AD-A102 185 INPUT OUTPUT COMPUTER SERVICES INC WALTHAM MA F/G 17/2
TWENTY-CHANNEL VOICE RESPONSE SYSTEM.(U)
JUN 81 DOT-TSC-1313
UNCLASSIFIED FAA-RD-81-51 NL

1 of 5
AD A
102185

LEVEL II

(13)

REPORT NO. FAA-RD-81-51

AD A102185

TWENTY-CHANNEL VOICE RESPONSE SYSTEM

INPUT OUTPUT COMPUTER SERVICES, INC.
400 Totten Pond Road
Waltham MA 02154

JUNE 1981
FINAL REPORT

Prepared for
U.S. DEPARTMENT OF TRANSPORTATION
FEDERAL AVIATION ADMINISTRATION
Systems Research and Development Service
Washington DC 20591

(18) FAA-RD, TSC (19) 81-51, FAA-81-5

Technical Report Documentation Page

1. Report No. FAA-RD-81-51	2. Government Accession No. AD-A102 185	3. Recipient's Catalog No.
4. Title and Subtitle (6) TWENTY-CHANNEL VOICE RESPONSE SYSTEM		5. Report Date (11) June 1981
		6. Performing Organization Code
7. Author(s) (12) 435		8. Performing Organization Report No. DOT-TSC-FAA-81-5
9. Performing Organization Name and Address Input Output Computer Services, Inc.* 400 Totten Pond Road Waltham MA 02154		10. Work Unit No. (TRAIS) FA031/R1115
		11. Contract or Grant No. (15) DOT-TSC-1313
12. Sponsoring Agency Name and Address U.S. Department of Transportation Federal Aviation Administration Systems Research and Development Service Washington DC 20591		13. Type of Report and Period Covered (9) Final Report, Mar 1977-Sep 1978
		14. Sponsoring Agency Code

15. Supplementary Notes

*Under Contract to: U.S. Department of Transportation
Research and Special Programs Administration
Transportation Systems Center
Cambridge MA 02142

16. Abstract

This report documents the design and implementation of a Voice Response System (VRS), which provides Direct-User Access (DUA) to the FAA's aviation-weather data base. This system supports 20 independent audio channels, and as of this report, speaks three weather products over a push-button telephone interface: hourly surface observations (SA), terminal forecasts (FT), and forecast winds aloft (GF). The system is implemented on two linked computers: a PDP 11/70 host which maintains the data base, and a PDP 11/34 front-end which manages the weather briefings.

17. Key Words Voice Response System (VRS) Direct-User Access (DUA) Flight Service Station (FSS)	18. Distribution Statement DOCUMENT IS AVAILABLE TO THE PUBLIC THROUGH THE NATIONAL TECHNICAL INFORMATION SERVICE, SPRINGFIELD, VIRGINIA 22161

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	444	

Form DOT F 1700.7 (8-72) Reproduction of completed page authorized

407553

PREFACE

The development work summarized in this final report was carried out by Input Output Computer Services, Inc., under contract to the U.S. Department of Transportation, Transportation Systems Center (DOT/TSC). The research was sponsored by the Federal Aviation Administration (FAA) as part of their Flight Service Station (FSS) automation program.

The system described in this report is intended to provide preflight weather briefings to the aviation community via computer-generated voice output. It is a 20-channel Voice Response System (VRS) which uses Adaptive Differential Pulse Code Modulation (ADPCM) speech-compression techniques and a push-button telephone communication interface for a real-time pilot self-briefing system.

The work reported here was completed under the direction of the TSC program manager, Manuel F. Medeiros, and the technical monitors, John J. Sigona and Vito P. Maglione. Carey Weigel of the FAA provided overall program guidance.

METRIC CONVERSION FACTORS

Approximate Conversions to Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
in	inches	2.5	centimeters	cm
ft	feet	30	centimeters	cm
yd	yards	0.9	meters	m
mi	miles	1.6	kilometers	km
		AREA		
in^2	square inches	6.5	square centimeters	cm^2
ft^2	square feet	0.09	square meters	m^2
yd^2	square yards	0.8	square meters	m^2
mi^2	square miles	2.6	square kilometers	km^2
	acres	0.4	hectares	ha
		MASS (weight)		
oz	ounces	28	grams	g
lb	pounds	0.45	kilograms	kg
	short tons (2000 lb)	0.9	tonnes	t
		VOLUME		
tsp	teaspoons	5	milliliters	ml
Tbsp	tablespoons	15	milliliters	ml
fl oz	fluid ounces	30	milliliters	ml
c	cups	0.24	liters	l
pt	pints	0.47	liters	l
qt	quarts	0.95	liters	l
gal	gallons	3.8	liters	l
ft^3	cubic feet	0.03	cubic meters	m^3
yd^3	cubic yards	0.76	cubic meters	m^3
		TEMPERATURE (exact)		
°F	Fahrenheit temperature	5/9 (after subtracting 32)	Celsius temperature	°C

Approximate Conversions from Metric Measures

Symbol	When You Know	Multiply by	To Find	Symbol
		LENGTH		
mm	millimeters	0.04	inches	in
cm	centimeters	0.4	inches	in
m	meters	3.3	feet	ft
m	meters	1.1	yards	yd
km	kilometers	0.6	miles	mi
		AREA		
cm^2	square centimeters	0.16	square inches	in^2
m^2	square meters	1.2	square yards	yd^2
km^2	square kilometers	0.4	square miles	mi^2
ha	hectares (10,000 m^2)	2.5	acres	
		MASS (weight)		
g	grams	0.035	ounces	oz
kg	kilograms	2.2	pounds	lb
t	tonnes (1000 kg)	1.1	short tons	
		VOLUME		
ml	milliliters	0.03	fluid ounces	fl oz
l	liters	2.1	pints	pt
l	liters	1.06	quarts	qt
l	liters	0.26	gallons	gal
m^3	cubic meters	35	cubic feet	ft^3
m^3	cubic meters	1.3	cubic yards	yd^3
		TEMPERATURE (exact)		
°C	Celsius temperature	9/5 (then add 32)	Fahrenheit temperature	°F

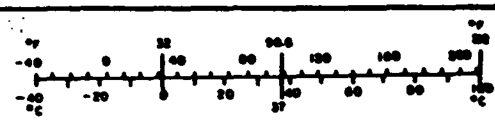


TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF ILLUSTRATIONS

LIST OF TABLES

1. INTRODUCTION

The Direct User Access (DUA) system is presently being developed as a component of the FAA Flight Service Station Automation Program. The system will enable pilots to interact with a computer system to obtain weather briefings and file flight plans. Transactions will be made over CRT and hardcopy terminals for graphical and textual output, and over Touch-Tone® telephones for spoken briefings. The spoken material is the output of the 20-channel Voice Response System (VRS) developed at the Transportation Systems Center (TSC) in Cambridge, Massachusetts. To date, the VRS gives (speaks) three weather products over the telephone with stored words: Hourly Surface Observations (SA), Terminal Forecasts (FT), and Forecast Winds Aloft (GF) [Air Transport Association (ATA) Grid Winds -- prepared by the National Meteorological Center for the airlines]. Using a special Touch-Tone protocol, the pilot enters the three-character location identifier for each airport or weather station of interest. The VRS prompts the pilot to indicate which weather products are needed, and, if necessary, to enter specific altitudes and time for Winds Aloft data.

1.1 VRS FUNCTIONAL OVERVIEW

A Digital Equipment Corporation (DEC) PDP-11/34® computer issues the prompts and receives the user's requests, sending the requests to a second computer, a DEC PDP-11/70® which has access to the National Weather Service files in Kansas City, Missouri. The 11/70 weather processors are constantly translating incoming weather products into sets of pointers which reference the VRS dictionary of recorded words and phrases.

When the 11/70 weather report retrieval program receives a request, the pointers corresponding to the required weather report are located and sent back to the 11/34. The specified locations in the dictionary file are read and the data sent to an output subsystem (the Adaptive Differential Pulse Code Modulation (ADPCM) decoder) which decodes the digital data and converts it to analog signals (stored records) that the user can hear over the telephone.

1.1.1 PDP-11/34® Functions

The VRS computer (i.e., the PDP 11/34) performs all "terminal" functions. These functions include: accepting input from the user via Touch-Tone® phone, transmitting this input to the 11/70 and providing voice output of information sent back from the 11/70. The basic software flow diagram is presented in Figure 1-1. A brief discussion on each block function is presented as follows in the sequence that the computer processes the information.

The user input enters the software through the Touch-Tone driver. The driver provides device-dependent function handling, such as phone answering and producing ASCII characters from the Touch-Tone input. The driver also separates the input from all channels into separate storage areas.

The separate storage areas are then examined by the dialogue program. This module collects all information needed by the 11/70 to perform data retrieval. The information collected includes location identifiers, altitudes and weather types.

At this point, the program prompts (speaks to) the user to input the data required. The program has a collection of responses that it "speaks" to the user. These responses are retrieved and spoken to the user by using the disk driver, the disk driver completion routine, the ADPCM driver, and the ADPCM completion routine. The disk driver reads a portion of the message to be spoken and executes

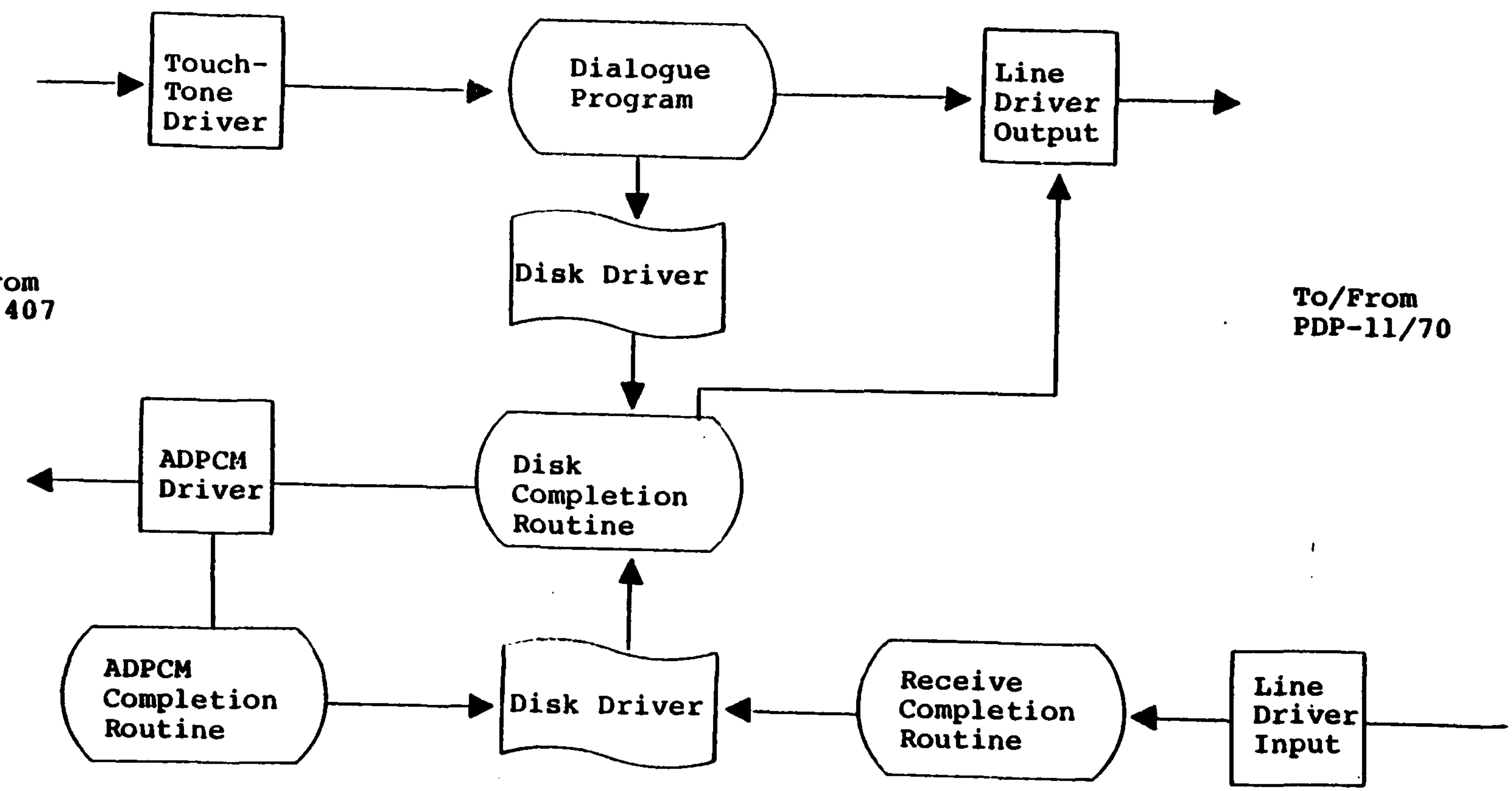


Figure 1-1: PDP-11/34* VRS Software

the disk completion routine. The disk completion routine sends the message fragment to the ADPCM driver. The ADPCM driver speaks the message fragment and executes the ADPCM completion routine which requests another disk read from the disk driver. This process of disk driver, to disk read completion, to ADPCM handler, to ADPCM completion, continues until the entire message is spoken. After completing the spoken message the ADPCM completion routine returns control to the dialogue program.

The information collected by the dialogue program is formatted and transmitted to the PDP-11/70® by the line driver output. This driver performs the functions required by the line protocol. This includes insertion of all protocol characters, and data retransmissions required by invalid user entries or line interference.

The 11/70 prepares the requested data for transmission. The data arrives at the line driver input in "message units" (defined in Section 2.4.3.4). The message units must be specifically requested by the VRS computer before they are sent. A request for the next message unit is sent by the ADPCM completion routine when it has completed the speaking of the previous one.

1.1.2 PDP-11/70 Functions

The PDP-11/70 maintains all of the weather data which are required to be vocalized by the VRS computer. The PDP-11/70 will eventually contain the software required to process eleven different weather report types. It currently contains three weather processors: Surface Observations (SA), Terminal Forecasts (FT), and Forecast Winds Aloft. The processing procedure consists of three operations: accessing a dynamic data base of weather information to recover raw weather data; translating the raw weather data into a format which is recognized by the VRS 11/34 computer; and storing the translated information in data files that are organized to process is one of mapping ASCII* weather report words and phrases into their corresponding dictionary file addresses of the locations where the actual digitized utterances are located.

The translation requires a dictionary (sort for indicating) where each word and phrase are located in the vocabulary file. Two copies of the dictionary exist, one on the 11/34 fixed head disk where the vocabulary file itself resides, and the other on the 11/70 disk where it is accessed by the weather processors. (When the dictionary is updated at the 11/34, it is sent to the 11/70 using an off-line utility, SENDIC.)

In addition to translating the raw data, validity checks are made and unrecognized words or formats are flagged as errors for manual editing. The method of handling unrecognized ASCII combinations is described in detail in Section 2.4.3.5.

The PDP-11/70® is required to retrieve weather information upon request by the VRS computer. Three modes of retrieval (selected by the pilot) have been defined as follows:

1. Local - Predefined data for particular locations are presented in the following order, if available: Area Forecasts e.g., (WA, WS, WW, WH) Notices to Airmen-NOTAMS (NO), Density Altitude, Surface Observations (SA), Pilot Reports (UA), Terminal Forecasts (FT), Forecast Winds Aloft, and Weather Synopsis (SY).

2. Selected Weather - The weather reports: SA, FT, UA, NO, SY, and Winds Aloft (time, altitude) are retrieved for each location specified.

*American Standard Code for Information Interchange (ASCII)

3. Prompt - The user is asked a series of questions requiring yes/no answers concerning the report he wants for the specific locations. The prompt mode is currently the mode in operation for the 20-channel system.

The PDP-11/70® uses a Location Index Table (LIT) in a Universal Data File (UDF) to locate the disk block numbers of the translated weather reports requested by the user. A briefing table of these block numbers is constructed and used for reading the blocks containing disk pointers that indicate the stored utterances as transmitted to the 11/34. The disk pointers are grouped into logical divisions called message units (see Section 2.4.3.4). The 11/34 begins requesting successive message units when it is ready to speak, and the 11/70, following its briefing table, reads the blocks into a buffer and sends the data message a unit one at a time to the 11/34. The 11/70 software configuration is shown on Figure 1-2.

1.1.3 Global Functions

The division of work between the two systems implies a number of functions are handled by both. These functions are system initialization, error handling, and communications.

