labels to find the correct branch points of an IF Statement. It will generate a label for the stepping and testing of any subscript specified by a complete FOR statement.

Functions of Section B Cont.

5. It checks every subscript used to insure that it has been preset by a FOR statement which is still active according to the object programmer's arrangements of BEGINs and ENDs

6. It keeps a check on all Procedure and Close Declarations and where they end. It gives each procedure a unique number to insure that labels within the procedure that match labels outside the procedure or within another procedure are not considerd duplicates. It decides which procedure calls are for "multiple output procedures" and which are "one output procedures," in order to insert an entry in ILT referring to the accumulator.

Functions of Section B1

This section assigns levels to the operators of an expression, so that Section B may determine the correct order of expansion. This section operates whenever a statement which may contain an arithmetic or logical expression is encountered in Section B.

1. When control is transferred, an entrance parameter is given as to where to begin in the Card Analysis Table (CAT).

2. An entry is made in the Level Analysis Table (LAT1) for all terms (identified by a class greater than 6). All entries except the initial entry have an operator and operator level (some separators are considered as operators at this stage). Each operator and some separators have values to indicate priority of operation in laying out ILT:

Operator or Separator	Value
OF	0
↑	1
* /	2
- +	3
= (For output parameters of procedures)	4
,	5
relation operator	6
AND	7
OR	8
= (Assignment)	9
( )↓	10

Functions of Section B1 Cont.

The level is computed in the following manner:

(a) A counter is initiated at 10.

(b) Left parenthesis causes 10 to be added to the counter.

(c) The operator level is computed by subtracting the level value (above from the counter. However, the counter is not changed.

(d) The right parenthesis causes 10 to be subtracted from the counter.

(e) For the up exponent (↑), after the operator level is computed, the counter is advanced by 10.

(f) At the end of the statement, the counter is checked for 10 to see if the statement is correct.

3. If an AND or OR operator occurs, entries are made in a secondary Level Analysis Table (LAT2) in addition to LAT1. The same level as given in LAT1 is used.

4. If a negation symbol (NOT) occurs, no information is given in LAT1. An entry is made in a special table (NOT) which gives the current value of the counter as the level and indicates the relative in LAT2 for the associated condition. If the negation symbol NOT occurs prior to any AND or OR, a special indication is made in the NOT Table.

Functions of Section B2

Control is transferred by Section B to this section for IF statements in order to compute the correct branch points for any logical expression. in this description, each segment of a statement containing a relational operator is called a condition. The NEXT statement is referred to as TRU. The NEXT +1 statement is referred to as FAL (note: FAL is not known at this time so an indication of 1 is put in for later processing in Section C).

1. Additional labels have been previously generated and are available in the Statement Label Table (SLT) for the conditions within the statement and for TRU.

2. Reference is given to the first label (in SLT) following the first condition (in ILT). Branch points are inserted in ILT based on an analysis of the secondary Level Analysis Table (LAT2). The following rules apply:

Functions of Section B2 Cont.

Let L be the level of the AND or OR immediately following the condition being processed, let AND_L be the level of the current or a sequential AND (in LAT2), and OR_L be the level of the current or a sequential OR.

(a) For the true branch:

(1) If $AND_L \leq L$, the branch is the next condition.

(2) If $OR_L < L$, $OR_L \rightarrow L$, and continue looking (as 1, above).

(3) If no $AND_L \leq L$, the branch is TRU.

(b) For the false branch:

(1) If $OR_L \leq L$, the branch is the next condition.

(2) If $AND_L < L$, $AND_L \rightarrow L$, and continue looking (as 1, above).

(3) If no $OR_L \leq L$, the branch is FAL.

These rules are applied for each condition in the statement.

3. The negation symbol (NOT) modifies the branch points in the following manner:

(a) The range of influence for a particular NOT is determined.

(1) If the NOT precedes the connectors (AND, OR), the range begins at the first condition of the statement.

(2) Otherwise, using the reference to the condition following the NOT, the range begins there and extends to the condition preceding the connector (AND, OR) whose level $\leq$ the level of the NOT.

(3) If no level is found as in 2, above, the extent is to the end of the statement.

(b) Determine the two relative branch points which are outside this given range (two and only two - one or both may be repeated).

(c) Interchange these two outside branches for each condition throughout the range.

This sequence is repeated for each NOT symbol in the statement. Obviously, if a condition falls within the range of more than one negation, it may have its branch points remodified.

Functions of Section B2 Cont.

After Section B2 is complete, all relative branches should be given in relative with the exception of FAL (indicated by integer 1). Because the statements processed here are only one at a time, the relative address is determined later (in Section C) when ILT is complete.

Functions of Section C

Section C completes the logical information and cross referencing in the internal tables. The main control (WJIPC*) determines the operation of the major areas of processing through the use of the sense indicators (Preset from WJIC*). It is possible to bypass or interrupt by presenting or dynamically setting the indicators.

----	4	5	6	7	---	10	11	---	----	21	22	23	24	---	27	-----

A bit in the position indicated will cause the associated result:

Position	Result
4	Bypass checking SLT
5	Bypass ILT - first pass.
6	Bypass storage allocation
7	Bypass ILT-second pass
10	Interrupt after completion of JILL Section.
11	Bypass processing VAT.
21	Interrupt after checking SLT
22	Interrupt after ILT - first pass-relative.
23	Interrupt after storage allocation.
24	Interrupt after ILT - second pass - absolute.
27	Interrupt after processing VAT

* Symbolic identification used in the Interpreter.

Functions of Section C Cont.

The major areas of processing are:

A. Checking the Statement Label Table (SLT) to determine that all labels are unique.

   1. The Hollerith labels and their associated procedure numbers (PNSL) are checked. If at least one pair of duplicate labels is found, the program dynamically sets the interrupt after ILT - first pass and continues checking.

   2. The routine will check for interrupt after SLT check before returning to main control.

B. The ILT Table is processed to complete the relative information (first pass). The operator value is used to determine the content of the entry and the sequence of information in ILT.

   1. For EQ, NQ, and (1, 2, 22), the status value in the right term (if any) is checked against the STAT Table entries for the status item in the left term for legality.

   2. For the relational operators (1-6), the branch points are checked for the FAL indication (1). If the indication is found, the SLT is used to determine the type of statement TRU is (the next statement). If TRU is simple, the next statement in SLT is FAL and the relative is inserted for the 1 in ILT. If TRU is compound, the next simple statement with a level less than TRU or the next compound statement, whichever is first sequentially in SLT, is FAL. For a description of the meaning of the branch points, see "Functions of Section B2".

   3. For a procedure call (50), the form value is checked to determine the type of procedure:

      a. For an ordinary procedure (LFORM of 50), the procedure name is found in SLT and the relative is inserted in ILT. The Variable Table is then searched for the first variable with VIP set and with a VPRO the same as the PNSL found in SLT. The next entry in ILT should be an input parameter (OPERI = 62). This entry is changed to a set (=) using the information from the variable table for the left term. All succeeding input parameters are processed in the same manner. When the first output parameter is found, the input parameter sets are moved up and followed immediately by the procedure call in ILT. Then the output parameters are processed in a similar manner with the exception of checking if VOP is set and using the VAT information for the right term in ILT. If no output parameters occur, the sets are moved and the sequence will

Functions of Section C Cont.

end with the procedure call. (please note: There must be at least one input parameter.)

b. For a special procedure (LFORM of 51-55), the program checks to see if there is at least one input parameter in the next entry, skips that entry, and continues the ILT processing.

4. Procedure and close declaration areas are handled in the following way:

a. For procedure declarations (46,59), the procedure number count (WPNO) is advanced and the procedure number image is set (WPNUI) from WPNO for use in determining the correct branch points for labels.

b. For procedure and close declarations (46,57,59), the relative address of the associated end is stored in a special table (WCTRL).

c. For returns (33), the entry is changed to a GOTO (31) and the last entry in the WCTRL table is subtracted from the WCTRL table. If the table is empty, the WPNUI is set to zero to indicate we are now in the main program.

5. All Hollerith labels are changed to relative addresses using the WPNUI to determine the correct range. The addresses in the Switch Table (SWT) are changed. If the Status Switch Declaration (47) is for a status item, the status values in SWT are checked against STAT for legality.

6. SLT and ILT are written in their relative format on tape C1. The program checks for interrupt before returning to the main control. If any errors had been found during the first pass at ILT, the program will interrupt at this point.

C. The Variable Table (VAT) is processed to allocate storage for the data tables.

1. The data is placed in high core $(77777)_8$ leaving a block for parameter items (if any--specified by HPARAM); each new table is placed in front of the previous one. The last table in VAT will be the first in the data block.

2. The table information in VAT is processed first so that the table size can be computed and the absolute address of the control word inserted into VAT. The total block is recorded in TABREG. Checks are made to see if the data tables exceed the storage available in the interpreter or the amount necessary for data reduction (DAISY). LIKE tables are cross referenced to their similar tables

Functions of Section C Cont.

and the information completed--their VTYPE is changed to Table (2).

3. The items are then assigned their absolute addresses. The parameter items are skipped at this time.

4. If no errors have occurred or the interrupt is not set, the routine will return to main control.

D. Storage is allocated for the permanent internal tables and recorded in the Master Table (MATTBL).

1. The Master Table is made up by using TABREG to indicate the last available space in front of the data block and determining the space needed for each table. MATTBL is processed in reverse order so that the first entry will refer to the first table in the internal table block. A check is made to see if the internal tables have exceeded the storage available in the interpreter.

2. Some communication registers are set at this time. The total number of words for SLT (including the control word) is placed in the decrement of the word following MATTBL. The current addresses (as used in interpreter first pass--not reflecting any allocation) of some internal tables are saved in the following registers: BVAT, BSTAT, BSWT, BILT (refering to the respective internal table names without the B).

3. If the interrupt is not set, the program will return to main control.

E. The ILT Table is processed (second pass) to assign the correct absolute address to all relative addressing (with the exception of subscripts and temporary storage). The operator value is used to determine the content of the entry and the sequence of information in ILT.

1. All term information (right, left, store) is checked for legality and the absolute addresses (if any) assigned.

2. The branch points and all relative addressing in ILT are assigned absolute addressing. Flags indicating relative 0 are interpreted (set in ILT--1st pass).

3. All the permanent internal tables as specified in MATTBL are written on Tape C1. If no errors have occurred or the interrupt is not set, the interpreter will return to main control which will return to TCP.

OUTPUT

Tables to be Used by JOLIZ

Five tables are written on tape C-1 for later use by JOLIZ. The number of words in these tables and their core addresses as computed by Section C are recorded in MATTBL, the master table which exists in permanent core. Each word in the master table contains the location and number of words for a specific table. The structure of this table is as follows:

Octal Register	Location Tag	Decrement	Address
37	MATTBL	# WORDS CON	STARTING ADDRESS CON
40	MATTBL+1	# WORDS SUB	STARTING ADDRESS SUB
41	MATTBL+2	# WORDS VAT	STARTING ADDRESS VAT
42	MATTBL+3	# WORDS VAT	STARTING ADDRESS SWT
43	MATTBL+4	# WORDS ILT	STARTING ADDRESS ILT

The following tables list all class and form values and their respective meanings. These values will be contained in any items which contain the letters CLAS or FORM in their name, also the item OPERI in ILT will contain one of the FORMS as noted by the asterisk (*).

Code Numbers

CLASS

1	Relational Operator
2	Logical Operator
3	Sequential Operator
4	Arithmetic Operator
5	Separator
6	Statement Label
7	Constant
8	Subscript
9	Subscripted Variable
10	Variable

Code Numbers Cont.

