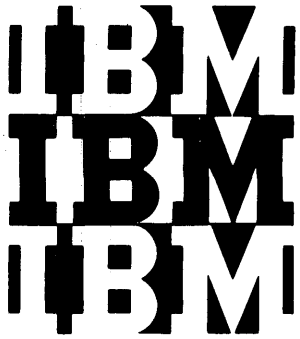


File No. S360-29
Form Y33-7000-0

Program Logic

# IBM System/360 Conversion Aids:
# FORTRAN IV-to-PL/I Language Conversion Program for IBM System/360 Operating System

Program Number 360C-CV-710

This document describes the internal logic of the FORTRAN IV-to-PL/I (F) Language Conversion Program for the IBM System/360 Operating System.

Program logic manuals are intended for use by IBM system engineers involved in program maintenance, and by system programmers involved in altering the program design. Program logic information is not necessary for program operation and use; therefore, distribution of this manual is limited to persons with program maintenance or modification responsibilities.

Restricted Distribution

# PREFACE

This document describes the structure and functions of the FORTRAN IV-to-PL/I Language Conversion Program: its components, their functions, and the control flow among them. The detailed organization of each component and the instructions used to implement its functions are described in the program listing. This manual is intended to help the user find any portion of the listing he requires.

The manual consists of eight sections. Section 1 is an introduction to the Language Conversion Program. Section 2 describes the Control Phase of the program; sections 3, 4, 5, and 6 describe the four logical phases into which the program is divided.

The reader should have some familiarity with the contents of the following publications:

- IBM System/360 FORTRAN IV Language, Form C28-6515
- IBM System/360 Operating System, PL/I (F) Programmer's Guide, Form C28-6594
- IBM System/360 Conversion Aids: FORTRAN IV-to-PL/I Language Conversion Program for IBM System/360 Operating System, Form C33-2002
- IBM System/360 PL/I Reference Manual Form C28-8201

In addition, the reader would find it useful to be acquainted with the publications:

- IBM System/360 Operating System, Job Control Language, Form C28-6539
- IBM System/360 Operating System, PL/I Language Specifications, Form Y33-6003.

First Edition (January, 1969)

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

Request for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM France, Centre d'Etudes et Recherches, Programming Publications, Department 843, 06 - La Gaude, France.

© Copyright International Business Machines Corporation 1969

INTRODUCTION . . . . . 7
Tables . . . . . 7
Overall Logic of the LCP . . . . . 7

CONTROL phase . . . . . 10

INITIALIZATION PHASE . . . . . 12
Processing Routines . . . . . 12
IPPRO . . . . . 12
GETCRD . . . . . 12
IMPRO . . . . . 13

PHASE 10 . . . . . 21
Specification Statement Processing Routines . . . . . 21
BLPRO . . . . . 22
CMPRO . . . . . 22
DAPRO . . . . . 22
DMPRO . . . . . 22
EDPRO . . . . . 22
EQPRO . . . . . 22
EXPRO . . . . . 23
FNPRO . . . . . 23
NLPRO . . . . . 23
YIPRO . . . . . 23
Executable Statement Processing Routines . . . . . 23
ALPRO . . . . . 24
ASPRO . . . . . 24
CAPRO . . . . . 24
COPRO . . . . . 24
CTPRO . . . . . 25
DOPRO . . . . . 25
FTPRO . . . . . 25
GTPRO . . . . . 26
IFPRO . . . . . 26
IOPRO . . . . . 27
PSPRO . . . . . 28
RTPRO . . . . . 28
STPRO . . . . . 28
Utility Routines . . . . . 28
ARPRO . . . . . 28
ENTTDI . . . . . 30
ERMS . . . . . 30
GETTBl . . . . . 31
GETWRD . . . . . 31
LTCOL . . . . . 31
PUTTBL . . . . . 31
SLPAR . . . . . 31
SPDIM . . . . . 32
SPDTA . . . . . 32
SPPRO . . . . . 32
SPSUB . . . . . 32
XTBPRO . . . . . 32

PHASE 20 . . . . . 74
Phase 20 Routines . . . . . 74
DTPRO . . . . . 74
EVPRO . . . . . 74
EVPRO . . . . . 74

PHASE 30 . . . . . 78
Phase 30 Routines . . . . . 78
UPRNT . . . . . 78

APPENDIX A. TABLES . . . . . 82
General Considerations . . . . . 82
The Dictionary (Table TDI) . . . . . 82
Specification Tables . . . . . 83
Table TBK (Common Block Table) . . . 83
Table TCM (Common Variable Table) . 83
Table TDA (Table of Variables with Initial Values) . . . . . 83
Table TDM (Dimension Table) . . . . 83
Table TDU (Dummy Variable Table) . . 83
Table TDT (Table of Initial Values) 84
Table TEH (Table of EQUBLK's) . . . 84
Table TEQ (Equivalence Table) . . . 84
Table TFM (Format table) . . . . . 85
Table TIM (Implicit Statement Table) . . . . . 85
Table TNL (NAMELIST Table) . . . . . 85
Table TNV (List of Variable Table) . 85
Table TOV (Overflow Table) . . . . . 85
Table TPD (DO Loop Label Table) . . 86

APPENDIX B. FORTRAN TO PL/I SUBROUTINE LIBRARY . . . . . 87

APPENDIX C. STORAGE MAP . . . . . 88

INDEX . . . . . 89

Chart 001. Overall Logic of Control Phase . . . . . 11
Chart 002. IPPRO . . . . . 14
Chart 003. GETCRD (Part 1) . . . . . 15
Chart 004. GETCRD (Part 2) . . . . . 16
Chart 005. GETCRD (Part 3) . . . . . 17
Chart 006. GETCRD (Part 4) . . . . . 18
Chart 007. GETCRD (Part 5) . . . . . 19
Chart 008. IMPRO Routine . . . . . 20
Chart 009. Overall Logic of Phase 10 . . . . . 33
Chart 010. BLPRO Routine . . . . . 34
Chart 011. CMPRO Routine . . . . . 35
Chart 012. DAPRO Routine . . . . . 36
Chart 013. DMPRO Routine . . . . . 37
Chart 014. EDPRO Routine . . . . . 38
Chart 015. EQPRO Routine . . . . . 39
Chart 016. EXPRO Routine . . . . . 40
Chart 017. FNPRO Routine . . . . . 41
Chart 018. NLPRO Routine . . . . . 42
Chart 019. YIPRO Routine . . . . . 43
Chart 020. ALPRO Routine . . . . . 44
Chart 021. ASPRO Routine . . . . . 45
Chart 022. CAPRO Routine . . . . . 46
Chart 023. COPRO Routine . . . . . 47
Chart 024. CTPRO Routine . . . . . 48
Chart 025. DOPRO Routine . . . . . 49
Chart 026. FTPRO Routine . . . . . 50
Chart 027. GTPRO Routine . . . . . 51
Chart 028. IFPRO Routine . . . . . 52
Chart 029. IOPRO Routine, Part 1 of 2 . . . . . 53
Chart 030. IOPRO Routine, Part 2 of 2 . . . . . 54
Chart 031. PSPRO Routine . . . . . 55
Chart 032. RTPRO Routine . . . . . 56
Chart 033. STPRO Routine . . . . . 57
Chart 034. ARPRO Routine, Part 1 of 4 . . . . . 58
Chart 035. ARPRO Routine, Part 2 of 4 . . . . . 59
Chart 036. ARPRO Routine 3 of 4 . . 60
Chart 037. ARPRO Routine, Part 4 of 4 . . . . . 61
Chart 038. ENTTDI Routine . . . . . 62
Chart 039. ERMS Routine . . . . . 63
Chart 040. GETTBL Routine . . . . . 64
Chart 041. GETWRD Routine . . . . . 65
Chart 042. LTCOL Routine . . . . . 66
Chart 043. PUTTBL Routine . . . . . 67
Chart 044. SLPAR Routine . . . . . 68
Chart 045. SPDIM Routine . . . . . 69
Chart 046. SPDTA Routine . . . . . 70
Chart 047. SPPRO Routine . . . . . 71
Chart 048. SPSUB Routine . . . . . 72
Chart 049. XTBPRO Routine . . . . . 73
Chart 050. Overall Logic of Phase 20 . . . . . 75
Chart 051. DTPRO Routine . . . . . 76
Chart 052. EVPRO Routine . . . . . 77
Chart 053. Overall Logic of Phase 30, Part 1 of 2 . . . . . 79
Chart 054. Overall Logic of Phase 30, Part 2 of 2 . . . . . 80
Chart 055. UPRNT Routine . . . . . 81

## FIGURES

Figure 1. Overall Logic of the LCP . . . . . 9
Figure 2. Tables TFM, TDT, TDU . . 27
Figure 3. ARPRO Processing . . . . 30

The FORTRAN IV-to-PL/I Language Conversion Program, henceforth referred to in this publication as "the LCP," is intended to facilitate transition to PL/I by converting "error-free" System/360 Operating System FORTRAN IV source programs into PL/I (F) programs. (For the purposes of this publication, the converted PL/I programs are referred to as "target" programs.) The LCP is itself expressed principally in PL/I (F) language, and can be compiled, using a System/360 Operating System PL/I (F) compiler, to give a System/360 machine code version. Note that certain routines are expressed in System/360 Operating System Assembly Language and are compiled using a System/360 Operating System (F) Assembler. The latter can then be used to convert into PL/I the FORTRAN programs written by the user.

An "error-free" source program is one that meets the following requirements:

1. it consists of statements that are in strict conformity with the syntactic rules laid down in the publication IBM System/360 FORTRAN IV Language, Form C28-6515

2. It has been successfully compiled by the user's FORTRAN IV compiler, giving a System/360 machine code version that produces results in accordance with the user's intentions.

FORTRAN statements that do not conform to these rules of syntax (note 2 above) are not converted, and a message to that effect is printed in the output listing. A message is also issued when the conversion of a source statement is impossible for technical reasons.

Note that, the LCP will convert FORTRAN IV source programs written for current IBM systems other than the System/360. In this connection, however, the user should note that:

- From the point of view of language, all FORTRAN IV source programs can be converted by the LCP, subject to the restrictions indicated in Appendix C of IBM System/360 Conversion Aids: FORTRAN IV-to-PL/I Language Conversion Program for IBM System/360 Operating System, Form C33-2002.

- There is no guarantee that the converted programs will be correctly executed. This is due to differences in implementation (storage allocation, magnitude of data, etc.).

The LCP is divided into a Control Phase and the following four logical phases:

1. Initialization phase, which scans the FORTRAN source program and classifies the FORTRAN statements.

2. Phase 10, which converts executable FORTRAN statements and builds various tables from the specification statement.

3. Phase 20, (optional) which rearranges certain tables created during Phase 10.

4. Phase 30, which generates the PL/I declarations for the identifiers used in the program, together with all comments, statements and messages.

It should be noted that, as far as possible:

- Executable statements are converted one at a time.

- FORTRAN specification statements are converted by collecting declarations of source program identifiers.

## TABLES

The LCP contains a number of tables. The way in which these tables are used by the individual routines is described in the relevant sections; the layout of the tables is shown in Appendix A.

## OVERALL LOGIC OF THE LCP

Figure 1 illustrates the overall internal logic of the LCP and shows the relationship among the Control Phase and the 4 logical phases: Initialization Procedure, Phase 10, Phase 20, and Phase 30.

After receiving control from the System/360 Operating System, the Control Phase calls the Initialization Procedure, whose input is the FORTRAN source program on SYSIN. This phase examines the source program to determine whether the FORTRAN

statements are executable or not, initializes various tables, and, as output, places FORTRAN statements on SYSUT1. Optionally, a listing of the source program can be printed on SYSPRNT. Control returns to the Control Phase which calls Phase 10.

Phase 10 converts, if possible, the executable FORTRAN statements and comments. Certain parts of these statements are placed in the tables with the contents of the specification statements. If the size of the tables exceeds the amount of main storage available, the tables are placed on SYSUT2.

Control again returns to the Control Phase, which either calls Phase 20 (under conditions described in the section 'Phase 20'), or calls Phase 30 directly.

Phase 20, if called, rearranges EQUIVALENCE and DATA tables and returns control to the Control Phase, which then calls Phase 30.

Phase 30 takes the converted statements (on SYSUT1) and the tables (which may be on SYSUT2) as its input. It generates the PL/I declarations for the identifiers used in the program and edits the converted statements, together with all comments and messages; (its output is in the form specified by the user).

On completion of Phase 30, control returns to the Return Procedure of the Control Phase which either calls the Initialization Procedure if another FORTRAN program must be converted, or returns to the Operating System.

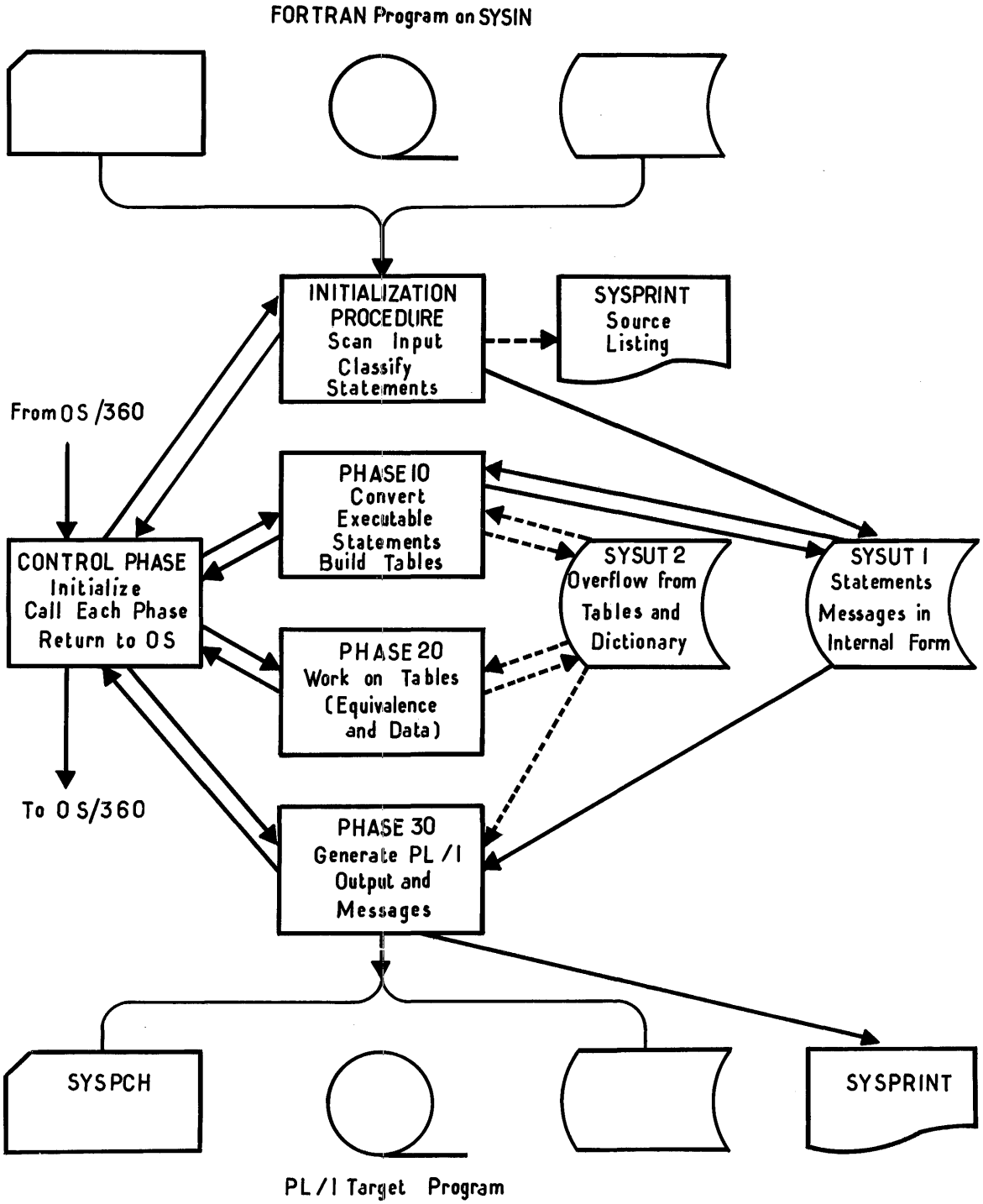



Figure 1. Overall Logic of the LCP

## CONTROL PHASE

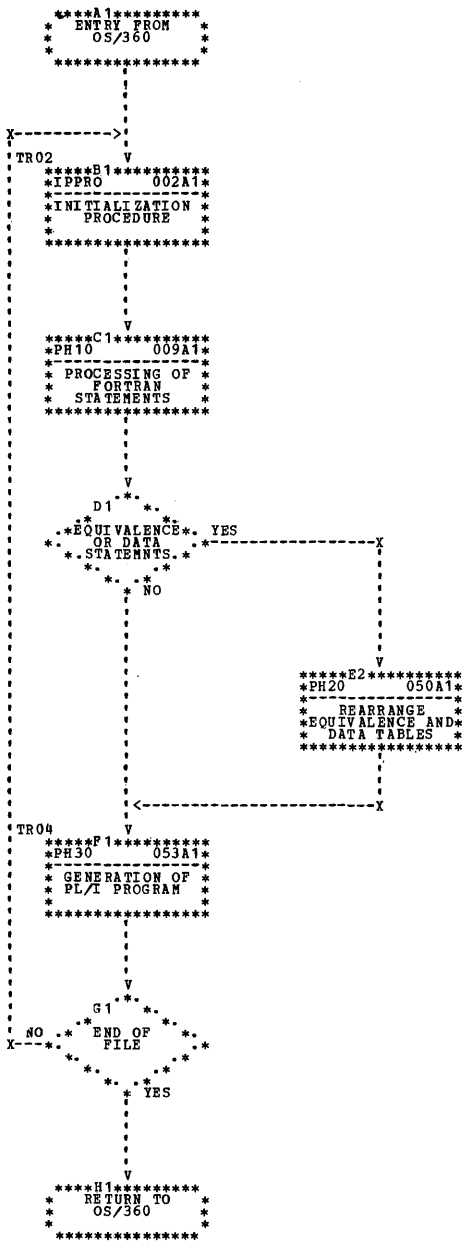
The LCP Control Phase (see Chart 001) receives control from the System/360 Operating System by means of an EXEC control card.

The Control Phase invokes the execution of the Initialization Procedure which is the first logical phase of the LCP.

On completion of each logical phase of the program, the following phase is called.

If, on completion of Phase 30, another FORTRAN program is next in the job stream, the LCP retains control and the Control Phase recalls the Initialization Procedure. After the last consecutive FORTRAN program in the job stream has been processed, the Control Phase restores control to the Operating System.

Chart 001. Overall Logic of Control Phase



## INITIALIZATION PHASE

The Initialization phase is divided in three processing routines:

1. IPPRO, which initializes the settings of the switches in the LCP communication area.
2. GETCRD, which scans the source program and classifies the FORTRAN statement by type.
3. IMPRO, which processes the IMPLICIT specification statement.

The Initialization phase also makes use of the utility routines GETWRD, GETTBL, and PUTTBL. These routines are also used by Phase 10 and are described in the next chapter.

The input to the Initialization phase is the FORTRAN source program on SYSIN. The output of this phase is the FORTRAN source statements on SYSUT1.

If a FORTRAN source statement cannot be classified or if a syntactical error is detected during the scan, the statement is treated as a comment.

### PROCESSING ROUTINES

This section contains detailed descriptions of the routines IPPRO, GETCRD and IMPRO. They are described in the following manner:

- Purpose of the routine
- Calling phase or routine (s)
- Entries (if more than one)
- Processing
- Routine (s) called
- Exit from the routine

#### IPPRO ---------- Chart 002

Purpose: To initialize the status of the switches in the LCP communication area, according to the options specified by the user in his control cards.

Called by: Control Phase

Processing: For the first FORTRAN program in a given batch, the contents of the PARM field in the EXEC control card indicates the status of the various switches. These are not modified when IPPRO is recalled to process subsequent FORTRAN programs in this batch.

The LCP control cards, if any, are scanned:

- any numeric field of two digits (maximum) is considered as a data-set reference number, and is marked as a PRINT file. This information is kept for use in the other phases.
- any FORTRAN mathematical function name is considered as a reserved word; this name is not converted into its equivalent built-in function name.

The original LCP control information is kept during processing of a single batch unless the user specifies new LCP control cards for a subsequent FORTRAN program in the batch.

For further information concerning the status of the switches, refer to the publication FORTRAN IV-to-PL/I Language Conversion Program for IBM System/360 Operating System, Form C33-2002, Appendix E, "Control Card Options." (Note that, for the sake of brevity, this publication will henceforth be referred to as "the language conversion manual.")

Routines called: None.

Exit: Go to GETCRD

#### GETCRD ---------- Charts 003, 004, 005, 006, 007

Purpose: To get a FORTRAN statement and classify it according to type.

Called by: IPPRO

Processing: The input to this routine consists of the FORTRAN source statements from SYSIN. Each FORTRAN statement is collected into a work area and two scans are performed.

1. Classification scan: the statement is examined one character at a time from

left to right. It is classified as non-arithmetic if:

- The following character sequence indicating the presence of an literal field is found:

  ,nH or /nH or (nH

  where n is a numerical value. Switches CSNU for n and CSHL for H are used.

- The sequence A(.....)x where x is not an equal sign, is encountered. Switch CSNA is used.

- There is a comma outside the parentheses.

- There is an equal sign inside the parentheses.

In the absence of these conditions, the statement is classified as arithmetic.

2. Scan of Keyword Dictionary (KEYDIC): the first four characters of each statement classified as non-arithmetic are compared to a list of FORTRAN keywords. This comparison determines the type of FORTRAN statement. Each source statement is given a statement code from 1 through 42 according to its type.

If an IMPLICIT statement is detected, the routine IMPRO is invoked.

When a FORTRAN statement is recognized as a READ, WRITE, PRINT, or PUNCH statement, the following additional processing takes place:

- The data set and FORMAT reference numbers are collected, if present.
- The LCP creates a table (TFM) containing FORMAT reference numbers, and corresponding data set attributes (PRINT, non-PRINT or both).

In addition, GETCRD builds table TPD and uses it to generate an END statement at the end of each DO loop. Note that a logical IF statement is treated as two separate statements.

The output from GETCRD is the FORTRAN source program on SYSUT1; FORTRAN statements with the same statement code are chained together. When an END card is encountered, this routine returns to the Control Phase.

Routines called: GETWRD, GETTBL, IMPRO, PUTTBL.

Exit: Control Phase

IMPRO Chart 008

Purpose: To convert IMPLICIT statements.

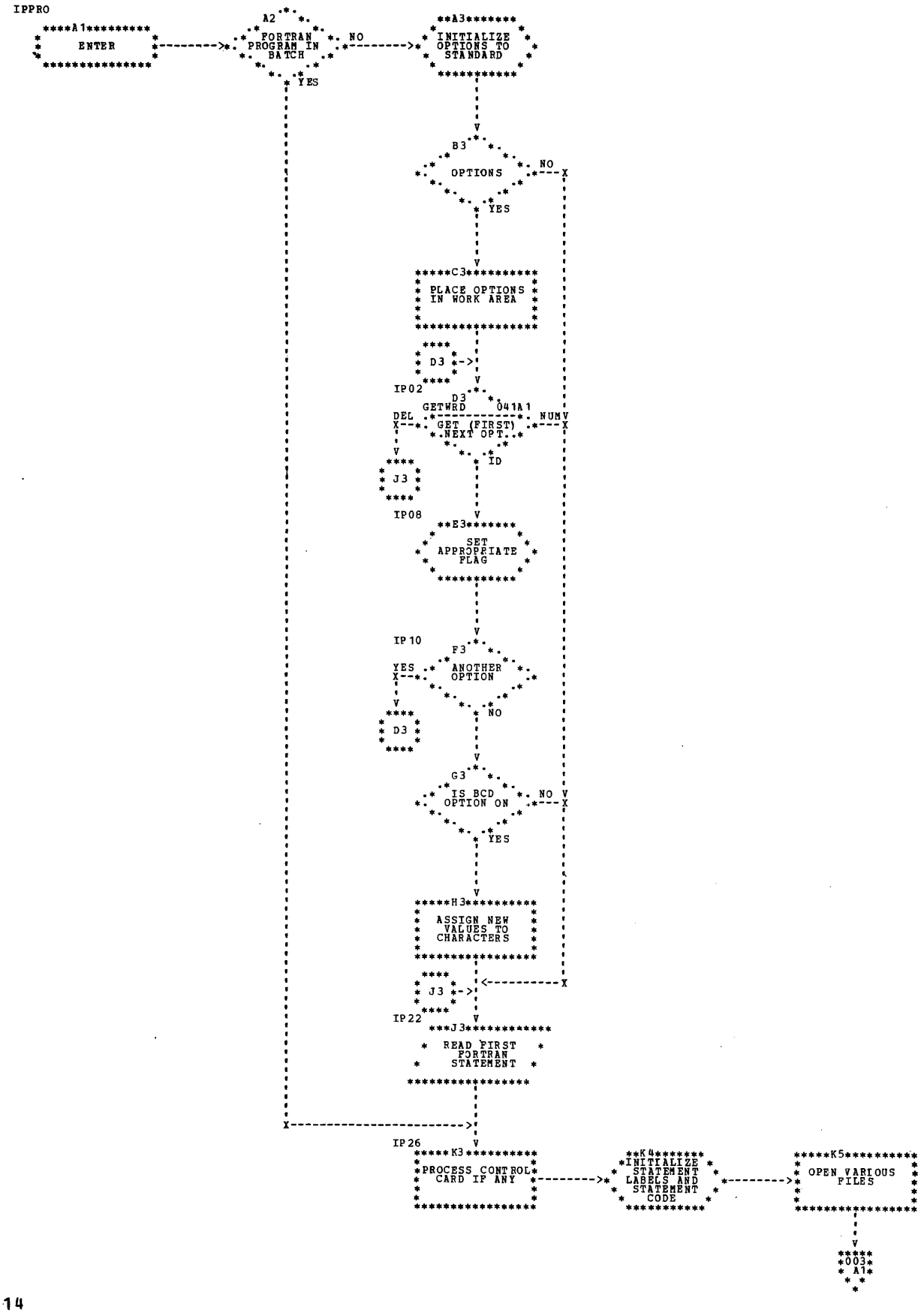
Called by: GETCRD.

Processing: The input to this routine is the IMPLICIT statement in the work area WKBF. The two alphabetic characters used as limits of an implicit type declaration are saved. The original entry in the implicit table (TIM), which contains the standard type of any name beginning with an alphabetic character, is updated to conform with the type given in the IMPLICIT statement. Any modification in length is also entered in this table.

Routines called: GETWRD.

Exit: GETCRD

## Chart 002. IPPRO



## Chart 003. GETCRD (Part 1)

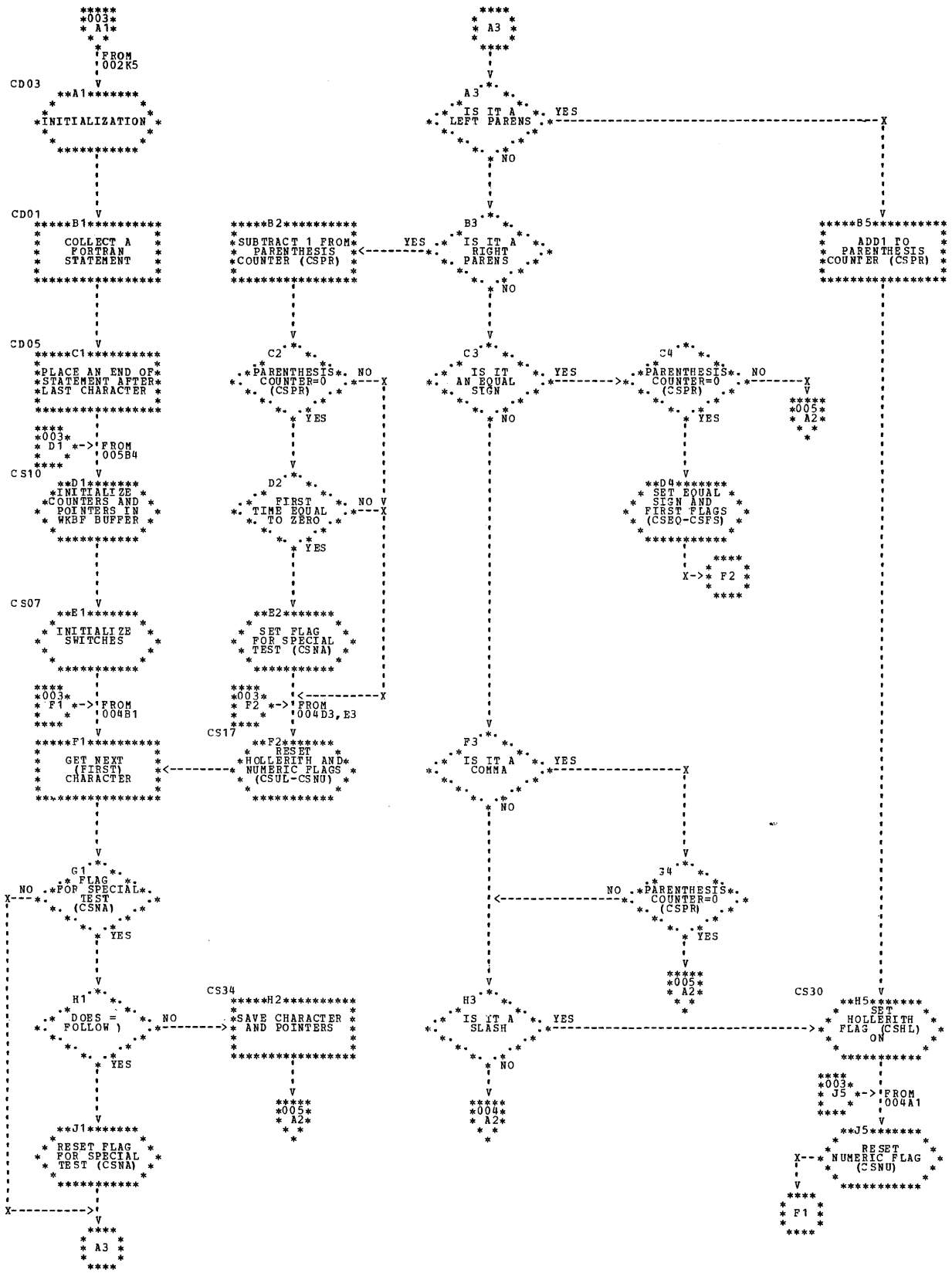


Chart 004. GETCRD (Part 2)