1.1.3.1 Initialization - Initialization of the VRS involves two distinct operations, program startup and establishing communications. The exact implementation of operations may be different in the two computers, but the function is the same.

Program startup is internal to the two systems. The proper programs must be brought into core memory and all run time data bases, such as I/O buffers, must be initialized. Establishing communications consists of the 11/34 logging onto the 11/70, as a human would, and issuing an RSX-11D monitor command to load and execute the retrieval program (RETREV). Continued execution of

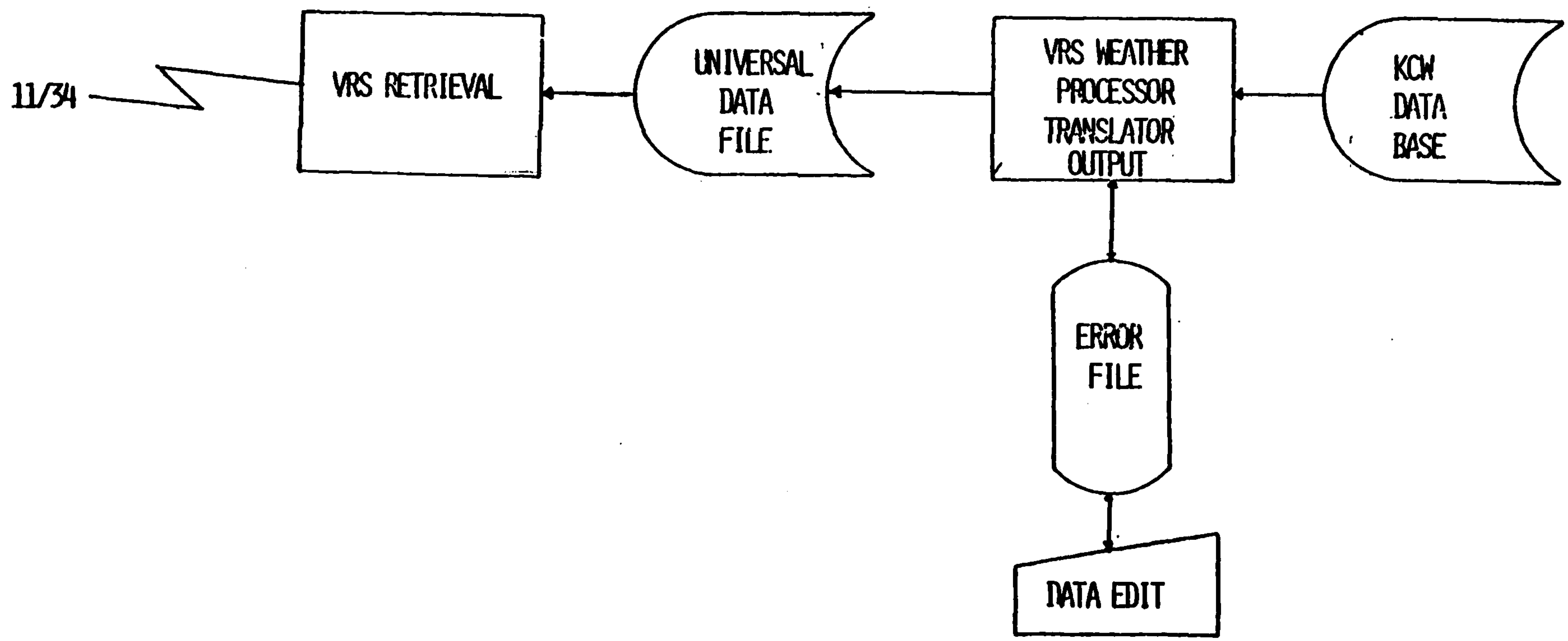


Figure 1-2: PDP-11/70® VRS Software

RETREV is thereafter verified by polling. If the 11/70 does not respond to the polls, the 11/34 software prints an error message and aborts.

1.1.3.2. Error Handling - Errors may occur in the actual operation of the program. A reporting function must exist to permit tracing sources of error to improve operation.

Errors fall into two major categories. The first areas are those which totally incapacitate the VRS. The second are those which permit the system to continue operation, but in a degraded manner.

The first category includes the following principal areas:

1) Disablement of the VRS computer. Hardware failure to prevent the VRS computer from performing its VRS functions. This type of error is determined using device status registers, and bus timeouts induced by accessing totally disabled I/O registers.

2) Line Failure. Both the 11/70 and the VRS computer are prevented from communicating as a result of serial line failure. The total failure of either machine will appear to the other as a line failure. Failures are determined by timeouts on the communication line.

The second category of errors includes:

1. Raw Weather Data Errors. Format problems of the raw weather data due to spelling errors, or other format problems result in these errors being sent to the Data Editor (see Section 2.4.3.5).

2. Garbled Transmission. Messages sent on the Communications line will occasionally suffer from noise and line outages. This

includes only occasional distortion of messages, not total line failure which was discussed previously.

3. I/O Errors. On occasion, peripheral devices will fail on an attempted I/O transfer. This type of error is rare with current technology but should be accounted for on the few occasions when they do occur.

Other errors such as software failures can also occur. The above list can be expanded as implementation proceeds, but is adequate to define the error problem.

1.1.3.3. Communications - The communications task provides the link between the systems. It must format data in a manner suitable for serial transmissions, and must receive the data, checking it for integrity and acknowledging receipt.

The line is bi-directional and the messages are of 4 types. The first is a briefing request. This message is transmitted from the 11/34 to the 11/70. It contains data used by the 11/70 to access the processed weather files. The 11/70 responds with either a positive acknowledgment, or a diagnostic message indicating such things as improperly spelled data, etc. If the request is accepted, 11/70 then internally prepares the data corresponding to the retrieval request. Communications integrity is checked by check-sum logic via the 11/34 and the Retrieval (11/70) program. This is explained further in Chapter 2.

1.2 PDP-11/34® HARDWARE

The various components of the 11/34 system (see Figure 1-3) are as follows:

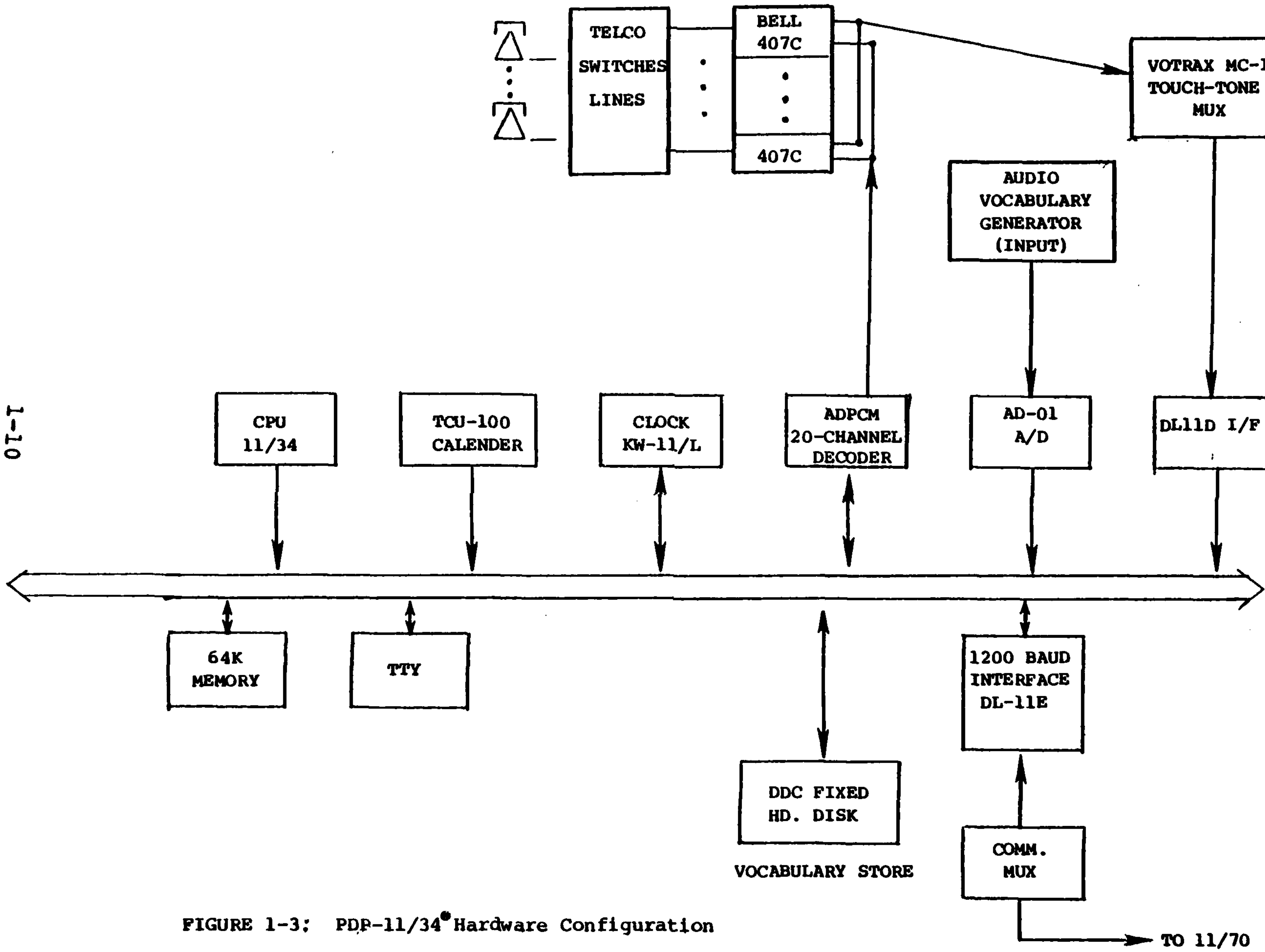


FIGURE 1-3: PDP-11/34* Hardware Configuration

*Required for vocabulary development

- CPU - PDP-11/34A processor.
- Memory - 64K word parity core memory for program execution.
- TTY - System master console (CDI Teleterm 1030) for running the VRS system and for software development.
- Calendar - TCU-100 Hardware clock calendar unit used by the VRS to obtain the current date and time of day.
- Clock - KW-11/L real-time clock required by the operating system to perform timing functions such as timing user response time.
- Magtape - TU-10 Mag tape drive. Required for regular back-up. Used to copy programs and vocabulary.
- Telephone Company (TELCO) Switched Lines - provides access to VRS using telephones.
- Bell 407C Data Sets - Converts the Touch-Tones® into signals the equipment can handle incorporated in the Bell.
- Touch-Tone Mux - VOTRAX MC-I decodes and multiplexes the Touch-Tone input from the twenty 407C units.
- DLII-E - Asynchronous interface to the 11/34 unibus for the VOTRAX unit.
- 20 Channel ADPCM Decoder - a specially designed interface for decoding the ADPCM code words into PCM samples and then into analog signals.

*More details can be found in the references. See (1) for Digital Equipment Corporation peripherals, Reference 2 for special purpose hardware. See also (3) and (4) for the Bell Equipment.

- Audio Vocabulary Generator and A/D - audio hardware for inputting the vocabulary (typically a tape recorder or microphone).

- Fixed-Head Disk - Digital Development Corporation (DDC-9112-D-8) fixed-head disk. The disk is used for storage of VRS software, program library, operating system, and the VRS vocabulary. Capacity of 4 million 16-bit words, 1800 RPM, 17 ms access time.

- DL-11E - 1200 bps Asynchronous Interface.

- Communications Multiplexor - A Computer Transmission Corporation Model 1315 communications multiplexor for communicating with the PDP-11/70® computer.

1.3 PDP-11/70 HARDWARE

The PDP-11/70 hardware consists of 768K bytes of memory with memory management and a dual 88 mega-byte disk storage system. The PDP-11/70 communicates with the VRS computer via a single channel in the multi-channel DH-11 interface.

The PDP-11/70 system is controlled by RSX-11D/V6B, which is an event driven, multiprogramming operating system offering up to 250 priority levels for task execution, multiple activity monitoring, priority interrupt servicing, task scheduling, dynamic memory partitioning, event flags for task notification and synchronization, support of multiuser programs, etc., as well as on-line software development, concurrent with task execution. A diagram of the 11/70 configuration is shown in Figure 1-4.

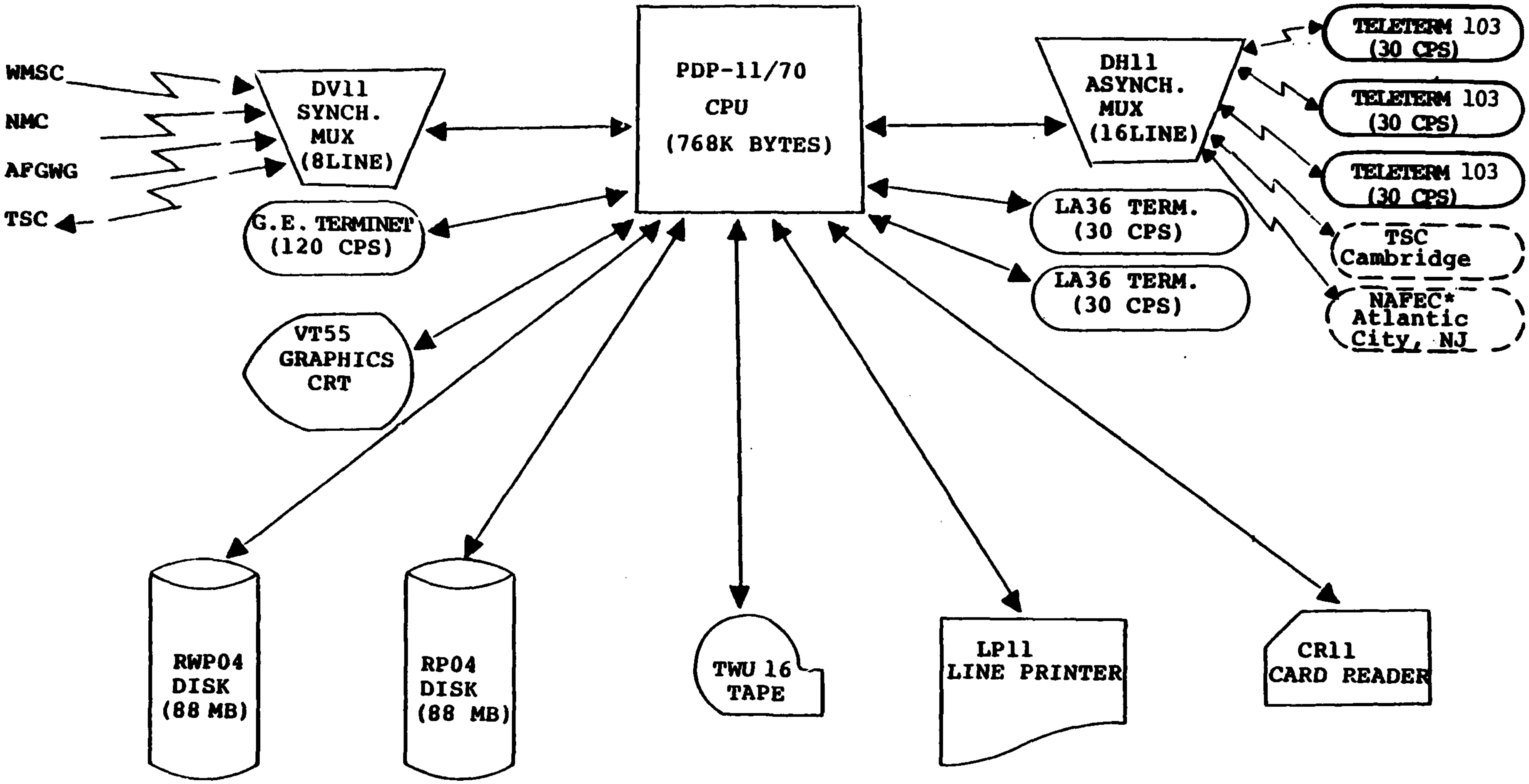


FIGURE 1-4: PDP-11/70* Hardware Configuration

*NAFEC - National Aviation Facilities Experimental Center

2. VRS SOFTWARE DESIGN

2.1 VRS COMMUNICATIONS

The nature and formats of the data transmitted between the two VRS computers are described in this section. The topic of communications line protocol and the associated protocol characters is addressed in Appendix B.

