77-FM-18
Vol.V, Rev.1

JSC-12564

Flight Design System-1 System Design

(NASA-TM-80468) FLIGHT DESIGN SYSTEM-1 SYSTEM DESIGN. VOLUME 5: DATA MANAGEMENT AND DATA BASE DOCUMENTATION SUPPORT SYSTEM (NASA) 175 p HC A08/MF A01 CSCL 22A

N79-30278

Unclas
G3/16 31940

Data Management and Data Base Documentation Support System

Mission Planning and Analysis Division

June 1979

SEP 1979
RECEIVED
NASA STI FACILITY
ACCESS DEPT.

National Aeronautics and
Space Administration

Lyndon B. Johnson Space Center
Houston, Texas

77FM18:V

77-FM-18
Vol. V, Rev. 1

JSC-12564

SHUTTLE PROGRAM

FLIGHT DESIGN SYSTEM-1
SYSTEM DESIGN

DATA MANAGEMENT AND DATA BASE
DOCUMENTATION SUPPORT SYSTEM

By Software Development Branch

Approved: Elric N. McHenry
Elric N. McHenry, Chief
Software Development Branch

Approved: Ronald L. Berry
Ronald L. Berry, Chief
Mission Planning and Analysis Division

Mission Planning and Analysis Division

National Aeronautics and Space Administraton

Lyndon B. Johnson Space Center

Houston, Texas

June 1979

PREFACE

The Flight Design System-1 (FDS-1) is a pilot project to evaluate current concepts, and to determine the hardware/software capability that will be required for the operational era to support Shuttle flight planning. This software system is being implemented on a Hewlett-Packard 21MX (HP21MX) computer combined with a Daconics documentation system, and will provide terminal-based interactive flight planning capability.

The System Design Document (SDD) for FDS-1 is the specification for, and description of, this hardware/software (HW/SW) facility. The SDD is logically organized into 10 published volumes. This organization is presented in the accompanying table. The material in the early volumes is primarily presented from the user's point of view, whereas the latter material is software-developer oriented. The SDD will be published by volumes over a period of time, and various volumes will be updated and republished during the development of FDS-1.

PRECEDING PAGE BLANK NOT FILMED

FDS-1 SYSTEM DESIGN DOCUMENT

Volume I	Introduction, Overview, and User Interface
Volume II	Utility Processor Library
Volume III	Application Processor Library
Volume IV	System Architecture and Executive
Volume V	Data Management and Data Base Documentation Support System
Volume VI	Standards
Volume VII	Utility Support Software
Volume VIII	Build and Delivery Procedures; Software Development, Debug, and System Build Aids
Volume IX	Executive Logic Flow - Program Design Language
Volume X	Document Change Request Procedure and Submittal Form

VOLUME V

DATA MANAGEMENT AND DATA BASE DOCUMENTATION SUPPORT SYSTEM

CONTENTS

Section		Page
1.0	INTRODUCTION	1-1
2.0	CONCEPT	2-1
3.0	OPERATIONAL PROCESSOR DESCRIPTION	3-1
3.1	CONSTRUCTION OF FORM AND DDT	3-1
3.2	CONSTRUCTION OF DEFAULT LINKAGE TABLES	3-1
3.3	MODIFICATION OR COMPLETION OF LINKAGE TABLES	3-1
3.4	CONSTRUCTION OF COMPLETED DOCUMENT SEGMENT	3-2
3.5	EDITING AND PRINTING	3-2
4.0	ASSUMPTIONS AND RESTRICTIONS	4-1
5.0	SYSTEM REQUIREMENTS	5-1
5.1	OPERATIONAL REQUIREMENTS	5-1
5.2	HW/SW CONFIGURATION REQUIREMENTS	5-1
5.3	USER INTERFACE REQUIREMENTS	5-1
5.4	DATA STORAGE REQUIREMENTS	5-2
6.0	GENERAL SOFTWARE AND DATA ORGANIZATION	6-1
6.1	DACONICS	6-1
6.1.1	Software	6-1
6.1.2	Data Elements	6-1
6.2	HP21MX	6-2
6.2.1	Software	6-2
6.2.2	Data Elements	6-3
7.0	LINKAGE TABLE BUILD PROGRAM (LTBLD)	7-1
7.1	PURPOSE	7-1
7.2	FUNCTIONAL DESCRIPTION	7-1
7.3	ASSUMPTIONS AND LIMITATIONS	7-3

PRECEDING PAGE BLANK NOT FILMED

Section		Page
7.4	PROGRAM INPUT/OUPUT VARIABLES	7-3
7.4.1	Displays	7-6
7.4.2	File Outputs	7-11
7.4.3	Linkage Table Editor Prompts	7-11
7.5	LINKAGE TABLE BUILD PROGRAM (LTBLD) SUBROUTINES	7-16
7.5.1	Program Name - Main Program LTBLD	7-17
7.5.2	Program Name - Main Program LINLG	7-21
7.5.3	Subroutine Name - LTBSG	7-24
7.5.4	Program Name - Main Program LNXLG	7-28
8.0	DOCUMENTATION PROCESSOR (DOC)	8-1
8.1	PURPOSE	8-1
8.2	FUNCTIONAL DESCRIPTION	8-1
8.3	ASSUMPTIONS AND LIMITATIONS	8-1
8.4	PROCESSOR INPUT/OUTPUT	8-2
8.5	DOCUMENTATION PROCESSOR (DOC) ROUTINES	8-18
8.5.1	Routine Name - Main Program DOC	8-18
8.5.2	Subroutine Name - AWAG	8-22
8.5.3	Subroutine BRUPK	8-26
8.5.4	Subroutine DACIG	8-29
8.5.5	Subroutine DOPRG	8-35
8.5.6	Subroutine DOSEG	8-44
8.5.7	Routine Name - Main Program DPBSG	8-50
8.5.8	Subroutine DREAD	8-54
8.5.9	Subroutine DWRIT	8-58
8.5.10	Subroutine FMING	8-62
8.5.11	Routine Name - Main Program LINDG	8-66
8.5.12	Routine Name - Main Program LNXDG	8-68
8.5.13	Subroutine LPRMG	8-71
8.5.14	Subroutine LTDIG	8-72
8.5.15	Subroutine LTIFG	8-76
8.5.16	Subroutine LTMAG	8-80
8.5.17	Subroutine LTMOG	8-84
8.5.18	Subroutine LTNMG	8-88
8.5.19	Subroutine LTNNG	8-92
8.5.20	Subroutine LTPFG	8-96
8.5.21	Subroutine LTPRG	8-100
8.5.22	Subroutine LTXIG	8-104
8.5.23	Subroutine MERG	8-109
8.5.24	Subroutine SDISG	8-113
8.5.25	Subroutine SOUTG	8-117

TABLES

Table		Page
7.4-I	PROGRAM SOLICITED (PROMPTED) INPUTS (LTBLD)	7-4
7.4.1-I	PROGRAM MESSAGE TABLE (LTBLD)	7-7
7.5.1-I	INPUT/OUTPUT VARIABLES (LTBLD)	7-19
7.5.3-I	INPUT/OUTPUT VARIABLES (LTBSG)	7-26
8.4-I	PROCESSOR SOLICITED (PROMPTED) INPUTS (DOC)	8-3
8.4-II	PROCESSOR DISPLAY FORMAT	
	(a) Document ID Directory	8-10
	(b) Display parameter definition table for the Document ID Directory	8-11
	(c) Segment ID Directory	8-12
	(d) Display parameter definition table for the Segment ID Directory	8-13
	(e) Linkage Table Directory	8-14
	(f) Display parameter definition table for the Linkage Table Directory	8-15
8.4-III	PROCESSOR MESSAGE TABLE (DOC)	8-16
8.5.1-I	INPUT/OUTPUT VARIABLES (DOC)	8-20
8.5.2-I	INPUT/OUTPUT VARIABLES (AWAG)	8-24
8.5.3-I	INPUT/OUTPUT VARIABLES (BRUPK)	8-27
8.5.4-I	INPUT/OUTPUT VARIABLES (DACIG)	8-31
8.5.5-I	INPUT/OUTPUT VARIABLES (DOPRG)	8-38
8.5.6-I	INPUT/OUTPUT VARIABLES (DOSEG)	8-47
8.5.7-I	INPUT/OUTPUT VARIABLES (DPBSG)	8-52
8.5.8-I	INPUT/OUTPUT VARIABLES (DREAD)	8-56
8.5.9-I	INPUT/OUTPUT VARIABLES (DWRIT)	8-60
8.5.10-I	INPUT/OUTPUT VARIABLES (FMING)	8-64
8.5.14-I	INPUT/OUTPUT VARIABLES (LTDIG)	8-74
8.5.15-I	INPUT/OUTPUT VARIABLES (LTIFG)	8-78

Table | Page

8.5.16-I INPUT/OUTPUT VARIABLES (LTMAG) 8-82

8.5.17-I INPUT/OUTPUT VARIABLES (LTMOG) 8-86

8.5.18-I INPUT/OUTPUT VARIABLES (LTNMG) 8-90

8.5.19-I INPUT/OUTPUT VARIABLES (LTNNG) 8-94

8.5.20-I INPUT/OUTPUT VARIABLES (LTPFG) 8-98

8.5.21-I INPUT/OUTPUT VARIABLES (LTPRG) 8-102

8.5.22-I INPUT/OUTPUT VARIABLES (LTXIG) 8-106

8.5.23-I INPUT/OUTPUT VARIABLES (MERG) 8-111

8.5.24-I INPUT/OUTPUT VARIABLES (SDISG) 8-115

8.5.25-I INPUT/OUTPUT VARIABLES (SOUTG) 8-119

FIGURES

Figure		Page
2-1	Sample form	2-2
2-2	Document descriptor table definition	2-2
2-3	Document descriptor table sample	2-3
2-4	Linkage table	2-4
2-5	Sample linkage table	2-5
2-6	Data organization and flow	2-6
7.4.1-1	Document ID Directory	7-9
7.4.1-2	Segment ID Directory	7-9
7.4.1-3	Linkage Table Directory	7-9
7.4.1-4	Linkage table listing	7-10
7.4.1-5	DDT for document PRFP, segment ascent	7-10
7.5.1-1	LTBLD functional logic flow	7-20
7.5.2-1	LINLG functional logic flow	7-23
7.5.3-1	LTBSG functional logic flow	7-27
7.5.4-1	LNXLG functional logic flow	7-29
8.4-1	Logic diagram of the DOC processor solicited inputs	8-6
8.5.1-1	DOC functional logic flow	8-21
8.5.2-1	AWAG functional logic flow	8-25
8.5.3-1	BRUPK functional logic flow	8-28
8.5.4-1	DACIG functional logic flow	8-32
8.5.5-1	DOPRG functional logic flow	8-39
8.5.6-1	DOSEG functional logic flow	8-48
8.5.7-1	DPBSG functional logic flow	8-53
8.5.8-1	DREAD functional logic flow	8-57

Figure Page

8.5.9-1 DWRIT functional logic flow 8-61

8.5.10-1 FMING functional logic flow 8-65

8.5.11-1 LINDG functional logic flow 8-67

8.5.12-1 LNXDG functional logic flow 8-70

8.5.14-1 LTDIG functional logic flow 8-75

8.5.15-1 LTIFG functional logic flow 8-79

8.5.16-1 LTMAG functional logic flow 8-83

8.5.17-1 LTMOG functional logic flow 8-87

8.5.18-1 LTNMG functional logic flow 8-91

8.5.19-1 LTNNG functional logic flow 8-95

8.5.20-1 LTPFG functional logic flow 8-99

8.5.21-1 LTPRG functional logic flow 8-103

8.5.22-1 LTXIG functional logic flow 8-108

8.5.23-1 MERG functional logic flow 8-112

8.5.24-1 SDISG functional logic flow 8-116

8.5.25-1 SOUTG functional logic flow 8-120

1.0 INTRODUCTION

A prime objective of the Flight Design System (FDS) is to increase the productivity of the flight planning engineers sufficiently to permit support of the high flight rate of the Shuttle operations era within present staffing levels. One of the features of the FDS design that is intended to significantly reduce the man-hours required per flight design cycle is the ability to produce the major flight design documents with little or no manual typing. This volume describes the set of application software that provides this capability.

The documentation support software is divided into two separately executable processors. However, since both processors support the same overall function, and most of the software contained in one is also contained in the other, both are collectively described in this section.

2.0 CONCEPT

The basic concept of semiautomated documentation for the FDS consists of combining prestructured document formats with flight-dependent data and formatting the result in a form suitable for printing as a complete document. The prestructured document formats are created and maintained on the Daconics word processing system. The flight-dependent data are produced by the FDS and are stored in named data elements on the HP21MX computer. The two hardware systems are connected by a core-to-core data channel over which the empty document formats are transmitted to the HP21MX, and the combined formats and data are returned to the Daconics. The combining of data with formats is performed by the documentation processor (DOC), a set of application software that resides on the HP21MX.

A document is constructed from a set of separately complete portions called segments. Each segment, identified by a document and segment identifier, is defined by a preformatted form and an associated table that defines each data element to be inserted into reserved spaces in the form. This table, called the document descriptor table (DDT), defines the order in which data are to be inserted into the form, and specifies the format in the segment for each data element. Figure 2-1 shows a sample form, figure 2-2 describes the contents of a DDT, and figure 2-3 shows the sample DDT that corresponds to the sample form in figure 2-1.

A second table is required to establish the linkages between the parameters defined in the DDT and the named data elements in the active work area (AWA) of the FDS. This table is called the linkage table (LT) and is described in figure 2-4. A sample LT is shown in figure 2-5.

A separate program exists that produces and modifies LT's. This program, called the Linkage Table Build Program (LTBLD), is used to construct default LT's from DDT's and may also be used to modify existing LT's or to create new LT's from existing LT's. The DOC utilizes the LT's to retrieve the required data elements from the AWA, reformats them according to the format specifications contained in the DDT, merges them with the forms, and transmits the completed forms to the Daconics. The DOC can also modify LT's but cannot create a default LT from the DDT. The data organization and flow are shown in figure 2-6.

77FM16:V

DOCUMENT = RFP, SEGMENT = ASCENT

Lift-off from [ ] occurs at [ ] seconds after solid rocket booster (SRB) ignition (GET = [ ] seconds) after which a vertical rise to tower clearance phase is initiated, ending at a relative velocity of [ ] fps (GET = [ ] seconds). After tower clearance, a controlled pitchover program is initiated to achieve MECO targets while reducing aerodynamic loads. A maximum dynamic pressure of [ ] lb/ft squared is achieved at a GET of [ ] seconds (geodetic altitude [ ] feet, alpha [ ], beta [ ], Earth relative velocity [ ] fps).

Figure 2-1.- Sample form.

Field	Description
NAME	Name of the variable as defined by the document segment designer
TYPE	Type of data element in AWA
REPEAT FLAG	Indicates arrays to be presented in columnar table format
DIMENSION	Maximum number of occurrences provided for in the form
FORMAT	Format by which variable is to be displayed in completed document segment
DESCRIPTION	40-character description of variable for user prompting

Figure 2-2.- Document descriptor table definition.

Name	Type	Repeat flag	Dimension	Format	Description
LSITE	C6			A3	Launch site
TLO	R			TO:0:0:3.1	Time of lift-off (GET)
TLO	R			TO:0:0:3.1	Time of lift-off (GET)
RELVEL	R			I3	Relative velocity
TOLNCE	R			TO:0:0:3.1	Time of tower clearance
MAXQPR	R			I3	Maximum dynamic pressure
TMAXQ	R			TO:0:0:4.1	Time of max Q
GDALT	R			I5	Geodetic altitude
ALPHA	R			F5.1	Flightpath angle
BETA	R			F4.1	Sideslip
ERVEL	R			I4	Earth-relative velocity

Figure 2-3.- Document descriptor table sample.

Field	Description
CLASS	SET = 2 to indicate memory resident data element
TYPE	FDS Executive data type
NAME	Name of variable used to satify the parameter
	SET = 0 if LITERAL is input
I/O	I/O flag. SET = 2 to indicate input
IDIM	Dimension(s)

Figure 2-4.- Linkage table.

Class	Type	Name	I/O[2]	IDIM
2	5	LNCHS	2	1
2	2	TLNCH	2	1
2	2	TLNCH	2	1
2	2	SUMTAB(6)	2	1
2	2	SUMTAB(7)	2	1
2	2	SUMTAB(12)	2	1
2	2	SUMTAB(13)	2	1
2	2	POS(3)	2	1
2	2	ALPHA	2	1
2	2	BETA	2	1
2	2	VEL(1)	2	1

Figure 2-5.- Sample linkage table.

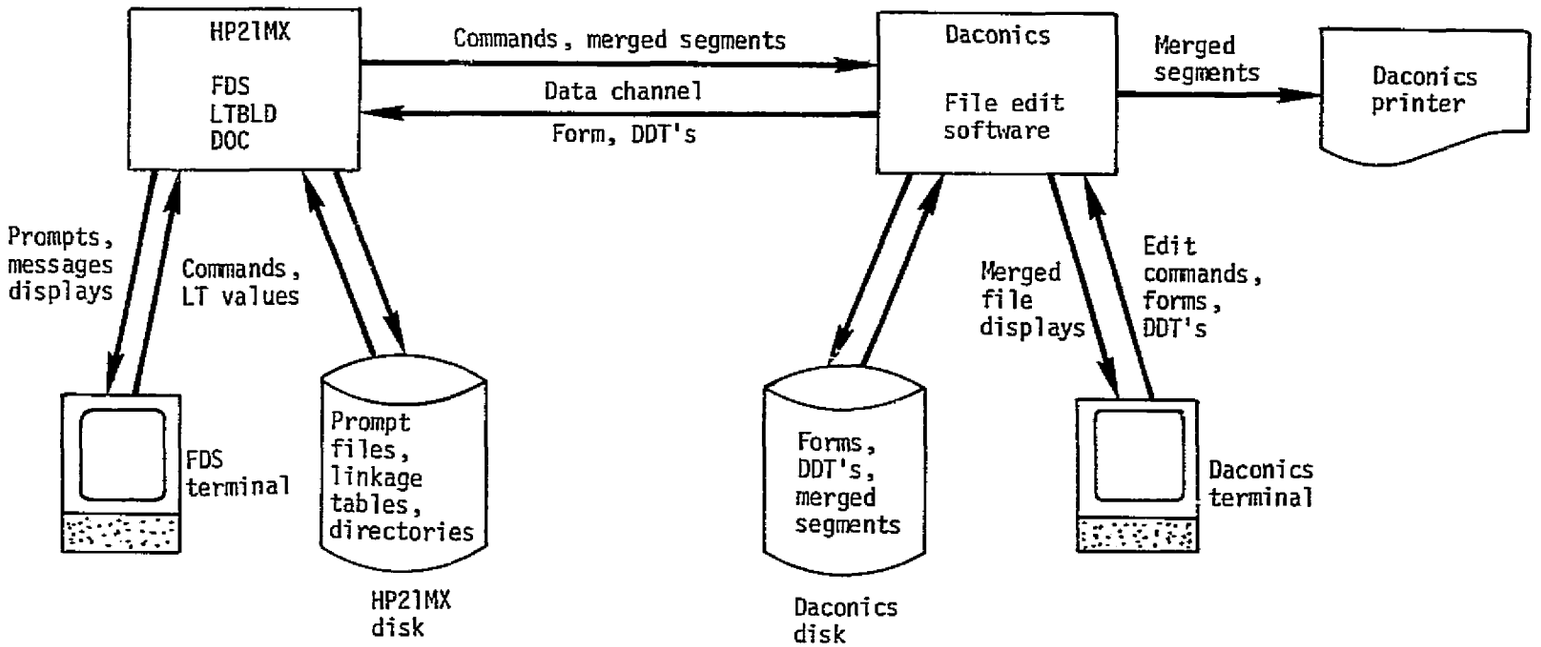


Figure 2-6.- Data organization and flow.

3.0 OPERATIONAL PROCESSOR DESCRIPTION

The semiautomated documentation process consists of four major steps. These steps do not include the execution of the FDS processors to provide the required data.

3.1 CONSTRUCTION OF FORM AND DDT

The first step is the design of the standard segment and the construction of the form and its associated DDT. The definition of a segment is a task of the document designer. It may consist of a single table or may be an entire section of a document several pages in length. Segments are designed such that they may be used as building blocks to produce documentation for a variety of flights. For example, a rendezvous segment may appear once, twice, or not at all in a document depending on the particular flight profile.

The form is developed in the exact format desired in the final document, with the correct number of blank spaces set aside to receive the needed variable data.

This form is stored and maintained on the Daconics mass storage system for use by the DOC, and is under strict configuration control. The DDT is developed at the same time, also by the document designer, and is stored on the Daconics system. The DDT is also under configuration control, and any maintenance on it is carried out by Daconics operations, but a copy of each DDT is also transmitted to the HP21MX and reformatted there for use by the DOC.

The forms and DDT's are constructed and maintained using the standard Daconics document create, edit, and storage operations.

3.2 CONSTRUCTION OF DEFAULT LINKAGE TABLES

After the DDT has been placed in an HP21MX file, the LT can be constructed. The LTBLD is executed for the initial creation of the default LT. Given the desired document and segment identifiers, the LTBLD determines the name of the desired DDT file and retrieves the DDT. The LTBLD then obtains the characteristics of the desired data elements from the DDT and prompts the user for the values required to complete the LT. The user may either supply the names of the required AWA data elements or, as an option, may directly supply literal values that are to appear in the completed document, or may "null" the entry. The LTBLD will normally be executed once for each DDT or each release of a new version of a DDT.

3.3 MODIFICATION OR COMPLETION OF LINKAGE TABLES

If a user needs an LT in a form different from the default table, he may create a new table or make temporary changes to the default table for use during the current session.

After the LT is retrieved, the user is given the opportunity to make changes to the values in the table. If he elects to make changes, he may then choose to make those effective for the current session only, or make them permanent by creating a new LT. He may also elect to recreate the source LT if it (the source LT) was created initially by him.

3.4 CONSTRUCTION OF COMPLETED DOCUMENT SEGMENT

Given the existence of a form, DDT, LT, and the required data elements for a given segment, the DOC is executed to combine these ingredients into a completed document segment. The user specifies the document segment to be constructed by supplying the document and segment identifiers. Directories of existing document and segment identifiers are searched to ensure that the requests are valid, and to obtain a two-character code associated with each identifier. These codes are used to check the validity of supplied LT's and to construct default names for the various files that are used internally by the DOC.

The DOC then retrieves the specified data from the AWA. The data are retrieved in the format in which they are specified in the LT (stored in the AWA). They must then be converted into the formats specified in the DDT. The converted data are then ready to be inserted into the blank spaces in the form.

Since the form is retained on the Daconics mass storage system, a request for the form for the current document segment is sent via the data channel to the Daconics. If the Daconics is up and the form file is available, the form is sent to the HP21MX and the data are merged with the form. The completed segment is then made available for review by the user. The user may then elect to (1) send the segment to Daconics without review, (2) review it on his terminal and then transmit it, or (3) review it and not send it to Daconics.

3.5 EDITING AND PRINTING

After the merged file is returned to the Daconics, the Daconics operation is notified that the file is available, and the file name is provided. If editing is required (e.g., suppression of extraneous blanks), this is carried out by the Daconics operation.

When the file is ready for printing, it is printed on the Daconics printer and returned to the engineer for review. These printed segments are then accumulated, any plots and figures prepared separately are inserted, a table of contents is printed, and the signoff copy is ready for approval.

4.0 ASSUMPTIONS AND RESTRICTIONS

A successful execution of the documentation processor is dependent on the following conditions being satisfied.

a. A form for the requested document segment must exist on the Daconics mass storage system.

b. The associated DDT must exist, and must reside in the correct format, on the Daconics.

c. A default LT must exist on the HP21MX.

d. The associated prompt file must exist on the HP21MX.

e. The version numbers of the form, DDT, prompt file, and LT must all agree.

f. LT's can be created from LT's belonging to other users, but only those belonging to the current user may be purged or modified and recreated.

g. Only data elements appearing in the AWA can be retrieved for use by the DOC; disk resident data elements (DRDE) cannot be accessed directly.

h. The FDS data type for each data element must be specified in the DDT and correspond to the type in the AWA.

i. All data elements to be displayed in time format must be stored as real seconds.

j. No units conversions are performed by the DOC except time conversions. Since time may be displayed in a variety of formats, the DOC will perform the time conversions specified by the format specification in the DDT.

k. The HP21MX must be able to communicate with the Daconics (i.e., all components must be up), and the required form file must be available on the Daconics files.

l. The FDS-1 DOC will be limited to 64 variable names per segment.

5.0 SYSTEM REQUIREMENTS

5.1 OPERATIONAL REQUIREMENTS

The DOC will be operational during the normal FDS operational periods. To be able to complete the construction of a document segment, the Daconics system and the connecting data channel must be available. The required form files must also be available on the Daconics on-line file system. If the form file cannot be retrieved during a session, the user may either retrieve the data for review only or may terminate the session.

In the event that a magnetic tape interface is to be used as an alternative to the data channel interfaces, the required form must be available on tape, and a tape drive must be available on the HP21MX. If a hard copy of the completed segment is required from the HP21MX, access to the line printer will be required by the DOC.

5.2 HW/SW CONFIGURATION REQUIREMENTS

The hardware required for the operation of the DOC consists of the HP21MX, its disk subsystem, and one interactive terminal, plus the Daconics computer, its disk subsystem, a core-to-core data channel connecting the two computers, and a Daconics terminal. A Daconics terminal and the line printer are also needed for editing and printing the completed segment.

The FDS software configuration consists of the standard FDS Executive and the supporting software required for its operation plus:

a. The file manager and FORTRAN interface

b. The FORTRAN I/O package

c. The driver for the data channel

d. LTBLD and DOC

The Daconics software will consist of the standard document create and modify software and the interface to the data channel.

In addition, the standard FDS Interface Table Editor (ITE) will be used for all linkage table creation and modification operations. A special modified version of the ITE control program will be used to permit the ITE to handle the linkage table operations.

5.3 USER INTERFACE REQUIREMENTS

All user interaction with the DOC will be via a standard FDS terminal. The user interface will be designed to minimize the typing required of the user. Default file and table names are generated internally from the requested document and

segment identifiers. When the processors ask for file or table names, a user response of "space-carriage return" will result in the default names being used. If other names are to be used, the user would respond with the desired name. When the processor asks whether or not to exercise an optional capability (e.g., list a file?), the null response will indicate the "Yes" choice and is assumed to be the nominal response in a production environment.

5.4 DATA STORAGE REQUIREMENTS

For the purpose of data storage estimating, it is assumed that there will be 10 documents defined in the system, and the documents will average 20 segments each. It is further assumed that the average segment is 2 pages in length and contains 65 data elements. With these assumptions, there will be 200 forms stored on the Daconics requiring 1.5 million characters and 200 DDT's requiring 0.9 million characters.

On the HP21MX, there will be 200 reformatted DDT's requiring 1.5 million characters, and at least 200 LT's requiring 0.2 million characters; this assuming that all users use the same set of default LT's. If we assume that there are 10 users and each has his own set of LT's, then there would be 2000 LT's requiring 2 million characters. Reality will probably be nearer the first assumption, with approximately 500 LT's. Then the total HP21MX storage would be 700 files and 2 million characters.

6.0 GENERAL SOFTWARE AND DATA ORGANIZATION

6.1 DACONICS

6.1.1 Software

Two separate software capabilities residing on the Daconics are used in support of the DOC. These are the standard Daconics editing software and the driver for the Daconics/HP21MX data channel.

The standard editing software is used for the creation and maintenance of the DDT files, and for any editing that may be required on the final merged document segments prior to printing.

The data channel driver is a special-purpose set of software, also provided by Daconics, whose purpose is to provide the interface between the Daconics operating system and the I/O channel that connects the Daconics to the HP21MX.

6.1.2 Data Elements

The four types of files residing on the Daconics mass storage system are as follows.

a. Form files - A form file exists for every document segment to be produced by the DOC. Form files are characterized by the existence of the static structure of the segment and the reserved spaces into which the data are to be placed (fig. 2-1).

b. DDT files - A DDT exists for each form file. A DDT is a structured file, consisting of columns of information describing the data elements to be inserted into the reserved spaces in the form (fig. 2-3).

c. Static files - Static files are created for any portions of a document that remain unchanged from one printing of a document to the next. These files are created, maintained, and used exclusively on the Daconics. An example of a static file is a glossary of terms appearing in a given document.

d. Merged files - After a segment has been processed, the merged result is stored on a named file on the Daconics.

6.2 HP21MX

6.2.1 Software

Only that software that is uniquely used for documentation processing will be described here. Those software elements that are shared with other functions will simply be identified.

6.2.1.1 Shared Software

a. RTE file manager

b. RTE Executive

c. FORTRAN I/O

d. FDS Executive

e. FDS Interface Table Editor

6.2.1.2 Unique Software

a. Linkage table build program (LTBLD) - The LTBLD runs as a stand-alone program under the RTE operating system. It is used for the creation and maintenance of default linkage table files. The LTBLD contains an unmodified copy of the FDS Interface Table Editor, which performs all the actual linkage table creation and modification operations.

b. Documentation processor - The DOC runs as a processor under the FDS Executive and provides the following capabilities:

   (1) Modification of linkage tables

   (2) Communications with Daconics

   (3) Retrieval of data from AWA

   (4) Reformatting of data

   (5) Insertion of data into reserved spaces in forms

c. Daconics driver - The Daconics driver is a set of special-purpose software that will permit communication between the HP21MX programs and the data channel to the Daconics. This driver is documented in section TBD.