CLASS

11	Special Variable
12	Special Subscripted Variable
13	Temporary Storage
14	Accumulator
15	Declarators
16	Parameters

FORM

Relational Operators

*	0	NOP
*	1	EQ
*	2	NQ
*	3	LS
*	4	GR
*	5	GQ
*	6	LQ

Arithmetic Operators

*	7	+
*	8	-
*	9	*
*	10	/
*	11	OF

Logical Operators

	12	AND
	13	OR
	14	NOT

Separators

	16	(
	17	)
	18	,
	19	BEGIN
*	20	END
*	22	:=
	23	=:
	24	$\longrightarrow$
*	25	$\uparrow$
	26	$\downarrow$
	58	FOR SEPARATOR

Sequential Operators

	27	IF
*	30	STOP
*	31	GOTO
	32	FOR
*	33	RETURN
*	34	TERM
	35	START
	36	TEST

Special Variable & Special Subscripted Variables

37	ENT	
39		BIT
41		BYTE
42	VALUE	
43	NWDS	
44	NENT	
45	NWDSEN	

Declarators

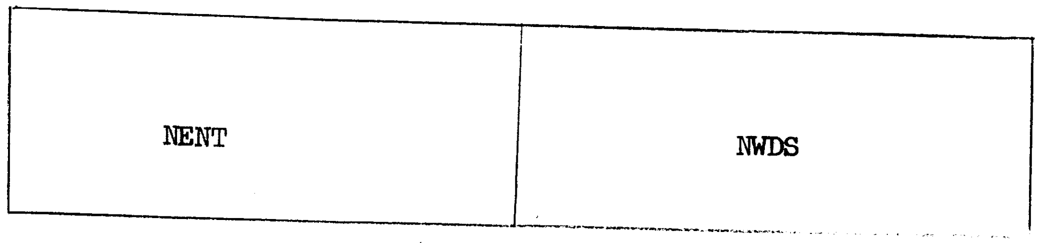
*	46	Procedure Declaration
*	47	Status Switch Declaration
*	48	Numeric Switch Declaration
	49	Switch Call
*	50	Procedure Call
	51	ABS
	54	CHAR
	55	MANT
	56	COMM
*	57	Close Declaration
*	59	One Output Procedures

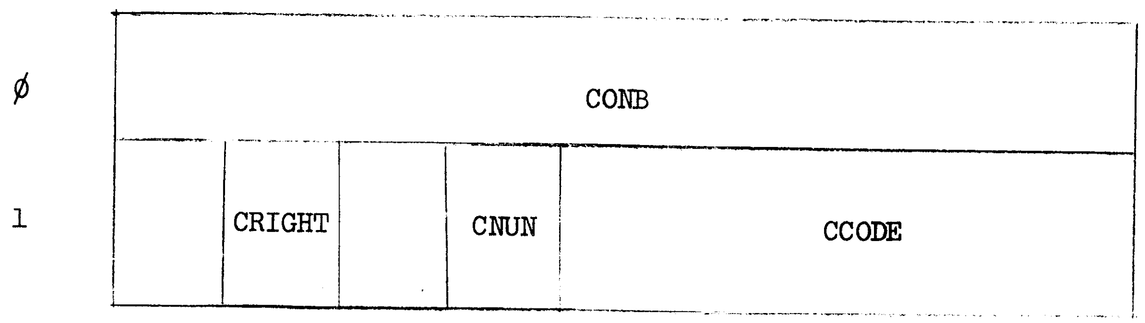
Parameters

*	62	INPUT
*	63	OUTPUT

* Will appear as OPERI in ILT

CONSTANT TABLE - CON





ITEM NAME	BIT LOCATION	DEFINITION
CONB	ø-35	Binary Value of Constant
CRIGHT	3-8	No. of bits to right of binary point (if CCODE = Fixed Fraction)
CNUM	12-17	No. of bits
CCODE	18-35	Constant Type: 1 - Floating 2 - Integer 3 - Hollerith 5 - Fixed Fraction

SUBSCRIPT TABLE - SUB

NENT		NWDS	
			0
SUBL		SUBV (for [A])	1
		(for [b])	2
		(for [c])	3
		(for [Z])	26
SIGNS	REPOS	NUMB	27

ITEM	BIT	DEFINITION OF THE ITEM
NENT	3-17	Number of entries currently in this table.
NWDS	21-35	Number of words currently in this table.
SUBL	3-17	Subscript level. Used by Section B only.
SUBV	21-35	Subscript value. Used by JOLLZ only. Relative position zero of the table is empty. Relative positions one through twenty-six are for the values of the subscripts A through Z in alphabetical order. The remaining words are used for subscripts that are constants, (e.g.,5) or incremented subscripts (e.g., J + 2). These words have the second format above the definitions that follow.
SIGNS	0-0	Zero is plus and one is minus.
REPOS	3-17	Relative position of this subscript in the first part of this table, if REPOS is zero the subscript is a constant.
NUMB	21-35	The actual constant in binary.

VARIABLE TABLE - VAT

NENT	NWDS

∅	VTAG						
1	VTYPE			VCODE		VLENT	
2	VFIX	VIP	VOP	VNENT		VLOC	VPAR
3	VSIGN	VDEFC	VDEFP	VRIGHT	VNUM	VPRO	

The final column in the table below lists the values of VTYPE for which each definition holds. Other conditions governing the range of individual definition are also listed. (Note that the VAT table used in making up the Baby Compool and the VAT table used by the final pass of the Interpreter are somewhat different.)

ITEM NAME	BIT LOCATION	ITEM DEFINITION	DEFINED FOR
VTAG	∅-35	Variable Name	I,T,P
VTYPE	∅-2	Variable Type: ∅-undefined 1-Item 2-Table 3-Parameter Item	I,T,P
VCODE	3-17	Variable Coding: ∅-Status 1-Floating 2-Integer 3-Hollerith 5-Fixed Fraction	I,P
VLENT	18-35	No. of words/Entry	T
		Absolute address in STAT table of first status value for this item (if VCODE=status).	I
VFIX	∅	Fixed/Variable length table indicator. ∅-Fixed 1-Variable	T
VIP	1	Procedure dummy-input parameter indicator: ∅-not parameter 1-input parameter	I
VOP	2	Procedure dummy output parameter indicator: ∅-not parameter 1-dummy output parameter	I
VNENT	3-17	No. of entries in table (Maximum if VFIX = variable).	T
		relative position of VAT entry for table containing this item.	I

ITEM NAME	BIT LOCATION	ITEM DEFINITION	DEFINED FOR
VPRO	18-35	No. of procedure for which this variable is defined (=ø if defined for whole program)	I,T,P
VPAR	0-35	binary value of parameter-item.	P
VSIGN	ø	Signed/unsigned indicator (if VCODE= integer or fixed fraction) ø-unsigned 1-signed	I,P
VDEFC	1	Compool defined indicator: ø-not compool defined 1-compool defined	I,T,P
VDEFP	2	Program defined indicator: ø-not program defined 1-program defined	I,T,P
VRIGHT	3-8	No. of bits to right of binary point (if VCODE-fixed fraction)	I,P
VNUM	12-17	No. of bits (if VCODE=integer or fixed fraction, or Hollerith).	I,P
		No. of status values (if VCODE=status)	I
VLOC	18-35	Absolute address of table's control word	T
		Absolute address of ITEM [ø]	I

NOTE: The first entry in VAT will always define a 'dummy' table which contains all items which are defined within a program but are not defined within any table.

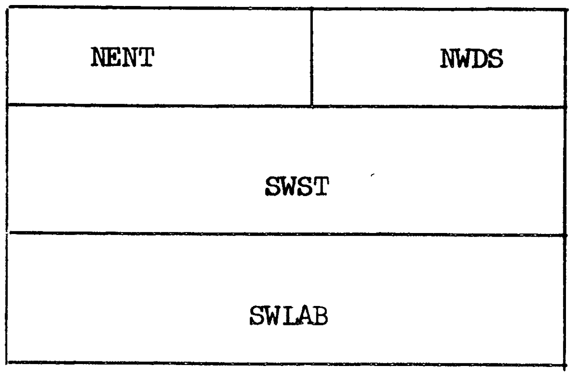
Items always set to constant values are:

VTAG	- 1TABLE
VTYPE	- TABLE
VFIX	- 1
VDEFP	- 1
VNENT	- 1
VPRO	- ∅

(The table name is not a legal variable name and cannot be duplicated by a program - or compool - defined variable name.)

SWITCH TABLE

SWT



ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
NENT	3-17	Number of entries currently in this table.
NWDS	21-35	Number of words currently in this table.
		ITEM SWITCH
SWST	0-35	Value
SWLAB	0-35	Absolute ILT Address of statement to which to transfer.
		SUBSCRIPT SWITCH
SWST	21-35	Relative position within the switch.
SWLAB	0-35	Absolute ILT address of statement to which to transfer.

Intermediate Language Table (ILT)

This table is serial in form and variable in length. Each entry contains five words but the format vaies. The format is dependent upon the operator (OPERI) of the entry. Therefore the format will be described according to their operator.

The control word of the table has the following format:

NENT	NWDS

ITEM NAME	BIT LOCATION	
NENT	3-17	Number of entries currently in this table.
NWDS	21-35	Number of words currently in this table.

The format of the entries fall into four major categories and can be designated for referencing as follows:
Category A - entry complete (full five words with three terms indicated: L-R-S)

SIGNL	RNEW	OPERI		LFORM	LCLAS	Left Term
LRELS				LRELV		(LLABL)
SIGNR				RFORM	RCLAS	Right Term
RRELS				RRELV		(RLABL)
STRUE				SFLSE		Store Term

ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
SIGNL	S	Sign of left term (no sign = 0)
RNEW	2	Set to 1 if new statement
OPERI	3-17	Operator
LFORM	24-29	Form of left term
LCLAS	30-35	Class of left term

ITEM NAME	BIT LOCATION	DEFINITION OF THE TERM
LRELS	3-17	Relative position of subscript for left term (if not subscripted, contains 0).
LRELV	21-35	Absolute address of the variable in the variable table for left term. If left term constant, LRELV is the absolute address of the constant in the constant table. If left term is temporary storage, LRELV is the T#.
Note: LLABL	0-35	Label of whole left term (word for LRELS & LRELV)
SIGNR	S	Sign of right term (no sign = 0)
RFORM	24-29	Form of right term.
RCLAS	30-35	Class of right term.
RRELS	3-17	Relative position of subscript for left term. (If not subscripted, contains 0)
RRELV	21-35	Same as LRELV, except for the right term.
Note: RLABL	0-35	Label for right term.
STRUE	3-17	Store Class
SFLSE	21-35	Relative address in temporary storage table. (If accumulator, will be zero.)

Category B - Entry complete (full five words with two terms indicated: L - R; and last two words used as branch points)

Category B Cont.

SIGNL	RNEW	OPERI			LFORM	LCLAS	Left Term
LRELS				LRELV			
SIGNR					RFORM	RCLAS	Right Term
RRELS				RRELV			
STRUE				SFLSE			

For description of first 4 words (L-R terms), see Category A above.

ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
STRUE	3-17	Branch point(absolute address in ILT) if relationship is true.
SFLSE	21-35	Branch point (absolute address in ILT) if relationship is false.

Category C - Only two terms (L-R) in entry (last word contain zero).

For description of first 4 words (L-R terms) see Category A above.

Category D - Special

The content of the entry format will be described in detail under respective use of Category D.

OPERI: 0 Indicates an entry in ILT expressly for the purpose of providing additional information for the preceding entry (this is needed if the proceeding entry contains special subscripted variable of bit or byte form). This additional information may refer to the left and/or right term of the entry.

If a bit or byte entry is present in ILT.