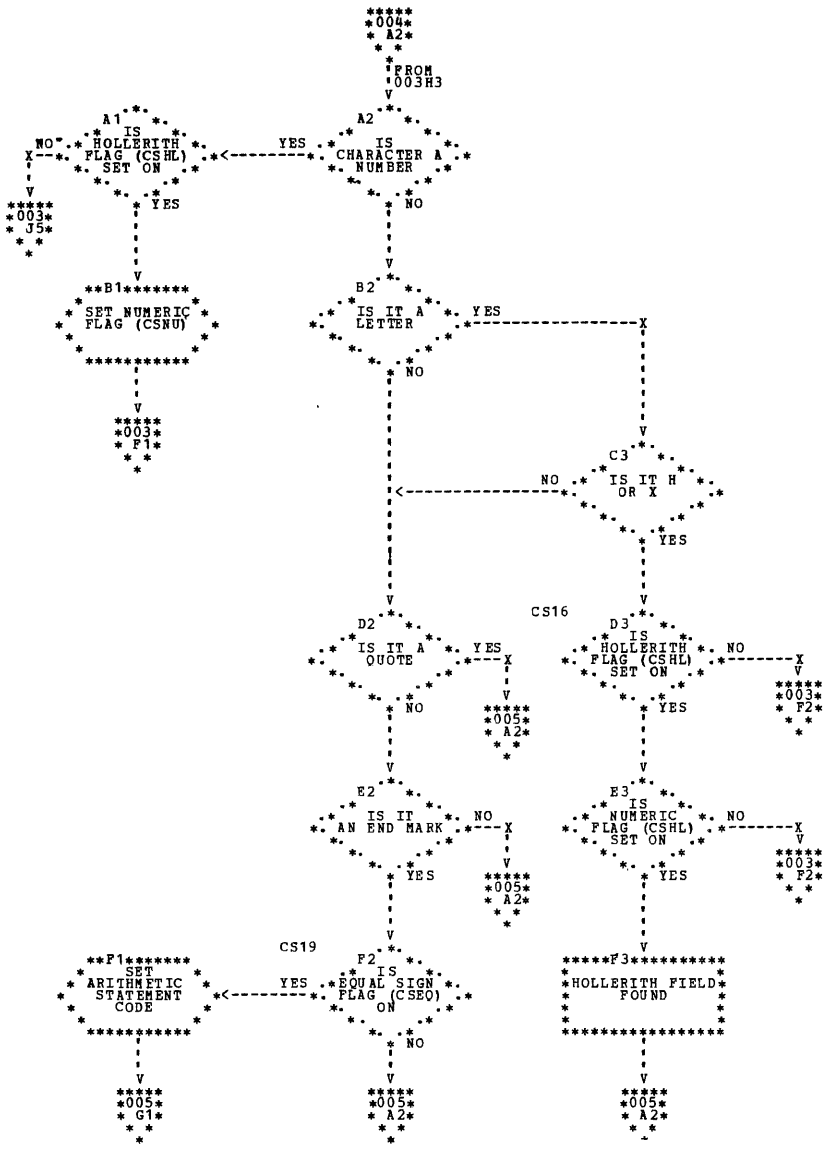


Chart 005. GETCRD (Part 3)

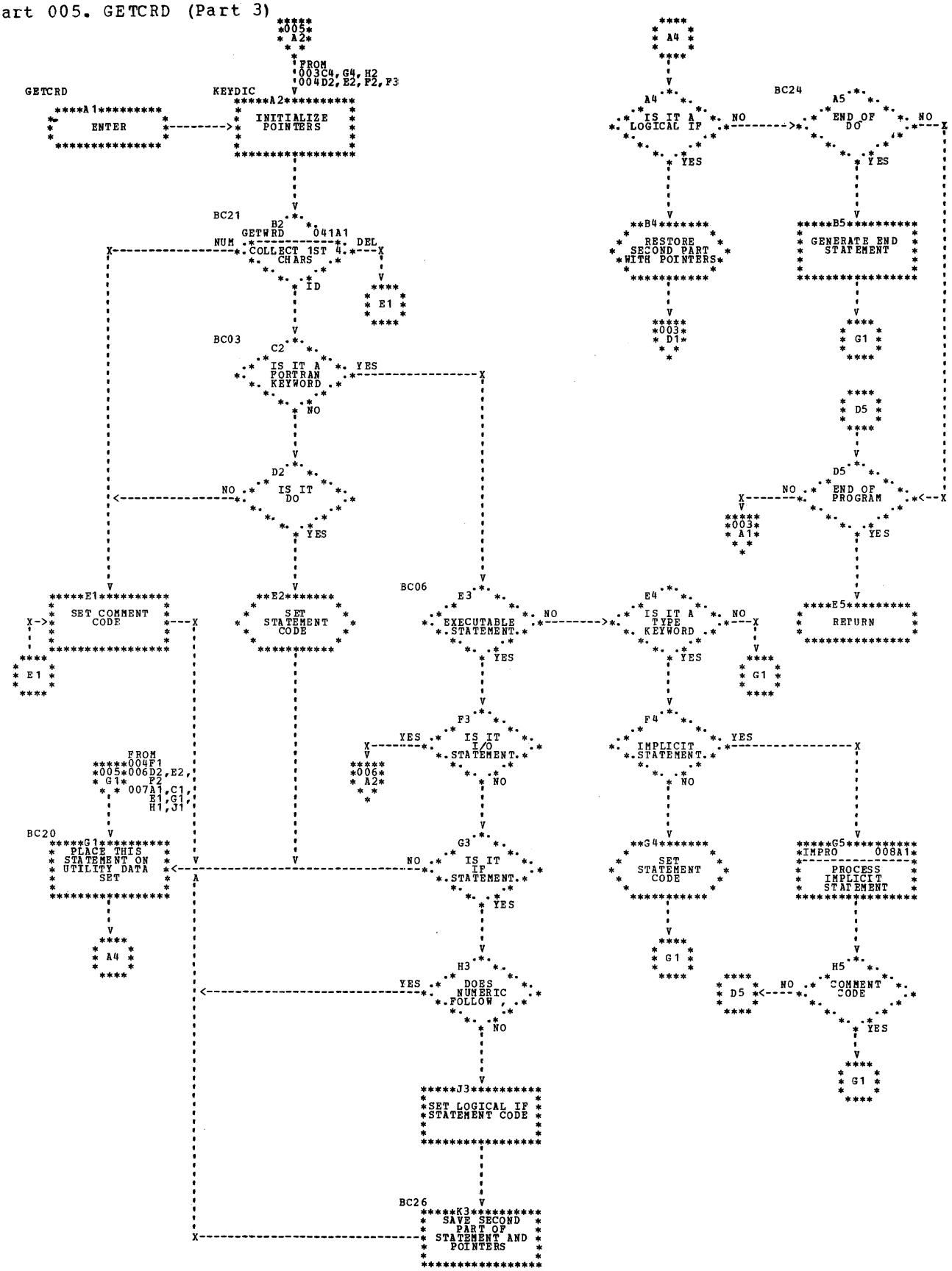


Chart 006. GETCRD (Part 4)

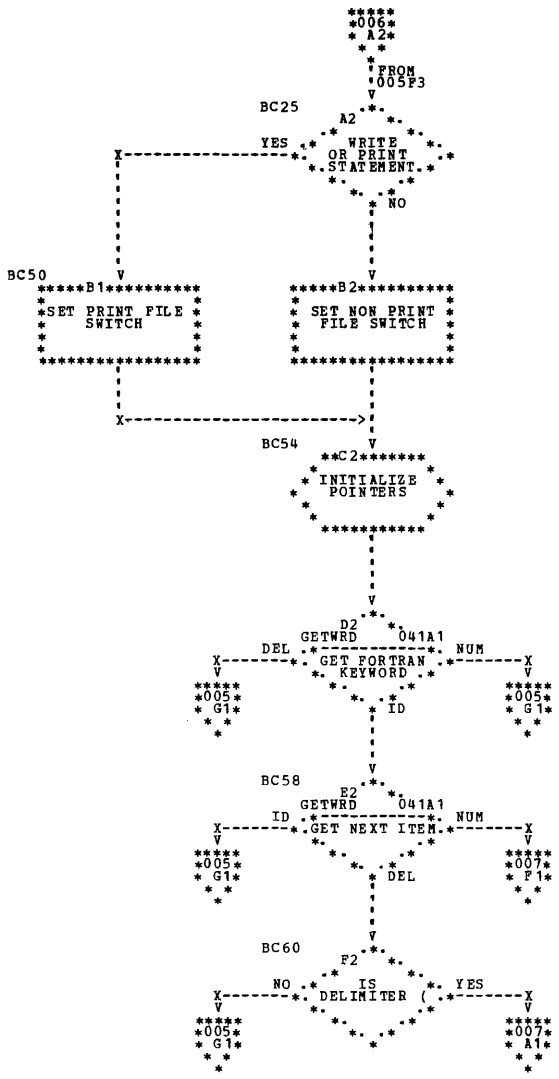
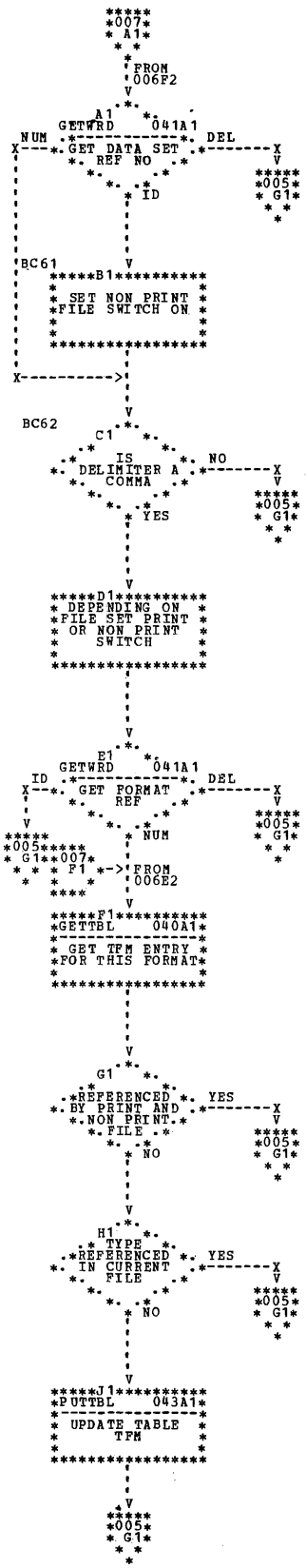
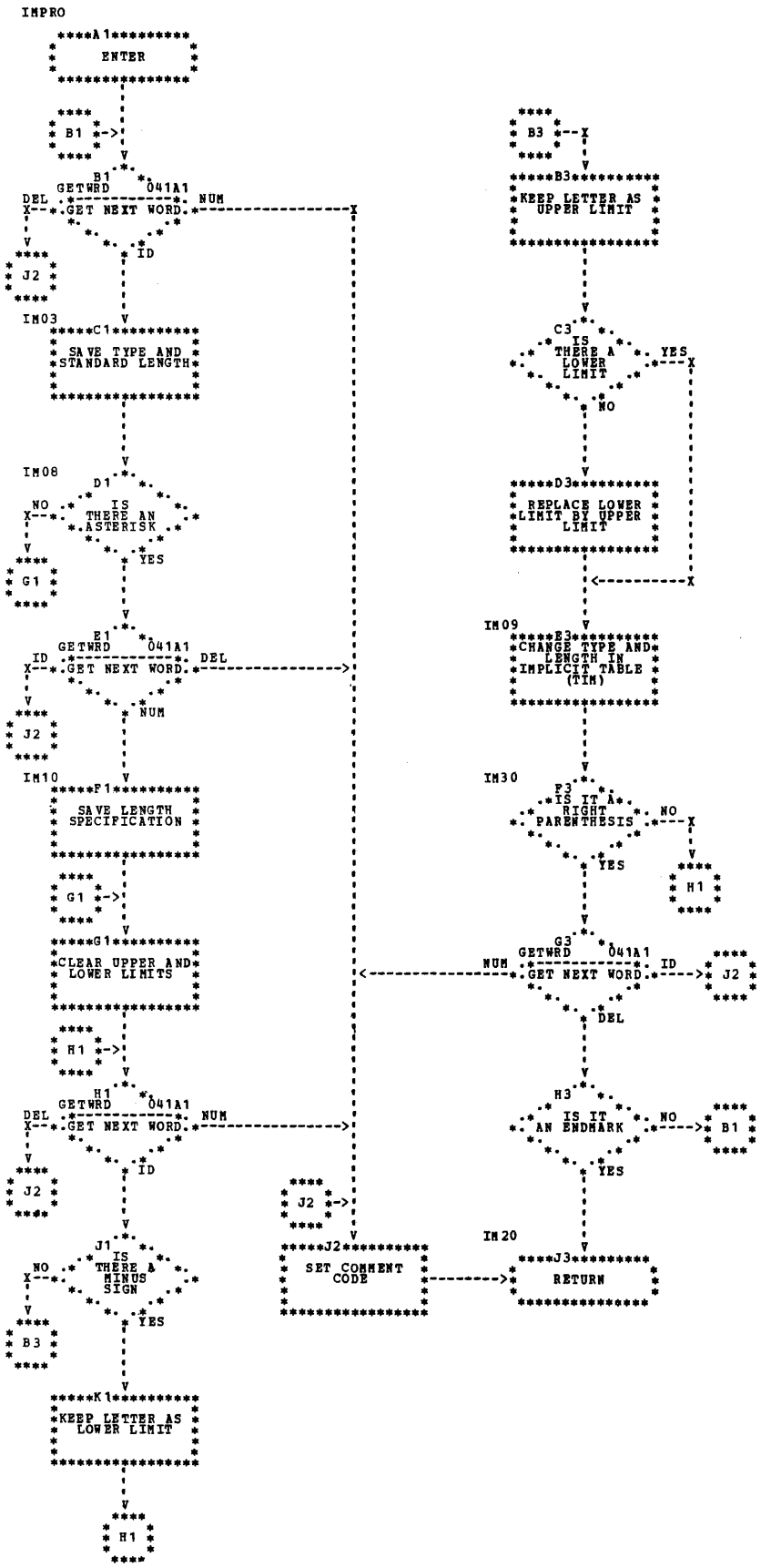


Chart 007. GETCRD (Part 5)



## Chart 008. IMPRO Routine



Phase 10 (see Chart 009) converts executable FORTRAN statements and builds tables from the specification statements. Certain information collected during the scan is saved in tables to be processed subsquently by Phase 20 and/or Phase 30.

The input to Phase 10 is the FORTRAN statements placed on SYSUT1 by the Initialization Procedure.

Phase 10 passes control to the appropriate statement processing routine, which processes a chain of statements at a time by attempting to convert as much of the statements as possible into the corresponding PL/I statements. These routines fall into one of two categories:

1. Specification statement processing routines, which enter in the dictionary (TDI) any identifier that appears within a specification statement. The dictionary includes specification information and pointers to tables containing additional information for each entry. The routines included in this category are the following:

| | | |
|---|---|---|
| BLPRO | EDPRO | |
| CMPRO | EQPRO | NLPRO |
| DAPRO | EXPRO | YIPRO |
| DMPRO | FNPRO | |

2. Executable statement processing routines, which convert executable statements and place the result on SYSUT1 in external (printable) form. The routines included in this category are the following:

| | | |
|---|---|---|
| ALPRO | DOPRO | PSPRO |
| ASPRO | FTPRO | RTPRO |
| CAPRO | GTPRO | STPRO |
| COPRO | IFPRO | |
| CTPRO | IOPRO | |

In addition, Phase 10 makes use of a number of utility routines which perform simple and frequently recurring functions, such as collecting characters, placing information in tables, etc. These routines are the following:

| | | |
|---|---|---|
| ARPRO | LTCOL | SPPRO |
| ENTTDI | PUTTBL | SPSUB |
| ERMS | SLPAR | XTBPRO |
| GETTBL | SPDIM | |
| GETWRD | SPDTA | |

Note: Routines GETTBL and PUTTBL are also used by Initialization Procedure, Phases 20 and 30.

A detailed description of each routine will be found in the individual section allotted to it.

Executable statements that cannot be converted at all, i.e. the statements BACKSPACE and ENDFILE (see Appendix D of the language conversion manual) are placed on SYSUT1, together with an appropriate message.

Comments cards in the FORTRAN program are placed on SYSUT1, in the same way as executable statements. If the sequence */ appears in the text of a comment, it is replaced by the sequence *-.

Any syntactical error in a source statement detected during the scan terminates the conversion of the statement concerned. The statement is treated as a comment and a diagnostic message is issued. No further assumption is made concerning the possible effect of the erroneous statement on the rest of the program.

Source program identifiers are checked to ensure against conflict with PL/I built-in function names or reserved words. In the event of possible conflict, the identifier is entered in the dictionary (TDI) with an appropriate flag. Moreover, if the identifier occurs within the text of an executable statement, a substitution name created by the LCP is automatically generated during conversion.

On completion of Phase 10, control returns to the Control Phase.

## SPECIFICATION STATEMENT PROCESSING ROUTINES

This section contains detailed descriptions of the routines used by Phase 10 to process FORTRAN specification statements. In order to simplify reference, the routines appear in alphabetical order.

BLPRO Chart 010

Purpose: To convert BLOCK DATA statements.

Called by: PH10

Processing: The input to this routine is the BLOCK DATA statement in the work area WKBF. A switch (BLSW) is set in this routine to be tested during the processing of the END statement (EDPRO) in order to determine whether the next END statement is part of a BLOCK DATA subprogram. A BEGIN statement is generated on SYSUT1.

Routine called: XTBPRO.

Exit: Calling routine.

CMPRO Chart 011

Purpose: To convert COMMON statements.

Called by: PH10

Processing: The input to this routine is the COMMON statement in the work area WKBF. The common block name is collected and placed in table TBK with a pointer to the list of variables. The list of variables is placed in table TCM and the names of variables are placed in the dictionary.

Routines Called: ENTTDI, ERMS, GETWRD, GETTBL, SPDIM, PUTTBL.

Exit: Calling routine.

DAPRO Chart 012

Purpose: To convert DATA statements.

Called by: PH10

Processing: The input to this routine is the DATA statement in the work area WKBF. The list of elements to be initialized is collected and if necessary, the address of an element, relative to the beginning of the array, is computed. The element and its address are placed in table TDA. The output from this routine is an entry in the dictionary (TDI) for each new name and an entry in table TDA for the names of variables and their literal value.

Routines Called: ENTTDI, ERMS, GETWRD, GETTBL, PUTTBL, SPSUB, SPDTA.

Exit: Calling routine.

DMPRO Chart 013

Purpose: To convert DIMENSION statements.

Called by: PH10

Processing: The input to this routine is the DIMENSION statement in the work area WKBF. The part of the statement following the keyword DIMENSION is scanned and the array name is entered in the dictionary (TDI) if it is not already there. The dimension information is collected and placed in table TDM and a pointer to this table entry is placed in the dictionary. In the case of an array name already in the dictionary, the existing entry is updated.

Routines called: ENTTDI, ERMS, GETWRD, GETTBL, PUTTBL, SPDIM.

Exit: Calling routine.

EDPRO Chart 014

Purpose: To convert END statements.

Called by: PH10

Processing: The input to this routine is the END statement in the work area WKBF. The END statement is converted and placed on SYSUT1.

Routine called: XTBPRO.

Exit: Calling routine.

EQPRO Chart 015

Purpose: To convert EQUIVALENCE statements.

Called by: PH10

Processing: The input to this routine is the EQUIVALENCE statement in the work area WKBF. The elements of the statement are collected and the equivalenced variables are placed in the equivalence table TEQ and in the dictionary TDI. If necessary, the address of an element relative to the beginning of an array is computed. The element and its address are placed in table TEQ.

Routines called: ENTTDI, ERMS, GETWRD, GETTBL, PUTTBL, SPSUB.

Exit: Calling routine.

EXPRO Chart 016

Purpose: To convert EXTERNAL statements.

Called by: PH10

Processing: The input to this routine is the EXTERNAL statement in the work area WKBF. The part of the statement following the keyword EXTERNAL is scanned. The subprogram names are checked for validity and added to the dictionary (TDI) with flags indicating their class, type and usage.

Routines called: ENTTDI, ERMS, GETWRD, PUTTBL.

Exit: Calling routine.

FNPRO Chart 017

Purpose: To convert FUNCTION, SUBROUTINE, and ENTRY statements.

Called by: PH10

Entries: There are three entry points to this routine: FNPRO, SBPRO, and ETPRO. The input, at entry point FNPRO, is the FUNCTION statement in the work area WKBF; at entry point SBPRO it is the SUBROUTINE statement; at entry point ETPRO it is the ENTRY statement.

Processing: The procedure name is placed on SYSUT1 with the keyword PROCEDURE (or ENTRY in the case of an ENTRY statement), followed by the formal parameters (unconverted) and the nonstandard return specifications (converted). A name followed by a number is assigned to each nonstandard return. This number represents the sequential number of each return within the parameter list. An additional formal parameter is created for a function subprogram and this name is entered in the dictionary (TDI). The converted statement is placed on SYSUT1.

Routines called: ENTTDI, ERMS, GETWRD, PUTTBL, SLPAR, XTBPRO.

Exit: Calling routine.

NLPRO Chart 018

Purpose: To convert NAMELIST statements.

Called by: PH10

Processing: The input to this routine is the NAMELIST statement in the work area WKBF. The namelist name is collected and placed in table TNL with a pointer to the list of variables. This list is placed in table TNV and the variable names are placed in the dictionary (TDI).

Routines called: ENTTDI, ERMS, GETTBL, GETWRD, PUTTBL.

Exit: Calling routine.

YIPRO Chart 019

Purpose: To process explicit type statements.

Called by: PH10

Entries: This routine has five entry points; the input to it consists of the following type statements: INTEGER (entry point YIPRO), REAL (entry point YRPRO), COMPLEX (entry point YCPRO), LOGICAL (entry point YLPRO), DOUBLE PRECISION (entry point YDPRO).

Processing: The type and length specifications of the input statement are saved. The length specification, the variable names, and the initial values, if any, are then scanned. The variable names are entered in the dictionary (TDI), with specification information and pointers to table TDM, and the initial values are processed by the routine SPDTA.

Routines called: ENTTDI, ERMS, GETWRD, PUTTBL, SPDTA.

Exit: Calling routine.

## EXECUTABLE STATEMENT PROCESSING ROUTINES

This section contains detailed descriptions of the routines used by Phase 10 to process executable statements in the FORTRAN source program. The organization of this section is the same as that of the preceding section.

ALPRO Chart 020

Purpose: To convert assignment statements.
Called by: PH10

Processing: The input to this routine is the assignment statement in the work area WKBF. The part of the statement located to the left of an equal sign is tested to determine if it is a variable, an array or a statement function name. Then:

1. a) If it is a variable, the name is placed in the dictionary (TDI).

   b) If it is an array, the subscript routine (SPPRO) is called.

2. If the name appearing to the left of an equal sign is followed by elements between parentheses and has not been given any dimension, it is a statement function name. A PL/I procedure statement is generated and the parameter list is processed. The pointers to parameter entries in the dictionary (TDI) are saved in an argument table. These pointers are placed on SYSUT1 after the PL/I PROCEDURE statement; they are separated by commas and followed by a semi-colon. The LCP , then, generates the following statements:

   RETURN (<expression>);

   END;

The routine ARPRO is called to process the expression located to the right of the equal sign.

The converted statement is placed on SYSUT1.

Routines called: ARPRO, ENTTDI, ERMS, GETWRD, PUTTBL, SPPRO, XTBPRO.

Exit: Calling routine.

ASPRO Chart 021

Purpose: To convert ASSIGN statements.

Called by: PH10

Processing: The input to this routine is the ASSIGN statement in the work area WKBF. The statement number following the keyword ASSIGN is collected and saved. The word TO is bypassed and the variable is placed in the dictionary (TDI). The converted statement is placed on SYSUT1.

Routines called: ENTTDI, ERMS, GETWRD, PUTTBL, XTBPRO.

Exit: Calling routine.

CAPRO Chart 022

Purpose: To convert CALL statements.

Called by: PH10

Processing: The input to this routine is the CALL statement in the work area WKBF. The FORTRAN CALL statement is converted into the PL/I CALL statement. The latter is placed on SYSUT1 followed by the subprogram name and its arguments, which are processed in the routine ARPRO.

Routines called: ARPRO, ERMS, ENTTDI, GETWRD, XTBPRO, LABPRO, PUTTBL.

Exit: Calling routine.

COPRO Chart 023

Purpose: To process comments cards.

Called by: PH10

Processing: The input to this routine is the comments card. The comments card is copied onto SYSUT1, the delimiters used being: /* for the beginning and */ for the end.

The */ sequence, where it occurs within the text of a comment, is replaced by the *-sequence.

Routine called: LTCOL.

Exit: Calling routine.

## CTPRO Chart 024

Purpose: To convert CONTINUE statements.

Called by: PH10

Processing: The input to this routine is the CONTINUE statement in the work area WKBF. If a label is associated with the CONTINUE statement, this statement is converted into a semi-colon. The label and the semi-colon are placed on SYSUT1. If no label is attached to the statement, this statement is ignored.

Routines called: ERMS, GETWRD, XTBPRO, LABPRO.

Exit Calling routine.

## DOPRO Chart 025

Purpose: to convert DO statements.

Called by: PH10

Processing: The input to this routine is the DO statement in the work area WKBF. The statement number specifying the end of the DO loop is ignored. The DO index is collected and placed on SYSUT1. The DO parameters (PAR1, PAR2, PAR3) are collected and saved. If one of the two limits is variable, switch DOSW is set on. Then, one of the following three PL/I statements is generated:

1. If one of the limits is variable:

   DO index = PAR1 TO MAX(PAR1, PAR2) BY PAR3;

2. If the limits are constant and PAR1 is less than PAR2:

   DO index = PAR1 TO PAR2 BY PAR3;

3. If the limits are constant and PAR1 is greater than or equal to PAR2:

   DO index = PAR1;

The output from this routine is the converted DO statement on SYSUT1.

Routines called: ERMS, GETWRD, PUTTBL, XTBPRO, LABPRO, ENTTDI.

Exit: Calling routine.

## FTPRO Chart 026

Purpose: To convert FORMAT statements.

Called by: PH10

Processing: The input to this routine is the FORTRAN FORMAT statement in the work area WKBF. The FORTRAN format codes are translated as follows:

| | | |
|---|---|---|
| A | A | to A |
| D | E | |
| E | E | |
| F | F | with call to LBLNK if BLKZR option is on (see Appendix B) |
| G | G | with a warming message |
| I | F | with call to LBLNK if BLKZR option is on (see Appendix B) |
| L | B | with a warning message |
| Z | Z | with a warning message. |
| / | SKIP(n) | where n is the number of consecutive slashes |
| T | COLUMN( ) | |
| X | X | |

A string of characters appearing with H and quotes codes are replaced by an A-format code with the number of characters contained in the string.

The field count and the decimal count are enclosed in parentheses and the period is replaced by a comma.

COLUMN(1) is generated at the beginning of each PL/I FORMAT statement, to force the end of the current record.

The carriage control character read at the beginning of the FORMAT statement and after a slash is converted into the appropriate PL/I statement (PAGE, SKIP or COLUMN).

The contents of the FORMAT statement is saved in a packed form, "packed FORMAT", in the following way:

- n ( format-code.

  The left parenthesis is given an internal code '*' and saved with the associated iteration factor.

- Character-strings ('-' or nH)

The string is collected and placed in table TDT.

The pointer to the entry in table TDT is then associated with an LCP-created variable which is saved in table TDU.

A reference to this entry in table TDU is saved in packed format following the internal code P.

- A format-code

  The total number of successive A-format codes not separated by any other code, by a character string, or by a left parenthesis, is placed in packed FORMAT after the internal code A.

- D, E, F, G, I, L, Z format-codes

  The total number of successive items associated with these codes and not separated by any of the previous ones is saved in packed FORMAT with the internal code N.

- Right parenthesis

  A right parenthesis is used as a packed FORMAT internal code to indicate the end of an iteration group and of packed FORMAT.

  Packed FORMAT information is placed in the table TDT and the pointer to the corresponding entry is saved with the entry in table TFM.

  If the FORMAT statement does not contain any A-format code or character string, the packed FORMAT is not saved.

  The converted PL/I FORMAT statement is placed on SYSUT1. The tables TFM and TDU are updated (see IOPRO for use of these tables).

Figure 2 illustrates tables TFM, TDT, and TDU.

Routines called: ERMS, GETWRD, LTCOL, XTBPRO, GETTBL, PUTTBL, LABPRO.

Exit: Calling routine.

GTPRO_____Chart 027

Purpose: To convert GO TO statements.

Called by: PH10

Processing: The input to this routine is the GO TO statement in the work area WKBF. This routine first tests what type of GO TO statement is concerned. Then,

1. In the case of an unconditional GO TO statement, the PL/I statement "GO TO EXTLABn" is placed on SYSUT1.

2. In the case of an assigned GO TO statement, the PL/I statement "GO TO variable" is placed on SYSUT1. An entry for the variable is made in the dictionary (TDI).

3. In the case of a computed GO TO statement, the following PL/I statement is placed on SYSUT1:

   IF (<index><=<number of parameters> &<index>>0) THEN GO TO BRANCHi (<index>)

   The index is placed in the dictionary. An entry in the dictionary is also created for the sequence number associated with the label BRANCH, with a pointer to table TDT, which contains the list of the GO TO parameters.

The output from this routine is an updated entry in the dictionary (TDI), an entry in table TDT in the case of a computed GO TO statement, and a converted PL/I statement on SYSUT1.

Routines called: ENTTDI, ERMS, GETWRD, PUTTBL, XTBPRO, LABPRO, BRNPRO.

Exit: Calling routine.

IFPRO_____Chart 028

Purpose: To convert IF statements.

Called by: PH10

Processing: The input to this routine is the IF statement in the work area WKBF. The FORTRAN IF statement is converted into a PL/I IF statement and placed on SYSUT1, followed by the arithmetic expression processed by the routine ARPRO. In addition, if the IF is an arithmetic one, the three branch labels are analyzed and converted, taking into account the relationship of the three branch labels and the label of the statement immediately following. The output is the PL/I statement on SYSUT1.

Routines called: ARPRO, ERMS, GETWRD, XTBPRO, LABPRO, BRNPRO.

Exit: Calling routine.

Purpose: To convert input/output statements.

Called by: PH10

Processing: The input to this routine is input/output statement in the work area WKBF.

The first part of the converted statement, for READ or WRITE, respectively, is:

PUT FILE (FT nF01) or GET FILE (FTnF01)

where n is the data-set reference number of the FORTRAN statement.

This part of the converted statement may be preceded by ON conditions, if necessary.

If the ERROR option is present, the routine generates the statement ON TRANSMIT.

If the END option is present, the routine generates the statement ON ENDFILE.

If the NAMELIST option is present, the routine generates DATA followed by the list of variables. If this option is not present, the routine generates the EDIT option followed by the data list.

The output from this routine consists of the appropriate PL/I input/output statements on SYSUT1. The data set and FORMAT reference numbers are saved. The corresponding table TFM and packed FORMAT entries are collected.

Each element of the I/O list is associated with a packed FORMAT element.

- N code

  An N code indicates the number of variables in the I/O list which are placed into the PL/I converted statement.

- A code

  An A code indicates the number of variables in the I/O list which are placed into the converted PL/I statement. The DICTIONARY entries of these variables are flagged as type CHARACTER.

- P code

  When a P code is encountered, the corresponding entry is table TDU is collected. The LCP-created variable is inserted in the I/O list.

  If the end of packed FORMAT information is reached before the end of the I/O list, the processing described above goes on from the last * which corresponds to the last pair of parentheses of level-1.

  This processing stops when the end of the I/O list is reached or when a DO loop in the I/O list is encountered.

- Empty I/O list

  If there is no I/O list, the packed FORMAT information is checked to determine whether an LCP-created variable or only a carriage control option (SKIP, PAGE) need be generated.

  If there is no packed FORMAT information associated with a FORMAT reference, the I/O statement is converted directly into PL/I.

  No conversion occurs if data set or FORMAT references are not integer constants, or if there is no FORMAT reference.

Figure 2 illustrates tables TFM, TDT, and TDU.

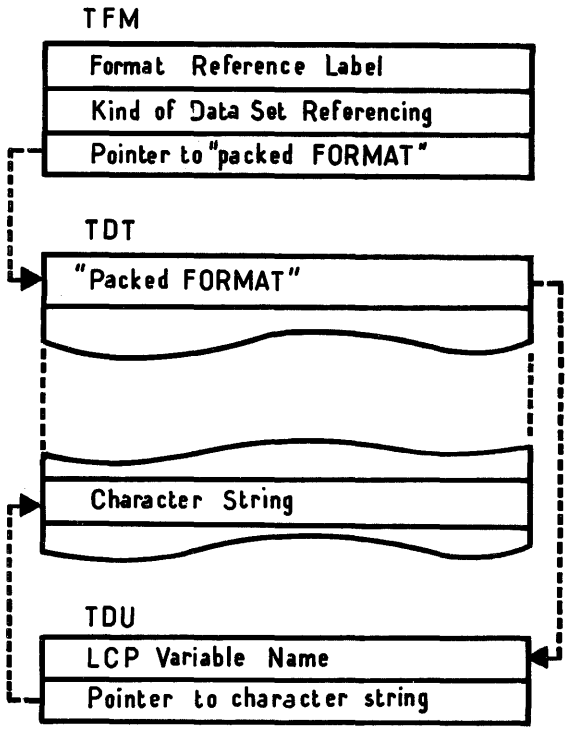



Figure 2. Tables TFM, TDT, TDU

Routines called: ENTTDI, ERMS, GETWRD, PUTTBL, XTBPRO, LABPRO, BRNPRO, GETTBL.