6.2.2 Data Elements

The LTBLD and DOC use three types of data elements (disk files) on the HP21MX. These are PROMPT files, LINKAGE TABLE files, and directories.

a. PROMPT files - When a DDT is brought from the Daconics to the HP21MX, it is restructured into the form of a PROMPT file. PROMPT files are used by the Linkage Table Editor, and their format is dictated by the FDS Interface Table Editor (the Linkage Table Editor is actually a copy of the Interface Table Editor). In addition to being used for prompting, information contained in the PROMPT file is used in the initial creation of the default linkage table, and to define the format conversions and data reordering to be done by the DOC. One PROMPT file exists per document segment.

b. LINKAGE TABLE files - Linkage tables establish the name relationships between the reserved spaces in the forms and the named data elements in the AWA. They also contain all the information required by the FDS Executive for retrieval of the needed data from the AWA. The linkage tables are retained on disk in the form of permanent files. The LT files are either named automatically using the document, segment, and user identifiers, or may be named by the user. At least one LT exists for each segment, but multiple LT's may exist per segment.

c. Directories - Due to the potentially large number of files created and used by the DOC, directories of valid files are maintained to facilitate file name validation and error checking. These directories are as follows:

   (1) Document ID - The document ID directory contains a list of all valid four-character document identifiers, the associated two-character document codes, and the full document titles.

   (2) Segment ID - The segment ID directory contains a list of all valid four-character segment identifiers, the associated two-character segment code, the two-character document code that identifies the document which contains the segment, and the segment name.

   (3) LINKAGE TABLE and PROMPT file - Both the LINKAGE TABLE files and PROMPT files appear in this directory. The full six-character file name appears, along with the two-character document and segment codes. This permits not only the testing for the existence of a requested linkage table, but also the determination that it is associated with the current document and segment to be created.

7.0 LINKAGE TABLE BUILD PROGRAM (LTBLD)

7.1 PURPOSE

The principal purpose of the LTBLD is to create default linkage tables from DDT's. It also creates, at the same time, an associated PROMPT file that is used internally whenever modifications are made to a linkage table.

As a secondary purpose, it can also create new linkage tables from existing ones, but this is more commonly done by the documentation processor (DOC).

7.2 FUNCTIONAL DESCRIPTION

The LTBLD utilizes a copy of the FDS Interface Table Editor for the creation and modification of LT's. No changes were made to the Interface Table Editor when it was incorporated into the LTBLD to function as a Linkage Table Editor, so all capabilities and restrictions are as documented in the SDD volume I (rev. 1, sec 3.4), and will not be reproduced here. A list of prompts and commands will be included, however, for user convenience.

The function of the remaining portion of the LTBLD is to determine the source file to be used by the editor, and the destination file for the results of the editing process.

The editor can construct LT's from any of the following sources:

a. DDT

b. Default LT

c. User-owned LT

d. LT belonging to another user

It can then produce the following output LT files:

a. Default

b. User-owned default

c. User-owned nondefault

Due to the potentially large number of FORM, DDT, PROMPT, and LINKAGE TABLE files, a naming convention has been developed to minimize user typing and file-name confusion.

When a new document is to be added to the system, the originating organization assigns a document ID. This ID can be from one to six characters long. When the form and DDT for a new segment is developed, a similar ID is assigned to that segment. All subsequent user references to a document segment utilizes

these two ID's. In addition, each document and segment has a two-character ID associated with it for internal use by the program.

At the time that each document and segment is added to the system, the two-character document ID and a two-character segment ID is assigned. When a user signs on to the LTBLD and identifies the document and segment for which an LT is to be built, these two ID's are retrieved from the document and segment directories. They are subsequently used for validation of all input file requests, and for the creation of all default file names.

All PROMPT and LINKAGE TABLE files are constructed from three parts: (1) the first character, (2) the middle four characters, and (3) the last character.

The first character of all PROMPT file names is a "P," and for all LINKAGE TABLE files it is an "L." This character is provided by the program and is never input by the user.

The middle four characters of all default file names are the two-character ID's. For nondefault names, the user supplies those four characters.

The last character indicates ownership of the file, and is the one-character user ID (except for the PROMPT file and the default LINKAGE TABLE file, which is created directly from the DDT). These files are user-independent, and are indicated by an "*" for a last character.

In the normal operational environment, the LTBLD will use DDT's to produce default LT's. The other capabilities are available should they be needed, but will normally be functions performed by the DOC.

The user initiates the process by supplying his one-character user ID and the six-character document and segment ID's. He then must select either the DDT or an LT as the source file for the editor. If the DDT is selected, the DDT for the requested document segment is retrieved, and the data elements are placed into the default LT.

If the existing LT option is selected, the user is prompted for the input and output LT names. The responses are checked for validity, and the requested LT is retrieved from mass storage.

The Linkage Table Editor then prompts the user for any missing values. Values may be entered, changed, or deleted, according to the rules documented in SDD volume I (rev. 1, sec. 3.4). Note that only values are accessible to the user. The characteristics are obtained from the DDT, which is under configuration control, so changes to the characteristics are accomplished by altering the DDT and then recreating the default LT.

When the user indicates that he is finished with the editor, the new LT file is created, and its name is added to the directory. The program is then terminated.

7.3 ASSUMPTIONS AND LIMITATIONS

a. The requested document identifier must appear in the directory.

b. The requested segment identifier must appear in the directory, and must be associated with the requested document.

c. Any requested DDT must be present on the Daconics.

d. Any requested source LT must be present on the HP21MX.

e. The name of any requested source LT must exist in the LT directory, and the associated document and segment ID's must agree with the document and segment for which the current LT is to be used.

f. LT's may be created from any other LT (for that segment), including those owned by other users.

g. LT's may be recreated or purged only if they belong to the current user.

h. Any requested LT must have the same cycle number as the PROMPT file for that segment.

i. Any created LT's will have the user's ID appended as the last character of the file name except the default LT, which is user-independent.

7.4 PROGRAM INPUT/OUTPUT VARIABLES

Since LTBLD is a stand-alone program and does not execute under the FDS Executive, it does not have an associated interface table.

All inputs to the LTBLD are prompted inputs. When a "yes/no" response is required, the default response is always "yes" (i.e., a response of "space, carriage return" will result in the "yes" option being executed). All input errors will result in appropriate messages being displayed at the user's terminal, and the user is then given an opportunity to correct the erroneous input. The user may terminate a session at any point by responding to any prompt with a "%." Any nonrecoverable error conditions, such as encountering an error while reading a disk file, will result in a diagnostic message followed by termination of the run.

A list of prompts, acceptable responses, and the resulting action, is shown in table 7.4-I.

Since the Linkage Table Editor is identical to the FDS Interface Table Editor, it is not necessary to duplicate the entire editor documentation. However, for user convenience, the section that describes the editor prompts and user responses is duplicated in section 7.4.3.

TABLE 7.4-I.- PROGRAM SOLICITED (PROMPTED) INPUTS

PROGRAM LTBLD

Prompt	Meaning	Valid responses
ENTER DOCUMENT ID	List directory	? Six-character document ID
ARE YOU ADDING A NEW DOCUMENT TO THE SYSTEM?	Prompt for new document ID Reprompt	Yes No
ENTER NEW TWO-CHARACTER DOCUMENT ID		Two-character document code
ENTER DOCUMENT DESCRIPTION (UP TO 48 CHARACTERS)		Document description
ENTER SEGMENT ID	List directory	? Six-character document ID
ARE YOU ADDING A NEW SEGMENT TO THE SYSTEM?	Prompt for new segment ID Reprompt	Yes No
ENTER NEW TWO-CHARACTER SEGMENT ID		Two-character segment ID
ENTER SEGMENT DESCRIPTION (UP TO 48 CHARACTERS)		Segment description
USE DDT?	Create master default LT from DDT Use existing LT as the source	Yes No
DISPLAY DDT?		Yes No

TABLE 7.4-I.- Concluded

PROGRAM LTBLD

Prompt	Meaning	Valid responses
ENTER SOURCE LINKAGE TABLE NAME	Use default LT	Space - carriage return
	Use user-default LT	Space - user ID
	Use specified LT	Four-character LT name
	Use specified LT that belongs to a different user	Five-character LT name
	List directory	?
INPUT NEW LINKAGE TABLE NAME	Create master default table	Space - carriage return
	Create user-default table	Space - user ID
	Create named table	Four-character name
	List directory	?
DO YOU WANT TO RE-CREATE THIS LT	Purge existing LT and create new one with same name	Yes
	Reprompt user for new LT name	No
ENTER FILE DESCRIPTION (UP TO 48 CHARACTERS)		File description
DELETE FILE XXXXXX?	Delete source LT file	Yes
	Retain source LT	No
NOTE: In response to any prompt, a user input of a "%" will terminate the run.		

7.4.1 Displays

The only displays produced by the LTBLD are listings of directories, linkage tables, and DDT's. Directory listings are obtained by responding with a "?" when prompted for a document ID, segment ID, or linkage table name. Examples of a Document ID Directory, a Segment ID Directory, and a Linkage Table Directory are shown in figures 7.4.1-1, 7.4.1-2, and 7.4.1-3.

The linkage table listing is obtained by invoking the "LIST" command while executing the Linkage Table Editor.

A sample linkage table is shown in figure 7.4.1-4.

The DDT listing is obtained by responding with a null response when prompted by the program with the prompt "LIST DDT?" A sample DDT is shown in figure 7.4.1-5.

In addition to these displays, various messages are produced that inform the user of error conditions or the completion of significant tasks, such as the creation of a file. These messages are shown in table 7.4.1-I.

TABLE 7.4.1-I PROGRAM MESSAGE TABLE

PROGRAM LTBLD

MSG no.	Message ID block	Message text block and explanation
1		REQUESTED DOCUMENT DOES NOT EXIST.
2		DOCUMENT CODE XX ALREADY EXISTS. YOU MUST SELECT ANOTHER CODE.
3		REQUESTED SEGMENT DOES NOT EXIST FOR THIS SEGMENT.
4		SEGMENT CODE ALREADY EXISTS FOR THIS DOCUMENT. YOU MUST SELECT ANOTHER CODE.
5		YOU HAVE REQUESTED SOMEONE ELSE'S DEFAULT TABLE.
6		UNABLE TO OPEN LINKAGE TABLE DIRECTORY. IERR = XX (FATAL ERROR)
7		REQUESTED LT NAME NOT IN DIRECTORY.
8		REQUESTED LT IS NOT FOR REQUESTED DOCUMENT AND SEGMENT.
9		SEARCH FOR COMPATIBLE LINKAGE TABLE ABANDONED.
10		LINKAGE TABLE XXXXXX ALREADY EXISTS.
11		REQUESTED LT NAME DOES NOT PRESENTLY EXIST.
12		A LINKAGE TABLE NAMED XXXXXX ALREADY EXISTS FOR ANOTHER DOCUMENT AND SEGMENT. YOU MUST SELECT ANOTHER NAME.
13		YOU MAY NOT CREATE A TABLE WITH ANOTHER USER'S ID.
14		DACONICS INPUT FILE NAME IS XXXXXX:XXXXXX:DDT.

TABLE 7.4.1-I.- Concluded

PROGRAM LTBLD

MSG no.	Message ID block	Message text block and explanation
15		I/O ERROR IN DACIG. IERR = XX
16		FILE XXXXXX SUCCESSFULLY CREATED.
17		FILE XXXXXX BEING ADDED TO THE DIRECTORY.
18		FILE XXXXXX DELETED.

DOCUMENT ACRONYM	DOCUMENT ID	DOCUMENT TITLE
PRFP	PP	PRELIMINARY REFERENCE FLIGHT PROFILE
RFP	RP	REFERENCE FLIGHT PROFILE
CFP	CP	CONCEPTUAL FLIGHT PLAN
OFP	OP	OPERATIONAL FLIGHT PLAN

Figure 7.4.1-1.- Document ID Directory.

SEGMENT ACRONYM	DOCUMENT ACRONYM	SEG ID	SEGMENT TITLE
INTROD	PRFP	IN	INTRODUCTION
ASCENT	PRFP	AP	ASCENT PHASE
ONORBT	PRFP	OO	ON-ORBIT PHASE
RENDEZ	PRFP	RZ	RENDEZVOUS PHASE
DEORB	PRFP	DO	DE-ORBIT PHASE
TAB3I	PRFP	CT	CONSUMABLES TIMELINE TABLE

Figure 7.4.1-2.- Segment ID Directory.

NAME	DOC	SEG	V	DESCRIPTION
LPPAPG	PRFP	ASCENT PP AP	1	CONTAINS ALL LITERALS FOR TESTING DOC
LLITTG	PRFP	ASCENT PP AP	1	CONTAINS ALL LITERALS FOR TESTING DOC
PPPAP*	PRFP	ASCENT PP AP	1	PROMPT FILE FOR PRFP, ASCENT
LPPAP*	PRFP	ASCENT PP AP	1	EMPTY LINKAGE TABLE

INPUT NEW LINKAGE TABLE NAME :%
:

Figure 7.4.1-3.- Linkage Table Directory.

```
LPPAP*  - INCOMPLETE INTERFACE TABLE FOR VERSION    1 of PPAP*

  LSITET=    "ABCDEF            "
  TLO    =   A
  TLO    =   B
  RELVEL=    C
  TCLNCE=    D
  MAXQPR=    E
  TMAXQ =    F
  GDALT =    G
  ALPHA =      1.230000E+02
  BETA  =    H
  ERVEL =    I
  FLNAME=    040501
  LODATE=         9,      28,        77
  EVENT =    O
  GMT   =    P
  GET   =    U
  POSVEC=    V
  CMNTS =
```

Figure 7.4.1-4.- Linkage table listing.

DATA NAME	DATA TYPE	COLUMN FLAG	DIMENSION	FORMAT	DESCRIPTION
LSITETC	18		1	A3	LAUNCH SITE
TLO R			1	T0:0:0:3.1	TIME OF LIFTOFF (GET)
TLO R			1	T0:0:0:3.1	TIME OF LIFTOFF (GET)
RELVELR			1	I3	RELATIVE VELOCITY
TCLNCER			1	T0:0:0:3.1	TIME OF TOWER CLEARANCE
MAXQPRR			1	I3	MAXIMUM DYNAMIC PRESSURE
TMAXQ R			1	T0:0:0:4.1	TIME OF MAXIMUM DYNAMIC PRESSURE
GDALT R			1	I5	GEODETIC ALTITUDE
ALPHA R			1	F5.1	FLIGHT PATH ANGLE
BETA R			1	F4.1	ANGLE OF SIDESLIP
ERVEL R			1	I4	EARTH RELATIVE VELOCITY
FLNAMEC	6		1	A5	FLIGHT NAME
LODATEI			3	I2	DATE OF LIFTOFF
EVENT C	36	*	10	A36	EVENT NAME
GMT R			10	T0:2:2:2	GMT OF EVENT
GET R			10	T0:2:2:2	GET OF EVENT
LAT R			10	F6.1	GEODETIC POLAR POSITION VECTOR
LONG R			10	F6.1	GEODETIC POLAR POSITION VECTOR
ALT R			10	F6.1	GEODETIC POLAR POSITION VECTOR
CMNTS C	18	*	10	A12	COMMENTS

INPUT NEW LINKAGE TABLE NAME :

Figure 7.4.1-5.- DDT for document PRFP, segment ascent.

7.4.2 File Outputs

The LTBLD produces LINKAGE TABLE files and PROMPT files. Since these files are only used internally by the documentation processor, and are not in a format that is directly usable by a user, their format is documented in the documentation for the subroutines which produce them.

In addition, entries are made in the appropriate directories as documents, segments, and linkage tables, and are added to the system. The formats of these directories are shown in section 7.4.1.

7.4.3 Linkage Table Editor Prompts

a. Syntax

(1) \:(user response)Δ

(2) \parameter keyword $\left\{=\right\}$:(user response)Δ

(3) \parameter keyword = $\left\{\begin{array}{l}\text{label[(sub,[sub])]}\\ \text{literal list}\end{array}\right\}$: (user response)Δ

b. Purpose - The table edit directives allow the FDS user to enter new parameter values or labels, and/or to modify existing values or labels in a specified linkage table. When the editor is entered, the user is prompted with syntax 2. The user is normally prompted only for those parameters with either missing or incomplete data. However, the user can control the prompt mode with the PROMPT directive (SDD vol. I, rev. 1, sec. 3.4.4). If any other prompt mode is selected, the user will be prompted with either syntax 2 or syntax 3, depending upon whether or not the parameter being displayed has associated values or labels.

c. Definitions

(1) User responses for syntax (1)

$$\left\{\begin{array}{l}\text{executive responses}\\ \text{editor directives}\\ \\ \text{parameter keyword} = \left\{\begin{array}{l}\&\\ \text{label[(sub,[sub])]}\\ \text{literal list}\end{array}\right\}\end{array}\right\}\Delta$$

(a) Executive responses - A one-character response causing executive action. The options are as follows:

(?)

A list of all available Linkage Table Editor directives is displayed. (Note: For syntax 2 and syntax 3, this response will cause the syntax of the appropriate statement to be displayed.)

(%)

Abnormally terminates the Linkage Table Editor, and all modifications are discarded.

(Δ)

A carriage return only; causes no change. If in response to syntax 1, the user will be reprompted with the (\:) symbol. If in response to either syntax 2 or syntax 3, the next applicable parameter will be prompted.

(\)

A user request to be prompted with only the (\:) symbol (syntax 1). This action removes the user from the editor prompting mode, and allows the modification of any selected parameter in the table or use of any editor directive. To be placed back into the editor prompt mode, the user must use the PROMPT directive (SDD vol. I, rev. 1, sec. 3.4.4).

(b) Editor directives - See sections 3.4.3, 3.4.4, and 3.4.5 of SDD volume I, rev. 1.

(c) Parameter keyword - An alphanumeric name of up to six characters identifying one of the parameters in the linkage table being edited. The keyword must match an existing keyword in the table or no action is taken and the user is reprompted with the (\:) symbol.

The (=) symbol is used to denote an input parameter.

The field, label [(sub,[sub])], is an alphanumeric name of up to six characters identifying a data element to be located at processor execution time and used as a set of input parameters. The label may be optionally subscripted, in which case the subscript identifies the initial array position to find the first input value, or to place the first output value. Subsequent values are found or are placed in the following contiguous array positions. The data elements referenced by the label will be checked at execution time to ensure that they are of the correct data type and of sufficient size to satisfy the parameter usage. If either the size or type checks fail, the user is notified, processing is terminated, and the user is reprompted with the (%:) symbol. Note that subscripting will be accomplished using the type and dimension attributes of the data element existing in the AWA at execution time. Data

elements not then in existence may not be used as processor inputs, or for processor outputs if subscripted.

The literal list - allows the FDS user to assign values to parameters. The literal list is defined as:

```
{ {[(sub,[sub])] value        }
{ {[(sub,[sub])] repeat list  }
{ 'symbolic string'

[ {[(sub,[sub])] value        }...]}Δ
[,{[(sub,[sub])] repeat list  }   ]
```

where:

[(sub,[sub])]value - an immediate input data value to be entered into the linkage table for this parameter. Values to be input must be of the predefined type for the parameter. If not, no action is taken, the user is notified and reprompted. The valid data value types are:

- Integers consisting of numerical digits and prefixed algebraic sign (e.g., 1, -7, +17)

- Single-precision real numbers consisting of numerical digits, a prefixed sign, and a decimal point; may also contain an embedded sign and the letter E (e.g., -6.213, -6.213E+01, +7.2475E-03)

- Double-precision real numbers consisting of numerical digits, a prefixed sign, a decimal point, an embedded sign, and the letter D (e.g., +17.3241D+03, -4.8245D-07

- Character data consisting of alphanumeric data enclosed by quotation marks ("). (Note: quotation marks within the character data are not permitted.) If the number of characters supplied is greater than the permissible number for that parameter, no action is taken, the user is notified and reprompted. If the number of characters supplied is less than the required number, the supplied data are left adjusted in the data string, and the remainder of the data string is padded with blanks. Permissible data string lengths are 2, 6, 18, 36, or 72 characters (e.g., "AB", "XY17BC"). A parameter may consist of an array of character strings of any combination of the permissible data string lengths. Length checking and blank propagation is done only within a string, and is not performed on an array basis.

- A null character (&) causes the existing value for this parameter to be set to zero, and the parameter is considered to be incomplete.

-A mixed type is a combination of any of the above data types occurring in a fixed, predefined pattern.

- A free type is a random combination of any of the above data types. No error checking occurs for a free data type.

All or part of a parameter data array may be specified by the [(sub,[sub])] value. If the value is optionally subscripted (preceded by a numerical digit(s) enclosed in parentheses, e.g., (3)), the value will be placed in the position of the parameter data array indicated by the subscript. Subsequent values in the list will be placed in the following contiguous locations unless another subscript is encountered. (Note: Arrays may be singly or doubly dimensioned, but the subscript values must be integers.) If either the subscript is greated than the maximum allowable element in the list or the subsequent list of values extends beyond the allowable list, no action is taken, the user is notified and reprompted with the (\:) symbol.

[(sub,[sub])] repeat list - the repeat list is defined as:

$$\left\{ \begin{array}{l} nR \left\{ \begin{array}{l} \text{value} \\ \text{repeat list} \end{array} \right\} \\ \\ nR \left(\left\{ \begin{array}{l} \text{value} \\ \text{repeat list} \end{array} \right\} \left[, \left\{ \begin{array}{l} \text{value} \\ \text{repeat list} \end{array} \right\} \cdots \right] \right) \end{array} \right\}$$

The repeat list allows the FDS user to repeat a set of values a specified number of times. The subscript, if present, indicates the starting position in the parameter data array, the same as for the [(sub,[sub])] value. A repeat list is specified by a numerical value suffixed with the letter R (e.g., 3R). A repeat list may be a single value, a list of values, and/or other repeat lists. If more than one value appears in the repeat list, the list must be enclosed by parentheses (e.g., 3R5, 3R(5, 7, "AB")). An entire repeat list may be subscripted, but no subscripts may appear embedded within the repeat list.

If the entry of a literal list takes more space than is available on one terminal line, the user must end the line at a component value (e.g., a character string, real number, etc., cannot be started on one line and completed on the next) followed by a carriage return. The Linkage Table Editor will then reprompt the user with the parameter keyword and the indices of the next incomplete array location. For example, given the parameter ABC, which requires five fields, the input might look like the following:

```
\ABC=:"ABCDEF",1.0,3.4Δ
\ABC=(4):1.75,"END"Δ
```

Note: The mixing of values and labels is not meaningful and is prohibited. If a label is provided, only one label (not a list of labels) is allowed, and the parameter is considered to be complete. For literal arrays, all elements must be complete before the parameter is considered complete.

The symbol & indicates that any existing label for this parameter is removed, and the parameter is considered as incomplete. If used in conjuction with a literal list, it causes individual elements within the list to be set equal to zero (blanks in the case of Hollerith strings), and the parameter keyword marked as incomplete (in this context, & may be used with subscripts and repeat fields).

The 'symbolic string' is a processor-dependent string of symbols to be evaluated/interpreted by the processor at execution time (e.g., expressions and conditions for certain utility processors, SDD vol. II/III (to be published)). This string of symbols is lexically analyzed (see SDD vol. IV) by the Linkage Table Editor and passed to the processor as an encoded buffer within the interface table. A symbolic string may not be subscripted, be part of a mixed type, or be contained in a repeat list.

If a symbolic string is longer than a terminal line, the user may terminate the line with a carriage return at any component boundary (e.g., a real number, label, etc., may not be split over two lines). The user will then be reprompted and allowed to continue the input string. For example

```
\EXP=:'TABOUT(I)=(TAB(I)/64)*3+2*Δ
\CONTINUE:SIN(THET(I))+E**2;I=1,3'Δ
```

(2) User response for syntax (2) or (3)

```
{executive responses}
{label [(sub,[sub])]}
{literal list       }
{&                  }
```

See definitions under syntax (1) response for an explanation of these terms.

7.5 LINKAGE TABLE BUILD PROGRAM (LTBLD) SUBROUTINES

In addition to those subroutines that follow, LTBLD uses several subroutines that are shared with the documentation processor. Instead of duplicating the documentation for those subroutines here, the reader is referred to section 8.0 of this document.

The subroutines are:

DACIG
DREAG
WRITG
LTNMG
LTNNG
DOFEG
LTMAG
LTPRG
LTPFG
LTMOG
LTIFG
LTOIG
LTXIG
LPRMG

7.5.1 Program Name - Main Program LTBLD

7.5.1.1 Purpose

Due to the memory size restriction on the HP21MX computer, the Linkage Table Build Program is a segmented program. The principal purpose of LTBLD is to serve as the master segment, and to direct flow through the lower segments that actually perform the tasks required for the creation and modification of linkage tables.

7.5.1.2 Functional Description

Before the first segment is called, RMPAR is called to obtain the user's terminal logical unit number and the user's one-character ID. These are stored in common for use by the other segments. The security code and cartridge number for the directory files are also stored in common, along with the security code and cartridge number for the LINKAGE TABLE and PROMPT files. One additional quantity is also stored in common that tells the lower segments that they are being called by LTBLD. Most of the subroutines that are used by LTBLD are also used by DOC. Of these, most are identical for both uses; however, a few have some restrictions when used by DOC that they do not have when used by LTBLD. To avoid the creation of an excessive number of nearly identical subroutines, the differences are incorporated as options and are triggered by the variable PROFLG, which tells the subroutines which program is calling them.

After each segment is called, an error flag is interrogated. If it has been turned on by the segment, the run is terminated immediately without calling the remaining segments.

7.5.1.3 Assumptions and Limitations

The only assumption made by LTBLD is that the user has input his user ID as part of the run command. However, if this was not done, the first segment will detect this omission and prompt the user for this information.

7.5.1.4 Method

None.

7.5.1.5 Routine Input/Output Variables

The LTBLD input/output variables are presented in table 7.5.1-I.

7.5.1.6 Functional Logic Flow

The functional logic flow for LTBLD is presented in figure 7.5.1-1.

7.5.1.7 Diagnostics and Debug

None.

7.5.1.8 Special Comments

None.

7.5.1.9 References

None.

TABLE 7.5.1-I.- INPUT/OUTPUT VARIABLES

Routine LTBLD

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LU		Intg	I		SP		Unit number of user terminal.
USERID		Intg	I		SP		One-character user ID.
CARTRG		Intg	O		C		Cartridge containing LINKAGE TABLES and PROMPT files.
ISECU		2CH	O		C		Security code for LINKAGE TABLE and PROMPT files.
JCR		Intg	O		C		Cartridge containing directories.
JSEQ		2CH	O		C		Security code for directories.
PROFLG		Intg	O		C		Processor flag; set = 0
ISW		Intg	O		C		Set = 1 for first call to LINLG, = 2 For second call
XE(1)		Intg	I		C		Segment error flag.

NOTES:

SP = System Parameter

TYPE

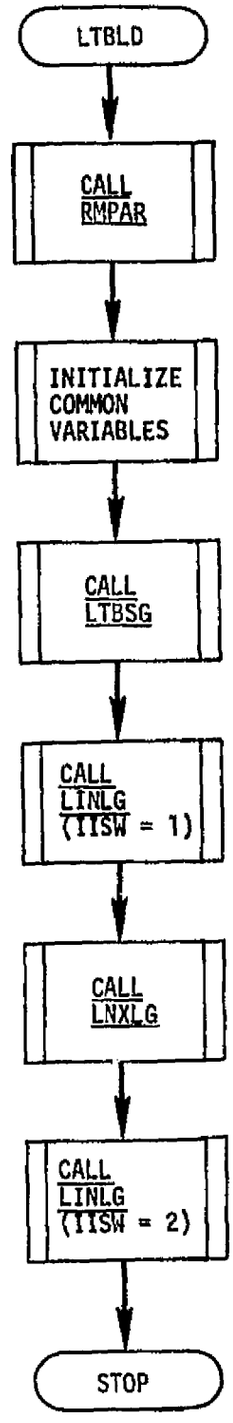
Free	Dubl	18CH	Mix
Intg	2CH	36CH	Char
Real	6CH	72CH	Bin

USE

I = Input
O = Output
I/O = Input/Output

SOURCE

IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 7.5.1-1.- LTBLD functional logic flow.

7.5.2 Program Name - Main Program LINLG

7.5.2.1 Purpose

The sole purpose of LINLG is to serve as the main program of the segment containing subroutine LTMAG. LTMAG is used by both LTBLG and DOC. However, the main program of any segment must contain the name of the master segment that called it. Therefore, to avoid duplicating LTMAG with that single difference, the "do-nothing" program LINLG was written, which simply calls LTMAG then returns to LTBLG. A similar routine also exists for DOC.

There are no inputs, no outputs, no internal variables, and no logic structure.

7.5.2.2 Functional Description

None.

7.5.2.3 Assumptions and Limitations

None.

7.5.2.4 Method

None.

7.5.2.5 Routine Input/Output Variables

None.

7.5.2.6 Functional Logic Flow

The functional logic flow for LINLG is presented in figure 7.5.2-1.

7.5.2.7 Diagnostics and Debug

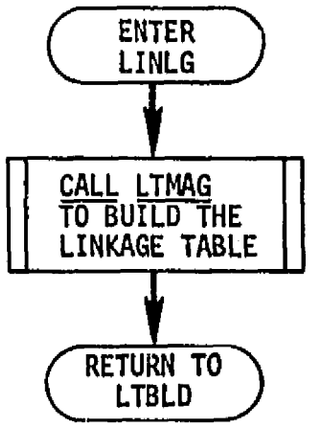
None.

7.5.2.8 Special Comments

None.

7.5.2.9 References

None.



Page 1 of 1

Figure 7.5.2-1.- LINLG functional logic flow.