The format is similar to Category C (however, one term may be zero) with the following definitions:

ITEM	BIT LOCATION	DEFINITION OF ITEM
LRELS	3-17	Relative position of the first bit or byte in the subscript (sub) for the left term.
LRELV	21-35	Relative position of the number of bits or bytes in the subscript (SUB) for the left term.
RELS	3-17	Same as LRELS except for right term
RELV	21-35	Same as RRELV except for right term

OPERI = 1-6 Relational operators (EQ[1], NQ[1], LS, GR, GQ, LQ)

Category B

OPERI = 7-10 Arithmetic operators (+,-,*,/)

Category A

Some Separators

OPERI: 22 SET (:=)[1]

Category C

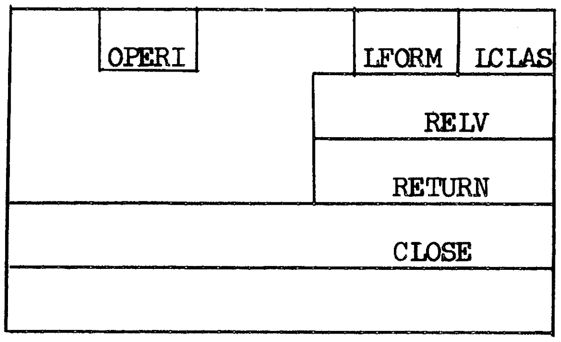
OPERI: 25 (1) Exponential

Category A

1. It is possible to have a special variation on the right term format for OPERI = EQ, NQ or :=. In this case the RCLAS will be special variable (11), the RFORM will be Value (42). Instead of relative addresses, the following word will contain the hollerith name of status value (0-35). In this case, the left term must be the status item.

OPERI: 20 END

Category D

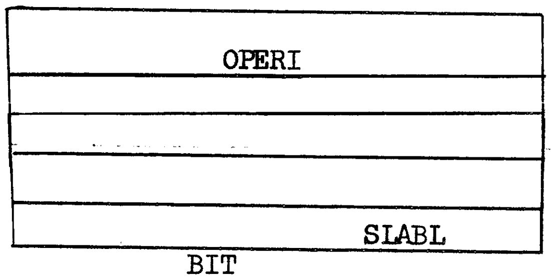


ITEM	BIT LOCATION	DEFINITION OF ITEM
OPERI	3-17	Operator
*LFORM	24-29	FORM } Zero is not a one output procedure.
*LCLAS	30-35	CLASS } Zero is not a one output procedure.
*RELV	21-35	Absolute location in the variable table of the procedure's output if it is a one output procedure.
RETURN	21-35	Interpreter uses this location to hold its return address while operating.
CLOSE	S-35	If word is zero the END is for a close type procedure.

Some Sequential Operators:

OPERI: 30 STOP

Category D

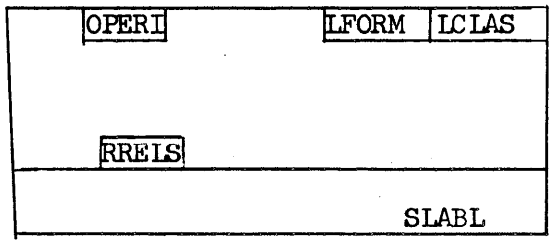


ITEM	BIT LOCATION	DEFINITION OF ITEM
OPERI	3-17	Operator
SLABL	0-35	Absolute address of branch point in ILT. (If no branch, contains zero)

* Used with one output procedures only.

OPERI: 31 GOTO

Category D



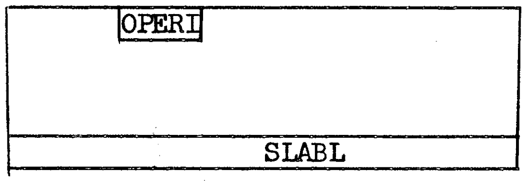
ITEM	BIT LOCATION	DEFINITION OF ITEM
OPERI	3-17	Operator
LFORM	24-29	Form of branch address
LCLAS	30-35	Class of branch address
SLABL	0-35	Absolute address of branch point in ILT.
RRELS	3-17	Relative address of subscript if LFORM is a switch call.

(1) If LFORM is for Item switch, this subscript will be used to find the correct item (relative by entry) for comparing on switch values.

(2) If LFORM is for a subscript switch, this subscript will be used to compare with the switch values for selecting the correct branch location.

OPERI: 34 TERM

Category D

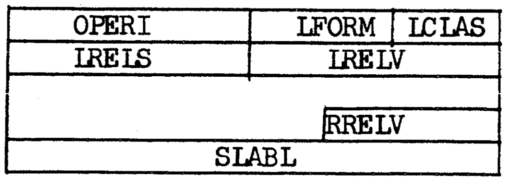


See OPER: 30 STOP above for description of format content.

FOR SWITCHES:

OPERI: 47 Item Switch Declaration

Category D



ITEM	BIT LOCATION	DEFINITION OF ITEM
OPERI	3-17	Operator
LFORM	24-29	Form of
LCLAS	30-35	Class of
LRELS	3-17	Number of entries in switch table (SWT)
LRELV	21-35	Absolute location of initial entry for this switch in SWT Table (SWT).
RRELV	21-35	Absolute position in variable table of item to be tested.
SLABL	0-35	Hollerith label of switch.

OPERI: 48 Subscript Switch Declaration

Category D

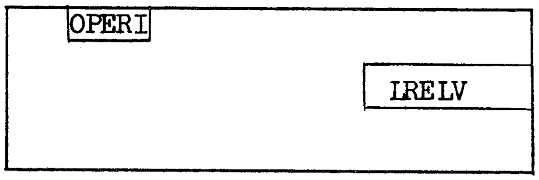
Same format as OPERI: 47 SSD above with the omission of RRELV (no item concerned).

Switch calls appear in the ILT table as GOTO (for OPERI) statements. Check above for format.

Procedures Declarations:

OPERI 46 Procedure Declaration

Category D



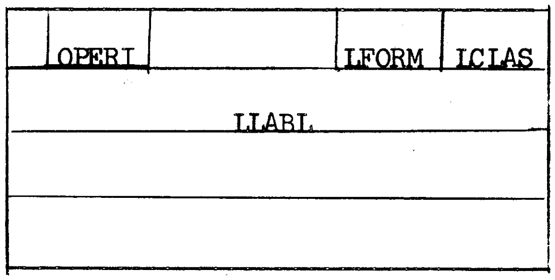
ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
OPERI	3-17	Operator
LRELV	21-35	Absolute location of the END associated with this procedure.

OPERI 59 One Output Procedure.

The same format as OPERI 46

OPERI 50 Procedure Call

Category D

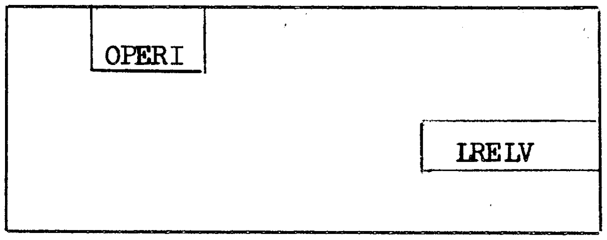


ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
OPERI	3-17	Operator
LFORM	24-29	Form will be one of the following: 50 Procedure Call 51 ABS
LCLAS	30-35	Class
LLABL	0-35	Absolute address in ILT of the procedure requested.

Note: All entrance and exit parameters are set with assignment statements preceding and following the procedure call.

OPERI 57 Close Declaration

Category D



ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
OPERI	3-17	Operator
LRELV	21-35	Absolute location of the END associated with this close routine.

Since there are no parameters associated with a CLOSE routine, the call to it is a GOTO.

Output for DAISY

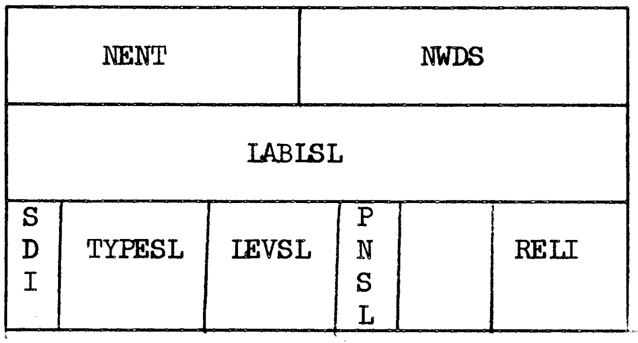
In addition to the tables written on Tape C-1 for JOLLZ, two tables are written for the data reduction program on the same tape, preceding the five previously discussed records. One is ILT, which has the same format as described in the Output for JOLLZ, with the following modifications:

LRELV and RRELV contain the relative addresses of the variables (or constants) in their respective tables. STRUE and SFALSE contain relative rather than absolute addresses in ILT. SLABL contains a relative address in ILT.

The other table written on Tape C-1 for DAISY is the Statement Label Table (SLT). Its format is described in the following:

STATEMENT LABEL TABLE

SLT



ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
NENT	3-17	Number of entries currently in this table.
NWDS	21-35	Number of words currently in this table.
LABLSL	0-35	Statement label. Hollerith
SDI	0	1 = Switch Declaration
TYPESL	2	Type of statement. 0. Conditional 1. Simple 2. Compound
LEVSL	3-14	Level of this statement.
PNSL	15-20	Procedure number
RELI	21-35	Relative position in the Intermediate Language (ILT) table.

Output Used by Other Programs

Certain control registers are set in permanent core for use by other programs. Addresses in these registers are the locations in core of tables left by JALLZ.

LOCATION	LOCATION TAG	CONTENTS
31	BILT	Decrement contains number of words in ILT.
32	BSWT	Address contains address of switch table.
33	BSTAT	Address contains address of status table.
34	BVAT	Address contains address of variable table.
45	TABREG	Decrement contains number of words of table data. Address contains starting address of table data.
54	HPARAM	Address contains number of parameter items contained in the variable table.

Output of Tables of Constants

When tables of constants are defined by the JOVIAL Source Program, these are converted to binary and saved on tape D-1 by JALLZ. Each record of this tape contains the constants for one item. The address of the first word of each record contains the relative position in the variable table of the item whose contents are contained in this record. An end-of-file is written to mark the last record on this tape.

Output for Baby Compool Assembly Program

At the conclusion of JALLZ, certain tables are left in core memory for use by the Assemble Baby Compool Program.

These are ILT, SWT, and VAT as described in "Output for JALLZ" and the Status Table (STAT) as described in the following figure.

STATUS TABLE

STAT

NENT	NWDS
SVALUE	

ITEM	BIT LOCATION	DEFINITION
SVALUE	∅-35	Status item value (Hollerith)

The STAT Table contains the list of Hollerith status values for each defined status item. For a status item entry in VAT, VLENT contains the relative position in STAT of the first SVALUE listed for the item; VNUM contains the number of SVALUEs defined for the item.

Intermediate Outputs

During the operation of JALLZ, certain tables are formed by the various sections and used by themselves and other sections. Although the existence of these tables is sometimes brief and they are not among the final output of JALLZ, this section contains a description of them for informational purposes and also because they are referred to in other sections of this document.

LEVEL ANALYSIS TABLE 1

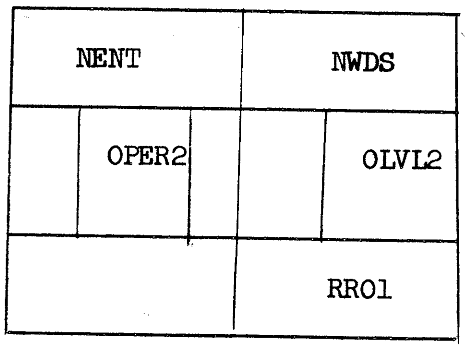
LAT1

	NENT				NWDS		
	LSIGN		OLVL		OPER	FORML	CLASL
LABLL	RELSL				RELVL		
	IGN	RELBL			TRUE		

ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
NENT	3-17	Number of entries currently in this table.
NWDS	21-35	Number of words currently in this table.
LSIGN	S	The sign of the term: Zeros for plus and one for minus.
OLVL	3-17	Level of this operation.
OPER	18-23	Operation
FORML	24-29	Form
CLASL	30-35	Class
RELSL	3-17	Relative position in the Subscript (SUB) table.
RELVL	21-35	Relative position in the Variable (VAT) table.
IGN	S	Used for control of rel. max. points.
RELBL	3-17	Relative position in BAB table.
TRUE	21-35	Used for control of rel. max. points.