Exit: Calling routine.

PSPRO Chart 031

Purpose: To convert PAUSE statements.

Called by: PH10

Processing: The input to this routine is the PAUSE statement in the work area WKBF. The field following the keyword PAUSE is placed in the converted statement on SYSUT1. This field can be a blank, a number or a character string.

Routines called: ERMS, GETWRD, LTCOL, XTBPRO, LABPRO.

Exit: Calling routine.

RTPRO Chart 032

Purpose: To convert RETURN statements.

Called by: PH10

Processing: The input to this routine is the RETURN statement in the work area WKBF. The data following the RETURN statement is collected. If this field is blank, the RETURN statement is placed on SYSUT1. If the field is a constant or a variable, a transfer is made to the return parameter created or to an array of returns. The names created, together with a reference to their initial values, are entered in the dictionary (TDI).

Routines called: ENTTDI, GETWRD, PUTTBL, XTBPRO, LABPRO.

Exit: Calling routine.

STPRO Chart 033

PURPOSE: To convert STOP statements.

Called by: PH10

Processing: The input to this routine is the STOP statement in the work area WKBF. The field following the keyword STOP is scanned for significant data. If a character string is present, it is placed with the converted statement on SYSUT1.

Routines called: ERMS, GETWRD, XTBPRO, LABPRO

Exit: Calling routine.

## UTILITY ROUTINES

This section contains detailed descriptions of the utility routines used by Phase 10. The organization is the same as that of the two preceding sections.

ARPRO Charts 034, 035, 036, 037

Purpose: To convert arithmetic or logical expressions.

Called by: ALPRO, CAPRO, IFPRO.

Processing: The input to this routine is the arithmetic or the logical expression in the work area WKBF. The variable names are placed in the dictionary (TDI) and the elements of the arithmetic or logical expression are placed on SYSUT1.

The expression is scanned twice in the FORTRAN statement. For a procedure CALL statement or a function call, all the arguments are scanned by each scan.

First Scan: (Charts 034, 035, 036) During the first scan, a "pushdown table" is used. Each entry in this table contains: the pointers to the beginning of a sub-expression, the type of the sub-expression (initialized integer), and the previous delimiter (operator or other).

An entry is added to the "pushdown table" when one of the following is encountered:

- a left parenthesis
- a function name
- a slash
- a NOT operator

These items constitute the beginning of a subexpression.

An entry is withdrawn from the "pushdown table" when one of the following is encountered:

- a right parenthesis
- a comma in a list of arguments
- a logical or relational operator
- an end of expression

These items constitute the end of a sub-expression.

During this scan, the entries in the pushdown table are updated according to the type of variable and the priority of the operator.

If integer division is found, pointers that indicate where to make an insert are taken from the pushdown table and are saved in an internal area (Insert Table).

The Insert Table is also used for inserts other than those for integer division. It contains pointers to the source expression that indicate where to make the insertion and the kind of insertion, as follows:

- TRUNC function for integer division.
- BINARY function for integer constants used as arguments.
- The sign between the two parts of a complex constant.
- The letter 'I' after the imaginary part of a complex constant.
- Right parenthesis at the end of an expression under the scope of a NOT operator or corresponding to the end of an expression used as an argument of the TRUNC or BINARY PL/I function.

Second Scan: (Chart 037)
When the end of an expression has been reached, the source expression is scanned again. While the scan continues, the expression is converted using information saved in the Insert Table. When the scan pointers equal the pointers in the Insert Table, the insertion is made in the translated expression.

Figure 3 describes the flow of ARPRO processing.

Routines called: ENTTDI, ERMS, GETWRD, PUTTBL, SPPRO, XTBPRO, LTCOL.

Exit: Calling routine.

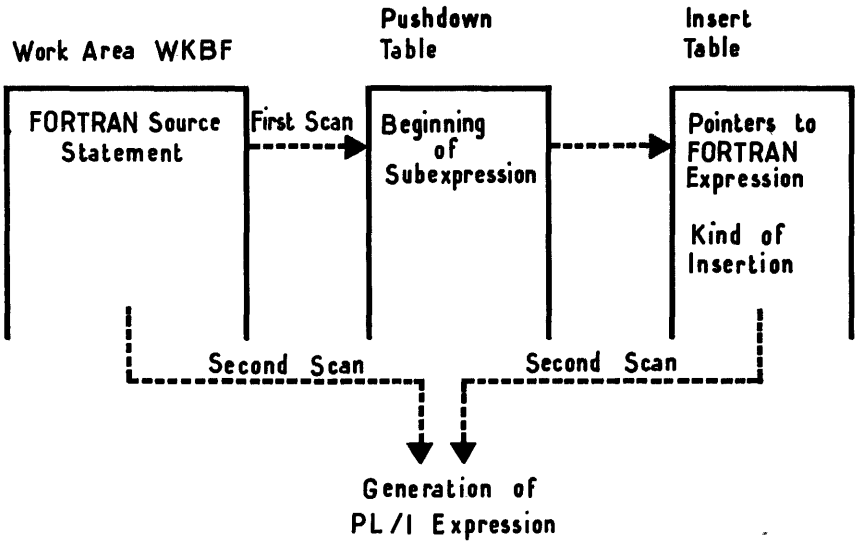



Figure 3. ARPRO Processing

ENTTDI Chart 038

Purpose: To extract the characteristics of a variable from the dictionary (TDI) according to the variable name.

Called by: All statement processing routines except BLPRO, CTPRO, EDPRO, PSPRO, STPRO.

Processing: The input to this routine is the name of a variable. The dictionary is scanned to determine whether it contains the variable. If it does, the return label is used; if not, the next sequential instruction is executed. Table entry pointer TBLCW is then updated.

Routine called: None.

Exit: Calling routine.

ERMS Chart 039

Purpose: To assign an error number to an erroneous statements.

Called by: Any routine in Phase 10.

Processing: The input to this routine is any FORTRAN statement in the work area WKBF. The number assigned to a particular error is placed on SYSUT1.

Routine called: LTCOL, XTBPRO.

Exit: Calling routine.

GETTBL Chart 040

Purpose: To get a table from SYSUT1.

Called by: CMPRO, DAPRO, DMPRO, EQPRO, FTPRO, NLPRO, IOPRO, PH30, PH20, Initialization Procedure.

Processing: The input to this routine is the table whose name appears as the argument of the statement CALL GETTBL. For example, in the statement CALL GETTBL (TAB1), TAB1 is the name of the table from which the information is fetched. The table entry pointer TBLCW contains the address of the table entry. The elements of the entry are fetched and placed in the entry format. In the case of a common block name, a namelist name, and a label format, a search is made by comparing these names with the identifiers placed in tables TBK, TNL, and TFM respectively. On output, the entry of the referenced table is placed in a buffer.

Routines called: None.

Exit: Calling routine.

GETWRD Chart 041

The characters collected are placed, with their type and length, in an area called WORD.

Purpose: To get a constant, a name or a delimiter.

Called by: Any routine in Phase 10, Initialization Procedure.

Processing: The input to this routine is the source statement in the work area WKBF. This routine divides the source statement into groups of characters classified as numeric, alphabetic or delimiter depending upon the first character collected.

The characters collected are placed, with their type and length, in an area called WORD. If a delimiter is collected, XTDLM contains the delimiter itself. If an alphabetic group is collected, XTDLM contains the character that follows the group (which is a delimiter). If a numeric group is collected, XTDLM contains the character that follows the group (which is either a delimiter or an alphabetic character).

Routine called: None.

Exit: Return to one of three labels, depending on whether a numeric string, an alphameric string, or a delimiter was collected.

LTCOL Chart 042

Purpose: To collect the alphameric data in FORTRAN statements.

Called by: ARPRO, COPRO, FTPRO, PSPRO, SPDTA.

Processing: The input to this routine consists of the alphameric literals in the work area WKBF. The routine collects the literals and places them on SYSUT1 or in table TDT. The character count is placed on SYSUT1 before the character-string, if necessary.

Routines called: XTBPRO.

Exit: Calling routine.

PUTTBL Chart 043

Purpose: To copy tables on SYSUT1

Called by: ALPRO, ARPRO, ASPRO, CMPRO, DAPRO, DMPRO, DOPRO, EQPRO, EXPRO, FNPRO, IOPRO, NLPRO, RTPRO, SPDIM, SPPRO, YIPRO in Phase 10; DTPRO, EVPRO in Phase 20, Initialization Procedure, Phase 30.

Processing: The input to this routine is the name of the table entry. In the statement: CALL PUTTBL (TAB1), TAB1 is the name of a table where the entry will be placed. The entry is placed in a buffer. All the entries of the same table are chained together using a pointer in each entry. The table entry pointer TBLCW is analyzed to see if its value is equal to zero: if it is, a new entry is placed in the table; if it is not, the existing entry is updated.

Routines called: None.

Exit: Calling routine.

SLPAR Chart 044

Purpose: To collect the parameter list of an entry point

Called by: FNPRO.

Processing: The input to this routine is the parameter list in the work area WKBF. Each parameter is placed on SYSUT1.

Routines called: ERMS, GETWRD, XTBPRO ENTTDI, PUTTBL.

Exit: Calling routine.

SPDIM Chart 045

Purpose: To process the list of dimensions of a declarative statement.

Called by: CMPRO, DMPRO, YIPRO.

Processing: The input to this routine is the specification statement in the work area WKBF. An entry in the dimension table (TDM) is made for each dimension collected. The pointer to this entry is placed in the overflow table (TOV).

Routines called: ENTTDI, ERMS, GETTBL, GETWRD, PUTTBL.

Exit: Calling routine.

SPDTA Chart 046

Purpose: To collect DATA literals and place them in table TDT.

Called by: DAPRO, YIPRO.

Processing: The input to this routine consists of the DATA and TYPE statements in WKBF. Literals are collected and rearranged to be entered in table TDT.

Routines called: LTCOL, PUTTBL.

Exit: Calling routine.

SPPRO Chart 047

Purpose: To process subscripts.

Called by: ALPRO, ARPRO, (first, second scan) IOPRO.

Processing: The input to this routine is the FORTRAN statement in work area WKBF. The pointers to the commas and to the last right parenthesis of the subscript are collected. Using the pointers, the subscript parameters are generated in reverse order and placed on SYSUT1. The pointers are then positioned on the last right parenthesis.

Routines called: ENTTDI, GETWRD, PUTTBL, XTBPRO.

Exit: Calling routine.

SPSUB Chart 048

Purpose: To compute the position of an element within a block, relative to the first element of the block.

Called by: DAPRO, EQPRO.

Processing: The input to this routine is the FORTRAN statement in the work area WKBF. Given DIMENSION A (a, b... n*), where A has subscripts (i, j... r*), the position of the element is computed using the following formula:

$$i+(j-1)a+(k-1)ab+\ldots(r*-1)ab\ldots n*$$

The result is passed to the calling routine.

Routines called: ERMS, GETTBL, GETWRD.

Exit: Calling routine.

XTBPRO Chart 049

Purpose: To place converted executable statements on SYSUT1.

Called by: LTCOL and all statement processing routines except CMPRO, DAPRO, DMPRO, EQPRO, EXPRO, NLPRO, YIPRO.

Entries: This routine has three entry points: XTBPRO, BRNPRO, and LABPPO. At entry points XTBPRO and BRNPRO the input is a character-string in the area WORD; at entry point LABPRO it is the label in the area LCCRT.

Processing: If the input character-string conflicts with a PL/I built-in function name, it is modified (see section "Form of LCP Substitution Names" in the language conversion manual). Each time a line in the buffer is full, this line is placed on SYSUT1, and the next line is initialized.

At entry point BRNPRO, switch BRSW is set off.

At entry point LABPRO, switch BRSW is set on. Leading zeros and all blanks are skipped.

A delimiter is placed on SYSUT1. This delimiter is a semicolon for BRNPRO and a colon for LABPRO.

Routines called: None.

Exit: Calling routine.

Chart 009. Overall Logic of Phase 10

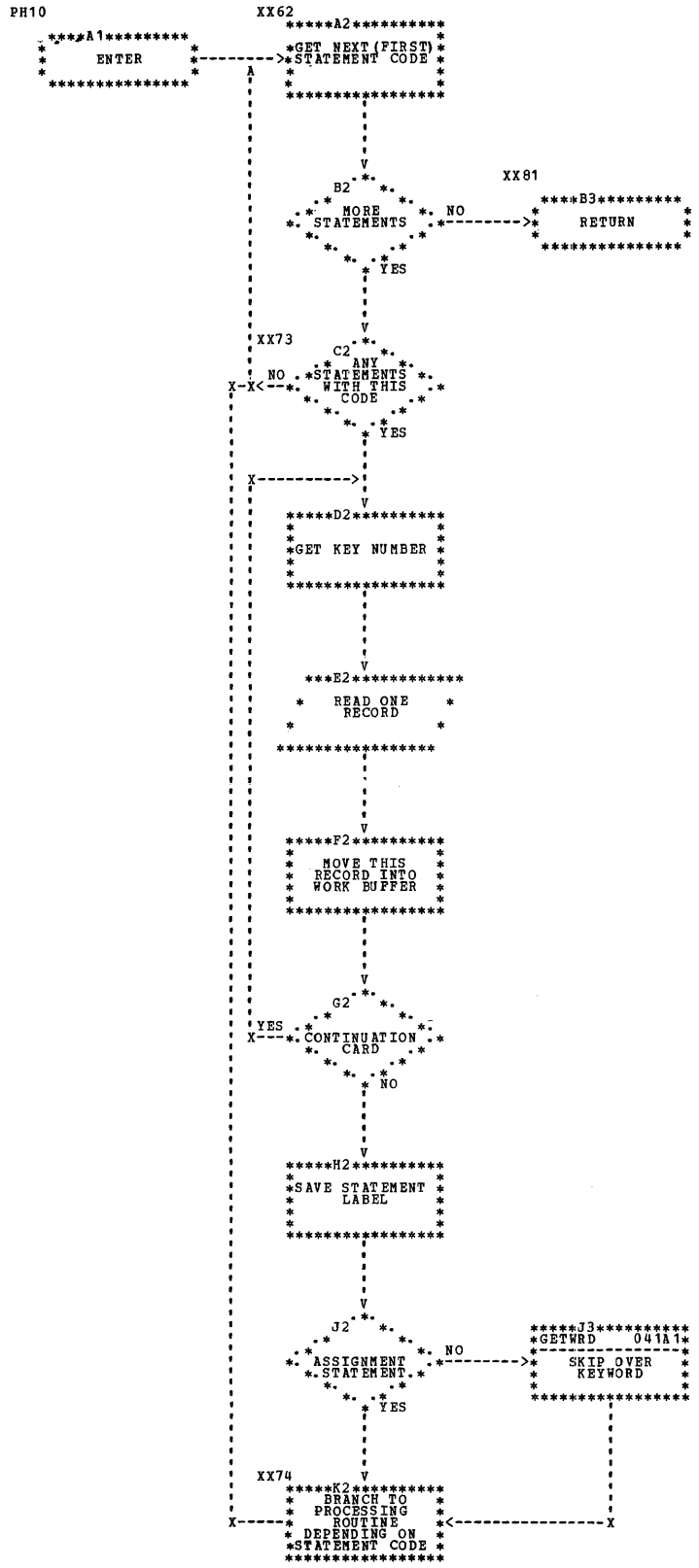
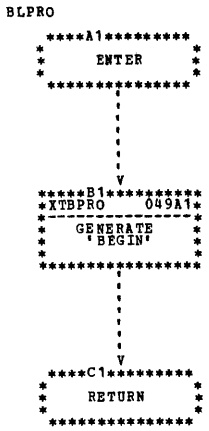
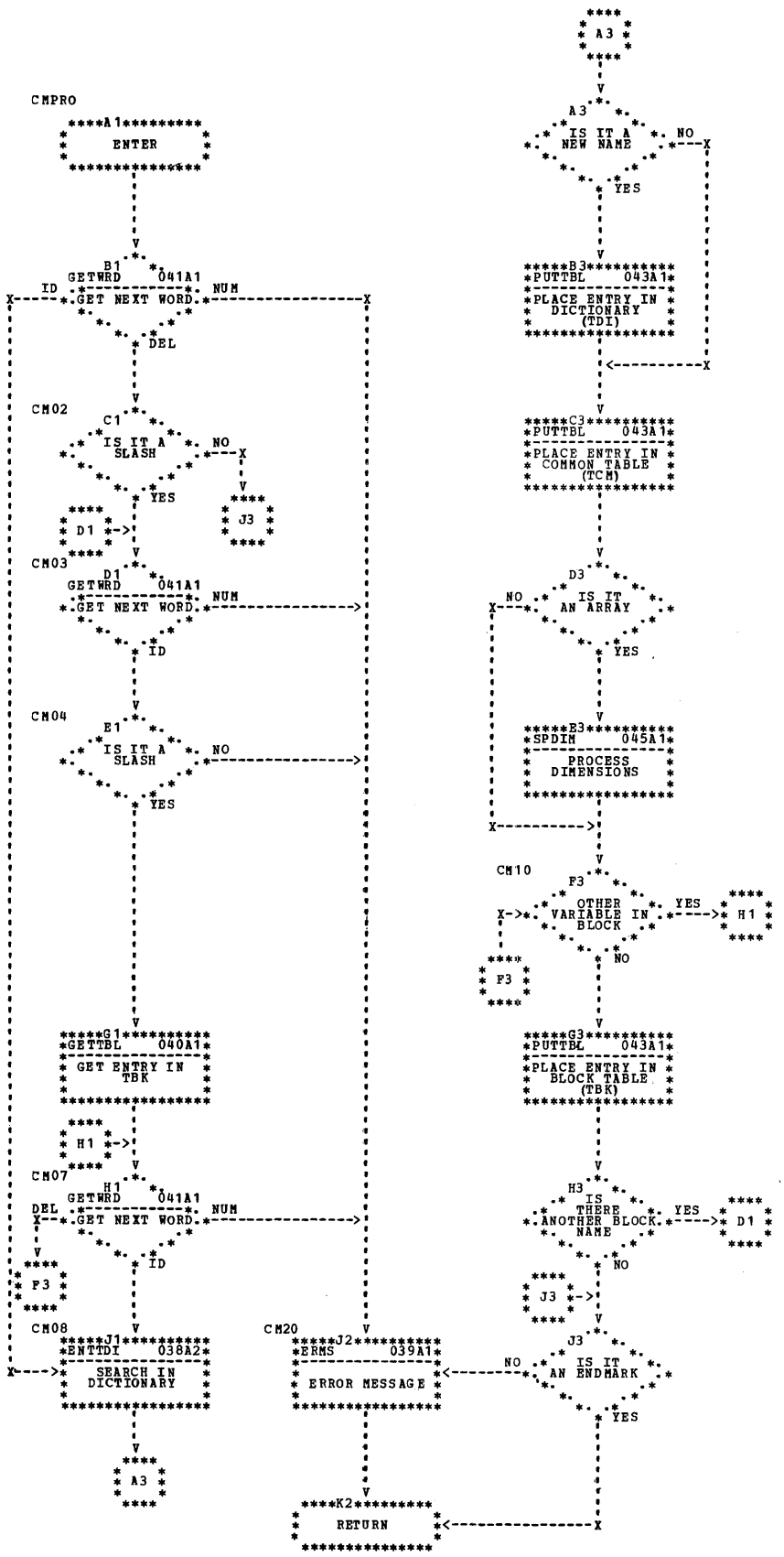


Chart 010. BLPRO Routine



## Chart 011. CMPRO Routine



## Chart 012. DAPRO Routine

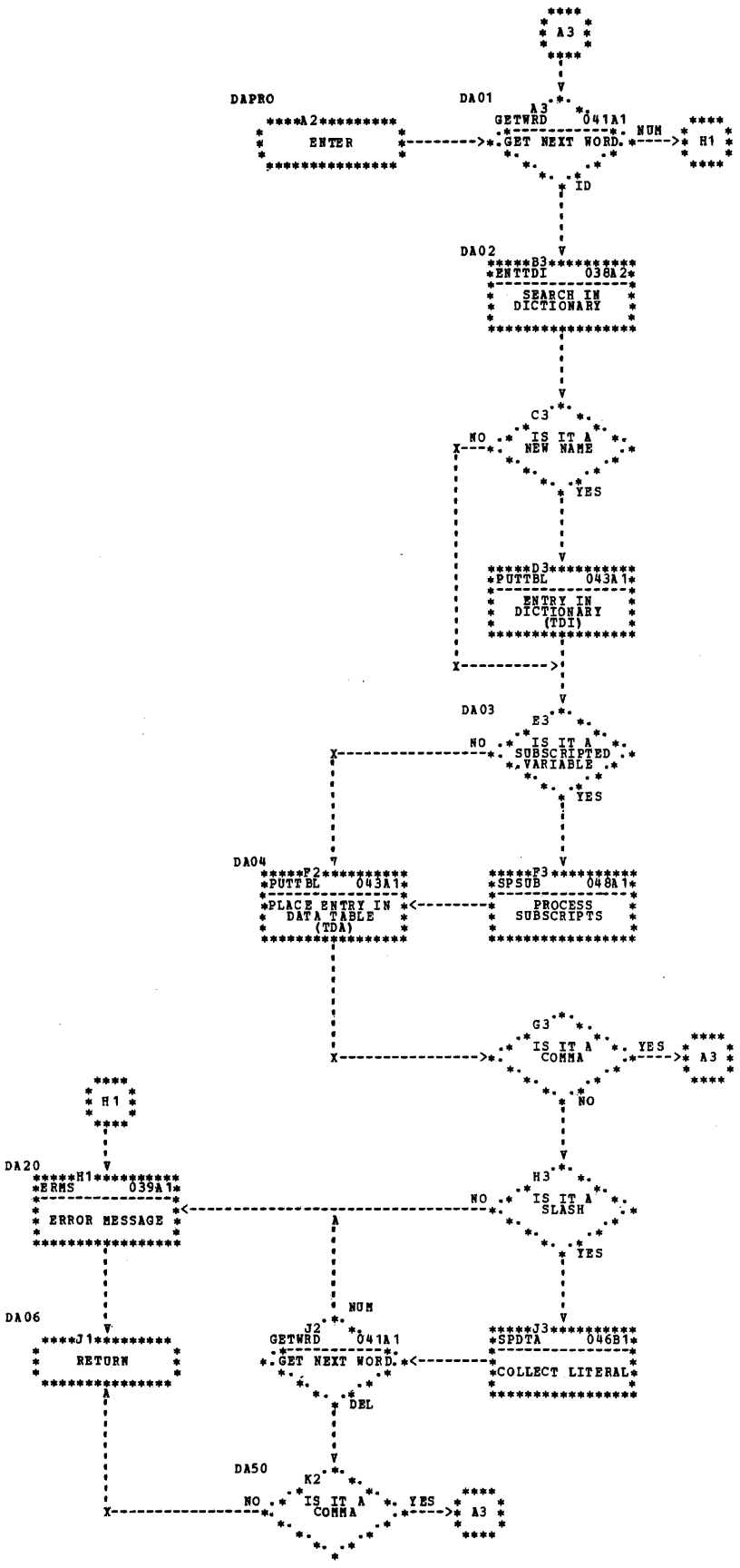
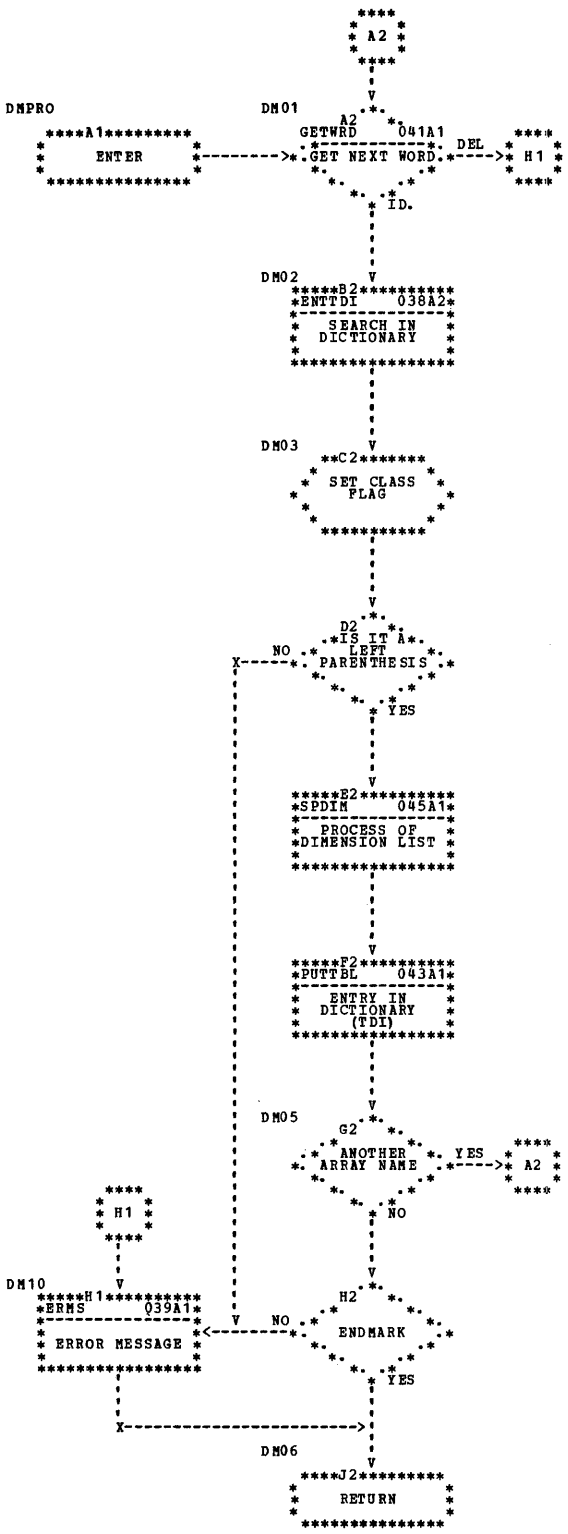
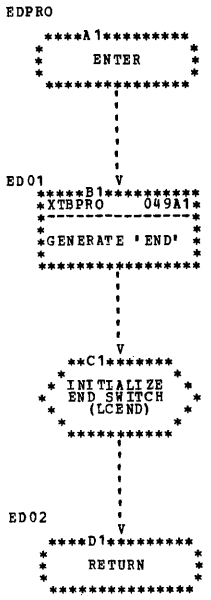