7.5.3 Subroutine Name - LTBSG

7.5.3.1 Purpose

Subroutine LTBSG is responsible for determining the name of the linkage table to be produced, the document segment with which the table is associated, and the source of information with which the table is to be created. A linkage table can either be created from an existing linkage table, or directly from a DDT. LTBSG obtains the six-character document and segment identifiers that are used in directory searches, and to construct the name of the DDT file on the Daconics. It also obtains the two-character document and segment codes that are used for the creation of linkage table names, and for validating that a requested linkage table is for the correct document segment.

7.5.3.2 Functional Description

The first task of LTBSG is to check for the existence of a valid user ID. In the event that the user failed to enter his ID correctly, LTBSG will detect this and issue a prompt to the user. Subroutine DOSEG is then called to process the user document and segment inputs. After DOSEG has stored the document and segment ID's and the associated two-character codes, LTBSG asks the user whether he wants to use a DDT or another linkage table as the source of information with which to create the new linkage table. If the DDT option is selected, subroutine DACIG is called to retrieve the appropriate DDT from the Daconics. The user is then given the opportunity to list the DDT. If the linkage table option is selected, subroutine LTNMG is called to process the input linkage table name request. The final task is to call subroutine LTNNG to obtain the name of the linkage table that is to be created.

7.5.3.3 Assumptions and Limitations

If the DDT input option is selected, LTBSG assumes that the appropriate steps have been taken to activate the communications interface between the HP21MX and the Daconics, and that the desired DDT is present on the Daconics on-line storage. In the event that either of these conditions are not met, an appropriate message is sent to the user and the session is terminated.

7.5.3.4 Method

None.

7.5.3.5 Routine Input/Output Variables

The LTBSG input/output variables are presented in table 7.5.3-I.

7.5.3.6 Functional Logic Flow

The functional logic flow for LTBSG is presented in figure 7.5.3-1.

7.5.3.7 Diagnostics and Debug

Diagnostic	Meaning	Result
DDT cannot be retrieved.	Either communications channel is not active, or requested DDT is not on on-line storage on Daconics.	Abort

7.5.3.8 Special Comments

None.

7.5.3.9 References

None.

TABLE 7.5.3-I.- INPUT/OUTPUT VARIABLES

Routine LTBSG

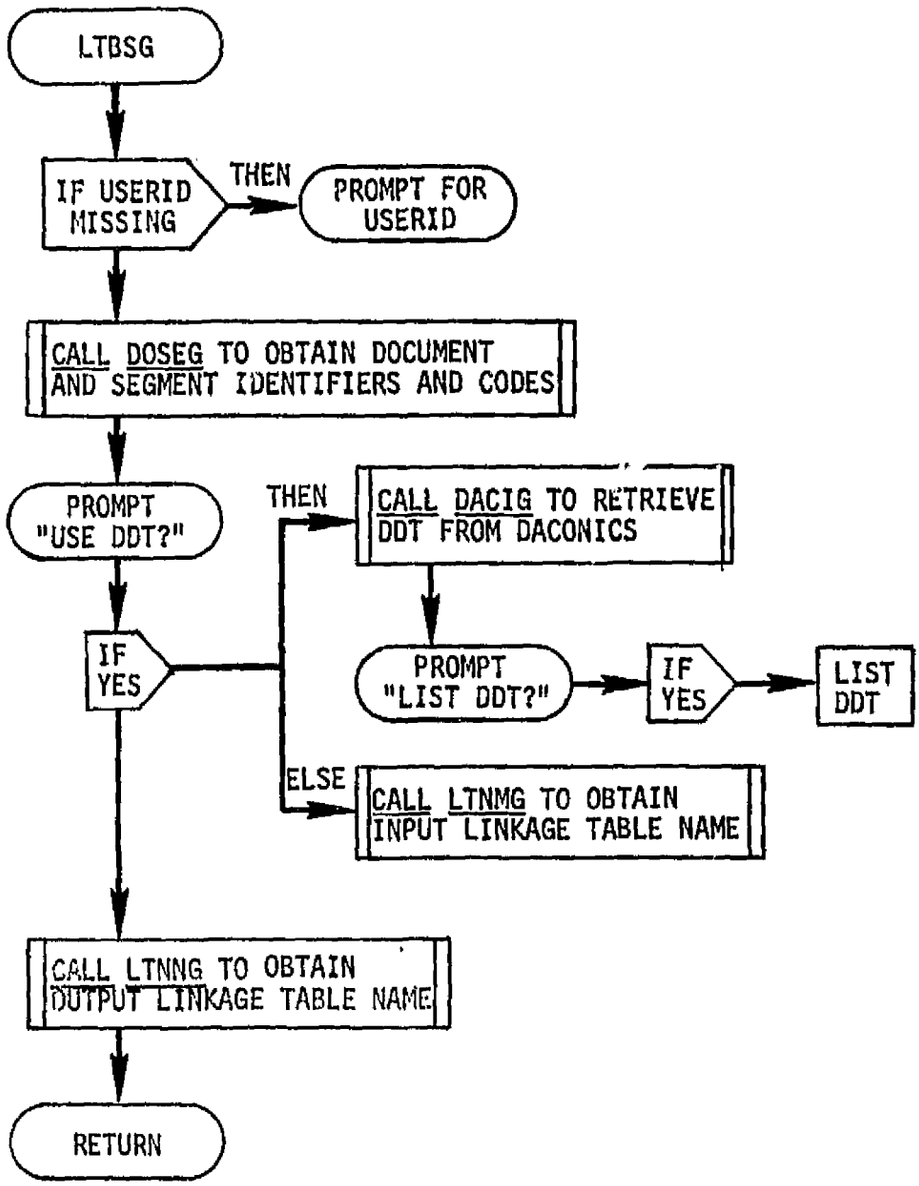
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
DOCID		2CH	I/O		C		Two-character document code.
DOCNAM		6CH	I/O		C		Six-character document identifier.
LTNAM		6CH	I/O		C		Input linkage table name.
LU		Intg	I		C		User terminal logical unit.
NLTNAM		6CH	I/O		C		Output linkage table name.
SEGID		2CH	I/O		C		Two-character segment code.
SEGNAM		6CH	I/O		C		Six-character segment identifier.
USERID		Intg	I/O		C or T		One-character user ID.

NOTES:

TYPE
Free, Intg, Real, Dubl, 2CH, 6CH, 18CH, 36CH, 72CH, Mix, Char, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 7.5.3-1.- LTBSG functional logic flow.

7.5.4 Program Name - Main Program LNXLG

7.5.4.1 Purpose

The sole purpose of LNXLG is to call subroutine LTXIG and to return to LTBLD. LTXIG is used by both LTBLD and DOC. However, the main program of a segment must contain the name of the master segment that called it. Therefore, to avoid duplicating LTXIG with that single difference, LNXLG was written to call LTXIG and return to LTBLD. A similar routine exists for DOC.

There are no inputs, no outputs, no internal variables, and no program logic.

7.5.4.2 Functional Description

None.

7.5.4.3 Assumptions and Limitations

None.

7.5.4.4 Method

None.

7.5.4.5 Routine Input/Output Variables

None.

7.5.4.6 Functional Logic Flow

The functional logic flow for LNXLG is presented in figure 7.5.4-1.

7.5.4.7 Diagnostics and Debug

None.

7.5.4.8 Special Comments

None.

7.5.4.9 References

None.

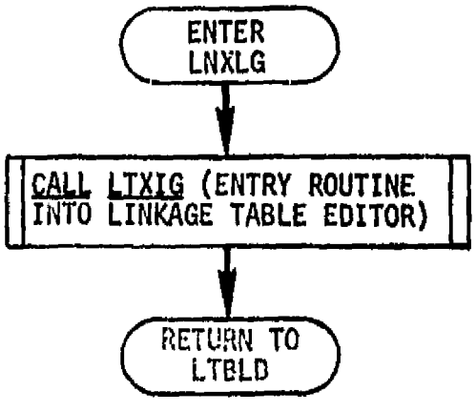


Figure 7.5.4-1.- LNXLG functional logic flow.

8.0 DOCUMENTATION PROCESSOR (DOC)

8.1 PURPOSE

The purpose of the documentation processor is to minimize mission planning documentation time and cost through the semiautomated production of standardized documents. Mission planning data from the AWA are combined with blank forms from the Daconics to form segments of the desired document. These segments are then combined with the required plots and figures to produce the complete document.

8.2 FUNCTIONAL DESCRIPTION

The operation of the DOC is divided into three distinct functions. These are linkage table editing, file management, and document processing. The editing function allows the user to define data element names and values within the linkage tables. This function is performed by a copy of the standard FDS Interface Table Editor.

The file management function consists of the construction of file names, the creation, storage, retrieval, and purging of files, and the maintenance of file directories. The files that are used by the DOC are LINKAGE TABLE files, PROMPT files, FORM files, and DDT files. The FORM and DDT files reside on the Daconics system. In addition to these files, DIRECTORY files are maintained for document names, segment names, and LINKAGE TABLE and PROMPT file names.

The actual document processing function consists of the following steps. The first "PAGE" of the blank form is moved into an internal buffer. In this context, a page is defined to be that portion of a form that will fit in the buffer. The required data element values are then retrieved, either from the AWA or the literal area of the LT. These values are converted and formatted according to the type and format specifications contained in the DDT, and are then merged into the reserved spaces in the current page of the form. This process is repeated until the entire form has been completed. The resulting document segment is then transmitted back to the Daconics for printing.

8.3 ASSUMPTIONS AND LIMITATIONS

a. All required data elements must either have been previously computed and placed into the AWA, or must have been input by the user into the literal area of the linkage table.

b. No units conversions are performed by DOC, with the single exception of time.

c. Variable-length tabular output is permitted. However, the form must contain enough lines of blank spaces to accommodate the worst case, and there must be no data fields following a variable-length table within a segment.

d. The Daconics HP21MX computer must be up, and the connecting data channel must be active.

e. The required Daconics files must be loaded on the working disk.

f. There can be no more than 64 separate data elements in the LT. However, any of the data elements may be dimensioned.

g. The version numbers of the FORM, DDT, LT, and PROMPT files must all agree.

h. The user cannot delete files from the Daconics system.

8.4 PROCESSOR INPUT/OUTPUT

a. Processor interface table - The FDS-1 version of the documentation processor does not use an interface table. All inputs are via user prompts.

b. Interface table data file definitions - none.

c. Processor solicited (prompted) inputs - The processor solicited inputs are provided in table 8.4-I, and are shown pictorially in figure 8.4-1.

d. Processor displays and display parameter definition tables - DOC produces five displays and one tabular listing. All displays are optional, and are produced only upon request from the user. Three directory listings and three display parameter definitions are shown in table 8.4-II(a) through (f). These are the Document ID Directory (table 8.4-II(a)), Segment ID Directory (table 8.4-II(c)), and the LT ID Directory (table 8.4-II(e)). The display parameter definitions table for the Document ID Directory are shown in table 8.4-II(b), the Segment ID Directory is shown in table 8.4-II(d), and the Linkage Table Directory is shown in table 8.4-II(f). The other two displays are the blank form display and the merged segment display. These are identical in format, differing only by the presence or absence of the merged data. The remaining display is a listing of the data elements as they are retrieved and processed.

   The blank form display, merged segment display, and data element listing have no format inasmuch as their formats are determined by the material being processed. Examples of these displays are shown in the test report.

e. Processor message table - Table 8.4-III contains the processor messages that may be displayed for the user. Some indicate error conditions, and are followed by a user prompt to allow the user to take corrective action. Others simply inform the user as to the status of the processing. The type of message is apparent from its content.

TABLE 8.4-I.- PROCESSOR SOLICITED (PROMPTED) INPUTS

PROCESSOR DOC

Prompt	Meaning	Valid responses
ENTER DOCUMENT ID	List directory	? Six-character document name
ENTER SEGMENT ID	List directory	? Six-character segment name
ENTER SOURCE LINKAGE TABLE NAME	List Directory	?
	Default LT	Space - carriage return
	User default LT	Space - user ID
	User-owned LT	Four-character LT name
NOTE: Next prompt comes from Linkage Table Editor. See section 8.0 for description of prompts and responses.	Nonuser-owned LT	Five-character LT name
BUILD NEW LINKAGE TABLE?	Build new table	Yes
	Do not build	No
ENTER NEW LINKAGE TABLE NAME	List directory	?
	Create user default	Space - user ID
	Create nondefault	Four-character LT name
DO YOU WANT TO RECREATE THIS TABLE?	Delete existing table and replace it with new one with same name.	Yes
	Discard LT mods and terminate run	No

TABLE 8.4-I.- Continued

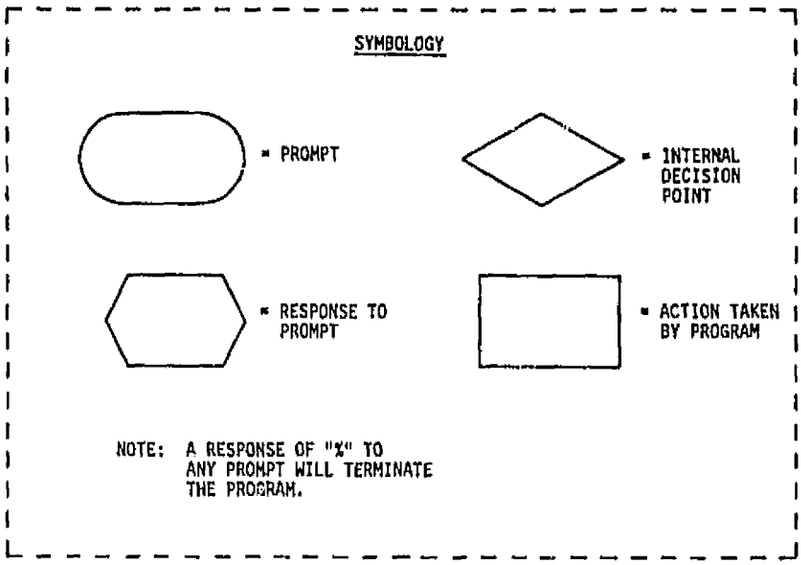
PROCESSOR DOC

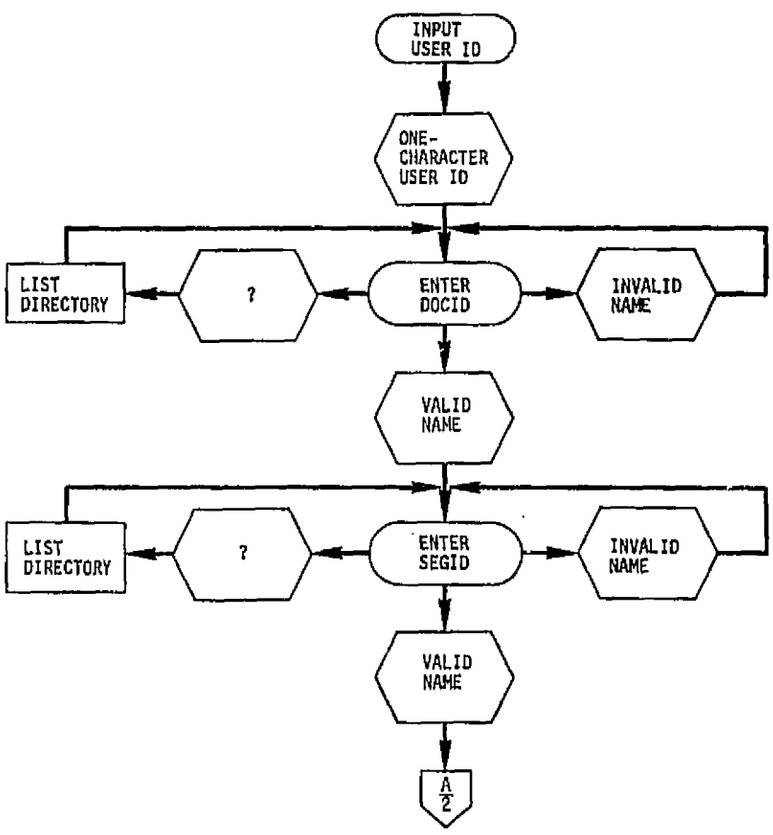
Prompt	Meaning	Valid responses
DISPLAY BLANK FORM?	Display No display	Yes No
ENTER HARDCOPY UNIT NUMBER, IF DESIRED	Display at terminal Print form on specified unit	Space - carriage return Unit number
DISPLAY DATA ELEMENTS?	List data element names, formats, raw values, and converted values as they are processed Do not list	Yes No
ENTER HARDCOPY UNIT NUMBER, IF DESIRED	Display at terminal List data on specified unit	Space - carriage return Unit number
DISPLAY MERGED SEGMENT	Display Do not display	Yes No
ENTER HARDCOPY UNIT NUMBER, IF DESIRED	Display at terminal Print segment on specified unit	Space - carriage return Unit number
SEND SEGMENT TO DACONICS?	Create output file on Daconics Terminate run	Yes No

TABLE 8.4-I.- Concluded

PROCESSOR DOC

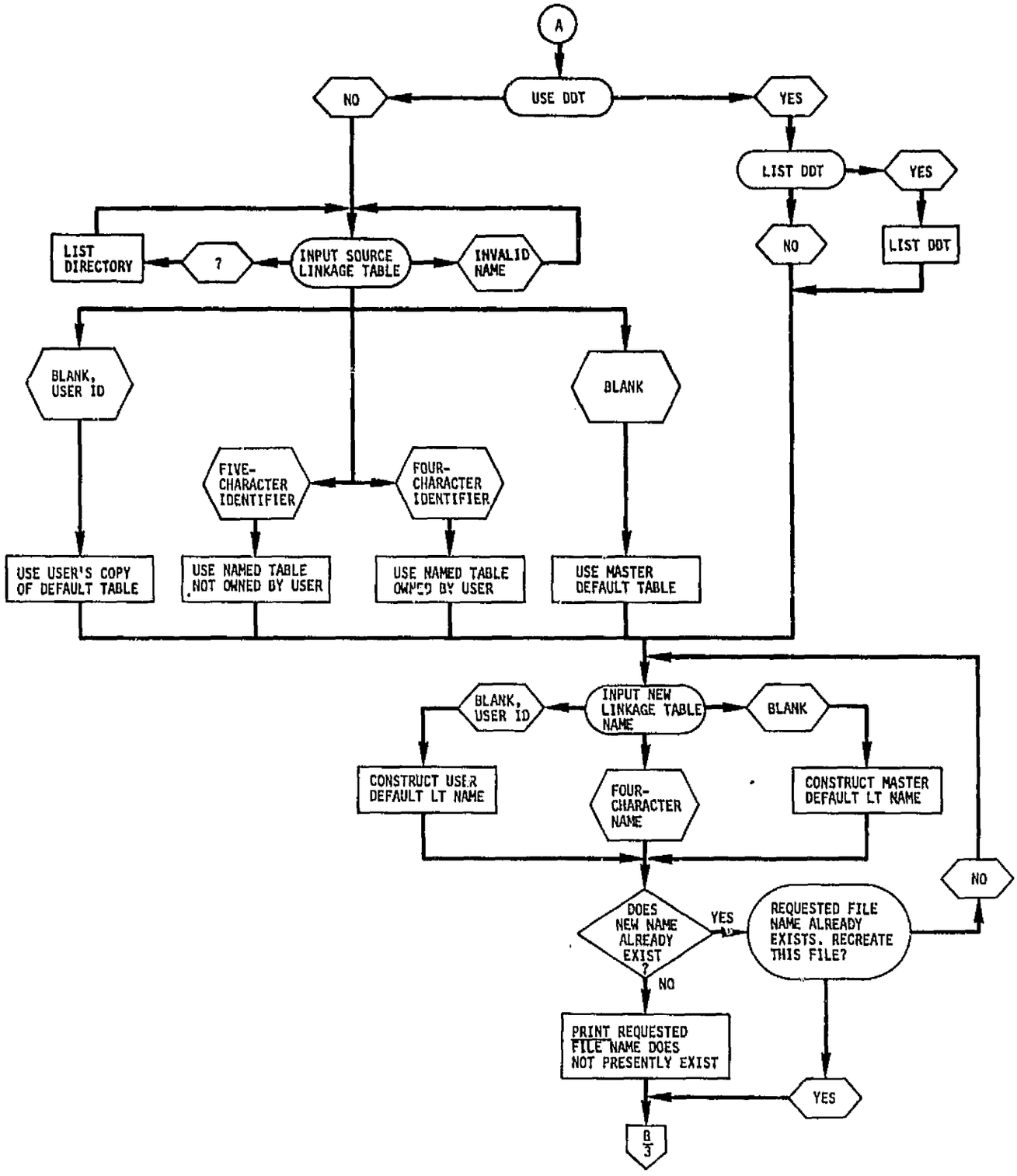
Prompt	Meaning	Valid responses
ENTER FOUR-CHARACTER FILE IDENTIFIER IF DESIRED	Use default name	Space - carriage return
	Append identifier onto default name	Unique identifier (may be one through four characters)
WOULD YOU LIKE TO LOOK AT THE NEW DACONICS FILE?	Recall segment from Daconics and display at terminal	Yes
	Terminate run	No
NOTE: In response to any prompt, a user input of a "%" will terminate the run.		





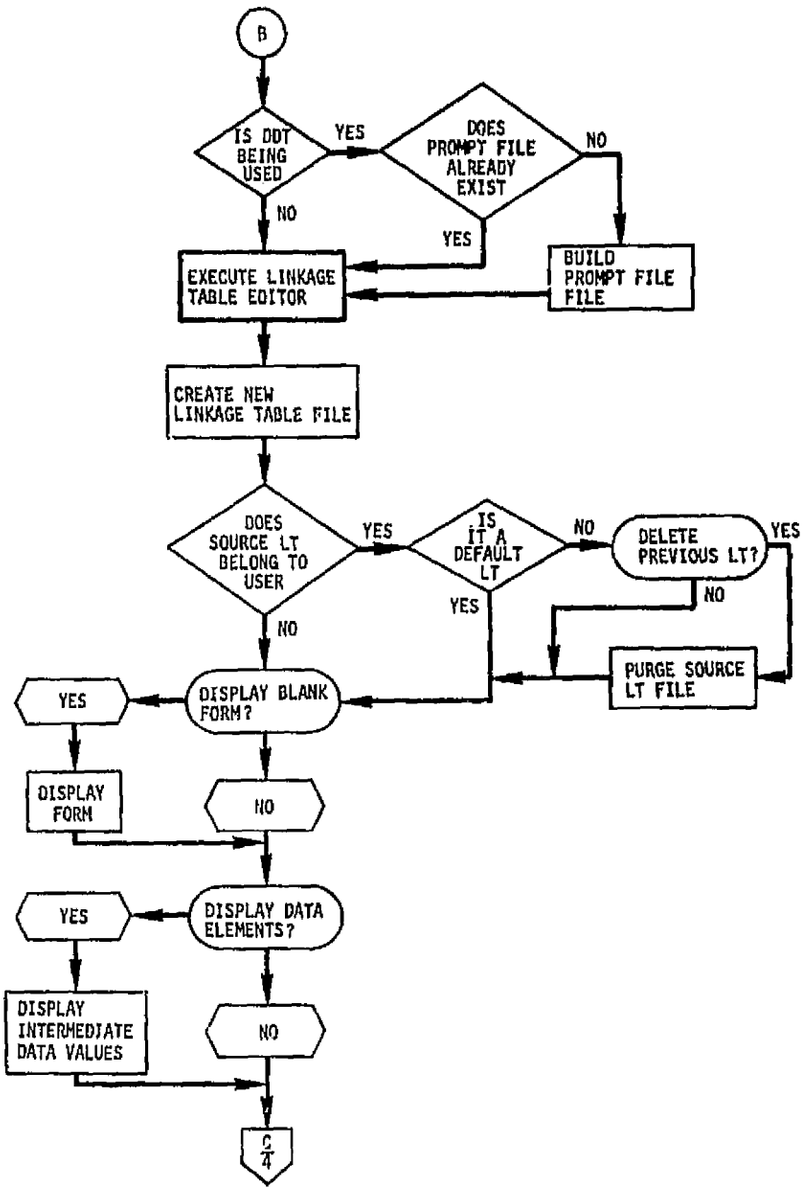
Page 1 of 4

Figure 8.4-1.- Logic diagram of the DOC processor solicited inputs.



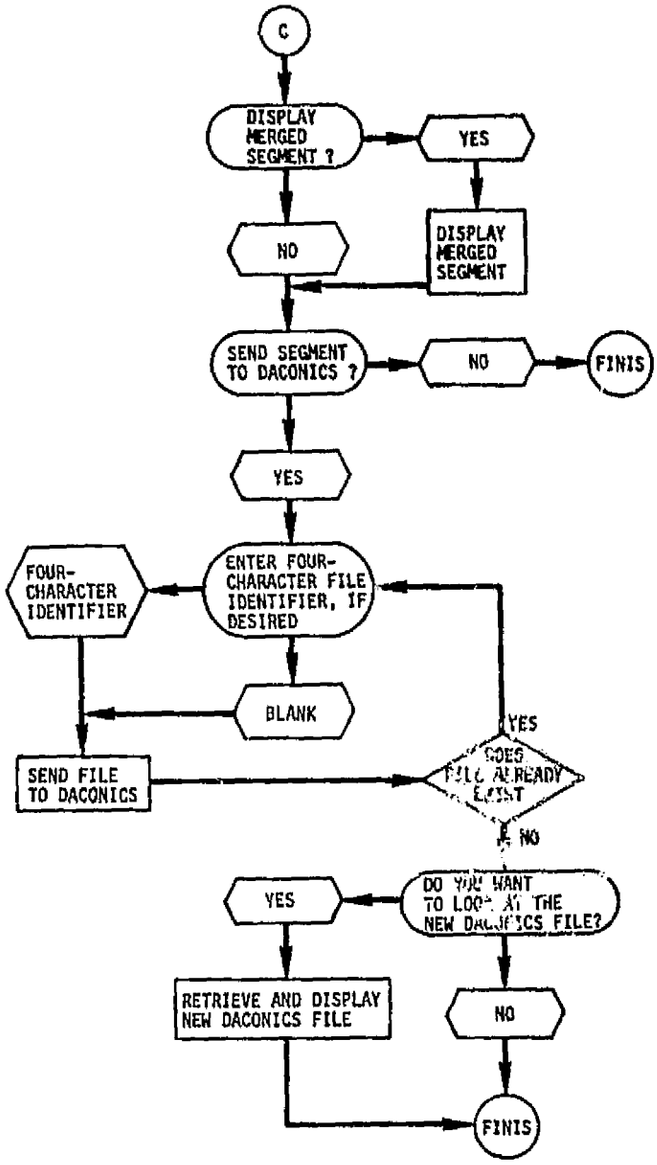
Page 2 of 4

Figure 8.4-1.- Continued.



Page 3 of 4

Figure 8.4-1.- Continued



Page 4 of 4

Figure 8.4-1.- Concluded.

TABLE 8.4-II.- PROCESSOR DISPLAY FORMAT

(a) Document ID Directory

PROCESSOR DOC

```
     1   5    10   15   20   25   30   35   40   45   50   55   60   65   70
 1                              DOCUMENT I.D. DIRECTORY

    NAME   I.D.                 DESCRIPTION

 5  αααααα αα   αααααααααααααααααααααααααααααααααααααααα
    αααααα αα   αααααααααααααααααααααααααααααααααααααααα
    αααααα αα   αααααααααααααααααααααααααααααααααααααααα
    αααααα αα   αααααααααααααααααααααααααααααααααααααααα
    αααααα αα   αααααααααααααααααααααααααααααααααααααααα
10  αααααα αα   αααααααααααααααααααααααααααααααααααααααα
    αααααα αα   αααααααααααααααααααααααααααααααααααααααα
    αααααα αα   αααααααααααααααααααααααααααααααααααααααα
    αααααα αα   αααααααααααααααααααααααααααααααααααααααα
    αααααα αα   αααααααααααααααααααααααααααααααααααααααα
15

20


24
         5    10   15   20   25   30   35   40   45   50   55   60   65   70
```

TABLE 8.4-II.- Continued

(b) Display parameter definition table for the Document ID Directory

PROCESSOR DOC

DOCUMENT I.D. DIRECTORY	
Display parameter label	Parameter definition
NAME	Six-character document identifier provided by document designer.
I.D.	Associated two-character document I.D. used internally by DOC.
DESCRIPTION	Forty-character document description.