LEVEL ANALYSIS TABLE 2

LAT2



ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
NENT	3-17	Number of entries currently in this table.
NWDS	21-35	Number of words currently in this table.
OPER2	3-17	Operator. (AND or OR)
OLVL2	21-35	Level of operator.
RRO1	21-35	Relative position of ILT for relational operator following AND or OR.

NOT TABLE

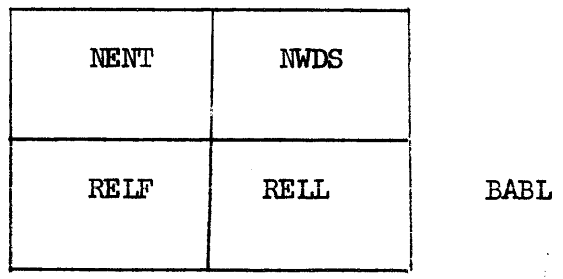
NOT

NENT		NWDS	
NSIGN	RELA		OLVLN

ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
NENT	3-17	Number of entries currently in this table.
NWDS	21-35	Number of words currently in this table.
NSIGN	S	Used to flag if NOT comes before any AND or OR in statement (will be negative with RELA of Ø).
RELA	3-17	Relative position in LAT2 table.
OLVLN	21-35	Level of the NOT.

BIT AND BYTE TABLE

BAB

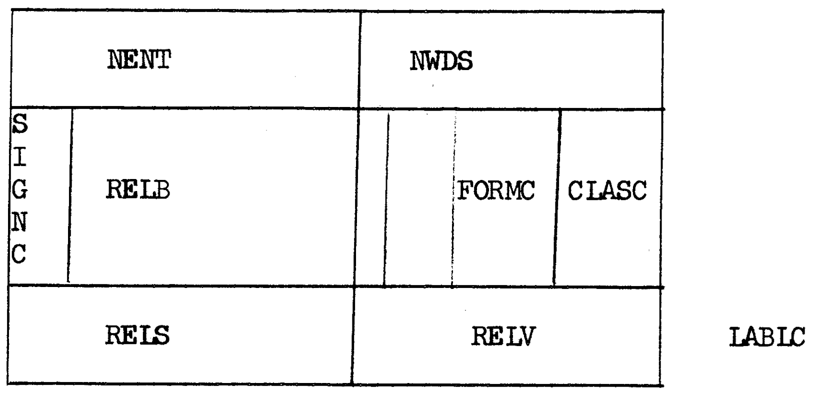


ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
NENT	3-17	Number of entries currently in this table.
NWDS	21-35	Number of words currently in this table.
RELF	3-17	Relative position of the first bit or byte Subscript (SUB) table.
RELL	21-35	Relative position in the subscript table of the number of bits or bytes.
BABL	0-35	

CARD ANALYSIS TABLE

Exists only for the processing of one.

CAT



ITEM NAME	BIT LOCATION	DEFINITION OF THE ITEM
NENT	3-17	Number of entries currently in this table.
NWDS	21-35	Number of words currently in this table.
SIGNC	0	Sign of term.
RELB	3-17	Relative position in the Bit and Byte (BAB) table.
FORMC	24-29	Form.
CLASC	30-35	Class
RELS	3-17	Relative position in the Subscript (SUB) table.
RELV	21-35	Relative position in the variable (VAT) table. or constant (CON) table.
LABLC	∅-35	

MESSAGE PRINTOUTS

Two types of output are printed by JALIZ. One is the START card, which is logged in its entirety as soon as it is read in. The other output available to the user consists of error messages. The format of the message is usually a function of the particular section of JALIZ from which it emanated. Except for those messages which are explicit and complete in their printed form, messages are given a unique reference number. The following discussion will include a list of these messages and their corresponding reference numbers (where applicable). Where it seems desirable, a fairly complete description of how the error is detected will be given.

In almost all cases, the existence of one or more of these messages will indicate that the program will not be allowed to go to JOLIZ. The only recourse the user has is to repair his errors and try again.

Error messages will be described under the particular section of the program which produces them. A description of these sections is included in "Functions of JALIZ".

Errors of Section A

The errors of this section will be printed out in the following format:

"MESSAGE STATEMENT LABEL + XXXXXX REFERENCE XXX"

The messages and their respective reference numbers and explanations are in the following list.

"ILLEGAL FIRST WORD"

21. First significant word[1] is non-alphanumeric first significant word[1] must be alphanumeric.
22. First significant word[1] is illegal JOVIAL terminology.
23. First significant word[1] has an illegal character following it.
24. First significant word[1] forms an illegal combination with word following it.

"EXCESSIVE WORD LENGTH"

25. Word is more than six consecutive alphanumeric characters.

"ILLEGALLY FORMED STATEMENT"

26. Illegal use of decimal point in this statement. Decimal point is used only in constants and following leading statement label.

27. Illegal character following label. Must have dollar sign, indicating end of statement, or bracket.[2]

28. Unrecognizable character for JOVIAL terminology found.

29. Should have set (=) at this point, but was not found.

30. Found comma, yet we are not in a "FOR" statement nor can comma be interpreted as parameter separator at this point.

31. Illegal to have equal sign at this point.

32. Statement has excess of words following legitimate portion of statement.

33. Illegal for status value type of operator to be first significant word of statement.

34. Illegal character in status value type of operator. Should be of form, V (STATUS).

"ILLEGAL CONSTANT"

36. Have accuracy factor twice in same constant.

37. Illegal letter found in constant. Only letters A, accuracy, H, Hollerith, and E, exponent, are allowed.

38. Letter H found, but is illegally positioned in constant.

39. Hollerith constant more than six characters long.

40. Right parentheses on Hollerith constant is either missing or is illegally positioned.

41. Constant should be integer in this case, but is not an integer.

42. Alphanumeric word and constant form an illegal combination in this statement.

"ILLEGAL VARIABLE"

43. Variable is an illegal length or has an illegal character contained within it. All variables must be from two to six alphanumeric characters, beginning with a letter.

44. Variable is followed by an illegal character. All variables in this case must be followed by a right parenthesis or, if a subscript follows, by a bracket. (2)

45. Illegal character follows this variable. Variable starting statement must have SET (=) following it.

46. Illegal subscripted variable position. Subscripted variable is used instead of variable. However, program will be compiled correctly.

"ILLEGAL SUBSCRIPT"

47. Subscript is illegal. Subscript must be either a single letter or an integer.

48. Subscript is followed by an illegal character. All subscripts in this case must be followed by a bracket (2) or, if increment or decrement follows, by a plus or minus sign.

49. Subscript is incremented or decremented by a non-integer. Only integer is legal in this case.

50. Subscript is illegal. Only can have single letter in this case.

"ILLEGAL ENTRY OPERATOR"

51. Entry is followed by an illegal character. Only set (=) can be used following entry.

52. Entry is set illegally. Entry may be set only to another entry or to zero.

"ILLEGAL BIT/BYTE OPERATOR"

53. Bit/Byte subscripted position is followed by an illegal character. This initial position must be followed by a bracket (2) or, if end position follows, by a comma.

54. Bit/Byte final bracket is followed by an illegal character. Left parentheses and variable are expected, but left parentheses is missing.

55. Illegal Bit/Byte subscripting. Subscript of Bit/Byte word must be single letter or integer.

"ILLEGAL STATEMENT LABEL"

56. Illogical to label this type of statement label will be deleted. However, program will be compiled correctly unless transfer is made to this label.

57. Label is illegal. Label must be two to six alphanumeric characters, beginning with a letter.

"ILLEGAL BEGIN OPERATOR"

---

(1) First significant word is not necessarily the first actual word of the statement. For example, a statement label preceding an "IF" statement places the first word, "IF", in a second word position.

(2) Check hardware language for bracket.

58. Illogical to precede this statement with a "begin" operator. However, program will be compiled correctly as long as there is a compensating "END" operator following.

59. Illegal to precede this statement with a "BEGIN" operator. "BEGIN" will be deleted and compiling will continue.

"ILLEGAL SWITCH"

60. Label of switch is less than two alphanumeric characters. Label must be two to six alphanumeric characters beginning with a letter.

61. Illegal character following switch label. Must have set (=) for numeric switch and left parentheses for status switch following switch label.

62. Illegal character following status item. Should have set (=) at this point.

63. Illegal character following set (=). Should have left parentheses at this point.

64. Illegal for status value to be special character. Must have number or alphanumeric word as status. Can use Hollerith to indicate special character.

65. Separator between status value and switch branch label is illegal. Should have equal sign (=) in hardware language.

66. Illegal switch branch label. Label must be two to six alphanumeric characters beginning with a letter.

67. Illegal character following switch branch label. Must have comma or right parentheses at this point.

"ILLEGALLY SIGNED EXPRESSION"

68. Illegal to precede left parentheses with arithmetic sign. Sign will be ignored and program will continue to be compiled.

69. Illogical to have a series of arithmetic signs. Correct arithmetic sign will be computed and program will continue to be compiled.

"ILLEGALLY FORMED STATEMENT"

70. Illegal word or character following alphanumeric word.

71. Illegal word or character following arithmetic sign.

72. Illegal word or character following left parentheses.

73. Illegal word or character following right parentheses.

74. Illegal word or character following relational operator.

75. Illegal word or character following logical operators: AND, OR.

76. Illegal word or character following dollar sign.

77. Illegal word or character following up arrow.

78. Illegal word or character following down arrow.

79. Illegal word or character following value type item.

80. Illegal word or character following logical operator, NOT.

81. Illegal word or character following constant.

"ILLEGAL FOR STATEMENT"

82. Illegal character following subscript. Should have set (=) following.

83. "ALL" type of "FOR" statement recognized. Should have item enclosed in parentheses following the word "ALL", but was not found.

84. More than two commas found in "FOR" statement.

87. Comma not found following "B" factor of "FOR" statement.

"ILLEGAL PROCEDURE STATEMENT"

88. Procedure label is illegal. Label must be from two to six alphanumeric characters beginning with a letter.

"ILLEGAL VARIABLE DEFINITION"

89. Illegal position for item/table definition statement. All definitions must be at beginning of program or, if procedure variable definitions, at beginning of procedure.

"TABLE LENGTH EXCEEDED"

90. Constant table exceeded. Storage of succeeding constants will be stored at start of constant table once again.

91. Subscript table exceeded. Storage of succeeding subscripts will be assigned duplicate relative positions at start of subscript table once again.

92. Bit/Byte table exceeded. Program unaffected.

93. CAT table exceeded. This statement rejected.

<u>Errors of Section A1</u>

The standard format for an error printout of Section A1 is the following:

"VARIABLE DEFINITION ERROR N"

Where N is the reference number. Listed below are the reference numbers, together with the associated error types.

95. Undefined Variable - A referenced variable has not been previously defined by the program, and is not listed in the compool.

96. Duplicate Definition - The variable defined by the declaration has been defined previously by the program.

97. Variable Definition Limit Exceeded - The total number of variables defined by the program and compool defined variable referenced by the program exceeds system capacity.

98. Status-item Value Limit Exceeded - The total number of status values for all program defined status items and compool defined status items referenced by the program exceeds system capacity.