2.1.1 Establishing Communications

When the 11/34 operator enters the RT-11 monitor command, 'R VRS,' to begin execution, one of the initialization procedures the 11/34 VRS software performs is logging onto a certain 11/70 disk area to initiate execution of the weather report retrieval program, RETREV. The 11/34 sends the characters necessary for an ordinary RSX-11D log on:

```
HEL  [300,100]
   (current password)
RUN RETREV.
```

The log-on characters are echoed back to the 11/34 which types them on the terminal as reassurance to the operator that the log-on is happening as it should. (After this, no further transmissions to the 11/70 are echoed.) If the log-on and all other initialization procedures (discussed in subsequent sections) are successfully completed, a message to that effect is typed on the terminal. If the message does not appear, communication with the 11/70 has very likely not been established and the operator would take off-line remedial action. When communication has been successfully established, however, the 11/34 undertakes to monitor it by sending a special polling message, NULL ESC, every seven seconds to RETREV, which must respond with '*1' (ASCII asterisk one) within 20 seconds or the 11/34 assumes that either RETREV, the 11/70, or the communication line has failed. Without RETREV, the 11/34 can access no weather data, so it informs the operator of the trouble and aborts itself.

2.1.2 PDP-11/34® to PDP-11/70® Transmissions

The 11/34 computer transmits two types of messages to the 11/70: briefing compilation requests (type 1) and demand response requests (type 2). Type 1 messages are further defined into two sub-types. One sub-type is briefing request message #1 (BRM1). The other sub-type is briefing request message #2 (BRM2).

The briefing compilation request messages consist of ASCII character strings (terminated by a carriage-return character) which supply the parameters that the PDP-11/70 employs to retrieve weather data. The parametric information required by the PDP-11/70 consists of such items as briefing mode, location identifiers, report types, hours, and altitude.

The demand response requests consist of ASCII character strings (terminated by a carriage-return character) which require either a transfer of verbalization data from the PDP-11/70 to the VRS computer or informs the PDP-11/70 of some special condition of the briefing (shut-down, hangup, etc.)

2.1.2.1 Type 1 VRS Computer to PDP-11/70 Transmission - There are two sub-types of the type 1 transmission. They are identified as briefing request message #1 (BRM1) and briefing request message #2 (BRM2).

BRM1 is used to inform the PDP-11/70® of three briefing parameters: channel, briefing mode, and location identifiers.

BRM2 is used to inform the PDP-11/70 of four briefing parameters: channel, report types, time (hours from current time), and altitude.

An entire series of BRM2 transmissions may logically be issued for a single BRM1 transmission and thus effectively cause a briefing session to be a series of sub-briefings for the locations indicated in the BRM1 transmission. This permits the user to be actively involved in the progressions of the briefing in order that he may make subsequent requests based upon previous weather information.

The general form of BRM1 is shown below. The two fields are generalized as F1 and F2.

BRM1: XF1-F2[CKS][CR]

X: Channel Number: ASCII 0-19

F1: Mode: LM, SM, PM, (for local, selected, or prompt)

F2: Location identifier string

CKS: A three-character check-sum consisting of a two-character encoded sum of all transmitted characters followed by a character total of the number of transmitted characters.

Example: X F1 F2
8PM-BOS/ALB/BUF [CKS]

Field	Entry	Meaning
F1	Mode	Prompt Mode
F2	Locations	Boston, Albany, Buffalo

This briefing compilation request informs the PDP-11/70® that the user has requested a prompt mode briefing for Boston, Albany, and Buffalo. The VRS computer has assigned the user to channel 8.

The general form of BRM2 is shown below. The three fields are generalized as F1, F2, and F3.

BRM2: XF1-F2-F3[CKS][CR]

X: Channel Number: ASCII 0-19

F1: Report types

F2: Times (hours from current time)

F3: Altitude (in feet or feet x 100)

```
Example:     X   F1   F2   F3
                4 SA/FD-12-9700[CKS][CR]
```

Field	Entry	Meaning
F1	Report types	SA's, FD's (winds)
F2	Hours	Winds for 12 hours in advance
F3	Altitude	Winds for 9700 feet

This briefing compilation request informs the PDP-11/70® that the user on channel 4 has requested Hourly Surface Observations and Forecast Winds Aloft for the locations previously entered during a BRM1 transmission. The winds aloft are desired for 9700 feet and the twelve-hour forecast is requested.

2.1.2.2 Type 2 VRS Computer to PDP-11/70 Transmission - This transmission type is the method by which the VRS computer demands an immediate response from the PDP-11/70. The transmission is in ASCII-mode. There are three fields of information supplied, with an optional fourth field. The request is terminated with a carriage-return character.

The general form of a type 2 transmission is shown below. The left and right brackets are used to indicate that the enclosed information is optional. The brackets are for illustrative purposes, and are not transmitted.

Type 2: &XY[$N_1N_2N_3N_4$] [CKS] [CR]

Field 1: &, type 2 identifier

Field 2: X, X = channel number ASCII 0-19

Field 3: Y, Y = command code (A, B, C, D)

Field 4: $N_1N_2N_3N_4$, message unit number

The command codes (Field 3) represent the different types of responses the VRS computer expects.

When Field 3 is an A, the VRS computer is informing the PDP-11/70 that the briefing session is completed and that the channel is released (i.e. telephone hang-up or disconnect).

When Field 3 is a B, the VRS computer is requesting that the PDP-11/70® supply the message unit data and, in addition, echo the message unit number (See Section 2.1.3.2).

When Field 3 is a C, the VRS computer is requesting that the PDP-11/70 send the message unit number and message unit data of the first message unit of the next report type of the briefing. When Field 3 is a D, the VRS computer is requesting that the PDP-11/70 send the message unit number and message unit data for the first message unit of the report that contains the requested message unit (i.e., backup to the beginning of the current spoken report).

Field 3	Field 4 Required
A	Yes = 0
B	Yes =
C	Yes =
D	Yes =

2.1.3 PDP-11/70 to PDP-11/34® Transmissions

The PDP-11/70 answers the two types of VRS computer transmissions with two types of responses. A type 1 response is an ASCII-mode transmission which is used for two purposes: to indicate a completely acceptable briefing request; and to "echo" an invalid command string representing a request for a briefing. A type 2 response is a transparent-mode transmission which responds to a demand response request. This is the transmission which delivers the voice pointers and size data which the VRS computer uses to vocalize the weather information.

2.1.3.1 Type 1 PDP-11/70 to PDP-11/34 Transmission - The type 1 response to the VRS computer is an ASCII-mode message which is a response to a briefing request. The ASCII-mode message is used for diagnostics: one of which is a statement that the PDP-11/70 can comply with the transmitted request; the second of which is an echo of a briefing request with @'s substituted for the subfields which are acceptable. Type 1 responses are terminated with carriage-returns.

Type 1: Acceptable

X [CR] [CKS]

This transmission consists of the channel number (ASCII 0-19).

Type 1: BRM1 echo

X@-BOP/@/IAE [CR] [CKS]

This is a diagnostic response to a request on channel X (ASCII 0-19) in which the briefing mode was acceptable and the second location identifier was acceptable. Locations BOP and IAE were not located in the system data base.

Type 1: BRM2 echo

XFS/@-@-7 [CR] [CKS]

This is a diagnostic response to a request on channel X (ASCII 0-19), in which an invalid report-type was requested (FS), a valid report-type was requested, the time field is valid and the altitude field is invalid.

2.1.3.2 Type 2 PDP-11/70® to PDP-11/34® Transmission - A type 2 transmission to the VRS computer is used to honor a demand response request. This transmission is in binary transparent-mode and consists of the command echo, the channel, the message unit number, and the message unit data (if applicable). The general form of the transmission (characters in brackets are optionally transmitted) is:

Type 2: CE $N_1N_2N_3N_4[A_1A_2. \ . \ . \ . \ . \ . \ . A_n]$

where, C is an eight bit echo of the demand;

E is an eight bit channel number;

N_1 to N_4 is a 32 bit message unit number;

A_1 to A_n are the 8-bit bytes of the message unit.

With reference to Section 2.1.2.2, request codes B, C, and D require the message unit data and request code A requires a special message unit number zero, which is a confirmatory signal to the PDP-11/34® that the PDP-11/70® is closing all activity on the specified channel. If any command other than A contains a response of message unit zero, a message unit has been requested which is beyond the range of the briefing.

2.2 PDP-11/34 RESIDENT SOFTWARE

Section 1.1 provides a brief introduction to the functions provided by the 11/34 VRS computer. The software to perform these functions is discussed here.

The RT-11 Version 3 Extended Memory monitor is used as the operating system for the VRS computer. The various components of the VRS system are depicted in Figure 2-1. The function of each of the components of the system will be given later. Here we will discuss the different priority levels of the components.

The driver components operate at three priority levels. Read or write I/O commands are initiated at priority zero, the lowest

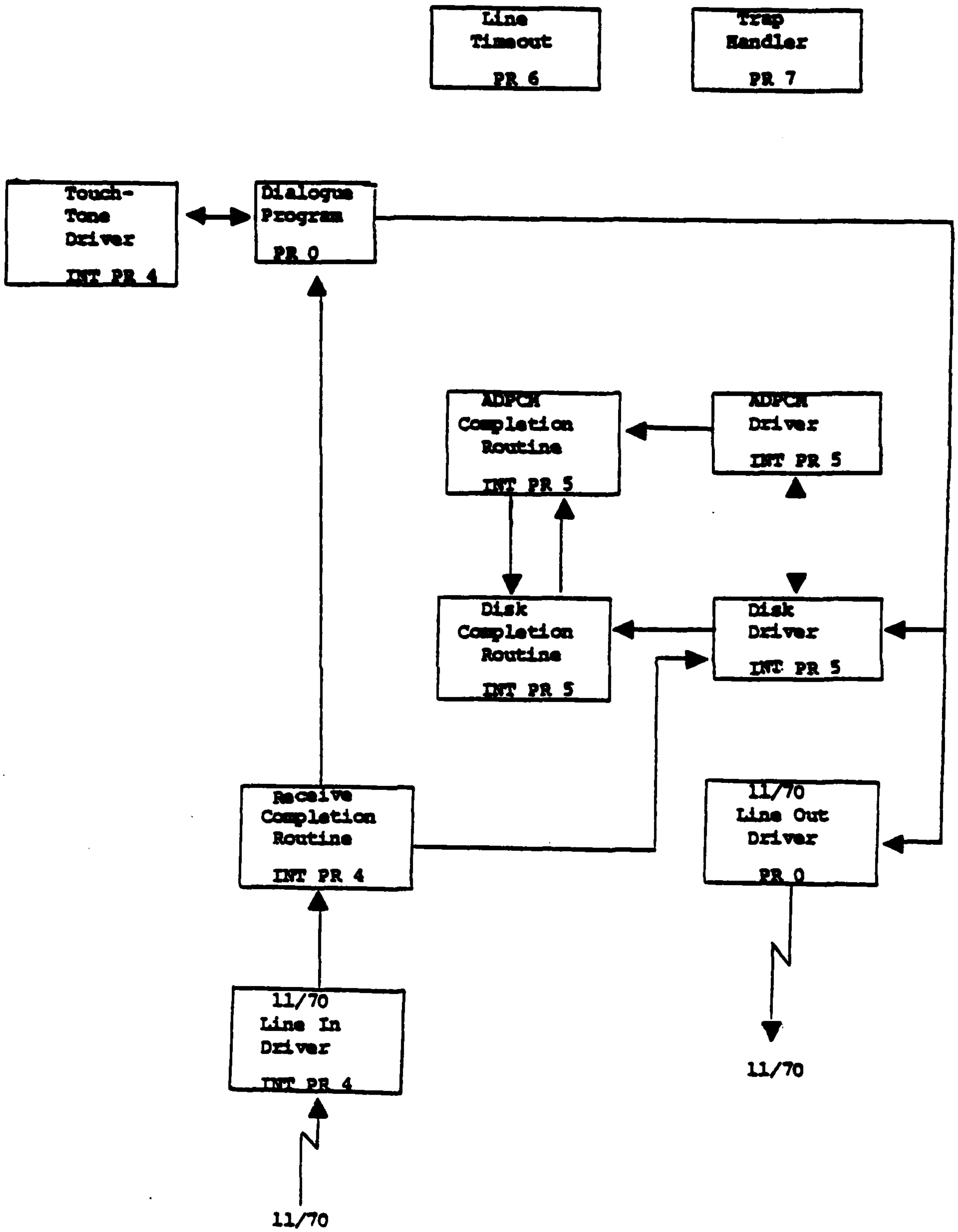


FIGURE 2-1: VRS System Components

processor priority. Data characters sent or received by the drivers are processed at priorities four or five. This guarantees instant response to data interrupts. The disk and ADPCM completion routines operate at interrupt priority five. The receive completion routine operates at priority four. The dialogue program operates at priority zero. The trap handler, the component synchronizer, operates at priority seven, the highest process level. The line timeout component, which monitors the activity of all lesser components, operates at priority six.

The 11/34 software is examined under the following section headings:

- Data Bases
- Device Drivers
- Dialogue Program
- Completion Routines
- Line Time-Out
- Trap Handler

2.2.1 Data Bases

The VRS computer maintains four data bases.

These data bases are:

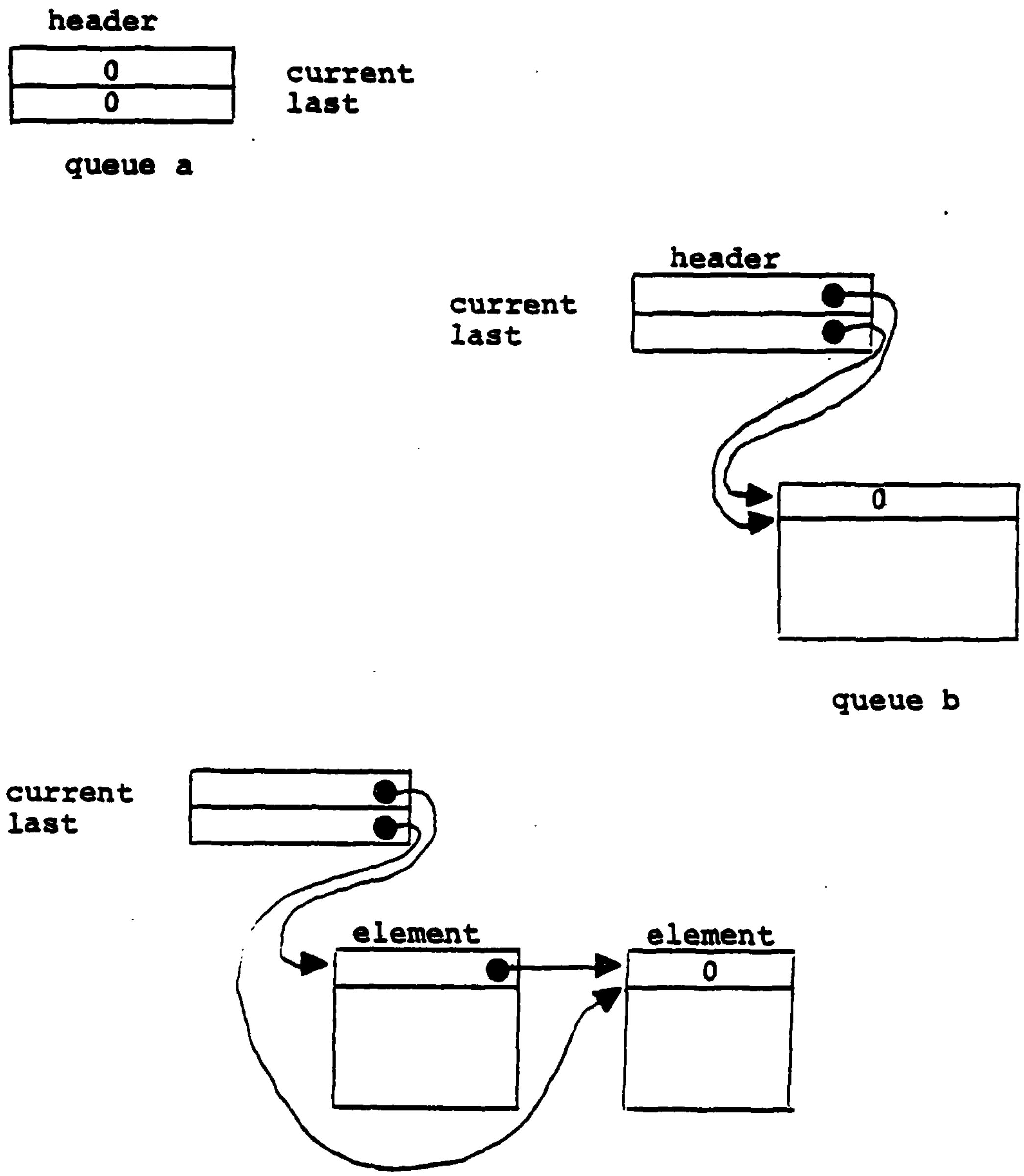
- Queues
- Buffers
- User Status Blocks
- Dialogue Protocol Index.