TABLE 8.4-II.- Continued

(c) Segment ID Directory

PROCESSOR ____DOC____

```
SEGMENT ID DIRECTORY

 SEG      DOC      I.D.                  DESCRIPTION

 aaaaaa   aaaaaa   aa     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 aaaaaa   aaaaaa   aa     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 aaaaaa   aaaaaa   aa     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 aaaaaa   aaaaaa   aa     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 aaaaaa   aaaaaa   aa     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 aaaaaa   aaaaaa   aa     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 aaaaaa   aaaaaa   aa     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 aaaaaa   aaaaaa   aa     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 aaaaaa   aaaaaa   aa     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
 aaaaaa   aaaaaa   aa     aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

TABLE 8.4-II.- Continued

(d) Display parameter definition table for the Segment ID Directory

PROCESSOR DOC

SEGMENT I.D. DIRECTORY	
Display parameter label	Parameter Definition
SEG	Six-character segment identifier provided by segment designer.
DOC	Six-character identifier of the document of which this segment is a part.
I.D.	Two-character identifier used internally by DOC.
DESCRIPTION	Forty-character segment description.

TABLE 8.4-II.- Continued

(e) Linkage Table Directory

PROCESSOR ____DOC____

LINKAGE TABLE DIRECTORY

NAME	DOC	SEG	D	S	V	DESCRIPTION
αααααα	αααααα	αααααα	αα	αα	X	αααααααααααααααααααααααααααααααααααααααα
αααααα	αααααα	αααααα	αα	αα	X	αααααααααααααααααααααααααααααααααααααααα
αααααα	αααααα	αααααα	αα	αα	X	αααααααααααααααααααααααααααααααααααααααα
αααααα	αααααα	αααααα	αα	αα	X	αααααααααααααααααααααααααααααααααααααααα
αααααα	αααααα	αααααα	αα	αα	X	αααααααααααααααααααααααααααααααααααααααα
αααααα	αααααα	αααααα	αα	αα	X	αααααααααααααααααααααααααααααααααααααααα
αααααα	αααααα	αααααα	αα	αα	X	αααααααααααααααααααααααααααααααααααααααα
αααααα	αααααα	αααααα	αα	αα	X	αααααααααααααααααααααααααααααααααααααααα
αααααα	αααααα	αααααα	αα	αα	X	αααααααααααααααααααααααααααααααααααααααα
αααααα	αααααα	αααααα	αα	αα	X	αααααααααααααααααααααααααααααααααααααααα

TABLE 8.4-II.- Concluded

(f) Display parameter definition table for the Linkage Table Directory

PROCESSOR DOC

LINKAGE TABLE DIRECTORY	
Display parameter label	Parameter Definition
NAME	Six-characters of LINKAGE TABLE file names and PROMPT file names.
DOC	Six-character identifier of document with which this file is associated.
SEG	Six-character identifier of segment with which this file is associated.
D	Two-character identifier of document with which this file is associated.
S	Two-character identifier of segment with which this file is associated.
V	Version number
DESCRIPTION	Forty-character table description.

TABLE 8.4-III.- PROCESSOR MESSAGE TABLE

PROCESSOR DOC

MSG no.	Message ID block	Message text block and explanation
1		INVALID DOCUMENT ID
2		INVALID SEGMENT ID
3		YOU HAVE REQUESTED SOMEONE ELSES DEFAULT TABLE.
4		UNABLE TO OPEN LINKAGE TABLE DIRECTORY. IERR = XX (FATAL ERROR)
5		REQUESTED LT NAME NOT IN DIRECTORY.
6		REQUESTED LINKAGE TABLE IS NOT FOR REQUESTED DOCUMENT AND SEGMENT.
7		SEARCH FOR COMPATIBLE LINKAGE TABLE ABANDONED (IN RESPONSE TO USER ABORT).
8		LINKAGE TABLE NAMED XXXXXX ALREADY EXISTS.
9		MASTER DEFAULT TABLE MAY NOT BE RECREATED FROM DOC.
10		REQUESTED LT NAME DOES NOT PRESENTLY EXIST.
11		A LINKAGE TABLE NAMED XXXXXX ALREADY [illegible] FOR ANOTHER DOCUMENT AND SEGMENT. YOU MUST SELECT ANOTHER NAME.
12		YOU MAY NOT CREATE A TABLE WITH ANOTHER USER'S ID
13		DACONICS INPUT FILE NAME IS XXXXXX:XXXXXX:FORM
14		I/O ERROR IN DACIG. IERR = XX

TABLE 8.4-III.- Concluded

PROCESSOR DOC

MSG no.	Message ID block	Message text block and explanation
15		WRONG FORM. FORM IS FOR DOCUMENT XXXXXX, SEGMENT XXXXXX. YOU ARE CURRENTLY PROCESSING DOCUMENT XXXXXX, SEGMENT XXXXXX.
16		VERSION NUMBERS ARE INCOMPATIBLE. LINKAGE TABLE VERSION IS XX, FORM VERSION NUMBER IS XX.
17		I/O ERROR IN DOPRG. IERR = XX.
18		FORM CANNOT BE RETRIEVED.
19		DIMENSIONS MUST BE THE SAME WITHIN A TABLE.
20		MATRICES NOT PERMITTED WITHIN A TABLE.
21		DACONICS SEGMENT FILE NAME IS XXXXXX.

8.5 DOCUMENTATION PROCESSOR (DOC) ROUTINES

8.5.1 Routine Name - Main Program DOC

8.5.1.1 Purpose

The purpose of DOC is to initialize various parameters in common, and to direct program flow through the program segments that are required for the completion of a document segment. It also monitors the status of the error flag upon return from each segment, and terminates the session if it has been set.

8.5.1.2 Functional Description

DOC first sets the common variable PROFLG to 1 to inform those routines which are shared with LTBLD that they are being used by DOC. Four segments are called by DOC; however, two of the segments are called more than once to perform different functions. The segment DPBSG is called first to process the document segment, and linkage table requests. Segment LINDG is then called to retrieve the specified LINKAGE TABLE and PROMPT files to read the linkage table into common.

Segment LNXDG is then called to activate the Linkage Table Editor, and to give the user the opportunity to make any desired modifications to the table before it is used by the processor. After the modifications (if any) have been completed, LINDG is called again to give the user the opportunity to create a new LINKAGE TABLE file. LTBSG is then called a second time to retrieve the appropriate FORM file from the Daconics.

This completes the preparations for the actual document segment creation process. Segment DOPRG utilizes the LINKAGE TABLE, PROMPT file, and FORM file as inputs to produce a completed segment ready for transmission back to the Daconics. Since segment DPBSG contains the Daconics interface, it is called a third time to carry out that function. This completes that portion of the semiautomated documentation process that is performed on the HP21MX.

8.5.1.3 Assumptions and Limitations

None.

8.5.1.4 Method

None.

8.5.1.5 Routine Input/Output Variables

The DOC input/output variables are presented table 8.5.1-I.

8.5.1.6 Functional Logic Flow

The functional logic flow for DOC is presented in figure 8.5.1-1.

8.5.1.7 Diagnostics and Debug

None.

8.5.1.8 Special Comments

None.

8.5.1.9 References

None.

TABLE 8.5.1-I.- INPUT/OUTPUT VARIABLES

Routine DOC

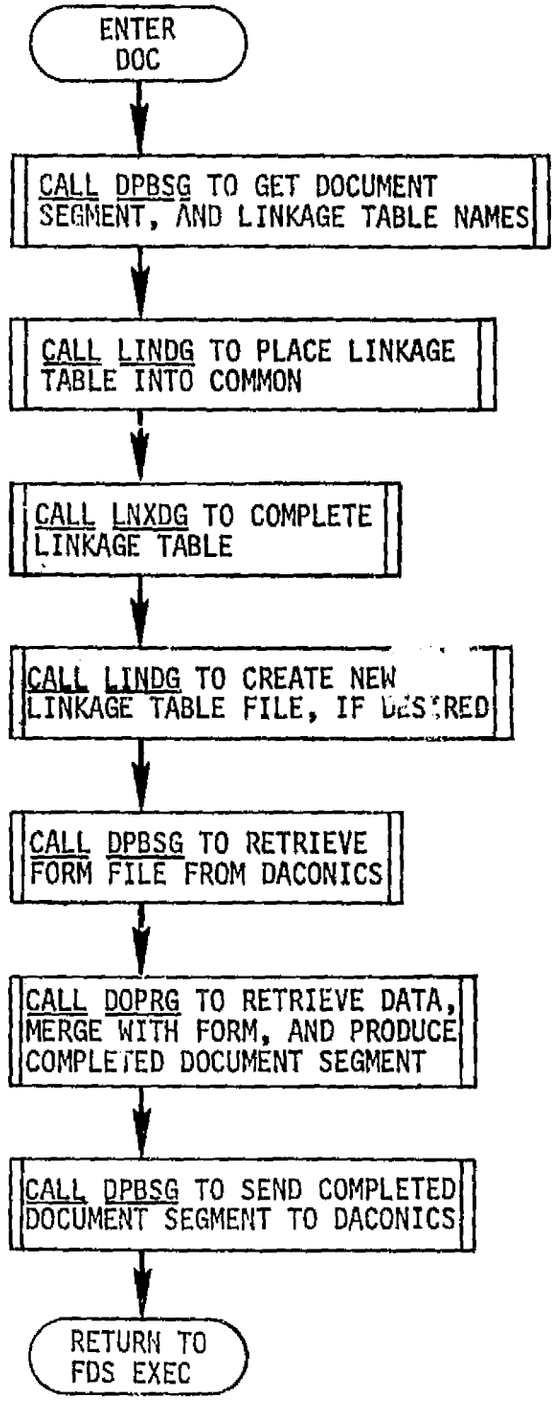
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
LU		Intg	I/O		C		Logical unit of user's terminal
USERID		Intg	I/O		C		One-character user-ID
NAMFRM		6CH	O		C		Name of FORM file
NOUTFL		6CH	O		C		Name of output segment file
ICR		Intg	O		C		Cartridge number - LT files
ISECU		2CH	O		C		Security code - LT files
JCR		Intg	O		C		Cartridge number - directories
JSEQ		2CH	O		C		Security code - directories
PROFLG		Intg	O		C		Processor flag: 0 = LTBLD 1 = DOC
IOPTN		Intg	O		C		Option flag: 0 = Retrieve DDT 1 = Retrieve form 2 = Send segment
ISW		Intg	O		C		Control flag for LTMAG: 1 = First entry 2 = Second entry

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.1-1.- DOC functional logic flow.

8.5.2 Subroutine Name - AWAG

8.5.2.1 Purpose

The purpose of this subroutine is to retrieve values from named data elements in the AWA.

8.5.2.2 Functional Description

AWAG used the FDS utility XPGET to access data values in the AWA. A special initialization call to XPGET is executed on (and only on) the initial entry into AWAG.

AWAG operates either in "TABLE" mode or "NORMAL" mode. If the data ele ment to be retrieved is dimensioned, and is one of a set of data elements whose values are to be displayed in column tabular format, then special processing is required. To illustrate, suppose two arrays, A and B, are each dimensioned by 10, and are to be displayed as two side-by-side columns; A in column 1 and B in column 2. Since the output from the documentation processor must be produced line-by-line, the values must be retrieved in the order A(1), B(1), A(2), B(2), ···. However, if the values of a dimensioned array are to be displayed in the order in which they are stored, all of the values will be retrieved with a single call to XPGET.

XPGET is designed to access data elements that appear in an interface table which is created and cataloged by the FDS Executive.

Since the data elements to be retrieved by AWAG are described in an LT, which is unknown to the FDS Executive, an intermediate step must be gone through before XPGET is called. The documentation processor has associated with it an interface table that contains a single dummy parameter. Upon each entry to AWAG, the dummy parameter is replaced with the desired entry from the LT. XPGET can then access the AWA exactly as it would if a normal interface table was being used.

When operating in "TABLE" mode, the information in the LT must be modified before placing it in the interface table. If the LT information was to be used directly, XPGET would retrieve all the values on the first call. To prevent this, and to ensure that the correct value is retrieved on each call, the FDS utility XPATR is called once for each parameter that is included in the tabular output. XPATR returns a pointer to the first value in the array. AWAG uses this pointer to address the first value to be retrieved. The data type is used to compute the length of the value in words, and for each subsequent call the pointer is incremented by the value length to compute the next address. In this way, AWAG is able to retrieve all of the values to be included in the tabular output one at a time in the order in which they are to appear.

8.5.2.3 Assumptions and Limitations

AWAG assumes that it is being executed under the FDS Executive, and that the FDS utility XPGET is available.

8.5.2.4 Method

None.

8.5.2.5 Routine Input/Output Variables

The AWAG input/output variables are presented in table 8.5.2-I.

8.5.2.6 Functional Logic Flow

The functional logic flow for AWAG is presented in figure 8.5.2-1.

8.5.2.7 Diagnostics and Debug

None.

8.5.2.8 Special Comments

None.

8.5.2.9 References

None.

TABLE 8.5.2-I - INPUT/OUTPUT VARIABLES

Routine AWAG

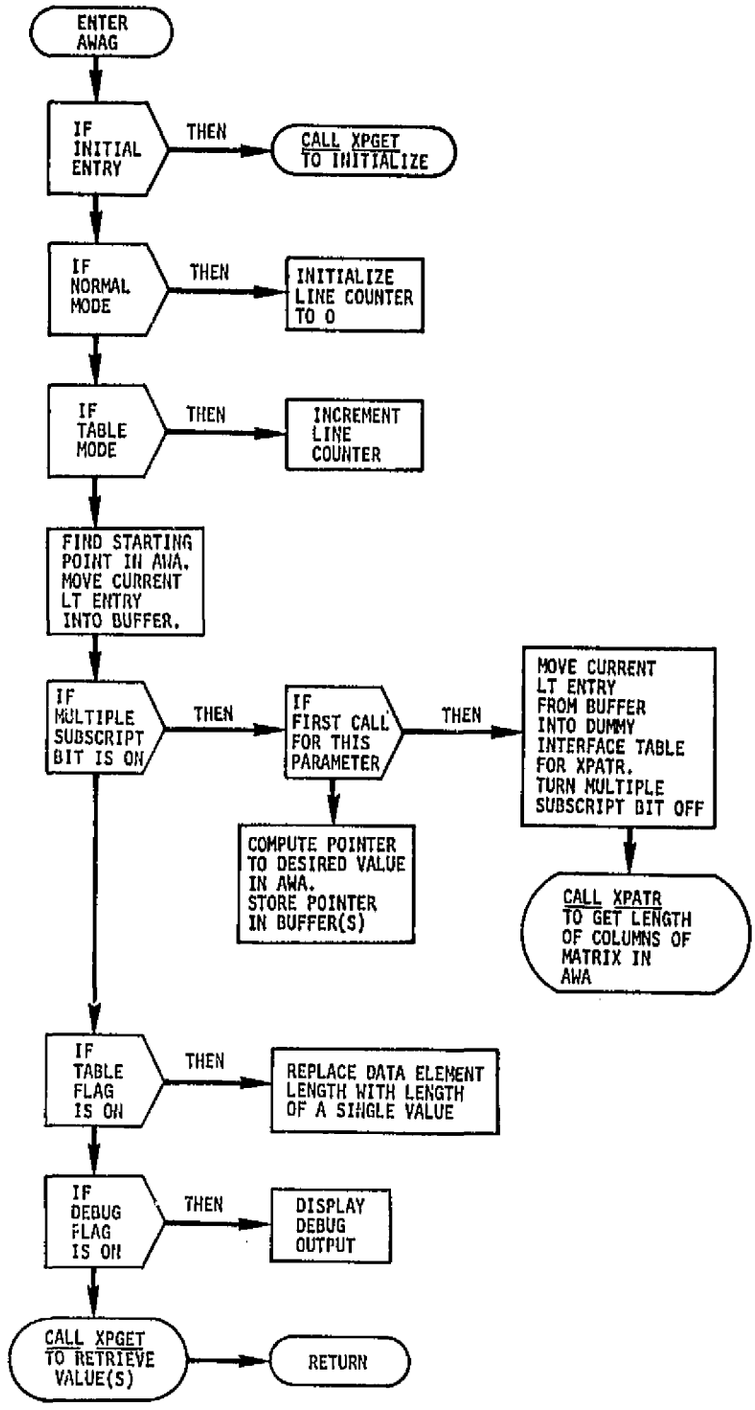
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IARRAY		Intg	I		A		Output array containing data values
IDEBUG		Intg	I		A		Debug output flag
ITYPE		Intg	I		A		Data element type
IUNIT		Intg	I		A		Debug hardcopy unit
LENS		Intg	I		A		Array of lengths of data types
LINE		Intg	I		A		Current line number in LT
LNSTRT		Intg	I		A		Starting line number in LT of tabular output
TABLE		Intg	I		A		Tabular output flag
LU		Intg	I		C		User's terminal unit number
LITPTR		Intg	I		C		Pointer to literal area in LT
HEDR		Intg	I/O		C		Linkage table header record
LTABLE		Intg	O		C		Linkage table

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.2-1.- AWAG functional logic flow.

8.5.3 Subroutine BRUPK

8.5.3.1 Purpose

The purpose of this subroutine is to unpack an array, which is in A2 format, into an array in R1 format.

8.5.3.2 Functional Description

BRUPK differs from the similar IBM utility XRUPK in two ways. First, it does not suppress blanks, allowing data to be right-justified in a field. Second, it has the option to replace blanks with zeros. This permits the presentation of time in the desired format. For example, a time of 1 day, 7 hours, 4 minutes, and 23.65 seconds would appear as 1 :7 :4 :23.65 using XRUPK, while using BRUPK will allow the time to be shown as 01:07:04:23.65.

8.5.3.3 Assumptions and Limitations

The calling program must have dimensioned IXY by at least IRETC.

8.5.3.4 Method

None.

8.5.3.5 Routine Input/Output Variables

The BRUPK input/output variables are presented in table 8.5.3-I.

8.5.3.6 Functional Logic Flow

The functional logic flow for BRUPK is presented in figure 8.5.3-1.

8.5.3.7 Diagostics and Debug

None.

8.5.3.8 Special Comments

None.

8.5.3.9 References

None.

TABLE 8.5.3-I.- INPUT/OUTPUT VARIABLES

Routine BRUPK

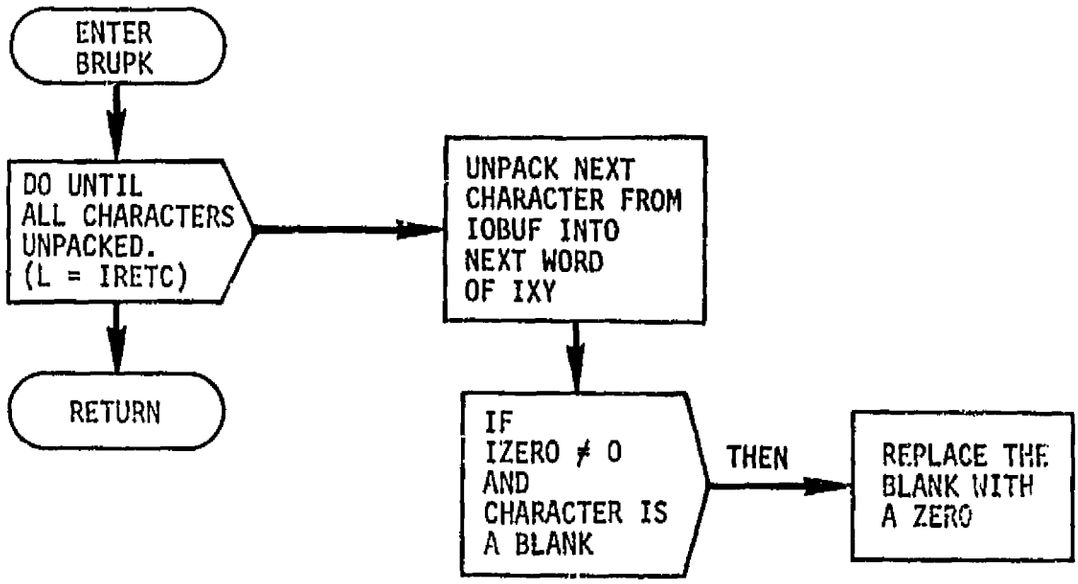
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IOBUF		Intg	I		A		Input character string in A2 format
IRETC		Intg	I		A		Number of characters to be unpacked
IXY		Intg	O		A		Output array in R1 format
IZERO		Intg	I		A		Zero replacement flag = 0, keep blanks = 1, replace blanks with zeros

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.3-1.- BRUPK functional logic flow.

8.5.4 Subroutine DACIG

8.5.4.1 Purpose

The purpose of subroutine DACIG is to manage the transfer of data between HP21MX data elements and Daconics files. For the LTBLD, DACIG retrieves DDT's from Daconics and places them on the temporary data element $DDTG on the HP21MX. For the documentation processor, it transfers blank forms from the Daconics to $FORMG on the HP21MX, and transfers completed segments from $OFRMG on the HP21MX to the Daconics.