99. Identical Table Limit Exceeded - The number of Identical Tables defined by the main program and any one procedure exceeds system capacity.

100. Item or Table Declaration does not, end with $

101. Table Name Format Error

102. Table Type Format Error

103. Error In Table No. of Entries

104. Identical Table name Format error (No. characters < 3)

105. Illegal Identical Table - No table has been previously defined (or is listed in the compool) to which this table can be made identical.

106. No End to table - The end bracket which terminates definition of items within a table is missing.

107. Item Name Format Error

108. Item Type Format Error

109. Undefined Compool Item - A compool defined item referenced by this table is now undefined because its table has been redefined by the program as other than a table.

110. Status Item Has No values

111. Status Item Has Too Many Status Values (>65)

112. Error in No. of Bits or No. of Bits to Right of Point

113. Signed/Unsigned Error

114. Parameter Item Value Error

115. Illegal Item Value in Table of Constants

116. Too Many Constants in Table of Constants

117. No END to Constant List - The END bracket which terminates a list of constants is missing.

118. Procedure Heading Format Error

119. Procedure Dummy Parameter is Undefined

120. Procedure Dummy Parameter Illegally Defined - As parameter item, status item, table, or item within a table.

Errors of Section B

The error printouts logged by this section are considered to be descriptive enough so that further description of them in this document via numbers appears unnecessary. Following is a list of these error messages.

All error messages are composed of two printed lines. The first line is, "Statement Label XXXXX rejected by ILT analysis". The second line is any one of the following fourteen specific messages:

"THE CARD ANALYSIS TABLE IS EMPTY"

"THIS SUBSCRIPT IS CURRENTLY IN USE"

"THE PROGRAM HAS THE FOLLOWING NUMBER OF EXTRA BEGINS"

"CANNOT TEST AN INACTIVE SUBSCRIPT"

"SWITCH DECLARATION IS INCORRECT"

"PROCEDURE DECLARATION WITHIN A COMPOUND"

"THIS END HAS NO BEGIN FOR IT"

"STATEMENT HAS AN INCORRECT OPERATOR"

"SUBSCRIPT NOT SET BY A FOR STATEMENT"

"INTERMEDIATE LANGUAGE TABLE EXCEEDED"

"STATEMENT LABEL TABLE EXCEEDED"

"ACTIVE FOR STATEMENT TABLE EXCEEDED"

"SWITCH TABLE EXCEEDED"

"END TABLE EXCEEDED"

Errors of Sections B1 and B2

If any errors are detected by either Sections B1 or B2, a message is printed for the cause, another for the reject (indicating the routine) and indication is made to the control operation of the Interpreter. If an error occurs in Section B1, the routine rejects the entry and continues processing the statement in the Card Analysis Table (CAT). The statement is rejected when the processing is complete. In section B2, the statement is rejected immediately when an error is detected.

The reject message is:

```
STATEMENT ØØØØØØ + ØØØØ REJECTED BY XXXXXX ANALYSIS;
```

Where XXXXXX is "LEVEL" for section B1 and "LOG" for section B2. The Hollerith Statement label follows STATEMENT. A decrement value follows + (since not all statements have programmer defined labels).

Section B1

The level analysis routine expects the CAT Table to have a basic sequence of alternating terms with operator (A+B-C, etc.) with separators appropriately placed.

The terms are recognized by the Class value ($6 < \text{CLASS} \leq 16$).

The operators and separators are determined first on Class value ($1 \leq \text{CLASS} \leq 5$) and secondly on form value. The following list contains the legal forms:

form value	description
1 - 6	relational operators
7 - 10	arithmetic operators
11	OF operator
12	AND operator
13	OR operator
16	left parenthesis - separator
17	right parenthesis - separator
18	comma - separator
22	set (:=) operator
23	=: separator
25	up exponent - operator (separator)
26	down exponent - separator
58	comma - separator (special)

The separators are reflected in the counter which determines the level value assignment for the operators.

In most messages, the relative address in the CAT Table (RELC) is given for reference.

Message:

INCOMPLETE STATEMENT IN CAT RELC ØØØØ

only a partial statement in CAT (ends with an operator).

Message:

LEVEL COUNTER VALUE INCORRECT ØØØØ

Reason: Illegal use of separators. Check to see for appropriate pairing of parenthesis, etc. Check is made at the end of the statement.

Message:

WRONG CLASS - FORM IN CAT RELC ØØØØ

Illegal form - class for entry. If terms and/or operators are out of sequence, the form or class is considered illegal.

Message:

ØØØØØØ TABLE LIMIT EXCEEDED RELCØØØØ

As an entry is added to each internal table, the length is checked first. If the limit is exceeded, the CAT entry is rejected and the table name is inserted in the message (NOT, LAT1, LAT2). Obviously, all subsequent CAT entries calling for the same table will be reject.

Possible Error Messages - Section B2

This section is given an initial entrance parameter into the Statement Label Table (SLT). This is the label of the first entry in the ILT after an entry containing a relational operator. The routine will then process all such entries, inserting the correct branch points and allowing for the connectives (AND, OR) and negation (NOT). A certain amount of legality checking is done.

Message:

ILT TABLE INCORRECT RELI ØØØØ ALE1

Reason:

Only a partial statement in ILT or the referencing from the SLT is incorrect (not after relational operator, etc.)

Message:

LAT2 TABLE INCORRECT REL2 ØØØØ ALE2

Reason:

Incorrect operator value (not AND or OR, i.e. 12 or 13)

Message:

SLT TABLE INCORRECT RELS ØØØØ ALE3

Reason:

Insufficient generated (or programmer-defined) labels in LST for relational operator entries in ILT. The next two conditions and/or statements should be labeled.

Message:

NEGATION ERROR - EXTENT = RELI ØØØØ ØØØØ ALE4

Reason:

Lack of correlation between NOT table and ILT. The extent of range of the NOT operator causing the difficulty is given. Within that range there should be two and only two extension (outside of range) branch points.

Errors of Section C

If any error occurs in this section, the operation of the Interpreter will be interrupted, causing the test program to be rejected. The cause of the interrupt will have been previously indicated. In order to catch as many errors as possible, this section operates in the following manner:

(1) If errors occur in checking the Statement Label Table (SLT) for doubly defined labels, the SLT for doubly defined labels, the SLT processing (*WSRS) and the first pass (relative) in processing the Intermediate Language Table (ILT) will be completed before interrupt. Please note: The doubly defined labels will cause illegal relative addressing in ILT (pass 1).

(2) If errors occur during the first pass at ILT (WILT 1), Section C will continue to the end of the pass before interrupt.

(3) If illegal information is discovered in processing the Variable Table (VAT) for storage allocation (WVTA), VAT is completely processed before interrupt.

* Interpreter reference symbols.

If the amount of core required for the data tables (as specified in VAT) exceeds the available storage, the section will interrupt as in (5) below.

(4) Calculation (end of WVTA) is made with respect to the input data. If the result is greater than the storage allotted in the Interpreter System for data reduction (DAISY), Section C is interrupted. The following formula is used for the calculation (expressed in JOVIAL):

```
IF 3*(NENT(VAT) + TABREG) + NENT (STAT)

     +2 + HPARAM LQ DSYLMT$ GOTO CONT$

     GOTO INTER$

WHERE TABREG = length block of data tables

       HPARAM = length of parameter items block

       DSYLMT = limit defined by Interpreter system
```

Note: 2 is added for control words needed for VAT and STAT.

CONT indicates continue

INTER indicates interrupt

(5) If the allocation for the internal tables exceeds available storage (WSTOR), the Interpreter interrupts.

(6) If errors occur during the second pass at ILT (absolute - WILT2), the Interpreter will continue to the end of the pass before interrupt.

POSSIBLE ERROR MESSAGES

The following are the error messages from Section C. The referencing on the right refers to the symbolic identification used.

Message:

PROGRAM INTERRUPTED IN JILL SECTION, CHECK INTERPRETER, ERROR LIST

Reasons:

(a) The sense indicators are pre-set to interrupt after some routine (see explanation of SI control-WJIC).

(b) If errors are detected during a given processing routine, the sense indicators are dynamically set to interrupt (see preceding explanation of error recognition).

Message:

```
ILLEGAL PROGRAM OPER. XR1 ØØØØØ  XR2 ØØØØØ  XR4 ØØØØØ
```

This message should not occur in normal program testing. It usually indicates that the interpreter deck is out of order or instructions are missing due to system or machine failure. It could also indicate possible program error.

XR4 should give the complement of the absolute address of the location of the error test.

Message:

```
DOUBLY DEFINED STATEMENT LABELS

LABEL PNSL RELI  LABEL PNSL RELI

ØØØØØØ ØØØ  ØØØØ ØØØØØØ ØØØ  ØØØØ
    |                     |
    |                     |
    |                     |
    v                     v
END DOUBLY DEFINED LABELS
```

Heading and end messages are printed out if there is at least one pair of labels equal. Equal labels are always printed in pairs. If three or more labels are equal, there will be a certain amount of repeats (i.e., if 1, 2, ..η equal labels, label numbers 2, 3, ..η - 1 will be repeated). Conditions for equality (specified in JOVIAL):

```
IF LAB1 EQ LAB2 AND PNSL1 EQ PNSL2$

GOTO PRINT$

GOTO CONT$
```

Where LAB1 and LAB2 are the Hollerith labels, and PNSL1 and PNSL2 are the procedure numbers associated with the statement labels (integers).

Messages for Processing ILT

Messages for indicating errors in ILT processing can be printed out in either pass. The numbers given at the right of the explanation indicate which pass (1,2). To cut down on the number of stored messages, the same message was used in similar cases with a specific reason number. All messages give reference relative information.

Message:

```
UNDEF. LABEL: ØØØØØØ  PNSL ØØØØ  RELI ØØØØ
```

If a given ILT entry contains a Hollerith label or reference to Hollerith labels to be converted to a relative address (RELI), this label is used to search the SLT table together with the current procedure number (WPNUI).

Conditions for a label to be undefined (expressed in JOVIAL):

```
FOR I = ALL (SLT) $

BEGIN  IF  LABI EQ LAB2  I   AND

WPNUI  EQ  PNSL   I   $   GOTO  CONT$

END  PRINT. ≤ , ≤ .....
```

There LABI is the label to be defined and WPNUI is the current procedure number.

If the above message is printed out for a procedure call label, the following message is printed, also:

Message:

```
SINCE PROC. CALL LABEL, I-O PARAMETERS

NOT PROCESSED.
```

The Input-Output parameter entries for that procedure call should then print-out as illegal operators (see below - WEIL).

Message:

```
OPER VALUE ØØ INCORRECT IN ILT

RELI ØØØØ
```

Only certain operators are expected to appear in ILT at the time of the JILL processing. If an entry is rejected, subsequent errors can result, or, this reject could be caused by a previous error.

The following operator values are legal for a given pass:

Oper.	Pass 1	Pass 2
BAB	0	0
Rel. operators	1 - 6	1 - 6
Arith. operators	7 - 10	7 - 10
STOP	30	30
GOTO	31	31

RETURN	33	
TERM	34	34
END	20	20
SET	22	22
↑	25	25
Procedure Declaration	46	46
Item Swt. dec.	47	47
Subscript Switch Declaration	48	48
Procedure call	50	50
Close declaration	57	57
Procedure declaration (1 output)	59	59

Message: WEI2

"ENTRY DATA WRONG IN ILT REAS. ØØØ"

RELI ØØØØ.

Reason Number	Explanation
1	Left term not a variable for status value (right term).
2	Branch points not set for relational operators.
3	Form value not correct for procedure call operator.
4	ILT entries out of sequence. No close declaration or procedure declaration before RETURN.
5	ILT entries out of sequence. No close declaration or procedure declaration before END.
6	No label in GOTO entry.