2.2.1.1 Queues - Queues are linked lists consisting of a queue header and a chain of any number of queue elements. The queue header is a two-word field that determines the limits of the chain of queue elements. The first word points to the first queue element and the second word points to the last queue element. If there are no queue elements in the queue, both words are set to a zero value. Figure 2-2 shows three examples of queued lists.

All queue elements linked to a specific queue header are members of that particular queue. Each queue element of a particular queue is a consecutive block of memory whose first word is a link pointer to the next element of the queue. If the queue element is the last element of the queue, the link pointer value is zero. The values contained in the remainder of the consecutive block of memory depend on the queue function.

Figure 2-3 shows an I/O queue element used by the RT-11 system to queue I/O orders. The link word's function is described in the previous paragraph. Word 1 contains the VRS channel number and the I/O function code. Word 2 is used by the RT-11 operating system. Word 3 is the block address for random access devices. Word 4 contains the input or output buffer address. Word 5 is the word count that determines the number of words to transfer. Word 6 is the completion code which determines the action to take upon initiating or completing the I/O.

The VRS contains three different types of queued elements: the I/O queue elements, disk read queue elements and 11/70 receive queue elements. The I/O queue elements were explained in the previous paragraph. The disk read queue elements are elements whose consecutive block of memory contains a link field, followed by a five word I/O parameter list, followed by a 1024 word input/output buffer. The element is used to read disk voice data and write the data to the ADPCM driver. The receive queue elements contain a link field followed by a 64-word data buffer used to send or receive data to or from the 11/70.



For a queue of length:

a) 0 elements
b) 1 element
c) 2 elements

FIGURE 2-2: Three Queue Examples

Word No.		
0	Link Word	
1	15 8 I/O Code	7 Ø VRS Channel
2	RT-11	
3	Block Address	
4	Buffer Address	
5	Word Count	
6	Completion Code	

FIGURE 2-3: I/O Queue Element

2.2.1.2 Buffers - The VRS software uses three types of buffers. The first is a 40-word Touch-Tone® input buffer permanently assigned to each of the VRS channels. All translated Touch-Tone input is placed into this buffer. The buffer is also used to transmit briefing requests to the 11/70. The second is a 1024-word buffer used for reading disk voice data and speaking the data using the ADPCM driver. The third is a 64-word buffer used to receive input from the 11/70 and to echo Touch-Tone input.

2.2.1.3 User Status Block - A user status block (USB) is assigned to each VRS channel. The USB is a separate data base enabling asynchronous operation of all VRS channels. Figure 2-4 defines the fields of the USB. The following describes each field of the USB:

- Bytes 0,1 contain the beginning address of the permanently assigned 40 word buffer.

- Bytes 2,3 contain the byte location within the 40 word buffer that will receive the next translated Touch-Tone input character.

- Bytes 4,5 contain the byte location within the 40 word buffer of the start of the last input field, i.e., beginning of last location identifier or weather report type, etc.

- Byte 6 contains the first character of a Touch-Tone input keystroke pair.

- Byte 7 contains the current position within the dialogue.

- Bytes 10,11 contain the identifier of the last component of the system that serviced the line.

Byte Number Octal		
0	BEGINNING OF BUFFER	
2	CURRENT INPUT LOCATION	
4	BEGINNING OF LAST INPUT	
7 6	DIALOGUE POINTER	FIRST KEYSTROKE
10	LINE STATUS	
12	COMPLETION MASK	
15 14	FLAG BITS	EVENT VECTOR
16	PERMANENT FLAG BITS	
20	REPORT TYPES	
22	MESSAGE POINTER	
24	LAST BLOCK COUNT	
26	NUMBER OF BLOCKS	
30	DISK BLOCK NUMBER	
32	TALK QUEUE HEADER	
34	TALK QUEUE TAIL	
36	RETURN ADDRESS	

Byte Number Octal		
40	READ QUEUE HEADER	
42	READ QUEUE TAIL	
44	SAVE AREA #1	
46	SAVE AREA #2	
50	BRIEFING MODE	
52	RECEIVED MESSAGE UNIT	
54	DOUBLE PRECISION	
56	MESSAGE RECEIVED QUEUE	
60	MESSAGE RECEIVED TAIL	
62	MESSAGE UNIT REQUESTED	
64	DOUBLE PRECISION	
66	SPEAK QUEUE HEAD	
70	SPEAK QUEUE TAIL	
72	MESSAGE UNIT SPEAKING	
74	DOUBLE PRECISION	
77 76	CHANNEL ASCII CODE	CHANNEL BINARY CODE

FIGURE 2-4: User Status Block

- Bytes 12,13 are the completion mask, which is a unique bit for each VRS channel. The bit is used to distinguish which particular VRS channel is signalling a significant event.

- Byte 14 contains an event vector to distinguish the particular event being signalled by the completion mask.

- Byte 15 contains the flag bits that signal the functions to take place during this particular step of the dialogue protocol.

- Bytes 16,17 contain flag bits that govern the functions to take place during two or more steps of the dialogue protocol.

- Bytes 20,21 contain the flag bits that signal what report types are available.

- Bytes 22,23 are the pointer to the sequence of field pairs that define the message to be spoken.

- Bytes 24,25 contain the number of words in the last block of the voice data for the current utterance being spoken.

- Bytes 26,27 are the number of disk blocks that contain the utterance being spoken.

- Bytes 30,31 contain the disk block number of the utterance being spoken.

- Bytes 32,33 are the queue header and bytes 34, 35 are the tail pointer of the read queue elements queued for the ADPCM handler.

- Bytes 36,37 are the address of the instruction where processing will resume when the current message is spoken.

- Bytes 40,41 contain the header and bytes 42, 43 contain the tail for the read queue elements currently queued to the disk handler.

- Bytes 44 through 46 contain the return address pointers to the subroutines that are-to be returned to after a briefing request completes.

- Bytes 50,51 define the current briefing mode: selected, local, or prompt.

- Bytes 52 through 55 contain the ASCII number of the last briefing message unit received from the 11/70.

- Bytes 56 through 61 are the queue header of all receive queue elements of message units received from the 11/70.

- Bytes 62 through 65 contain the ASCII number of the last briefing message unit requested from the 11/70.

- Bytes 66,67 contain the queue header and bytes 70,71 are the tail of the message units queued to be spoken.

- Bytes 72 through 75 contain the ASCII number of the message unit that is currently being spoken.

- Byte 76 is the channel binary code.

- Byte 77 is the channel ASCII code.

2.2.1.4 Dialogue Protocol Index - A dialogue protocol index is used to prompt the user through one step of the protocol. The dialogue protocol index indicates what functions are to take place immediately before, during, and immediately after a single step of the user dialogue. Figure 2-5 shows the fields of a dialogue protocol index.

- Bytes 0,1 contain the flag bits placed into the user status block at the beginning of this step of the user dialogue.
- Bytes 2,3 are the address of the special function subroutine to be performed before speaking the prompt message.
- Byte 4 contains the number of seconds to wait before speaking the prompt message.
- Byte 5 contains the number of seconds to wait before echoing the user response.
- Bytes 6,7 define a message link to enable all dialogue protocol indices that speak the same prompt message to use the same stored canned message.
- Bytes 10,11 contain the address of the stored canned message unit.
- Bytes 12,13 define the address of the special function subroutine to be executed before performing the syntax analysis check.
- Bytes 14,15 define the syntax analysis check mask to verify the user input.
- Bytes 20,21 define the address of the special function subroutine to be performed before beginning the next dialogue protocol index.
- Byte 22 defines the next dialogue protocol index to execute if the user makes a normal or yes response.

Byte Number Octal			
	0	FLAG BITS	
	2	SPECIAL FUNCTION BEFORE SPEAKING	
3	4	ECHO WAIT	PROMPT WAIT
	6	MESSAGE LINK	
	10	PROMPT MESSAGE	
	12	SPECIAL FUNCTION BEFORE SYNTAX ANALYSIS	
	14	SYNTAX CHECK MASK	
	16	SPECIAL FUNCTION BEFORE ECHOING RESPONSE	
	20	SPECIAL FUNCTION BEFORE NEXT DIALOGUE	
23	22	NO or ABNORMAL BRANCH	YES or NORMAL BRANCH

NOTE: All fields are optional except the prompt message and the yes/no branch vector fields.

FIGURE 2-5: Dialogue Protocol Index

Byte 23 defines the next dialogue protocol index to execute if the user responds with an abnormal or no response.

2.2.2 Device Drivers

The VRS software performs all of its I/O using the programmed requests provided by RT-11. Hence, all reads and writes of information must obey the conventions of the operating system. Reference 9, the RT-11 Advanced Programmers Guide describes these programmed requests and shows how specialized handlers must work within the constraints of RT-11. The RT-11 Advanced Programmers Guide is recommended reading for full comprehension of the specialized handlers.

2.2.2.1 Touch-Tone® Driver (MCX) - The Touch-Tone driver is RT-11 compatible with the exception of its servicing of read requests. The driver services the input Touch-Tone keystrokes by decoding and inserting the decode character into the fixed 40-word VRS Touch-Tone input buffer for the designated channel. It decodes a pair of input keystrokes if alphanumeric input is expected, or a single keystroke if numeric input is indicated. The Touch-Tone driver services write requests to enable or disable a VRS channel. The driver notifies the dialogue program when any significant event occurs on a VRS channel by setting the user status block completion mask bit into a fixed memory location. Significant events reported are: telephone ringing, disconnect, input complete, invalid input, etc.

2.2.2.2 DL-11 Line Interface Driver - The DL-11 interface is controlled entirely by line-in and line-out software.

2.2.2.3 Fixed-Head Disk Driver (RFX) - The fixed-head disk driver is an RT-11 driver. Exact details of what this implies are described in Reference 6, Chapters 2, 4, and 5.

2.2.2.4 ADPCM Driver (ADX) - When VRS wants to speak a message to the user, it calls the ADPCM driver, which initiates speech on the proper channel. The ADPCM hardware does not require processor intervention while speaking a message because it is a direct memory access device. When the ADPCM hardware runs out of speech data, it calls the ADPCM interrupt routine which checks for errors. Then it starts the next speech message to the channel. If there are no speech messages, it turns off the ADPCM hardware on that channel. Finally, the ADPCM handler initiates the ADPCM completion routine with the channel number.

2.2.3 Dialogue Program

The dialogue program, operating at priority zero (the lowest machine priority) constantly checks the status of a significant event completion indicator located in a fixed memory word. The Touch-Tone® driver indicates a significant event by setting the user status block completion mask bit for the affected channel. The Touch-Tone driver also sets the particular significant event code. Figure 2-6 is a schematic flow of the priority zero VRS software. Table 1 presents the functions performed and their effects.

The dialogue program significant event recognition routine sequentially checks each of the VRS channels. This sequential check guarantees consecutive servicing of all VRS channels. Using the completion event code set by the Touch-Tone driver, the significant event recognition routine vectors to the proper servicing routine.

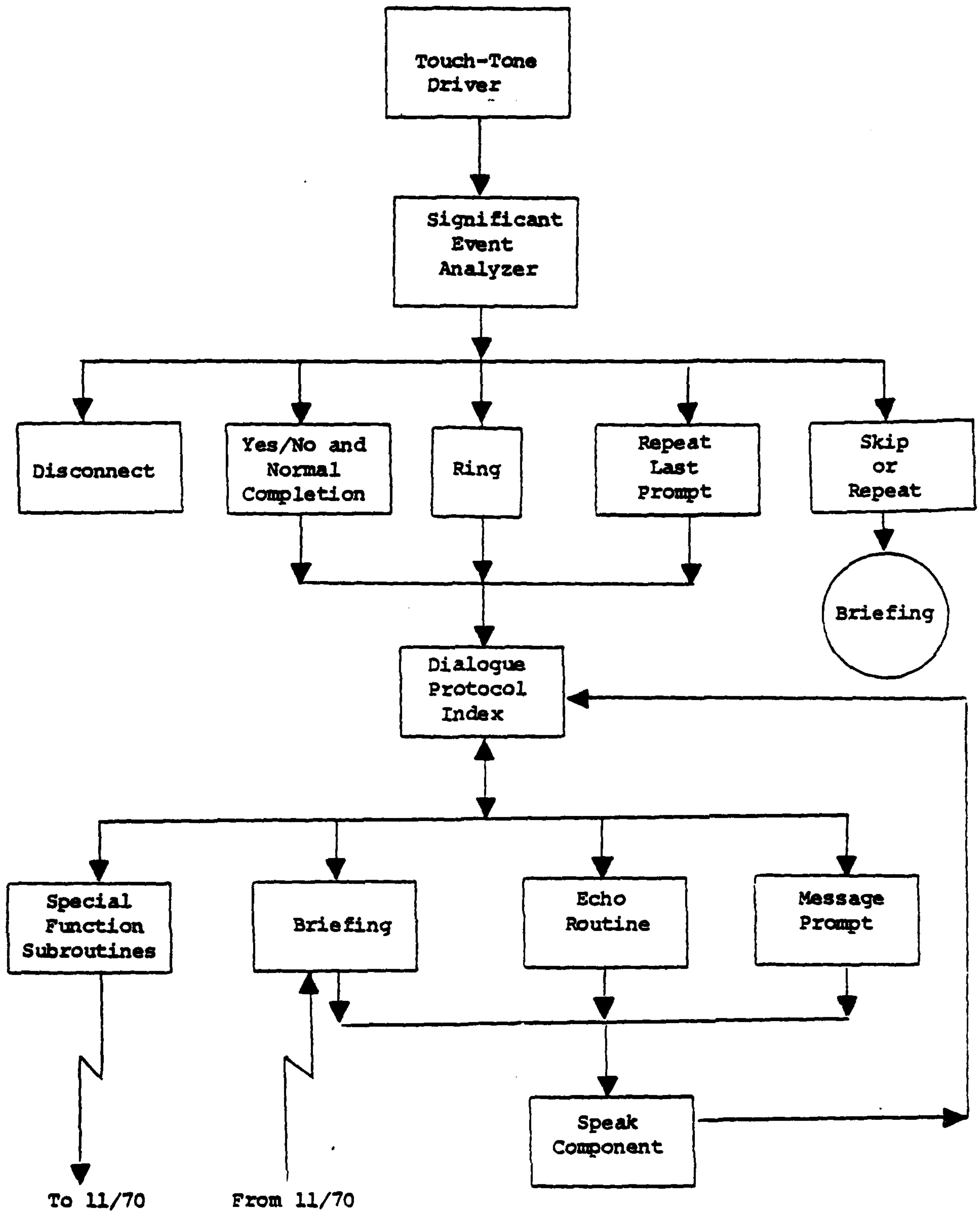


FIGURE 2-6: Dialogue Program

TABLE 1

BASE LEVEL FUNCTIONS PERFORMED

NAME	CAUSE	BRIEFING MODE EFFECT	PROMPT MODE EFFECT
INVALID KEYSTROKE	INVALID KEYSTROKE	NONE	SPEAK "INVALID ENTRY"
NORMAL COMPLETION	##	NONE	CONTINUE W/NEXT QUESTION
RECYCLE	*B	BEGIN WEATHER BRIEFING	SPEAK THE "HELLO" MESSAGE
SKIP	*J	SKIP TO NEXT WX REPORT	NONE
RING	RECEIVE RING CONDITION	NONE	SPEAK THE "HELLO" MESSAGE
DISCONNECT	1. RECEIVE HANGUP CONDITION 2. TRANSMISSION ERROR COUNT EXCEEDED	SOFTWARE EFFECTS A HANGUP	SOFTWARE EFFECTS A HANGUP
YES	Y##	NONE	VECTOR TO YES RESPONSE
NO	N##	NONE	VECTOR TO NO RESPONSE
RETURN	COMPLETION OF VRS FUNCTION	CONTINUE W/NEXT FUNCTION	CONTINUE W/NEXT FUNCTION
REPEAT	*R	REPEAT LAST WX REPORT	REPEAT LAST PROMPT
CANCEL	*D	NONE	CANCEL LAST ENTRY
GO	*G	CONTINUE SPEAKING	NONE
STOP	*	STOP SPEAKING	NONE
TIMEOUT	TIME ON SYSTEM GREATER THAN 15 MIN	SPEAK HANGUP MESSAGE AND EFFECT A DISCONNECT	SPEAK HANGUP MESSAGE AND EFFECT A DISCONNECT

The significant event service routines are:

- The telephone ringing service routine which activates the 11/70 retrieval program if no other VRS channels are active and initializes the user status block.