8.5.4.2 Functional Description

DACIG first determines what function it is expected to perform. If a file is to be retrieved from the Daconics, the file name is constructed in the form DOCNAM:SEGNAM:XXXX where XXXX is either DDT or FORM, depending on which is to be received. The name of the destination data element is then stored, the name being determined by the type of file to be retrieved.

The header record is read from the Daconics file, and the version number is extracted. If it is a DDT, the version number is stored, and the header is written to the HP21MX file. If it is a FORM, the version number is compared with the version number of the current LT, and the run is aborted if they are different. If the version number is correct (or it is a DDT), the remaining records are transferred and control is returned to the calling program.

If a completed segment is to be sent to the Daconics, the program first solicits confirmation from the user that the segment is actually to be sent. If it is, the user is given the opportunity to supply a four-character identifier to be appended onto the basic file name. The segment is then sent to the Daconics via a call to DWRIT. After the segment has been sent, the user is given the opportunity to recall it for display at his terminal. This provides a mechanism for the user to verify the transmission before terminating the run.

8.5.4.3 Assumptions and Limitations

None.

8.5.4.4 Method

None.

8.5.4.5 Routine Input/Output Variables

The DACIG input/output variables are presented in table 8.5.4-I.

8.5.4.6 Functional Logic Flow

The functional logic flow for DACIG is presented in figure 8.5.4-1.

8.5.4.7 Diagnostics and Debug

None.

8.5.4.8 Special Comments

None.

8.5.4.9 References

None.

TABLE 8.5.4-I.- INPUT/OUTPUT VARIABLES

Routine DACIG

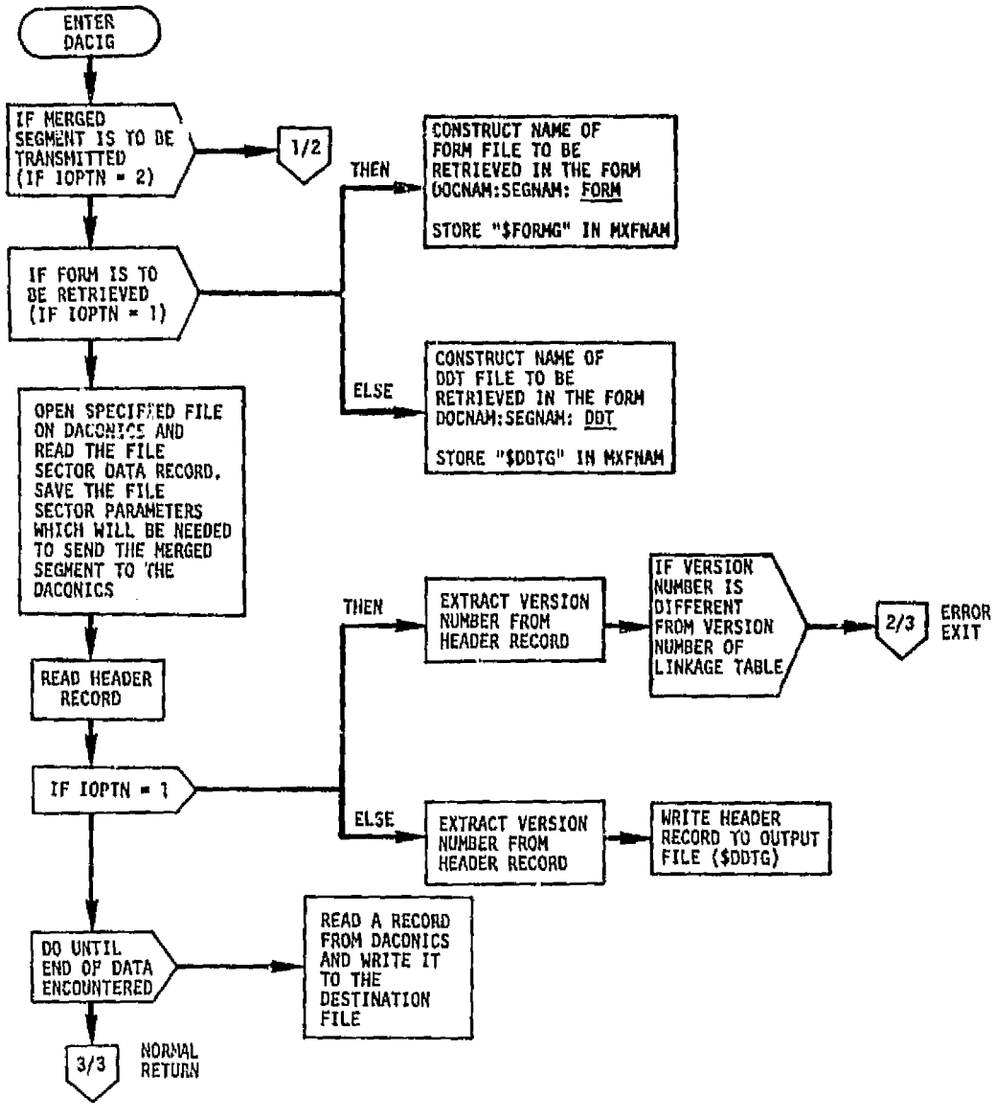
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
DOCID		Intg	I		C		Two-character document ID
DOCNAM		Intg	I		C		Six-character document ID
FORMSW		Intg	O		C		Form retrieval flag (0 = unsuccessful)
FVERS		Intg	O		C		Version number of DDT
IDCB		Intg	I		C		File manager data control block
IERR		Intg	I		C		Error flag from file manager
IOPTN		Intg	I		C		Option control flag
JCR		Intg	I		C		Cartridge number for forms and DDT's
LTNAM		Intg	I		C		Linkage table name
LU		Intg	I		C		Logical unit number of user's terminal
NAMDDT		Intg	I		C		Name of HP21MX file for DDT
NAMFRM		Intg	I		C		Name of HP21MX file for FORM
NOUTFL		Intg	I		C		Name of HP21MX file for merged segment
SEGID		Intg	I		C		Two-character segment ID
SEGNAM		Intg	I		C		Six-character segment ID
USERID		Intg	I		C		One-character user ID
VERS		Intg	I		C		Version number of linkage table

NOTES:

TYPE
Free, Intg, Real, Dubl, 2CH, 6CH, 18CH, 36CH, 72CH, Mix, Char, Bin

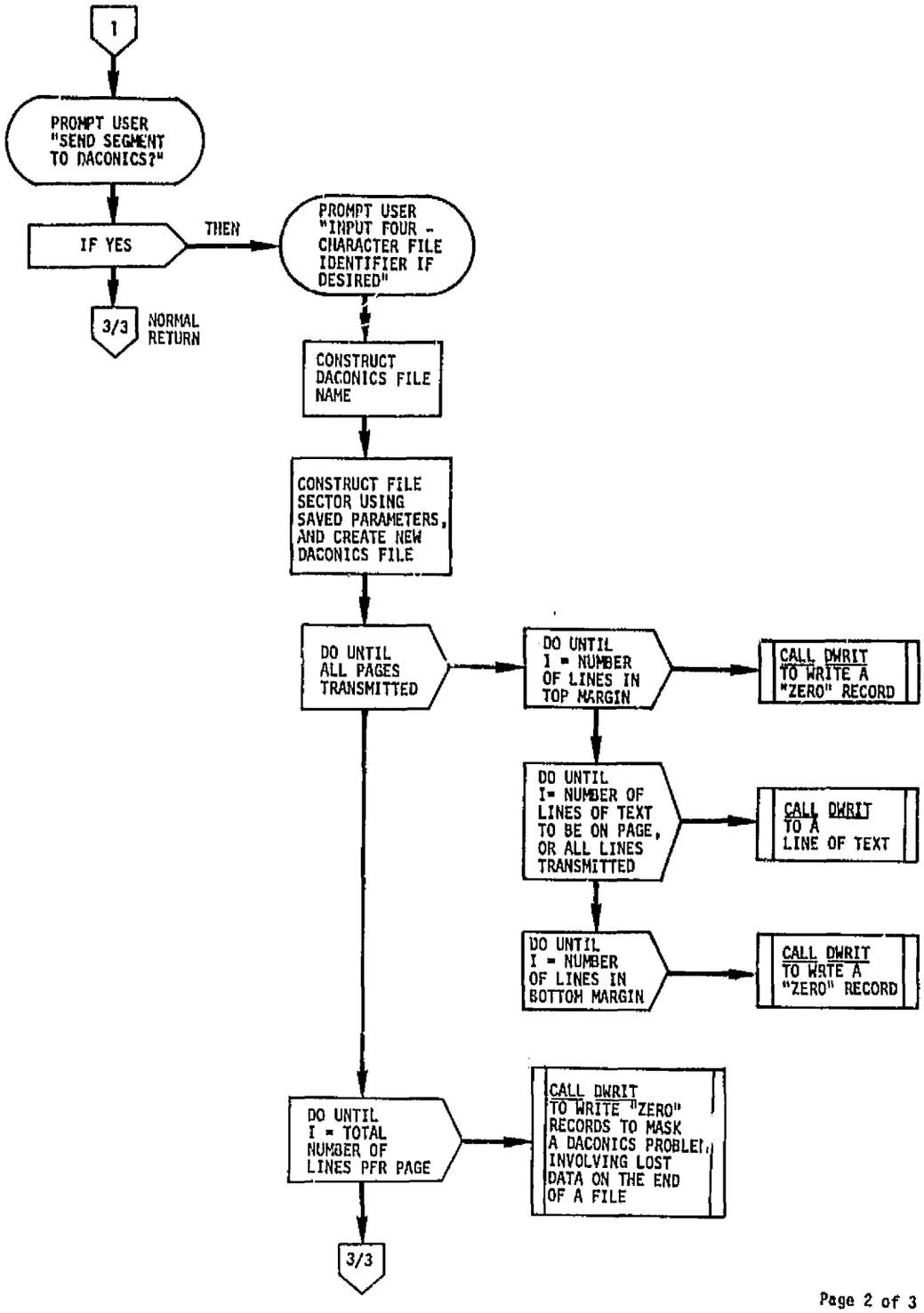
USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



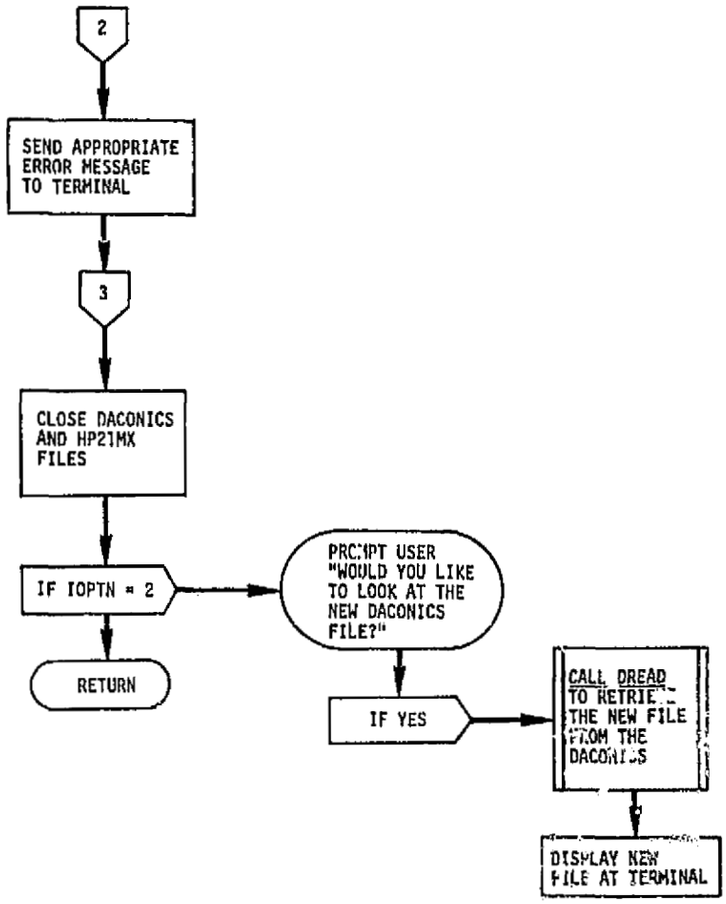
Page 1 of 3

Figure 8.5.4-1.- DACIG functional logic flow.



Page 2 of 3

Figure 8.5.4-1.- Continued.



Page 3 of 3

Figure 8.5.4-1.- Concluded.

8.5.5 Subroutine DOPRG

8.5.5.1 Purpose

The purpose of this subroutine is to retrieve the specified data elements, reformat them, and merge them with the blank form to produce a complete document segment.

8.5.5.2 Functional Description

DOPRG first checks to see if the form has been retrieved from the Daconics. If the form has not been retrieved, the user is given the option to continue processing, and display the data values at the terminal, or to abort the run.

If the form has been retrieved, or if processing is to continue without it, the parameter names, table flags, and format specifications are retrieved from the PROMPT file.

The first "PAGE" of the form is then read into the internal working buffer and prepared for processing. A page of the form is defined as that portion which will fit in the buffer. The segment is processed page-by-page until the segment is complete.

The actual processing is done in either of two modes; i.e., NORMAL mode or TABLE mode. The "TABLE FLAG," obtained from the DDT (via the PROMPT file), indicates those parameters that are to be processed in TABLE mode. In NORMAL mode, the parameters are processed in the order in which they appear in the LT. All values of a dimensioned array are retrieved as a group. However, in TABLE mode, those arrays that are to be presented in tabular format are retrieved one value at a time. For example, if two arrays (dimensioned by 10 each) are to be displayed in two parallel columns of a table, DOPRG processes the first value of the first array, the first value of the second array, the second value of the first array, etc., down through the tenth value of each array.

Upon entry into TABLE mode, the LT is searched for the end-of-table flag, and the beginning and ending parameter numbers (within the LT) are stored. The dimensions of all parameters to be included within the table are checked for consistency, and a check is also made for any multidimensioned arrays. If everything is proper, a pair of nested DO-LOOPS is initialized; the outer DO-LOOP steps through the number of lines to be displayed in the table, and the inner DO-LOOP steps through the number of parameters (or columns) in the table. If not in TABLE mode, the initial and final values of the inner DO-LOOP are set to the same value so that the parameters in the LT will be processed in order, with no looping.

For each parameter, the LT is examined to determine if it contains a value or a data element name. If a name is present, the FDS utility is called to obtain the value(s) from the AWA. Otherwise, the value(s) is/are obtained directly from the literal area of the LT.

The format specification for the current parameter is then analyzed to determine the conversion required. If it is a "T" format, the specification is subjected to a lexical scan, and the resulting formats are constructed.

The next step is the conversion and reformatting of the value(s) from their original form into the form specified by the DDT. To accomplish this, the raw value is placed into an integer input array. This array is equivalenced to a real array and a double-precision array. If the value is a character string, it is simply moved into the output array. Otherwise, a core-to-core write is performed using the Fortran CODE function to perform the conversion. CODE is called with the input array name that corresponds to the data type (e.g., the integer array name is used if the data type is integer). The format used for the core-to-core write is the one constructed in the previous step. In this way, a real number can be converted to an ASCII "I" format, an "F" format, a "T" format, or a "D" format. Similarly, integers and double-precision numbers can be expressed in any specified format.

After the conversion and reformatting has been performed, subroutine MERG is called to place the value in the next available reserved space in the blank form. If it is a "T" format, MERG is called for each field in the format specification.

The above process is repeated until all parameters in the LT have been processed. Subroutine SOUTG is then called to transfer the last page of the completed segment to the output data element where it will await transmission to the Daconics. Subroutine SDISG is called to give the user the option to display the completed segment at his terminal. (These last two steps are omitted if the form switch is turned off.)

8.5.5.3 Assumptions and Limitations

None.

8.5.5.4 Method

None.

8.5.5.5 Routine Input/Output Variables

The DOPRG input/output variables are presented in table 8.5.5-I.

8.5.5.6 Functional Logic Flow

The functional logic flow for DOPRG is presented in figure 8.5.5-1.

8.5.5.7 Diagnostics and Debug

None.

8.5.5.8 Special Comments

None.

8.5.5.9 References

None.

TABLE 8.5.5-I.- INPUT/OUTPUT VARIABLES

Routine DOFRG

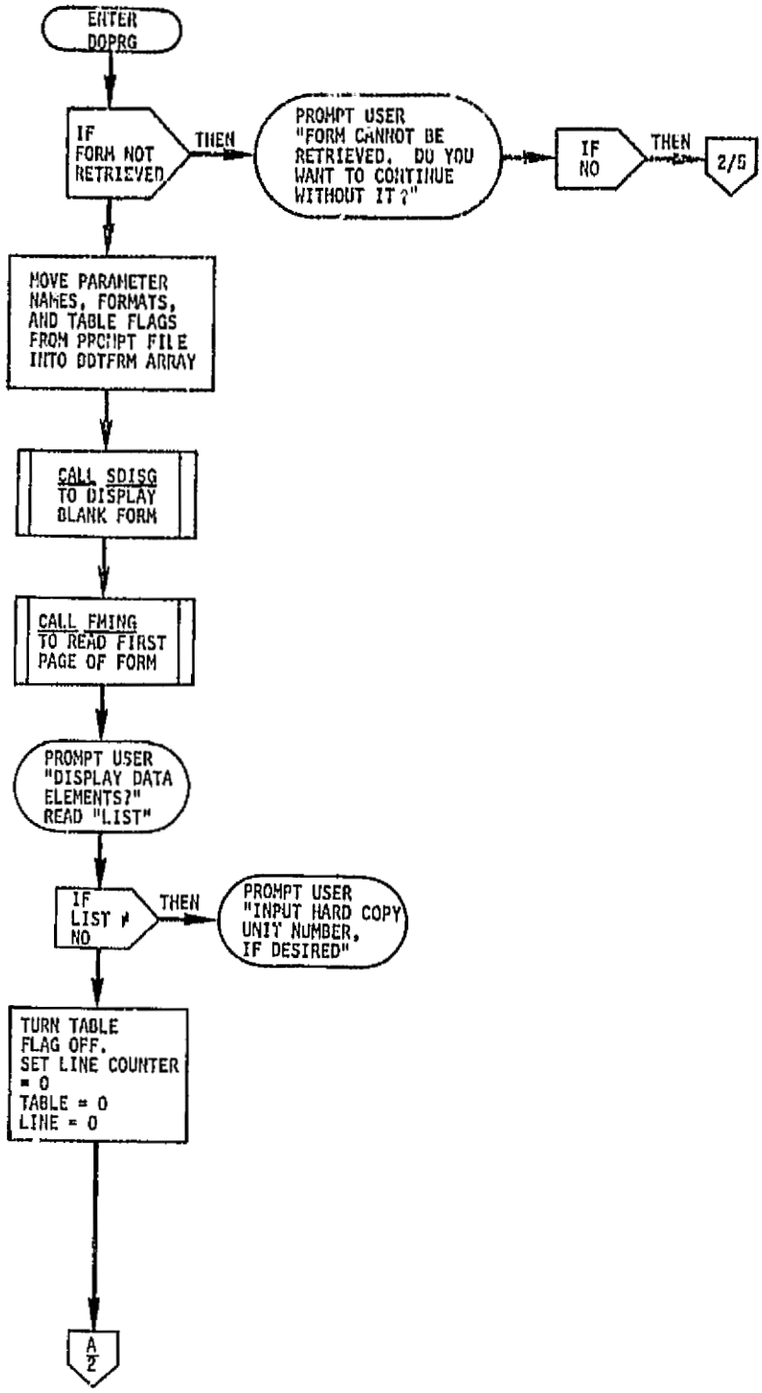
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
FORBUF		Intg	O		A		Buffer where data is merged with form
IBPNTR		Intg	O		A		Current character position in FORBUF
IDCB		Intg	I		A		File manager data control block
IERR		Intg	I		A		File manager error flag
IRETC		Intg	I		A		Number of characters unpacked by XRUPK
IXY		Intg	I		A		File manager output buffer
JPMAX		Intg	O		A		Length of FORBUF array
LI		Intg	O		A		Length of buffer for file manager
LLENTH		Intg	I		C		Array containing lengths of records in FORBUF
LTABLE		Intg	I		C		Linkage table buffer
IU		Intg	I		C		User's logical unit number
NAMDDT		6CH	I		C		Name of DDT file
NAMFRM		6CH	I/O		C		Name of FORM file
LIST		Intg	I		T		Data listing control flag
IUNIT		Intg	I		T		Data listing optional unit number

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

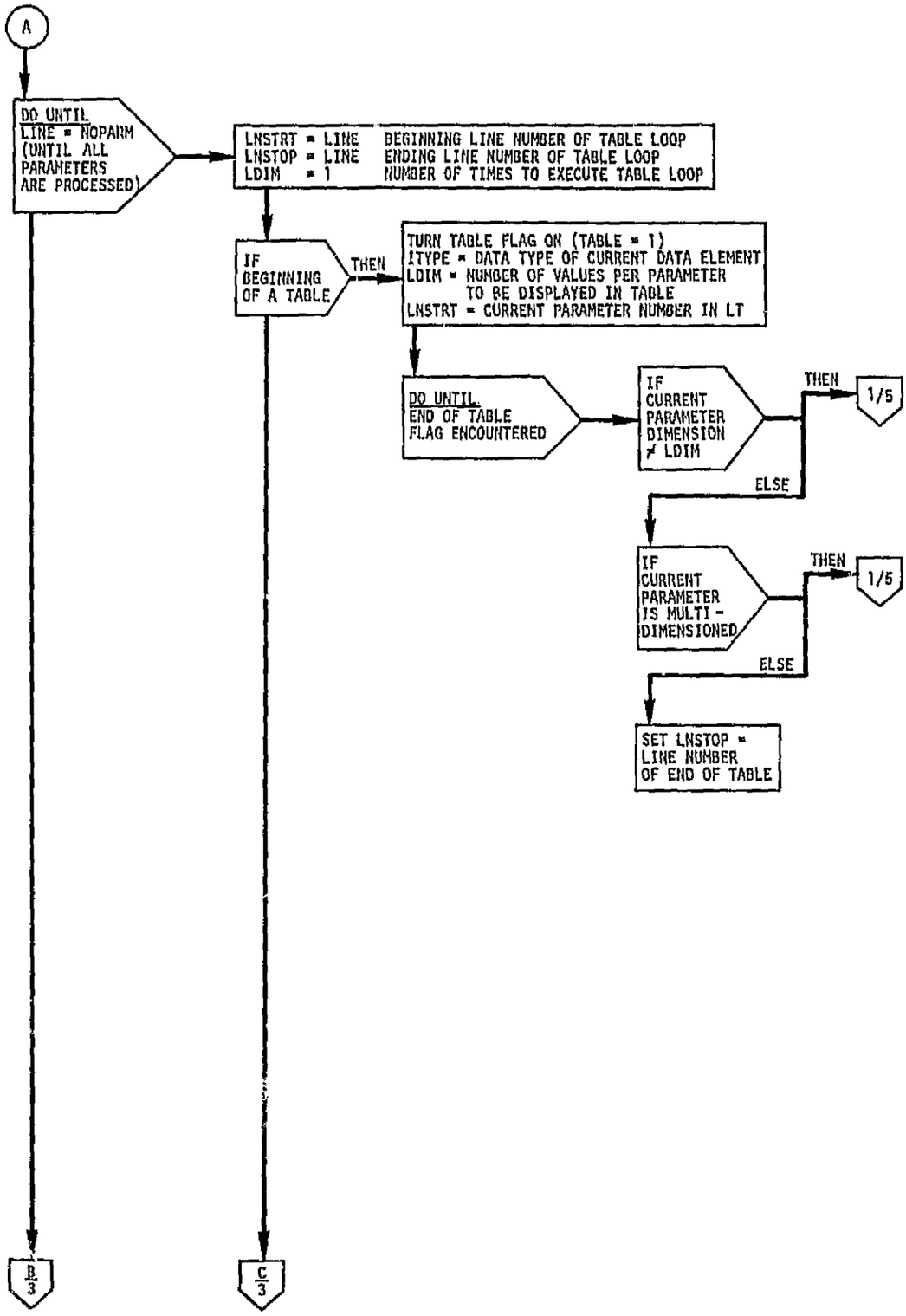
USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



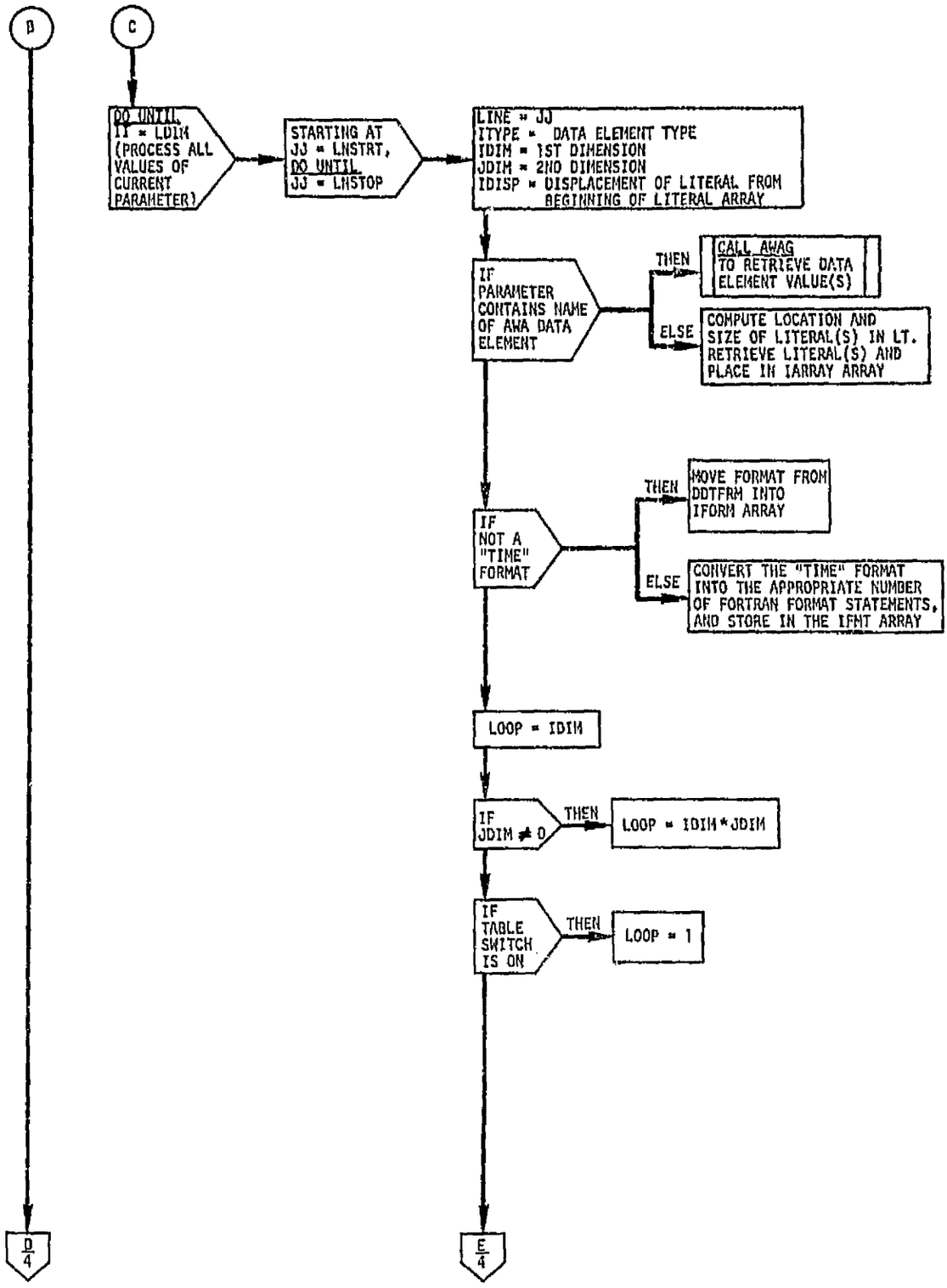
Page 1 of 5

Figure 8.5.5-1.- DOPRG functional logic flow.



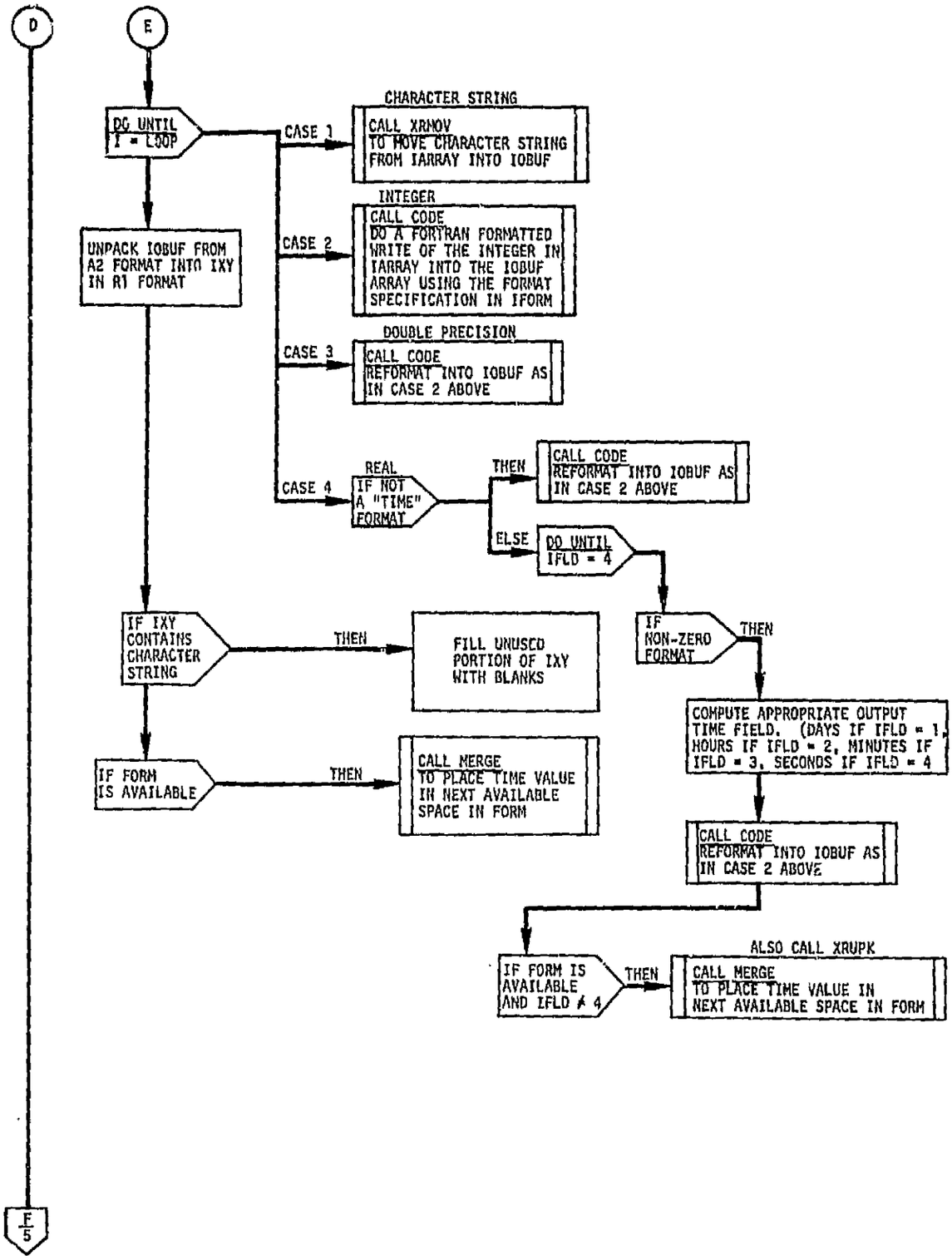
Page 2 of 5

Figure 8.5.5-1.- Continued.



Page 3 of 5

Figure 8.5.5-1.- Continued.



Page 4 of 5

Figure 8.5.5-1.- Continued.

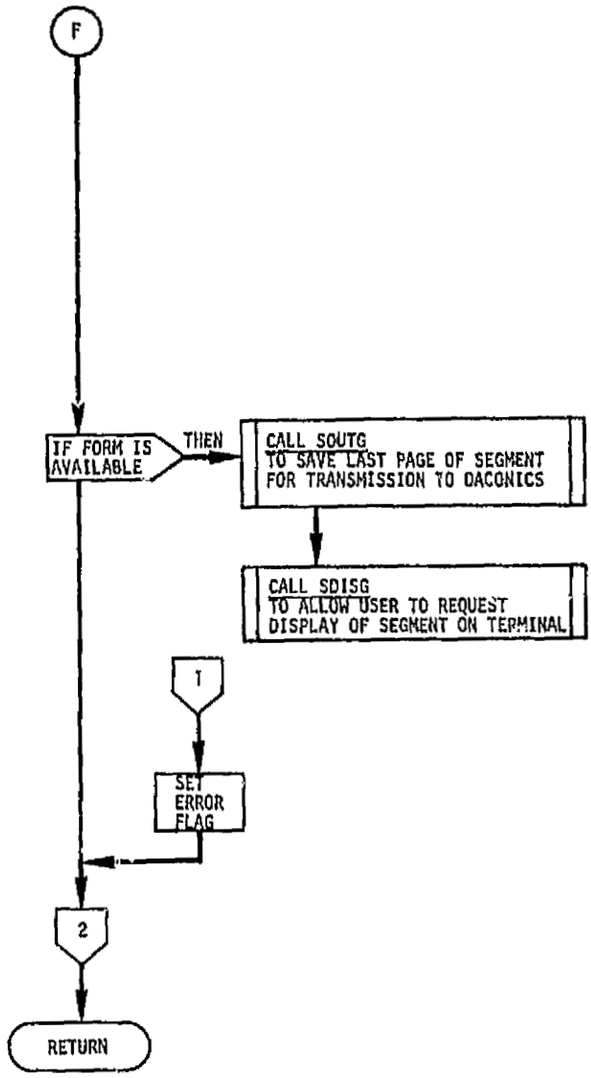


Figure 8.5.5-1.- Concluded.

8.5.6 Subroutine DOSEG

8.5.6.1 Purpose

Subroutine DOSEG obtains the document and segment identifiers from the user, and checks their validity against the document and segment directories. If they are valid, the associated two-character codes are retrieved from the directories, and the identifiers and codes are placed in common. If either is invalid, the user is given the opportunity to update the directory if a new segment is being added to the system, or to correct his input if an error was made.

8.5.6.2 Functional Description

The user is prompted for the document ID. This is the six-character identifier by which a document is known to the system. Possible responses are a "%" to terminate the run, a "?" to list the directory, an invalid ID, or a valid ID. If an ID is input that does not exist in the directory, the routine determines whether it is being called by LTBLD or by DOC. If it is called by DOC, a diagnostic is sent to the terminal and the user is reprompted. If called by LTBLD, the user is notified that the ID does not exist, and asks if the ID is to be added to the directory. If not, the user is reprompted. If it is to be added, the user is prompted for the two-character code to be assigned to the new entry, and for a description that accompanies the new entry in the directory.

When this process is completed for the document ID, an identical process is gone through for the segment ID, with one exception. If the requested document ID appears in the directory, it is valid. However, even if a segment ID appears in the directory, it must be for the correct document or it is considered invalid.

8.5.6.3 Assumptions and Limitations

Assumes that the two required directories are available on the HP21MX mass storage system.

8.5.6.4 Method

None.

8.5.6.5 Routine Input/Output Variables

The DOSEG input/output variables are presented in table 8.5.6-I.

8.5.6.6 Functional Logic Flow

The functional logic flow for DOSEG is presented in figure 8.5.6-1.

8.5.6.7 Diagnostics and Debug

Subroutine DOSEG contains two sets of diagnostics. One set is used if DOSEG is being used by DOC; the other is used if DOSEG is being used by LTBLD.

DOC Diagnostics

Diagnostic	Meaning	Action
Invalid document ID	Requested document ID does not exist in directory.	Reprompt
Invalid segment ID	Requested segment ID does appear in directory for this document.	Reprompt

LTBLD Diagnostics

Diagnostic	Meaning	Action
Requested document ID does not presently exist.	Self-explanatory	Ask user if he is adding new document to system.
Requested code already exists.	Requested two-character code has already been assigned.	Reprompt.
Requested segment ID does not exist for this document.	Requested segment ID either does not exist, or is for another document.	Ask user if he is adding new segment.
Requested code already exists for this document.	Self-explanatory.	Reprompt.

8.5.6.8 Special Comments

None.

8.5.6.9 References

None.

TABLE 8.5.6-I.- INPUT/OUTPUT VARIABLES

Routine DOSEG

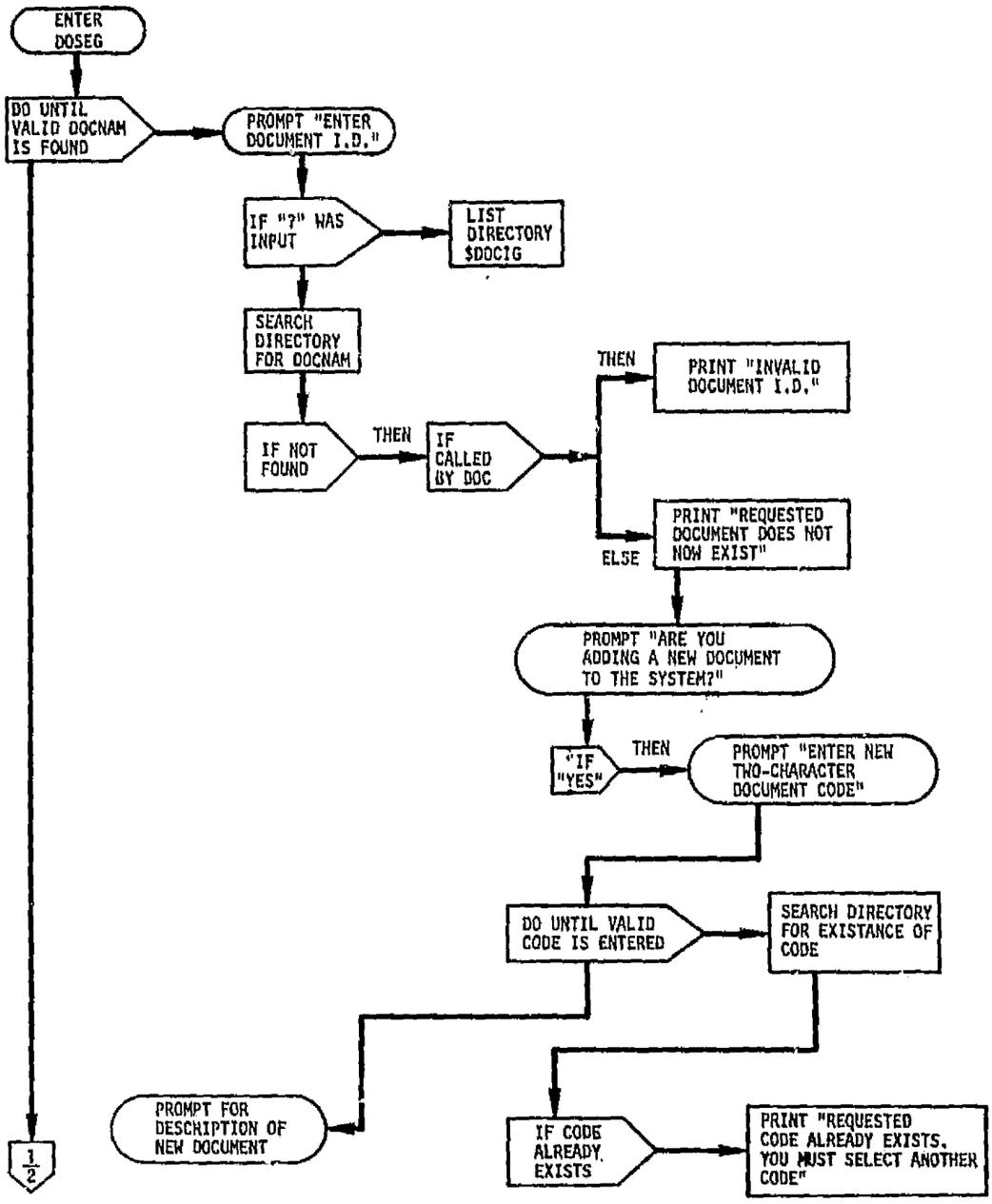
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
DOCID		2CH	O		C		Two-character document code
DOCNAM		6CH	I/O		T,C		Six-character document name
JCR		Intg	I		C		Cartridge containing directories
JSEQ		2CH	I		C		Security code for directories
LU		Intg	I		C		User terminal logical unit
PROFLG		Intg	I		C		Processor flag 0 = LTBLD 1 = DOC
SEGID		2CH	O		C		Two-character segment ID
SEGNAM		6CH	I/O		T,C		Six-character segment name