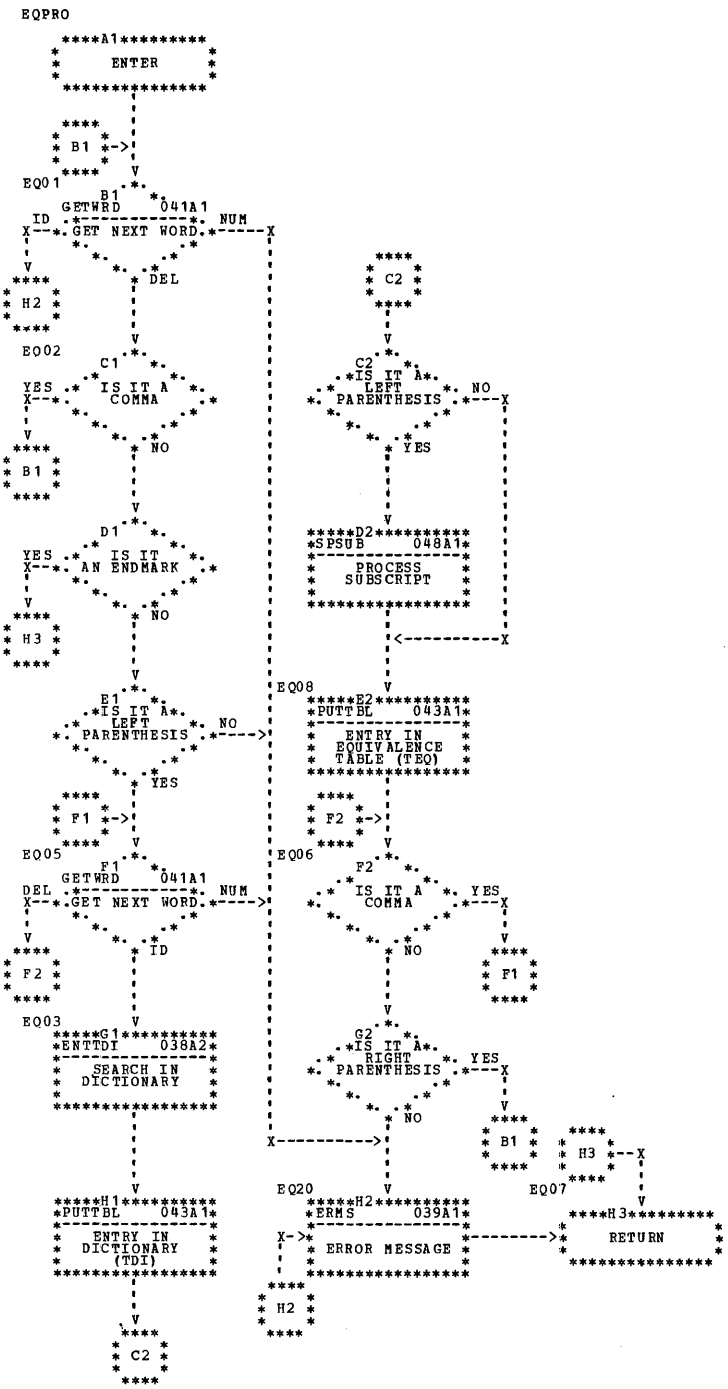
Chart 013. DMPRO Routine



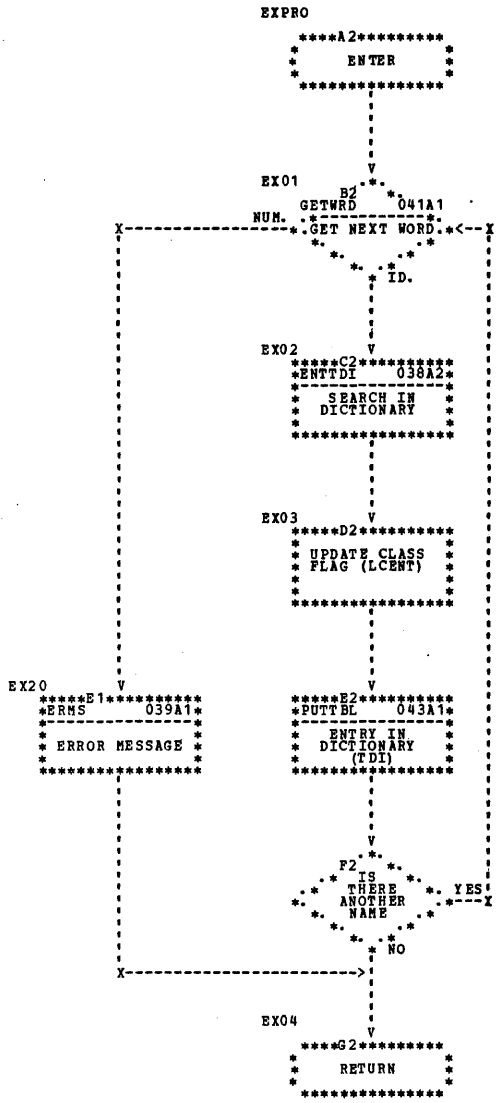
## Chart 014. EDPRO Routine



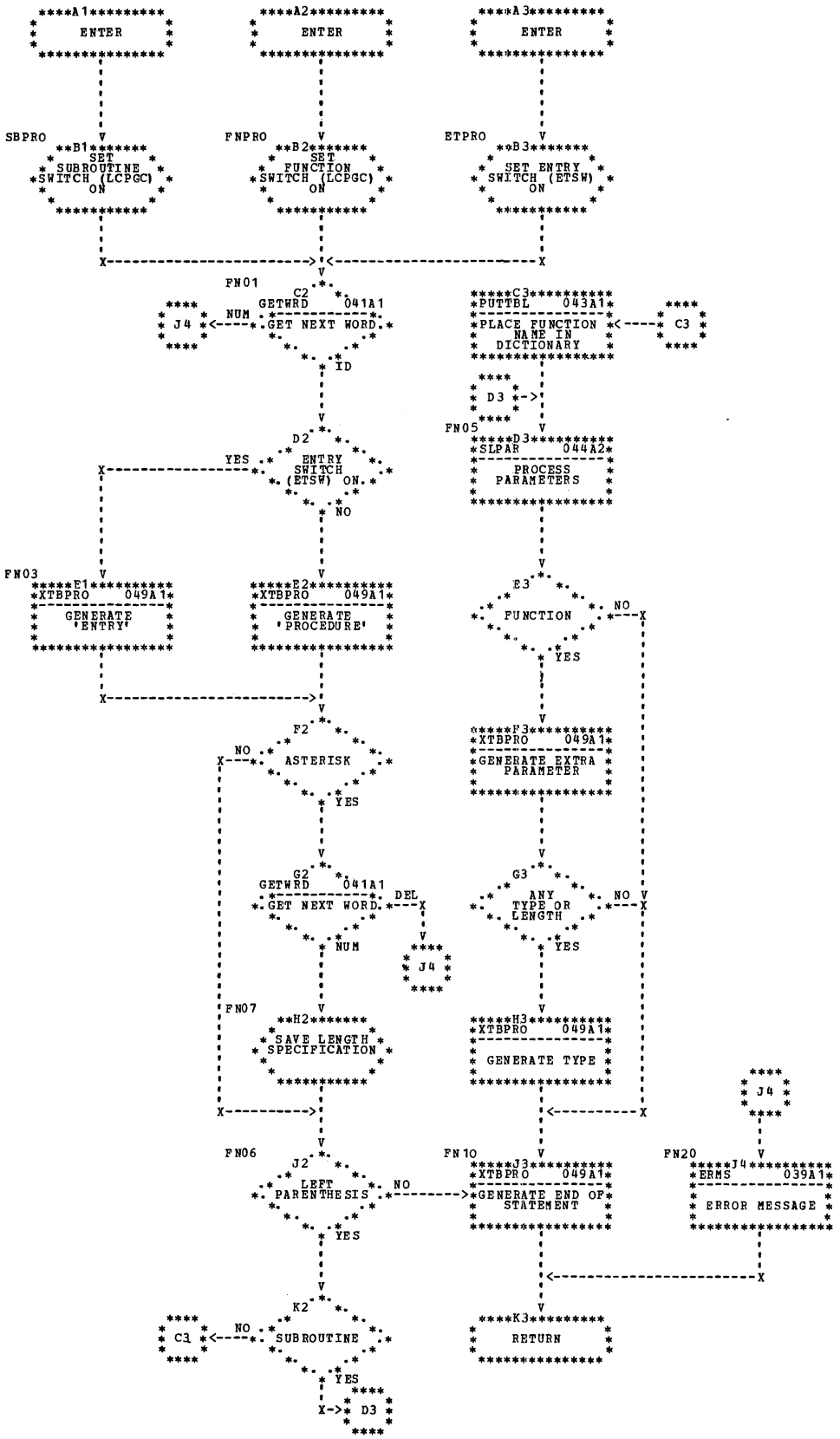
## Chart 015. EQPRO Routine



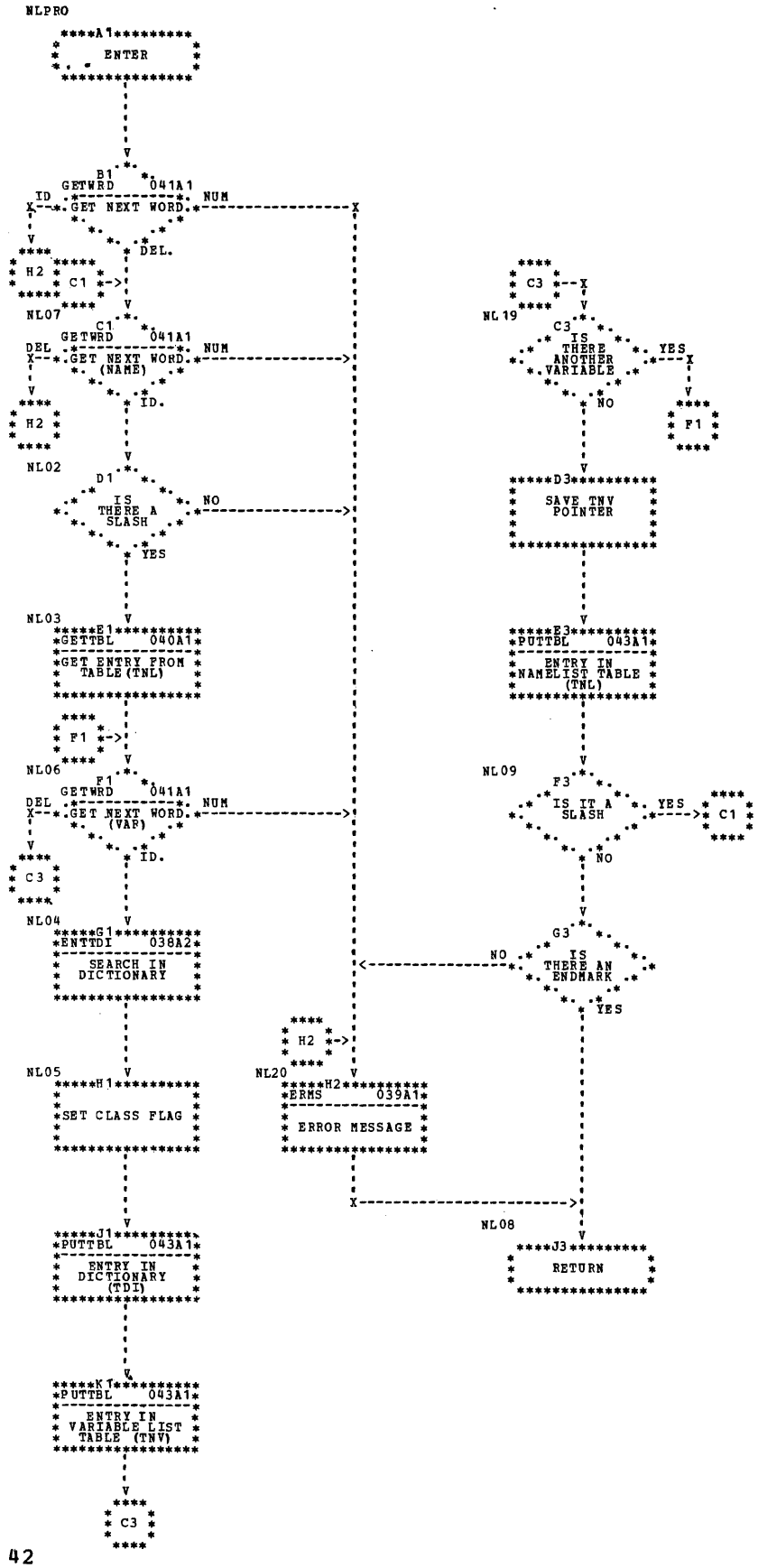
## Chart 016. EXPRO Routine



## Chart 017. FNPRO Routine



## Chart 018. NLPRO Routine



## Chart 019. YIPRO Routine

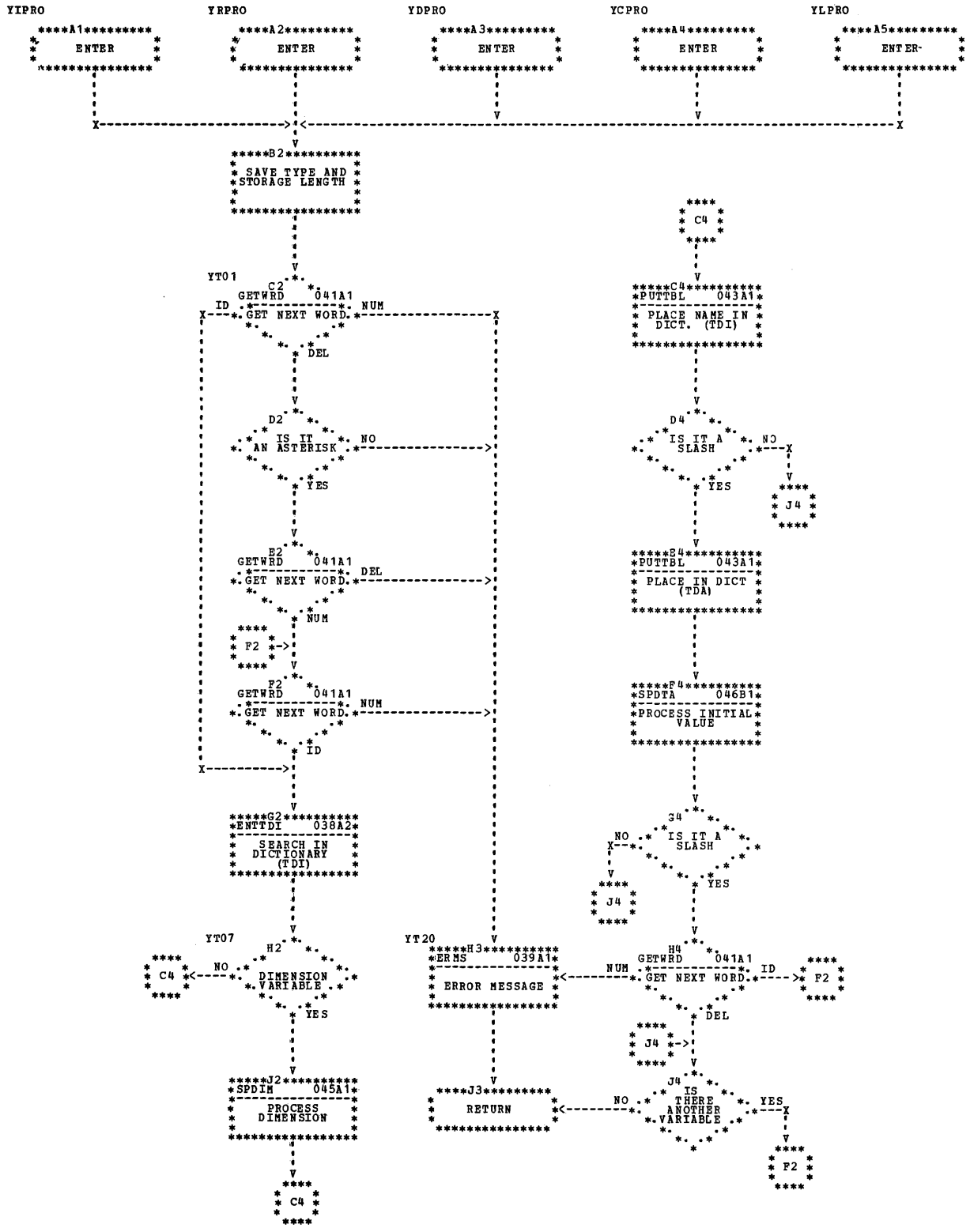
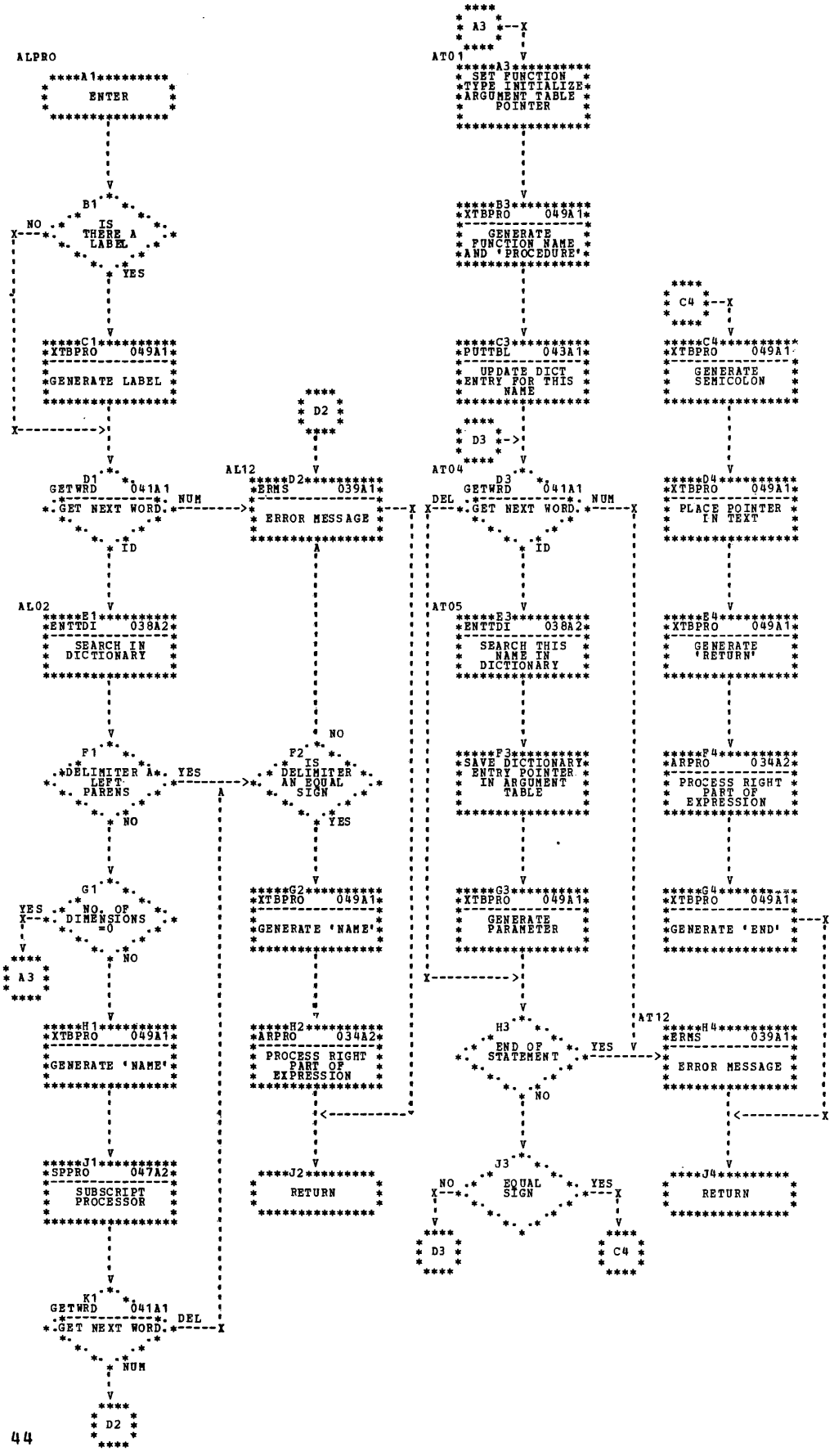
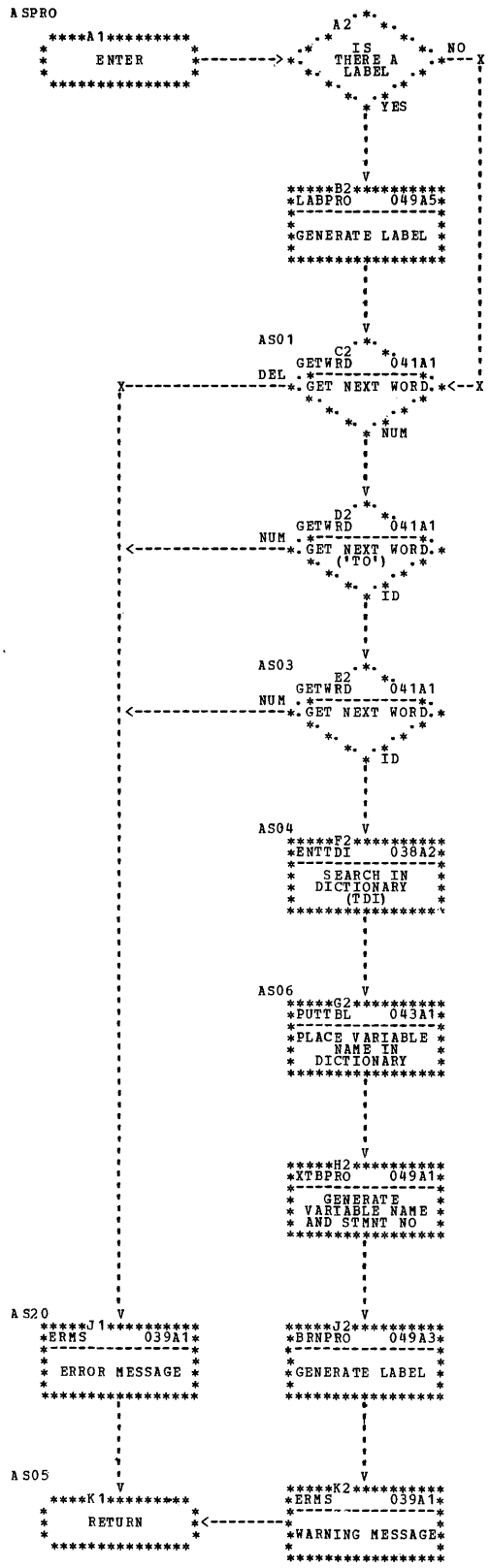


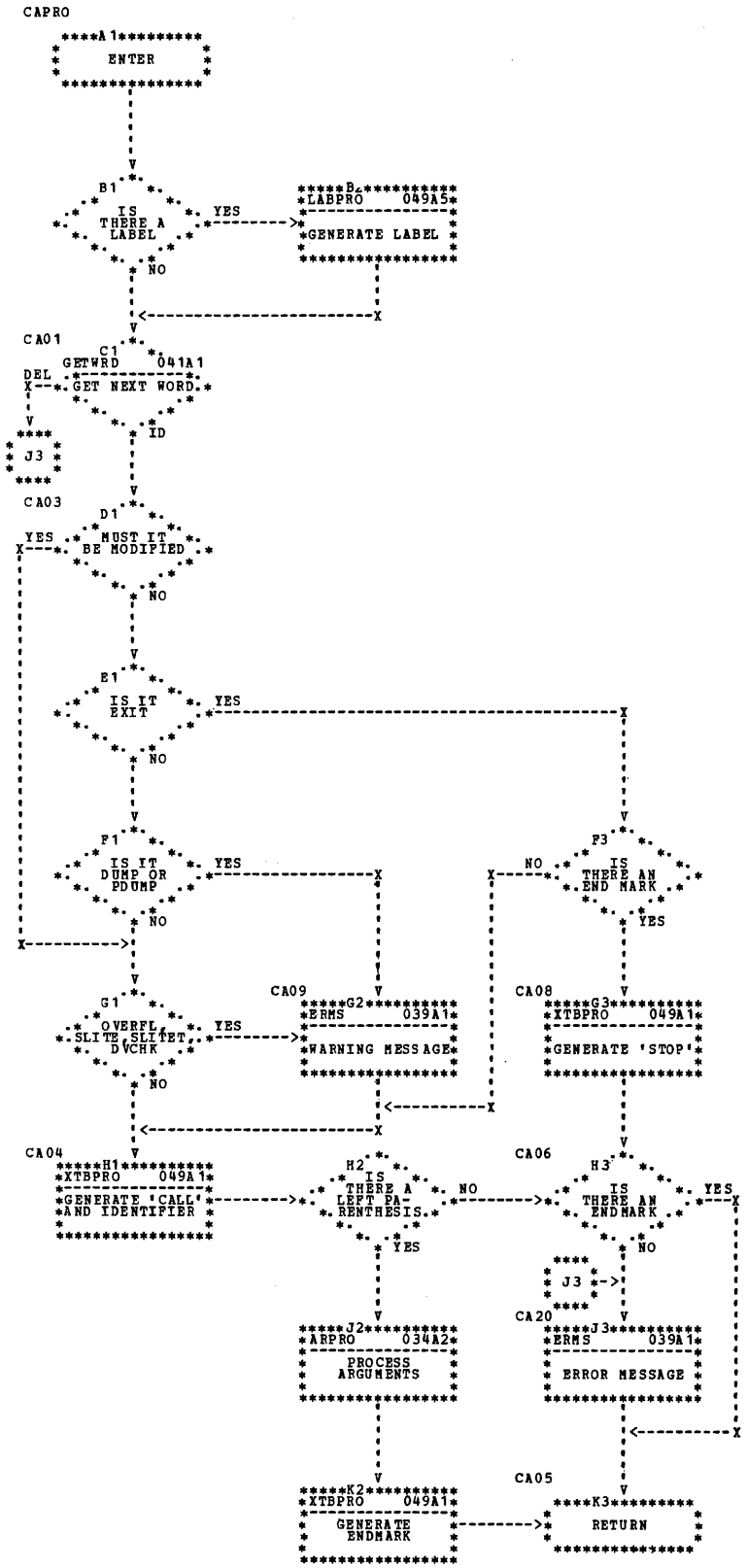
Chart 020. ALPRO Routine



## Chart 021. ASPRO Routine



## Chart 022. CAPRO Routine



## Chart 023. COPRO Routine

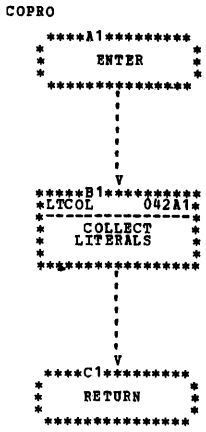
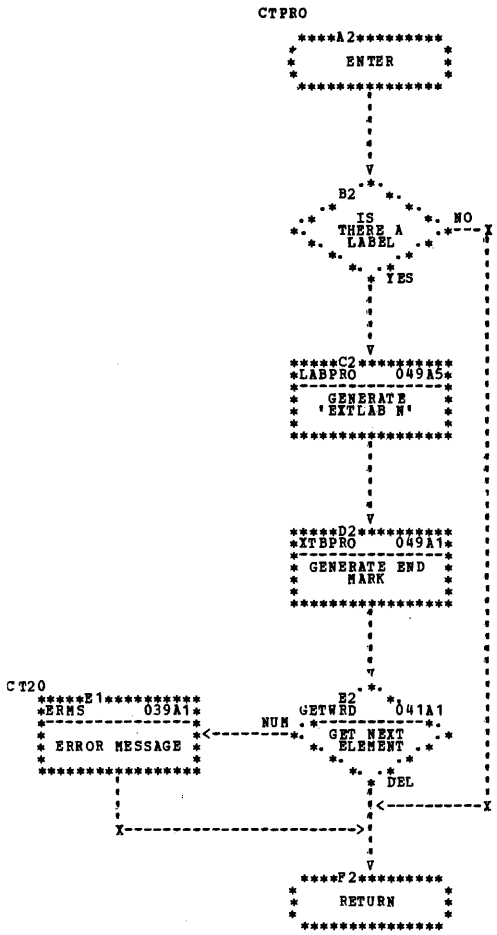
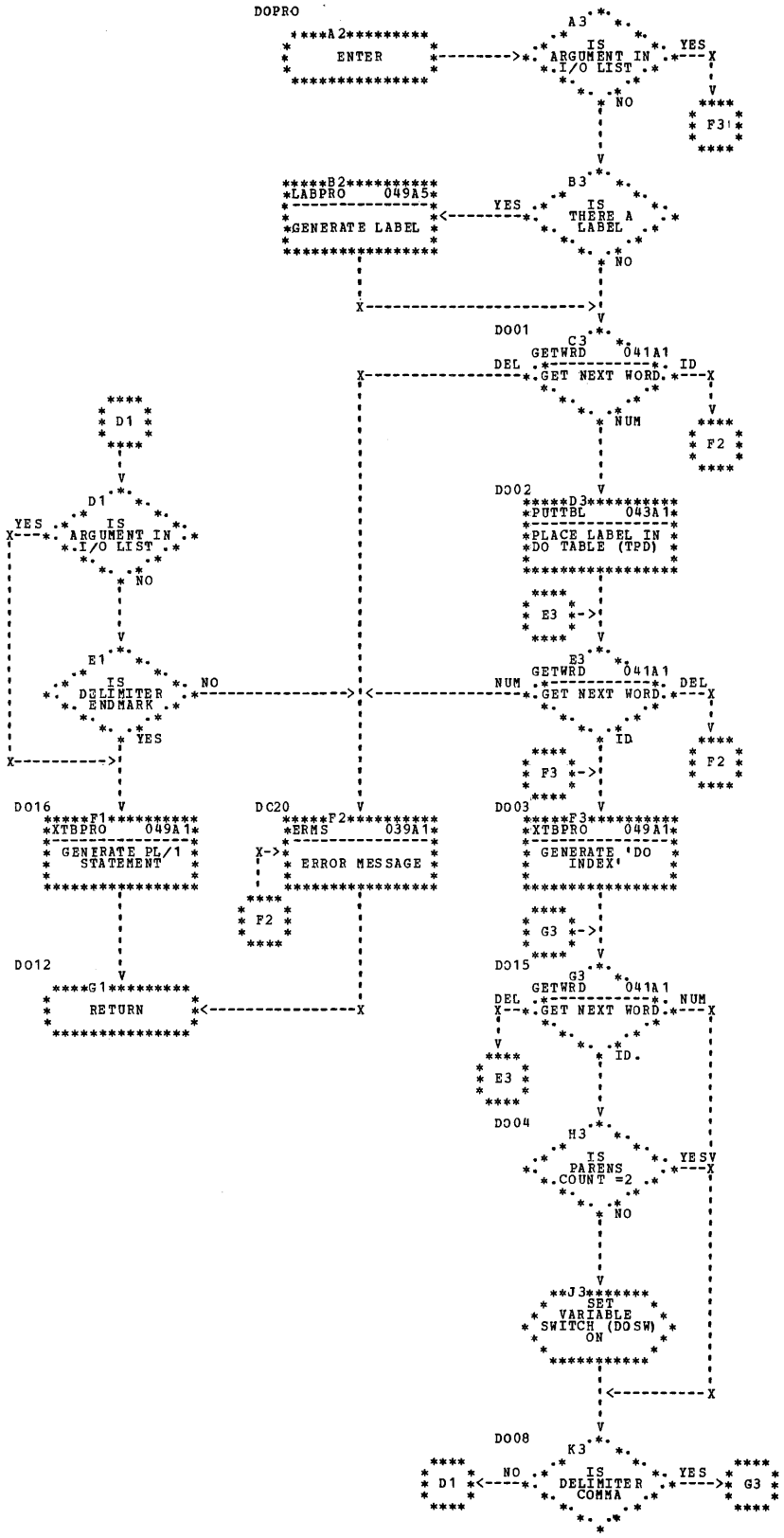