- The telephone disconnect service routine which notifies the 11/70 retrieval program that the briefing is complete for the given channel and if no other VRS channels are active, deactivates the 11/70 retrieval program.

- The yes/no and normal completion service routines set their unique status indicator into the status field of the user status block.

- The repeat last prompt service routine enables the repetition of the last message prompt.

- The skip or repeat service routine disables the current operation of the briefing component and requests from the 11/70 either the previous message unit for a repeat, or a skip to the next report.

All of the service routines, with the exception of the skip or repeat service routines, interface to the dialogue protocol index routine. The dialogue protocol index routine directs and conducts the operation on a VRS channel. Using the dialogue pointer contained in the USB, the dialogue protocol index routine executes one step of the protocol. The routine initiates the speaking of a message prompt to the user. The routine also directs the Touch-Tone® driver to decode the user responses as alphanumeric or numeric input. Finally, the routine performs a syntax analysis check on the user input, echoing a correct response if the dialogue protocol index indicates the user input is to be echoed. It executes the appropriate special service subroutines.

The special service subroutines perform services that are unique for a particular dialogue protocol index. Examples of some of the services performed are:

- o Formatting the Touch-Tone input to separate logical fields.
- o Changing briefing modes.
- o Clearing the Touch-Tone input buffer.
- o Recognition of last location identifier.
- o Skipping to another dialogue protocol index.
- o Formatting a specific weather report type.
- o Sending briefing requests to the 11/70.

The dialogue protocol index routine, using its special service subroutines, requests the user input location identifiers. The complete set of location identifiers is formatted and sent to the 11/70 retrieval program. The retrieval program validates each location identifier. If all location identifiers are valid, the 11/70 retrieval program sends back an acknowledgment to the 11/34 VRS software. If any location identifiers are invalid, the retrieval program sends back a diagnostic message which identifies which location identifiers were valid and which location identifiers were invalid. A special service subroutine within 11/34 VRS requests the user correct the invalid location identifiers by cancelling them or re-inputting another location identifier. The correct location identifiers are retransmitted to the 11/70.

Dependent upon the particular briefing mode, the dialogue protocol index routine may ask the user for additional input. For a local mode briefing, no other information is requested and the dialogue protocol index routine enters briefing mode. For a prompt briefing, the user is asked a series of questions requiring a yes or no response. For each yes response, a weather report type request is sent to the 11/70 retrieval program and the dialogue protocol index routine enters briefing mode. For a select mode briefing, the user is asked to input the weather report types. The input weather report types are sent to the 11/70, and the dialogue protocol index routine enters briefing mode.

The preceding material has explained the operation of the lowest priority routines of the VRS software. The operation services in a serial fashion each of the VRS channels that indicates a significant event. For a given VRS channel to perform the functions detailed above, there are a number of significant events. Each time a message is spoken to the user, requesting a user response, a significant event is required to cycle the user to the next step of the dialogue protocol. In general, the VRS completes instructions for a single VRS channel before it cycles back to check for a significant event on another VRS channel.

2.2.4 Completion Routines

The completion routines operate at an interrupt level priority zero. They are capable of interrupting the processing of the zero priority software. One of the completion routines is the receive completion routine which receives messages from the 11/70. If the received message is an acknowledgment from the 11/70 of a briefing request, the receive completion routine transfers control to the dialogue protocol index routine by setting a completion code and the completion mask in the same manner as the Touch-Tone® driver. Figure 2-7 demonstrates the logical flow of the completion routines.

If the received message from the 11/70 is a briefing message unit, the receive completion routine interfaces with the speech initiator. The speech initiator called by the receive completion routine or by the dialogue protocol index routine, initiates the verbal output by requesting a read of the appropriate voice data from the disk driver. The disk driver activates the disk completion routine when the disk read completes.

The disk completion routine requests the ADPCM driver speak the voice data. After speaking the voice data, the ADPCM driver executes the ADPCM completion routine. The ADPCM completion routine determines if the entire message prompt or the entire briefing has been spoken. If it determines that the entire speech has not been spoken, it requests another disk read of the next portion of the prompt message or briefing. If all of the current briefing verbalization has been spoken and it is not the end of the briefing, the ADPCM completion routine requests another briefing message unit from the 11/70.

To effect continuous speech, all read requests to the disk handler are buffered ahead so that the ADPCM driver always has the next portion of the verbal message to be spoken. The ADPCM driver automatically starts speaking the next portion upon completion of the last. When the entire message or briefing is complete, the ADPCM completion routine cycles back to the dialogue protocol index by setting a completion code and the completion mask, the same as the Touch-Tone driver and the receive completion routine.

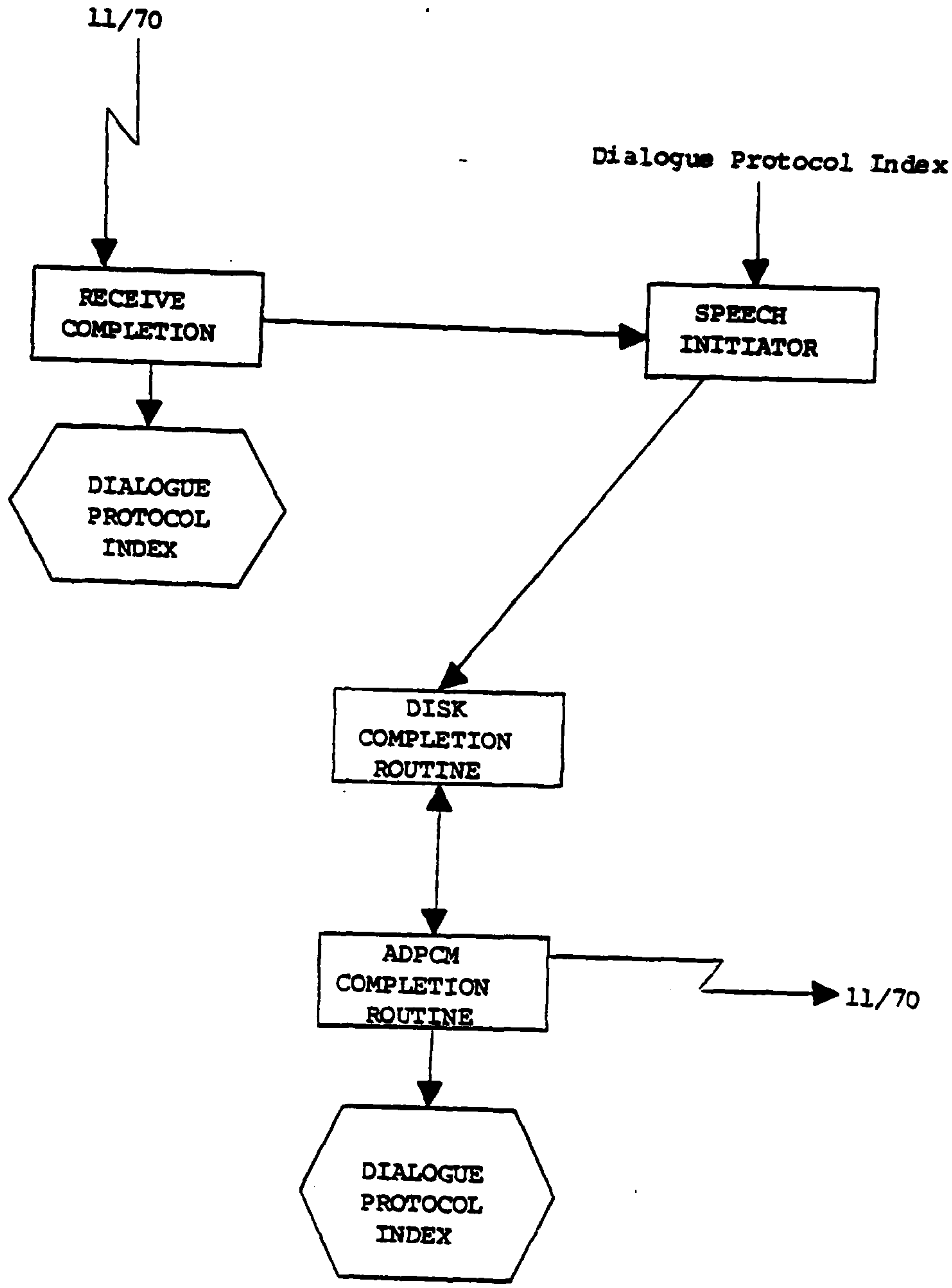


FIGURE 2-7 : Completion Routines

2.2.5 Line Timeout Routine

The line timeout routine performs two functions. First, it resends unanswered requests to the 11/70. If a communication error has occurred--either the 11/70 or the 11/34 has dropped a message--then line timeout will retransmit the request three times, at five-second intervals. If the data are not received, the user is disconnected.

The second function performed by line timeout is checking for pilot Touch-Tone® input. If no reply is made to a prompt by the 11/34 after fifteen minutes, then a disconnect message, "Your briefing has been terminated due to excessive time," is spoken and the line is disconnected.

2.2.6 Trap Handler

The trap handler operates at priority seven, the highest machine priority. The trap handler synchronizes operations among the various components of the operating system. An example is the adding or taking an element away from a queue header. Without the synchronizing feature of the trap handler, a component of the system operating at a certain priority could be taking the element from a given queue, be interrupted by a high priority routine that takes an element from the same queue. Without a synchronizing method, both components may well receive the same queue element. The trap handler routines are:

- Adding an element to a queue (queue)
- Taking an element from a queue (dequeue)
- Modifying the status field of the user status block
- Resolving an absolute user status block address

- Removing the significant event status bits from the fixed memory location.

2.3 STATISTICS PACKAGE OVERVIEW

In order to measure the use of the Voice Response System, the software on the PDP-11/34® maintains a data base describing each user's actions. A record is kept of when each user called, what reports were requested, which location identifiers were requested, if any special commands were requested, and when the caller hung up. The data base (VRDATA.DAT) is created by the VRS software each day and is a chronological file indicating all "significant events" for each call.

2.3.1 Statistics File Initialization

Each time the PDP-11/34 software is started, the statistics file (VRDATA.DAT) is initialized. There are three types of initialization:

1. Start with no statistics file - under the condition that the file VRDATA.DAT does not exist, the VRS software creates a file of 1,000 blocks in length. The file is zeroed such that all records are made blank.

2. Start with a complete file - under the condition that the system was taken down by the operator with an "EXIT" command, the file is defined to be complete. On normal EXIT of the system, pointers to the last data written in the file are written. When the system is started again, these pointers are used to define where to write subsequent data.

3. Start up after a system failure - under the conditions of a crash of the system, the pointers to the last data written in the file are not updated. On initialization, the software reads the file to the end and begins writing data at the end of the previous data.

2.3.2 Statistics File Structure

2.3.2.1 Overall File Structure - The statistics file is circular in nature and is 1,000 blocks long. The first block of the file is reserved as a pointer block. All other blocks in the file contain data. The pointer block depicted in Figure 2-8 shows the format of the pointer records.

As mentioned above, VRDATA.DAT is a circular file, that is, after the last physical block of the file is written, the software will begin writing over the existing oldest data in the file. The file has been constructed sufficiently large to accommodate 24 hours' worth of data for twenty users without wrapping. If the file should wrap, however, the pointers to the file are modified during initialization to reflect the new start and end of file.

2.3.2.2 Record Structure - The record definition appears in Figure 2-9. All values appearing in the text are octal. The first element is the record header containing a value of -16. The field data generated by each trace element is 16 bytes long. The second element is the length of the variable data record. It is equal to the number of bytes stored as data. The third element (US.CHN) is the channel being recorded. The low byte contains the binary value. The upper byte contains its ASCII equivalent (used in communications with the Retrieval Program). The fourth element (US.STA) contains the line status and as such defines the reason for the trace. The low byte of US.STA can take on the following values:

Word					
	0	2	4	6	10
	Date	Low Time	High Time	Block Start	Offset Start
	12	14	16	20	22
	Date	Low Time	High Time	Block End	Offset End

DATE ≡ 16 BIT INTEGER CONTAINING TODAY'S DATE (See Section 2.4.10 of RT-11 Advanced Programmer's Guide).

LOW TIME ≡ 16 BIT INTEGER CONTAINING LOW 16-BITS of the number of seconds since midnight.

HIGH TIME ≡ THE HIGH order number of seconds since midnight.

BLOCK START ≡ STARTING BLOCK of data in the file.(3 until file wraps).

OFFSET START ≡ How far into the block the data begins (usually Ø)

BLOCK END ≡ Last block of data in the field.

OFFSET END ≡ How far in the block the data are written.

FIGURE 2-8: Record Pointer Block

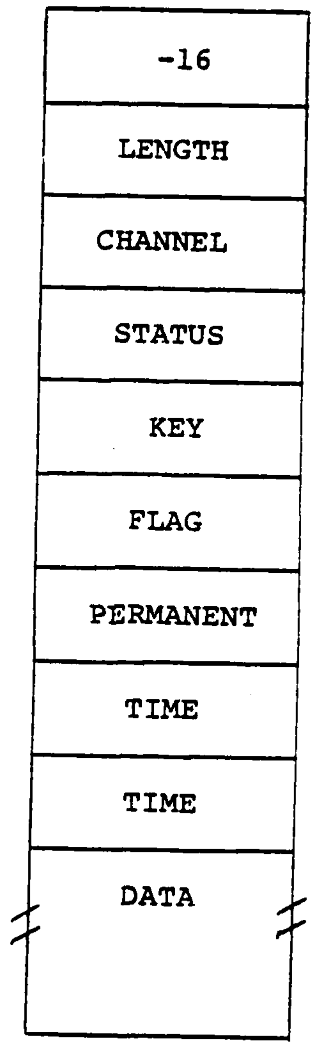


FIGURE 2-9: Record Definition

NAME	VALUE	EXPLANATION
RING	40	Channel is ringing
DISCON	41	Hang up in progress
STOP	42	Briefing stopped by user
GO	43	Briefing restarted by user
REPEAT	45	Briefing repeated by user
SKIP	46	Report skipped by user
ST.INV	47	Invalid entry by user
CANCEL	50	Cancel last entry
ST.SND	11	LOC-ID's Transmitted
ST.RNA	13	Receive from Washington not accounted for

The fifth element is the current value of the protocol, US. KEY. The high order byte of this record defines what the user is currently doing. The low order byte contains a value only if a control keystroke was the last character entered by the user.

The sixth element, US.FLG, contains temporary protocol bits describing what the user's current status is in the high byte, and a vector to the routine last executed at base level in the program in the low byte. Following is a list of low byte values of US.FLG.

NAME	VALUE	EXPLANATION
INVALK	0	User took abnormal (NO) response
NORMAL	1	User took normal (YES) response
RECYC	2	User typed "Begin Over"
SKIP	3	User requested a skip function
INVALK	4	User did not use valid Touch-Tone® entry
RING	5	Telephone is ringing
DISCON	6	Telephone has been disconnected
YES	7	User answered "Yes"
NO	10	User answered "No"
RETURN	11	Return from high level routine

BRIEFER	12	Leave briefing mode
REPEAT	13	Repeat question or report
CANCEL	14	Cancel last entry
GO	15	Proceed with briefing
STOP	16	Stop briefing

The high order byte contains the following status information:

Position	Name	ON	OFF
Bit 8	FL.ENP	User may not enter data	User may enter data
Bit 9	FL.NUM	User must enter numeric	May enter alpha-numeric
Bit 10	FL.DAP	Cyclic call	Non-cyclic call
Bit 11	FL.ECH	Response to be echoed	No echo of response
Bit 12	FL.PHE	Phonetic echo	Non-phonetic echo
Bit 13	FL.DIS	User may not enter data	User may enter data
Bit 14	FL.TKD	Speech is finished	Speech in progress
Bit 15	FL.ECD	Echo is finished	Echo in progress

The seventh element contains more status information (US.PER), and is depicted below:

Position	Name	ON	OFF
Bit 0	FL.TRA		Software maintenance
Bit 1	FL.YER	Yes response	No response
Bit 2	FL.DBL	Receive double buffered	Receive single buffered
Bit 3	FL.TRN	Hang up in progress	No hang up in progress

Bit 4	FL.BGN	Begin Protocol	Continue Protocol
Bit 5	FL.LST	Last LOC ID entered	Last LOC ID not entered
Bit 6	FL.BRF	Briefing Mode	Non-Briefing Mode
Bit 7	FL.BRD	Briefing finished	Briefing in progress
Bit 8	FL.FIR	First pass thru protocol	No first pass
Bit 9	FL.INT	Stop speech	Continue speaking
Bit 10	FL.SKP	Skip ahead in prog.	Not skipping data
Bit 11	FL.LOC	Entering LOC-ID's	Not entering LOC-ID's
Bit 12	FL.COR	Correcting LOC-ID's	Not correcting LOC-ID's
Bit 13	FL.SPC	Special Keystroke entered	Last character not special
Bit 14	FL.SPK	Speaking at base level	Not speaking at base level
Bit 15	FL.RTS	Skip or repeat	Neither skip or repeat

The eighth element contains the low order time since midnight in seconds. The ninth element contains the high order time since midnight.