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

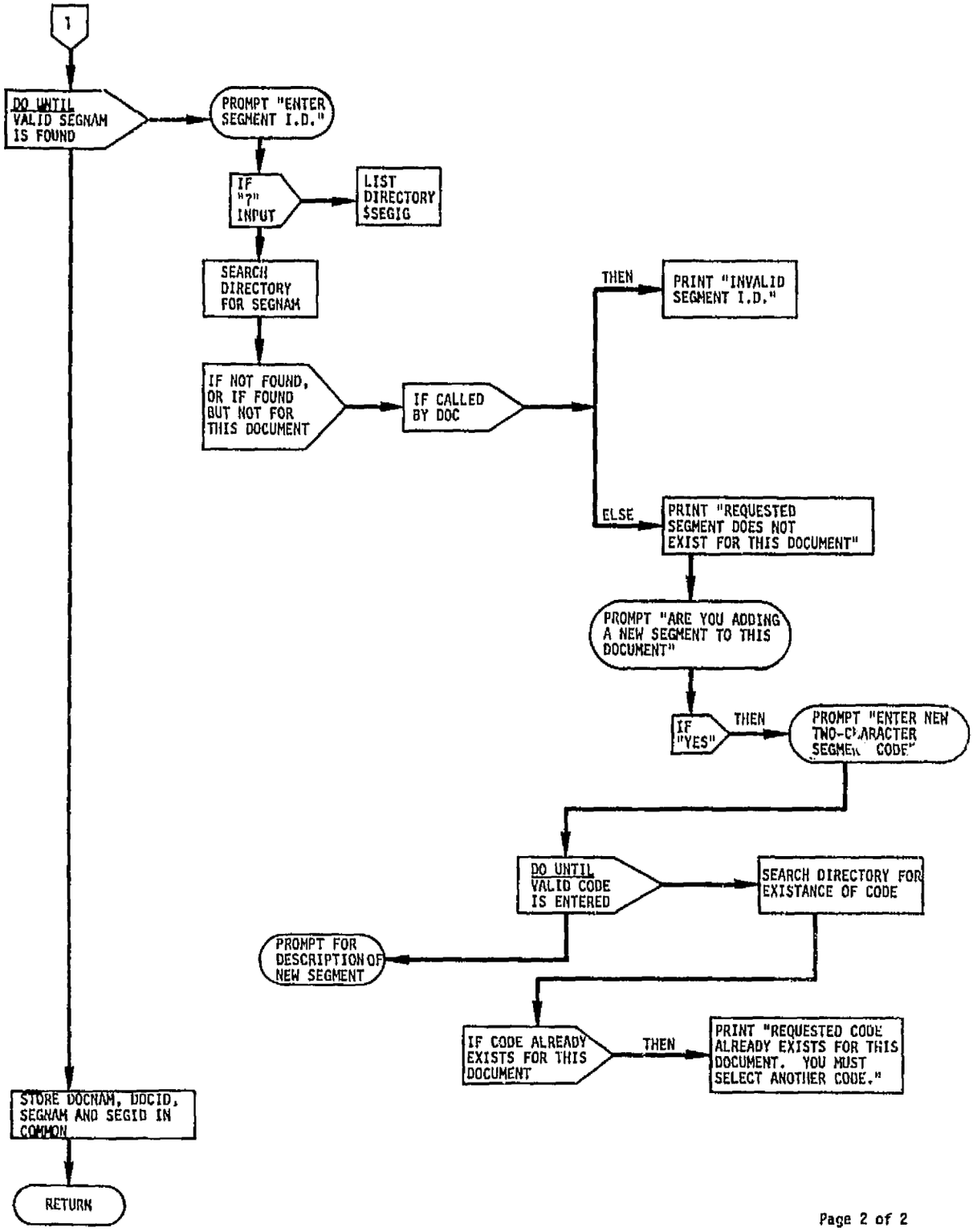
USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 2

Figure 8.5.6-1.- DOSEG functional logic flow.



Page 2 of 2

Figure 8.5.6-1.- Concluded.

8.5.7 Routine Name - Main Program DPBSG

8.5.7.1 Purpose

The purpose of DPBSG is to process the user requests for the document and segment to be processed, and the linkage table to be used. The reason these functions are performed in this small segment is to make more room for working buffers in segment DOPRG, the main document processing segment. The interface to the Daconics data channel is also in this segment for the same reason.

8.5.7.2 Functional Description

Processing within DPBSG is controlled by an option flag that directs the flow of execution to different parts of the program. Segment DPBSG is called three times during a normal documentation session. For the first call, the option flag is set to zero, and subroutines DOSEG and LTNMG are called to process the document, segment, and linkage table names.

For the second call, the flag has a value of 1, and subroutine DACIG is called to retrieve the appropriate blank form from the Daconics.

For the third call, the flag has a value of 2, and DACIG is called again to send the completed segment back to the Daconics.

8.5.7.3 Assumptions and Limitations

None.

8.5.7.4 Method

None.

8.5.7.5 Routine Input/Output Variables

The DPBSG input/output variables are presented in table 8.5.7-I.

8.5.7.6 Functional Logic Flow

The functional logic flow for DPBSG is presented in figure 8.5.7-1.

8.5.7.7 Diagnostics and Debug

None.

8.5.7.8 Special Comments

None.

8.5.7.9 References

None.

TABLE 8.5.7-I.- INPUT/OUTPUT VARIABLES

Routine DPBSG

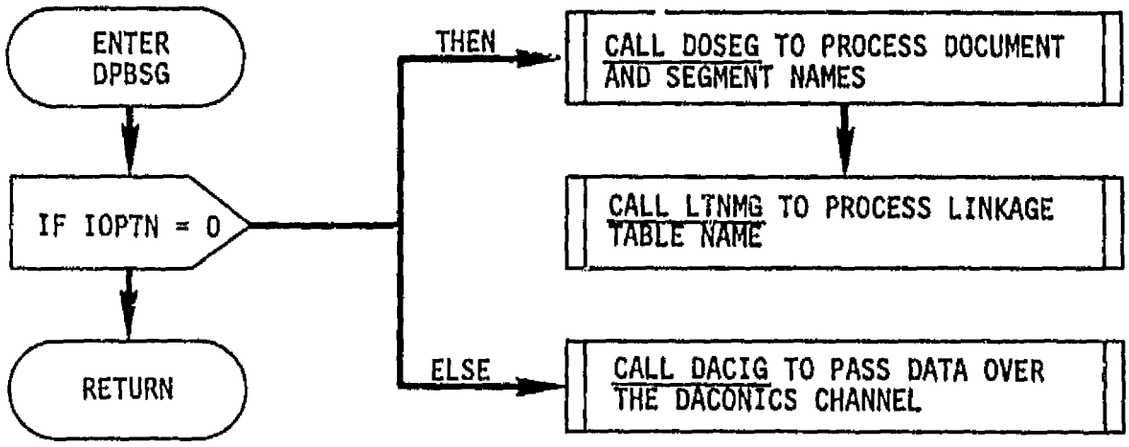
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IOPTN		Intg	I		C		Option control flag
LTNAM		6CH	O		C		Input linkage table name
NLTNAM		6CH	O		C		Output linkage table name
XE(1)		Intg	I/O		C		Error flag

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.7-1.- DPBSG functional logic flow.

8.5.8 Subroutine DREAD

8.5.8.1 Purpose

The purpose of this subroutine is to retrieve a record of data from the Daconics via the data channel, and to perform a character set transformation to convert the Daconics special characters into characters acceptable to the HP21MX.

8.5.8.2 Functional Description

DREAD first makes an EXEC call to retrieve a block of data from the Daconics. The amount of data to be retrieved is specified by the input parameter "LENGTH." After control has been returned from the EXEC call, a second EXEC call is made to obtain the completion status of the first call. If a nonzero status was returned, the status value is placed in the output parameter "ISTAT," and control is returned to the calling program.

If the read was successful, the received data are either returned to the calling program as is, or are processed as follows:

The record is first checked for nonzero values. If the record contains all zeros, the record is discarded and the next record is read. If the record contains nonzero data, the Daconics special characters are converted to standard HP21MX characters, and the length of the record is placed in the output parameter "LEN."

8.5.8.3 Assumptions and Limitations

None.

8.5.8.4 Method

None.

8.5.8.5 Routine Input/Output Variables

The DREAD input/output variables are presented in table 8.5.8-I.

8.5.8.6 Functional Logic Flow

The functional logic flow for DREAD is presented in figure 8.5.8-1.

8.5.8.7 Diagnostics and Debug

None.

8.5.8.8 Special Comments

None.

8.5.8.9 References

None.

TABLE 8.5.8-I.- INPUT/OUTPUT VARIABLES

Routine DREAD

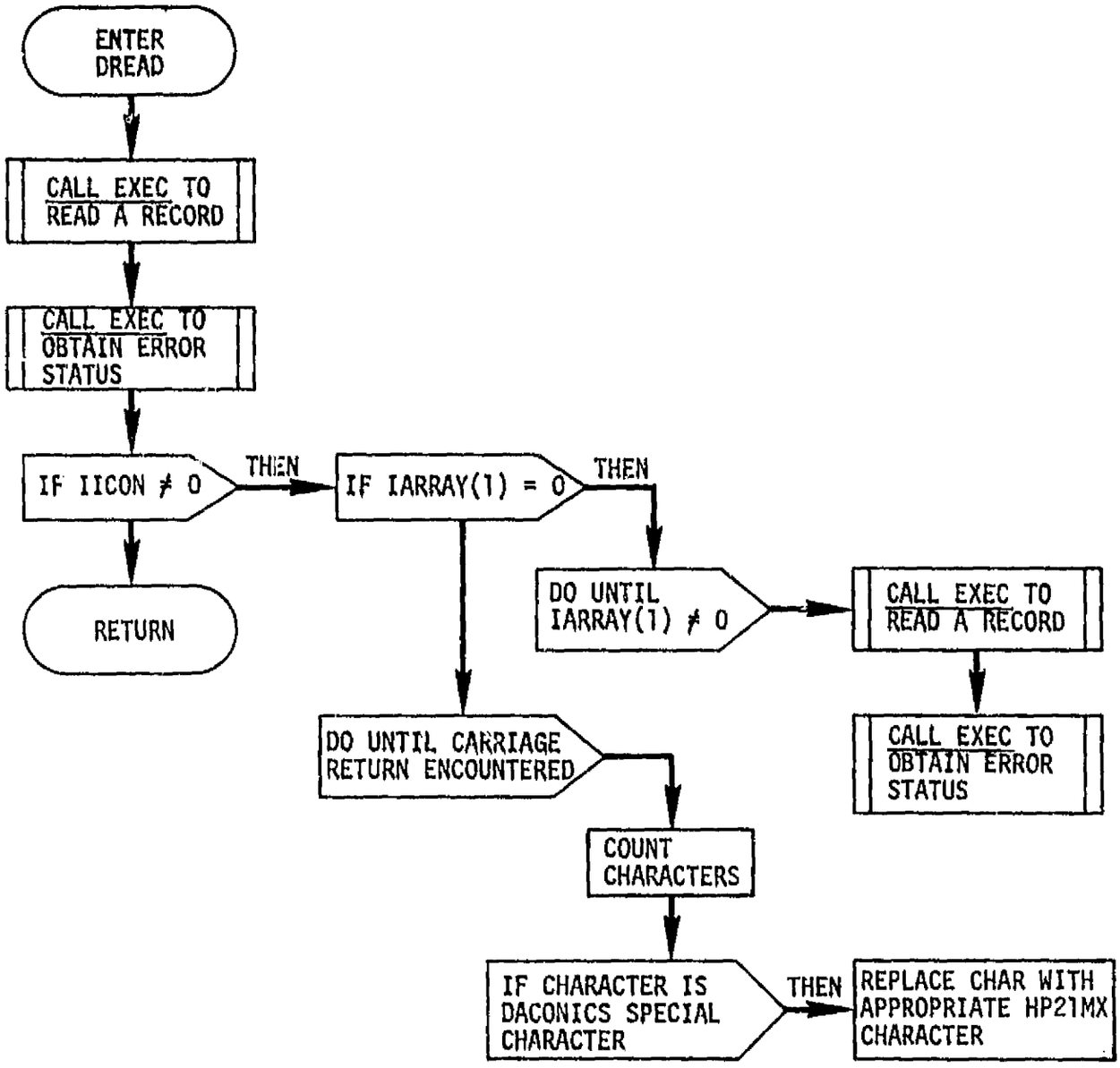
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IARRAY		Intg	O		A		Array containing data record
IICON		Intg	I		A		Data conversion flag 0 = no conversion
ISTAT		Intg	O		A		Error status 0 = no error
LEN		Intg	O		A		Length of data record read
LENGTH		Intg	I		A		Length of data block to be read (64 or 128 words)

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
TT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.8-1.- DREAD functional logic flow.

8.5.9 Subroutine DWRIT

8.5.9.1 Purpose

The purpose of this subroutine is to send a record of data to the Daconics via the data channel.

8.5.9.2 Functional Description

The input parameter "LENGTH" is first tested for valid input. If "LENGTH" is zero, the error status flag is set, and control is returned to the calling program. The input parameter "ICON" is then tested to see if a character set conversion is required. If ICON is zero, the re cord is transmitted as is. If ICON is nonzero, all spaces are converted to nulls, and hyphens (and minus signs) are converted to either "REQUIRED HYPHEN, END OF LINE," or "REQUIRED HYPHEN, NOT END OF LINE," as appropriate. The modified record is then transmitted.

8.5.9.3 Assumptions and Limitations

None.

8.5.9.4 Method

None.

8.5.9.5 Routine Input/Output Variables

The DWRIT input/output variables are presented in table 8.5.9-I.

8.5.9.6 Functional Logic Flow

The functional logic flow for DWRIT is presented in figure 8.5.9-1.

8.5.9.7 Diagnostics and Debug

None.

8.5.9.8 Special Comments

None.

8.5.9.9 References

None.

TABLE 8.5.9-I.- INPUT/OUTPUT VARIABLES

Routine DWRIT

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
IARRAY		Intg	I		A		Array to be transmitted
ICON		Intg	I		A		Character conversion flag 0 = no conversion
ISTAT		Intg	O		A		Error status code 0 = no error
LEN		Intg	I		A		Number of words of data in IARRAY
LENGTH		Intg	I		A		Length of record to be transmitted (64 or 128 words)

NOTES:

TYPE			
Free	Dubl	18CH	Mix
Intg	2CH	36CH	Char
Real	6CH	72CH	Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

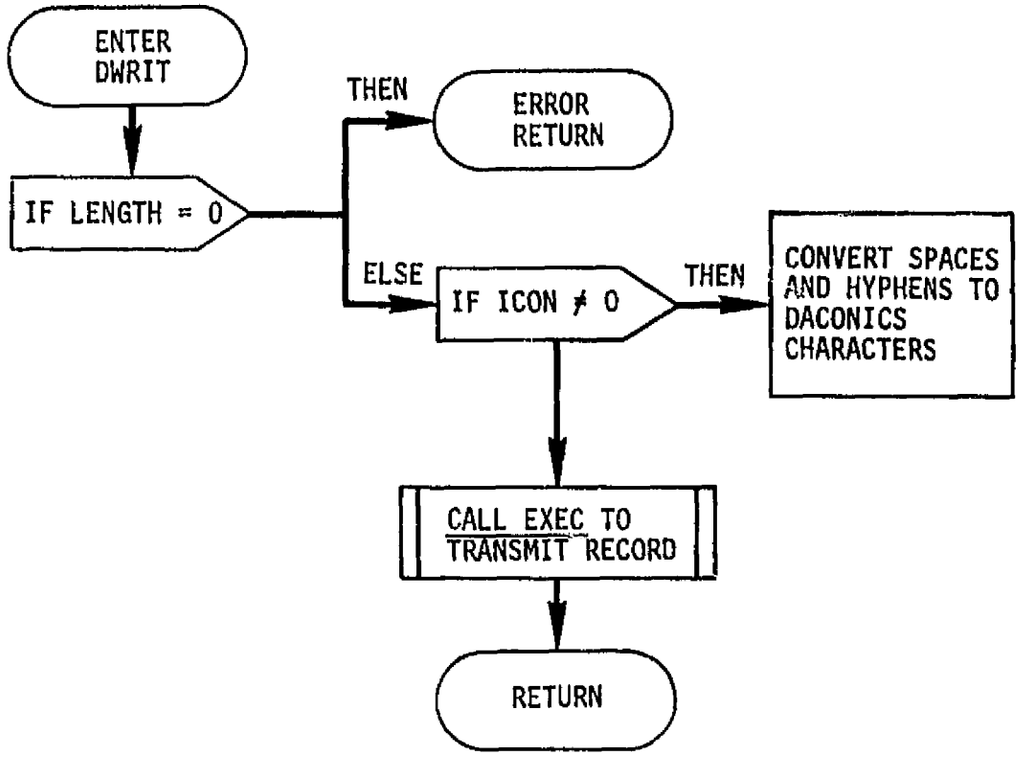


Figure 8.5.9-1.- DWRIT functional logic flow.

8.5.10 Subroutine FMING

8.5.10.1 Purpose

The purpose of this subroutine is to page the blank form into the internal working buffer. In general, document segments will be larger than the available buffer space. It is therefore necessary to read and process the form in a series of portions that do not exceed the buffer size.

8.5.10.2 Functional Description

The length of the available buffer is given by the input argument JPMAX. Successive records are read from the file named $FORMG until the current record will not fit. The current record number is saved, and the total number of characters that were stored are placed into the output argument IBPNTR. On subsequent entries, the reading from $FORMG begins with the record number that was saved from the previous entry, and the process is repeated until $FORMG is exhausted. After the last record is placed into the buffer, a value of 777 octal is stored to indicate end-of-data. As the data are read and stored, they are converted from A2 format into R1 format.

8.5.10.3 Assumptions and Limitations

The form to be processed must reside on the file named $FORMG.

The buffer length indicated by the input argument JPMAX must not exceed the dimension of the buffer.

8.5.10.4 Method

None.

8.5.10.5 Routine Input/Output Variables

The FMING input/output variables are presented in table 8.5.10-I.

8.5.10.6 Functional Logic Flow

The functional logic flow for FMING is presented in figure 8.5.10-1.

8.5.10.7 Diagnostics and Debug

None.

8.5.10.8 Special Comments

None.

8.5.10.9 References

None.

TABLE 8.5.10-I.- INPUT/OUTPUT VARIABLES

Routine FMING

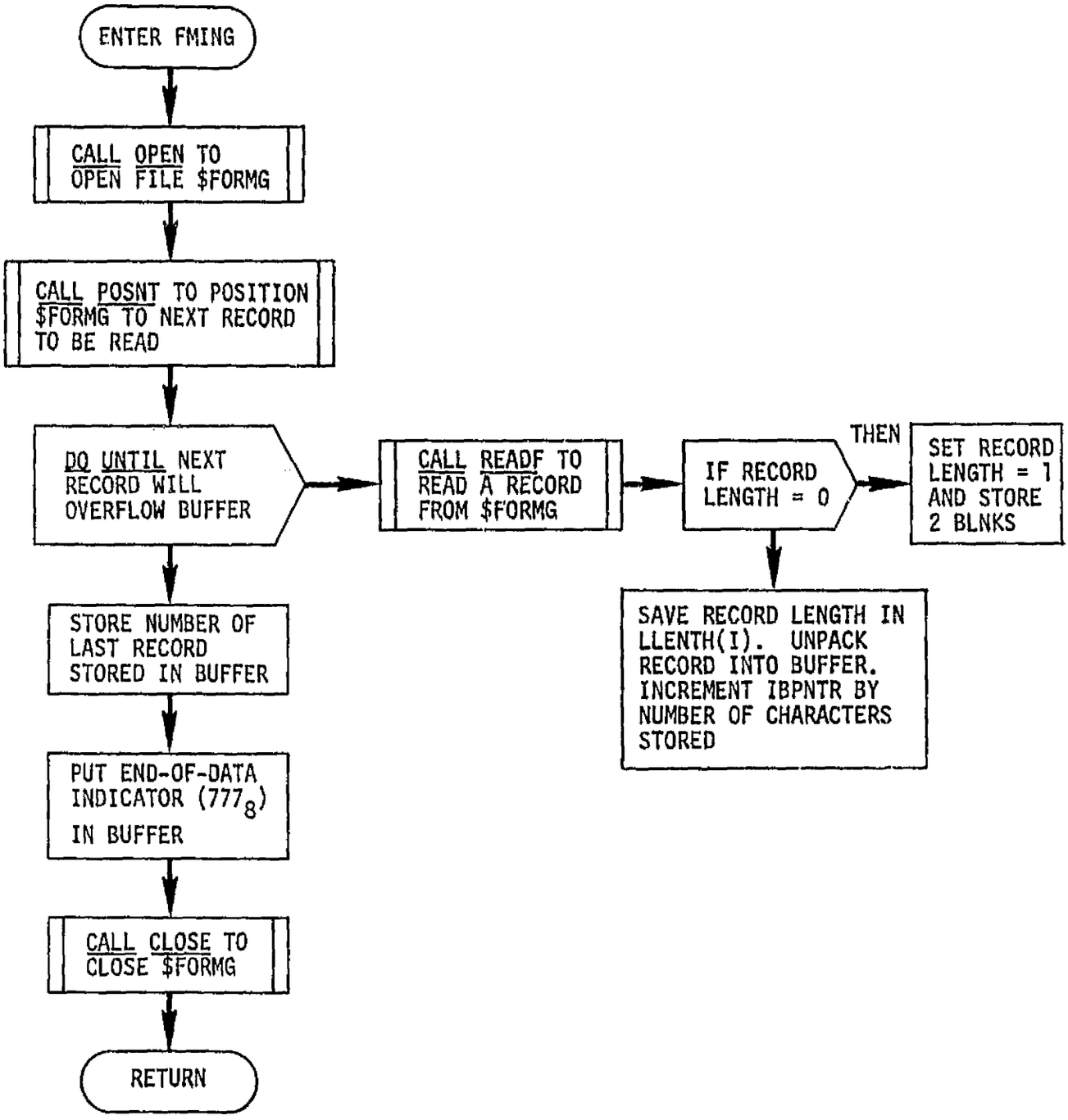
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
FORBUF		Intg	O		A		Name of buffer
IBPNTR		Intg	O		A		Total number of characters stored in FORBUF
IDCB		Intg	I		A		File manager data control block
IERR		Intg	I/O		A		File manager error flag
JCR		Intg	I		C		Unit containing FORM file
JPMAX		Intg	I		A		Length of FORBUF
LLENTH		Intg	O		A		Array containing lengths of records stored in FORBUF
LU		Intg	I		C		User terminal unit number
NAMFRM		Intg	I		A		Name of file containing the form

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.10-1.- FMING functional logic flow.

8.5.11 Routine Name - Main Program LINDG

8.5.11.1 Purpose

The sole purpose of LINDG is to provide a mechanism whereby the routine LTMAG can be used by both DOC and LTBLD. LINDG allows LTMAG to be used by DOC, while a similar routine allows LTMAG to be used by LTBLD.

8.5.11.2 Functional Description

The only function of LINDG is to call LTMAG, then return to DOC. There are no inputs, no outputs, no internal variables, and no executable logic.

8.5.11.3 Assumptions and Limitations

None.

8.5.11.4 Method

None.

8.5.11.5 Routine Input/Output Variables

None.

8.5.11.6 Functional Logic Flow

The functional logic flow for LINDG is presented in figure 8.5.11-1.

8.5.11.7 Diagnostics and Debug

None.

8.5.11.8 Special Comments

None.

8.5.11.9 References

None.

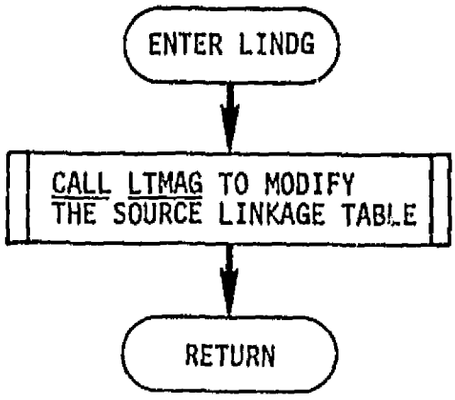


Figure 8.5.11-1.- LINDG functional logic flow.

8.5.12 Routine Name - Main Program LNXDG

8.5.12.1 Purpose

The sole purpose of LNXDG is to call subroutine LTXIG and to return to DOC. LTXIG is used by both LTBLD and DOC. However, the main program of a segment must contain the name of the master segment that called it. Therefore, to avoid duplicating LTXIG with that single difference, LNXDG was written to call LTXIG and return to DOC. A similar routine exists for LTBLD.

There are no inputs, no outputs, no internal variables, and no program logic.

8.5.12.2 Functional Description

None.

8.5.12.3 Assumptions and Limitations

None.

8.5.12.4 Method

None.

8.5.12.5 Routine Input/Output Variables

None.

8.5.12.6 Functional Logic Flow

The functional logic flow for LNXDG is presented in figure 8.5.12-1.

8.5.12.7 Diagnostics and Debug

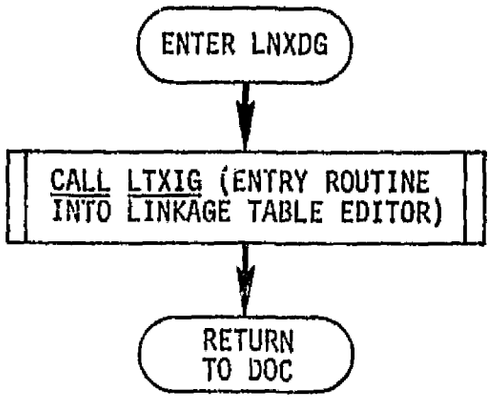
None.

8.5.12.8 Special Comments

None.

8.5.12.9 References

None.



Page 1 of 1

Figure 8.5.12-1.- LNXDG functional logic flow.

8.5.13 Subroutine LPRMG

8.5.13.1 Purpose

The purpose of this routine is to display extended prompts when requested by the Linkage Table Editor.

Special Note

This routine is almost identical to the FDS Executive routine named XTPRM. The function performed by LPRMG for the Linkage Table Editor is identical to the function performed by XTPRM for the Interface Table Editor. LPRMG was developed by copying XTPRM and changing the name of the PROMPT file to be read. The modified copy was then given the name LPRMG to avoid conflict with the original.

For this reason, LPRMG is not documented separately. Instead, the reader is referred to the documentation for subroutine XTPRM in the FDS-1 Executive documentation.

8.5.14 Subroutine LTDIG

8.5.14.1 Purpose

The purpose of this subroutine is to maintain a directory of LINKAGE TABLE and PROMPT files that are currently defined in the system. Each entry in the directory contains the file name, the six-character document and segment names, the two-character document and segment ID's, the version number, and a short (up to 48 characters) description of the contents of the file.

8.5.14.2 Functional Description

LTDIG is called either to add an entry or to delete an entry. In either case, the directory file is copied onto a new file, with the specified change being incorporated into the new file. If an entry is to be added, the present file is searched for the specified name. If the name is found, that entry is not copied to the new file. If the name is not found, the new entry is simply added to the new file.

If an entry is to be deleted, all entries are copied into the new file except the entry to be deleted. After the new file is completed, the old file is purged and the new file is given the name of the old file. If an entry was deleted, the file corresponding to the deleted entry is also purged from the disk.

8.5.14.3 Assumptions and Limitations

None.

8.5.14.4 Method

None.

8.5.14.5 Routine Input/Output Variables

The LTDIG input/output variables are presented in table 8.5.14-I.

8.5.14.6 Functional Logic Flow

The functional logic flow for LTDIG is presented in figure 8.5.14-1.

8.5.14.7 Diagnostics and Debug

None.

8.5.14.8 Special Comments

None.

8.5.14.9 References

None.

TABLE 8.5.14-I.- INPUT/OUTPUT VARIABLES

Routine LTDIG

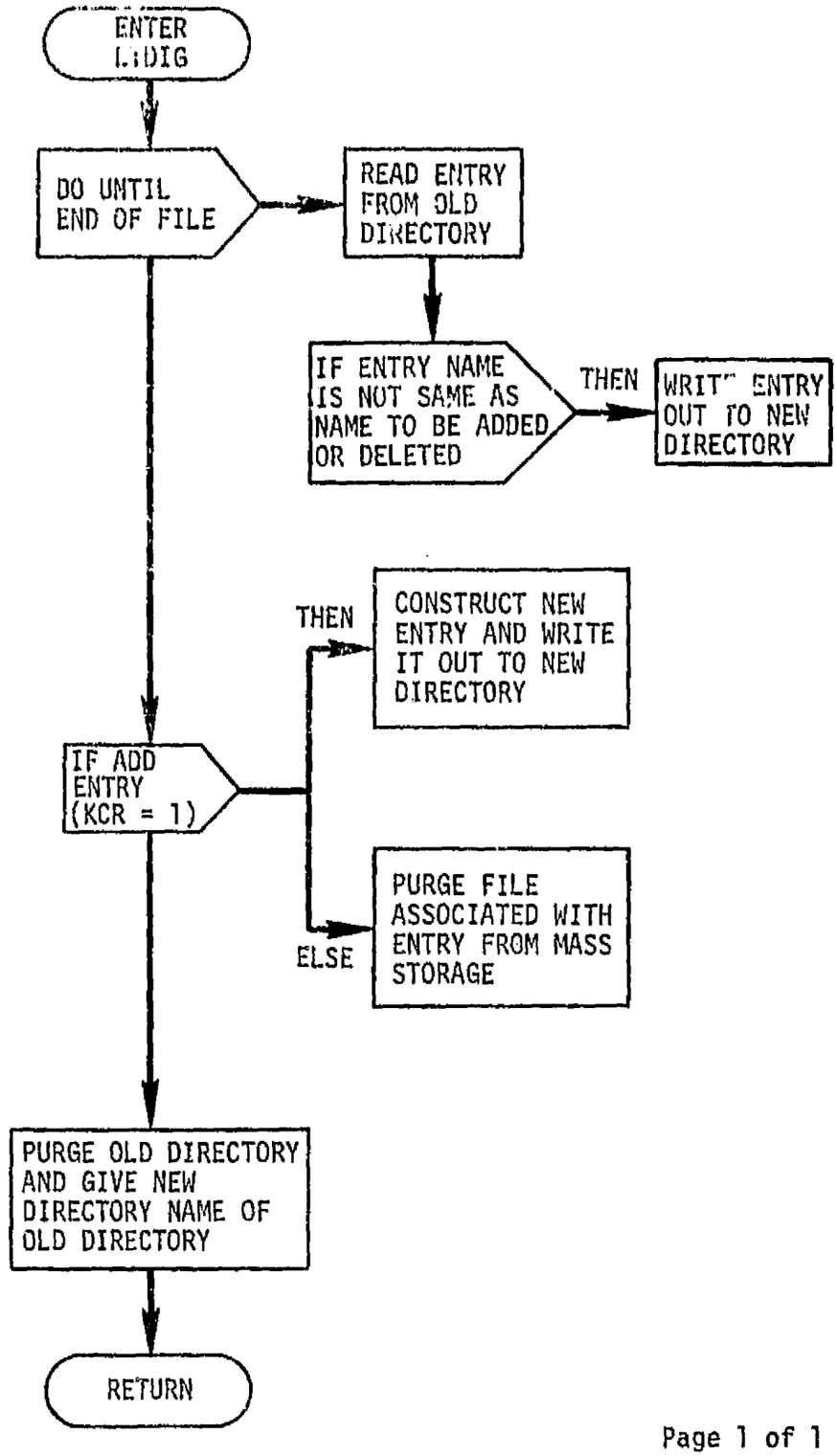
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
DOCID		2CH	I		C		Two-character document ID
DOCNAM		6CH	I		C		Six-character document name
ICR		Intg	I		C		Cartridge number of files
ISECU		2CH	I		C		Security code for files
JCR		Intg	I		C		Cartridge number - directory
JSEC		2CH	I		C		Security code for directory
KCR		Intg	I		A		Option: 1 = add 0 = delete
NAME		6CH	I		A		Entry name
SEGID		2CH	I		C		Two-character segment ID