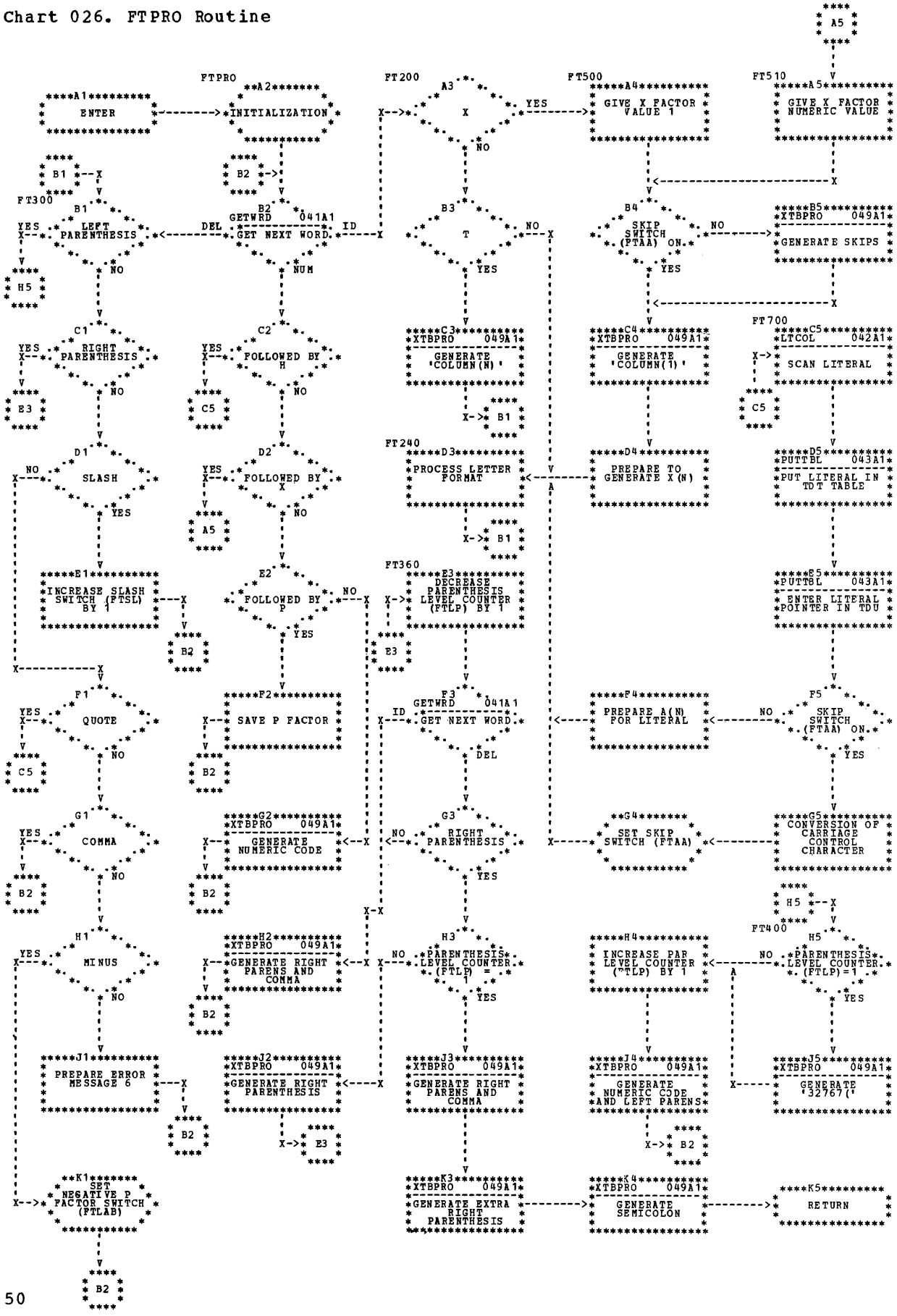
Chart 024. CTPRO Routine



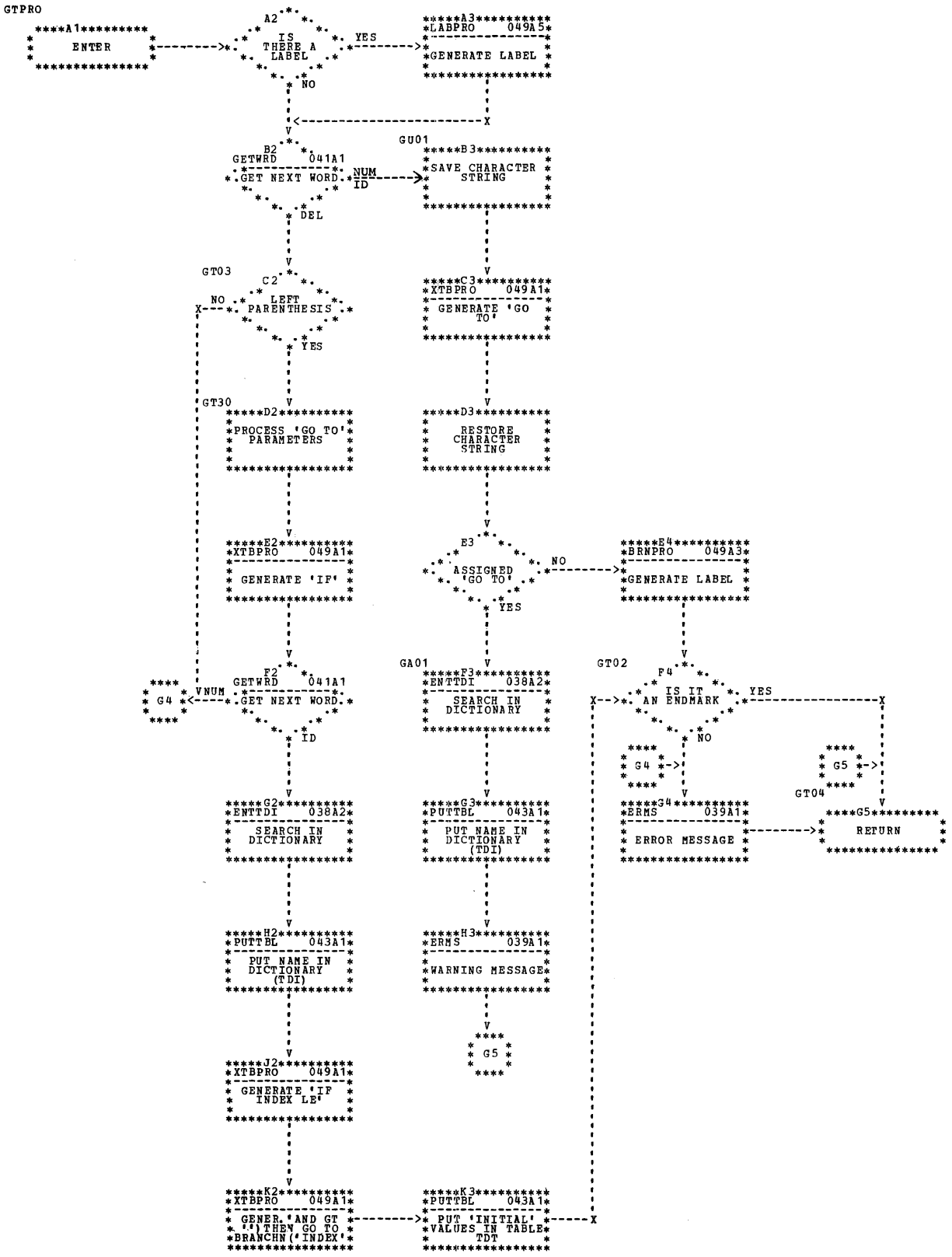
## Chart 025. DOPRO Routine



# Chart 026. FTPRO Routine



## Chart 027. GTPRO Routine



## Chart 028. IFPRO Routine

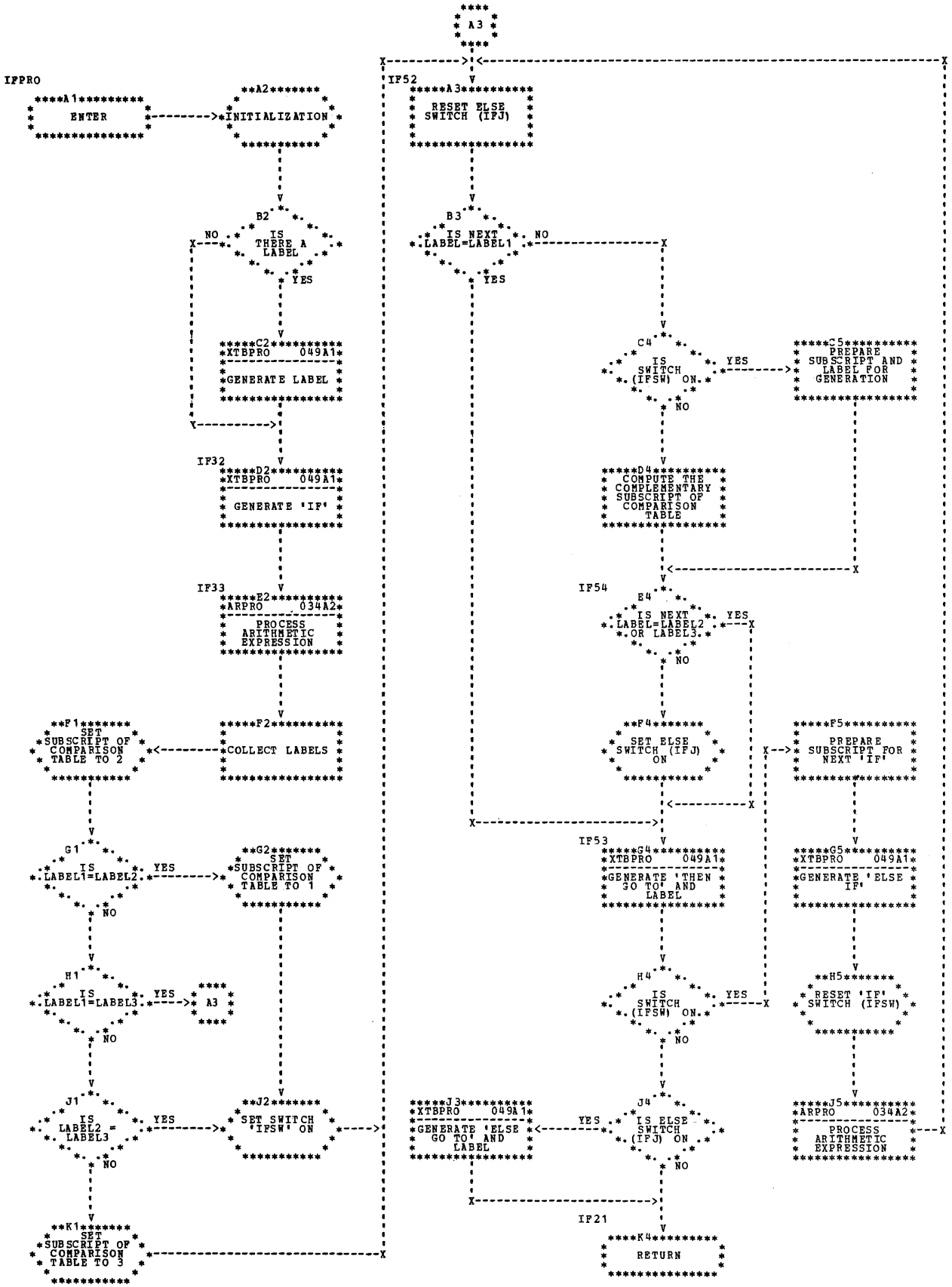


Chart 029. IOPRO Routine, Part 1 of 2

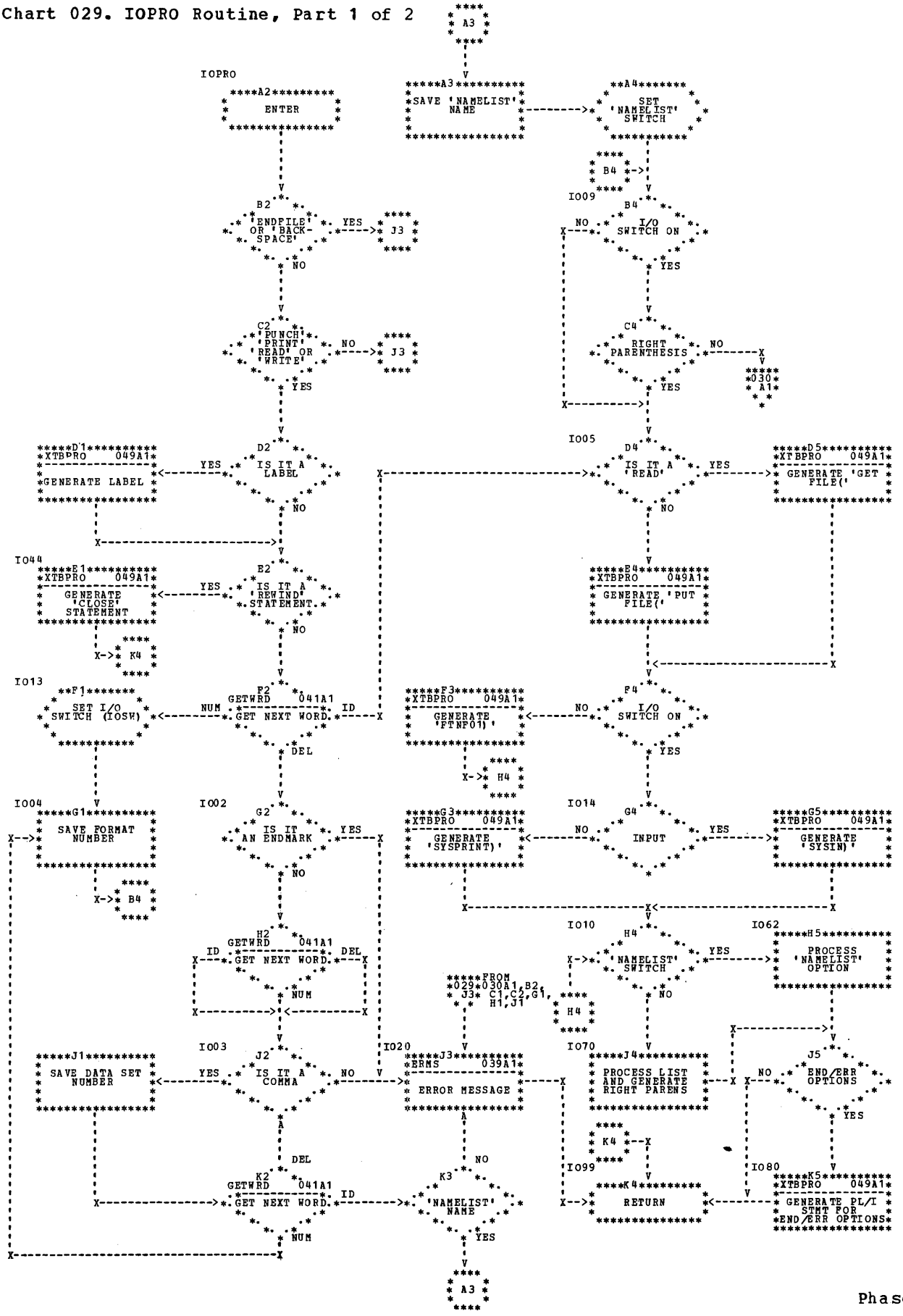


Chart 030. IOPRO Routine, Part 2 of 2

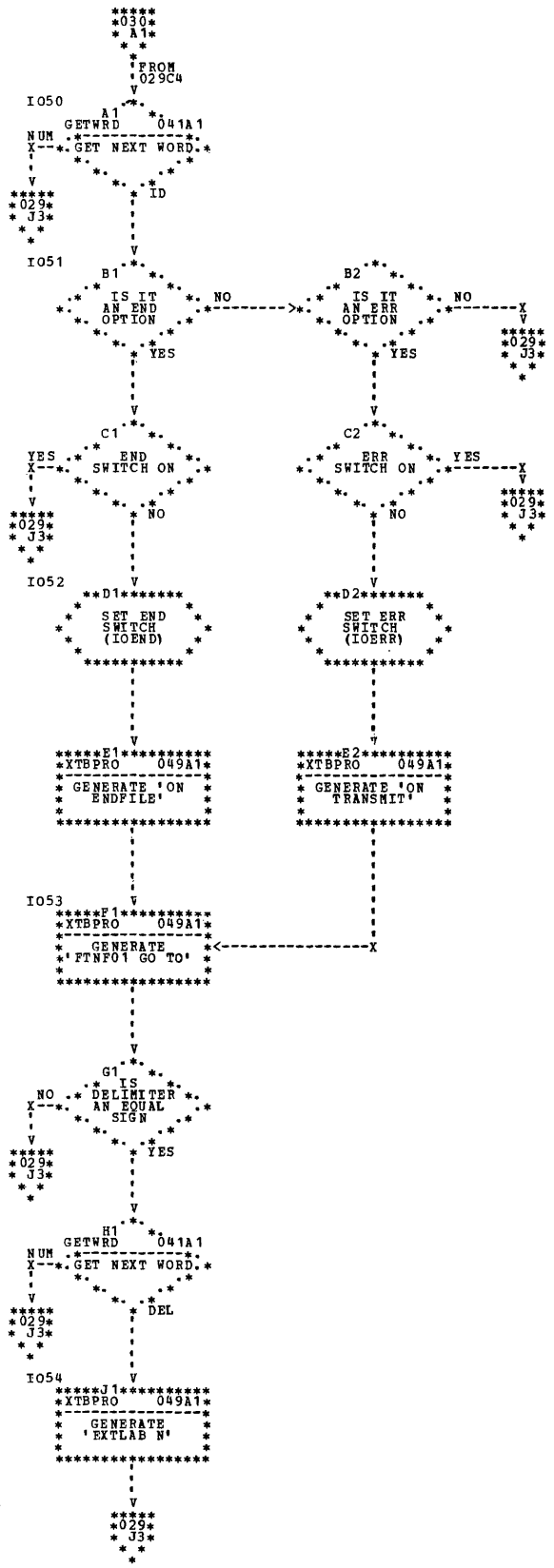


Chart 031. PSPRO Routine

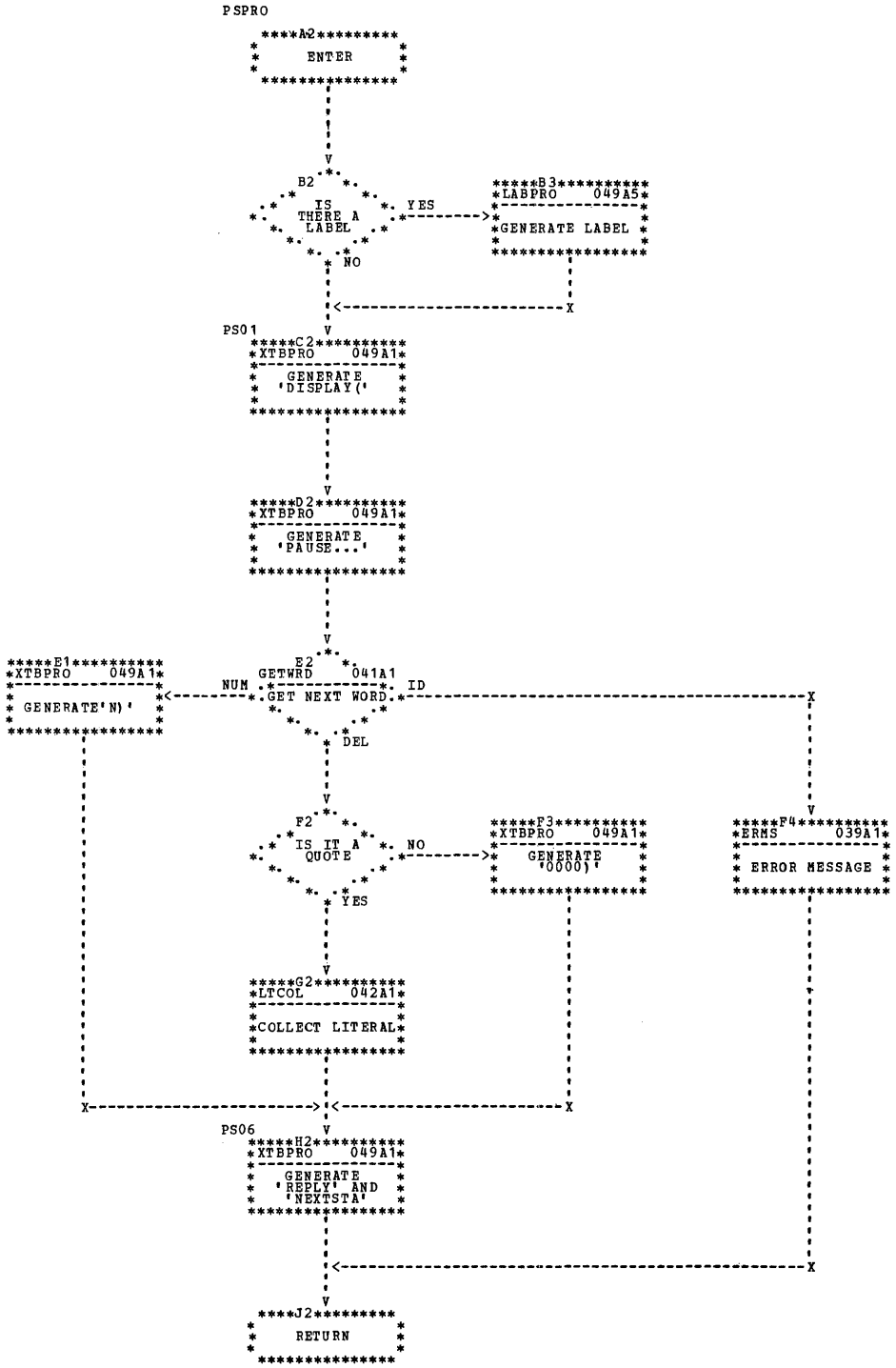
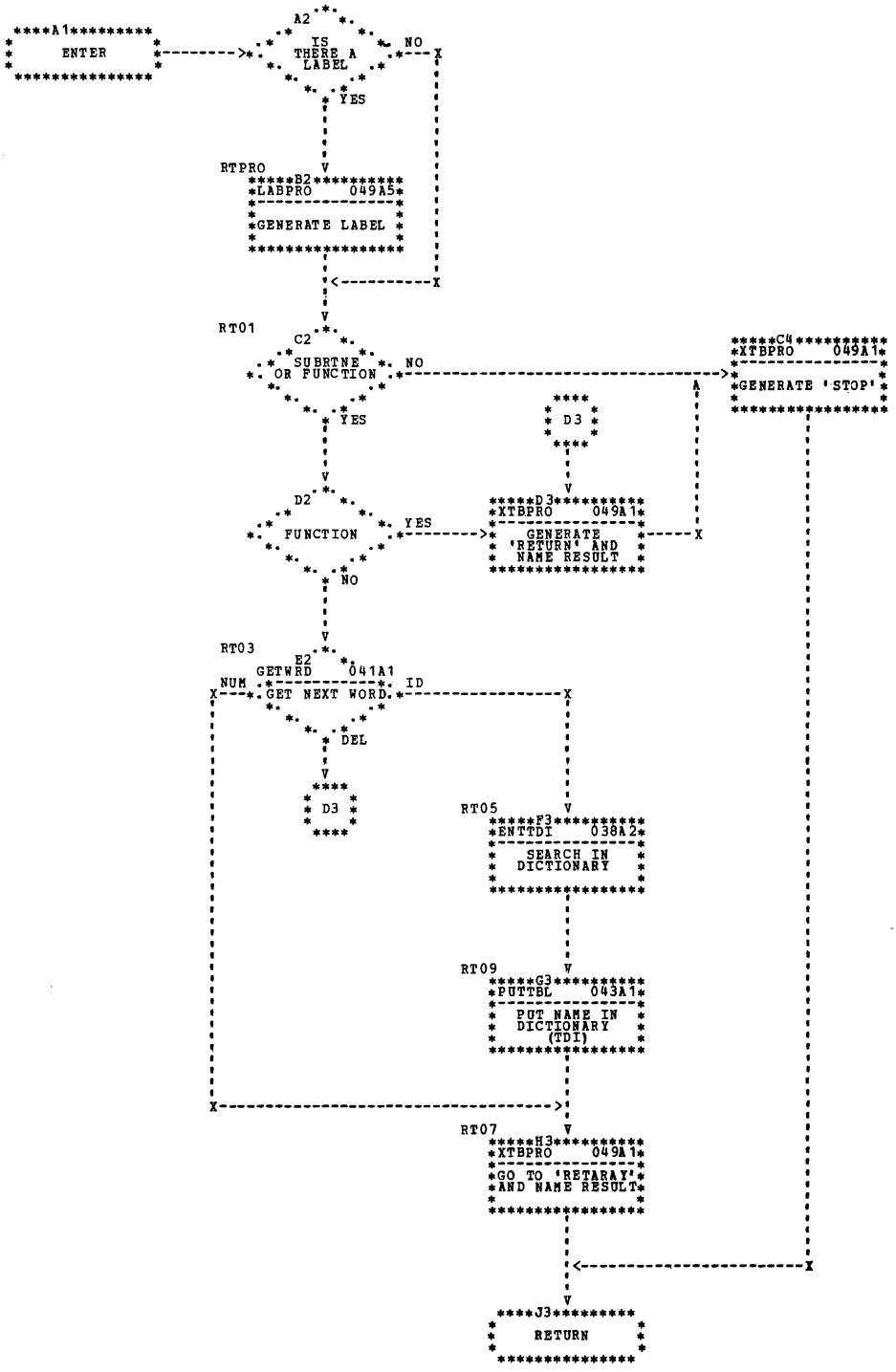


Chart 032. RTPRO Routine



## Chart 033. STPRO Routine

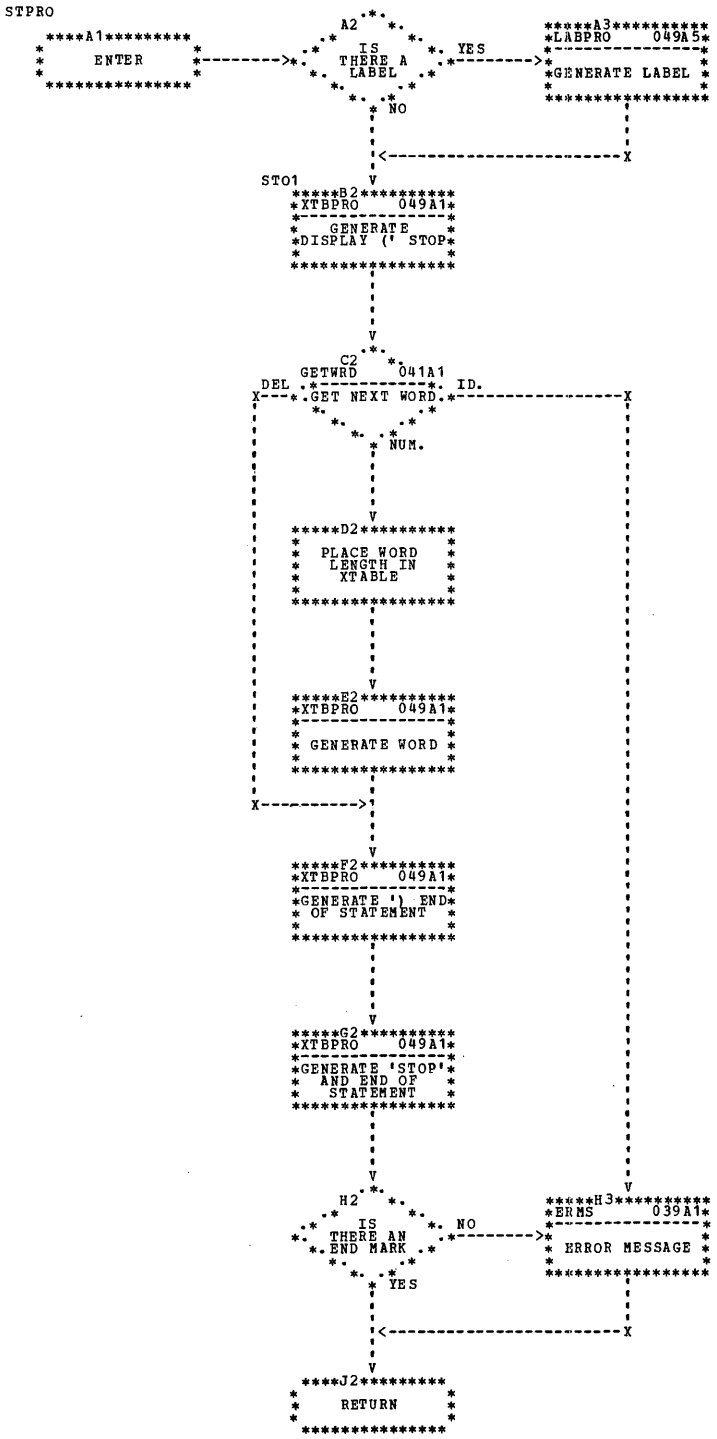


Chart 034. ARPRO Routine, Part 1 of 4

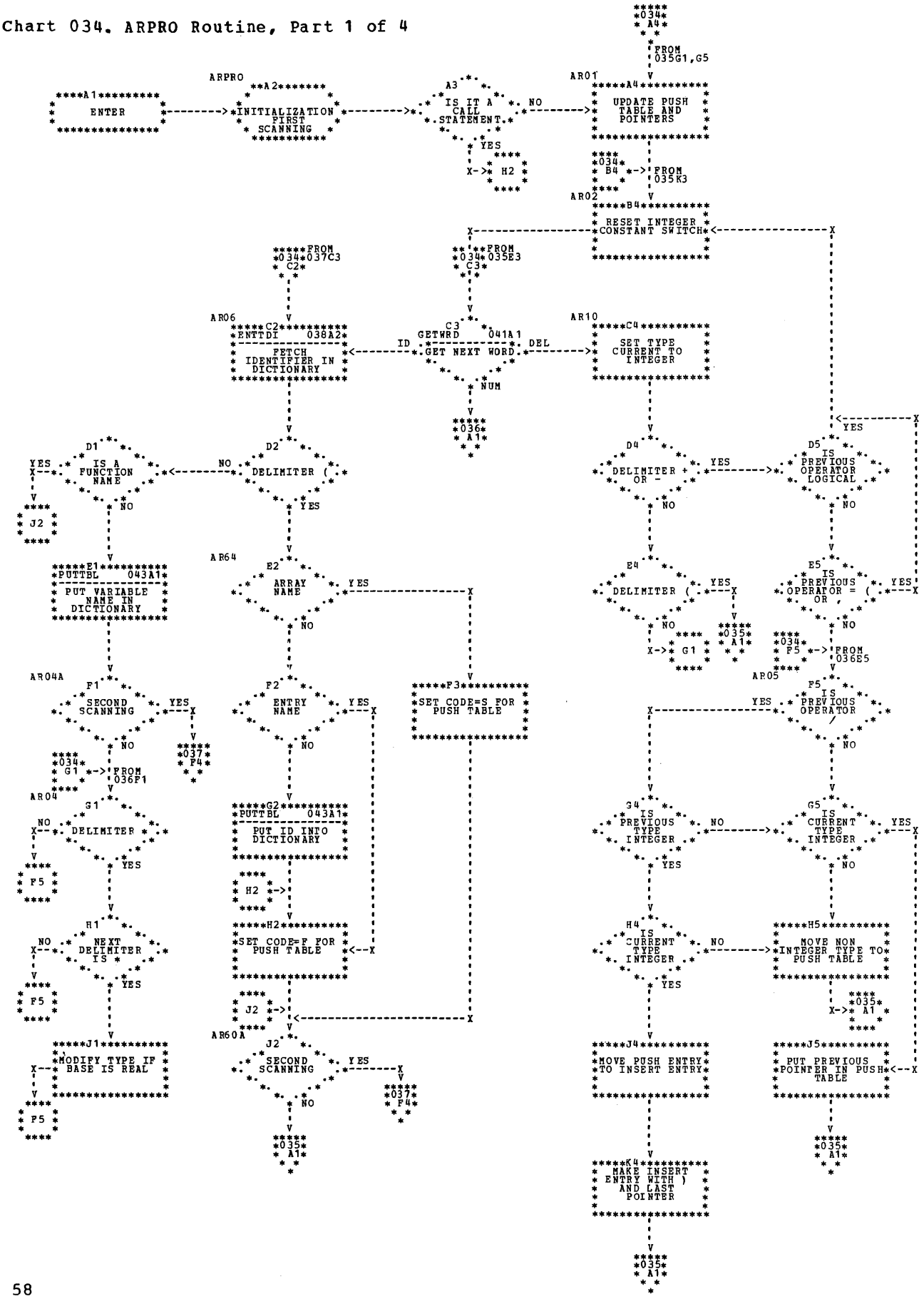
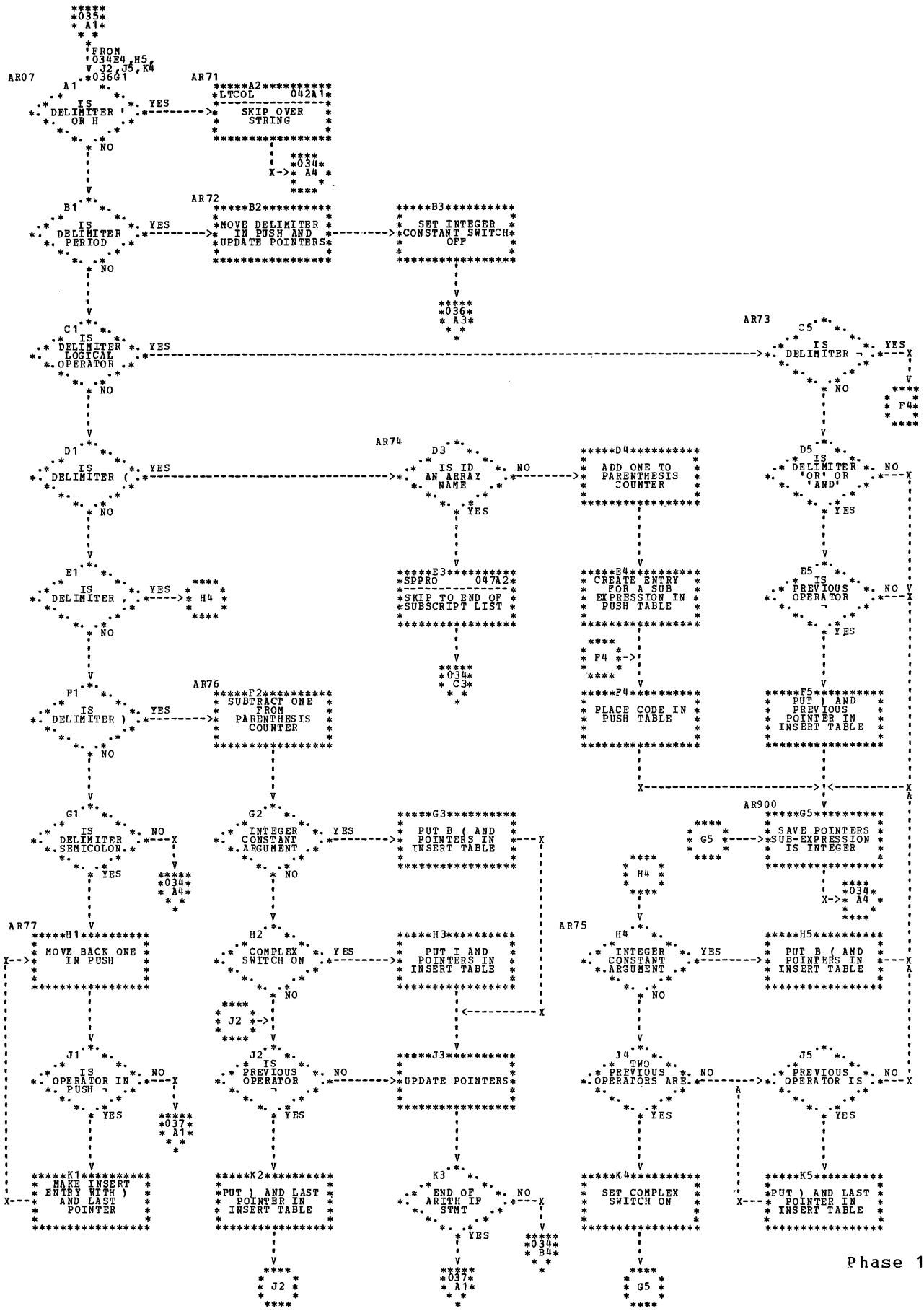
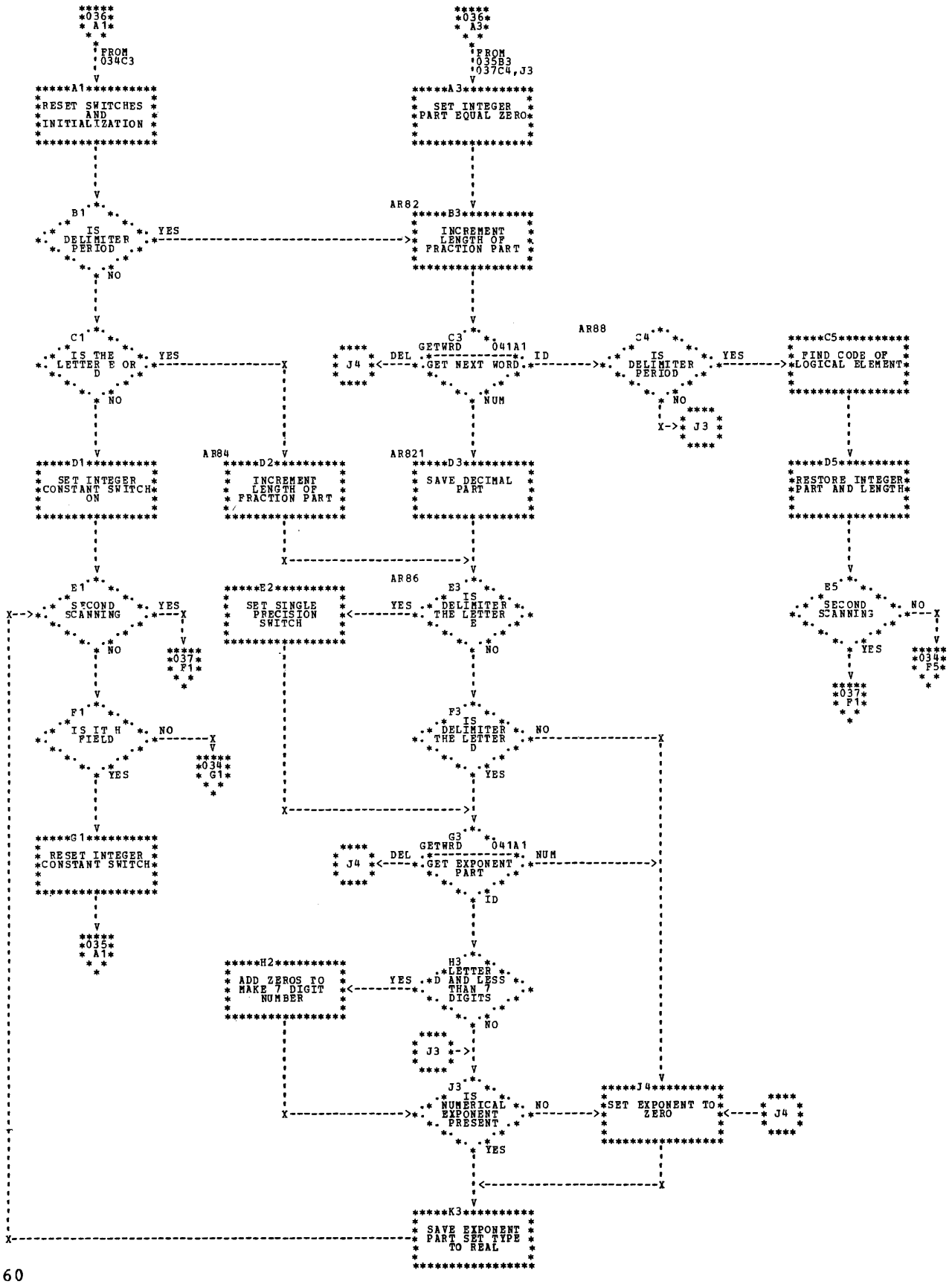