The tenth and final element is the data buffer for the user. This buffer contains the message to be transmitted to the PDP-11/70® retrieval program. It is variable in length and its length is defined as the second element in the record. This element will contain the location identifiers requested by the user.

Bit 4	FL.BGN	Begin Protocol	Continue Protocol
Bit 5	FL.LST	Last LOC ID entered	Last LOC ID not entered
Bit 6	FL.BRF	Briefing Mode	Non-Briefing Mode
Bit 7	FL.BRD	Briefing finished	Briefing in progress
Bit 8	FL.FIR	First pass thru protocol	No first pass
Bit 9	FL.INT	Stop speech	Continue speaking
Bit 10	FL.SKP	Skip ahead in prog.	Not skipping data
Bit 11	FL.LOC	Entering LOC-ID's	Not entering LOC-ID's
Bit 12	FL.COR	Correcting LOC-ID's	Not correcting LOC-ID's
Bit 13	FL.SPC	Special Key-stroke entered	Last character not special
Bit 14	FL.SPK	Speaking at base level	Not speaking at base level
Bit 15	FL.RTS	Skip or repeat	Neither skip or repeat

The eighth element contains the low order time since midnight in seconds. The ninth element contains the high order time since midnight.

The tenth and final element is the data buffer for the user. This buffer contains the message to be transmitted to the PDP-11/70® retrieval program. It is variable in length and its length is defined as the second element in the record. This element will contain the location identifiers requested by the user.

2.4 RESIDENT PDP-11/70® SOFTWARE

The function of the resident software on the PDP-11/70 is to transmit the requested weather data to the VRS computer. The accomplishment of this process requires two separate and distinct phases of data handling. The first is the translation of weather data into VRS recognizable pointers. The second function is the selection and transmission of the proper data to the VRS computer.

The translation of the raw weather data into VRS pointers and the update and maintenance of those files is referred to as the "message processing" function. The selection of the VRS pointers and their subsequent transmission to the VRS computer is the "retrieval" function. The remainder of this chapter is devoted to description of these two functions.

2.4.1 Overview of PDP-11/70 VRS Message Processing

The data base to be accessed by the VRS system consists of data which have been processed from a raw data file, KCW.DAT. The processing procedure performs a translation of weather data which are received via transmission line from the Federal Aviation Administration's Weather Message Switching Center (WMSC), in Kansas City, Missouri. The translation procedure involves the following steps: acquisition of the proper sub-file to access the reports of a particular type; identification of the individual reports of that type and correlation to a location identifier (LOC.ID) or geographic region; separation (parsing) of the recognized words within the report, and use of a dictionary look-up technique to translate the ASCII words to binary representation. The binary information represents position and length parameters that are correlated to digitized words and phrases which are stored on the VRS computer disk files.

Figure 2-10 is a block diagram representation of the translation procedures (message processing).

2.4.2 Data Bases

The VRS 11/70 Software uses three data bases and a global common area (GCA). The data bases are KCW.DAT, UDF.DAT, and ERR.DAT. The global common area, called VRSGLB, is a shareable global task area linked to by the VRS processor tasks. VRSGLB contains input and output arrays for report processing and a map array for report block allocation (See Section 2.4.2.2.1). The following sections describe KCW.DAT, UDF.DAT, and VRSGLB; however, ERR.DAT is described later in Section 2.4.3.5.1.

2.4.2.1 Kansas City Weather Data Base - The weather data which are to be translated reside in a disk file, KCW.DAT at the PDP-11/70® system. The file consists of an index, followed by thirteen mutually exclusive ASCII sub-files, each of which is a circular buffer. The index maintains the current status of each sub-file, with respect to sub-file boundaries, last disk block written, last character written, and circular wrap-around indicator. Each sub-file represents a different weather type, except in the case of area forecasts and significant meteorological events which reside in the same sub-file (see Figure 2-11).

Each sub-file consists of headers and reports, stored by weather type. The headers and reports are stored in the sub-files in ASCII, exactly as received from the WMSC. The weather reporting formats of all the weather types are described in the National Weather Service's Operations Manual.

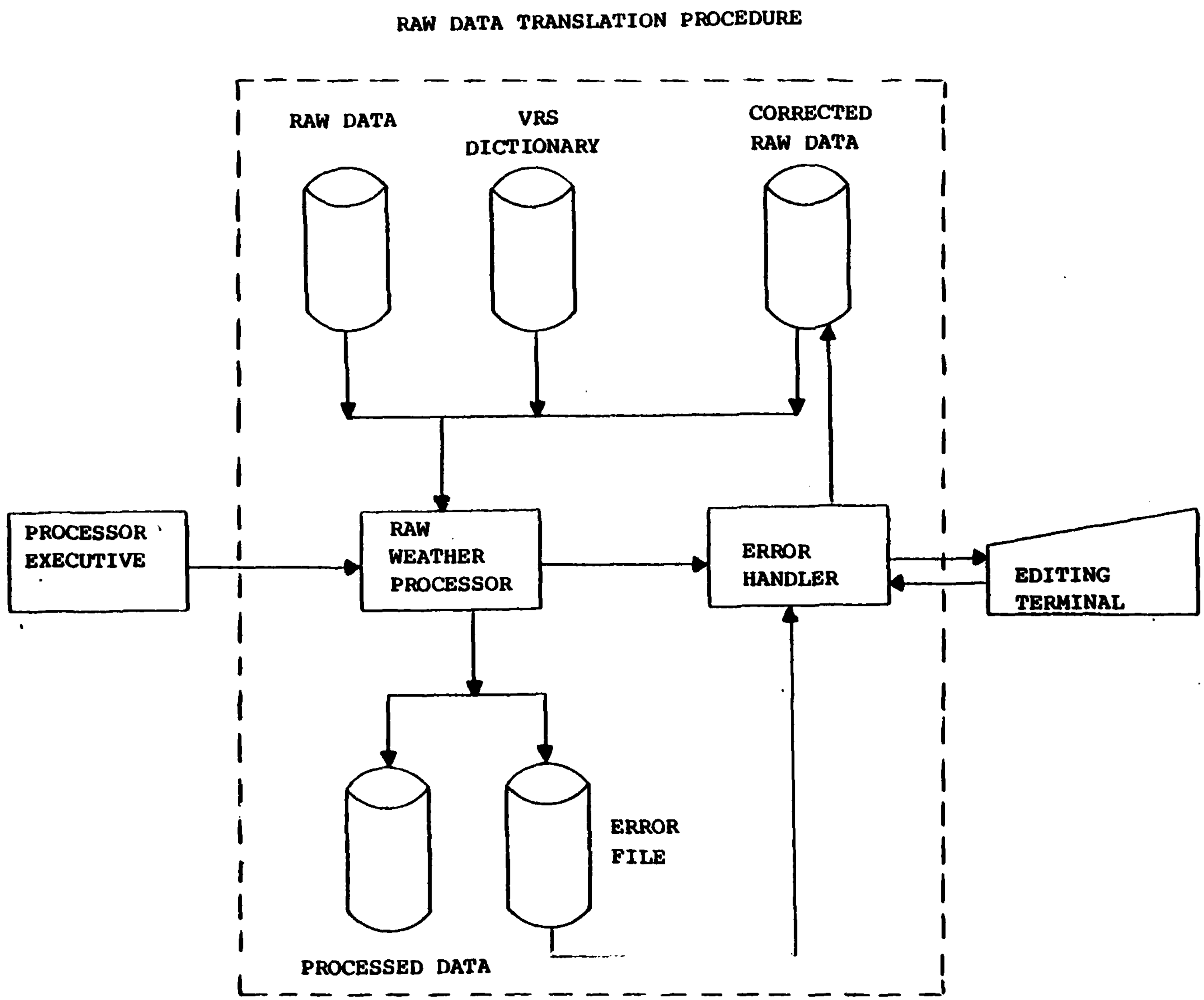


FIGURE 2-10: Raw Weather Message Processor

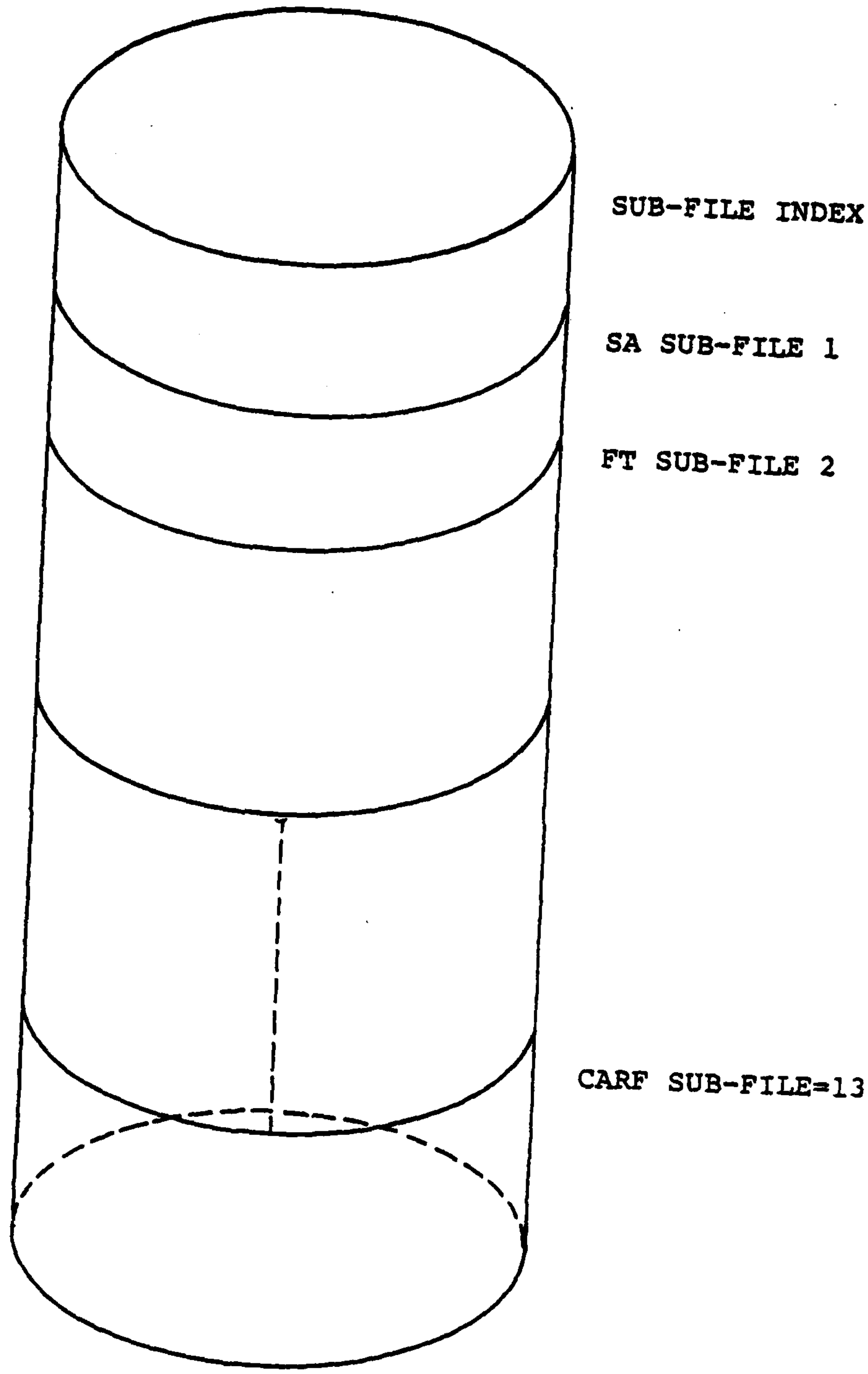


FIGURE 2-11: Raw Data Base File KCW.DAT

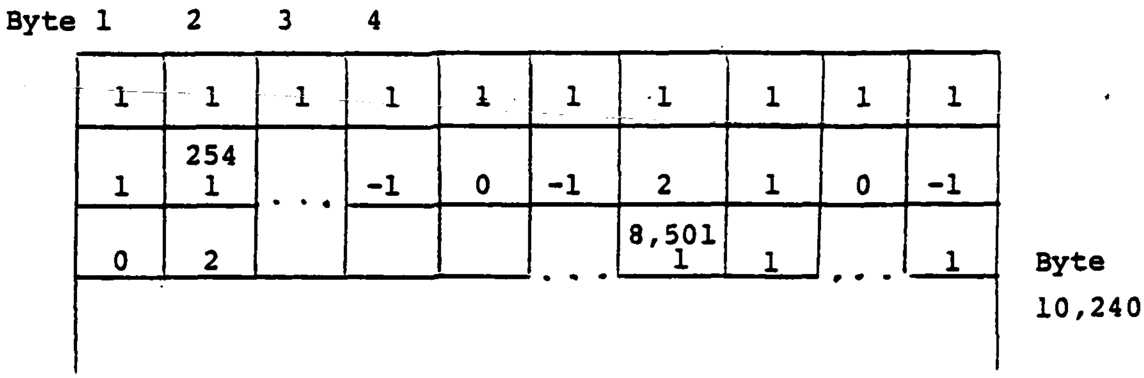
2.4.2.2 Universal Data File - The general aviation weather from the WMSC line is translated and placed in one file on the 11/70 disk. This Universal Data File (UDF) contains all the elements required to perform the processing (translation) of the raw weather data into retrievable VRS "message-units." The UDF occupies an area of 10,240 blocks of disk space and is comprised of five primary components (see Figure 2-12).

2.4.2.2.1 Map Array - A map array of 5120 words is used to depict the allocation status of all the disk blocks in the file. Each block of the disk is represented by a byte in the map array and its value indicates the current status of its corresponding data block. There are four general conditions represented by each byte in the map array. They are: block allocated and contains a valid report; block in use; block not in use, and available for a new report. The map is used by both the processing and the retrieval functions of the system. The map is read into the Global Common Area (GCA) at system initialization time. It will be replaced at system shut down or powerfail time (see Figure 2-13). In its initial design, the first twenty blocks of the UDF were occupied by the map array. Now, since the map is only in the GCA, these twenty blocks are free for system expansion.

2.4.2.2.2 Regional Report Table - The twenty-first block of the Universal Data File is the Regional Report Table (RRT). This area (256 words) will contain the identifiers for all regions of the U.S. and the virtual block number where that report resides. The dimension of the array will be the number of regional areas by the number of regional report types. When a regional report is being reported, the retrieval software will first determine the region for the requested location identifier, then get the report from the block number indicated by the address in the RRT.

UNUSED	- 20 blocks
REGIONAL REPORT TABLE	- 1 block
LOCATOR INDEX TABLE	- 233 blocks
PROCESSED WEATHER DATA IN MESSAGE UNIT FORMAT	Up to four message units (MU's) per block; One report per block; Blocks chained for reports larger than four MU's 8,246 blocks
WINDS ALOFT DATA	1,740 blocks Not in MU format. The first 1,271 blocks unused. One block used for Winds Aloft data status. 468 data blocks.

FIGURE 2-12: VRS Universal Data File



Each Byte represents the status of the corresponding Block in the UDF. The first 254 and the last 1,740 Indicator Bytes will always be set = 1 to indicate the presence of permanently allocated blocks.