SEGNAM		6CH	I		C		Six-character segment name

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.14-1.- LTDIG functional logic flow.

8.5.15 Subroutine LTIFG

8.5.15.1 Purpose

The purpose of this subroutine is to write the new linkage table, which has been constructed in common, out to a disk file.

8.5.15.2 Functional Description

The length of the file to be created is computed, and a new disk file is created. If a file already exists by the same name, the old file is purged. The main body of the linkage table is then written to the new file. If a literal record exists, it is then written to the file as the second record. If the new linkage table was created from a previous linkage table, the name of the old file is tested to see if it was a nondefault table that belonged to the current user. If it was, the user is given the opportunity to delete the old file.

8.5.15.3 Assumptions and Limitations

None.

8.5.15.4 Method

None.

8.5.15.5 Routine Input/Output Variables

The LTIFG input/output variables are presented in table 8.5.15-I.

8.5.15.6 Functional Logic Flow

The functional logic flow for LTIFG is presented in figure 8.5.15-1.

8.5.15.7 Diagnostics and Debug

None.

8.5.15.8 Special Comments

None.

8.5.15.9 References

None.

TABLE 8.5.15-I.- INPUT/OUTPUT VARIABLES

Routine LTIFG

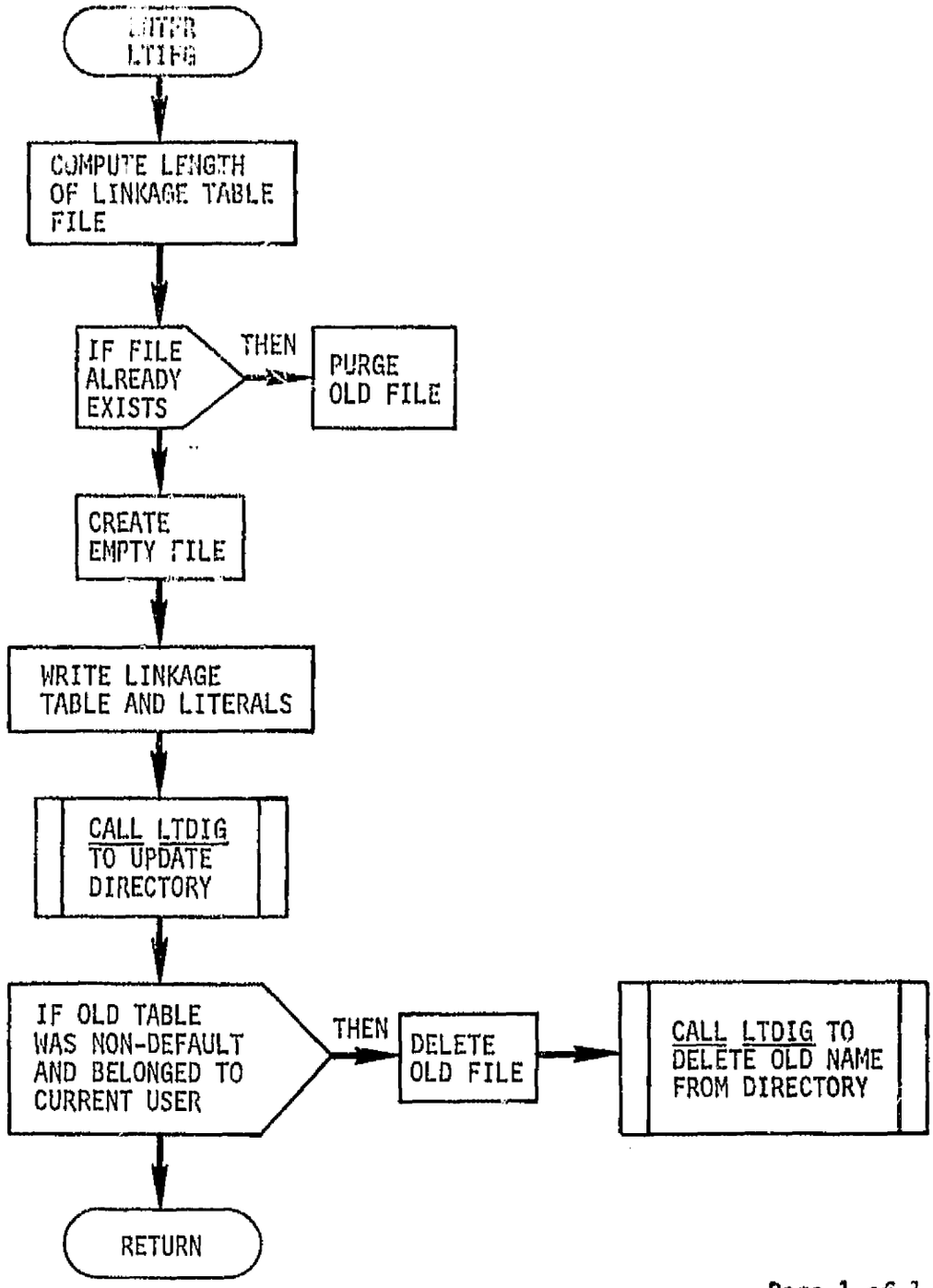
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
HEDR		Intg	I		C		Linkage table header record
ICR		Intg	I		C		Linkage table file cartridge number
ISECU		2CH	I		C		Security code
LITLEN		Intg	I		C		Length of literal area
LITPTR		Intg	I		C		Pointer to literal area
LTNAM		6CH	I		C		Old linkage table name
LU		Intg	I		C		User terminal unit number
NLTNAM		6CH	I		C		New linkage table name
NOPARM		Intg	I		C		Number of parameters
VALFLG		Intg	O		C		

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.15-1.- LTIFG functional logic flow.

8.5.16 Subroutine LTMAG

8.5.16.1 Purpose

The purpose of subroutine LTMAG is to initialize various common parameters and to direct the execution flow through the subroutines that actually create linkage tables. The creation of linkage tables from either DDT's or other linkage tables is supported.

8.5.16.2 Functional Description

LTMAG provides two different sets of capabilities; one set is available to LTBLD, and the other is available to DOC. Specifically, if called from LTBLD, either a DDT or another linkage table may be used as input, a new LINKAGE TABLE file will be built. If called from DOC, only a linkage table may be used as input, and the user is given the option to produce a new LINKAGE TABLE file. LTMAG cannot call the Linkage Table Editor directly due to the limitation of memory size on the HP21MX. For this reason, instead of simply calling the editor when it is time to complete the table, control is passed back to the master segment, which in turn calls the editor. After the editor has completed its execution, LTMAG is called a second time to complete its processing. This is accomplished by having the master segment (LTBLD or DOC) set a flag before each call to LTMAG indicating which call is being executed.

8.5.16.3 Assumptions and Limitations

None.

8.5.16.4 Method

None.

8.5.16.5 Routine Input/Output Variables

The LTMAG input/output variables are presented in table 8.5.16-I.

8.5.16.6 Functional Logic Flow

The functional logic flow for LTMAG is presented in figure 8.5.16-1.

8.5.16.7 Diagnostics and Debug

None.

8.5.16.8 Special Comments

None.

8.5.16.9 References

None.

TABLE 8.5.16-I.- INPUT/OUTPUT VARIABLES

Routine LTMAG

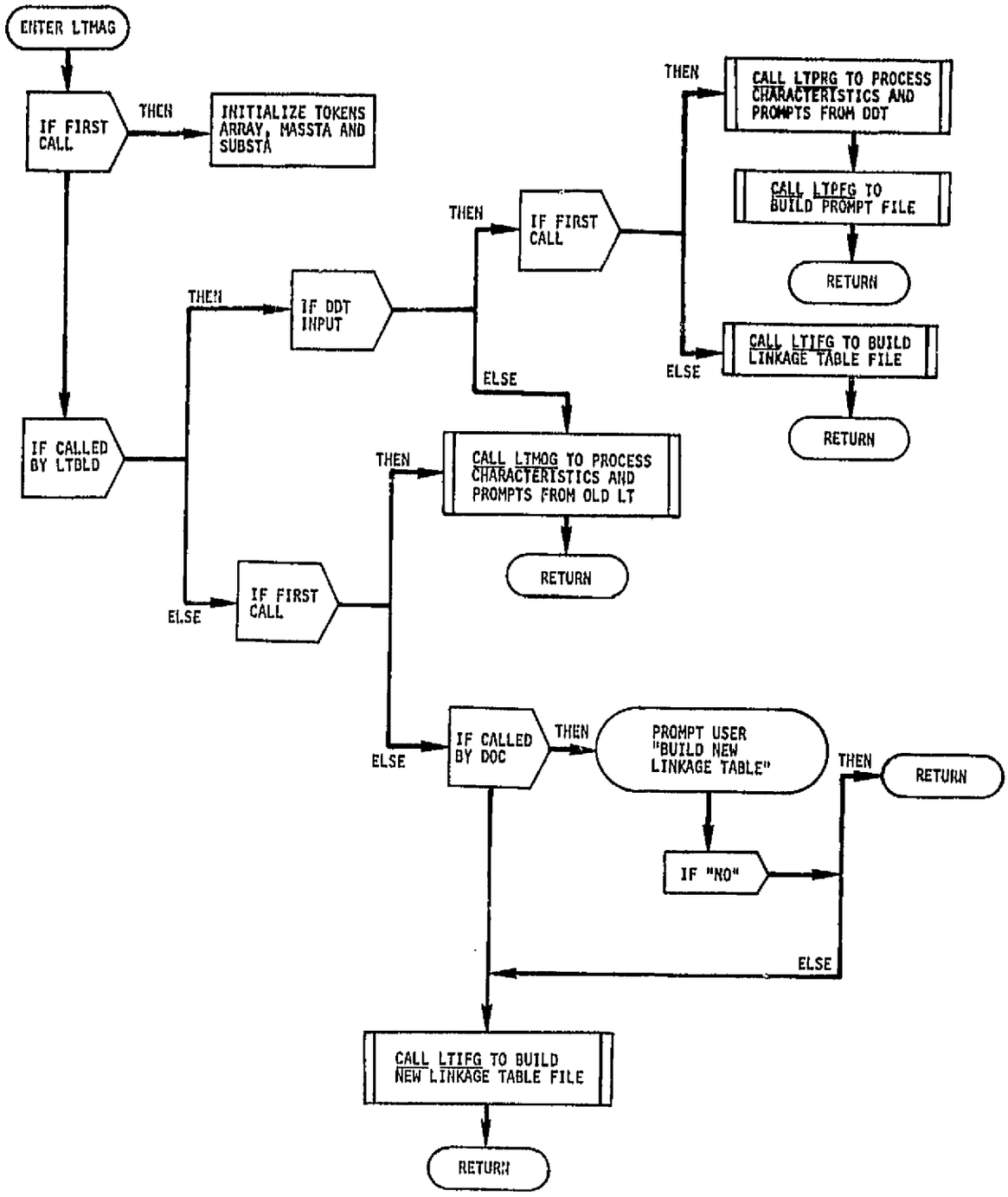
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
DDTRG		Intg	I		C		DDT input flag 0 = LT 1 = DDT
HEDR		Intg	O		C		Common buffer containing LT
ISW		Intg	I		C		1 = first call 2 = second call
LITLEN		Intg	O		C		Length of literal array
LITPTR		Intg	O		C		Beginning of literal array
MASSTA		Intg	O		C		
NLTNAM		6CH	I		C		Name of new linkage table
NOPARM		Intg	I		C		Number of parameters in LT
PROFLG		Intg	I		C		Processor flag 0 = LTBLD 1 = DOC
SUBSTA		Intg	O		C		
TOKENS		Intg	O		C		
VALFLG		Intg	O		C		

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.16-1.- LTMAG functional logic flow.

8.5.17 Subroutine LTMOG

8.5.17.1 Purpose

LTMOG is called when a new linkage table is to be created from an existing one. LTMOG initializes the common buffer with the old linkage table and various control flags in preparation for the modification of the table by the Linkage Table Editor.

8.5.17.2 Functional Description

A linkage table is valid only if it has the same version number as the current PROMPT file. Therefore, the first step in using a linkage table is to determine its validity by reading and comparing the version numbers from the specified LINKAGE TABLE and PROMPT files. If they agree, the linkage table (with literals) is read into common. The short prompts are then read from the PROMPT file and placed in common with the linkage table. Control is then returned to LTMAG for completion of the new linkage table.

8.5.17.3 Assumptions and Limitations

None.

8.5.17.4 Method

None.

8.5.17.5 Routine Input/Output Variables

The LTMOG input/output variables are presented in table 8.5.17-I.

8.5.17.6 Functional Logic Flow

The functional logic flow for LTMOG is presented in figure 8.5.17-1.

8.5.17.7 Diagnostics and Debug

None.

8.5.17.8 Special Comments

None.

8.5.17.9 References

None.

TABLE 8.5.17-I.- INPUT/OUTPUT VARIABLES

Routine LTMOG

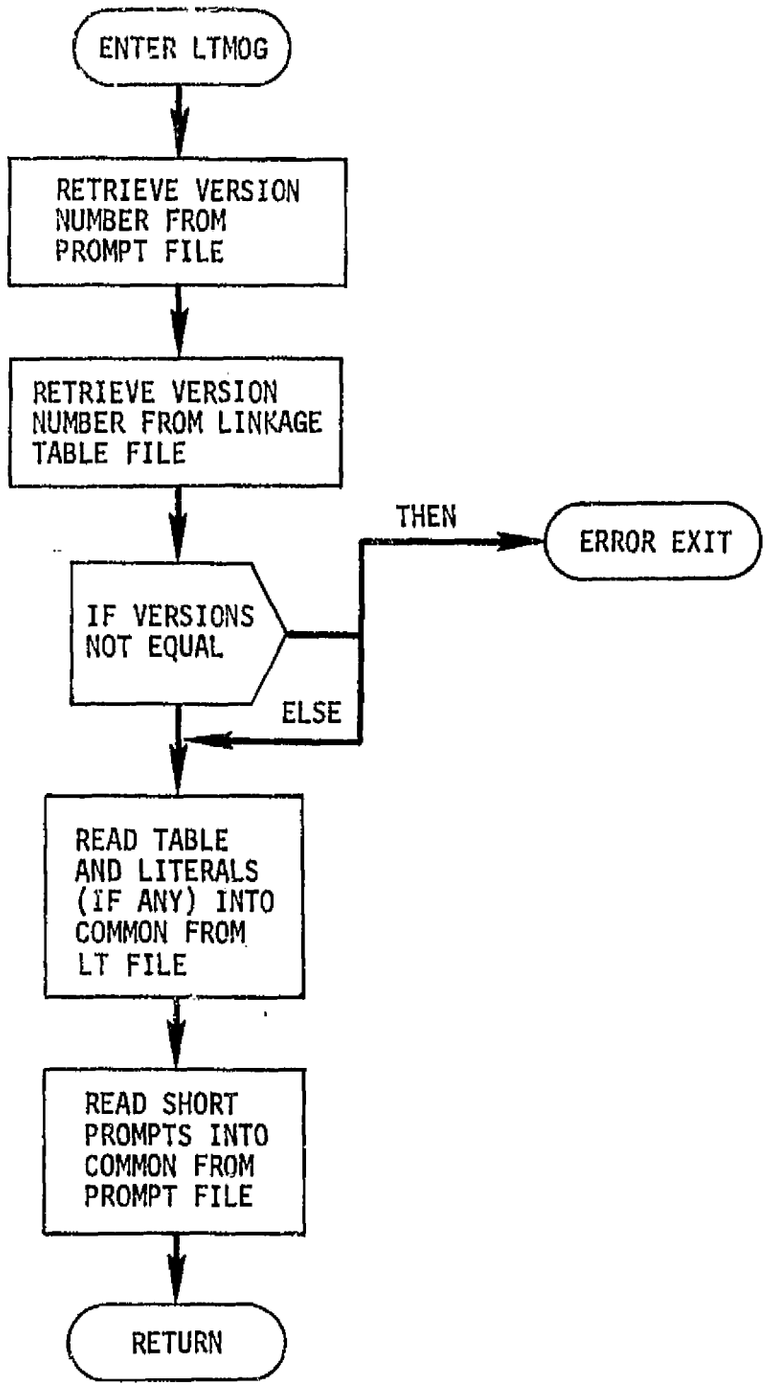
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
HEDR		Intg	O		Disk		Seven-word header record
ICR		Intg	I		C		Cartridge number of LT
INDX		Intg	O		Disk		
ISECU		2CH	I		C		Security code
LITLEN		Intg	O		Disk		Length of literal array
LITPTR		Intg	O		Disk		Pointer to literal array
LTNAM		6CH	I		C		Old linkage table name
MASSTA		Intg	O		Internal		Master state
NAMDDT		6CH	I		C		Name of DDT file
NLTNAM		6CH	I		C		Name of new linkage table
NOPARM		Intg	O		Disk		Number of parameters
PARMS		Intg	O		Disk		Body of linkage table buffer
SUBSTA		Intg	O		Internal		Substate
VERS		Intg	O		Disk		Version number

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.17-1.- LTMOG functional logic flow.

8.5.18 Subroutine LTNMG

8.5.18.1 Purpose

The purpose of this subroutine is to prompt the user for the name of the linkage table to be used for this execution and to verify its validity by searching a directory of valid names. A name is valid only if it exists in the directory, and the associated document and segment identifiers agree with the document and segment specified for the current execution.

8.5.18.2 Functional Description

The user is first prompted for the desired linkage table name. Linkage table names are constructed from three parts. The first character is always an "L." The last character is supplied by the program and is either an asterisk or the single-character user ID. The middle four characters are either provided by the user, or, if a default name is requested, are constructed from the two-character document and segment ID's.

The user may respond in the following ways to the prompt.

Response	Action
?	List directory
%	Abort session
Space, carriage return	Use default name (LDDSS*)
Space, user ID	Use user-default name (LDDSSU)
XXXX	Use table named LXXXXU
XXXXY	Use table names LXXXXY

Where DD = two-character document code

SS = two-character segment code

* = last character of master default name

U = one-character user ID

XXXX = user-specified name

Y = user ID of another user

After the user has specified a name, the directory of existing names is searched for that name. If the name is not found, the user is notified and reprompted. If the name is found, but the table with that name is for a different document

segment, the user is notified and reprompted. When the name is determined to be valid, it is stored in common.

8.5.18.3 Assumptions and Limitations

LTNMG assumes that a directory name $LTDIG exists on the HP21MX mass storage system.

8.5.18.4 Method

None.

8.5.18.5 Routine Input/Output Variables

The LTNMG input/output variables are presented in table 8.5.18-I.

8.5.18.6 Functional Logic Flow

The functional logic flow for LTNMG is presented in figure 8.5.18-1.

8.5.18.7 Diagnostics and Debug

Diagnostic	Action
Unable to open linkage table directory.	Abort.
You have requested someone else's default table.	Prompt to verify user's intentions.
Requested linkage table not in directory.	Reprompt.
Requested linkage table is not for requested document and segment.	Reprompt.
Search for compatible linkage table name abandoned.	Abort.

8.5.18.8 Special Comments

None.

8.5.18.9 References

None.

TABLE 8.5.18-I.- INPUT/OUTPUT VARIABLES

Routine LTMNG

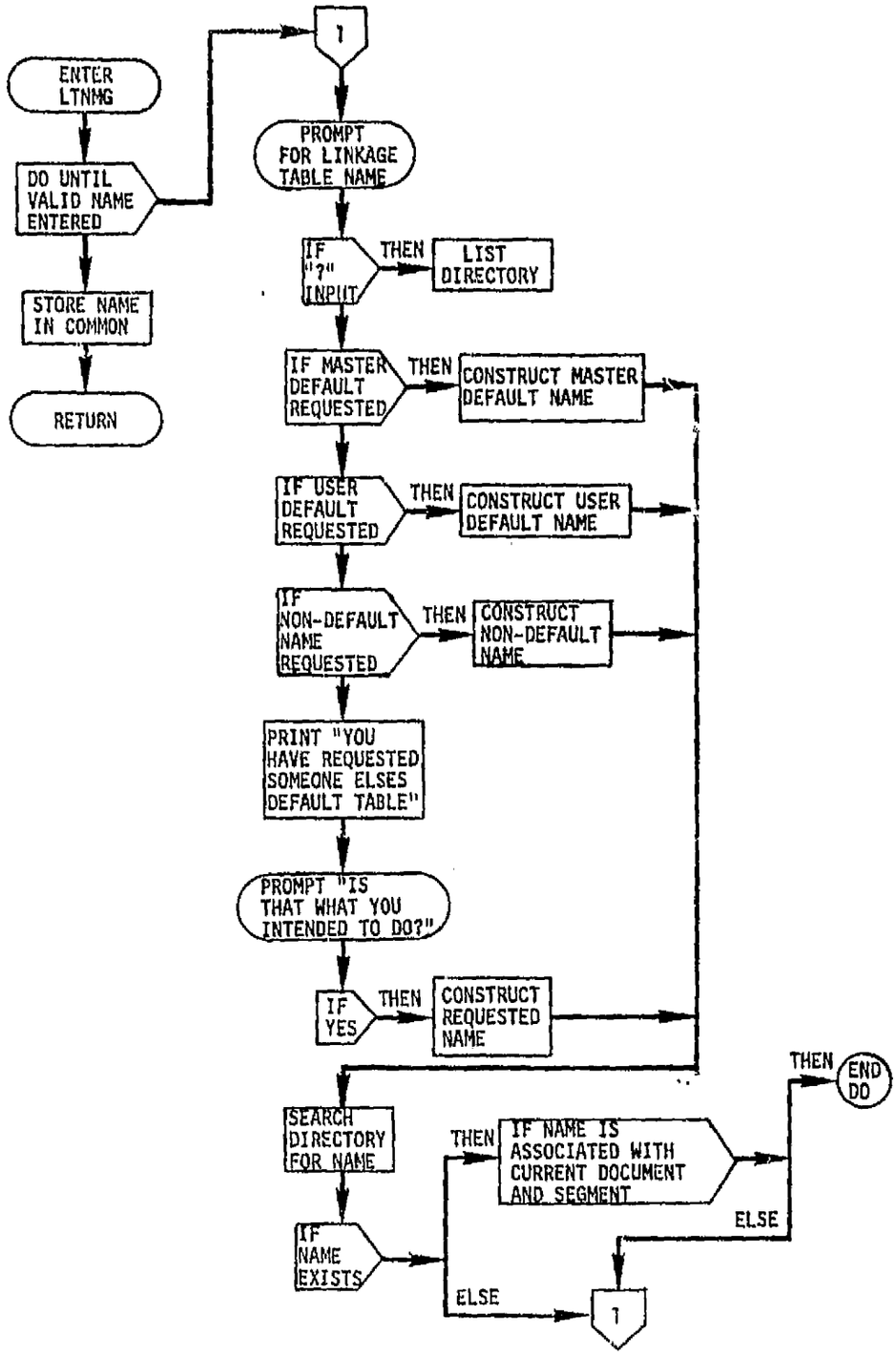
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
DOCID		2CH	I		C		Two-character document ID
JCR		Intg	I		C		Cartridge containing directory
JSEQ		2CH	I		C		Security code for directory
LTNAM		6CH	O		C		Linkage table name
SEGID		2CH	I		C		Two-character segment ID
USERID		Intg	I		C		One-character user ID
INBUF		4CH	I		T		User specification of linkage table name

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.18-1.- LTMAG functional logic flow.

8.5.19 Subroutine LTNNG

8.5.19.1 Purpose

The purpose of this subroutine is to prompt the user for the name of the linkage table to be created during this execution, and to determine whether or not the requested name already exists in the directory. If the name does not already exist, it is stored in common and control is returned to the calling program. If it does exist, the program determines whether or not it is appropriate to recreate the table, and if that is what the user intended.

8.5.19.2 Functional Description

If a DDT is being used as input, then the master default name is automatically created. Otherwise, LTNNG must determine the name to be created.

The user is first prompted for the desired name for the new linkage table. The acceptable responses and the linkage table naming conventions are the same as those described in LTNMG, with the exception that utilizing other user's ID's is not permitted.

After the user has specified a name, the type of name requested must be determined.

The creation of master default tables is only permitted from LTBLD. If LTNNG is called from DOC, a diagnostic is displayed and the user is reprompted.

If a user-default name is requested, the request is compared with the user ID for the session, and the request is rejected if they are different.

Similarly, if a nondefault name is requested, a name with another user's ID will be rejected.

When a name has been supplied that satisfies the above criteria, it is stored in common and control is returned to the calling program.

8.5.19.3 Assumptions and Limitations

LTNNG assumes that a directory named $LTDIG exists on the HP21MX mass storage system.

8.5.19.4 Method

None.

8.5.19.5 Routine Input/Output Variables

The LTNNG input/output variables are presented in table 8.5.19-I.

8.5.19.6 Functional Logic Flow

The functional logic flow for LTNNG is presented in figure 8.5.19-I.

8.5.19.7 Diagnostics and Debug

Diagnostic	Action
Master default table may not be recreated from DOC.	Reprompt (occurs only if called from DOC).
You may not create a table with another user's ID.	Reprompt.
Unable to open Linkage Table Directory.	Abort.
Linkage table named XXXXXX already exists.	Ask user if he wants to recreate this table.
A linkage table named XXXXXX already exists for another document segment. You must select another name.	Reprompt.

8.5.19.8 Special Comments

None.

8.5.19.9 References

None.

TABLE 8.5.19-I.- INPUT/OUTPUT VARIABLES

Routine LTNNG

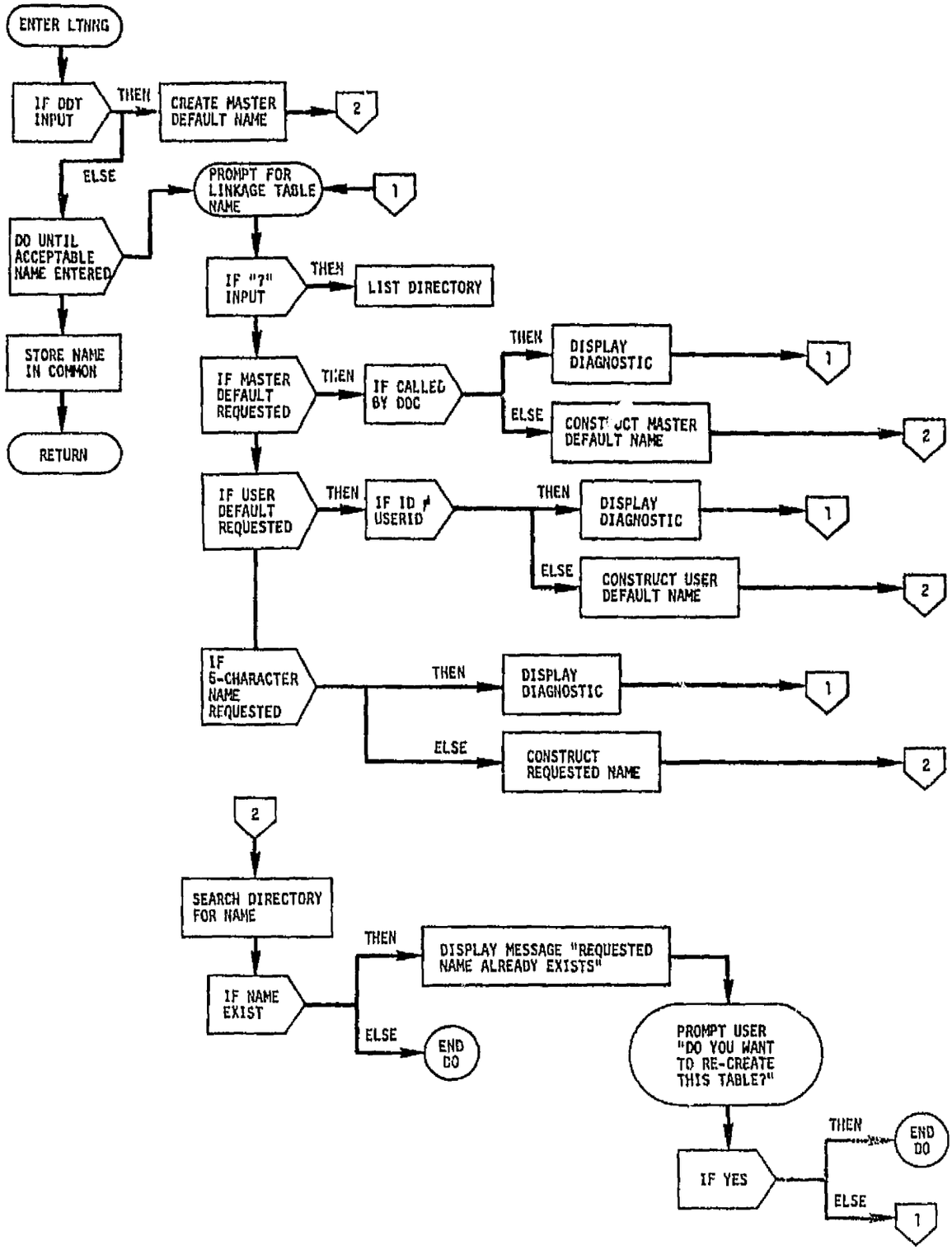
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
DOCID		2CH	I		C		Two-character document ID
JCR		Intg	I		C		Cartridge containing directory
JSEQ		2CH	I		C		Security code for directory
NLTNAM		6CH	O		C		New linkage table name
PROFLG		Intg	I		C		Processor flag 0 = LTBLD 1 = DOC
SEGID		2CH	I		C		Two-character segment ID
USERID		Intg	I		C		One-character user ID
INBUF		4CH	I		T		User specification of new linkage table name

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.19-1.- LTNNG functional logic flow.

8.5.20 Subroutine LTPFG

8.5.20.1 Purpose

The purpose of this subroutine is to create the PROMPT file that is used by the Linkage Table Editor. The short prompts (parameter names) and extended prompts (parameter descriptions) are constructed from information contained in the DDT.

8.5.20.2 Functional Description

Upon entry into LTPFG, the name of the current linkage table is extracted from the linkage table header record. The first character of the name is changed to a "P" to indicate PROMPT file. A new disk file is then created using the prompt file name.