8		Store class value incorrect.
9		Illegal term information for right or left term (absolute addressing). Check table size on relative addresses.
10		Branch points (absolute addressing) incorrect. Check table size (ILT).
11		No RELI for end for procedure declaration or close declaration.
12		Illegal class for variable in END entry (assuming one output procedure end).
13		No input parameter after special one output procedure call.
14		Interpreter limits exceeded. Too many nested procedure and close declarations for table controlling (CTRL) the associated ENDs.

Message: ØØØØ TABLE INCOMPLETE

REAS.ØØØ RELI ØØØØ

Reason number	Insert	Explanation
1	VAT	Variable of left term not correct for status item (information in variable table).
2	STAT	Status illegal (right term) for variable designated in left term.
3	SLT	SLT does not contain RELI of next two conditions and/or next two statements.
4	VAT	Variables do not correspond to I-O parameters give in ILT for procedure call.
5	ILT	No input parameter for special one output procedure call.

Message:

```
ILLEG. STATUS ØØØØØØ FOR SWT

RELI ØØØØ
```

If a status is used for an Item Switch declaration, which is not in the Status Table (STAT) for the variable given, this message is printed and program continues operating on the rest of the statuses for that switch.

POSSIBLE ERROR MESSAGES - VARIABLE

TABLE AND STORAGE ALLOCATION

In processing the Variable Table (VAT) for allocating storage and inserting absolute addresses, the following messages may be printed. If an error is indicated, the entry is rejected. Note: The processing is done in two passes: 1st pass, tables and like tables; 2nd pass, items (parameter items are skipped on both passes - they are processed later).

Reason #	Explanation
1	VTYPE wrong - 1st pass The following are legal: 0 - undefined - skipped 1 - Item - skipped 2 - table - processed 3 - parameter item - skipped 4 - like tables - processed (and changed to 2)
2	VTYPE wrong - 2nd pass The following are legal: 0 - undefined - skipped 1 - item - processed 2 - table - skipped 3 - parameter items - skipped

Checks are made to determine if the program input data will fit into available storage. The following message is printed and the interpreter interrupted, if the data exceeds the capacity.

Message:

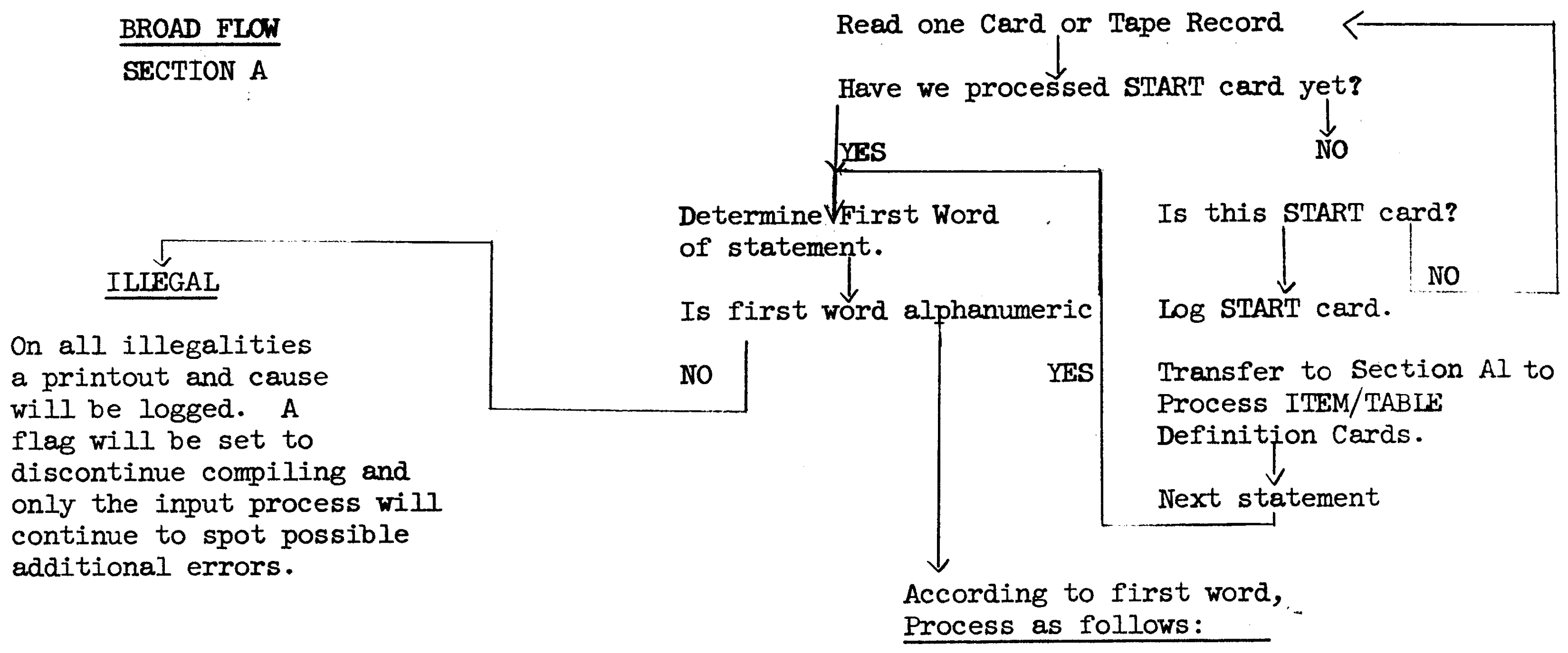
```
INTERPRETER STORAGE LIMIT

EXCEEDED ØØØØØØ
```

Insert.	Explanation
DATA	After the 1st pass in processing the VAT table (allocating tables), the data block (see TABREG) is too large for high core.
INT.	After storage is allocated for the internal table, these tables plus the data block is too large for high core.
DAISY	After 1st pass at VAT, the data block information is used together with VAT and STAT to check the limit of the Interpreter System (DAISY - see formula described above under error recognition).

BROAD FLOW

SECTION A


Read one Card or Tape Record
Have we processed START card yet?
YES
NO
Determine First Word of statement.
Is first word alphanumeric
NO
YES
ILLEGAL
On all illegalities a printout and cause will be logged. A flag will be set to discontinue compiling and only the input process will continue to spot possible additional errors.
Is this START card?
NO
Log START card.
Transfer to Section A1 to Process ITEM/TABLE Definition Cards.
Next statement
According to first word, Process as follows:

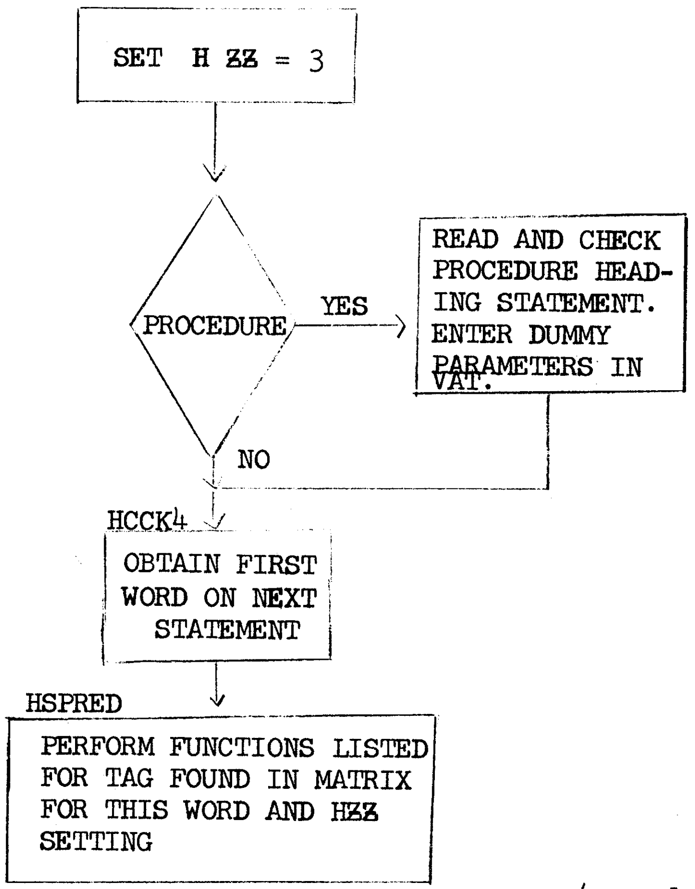
1) BEGIN, END, STATEMENT LABEL -- Process as one word and get next word as though it were first actual word of statement.

2) STOP, COMMENT, BIT/BYTE, GOTO, FOR, PROCEDURE, PROCEDURE CALL (MULTIPLE OUTPUT), SWITCH, TEST, CLOSE, RETURN, ENTRY, VALUE, TERM. -- Process each according to its particular format and variations following a rote procedure for each one.

3) IF -- Process each word encountered in the statement upon determining what it is. Convert and store all constants, subscripts and variables. Check legality positions of each word in relation to its preceding and following words in the statement. Make special legality checks also. Process Bit/Bytes and special variables encountered.

4) SET (=) -- Process left term of set statement, then process right term in similar manner as IF Statement.

FLOW DIAGRAM - SECTION A1


SET H ZZ = 3
PROCEDURE
YES
NO
READ AND CHECK PROCEDURE HEADING STATEMENT. ENTER DUMMY PARAMETERS IN VAT.
HCCK4
OBTAIN FIRST WORD ON NEXT STATEMENT
HSPRED
PERFORM FUNCTIONS LISTED FOR TAG FOUND IN MATRIX FOR THIS WORD AND HZZ SETTING

HZZ

	∅	1	2	3	4	5
ITEM	HIP	HIP	HIP	HIP	HIP	HERP1
TABLE	HERP2	HERP2	HERP2	HTP	HTP	HERP1
BEGIN	HERP2	HCP	HEXIT	HEXIT	HH12	HERP1
END	HEND	HEND	HEND	HEXIT	HEXIT	HWRTA
COMM	HCCK3	HCCK3	HCCK3	HCCK3	HCCK3	HCCK3
OTHER (ALPHANUMERIC)	HERP2	HERP2	HEXIT	HEXIT	HEXIT	HERP1
OTHER (NON-ALPHA)	HBAD	HBAD	HBAD	HBAD	HBAD	HBAD

Flow Diagram - Section A1 Cont.

HIP

1. SET HZZ = 3 If Single Item, SET HZZ = 1 If Item in Table.
2. Read and Check Remainder of Item Definition Statement.
3. Log Errors.
4. Enter Item in VAT, Status Values in STAT.
5. GOTO HCCK 4.

HTP

1. SET HZZ = 4
2. Read and Check Remainder of Table Definition Statement.
3. Log Errors.
4. Enter Table in VAT.
5. GOTO HCCK4.

HCP

1. SET HZZ = 5
2. Read and Check Remainder of Constant-list Statement.
3. Log Errors.
4. GOTO HCCK4.

HWRTA

1. SET HZZ = Ø
2. Write List of Constants on Tape.
3. Obtain Next Word in Statement.
4. GOTO HSPRED.

HERP2

1. Log Error 106.
2. SET HZZ = 3
3. GOTO HSPRED.

HERP1

1. Log Error 117
2. SET HZZ = Ø
3. GOTO HSPRED.

HH12

1. SET HZZ = 2
2. Obtain Next Word From Statement.
3. GOTO HSPRED.

Flow Diagram - Section A1 Cont.