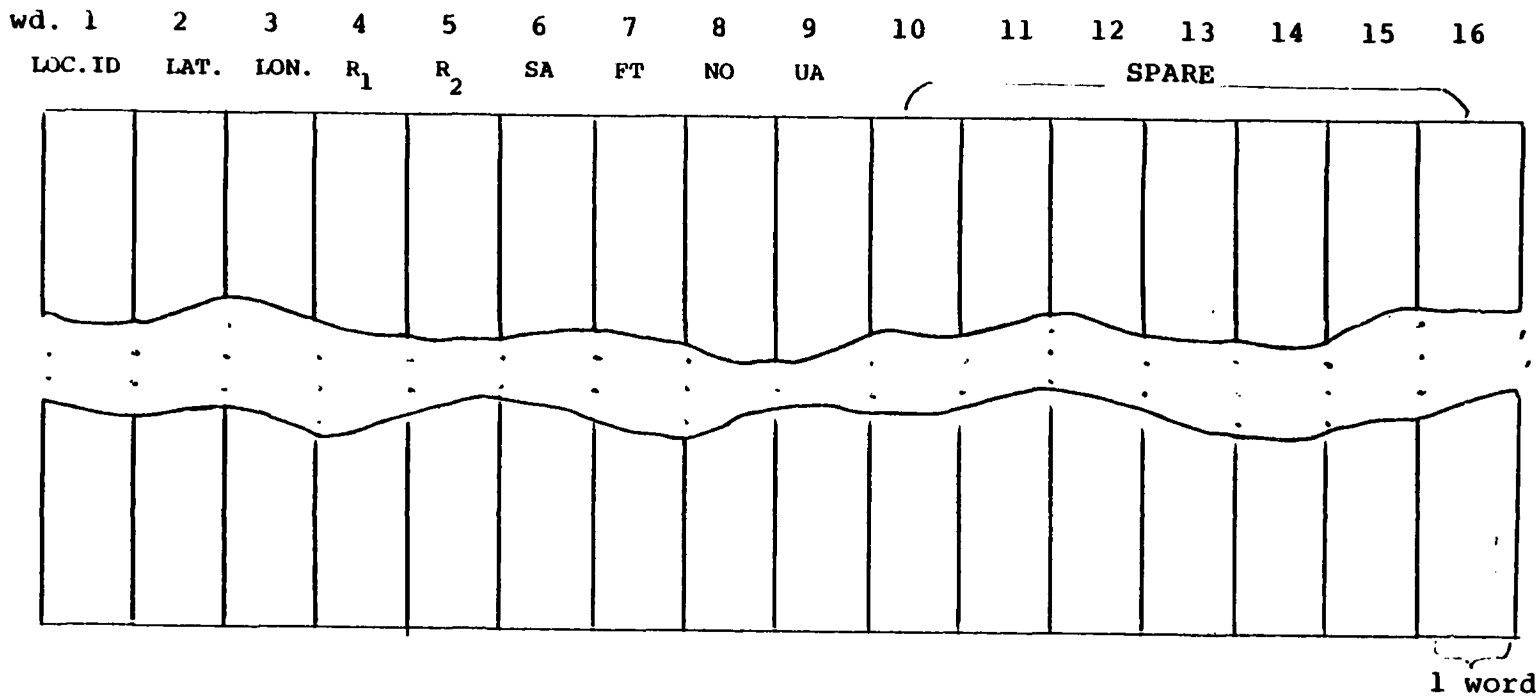
<u>Key</u>: Byte =

- -1 - block available for use
- 0 - block to be de-allocated; report no longer valid
- >0 - block contains valid report

FIGURE 2-13: VRSGLB Map Array

2.4.2.2.3 Location Index Table - The next area contains the matrix of location identifiers by report type. It is an area of approximately 60 thousand words and is used to determine the location of a particular report within the UDF. The value found at the juncture of the report type requested, for a given location identifier, represents the block number in the UDF where that report has been placed by the message processor. The LIT is contiguous in the file and does not contain any header or trailer information. A stand-alone program (UDFPRG) creates the LIT array and the program is also used to effect any updates to the index table. (See Figure 2-14.)

2.4.2.2.4 Message Unit Data - The remainder of the UDF is comprised of the processed weather data. These data (with the exception of the Winds Aloft data) reside in the file in message unit format. That is, the data have been processed and the reports have been translated into message units ready to be retrieved and sent to the 11/34. All retrieval is accomplished by using block I/O. Each block (512 bytes) contains up to four message units. Each message unit is prepended by eight words of header information in integer form. Also, each block contains an eight-word header. This leaves room for four 54-word message units (27 spoken items) per block. No block ever contains message units from more than one report. If a report requires more than four message units, several blocks may be chained together to link the message units together for the retrieval function. These linked blocks need not be contiguous to carry out this procedure. The link indicator in the header contains the block number of the lined block for access purposes. The internal format of the message units consists of paired voice pointers. Each recognized word of the original report is converted to a location pointer and corresponding length code via a dictionary look-up task. The pointers and lengths are then put in the message unit and stored in UDF. (See Figure 2-15.)



- o LOC.ID in RAD50 Notation
- o LAT. & LON. in minutes
- o R_1 - Region in which LOC.ID falls
- o R_2 - Sub-region (if needed)

For each entry (LOC.ID) a line contains: LAT. & LON. of that location; the region in which that location resides; a sub-region; the location (block number) in which the current reports can be found. A zero indicates there is no valid report of that type for that LOC.ID in the system.

FIGURE 2-14: Locator Index Table Format

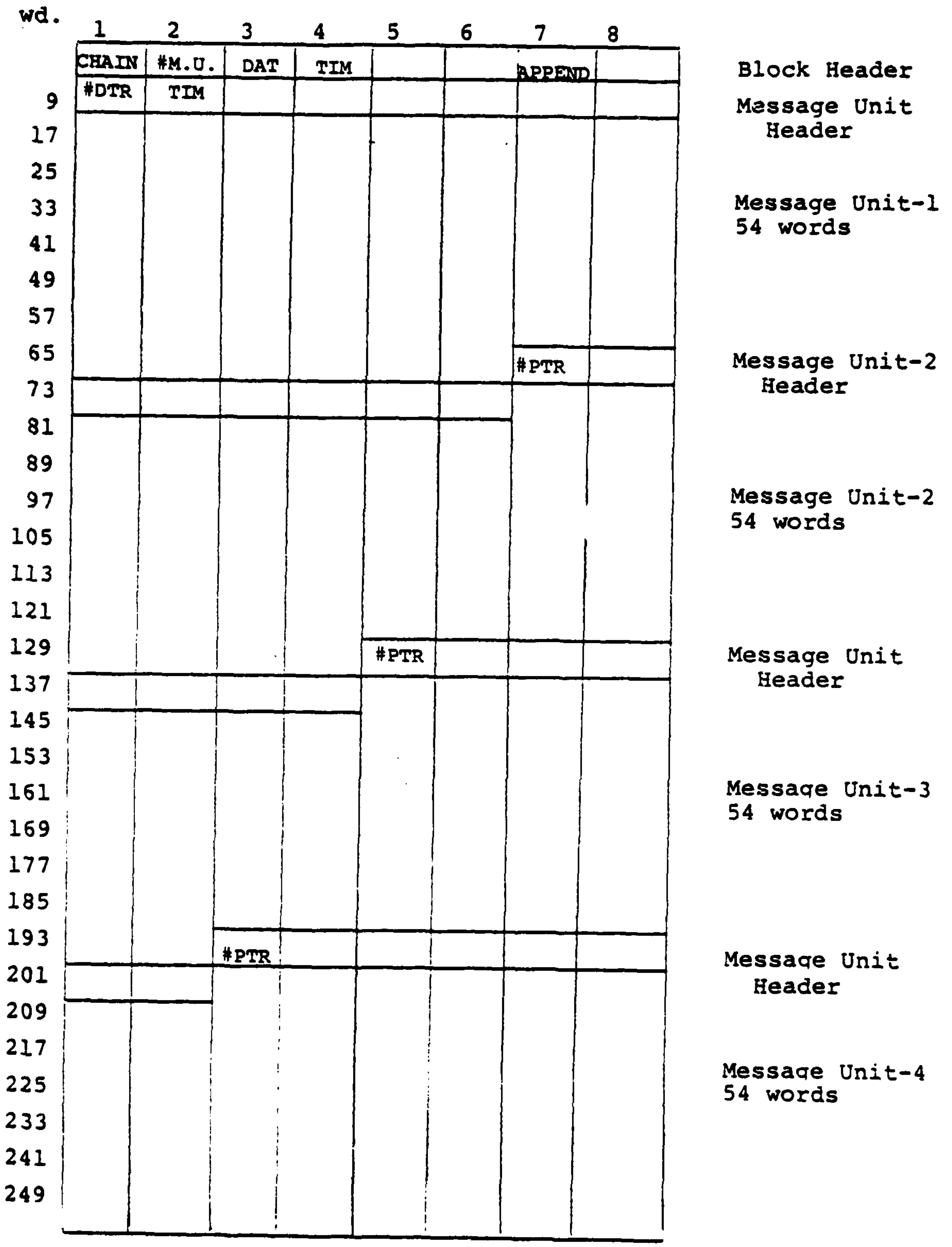


FIGURE 2-15: Message Unit Format for a 256-Word Block in UDF

2.4.2.2.5 Winds Aloft Data - The last 1740 blocks of the UDF contain the processed Grid Winds Aloft data. The Winds Aloft data are not stored in the message unit format as is the rest of the processed data, but rather contain numerical values of temperature, X and Y wind vector coordinates for various altitude levels at specific geographical points. The further processing of the data into message unit format is a function of the winds retrieval software (FDRTRV). This is due to the nature of the winds data. To report the wind speed, direction and air temperature, a specific location is required (latitude and longitude of a location identifier) and an altitude. The desired values are then obtained by interpolation of data for specific grid points. This process can only be done at retrieval time. The winds data also carry a header indicating effective time and date of the forecast.

2.4.2.3 Initialization of Data Base UDF.DAT - At system start-up a stand-alone program is run, VRINIT, to initialize the UDF data base. First the map array is initialized by setting the weather data blocks free, with all others, such as LIT and Wind Data Block, set for "in use." The LIT is then scanned for report blocks in use. If an error has occurred and one block is in use for two locations or reports, those reports are zeroed. After initializing the map array, the KCW file pointers for the VRS are reset to the last major weather transmission for each report type.

2.4.3 Raw Data Processing

The various types of weather data have significantly different characteristics. This creates the need for multiple processors, each tailored to the individual requirements of the data. Each sub-file of raw data is accessed by its own processor routine. The routines are in the form of overlaid modules to be used, in conjunction with the executive routine (Figure 2-10), to accomplish the raw data processing.

Each processor routine will be constructed to account for the differences in structure and content of the various report types. The general functions of recognizing individual words, inserting header of "blocking" words and performing maintenance procedures on the raw data file will be common to all processing routines.

2.4.3.1 Processor's Executive - An executive structure, called VRS on the PDP-11/70® maintains control of the execution of the individual processor routines. The routines are brought in and used as an overlay structure. The executive continuously monitors the sub-file activity and brings in each processor to translate the data in the raw KCW file. If there has been no activity (no new data have been received), the executive continues to scan through the sub-file indices. If there has been activity in the sub-files, the appropriate processor is invoked. If there has been no activity, the executive prints the processor statistics and then puts itself in a wait state for two minutes. After this time, the executive again begins polling the status of the raw data file.

2.4.3.2 Message Processing Routines - Each type of weather data is translated by a separate processor routine. Each routine is tailored to suit the raw data configuration of a particular report type. These routines are in the form of an overlay structure so that only one processor is in execution at any time. An overlay consists of the main processor and several supporting subroutines. Under the RSX-11D system, this procedure is carried out similar to regular Fortran subroutine calls after the overlay threading has been accomplished during the task-build phase.

Each processor executes the translation procedure on a full report basis. A complete report is translated and all recognized words, plus any "blocking" words required, are placed in a single array. This array of words is returned for dictionary translation. When the entire report has been processed, the processor returns program control to the executive.

The current weather processors available are for surface observations (SA) and surface observation remarks, terminal forecasts, and winds aloft. Following is a brief description of the processor design as it interacts with the VRS Executive. For a more detailed description of weather data and content checks for each processor, see Reference 7, "The Ten Channel VRS Processor Design Report."

2.4.3.2.1 Surface Observation (SA) Processor - The SA processor is an overlay module invoked by the VRS processor executive. The function of this module is to unpack, decode, and translate surface observation reports into ASCII text. The text is then translated into voice pointers and stored in a data base. The procedure used in decoding the SA data is of a scan and extract type. Initially, the report is scanned to determine the presence of four critical fields. These are the SA location identifier, the sky cover, the visibility, and the wind field. During this process pointers are set delimiting the fields present. After this is done, the individual components of the report are extracted, decoded, and placed in the output list. During this extraction process, limit and quality checks are applied to the data.

The SA Processor consists of a main routine (VRSSA) and four extraction subroutines (SUBFLD, VISWX, SKY, EXTHED). The VRSSA main routine begins the process by calling each of the extraction routines. The routines return translated pieces of the SA report. Then, VRSSA puts the pieces together in the proper order. If any of the routines has discovered a serious error (one that leaves some doubt regarding the validity of the translation), or if any of the key fields is missing, VRSSA will flag the report as erroneous and notify the executive that the report should not be placed in the processed weather data base.

2.4.3.2.2 Surface Observation Remarks Processor - After the SA Processor has decoded the report, the SA Remarks Processor Overlay is called to decode the remaining remarks of the report. Then the dictionary look-up task is called to translate the entire report. The SA Remarks processor uses a "key-word" approach to translating the data. The main routine (VRRMK) extracts one word at a time, using a blank character as a delimiter. The process begins at the start of the remarks field specified to VRRMK through a call argument received from SA subroutine SUBFLD.

The remarks processor is a separate overlay within the VRS program. It resides at the same level as the other processor modules.

The processor always begins scanning the data from the left and proceeds to the end of the remarks field. The beginning is usually one character past the end of the altimeter field. If the altimeter is missing, the beginning is assumed to be one character past the end of the wind field. The main processor routine (VRRMK) extracts a "word" from the raw data. A "word" in this context is any string of characters preceded by and followed by a blank. The word may be all numeric, all alpha, alpha-numeric, or alpha-numeric with special characters. When alpha or alpha-numeric data are found in the word, the program then attempts to identify a "key" within the word. If a key is found, then VRRMK invokes the proper subroutine. Each subroutine processes a particular type of remark. The subroutine receives the array and the pointer to where its key is found. The subroutine knows if preceding or following information is required and can step along the raw data to extract all the information pertinent to that particular type of remark. When the remark has been translated, the subroutine moves the pointer to where it ended and returns to VRRMK.

At this point, the process is begun again. This process continues until all remarks have been processed or until an unrecognized or all-numeric field signals the end of remarks and beginning of additive data. Each remark field is handled separately with no restrictions to sequence or amount of field type.

If a word containing alpha characters is extracted and no key is found in that word, it is assumed to be free text and is entered into the output array as such.

Using this approach, highly coded remarks or free text in any sequence or mix can be translated. Whenever a free-text entry is made, the processor notes its position in the raw remark. These pointers are saved and used by the on-line editor. It can be assumed that if an error occurs during the dictionary look-up task, it would be caused by a free-text entry and not by coded processing.

2.4.3.2.3 Terminal Forecast (FT) Processor - The principal objective of the raw weather data processor array is to insure reliability of the processed weather report. The Terminal Forecast (FT) Processor must be able to discern the properties of each raw weather data field to be processed such that the probability of misrecognition is reduced to zero.

It is better for the processor to flag a weather field as a non-recognition error than to process it incorrectly. The processor, however, must be sophisticated enough to reduce the amount of non-recognition errors being sent to the editor.

In order to achieve this goal of zero misrecognition errors and a low amount of non-recognized fields, the FT processor is designed not only to determine what a field is, but more importantly, what a field is not.

The Terminal Forecast (FT) Processor must process the eight fields contained in an FT report. The FT fields are:

1) Station Designator
2) Bulletin Notice
3) Date-Time Group
4) Sky/Ceiling Cover
5) Visibility/Precipitation
6) Winds
7) Remarks
8) Time.

An FT report always contains a heading of station designator, a possible bulletin notice, and a date-time group. The body of the report, however, contains multiple time groups in which the remaining fields may or may not occur. Also, the field may be embedded within a remarks field. In order to handle these discrepancies efficiently, the processor routine calls a recognition routine for each field as the characters are read in from the array. Each recognition routine scans the "character" group and reports one of three conditions: (1) it is definitely the recognizer's field; (2) it is probably the recognizer's field; or (3) the field is not recognized at all. The character group is then processed by the appropriate field processor according to the following protocol.

A single, definite recognition of a field is flagged as the correct field, even though other routines may have reported probable recognition. If there has been no definite recognition, then a single, probable recognition is flagged as the correct field. All other conditions cause the editor to be flagged. Thus, the processor is able to make a finer distinction between fields whose forms sometime seem identical and to recognize fields whose forms frequently change even within a single time frame.