The short prompts are written directly from the linkage table buffer into the first two records of the PROMPT file. The extended prompt records are produced by reading each entry from the DDT, and then writing that entry onto the PROMPT file, one entry per record.

8.5.20.3 Assumptions and Limitations

None.

8.5.20.4 Method

None.

8.5.20.5 Routine Input/Output Variables

The LTPFG input/output variables are presented in table 8.5.20-I.

8.5.20.6 Functional Logic Flow

The functional logic flow for LTPFG is presented in figure 8.5.20-1.

8.5.20.7 Diagnostics and Debug

None.

8.5.20.8 Special Comments

None.

8.5.20.9 References

None.

TABLE 8.5.20-I.- INPUT/OUTPUT VARIABLES

Routine LTPFG

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
HEDR		Intg	I		C		Linkage table header
PARMS		Intg	I		C		Linkage table buffer
PRMNAM		6CH	O		C		PROMPT file name
NOPARM		Intg	I		C		Number of parameters in LT
ISECU		2CH	I		C		Security code
ICR		Intg	I		C		Cartridge number for PROMPT file
NAMDDT		6CH	I		C		Name of DDT file
ABSTR		Intg	O		LTPFG		PROMPT file abstract
IBUF		Intg	I/O		T		Input/output buffer for file manager

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

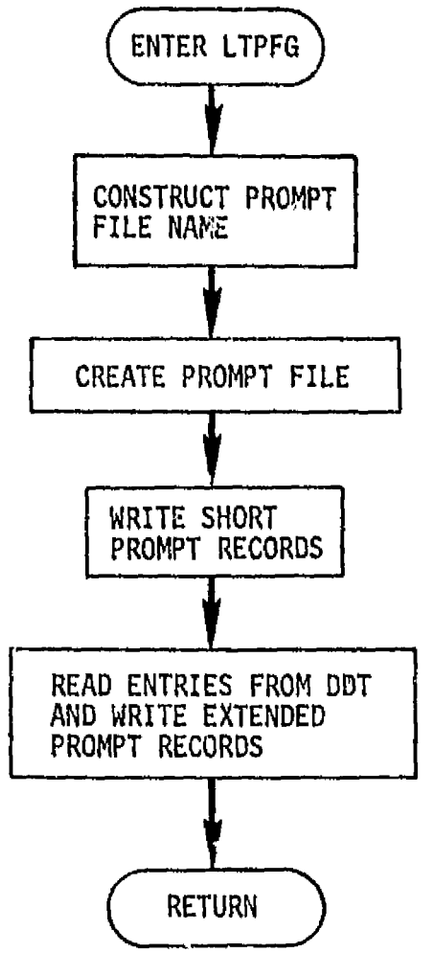


Figure 8.5.20-1.- LTPFG functional logic f

8.5.21 Subroutine LTPRG

8.5.21.1 Purpose

The purpose of this routine is to manage the creation of a new linkage table from the information contained in the DDT. The linkage table, with short prompts, is constructed in common in exactly the same format as an interface table. This permits the use of the unmodified Interface Table Editor as the editor for the linkage tables.

8.5.21.2 Functional Description

LTPRG first extracts data from the header record of the DDT and constructs the header of the linkage table. The header contains the number of parameters, the version number, the linkage table name, and the document segment and user codes associated with this table. The number of parameters is then used to determine the number of records to be read from the DDT. As each record is read, the required information is stored in common for that parameter. This information is the short prompt, class, type, dimension, and size.

8.5.21.3 Assumptions and Limitations

None.

8.5.21.4 Method

None.

8.5.21.5 Routine Input/Output Variables

The LTPRG input/output variables are presented in table 8.5.21-I.

8.5.21.6 Functional Logic Flow

The functional logic flow for LTPRG is presented in figure 8.5.21-1.

8.5.21.7 Diagnostics and Debug

None.

8.5.21.8 Special Comments

None.

8.5.21.9 References

None.

TABLE 8.5.21-I.- INPUT/OUTPUT VARIABLES

Routine LTPRC

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
DOCID		2CH	I		C		Two-character document ID
HEDR			O		DDT		Seven-word LT header
ICR		Intg	I		C		Cartridge number containing LT
LU		Intg	I		C		Unit number of user's terminal
MASSTA		Intg	O		C		Master state
NAMDDT		6CH	I		C		Name of DDT file
NLTNAM		6CH	I		C		Name of new LT file
NOPARM		Intg	O		DDT		Number of parameters
PARMS			O		DDT		Linkage table buffer
SEGID		2CH	I		C		Two-character segment ID
VERS		Intg	O		DDT		Version number

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

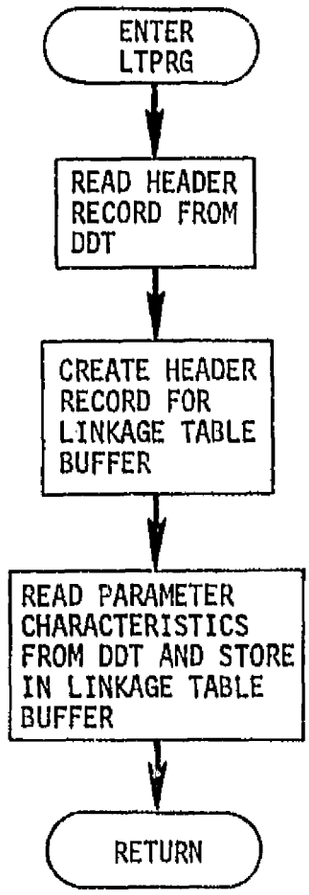


Figure 8.5.21-1.- LTPRG functional logic flow.

8.5.22 Subroutine LTXIG

8.5.22.1 Purpose

The purpose of LTXIG is to initialize all of the various flags that are used by the Linkage Table Editor, and to call XINIX, which is the entry routine into the editor. If LTXIG is called from DOC, then it will refuse to exit until the linkage table is complete.

8.5.22.2 Functional Description

The Linkage Table Editor is initialized by setting a series of control flags in common. If there are any literals in the literal array, subroutine XEINT is called to unpack them into a usable format. Subroutine XINIX is then called to access the editor. If LTXIG was called from LTBLD, control is returned to the calling program at this point. If it has been called from DOC, the linkage table is tested for completeness. If it is incomplete, a message is sent to the user, and XINIX is called again. This process is repeated, if necessary, until the table is complete.

8.5.22.3 Assumptions and Limitations

None.

8.5.22.4 Method

None.

8.5.22.5 Routine Input/Output Variables

The LTXIG input/output variables are presented in table 8.5.22-I.

8.5.22.6 Functional Logic Flow

The functional logic flow for LTXIG is presented in figure 8.5.22-1.

8.5.22.7 Diagnostics and Debug

None.

8.5.22.8 Special Comments

None.

8.5.22.9 References

None.

TABLE 8.5.22-I.- INPUT/OUTPUT VARIABLES

Routine LTXIG

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
ARGNO		Intg	O		C		Current argument number
DEBUG		Intg	O		C		Debug output flag
DIRECT		Intg	O		C		Directive list
END		Intg	O		C		Last two bits (=1 & 0)
FLAGS		Intg	I		C		Source of debug flag
ICLASS		Intg	O		C		Class I/O number
ISIZES		Intg	O		C		Length of each parameter type
LITLEN		Intg	I		C		Length of literal array
LITDWN		Intg	O		C		Pointer to first literal
LITPTR		Intg	O		C		Pointer to first literal
LSTFLG		Intg	O		C		List flag
LU		Intg	I		C		Logical unit of user's terminal
MASSTA		Intg	O		C		Master state
NARG		Intg	O		C		Index to short prompts
NEWTAB		Intg	O		C		New table name
NUMARG		Intg	I		C		Number of arguments
NUMDIR		Intg	O		C		Number of directives

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

TABLE 8.5.22-I.- Concluded

Routine LTXIG

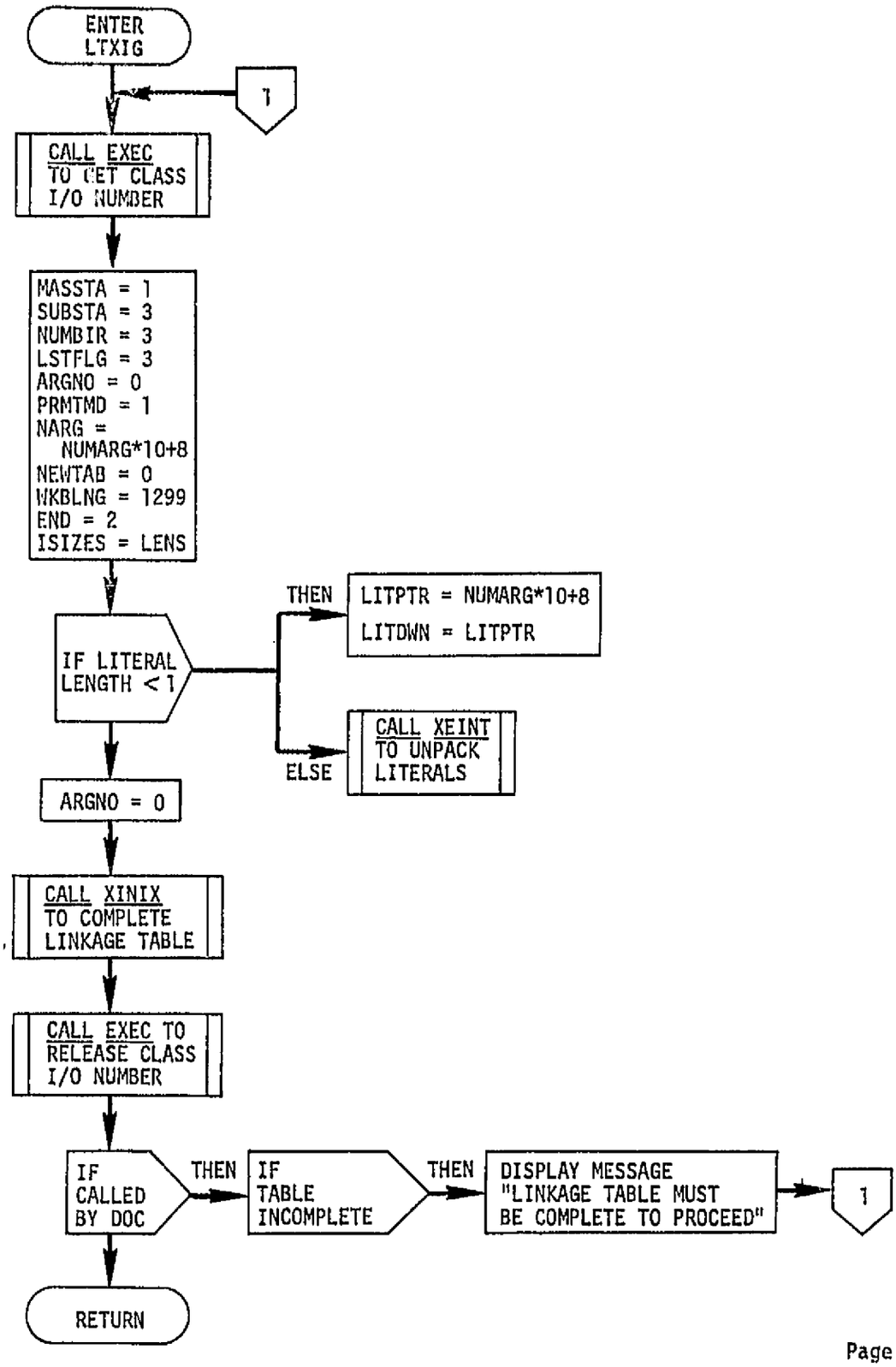
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
PRMTMD		Intg	O		C		PRINT mode
PROFLG		Intg	I		C		Processor flag 0 = LTBLD 1 = DOC
SUBSTA		Intg	O		C		Substate
WKBLNG		Intg	O		C		Length of LT buffer

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.22-1.- LTXIG functional logic flow.

8.5.23 Subroutine MERG

8.5.23.1 Purpose

The purpose of MERG is to merge the reformatted data into the reserved blank spaces in the form. MERG will continue to perform this function until the last blank space in the buffer has been used. When this happens, the completed portion of the segment is paged out to the OUTPUT file, and the next buffer of blank forms is paged into memory. The merging function is then resumed.

8.5.23.2 Functional Description

In general, a complete blank form will be larger than the available internal buffer space. For this reason, the form is paged into memory in portions that will fit in the buffer. The form is read in line-by-line. Because a blank data field cannot extend from one line to the next, there will never be a left bracket in the buffer that is not accompanied by a right bracket.

MERG is called for each data value that has been retrieved and reformatted by DOPRG. MERG searches the FORBUF array for the next left bracket. The data value is placed in the FORBUF array, beginning with the character position occupied by the left bracket, and continues through the position occupied by the right bracket. If the data value is too long, it is truncated. If it is too short, the remaining spaces in the form are filled with blanks.

If the end-of-data flag is encountered without finding a left bracket, subroutine SOUTG is called to write the completed portion out to the OUTPUT file. Subroutine FMING is then called to bring the next portion of the blank form into the buffer so that processing can be continued.

8.5.23.3 Assumptions and Limitations

None.

8.5.23.4 Method

None.

8.5.23.5 Routine Input/Output Variables

The MERG input/output variables are presented in table 8.5.23-I.

8.5.23.6 Functional Logic Flow

The functional logic flow for MERG is presented in figure 8.5.23-1.

8.5.23.7 Diagnostics and Debug

None.

8.5.23.8 Special Comments

None.

8.5.23.9 References

None.

TABLE 8.5.23-I.- INPUT/OUTPUT VARIABLES

Routine MERG

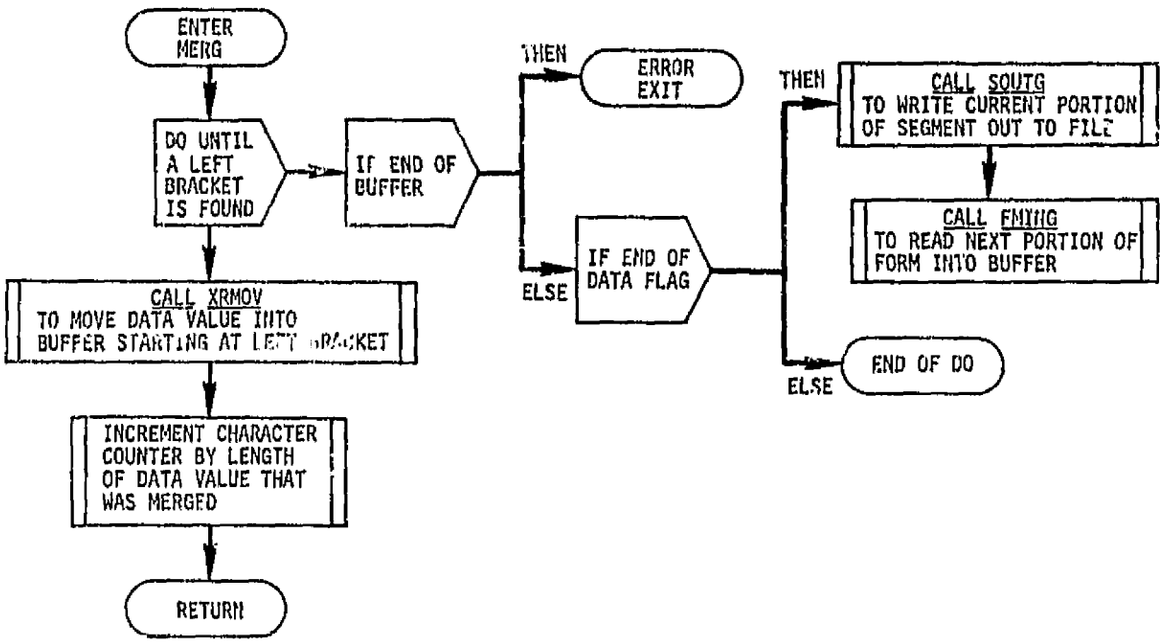
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
FORBUF		Intg	I/O		A		Buffer where data and form are merged.
IBPNTR		Intg	O		A		Number of characters stored in FORBUF.
IDCB		Intg	I/O		A		Data control block for file manager.
IERR		Intg	I/O		A		Error flag for file manager.
IRETC		Intg	I		A		Character length of current value.
IXY		Intg	I		A		Array containing data value.
JPMAX		Intg	I/O		A		Length of buffer.
JPOS		Intg	I		A		Current character position within buffer.
LLENTH		Intg	I/O		A		Array of line lengths within FORBUF.
LU		Intg	I		A		Unit number of user's terminal.
NAMFRM		6CH	I/O		A		Name of FORM file.

NOTES:

TYPE
Free, Dubl, 18CH, Mix
Intg, 2CH, 36CH, Char
Real, 6CH, 72CH, Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.23-1.- MERG functional logic flow.

8.5.24 Subroutine SDISG

8.5.24.1 Purpose

The purpose of this subroutine is to read either the blank form or the merged segment from the disk, and display it at the user's terminal (or an optional hard-copy device).

8.5.24.2 Functional Description

The file to be displayed is indicated by the input variable IOPTN. For IOPTN = 1, the merged segment is to be displayed. For IOPTN = 0, the blank form is to be displayed.

Immediately upon entry, SDISG prompts the user to find out if a display is wanted. If not, control is returned to the calling program. If a display is requested, the user is then prompted for the optional hard-copy unit number. A null response (space, carriage return) will suppress the hard copy.

If a display is requested, the specified file is read and displayed line-by-line. If the display is being sent to one of the HP21MX terminals, it is blocked into 22-line pages to facilitate reading by the user.

After each page, the user is prompted with a "CONTINUE?" to en[illegible]im to request the next page. If the display is going to a non-HP21MX [illegible]l, it is sent in its entirety without paging.

8.5.24.3 Assumptions and Limitations

None.

8.5.24.4 Method

None.

8.5.24.5 Routine Input/Output Variables

The SDISG input/output variables are presented in table 8.5.24-I.

8.5.24.6 Functional Logic Flow

The functional logic flow for SDISG is presented in figure 8.5.24-1.

8.5.24.7 Diagnostics and Debug

None.

8.5.24.8 Special Comments

None.

8.5.24.9 References

None.

TABLE 8.5.24-I.- INPUT/OUTPUT VARIABLES

Routine SDISG

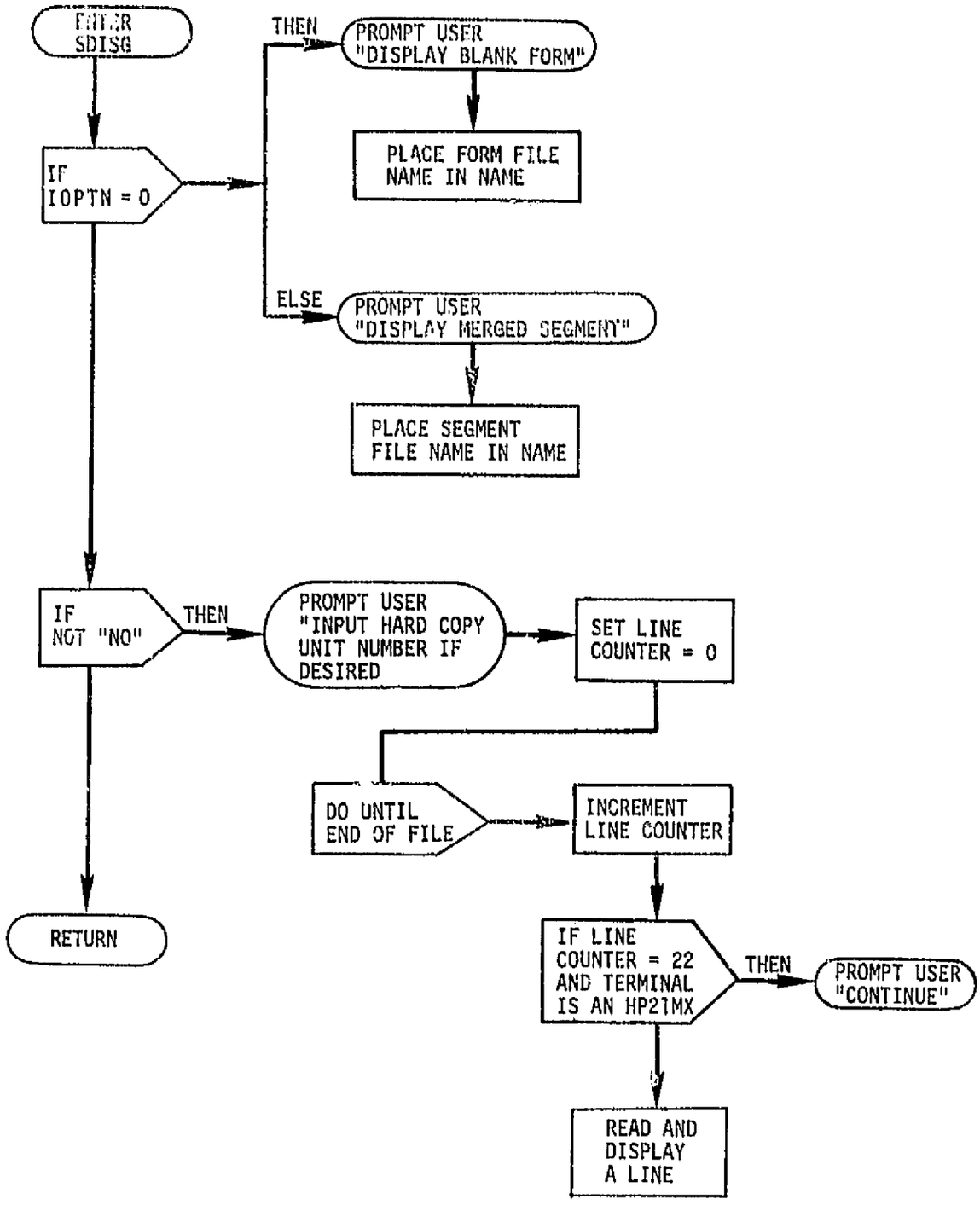
Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
FORBUF		Intg	I/O		A		Buffer through which segment is passed.
IDCB		Intg	I		A		Data control block for file manager.
IERR		Intg	I		A		Error flag for file manager.
LU		Intg	I		A		Unit number of user's terminal.
IOPTN		Intg	I		A		Option = 1, display segment ≠ 1, display blank form

NOTES:

TYPE			
Free	Dubl	18CH	Mix
Intg	2CH	36CH	Char
Real	6CH	72CH	Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory



Page 1 of 1

Figure 8.5.24-1.- SDISG functional logic flow.

8.5.25 Subroutine SOUTG

8.5.25.1 Purpose

The purpose of SOUTG is to page the merged portion of a document segment out to the OUTPUT file. Since segments may be larger than the available internal buffer space, it is necessary to read, process, and store the segments in smaller portions, called pages. These pages are accumulated in an OUTPUT file until the entire segment has been processed. The segment is then ready to be sent to the Daconics.

8.5.25.2 Functional Description

First the OUTPUT file is opened, and the file is positioned to the next record to be written. The page is written out to the file by determining the number of characters in each record, packing them into an A2 format, and writing to the file. As each record is written, the record counter is incremented so that the file can be positioned to the correct record for the next page.

8.5.25.3 Assumptions and Limitations

None.

8.5.25.4 Method

None.

8.5.25.5 Routine Input/Output Variables

The SOUTG input/output variables are listed in table 8.5.25-I.

8.5.25.6 Functional Logic Flow

The functional logic flow for SOUTG is presented in figure 8.5.25-1.

8.5.25.7 Diagnostics and Debug

None.

8.5.25.8 Special Comments

None.

8.5.25.9 References

None.

TABLE 8.5.25-I.- INPUT/OUTPUT VARIABLES

Routine SOUTG

Code symbol	Math symbol	Type	Use	Units	Source	External label	Definition
FORBUF		Intg	I		A		Buffer containing portion of segment.
IDCB		Intg	I		A		Data control block for file manager.
IERR		Intg	I		A		Error flag for file manager.
LLENTH		Intg	I		A		Array containing lengths of each record to be written.
LU		Intg	I		A		Unit number of user's terminal.

NOTES:

TYPE
Free Dubl 18CH Mix
Intg 2CH 36CH Char
Real 6CH 72CH Bin

USE
I = Input
O = Output
I/O = Input/Output

SOURCE
IT = Interface Table
T = Terminal
A = Calling Argument
C = Common
F = Disk File
SAM = System Available Memory

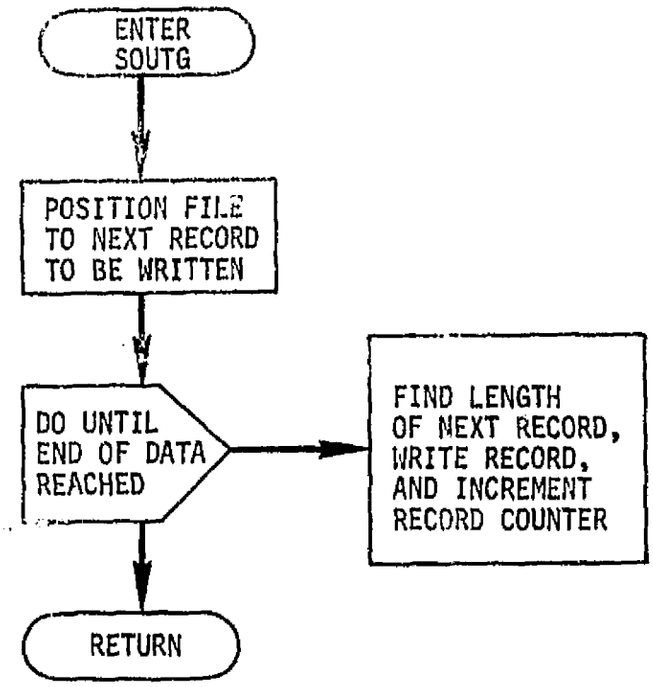


Figure 8.5.25-1.- SOUTG functional logic flow.

☆ U.S. GOVERNMENT PRINTING OFFICE: 1979—671-014/ 1563