HEXIT

1. Return to Section A.

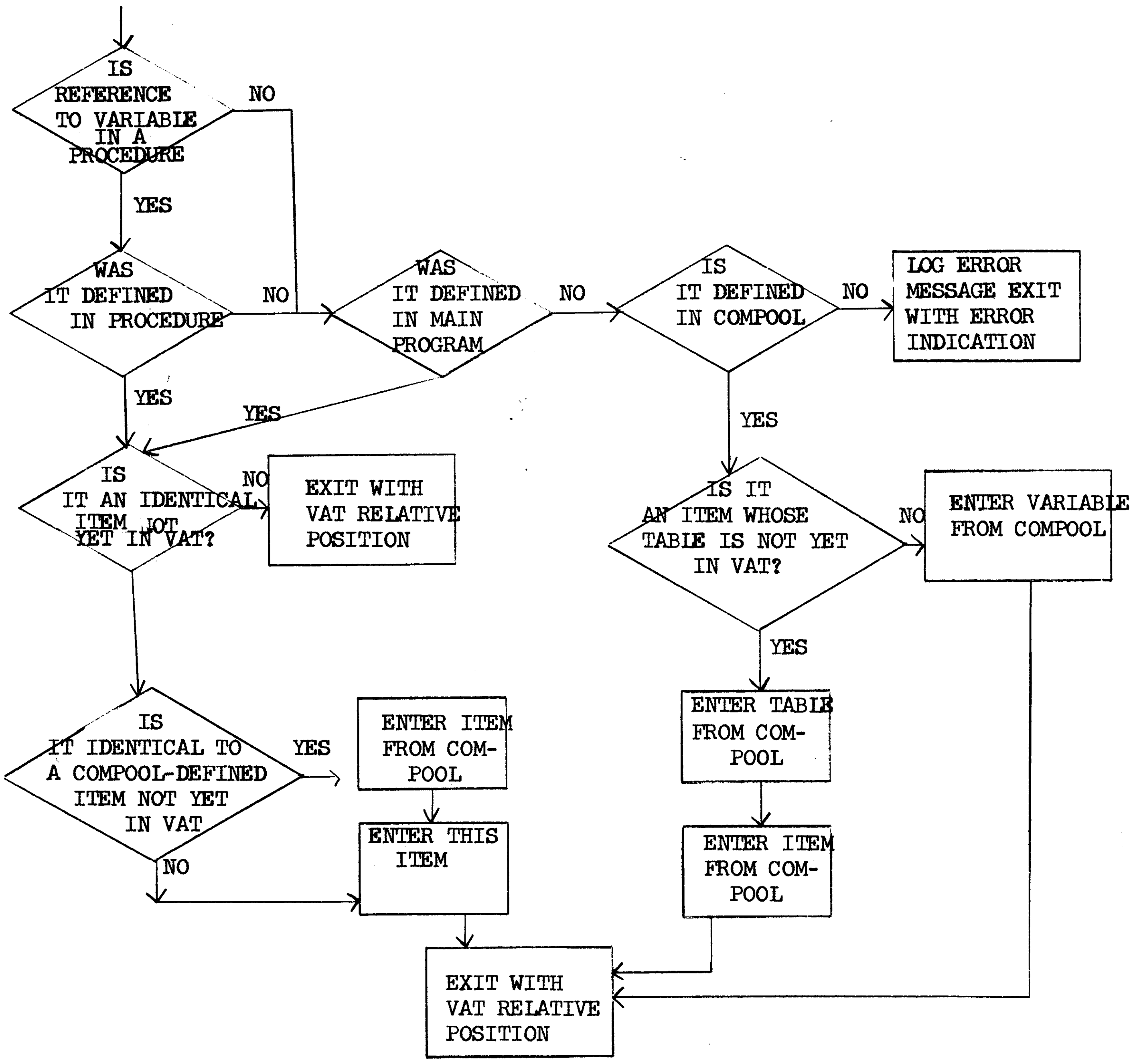
HCCK3

1. Skip to end of statement.

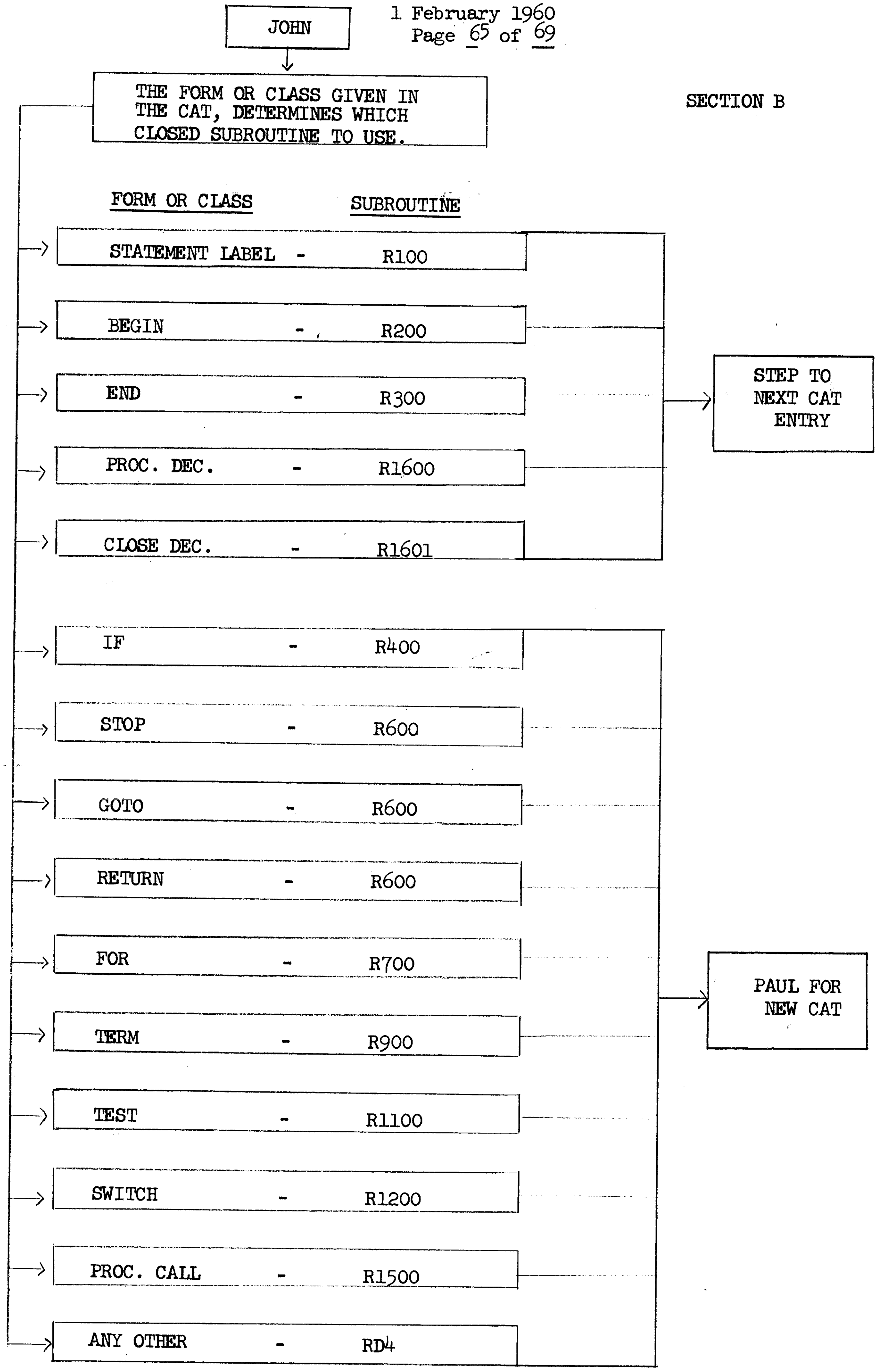
2. GOTO HCCK4.

HEND

1. SET HZZ = 3
2. Obtain Next Word in Statement.
3. GOTO HSPRED.

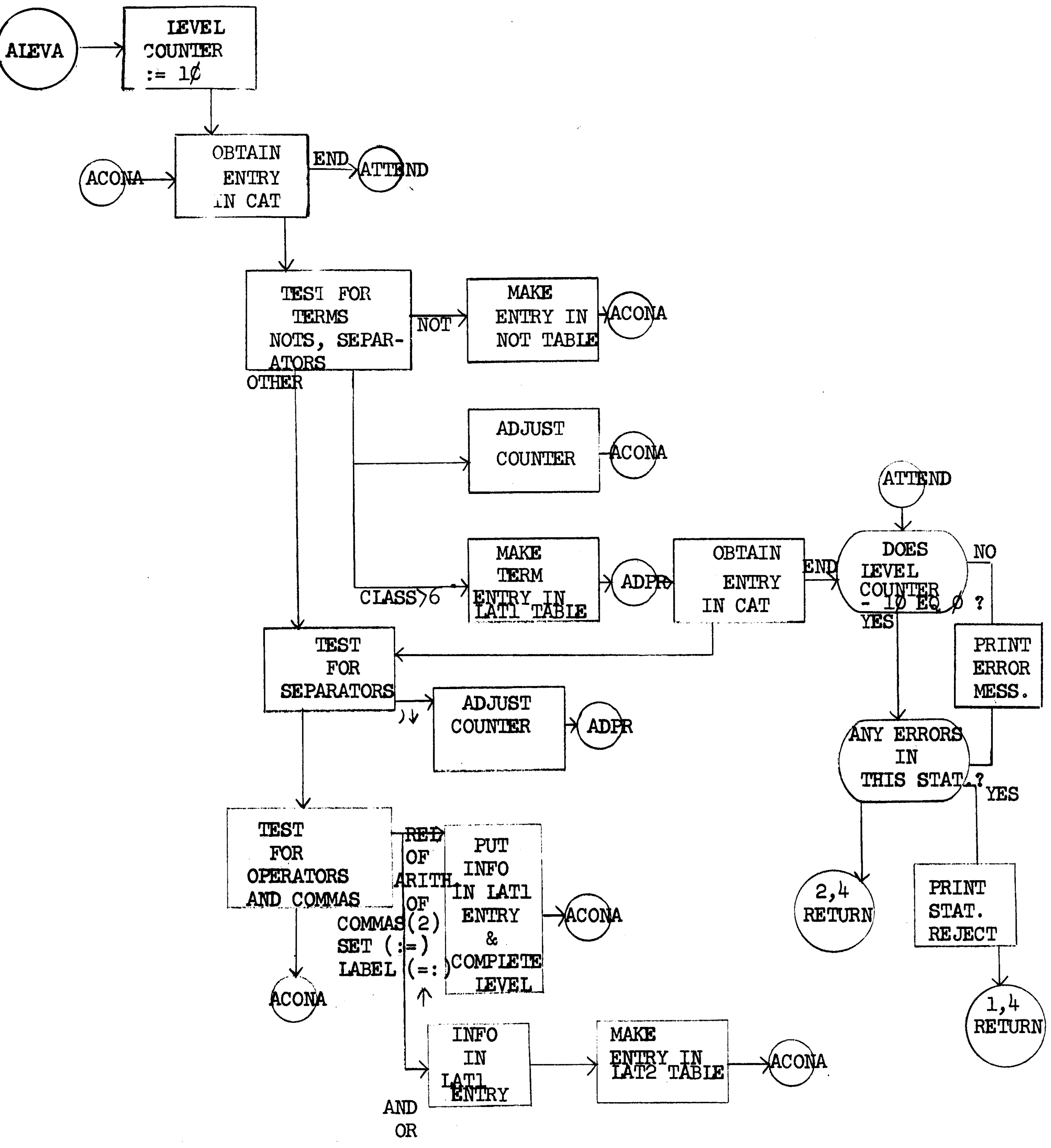

IS REFERENCE TO VARIABLE IN A PROCEDURE
NO
YES
WAS IT DEFINED IN PROCEDURE
NO
WAS IT DEFINED IN MAIN PROGRAM
NO
IS IT DEFINED IN COMPOOL
NO
LOG ERROR MESSAGE EXIT WITH ERROR INDICATION
YES
YES
YES
IS IT AN IDENTICAL ITEM NOT YET IN VAT?
NO
EXIT WITH VAT RELATIVE POSITION
IS IT AN ITEM WHOSE TABLE IS NOT YET IN VAT?
NO
ENTER VARIABLE FROM COMPOOL
YES
IS IT IDENTICAL TO A COMPOOL-DEFINED ITEM NOT YET IN VAT
YES
ENTER ITEM FROM COM-POOL
ENTER TABLE FROM COM-POOL
NO
ENTER THIS ITEM
ENTER ITEM FROM COM-POOL
EXIT WITH VAT RELATIVE POSITION