Chart 035. ARPRO Routine, Part 2 of 4



## Chart 036. ARPRO Routine 3 of 4



## Chart 037. ARPRO Routine, Part 4 of 4

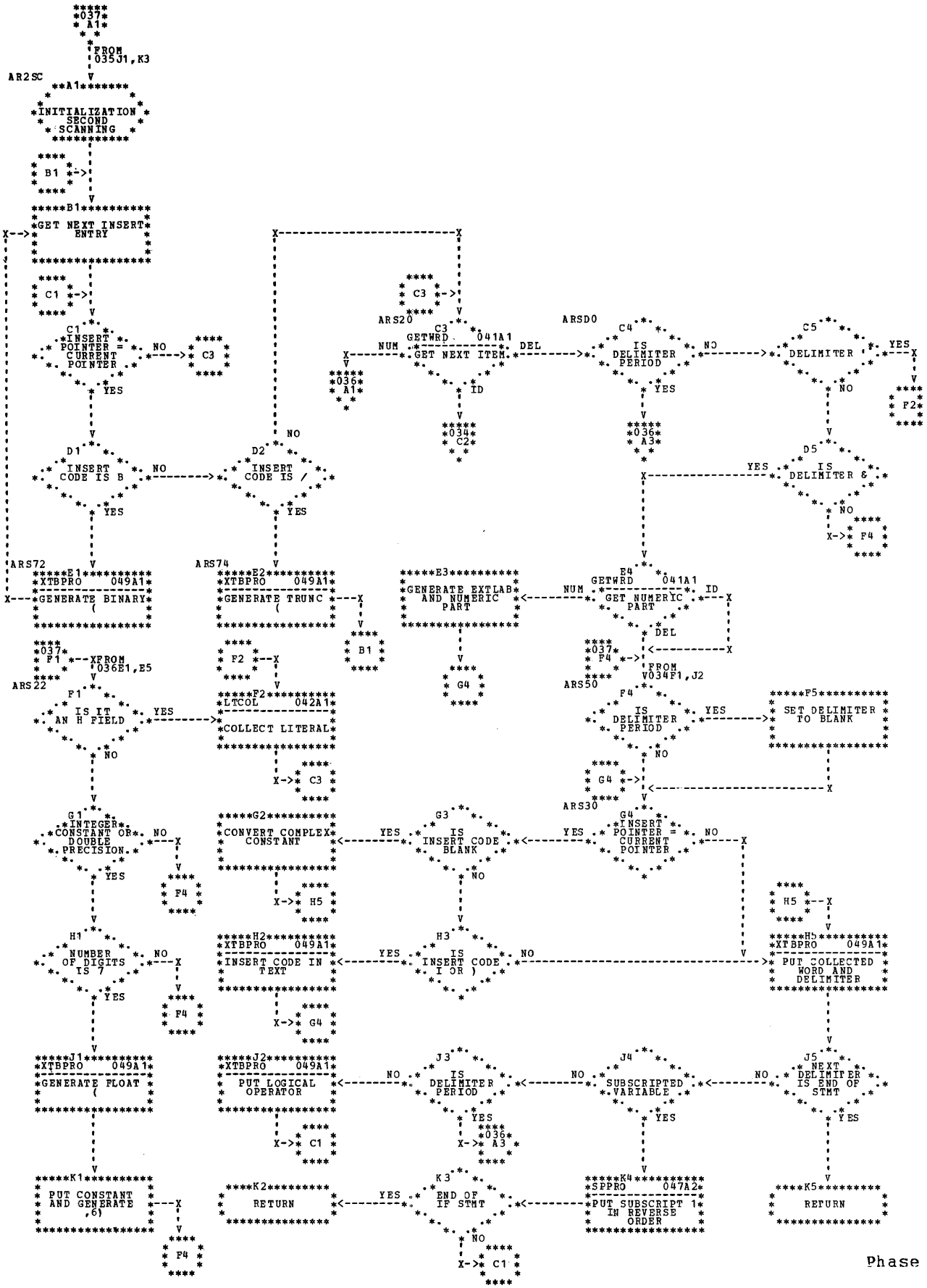
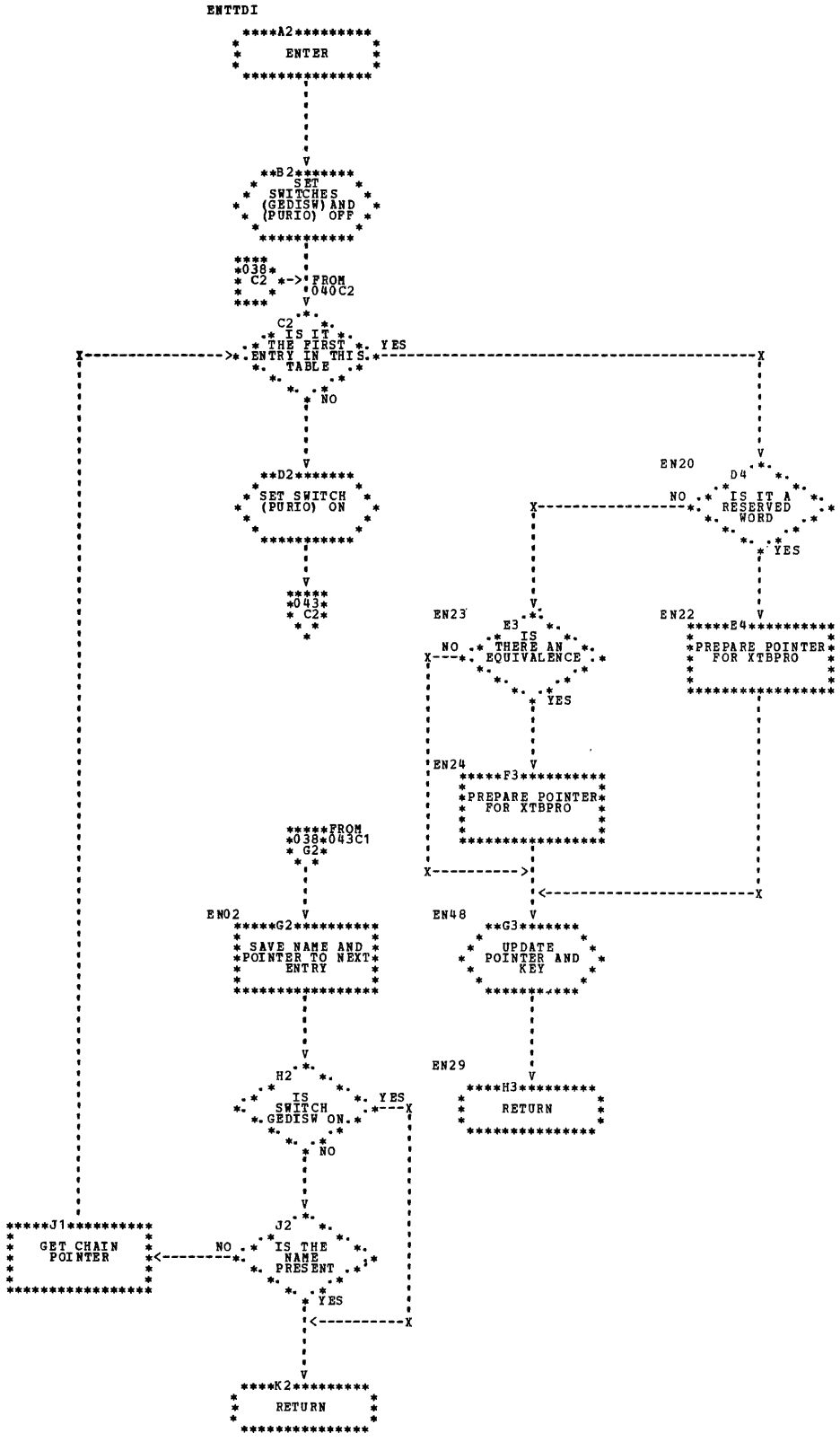
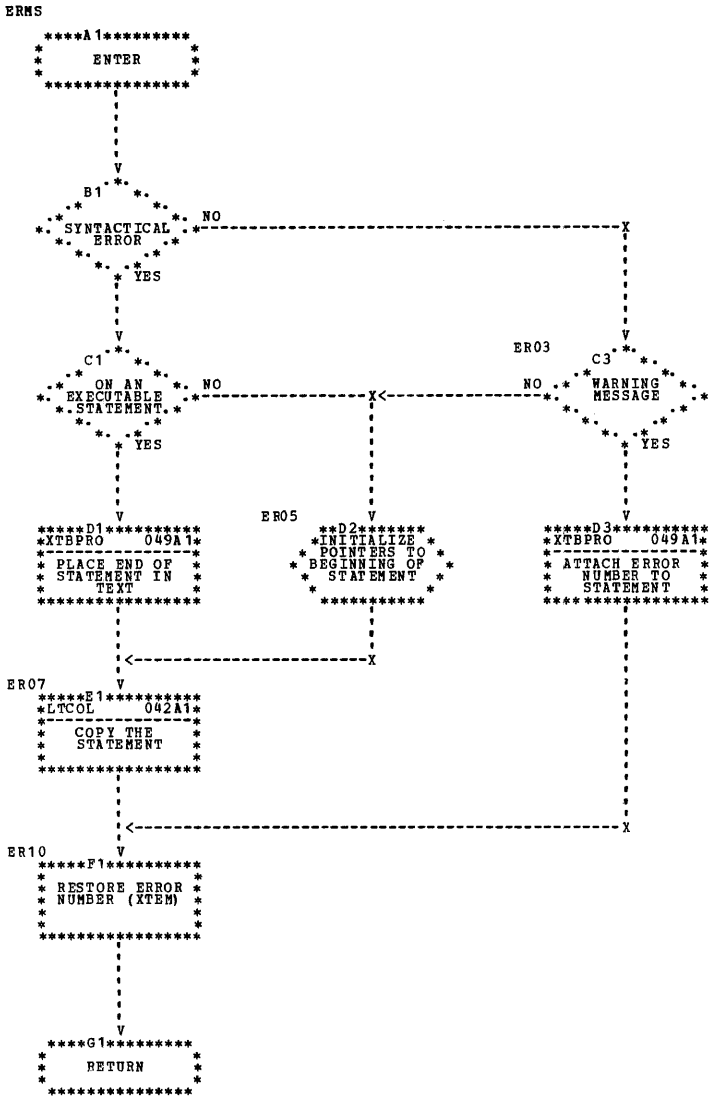