SECTION B

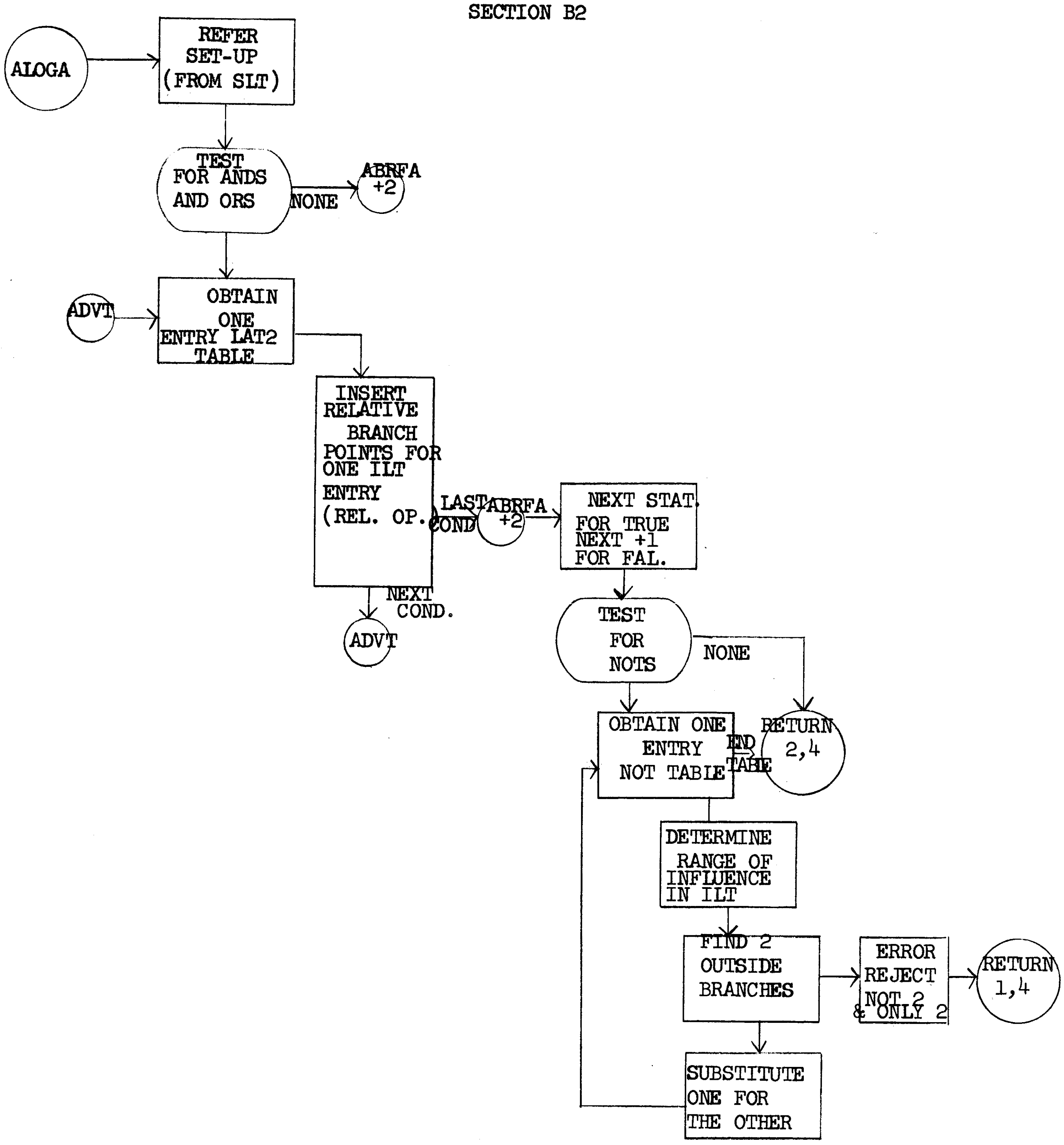

JOHN
THE FORM OR CLASS GIVEN IN THE CAT, DETERMINES WHICH CLOSED SUBROUTINE TO USE.
FORM OR CLASS
SUBROUTINE
STATEMENT LABEL - R100
BEGIN - R200
END - R300
PROC. DEC. - R1600
CLOSE DEC. - R1601
STEP TO NEXT CAT ENTRY
IF - R400
STOP - R600
GOTO - R600
RETURN - R600
FOR - R700
TERM - R900
TEST - R1100
SWITCH - R1200
PROC. CALL - R1500
ANY OTHER - RD4
PAUL FOR NEW CAT

BLOCK FLOW

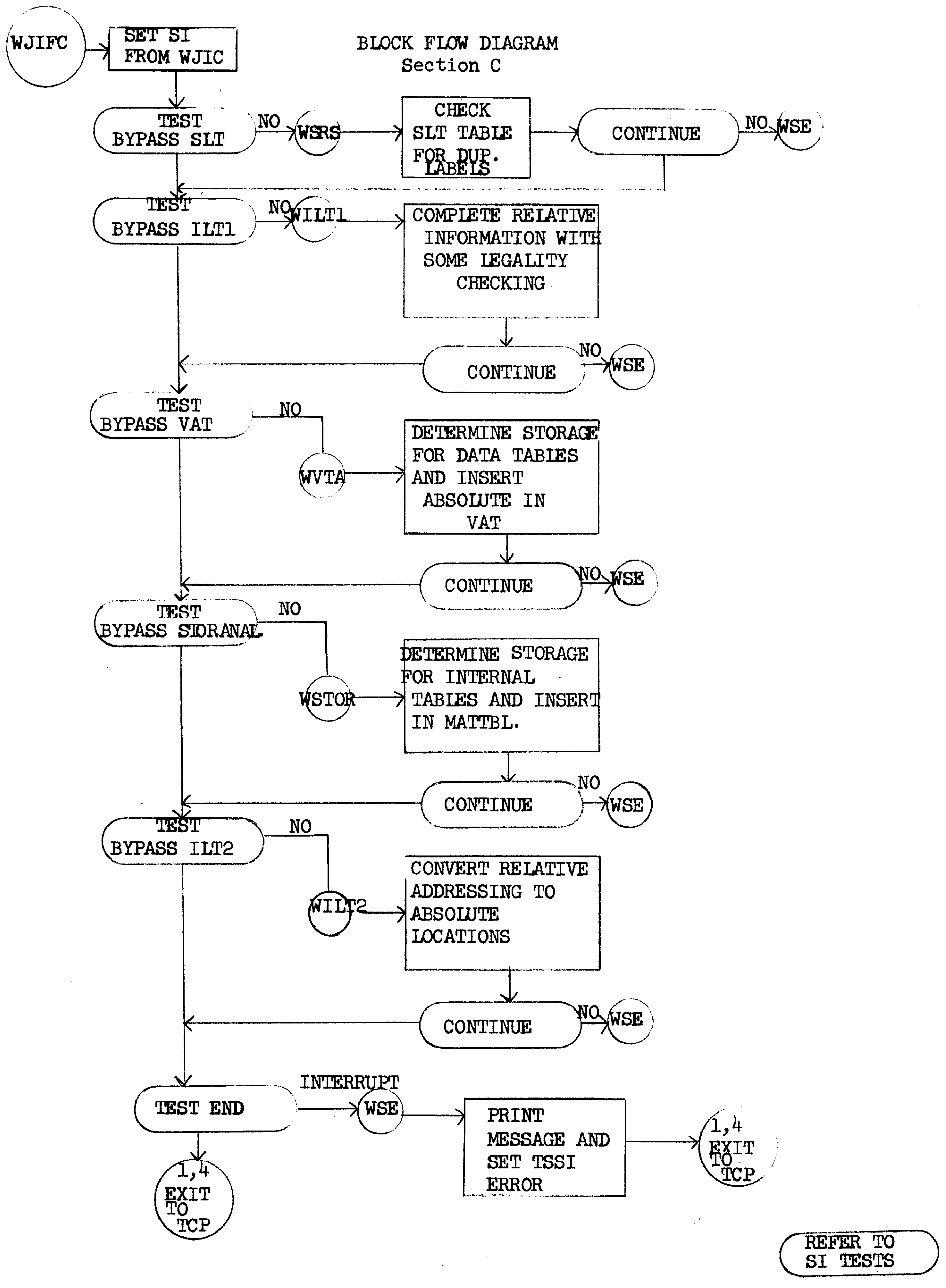
SECTION B1



BLOCK FLOW
SECTION B2


ALOGA
REFER SET-UP (FROM SLT)
TEST FOR ANDS AND ORS
NONE
ABRFA +2
ADVT
OBTAIN ONE ENTRY LAT2 TABLE
INSERT RELATIVE BRANCH POINTS FOR ONE ILT ENTRY (REL. OP.)
LAST COND
ABRFA +2
NEXT STAT. FOR TRUE NEXT +1 FOR FAL.
NEXT COND.
ADVT
TEST FOR NOTS
NONE
OBTAIN ONE ENTRY NOT TABLE
END TABLE
RETURN 2,4
DETERMINE RANGE OF INFLUENCE IN ILT
FIND 2 OUTSIDE BRANCHES
ERROR REJECT NOT 2 & ONLY 2
RETURN 1,4
SUBSTITUTE ONE FOR THE OTHER

BLOCK FLOW DIAGRAM
Section C


WJIFC
SET SI FROM WJIC
TEST BYPASS SLT
NO
WSRS
CHECK SLT TABLE FOR DUP. LABELS
CONTINUE
NO
WSE
TEST BYPASS ILT1
NO
WILT1
COMPLETE RELATIVE INFORMATION WITH SOME LEGALITY CHECKING
CONTINUE
NO
WSE
TEST BYPASS VAT
NO
WVTA
DETERMINE STORAGE FOR DATA TABLES AND INSERT ABSOLUTE IN VAT
CONTINUE
NO
WSE
TEST BYPASS STORANAL
NO
WSTOR
DETERMINE STORAGE FOR INTERNAL TABLES AND INSERT IN MATTBL.
CONTINUE
NO
WSE
TEST BYPASS ILT2
NO
WILT2
CONVERT RELATIVE ADDRESSING TO ABSOLUTE LOCATIONS
CONTINUE
NO
WSE
TEST END
INTERRUPT
WSE
PRINT MESSAGE AND SET TSSI ERROR
1,4 EXIT TO TCP
1,4 EXIT TO TCP
REFER TO SI TESTS

A listing of the program symbolic deck will be issued as the first supplement to this document (FN-LO-201,S1).

Distribution:

SDC (Lodi)

Division Staff (1 ea.)
Programming Branch Staff (1 ea.)
Program Production Group (1 ea.)
Program Design Group - M. Mineart (20)
Program Requirements Group - F. Diaz (5)
CUSS Project - J. I. Schwartz (10)

SDC (Santa Monica)

J. D. Madden
R. Bosak
J. Matousek
B. Morriss
G. Dobbs (10)
E. Gordon
C. M. Lawson
D. E. Henley
G. Jacobs

IEC

Standard Distribution (35)

:lb