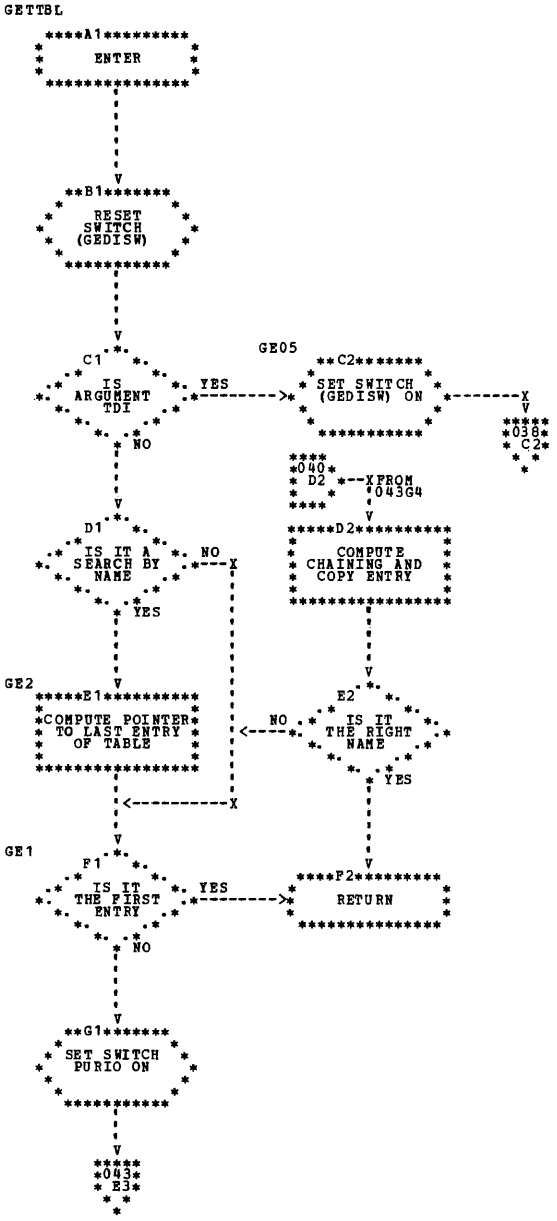
Chart 038. ENTTDI Routine



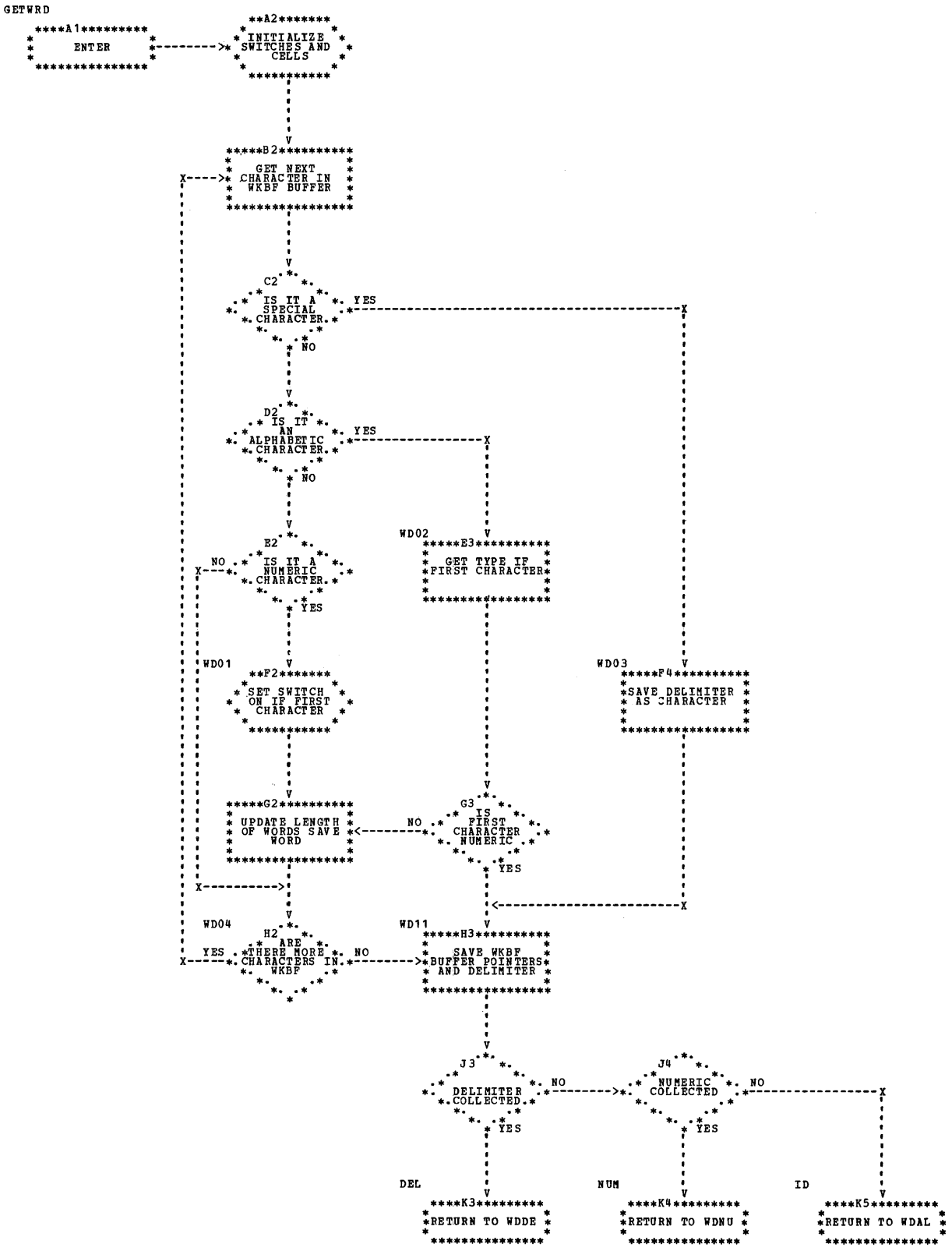
## Chart 039. ERMS Routine



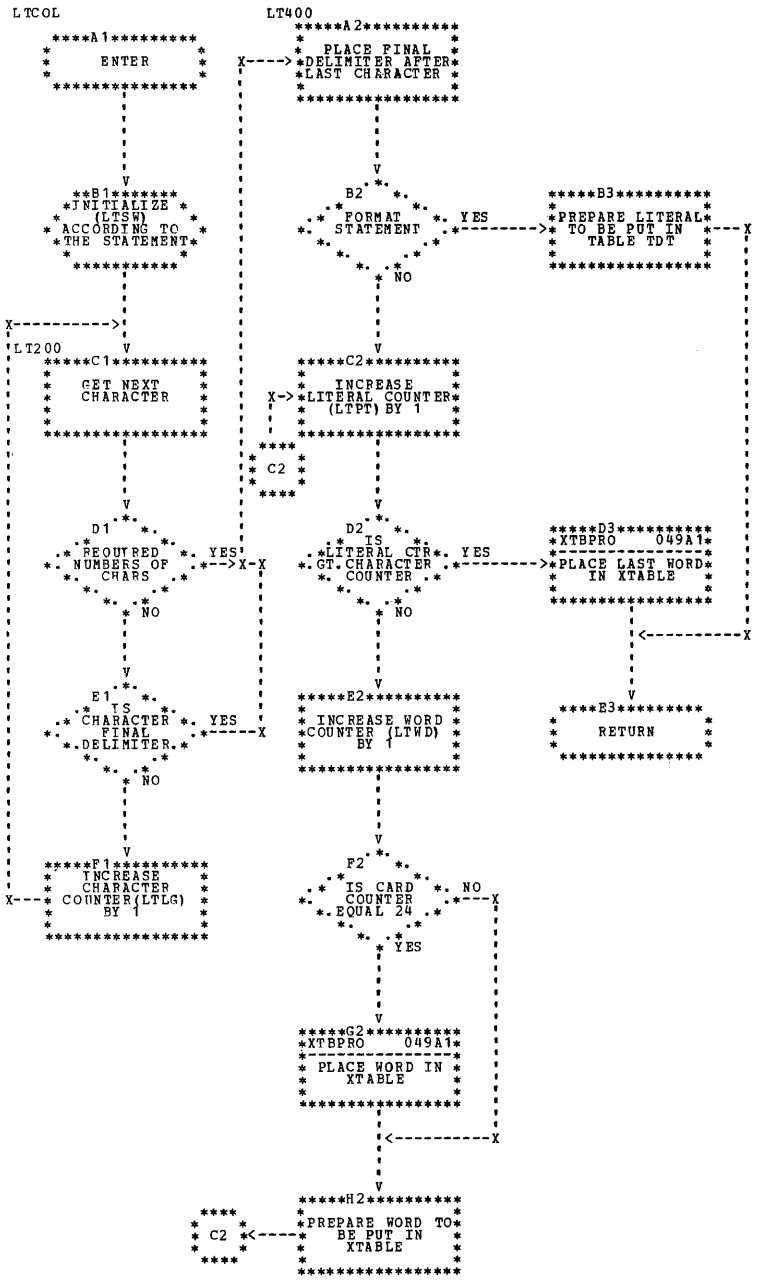
## Chart 040. GETTBL Routine



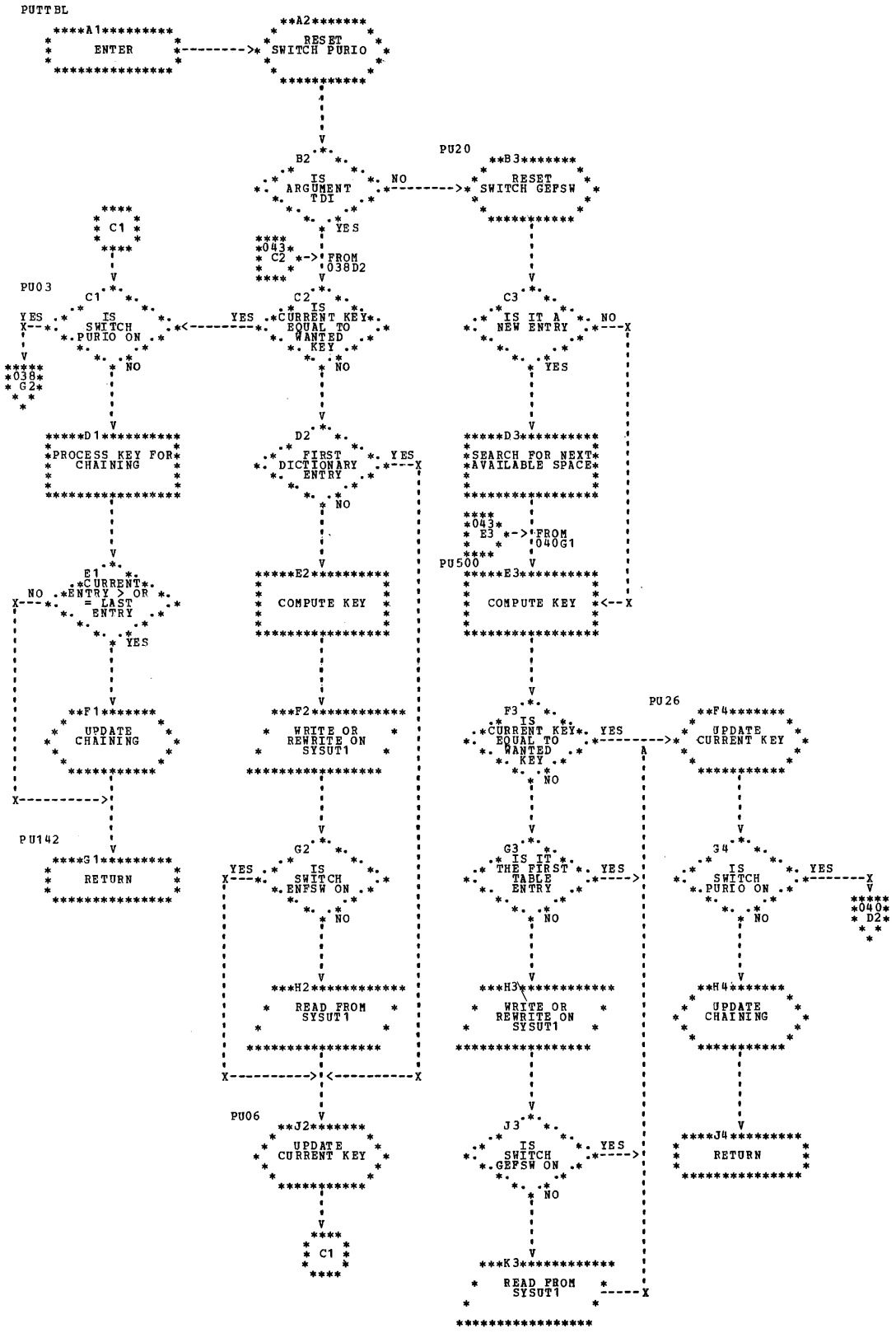
## Chart 041. GETWRD Routine



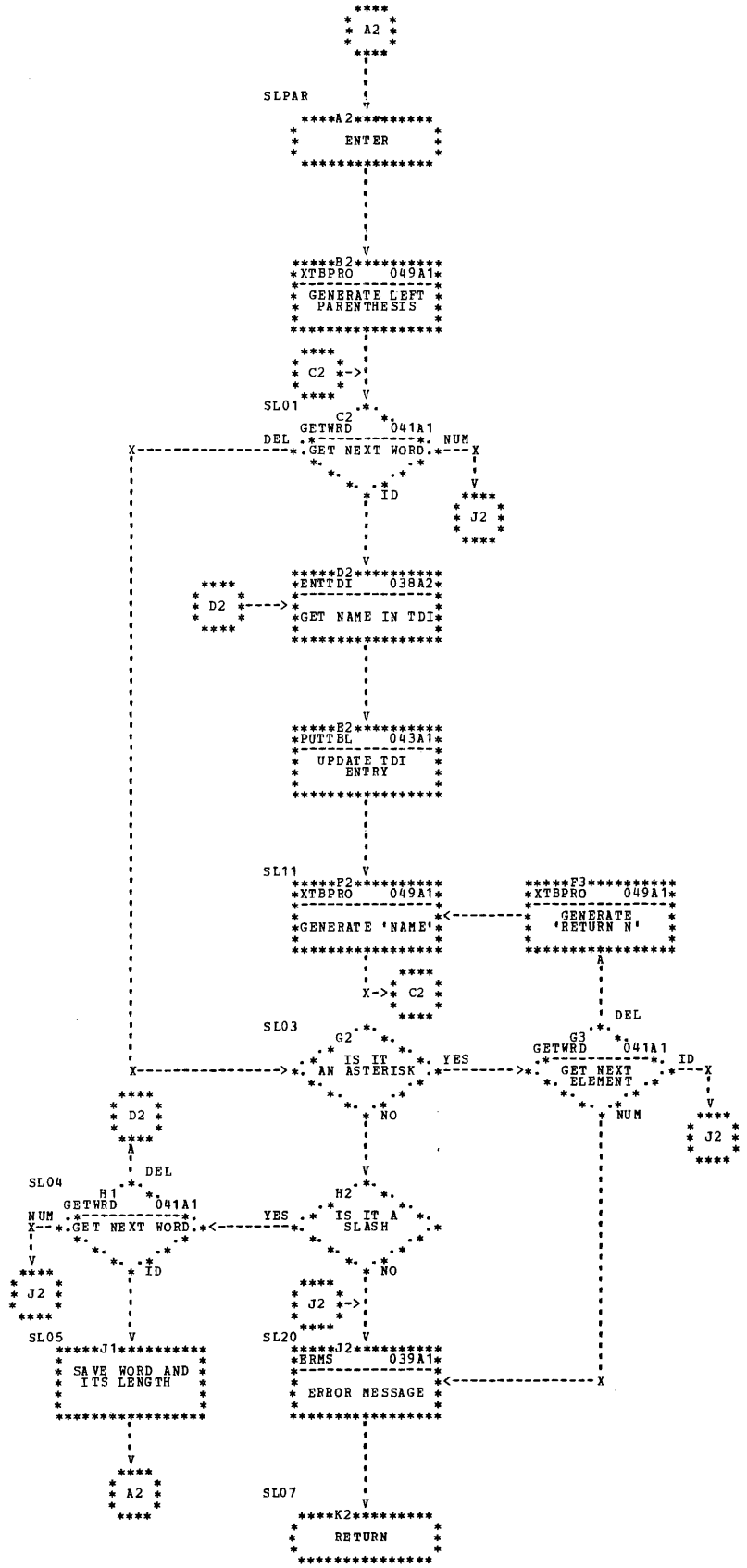
## Chart 042. LTCOL Routine



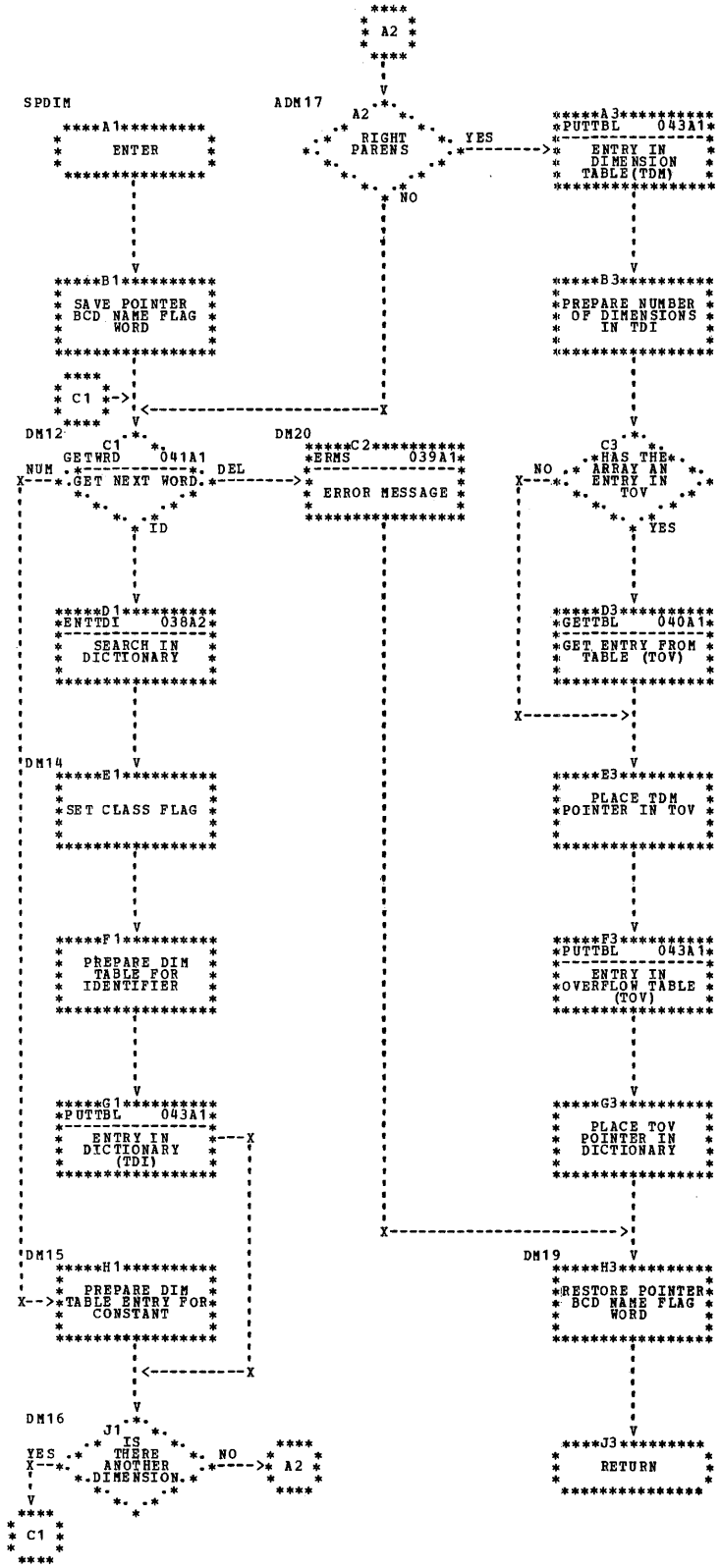
## Chart 043. PUTTBL Routine



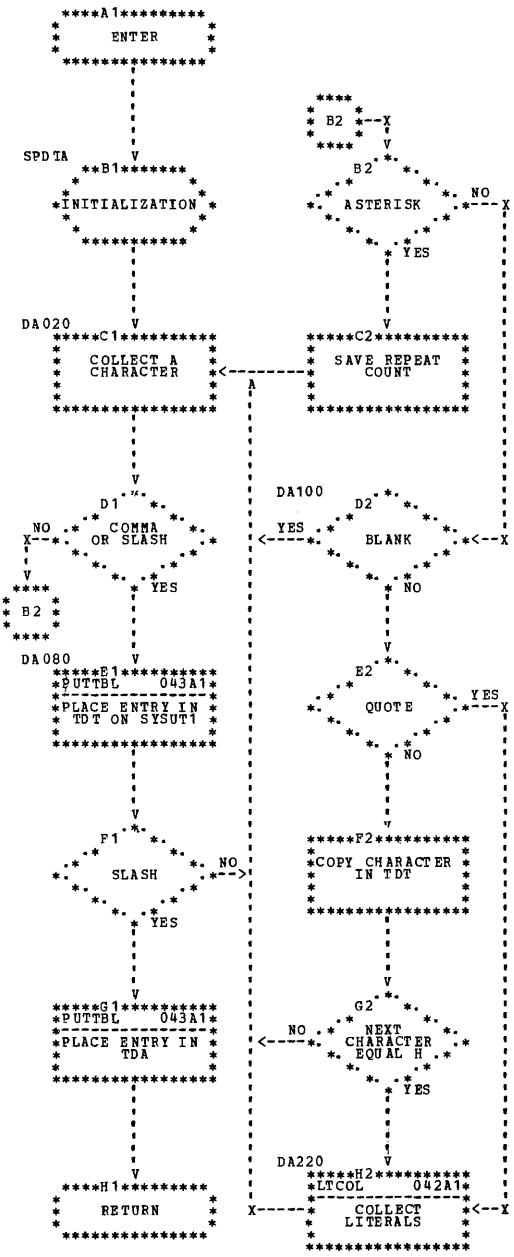
## Chart 044. SLPAR Routine



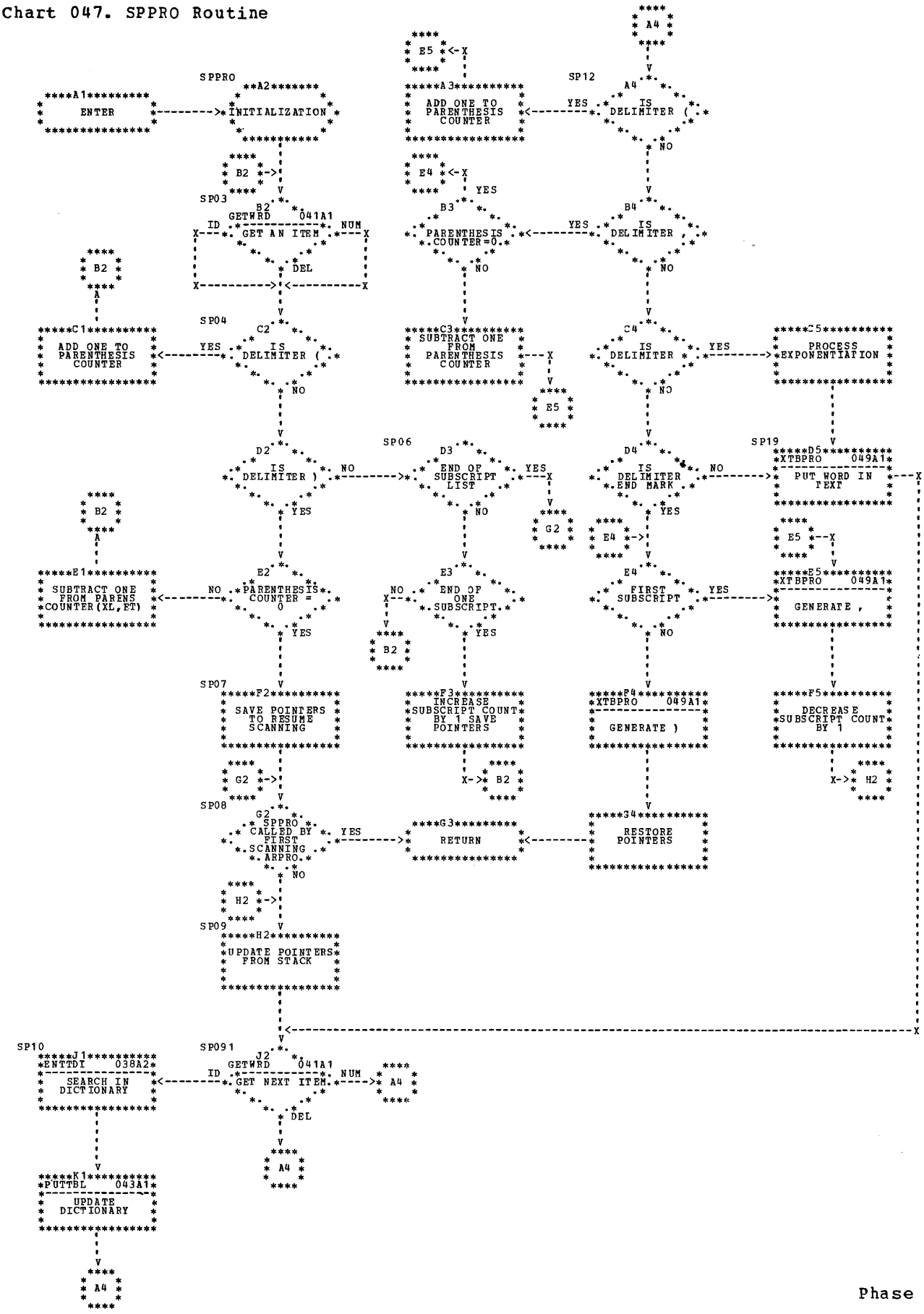
## Chart 045. SPDIM Routine



## Chart 046. SPDTA Routine



## Chart 047. SPPRO Routine



## Chart 048. SPSUB Routine

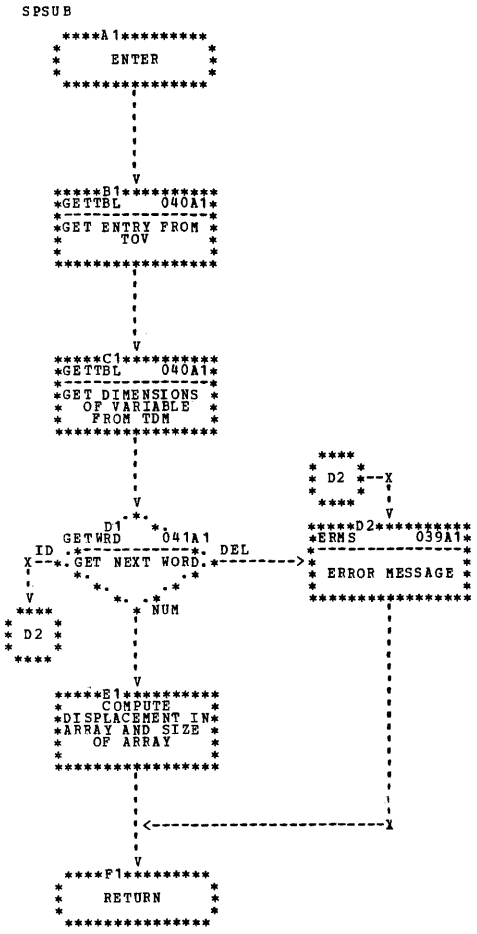
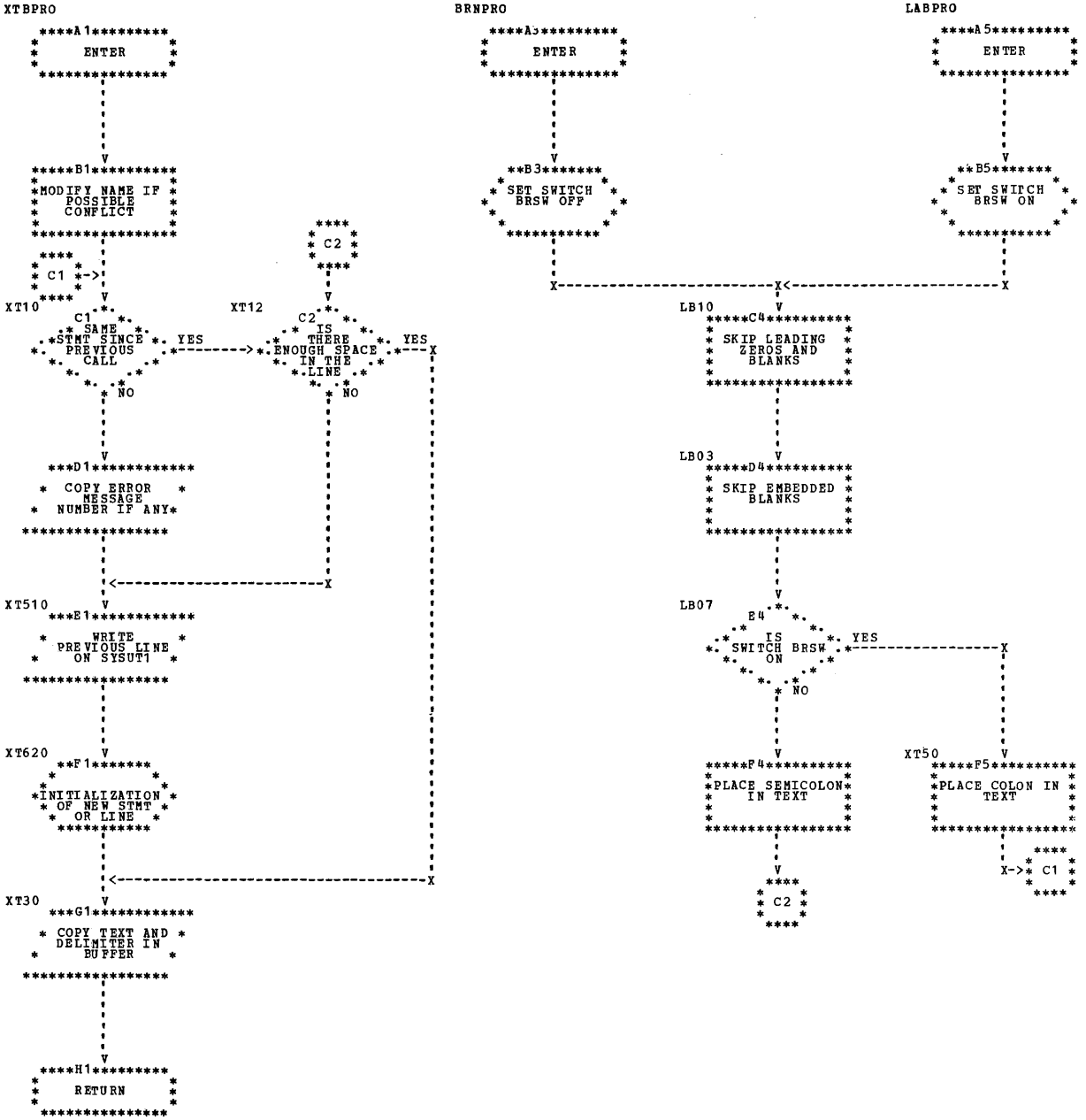


Chart 049. XTBPRO Routine



## PHASE 20

Phase 20 (see Chart 050) is called if the source program includes either of the two following types of statements:

1. EQUIVALENCE statements
2. Statements containing initial values

that is, if there are any entries in tables TEQ or TDA.

Phase 20 then rearranges the tables created during Phase 10 that are associated with these statements, in order to prepare the generation of the PL/I target program in Phase 30.

The input to Phase 20 consists of the dictionary (TDI), and its associated tables: TEQ, TDT, and TDA.

Names occurring in EQUIVALENCE statements are processed first. An EQUBLKj block is created,

Embracing all the elements of the equivalenced names (see the section "EQUIVALENCE Statement" in the language conversion manual). The size, type, and length information associated with each element of the block is saved in table TEH, and a reference to this table for each of the equivalenced names is entered into the overflow table (TOV). To this reference is added the relative position, within the embracing block, of the first element of the name. This procedure is repeated for each equivalence chain in the FORTRAN program.

Phase 20 then places the initial values in ascending numerical order corresponding to the order of the elements. If any elements within the sequence have not been assigned, Phase 20 takes into account the number of elements between two consecutively initialized elements of an array.

If the initial value is a string of characters, a flag is set on in the TDT entry which contains the string and the type of dictionary entry for this name is modified to CHARACTER.

Where initialized variables or arrays occur in an EQUIVALENCE statement, the converted variables cannot have the INITIAL attribute because of the method of conversion used by the LCP.

The output from Phase 20 is a modified dictionary (TDI, TDT, TEH). On completion of this phase, control returns the Control Phase.

### PHASE 20 ROUTINES

This section describes the two processing routines used by Phase 20: DTPRO and EVPRO. The two utility routines, GETTBL and PUTTBL, used by this phase have already been described in the section "Utility Routines."

#### DTPRO ---------- Chart 051

Purpose: To assign initial values to variables or to elements of arrays

Called by: PH20

Processing: The input to this routine consists of tables TDA and TDT. Variables included in DATA or TYPE statements are assigned a value determined by the literals appearing in these statements. The literals corresponding to each variable are chained together.

Routines called: GETTBL, PUTTBL.

Exit: Calling routine

#### EVPRO ---------- Chart 052

Purpose: To create EQUBLKs.

Called by: PH20

Processing: The input to this routine consists of tables TDI, TEQ, and TOV. All equivalenced variables are collected, and those which are included in the same EQUBLK are chained together. In addition, the size of each EQUBLK is computed, and the table TEH entry associated with each block is created. The chained variables are then collected and the overflow table (TOV) for each variable is updated.

Routines called: GETTBL, PUTTBL.

Exit: Calling routine.

Chart 050. Overall Logic of Phase 20

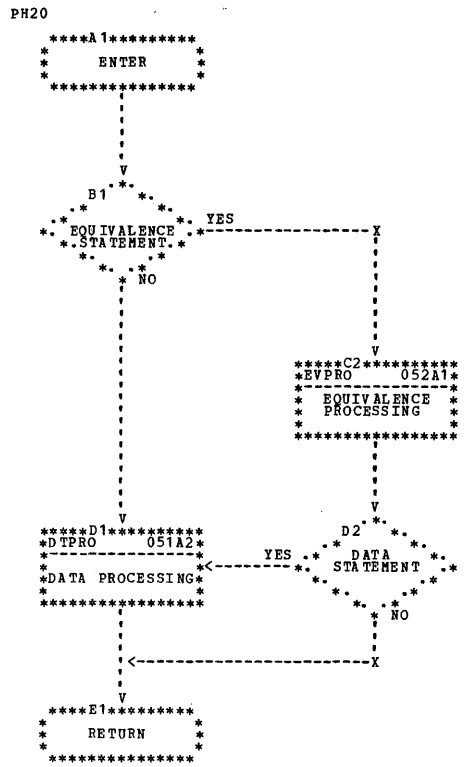


Chart 051. DTPRO Routine

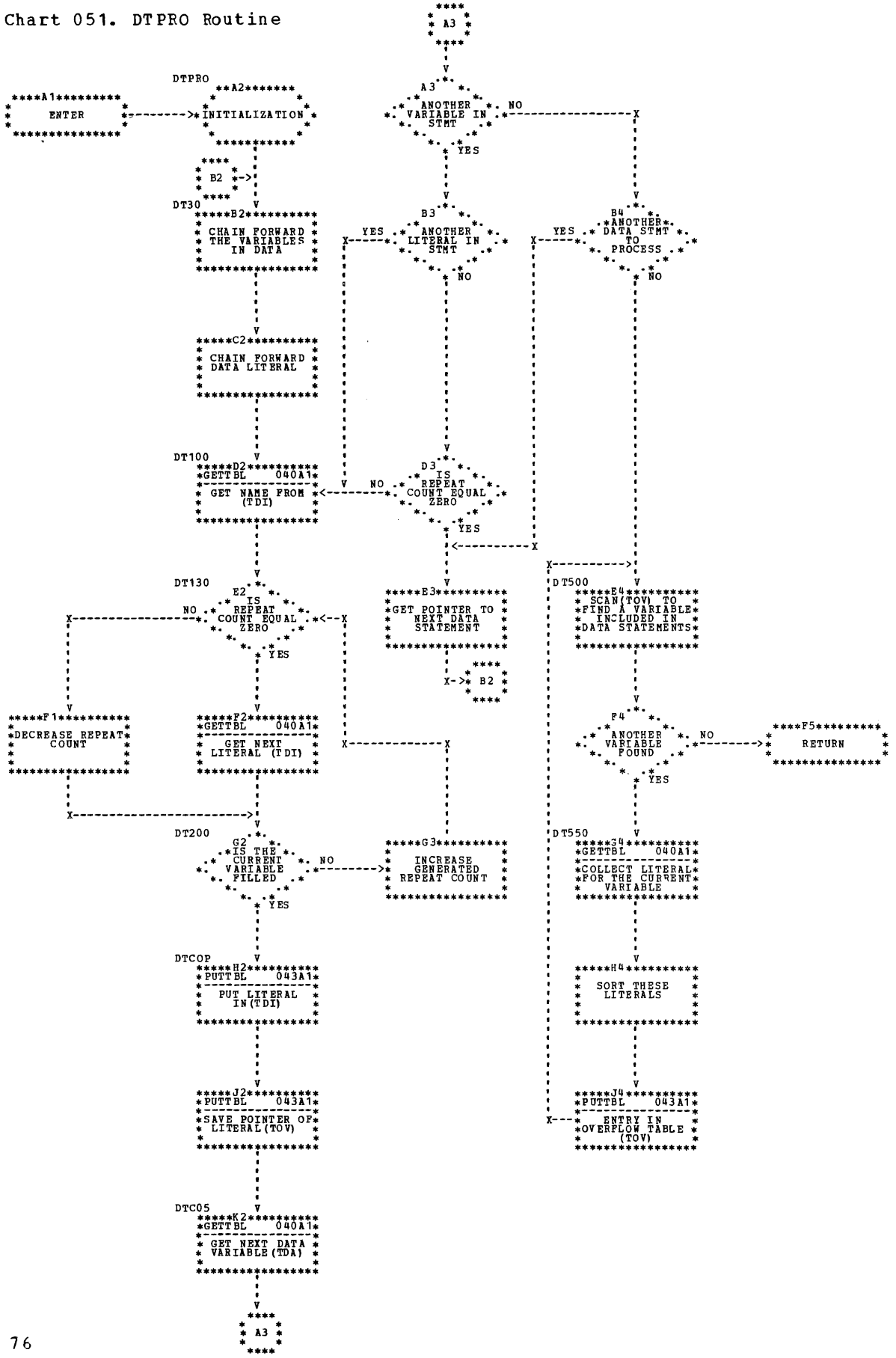
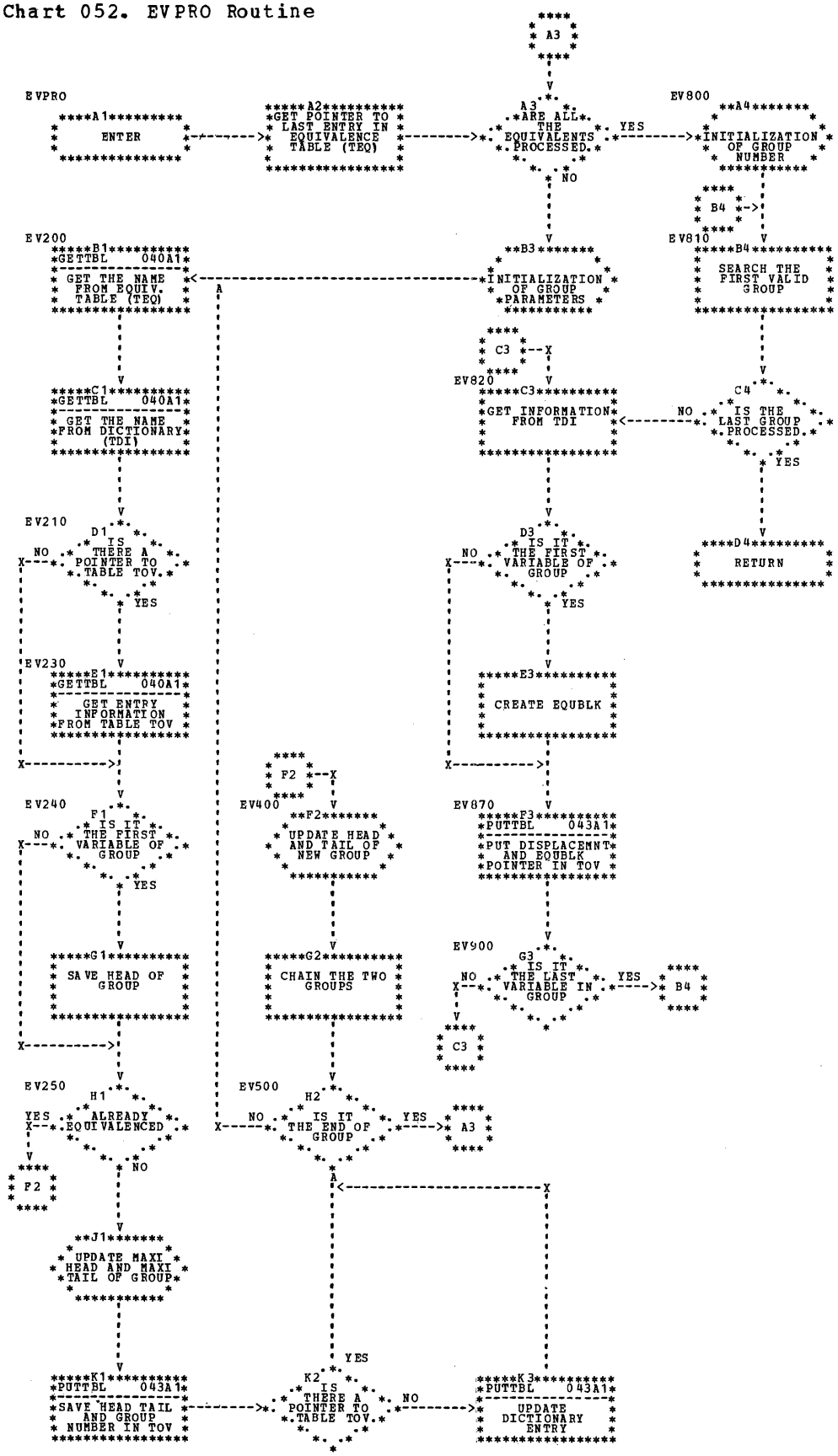


Chart 052. EVPRO Routine



Phase 30 generates the PL/I target program, and all comments and messages. It uses two utility routines: UPRNT, which is described in detail later in this section, and GETTBL.

This phase draws its input from SYSUT1 and from the dictionary (TDI) and the tables associated with it.

After initialization of all phase parameters , Phase 30 generates a PL/I PROCEDURE statement. The label of this statement is either that of a FORTRAN subprogram or the LCP name MAINPRO in the case of a main program.

Phase 30 then processes identifiers that appear in COMMON statements. The elements of a labeled or unlabeled block, together with their attributes, are declared as minor structures at level 2. The major structure is declared EXTERNAL; it is identified by the COMMON block name (TBK) if the block was labeled by the programmer; if not, it is identified by the LCP name UNLABCM.

All blocks created in Phase 20 are then placed in a DECLARE statement; these embrace EQUIVALENCE blocks when they do not include a variable which is part of a COMMON block.

Next, Phase 30 examines the dictionary, generating DECLARE statements for all identifiers that have not appeared in COMMON statements and which must be explicitly declared: variables of a specified type, array names, procedure names, labels, equivalenced items, and initialized elements.

The equivalenced items are declared in two-level structures, where the first element of level-2 is a dummy variable giving the displacement of the variable in front of the EQUIVALENCE block (or the COMMON block if this item is equivalenced with a variable appearing in COMMON) and the second element of level-2 is the equivalenced item.

- When the type of a variable is found to be CHARACTER in the dictionary (TDI) entry, the variable is declared with the CHARACTER attribute.
- If a TDU table exists, each LCP-created variable is declared with the CHARACTER attribute. Then, the string of characters saved in table TDT and associated with it is placed after the INITIAL attribute.
- Print files are declared according to the information saved by the LCP in the initialization procedure.

Executable statements are generated next; they are checked for being a PL/I internal procedure (converted statement function). When one is encountered, the attributes of each parameter are generated in a DECLARE statement, using pointers in PL/I text and dictionary.

Finally, the error messages, if any, are printed on SYSPRINT.

The output from Phase 30 is in the form specified by the user in his EXEC control card, either in printed form on SYSPRNT or in punched-card form on SYSPCH.

On completion of Phase 30, control returns to the Control Phase.

The overall flow of Phase 30 is illustrated in Charts 053 and 054.

## PHASE 30 ROUTINES

This section describes the utility routine UPRNT, which is used by Phase 30 only. Phase 30 also makes use of the utility routine GETTBL, which is described in the section " Utility Routines."

### UPRNT ---------- Chart 055

Purpose: To generate declarative statements.

Called by: PH30

Processing: The input to this routine is a character-string in the area WORD. A line of buffer is initialized. Each time a line in the buffer is full, this line is placed on SYSPRINT and the next line is initialized. The output from this routine is the PL/I statement in the buffer.

Routines called: None.

Exit: Calling routine.

# Chart 053. Overall Logic of Phase 30, Part 1 of 2

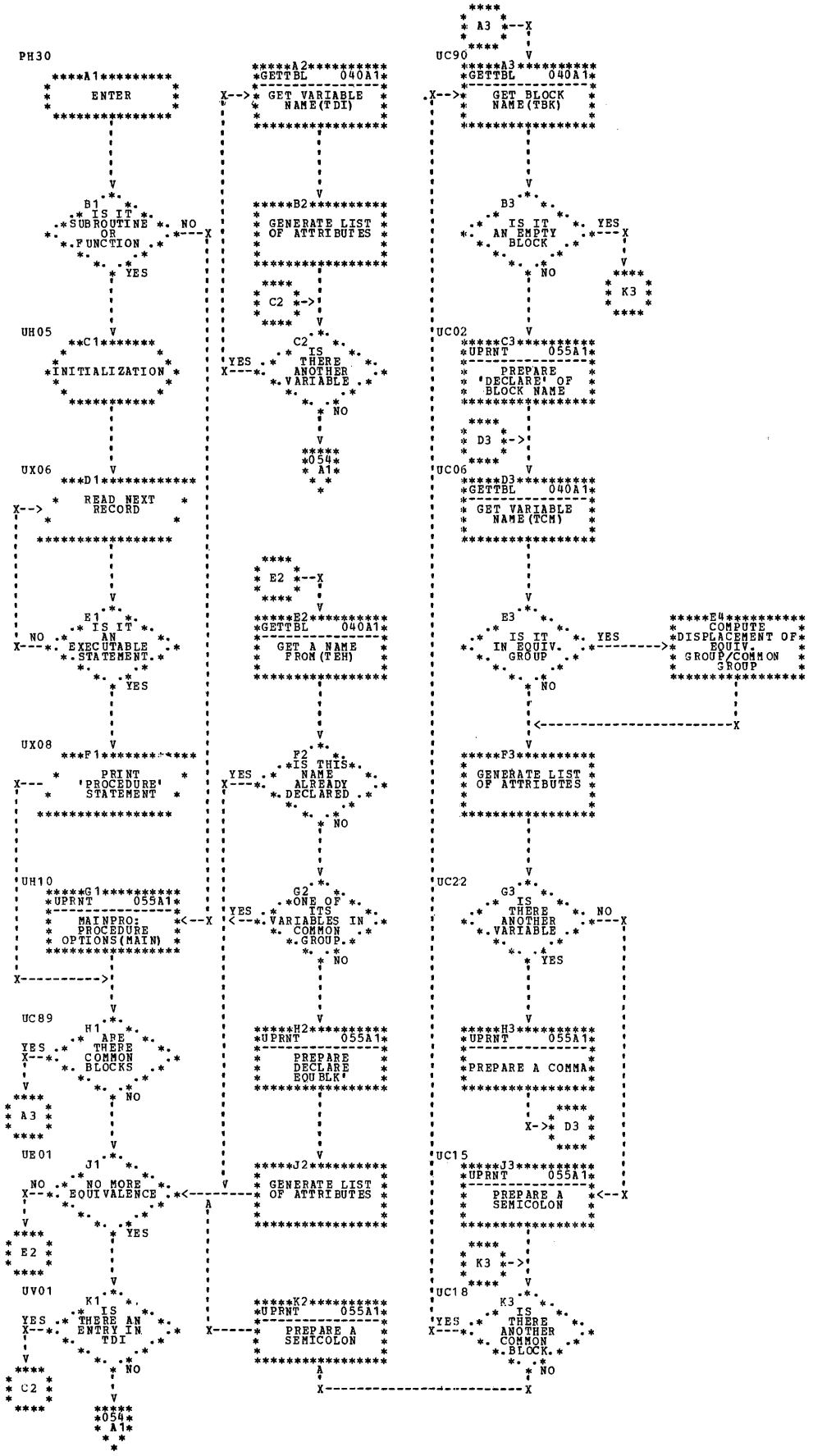
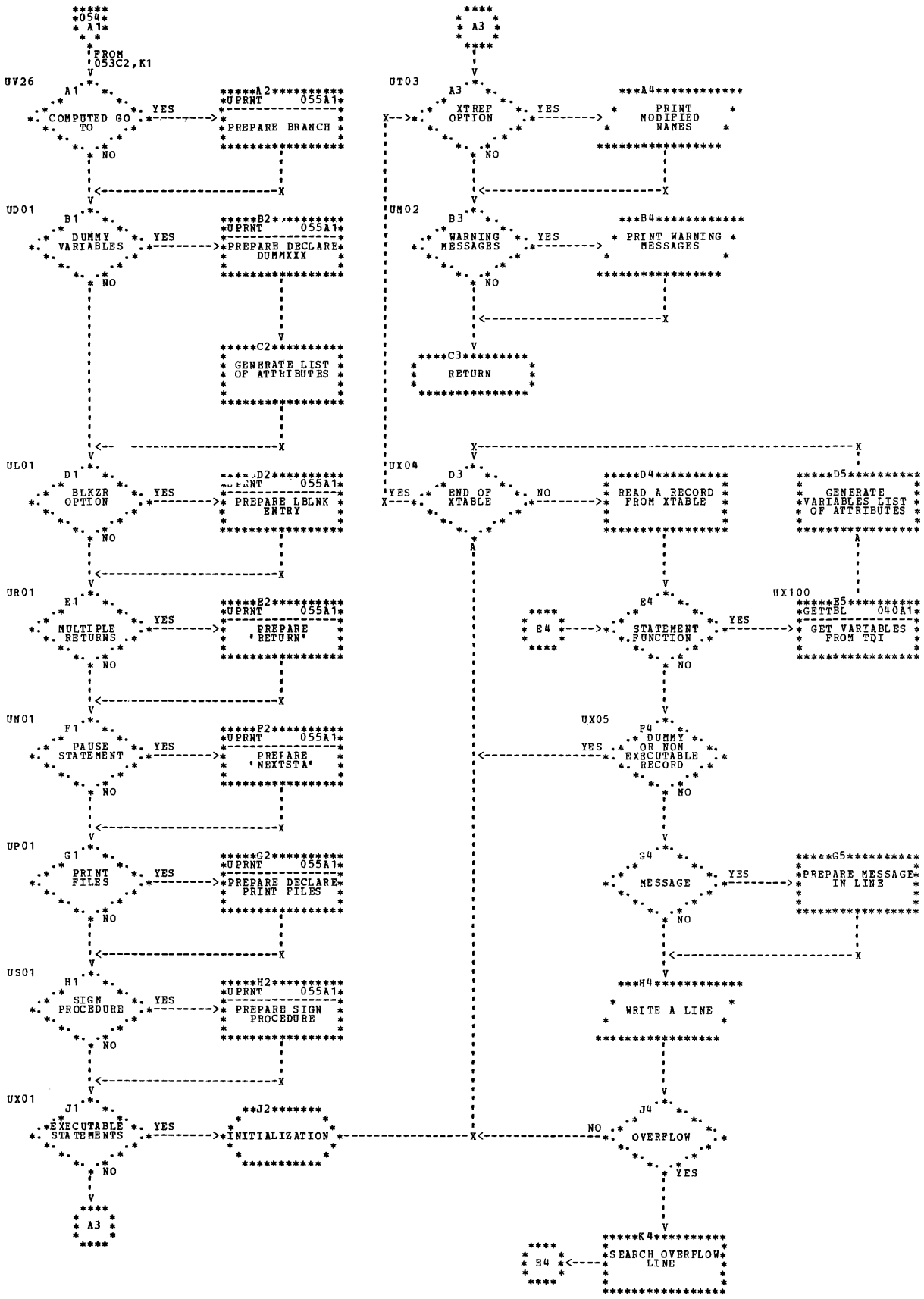
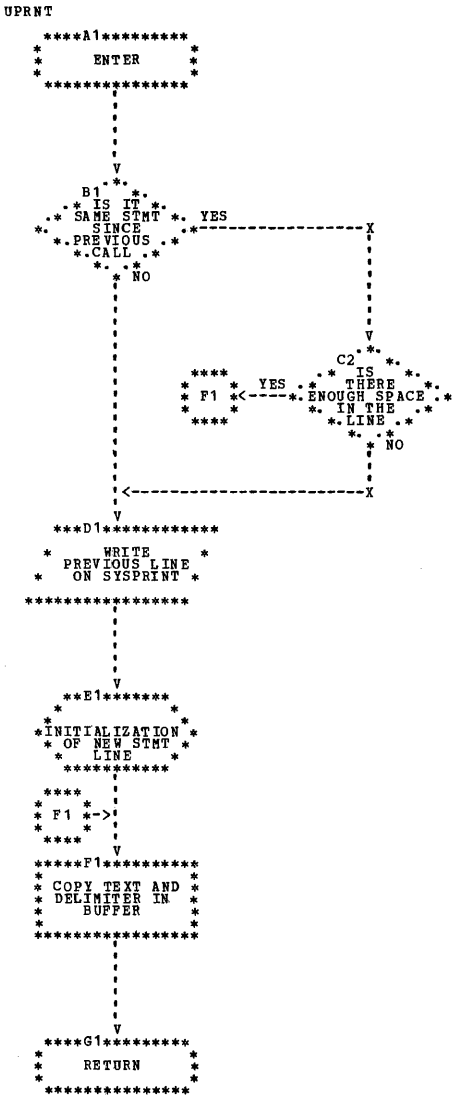


Chart 054. Overall Logic of Phase 30, Part 2 of 2



## Chart 055. UPRNT Routine



## APPENDIX A. TABLES

This appendix lists and describes all the tables used by the LCP.

### GENERAL CONSIDERATIONS

Required information is fetched from FORTRAN declarative statements and stored in various tables. This information is then used in the conversion of executable statements. Certain elements of the executable statements are also stored in tables for the same purpose. In Phase 30, the PL/I DECLARE statements are generated using these tables. The length of each element of a table is shown in the listing.

All entries for the tables are placed in an area of main storage; entries within the same table are chained. When the area of main storage assigned to these entries is full, the contents of that area are placed on SYSUT2, and the area is then used for new entries. There is therefore no limitation in the size of these tables.

The tables used by the LCP fall into one of two categories:

1. The dictionary (Table TDI).
2. Specification tables. This category includes all the other tables used by the program, namely: Tables TBK, TCM, TDA, TDM, TDT, TDU, TEH, TEQ, TFM, TIM, TNL, TNV, TOV, TPD.

### THE DICTIONARY (TABLE TDI)

The dictionary contains all the names used in the LCP and the class, type, and number of dimensions associated with each name. Names with the same number of characters are chained together.

Table Overflow: If the name has dimensions, or initial values, or appears in an EQUIVALENCE statement, the entry in the dictionary is extended by creating an entry in the overflow table (TOV). A pointer to this entry is then placed with the name in the dictionary. A description of Table TOV appears in the section "Specification Tables."

The layout of each entry in the dictionary is as follows:

| | |
|---|---|
| DINAM | BCD Name |
| DILT | Number of Characters |
| DICL | Class |
| DITY | Type |
| DIET | Explicit Type |
| DILG | Length |
| DIAU | Storage Class |
| DIMO | Modify |
| DIAR | Array |
| DIOV | Overflow |

DINAM
: contains the symbol given to the name, or the sequence number if the name is generated.

DILT
: contains the number of characters in the name.

DICL
: contains one of the following constants:

  LCVAR if DINAM is a variable
  LCLAB if it is a label
  LCENT if it is an entry name

DITY
: contains one of the following constants:

  LCTI if the type of DINAM is integer
  LCTR if it is real
  LCTC if it is complex
  LCTL if it is logical
  LCTA if it is a character string

DIET
: contains the constant LCET when DINAM is explicitly declared.

DILG
: contains, according to the length in bytes of DINAM:

  LCLO for a length of 1 byte
  LCLT for a length of 2 bytes
  LCLF for a length of 4 bytes
  LCLE for a length of 8 bytes
  LCLS for a length of 16 bytes

DIMO
contains one of the following constants:

LCSUF if DINAM is variable or array name to which a suffix must be added.
LCEQU if DINAM is a reserved word which requires a PL/I equivalent.
Zero Otherwise

DIAU
contains constant LCAU if DINAM is used as a parameter.

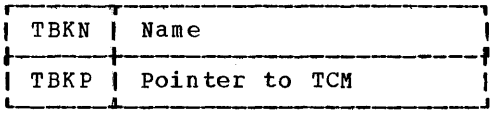
DIAR
contains the number of dimensions if DINAM is an array name.

DIOV
contains the overflow table pointer if DINAM appears in a DIMENSION, DATA, or EQUIVALENCE statement.

## SPECIFICATION TABLES

This section describes the layout of the specification tables used by the LCP, which appear in alphabetical order.

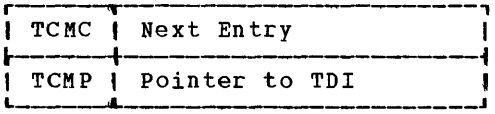
### Table TBK (Common Block Table)




TBKN
contains the common block name.

TBKP
contains the pointer to the entry in table TCM.

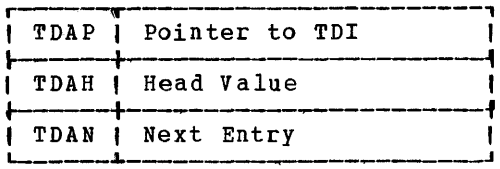
### Table TCM (Common Variable Table)




TCMC
contains the pointer to the next entry for this block.

TCMP
contains the pointer to the dictionary entry for the variable.

### Table TDA (Table of Variables with Initial Values)

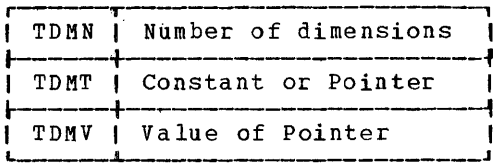



TDAP
contains the pointer to the dictionary entry.

TDAH
contains the head value in the EQUBLK including the variable.

TDAN
contains the pointer to the next entry in table TDA.

### Table TDM (Dimension Table)



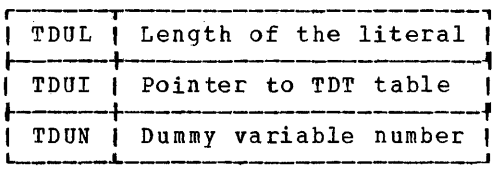

TDMN
number of subscripts in the array.

TDMT
contains one of the following constants:

TDMC if TDMV is a constant
TDMP if TDMV is a pointer to a dictionary entry

TDMV
contains the value of the pointer or of the constant.

### Table TDU (Dummy Variable Table)




TDUL
contains the length of the literal which is part of a FORMAT statement in table TDT

TDUI
contains the pointer to table TDT

TDUN
contains the number (3 characters) of the dummy variable created for the literal (DUMMxxx)

Table TDT (Table of Initial Values)

| | |
|---|---|
| TDTN | Next Entry |
| TDTL | Length of Literal |
| TDTR | Repetition Factor |
| TDTH | Position of Element |
| TDTT | Type of Data |
| TDTC | Character-String |

TDTN
contains the pointer to the next entry in table TDT.

TDTL
contains the number of characters in the string.

TDTR
contains the repetition factor.

TDTH
contains the value of the position, within the array, of the element to be initialized.

TDTT
indicates whether TDTC is numeric (set to 0) or alphameric (set to 1).

TDTC
contains the character-string collected.

Table TEH (Table of EQUBLK's)

| | |
|---|---|
| TEHN | EQUBLK Number |
| TEHT | Type |
| TEHE | Explicit Type |
| TEHL | Length |
| TEHM | Size |
| TEHA | Displacement EQUBLK/COMMON block (if necessary) |

TEHN
contains the sequence number of the EQUBLK.

TEHT
contains one of the following constants:

LCTI if the type of the EQUBLK is integer
LCTR if it is real
LCTC if it is complex
LCTL if it is logical

TEHE
contains the constant LCET if the length or type of the EQUBLK must be explicitly declared.

TEHL
contains, according to the length in bytes of each element of the EQUBLK:

LCLO for a length of 1 byte
LCLT for a length of 2 bytes
LCLF for a length of 4 bytes
LCLE for a length of 8 bytes
LCLS for a length of 16 bytes

TEHM
contains the size of the EQUBLK.

TEHA
contains the displacement of EQUBLK from the COMMON block if an item of the equivalence chain appears in COMMON.

Table TEQ (Equivalence Table)

| | |
|---|---|
| TEQP | Pointer to TDI |
| TEQH | Head |
| TEQT | Tail |
| TEQN | Next Element |

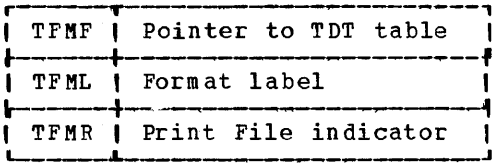
TEQP
contains the pointer to the dictionary entry.

TEQH
contains the head value in EQUBLK.

TEQT
contains the tail value in EQUBLK.

TEQN
contains the chain pointer to the next element in the equivalence group.

Table TFM (Format table)



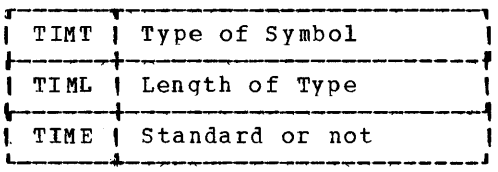

TFMF
contains the pointer to table TDT (packed FORMAT)

TFML
contains the label of the FORTRAN FORMAT statement

TFMR
indicates if the format is used with a PRINT or non-PRINT file or both.

Table TIM (Implicit Statement Table)

This table contains 256 entries, one for each of the possible bit configurations accepted by the System/360.




TIMT
contains one of the following constants:

LCTI if the type of symbol is integer
LCTR if it is real
LCTC if it is complex
LCTL if it is logical
Zero if the first character of the symbol is not alphabetical

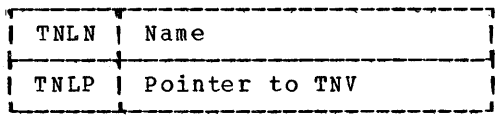
TIML
contains one of the following constants:

LCLO for a length of 1 byte
LCLT for a length of 2 bytes
LCLF for a length of 4 bytes
LCLE for a length of 8 bytes
LCLS for a length of 16 bytes
Zero if the first character of the symbol is not alphabetical

TIME
contains one of the following constants:

LCET if either TIMT or TIML are not standard
Zero if they are

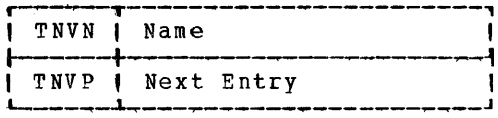
Table TNL (NAMELIST Table)




TNLN
contains the NAMELIST name.

TNLP
contains the pointer to table TNV.

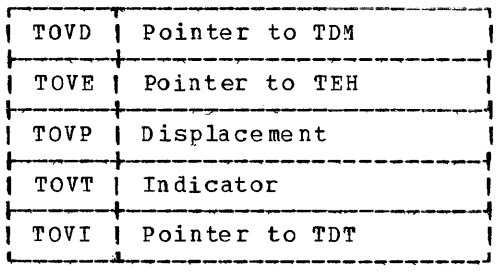
Table TNV (List of Variable Table)




TNVN
contains the variable name pointer.

TNVP
contains the pointer to the next entry under this NAMELIST name.

Table TOV (Overflow Table)




TOVD
contains the pointer to the entry in table TDM (array).

TOVE
contains the pointer to the entry in table TEH for the EQUBLK containing the variable (equivalenced item).

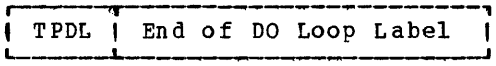
TOVP
contains the displacement.

TOVT
is used in Phase 20 to indicate whether or not the variable has already been placed in an EQUBLK.

TOVI
contains the pointer to the entry in table TDT.

Table TPD (DO Loop Label Table)

This is a push-down table which is built up by the GETCRD routine. When GETCRD returns control to Phase 10, this table is empty and no longer used.




TPDL
contains the label of the end of the DO loop.

The LCP provides a subroutine (LBLNK) which is invoked at PL/I target program execution time to process the external form of numeric input data.

If the option BLKZR has been specified at conversion time, LBLNK is generated to convert E-, F-, and I-format items which refer to at least one non-PRINT file. The length of the data field is passed as an argument. LBLNK makes the following alterations during the transmission of external data:

- plus sign in BCDIC (&) is replaced by +
- D is changed to E
- An all blank field is replaced by 0
- If a blank appears after E, it is replaced by +
- other embedded and trailing blanks are replaced by zeroes.

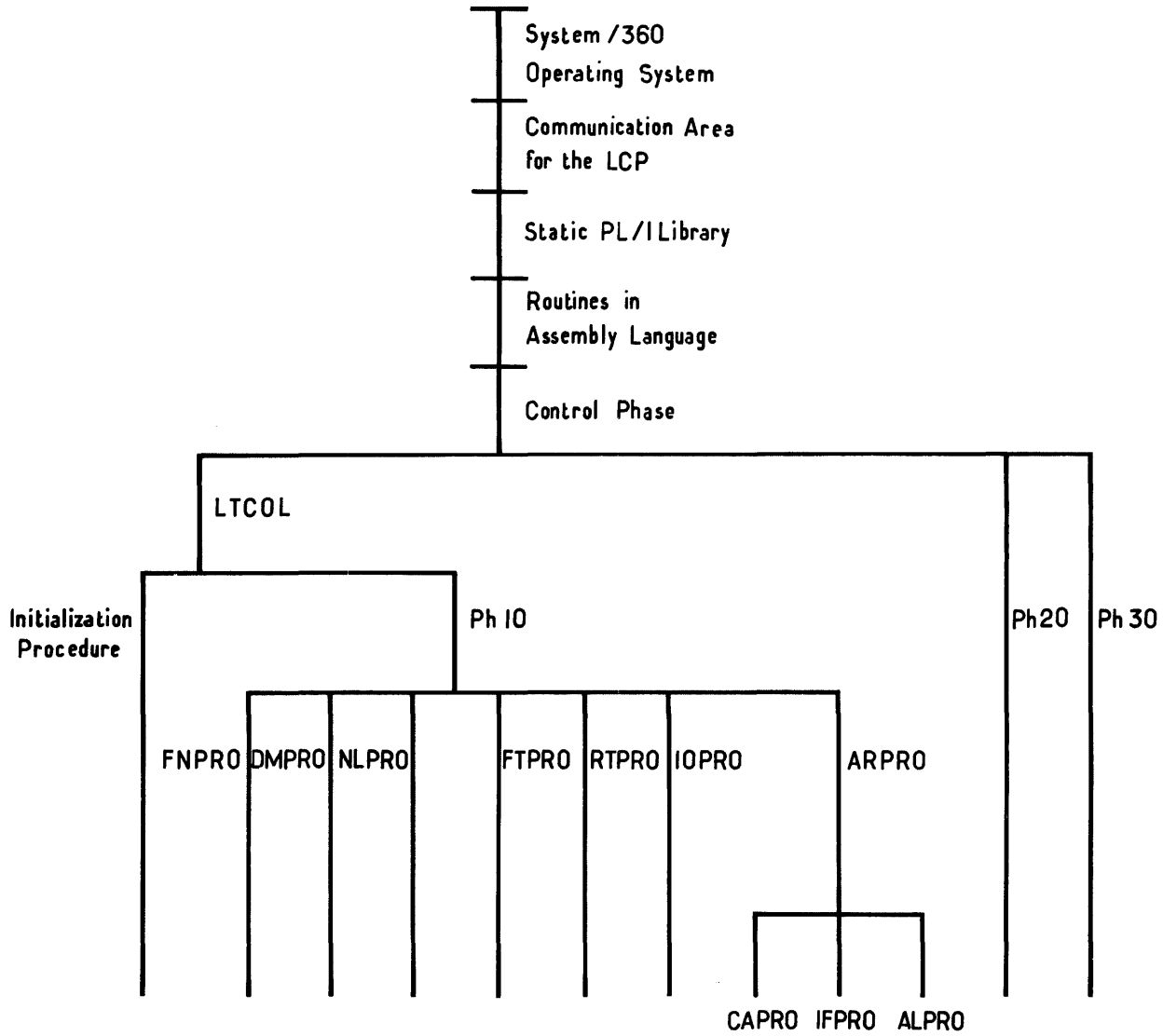
## APPENDIX C. STORAGE MAP

The following diagram illustrates the way in which the LCP uses main storage.

The routines PUTTBL, GETTBL, ENTTDI and CONVER are written in Assembly Language.

The various tables are in the Communication Area.

The exact sizes of the components of the program are not shown since they depend on the options used to compile the LCP.



Where more than one reference is given, the first page number indicates the major reference.


A code..... 27
A-format code..... 26,25
arithmetic expressions..... 28
ARPRO processing..... 30
ASPRO..... 24

BACKSPACE statement..... 21
BEGIN statement..... 22
BINARY function..... 29
BLKZR..... 87,25
BLOCK DATA statement..... 22
BLPRO..... 22
BLSW..... 22
BRANCH..... 26
BRNPRO..... 32
BRSW..... 32

CALL statement..... 24,28
CAPRO..... 24
carriage control option..... 27
CHARACTER..... 74,78
CHARACTER attribute..... 78
Chart 001..... 11,10
Chart 002..... 14,12
Chart 003..... 15,12
Chart 004..... 16,12
Chart 005..... 17,12
Chart 006..... 18,12
Chart 007..... 19,12
Chart 008..... 20,13
Chart 009..... 33,21
Chart 010..... 34,22
Chart 011..... 35,22
Chart 012..... 36,22
Chart 013..... 37,22
Chart 014..... 38,22
Chart 015..... 39,22
Chart 016..... 40,23
Chart 017..... 41,23
Chart 018..... 42,23
Chart 019..... 43,23
Chart 020..... 44,24
Chart 021..... 45,24
Chart 022..... 46,24
Chart 023..... 47,24
Chart 024..... 48,25
Chart 025..... 49,25
Chart 026..... 50,25
Chart 027..... 51,26
Chart 028..... 52,26
Chart 029..... 53,27
Chart 030..... 54,27
Chart 031..... 55,28
Chart 032..... 56,28
Chart 033..... 57,28
Chart 034..... 58,28
Chart 035..... 59,28
Chart 036..... 60,28
Chart 037..... 61,28
Chart 038..... 62,30
Chart 039..... 63,30
Chart 040..... 64,31
Chart 041..... 65,31
Chart 042..... 66,31
Chart 043..... 67,31
Chart 044..... 68,31
Chart 045..... 69,32
Chart 046..... 70,32
Chart 047..... 71,32
Chart 048..... 72,32
Chart 049..... 73,32
Chart 050..... 75,74
Chart 051..... 76,74
Chart 052..... 77,74
Chart 053..... 79,78
Chart 054..... 80,78
Chart 055..... 81,78
classification scan..... 18
CMPRO..... 22
COLUMN..... 25
comments cards..... 24
COMMON..... 84
COMMON block..... 78,84
COMMON statement..... 22,78
communication area..... 88
communication area, LCP..... 12
COMPLEX statement..... 23
CONTINUE statement..... 25
control cards, LCP..... 12
control phase..... 10,7
control phase, overall logic..... 11
COPRO..... 24
CSHL..... 13
CSNA..... 13
CSNU..... 13
CTPRO..... 25

D-format code..... 26
DAPRO..... 22
DATA..... 27
DATA literals..... 32
data-set reference number..... 12
DATA statement..... 22,32,74,83
DATA tables..... 8
declarative statement..... 32,78
DECLARE statement..... 78
DIAR..... 83
DIAU..... 83
DICL..... 82
DIET..... 82
DILG..... 82
DILT..... 82
DIMENSION statement..... 22,83
dimensions..... 32
DIMO..... 83
DINAM..... 82,83
DIOV..... 83
DITY..... 82

DMPRO..... 22
DO statement..... 25
DOPRO..... 25
DOSW..... 25
DOUBLE PRECISION..... 23
DTPRO..... 74

E format..... 87
E-format code..... 26
EDIT option..... 27
EDPRO..... 22
END option..... 27
END statement..... 22
ENDFILE statement..... 21
entry point parameter list..... 31
ENTRY statement..... 23
ENTTDI..... 30
EQPRO..... 22
EQUBLK..... 74,83-86
EQUBLKj..... 74
EQUIVALENCE block..... 78
EQUIVALENCE statement..... 22,74,83
EQUIVALENCE tables..... 81
ERMS..... 30
ERROR option..... 27
ETPRO..... 23
EVPRO..... 44
EXEC control card..... 10,12,78
executable statement processing routines..... 23,21
EXPRO..... 23
EXTERNAL statement..... 23

F format..... 87
F-format code..... 26
FORMAT reference number..... 13,27
FORMAT statement..... 25,83
FNPRO..... 23
FTPRO..... 25
function call..... 20
FUNCTION statement..... 23

G-format code..... 26
GET..... 27
GETCRD..... 12,86
GETTBL..... 31
GETWRD..... 31
GO TO statement..... 26
GTPRO..... 26

I format..... 87
I-format code..... 26
IFPRO..... 26
IF statement..... 26
IMPLICIT statement..... 13
IMPRO..... 13
INITIAL attribute..... 74,78
initial values..... 74
initialization phase..... 12,7
input/output statements..... 27
insert table..... 59
INTEGER..... 23
integer division..... 29
I/O List..... 27
IOPRO..... 27
IPPRO..... 12
iteration group..... 26

KEYDIC..... 13

L-format code..... 26
LABPRO..... 32
LBLNK..... 25,87
LCAU..... 83
LCCRT..... 32
LCET..... 85,82,84
LCENT..... 82
LCEQU..... 83
LCLAB..... 82
LCLE..... 82,84,85
LCLF..... 82,84,85
LCLO..... 82,84,85
LCP, overall logic..... 9,4
LCLS..... 82,84,85
LCLT..... 82,84,85
LCSUF..... 83
LCTA..... 82
LCTC..... 82,84,85
LCTI..... 82,84,85
LCTL..... 82,84,85
LCTR..... 82,84,85
LCVAR..... 82
LOGICAL..... 23
logical expressions..... 28
LTCOL..... 31

MAINPRO..... 78

N code..... 27
NAMELIST name..... 85
NAMELIST option..... 27
NAMELIST statement..... 23
NLPRO..... 23
non-PRINT..... 13,85,87

ON conditions..... 27

P code..... 27
packed FORMAT..... 25,27
PAGE..... 27
PAR1..... 25
PAR2..... 25
PAR3..... 25
PARM..... 12
PAUSE statement..... 28
phase 10..... 21,7,8
phase 20..... 74,7,8
phase 30..... 78,7,8
PL/I built-in function names..... 21
PRINT file..... 12
PRINT statement..... 13
PROCEDURE..... 23
PROCEDURE statement..... 24,78
PSPRO..... 28
PUNCH statement..... 13
pushdown table..... 28,29
PUT..... 27
PUTTBL..... 31

READ statement..... 27,13
REAL..... 23

RETURN statement..... 28
RTPRO..... 28

SBPRO..... 23
SKIP..... 25,27
SLPAR..... 31
source program..... 7
source program identifiers..... 21
source program listing..... 8
SPDIM..... 32
SPDTA..... 32
specification statement processing routines..... 21
specification tables..... 83
SPPRO..... 32
SPSUB..... 32
STOP statement..... 28
STPRO..... 28
storage map..... 88
subroutine library..... 87
SUBROUTINE statement..... 23
subscripts..... 32
substitution name..... 21
SYSIN..... 7,12
SYSPRINT..... 78
SYSPRNT..... 8
SYSUT1..... 8,13,21,25,28,30,32,78
SYSUT2..... 8,82
SYSPCH..... 78

tables..... 7,82
target program..... 7
target program generation..... 78
TBK..... 83,22,31,78
TBKN..... 83
TBKP..... 83
TBLCW..... 29,31
TCM..... 83,22
TCMC..... 83
TCMP..... 83
TDA..... 83,22,74
TDAH..... 83
TDAN..... 83
TDAP..... 83
TDI..... 82,21-24,26,28,30,74,78
TDM..... 83,22,82,85
TDMT..... 83
TDMV..... 83
TDT..... 84,27,26,32,74,86
TDTC..... 84
TDTH..... 84
TDTL..... 84
TDTN..... 84
TDTR..... 84
TDTT..... 84
TDU..... 83,26,27,78
TDUI..... 83
TDUL..... 83
TDUN..... 84
TEH..... 84,74
TEHA..... 84
TEHL..... 84
TEHE..... 84
TEHM..... 84
TEHN..... 84
TEHT..... 84
TEQ..... 84,74,22
TEQH..... 84
TEQN..... 84
TEQP..... 84
TFM..... 85,27,13,26,31
TFMF..... 85
TFML..... 85
TFMR..... 85
TIM..... 85,13
TIME..... 85
TIML..... 85
TIMT..... 85
TNL..... 85,23,31
TNLN..... 85
TNLP..... 85
TNV..... 85,23
TNVN..... 85
TNVP..... 85
TOV..... 85,32
TOVD..... 85
TOVE..... 85
TOVI..... 86
TOVP..... 85
TOVT..... 86
TPD..... 86,13
TPDL..... 86
TRUNC function..... 29
type statement, explicit..... 23
TYPE statement..... 32,74

UNLABCM..... 78
UPRNT..... 78
utility routines..... 28,21

WKBF..... 13,22,24,25,28,,30-32
WKBL..... 23
WORD..... 32,78
WRITE statement..... 27,13

XTBPRO..... 32
XTDLM..... 31

YIPRO..... 23
YCPRO..... 23
YDPRO..... 23
YLPRO..... 23

Z-format code..... 26
Zero..... 83,85

## READER'S COMMENT FORM

IBM System/360 Conversion Aids:
FORTRAN IV-to-PL/I Language Conversion Program
for IBM System/360 Operating System

Y33-7000-0

- How did you use this publication?

  As a reference source ..... ☐
  As a classroom text ..... ☐
  As a self-study text ..... ☐

- Based on your own experience, rate this publication . . .

| | Very Good | Good | Fair | Poor | Very Poor |
|---|---|---|---|---|---|
| As a reference source: | ..... | ..... | ..... | ..... | ..... |
| As a text: | ..... | ..... | ..... | ..... | ..... |

- What is your occupation? .....

- We would appreciate your other comments; please give specific page and line references where appropriate. If you wish a reply, be sure to include your name and address.

- Thank you for your cooperation. No postage necessary if mailed in the U.S.A.

## YOUR COMMENTS PLEASE . . .

This SRL bulletin is one of a series which serves as reference sources for systems analysts, programmers and operators of IBM systems. Your answers to the questions on the back of this form, together with your comments, will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of IBM.

Please note: Requests for copies of publications and for assistance in utilizing your IBM system should be directed to your IBM representative or to the IBM sales office serving your locality.

Fold

Fold

FIRST CLASS
PERMIT NO. 1359
WHITE PLAINS, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
112 East Post Road
White Plains, N. Y. 10601

Attention: Department 813

Fold

Fold

IBM

**International Business Machines Corporation**
**Data Processing Division**
**112 East Post Road, White Plains, N.Y. 10601**
**[USA Only]**

**IBM World Trade Corporation**
**821 United Nations Plaza, New York, New York 10017**
**[International]**

Printed in U.S.A. Y33-7000-0

Printed in U.S.A. Y33-7000-0

**International Business Machines Corporation**
**Data Processing Division**
**112 East Post Road, White Plains, N.Y. 10601**
**[USA Only]**

**IBM World Trade Corporation**
**821 United Nations Plaza, New York, New York 10017**
**[International]**