2.4.3.2.4 Winds Aloft Processor - The Winds Aloft Processor (VRSFD) accepts the winds aloft data in the order that they are transmitted and decodes them into temperature, X and Y coordinates of the wind vector, and additionally for Level 2 data, tropopause height. These data are written to the Universal Data File along with header information containing amendment designation, forecast day and time, transmission day and time, blockette header time code, and a file wrap index. The record location of the data within the UDF is determined by the blockette number, altitude level, and forecast time code.

The file structure for the Winds Aloft is organized so that data for six forecast time periods starting from a time zero reference point are available for retrieval. This is done by having a file structure which wraps around continuously, with each new forecast period data overlapping the previous forecast period data in the UDF where the data are for the same forecast time period measured from the zero reference point.

This file structure also allows accommodation of transmissions with missing or erroneous data. One block in the UDF is set aside for storing file record pointers, special information flags, and time data for both the Winds Aloft processing program and retrieval program. The information contained in this "master" block allows the Winds Aloft programs to function correctly after periods of computer down time and allows correct storage and retrieval of processed data at all times.

2.4.3.3 DICT - The dictionary task translates ASCII text to a group of speech file pointers. The task is installed and can be used by any caller. The data is entered in VRSGLB array PDICIN if called by the VRS processor and the speech file pointers are returned in the array PDICO. When called by FDRTRV for winds retrieval, the VRSGLB array is ATADII and output appears in ATADIO. DICT uses a binary search algorithm to find the data. It returns the speech file pointers and a word containing the length in bytes of the translated pairs. On the event of a failure of translation, the routine returns pointers to where the text was in the original string which could not be translated.

2.4.3.3.1 Dictionary Structure - The raw data in ASCII format must be put in a form recognizable by the VRS system before it can be spoken. This is accomplished by using a core resident dictionary and corresponding look-up procedure.

The dictionary contains the VRS spoken word index number and a length code for each word or phrase that can be spoken by the VRS unit. The dictionary program uses a binary search to locate the proper index and length code for each recognized ASCII word it receives.

The look-up procedure is carried out as an installed task. The task is invoked by the processor executive as stand-alone and is not re-entrant. The dictionary task, when activated, is presented with the array of recognized words prepared by the individual processor routine. The dictionary task proceeds to create a list of length codes and pointers on a one-for-one basis and returns this list to the executive by placing it in the GCA array. Also, an error flag is set to indicate if the report contained any words that could not be found in the VRS dictionary file. Control is then returned to the executive.

2.4.3.4 VRSOUT - A separate installed task VRSOUT is called by the VRS executive to write the array of dictionary pointers into the UDF. The array is stored in the VRS global common area by the dictionary. Upon being called by VRS (11/70) to output a report, first, VRSOUT checks for a Surface Observation (SA) special report. If the report is special, it is appended to the current SA report by the subroutine SASPEC.

The basic component of speech in the system is the message unit. Each message unit can contain up to 27 pairs of VRS pointers (i.e., 27 spoken words or phrases). During the retrieval process, the messages units are taken from the data file (UDF) and transmitted to the VRS computer. The format of a transmitted message unit is shown in Figure 2-16.

After a report has been translated by the processor, the array of VRS pointers is taken by the block formatting routine (BLCR8). This subroutine places the paired VRS pointers in the message unit format and creates an output block. Each message unit is prepended with appropriate header information for its report type. The format of a message unit within the UDF is shown in Figure 2-16.

The map array is scanned for free UDF blocks and their corresponding map bytes are set. The subroutine IOBLCK is called to output the block to the UDF. This procedure is repeated until the entire array is output. A chain word is used to indicate the next block of the sequence of blocks with zero indicating the last block. The new report block then replaces the old report in the LIT. The old block number and its chained block map values are decremented to free the unused blocks.

Before the VRS executive starts its wait cycle, it calls VRSOUT to exit. When VRSOUT receives an exit command, it first scans the map array for unused blocks (bytes equal to 0, see Figure 2-13). The free indicator (bytes equal to -1) is set for each unused block. VRSOUT then exits from memory.

<u>VRSPURG</u> - The function of the subroutine VRSPURG is to purge Hourly Surface Observation (SA's) and Terminal Forecast (FT's) reports from the data base when they are considered to be too old and no longer valid. The routine is called by VRSOUT once each hour during the time period of 15 minutes past the hour to 45 minutes past the hour. As most of the SA and FT reports come in between on-the-hour

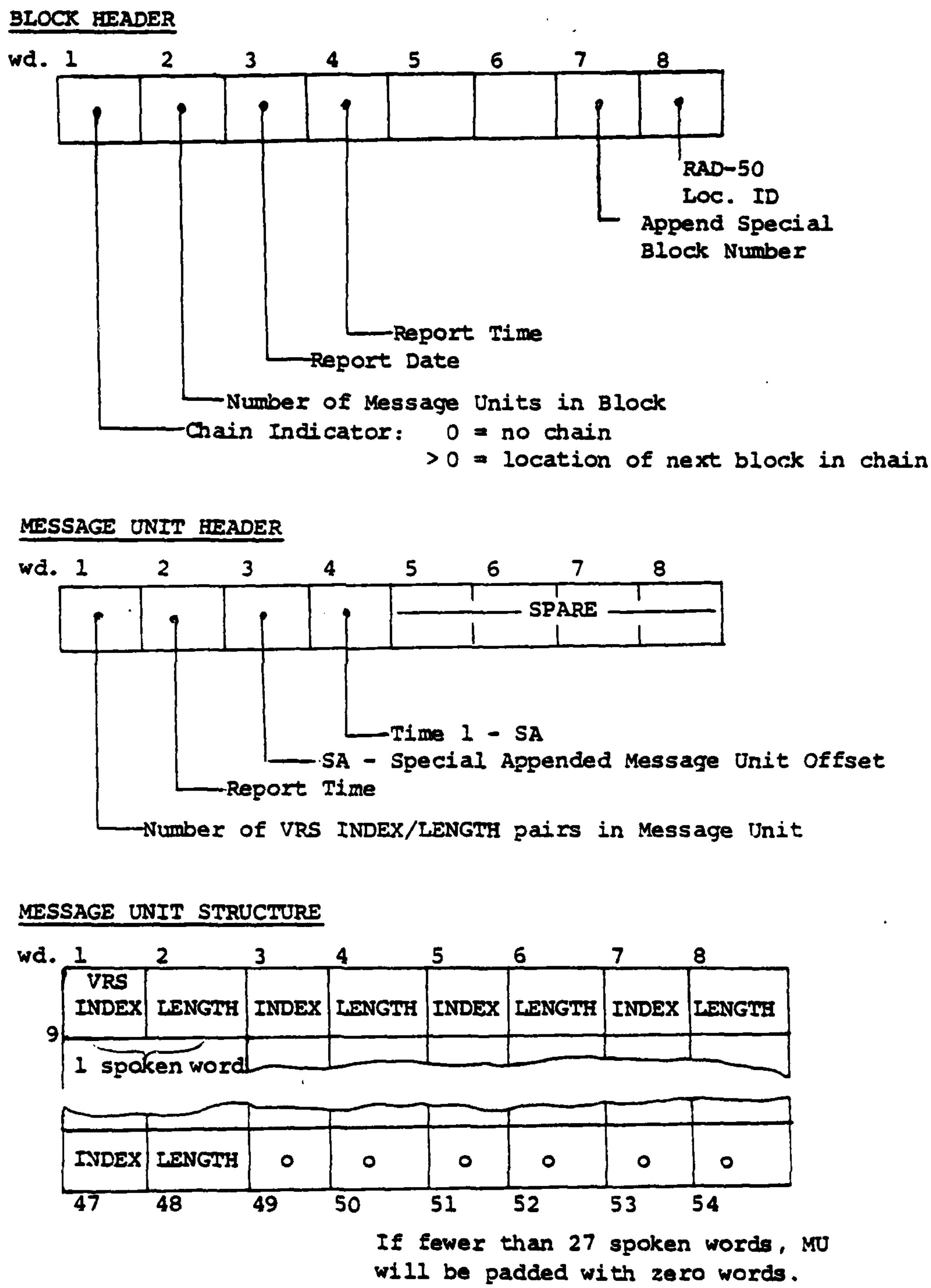


FIGURE 2-16: Transmitted Message Units

and 15 minutes past the hour, calling VRSPURG in the time frame given previously allows for new data to replace old data in a normal fashion and reduces the workload of VRSPURG by eliminating unnecessary purging. Hourly Surface Observations are purged when they have become more than 2 hours old. Terminal Forecasts are purged when they have become more than 8 hours old.

Each time VRSPURG is called, it scans every SA and FT report in each page of the locator index table (LIT). When a report is found to require purging, VRSPURG calls the subroutine NOTAVB. The sole purpose of NOTAVB is to create a standard message of "current report not available" to replace the report to be purged. It does this, returning the UDF block number of the canned message to VRSPURG. . VRSPURG then replaces the old SA/FT report block number in the LIT with the canned message block number. When every LIT page has been scanned, VRSPURG returns to VRSOUT.

2.4.3.5 Data Edit Position - When a report is determined untranslatable by a weather processor, the report is written to an error file. The Data Edit Position (DEP) software reads the report, displays it on a screen, and allows a DEP operator to correct it.

After an operator has made all the corrections to the report, it is written into another area in the file for later translation by the VRS weather processor. The data edit position software is composed of three major components; terminal tasks, (DEPTT), a service task, (DEPST), and a data base, (ERR.DAT). The following sections describe the functional description of the Data Edit Position. For a complete description of the Data Edit Position, including the Data Edit commands, see Reference 8.

2.4.3.5.1 Error File, ERR.DAT - The erroneous and corrected reports are kept in the error file, ERR.DAT. The file is structured into three parts: the pointer blocks, the error subfiles, and the corrected subfiles. This file is created by the stand-alone program ERRCRT.

The first section is contained in the first two blocks of the file. The first block contains the VRS executive read and write pointers to each subfile. The second block contains the DEP Service Task read and write pointers for the subfiles. Each subfile has a five parameter pointer set. These are the subfile start and end block, the next report block and integer offset, and the report sequence number. The only exception to this is that the VRS read pointers contain the next report block and byte offset to correspond to its GETRPT software. The next section of the file is the circular subfiles containing the error reports received from the VRS weather processors. Each subfile contains a report type.

The third section of the file is identical to the error file except that this section contains the corrected reports received from the Data Edit Position.

2.4.3.5.2 Data Edit Position Service Task - The DEP Service Task (DEPST) is a communications driven service module which provides information for the VRS and interfaces between the error file and the DEP terminal tasks. All requests for service are queued by the RSX-11D operation system and are handled in the order in which they occur. Hence, the DEPST is dedicated to a specific task which is making a request until the request is honored. After performing the indicated service, DEPST suspends itself until more requests are generated.

There are five types of requests sent to DEPST, one by the VRS (11/70) and four from DEPTT. The VRS executive only requests the service task to update its pointers to the corrected report subfiles.

When a terminal task enters memory, it requests the Service Task to assign it buffer space in the Global Common Area. The Service Task keeps track of which terminal has been assigned to each buffer space of 256 words. Upon request, the Service Task places the next error report into this common area for the Terminal Task. The Service Task obtains the error report from the proper error subfile. It checks the date and time of the error report and the current report in the UDF for the corresponding location. The error report is dropped if it is not the most recent report in either file. This insures that the operator would not have to correct an already expired report. When a report has been corrected, the Terminal Task requests it to be filed. The Service Task files the report in the error file and updates the pointers. A DEPTT requests exit permission when a DEP operator types the "EXIT" command.

Upon receiving the exit request, the DEPST frees the assigned buffer space. If there are no other terminal tasks being serviced, DEPST also exits memory.

2.4.3.5.3 Data Edit Position Terminal Tasks - The DEPTT's are dedicated tasks which, when run, communicate with the DEP operators by way of CRT displays. The tasks only interface with the rest of the DEP system through data stored in the Global Common area and the RSX-11D Send and Receive commands, which the Terminal Tasks use to request operations from the Service Task. After initialization, a Terminal Task first requests to be assigned buffer space by the DEPST. When this has been completed, the Terminal Task then awaits input from the operator requesting a report to edit. With this information, the Terminal Task requests the report from the Service Task. The report is placed into the Global Common Area assigned buffer (see Figure 2-17). The operator's edit commands are then performed on the report until a file or drop report is received. If another report is requested, this process is continued. When all error reports have been corrected, or when the operator types the exit command, the Terminal Task notifies the Service Task, and then exits memory.

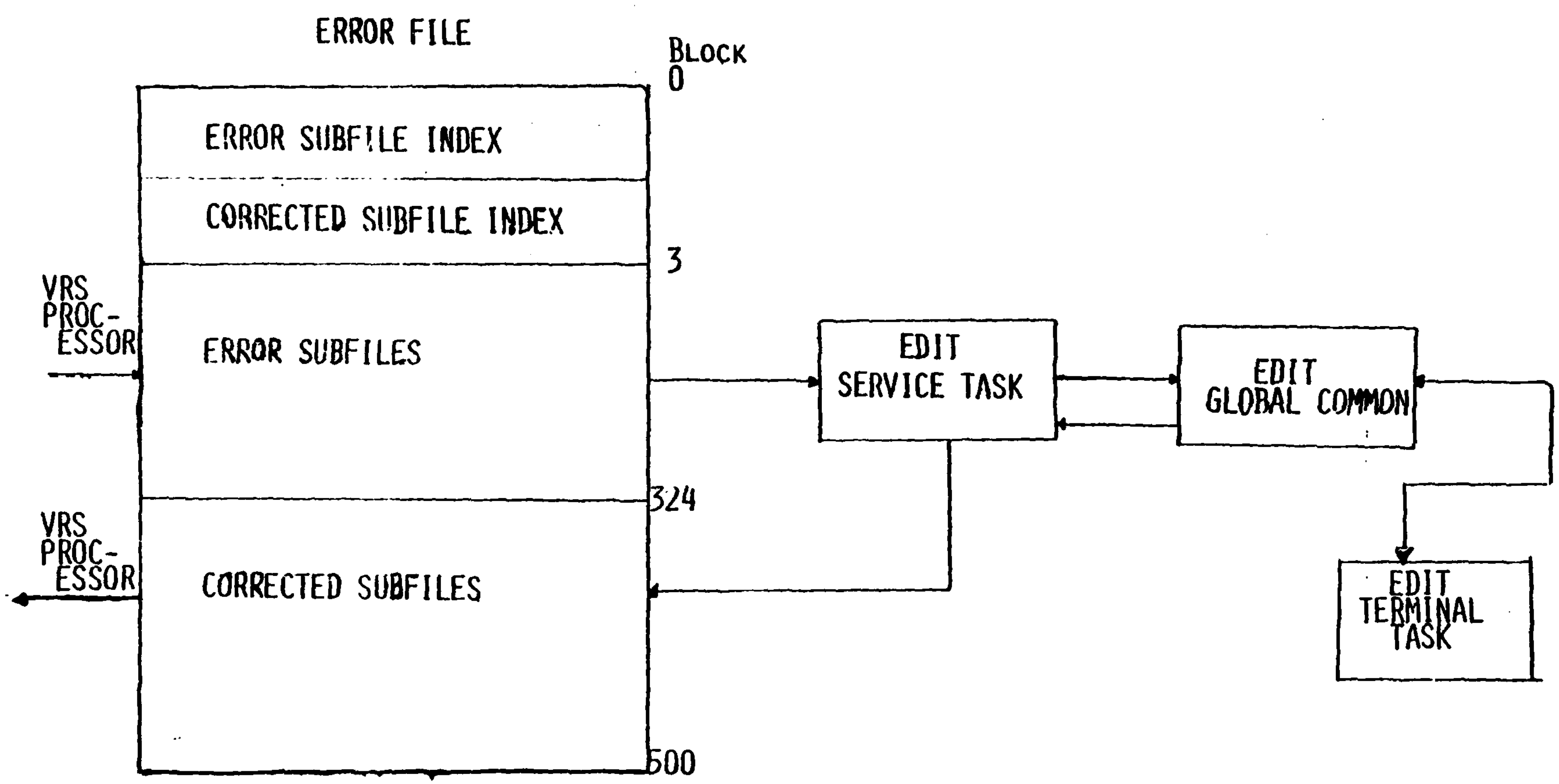


FIGURE 2-17: Data Edit Configuration

2.4.4 PDP-11/70® Retrieval Task

The twenty-channel resident PDP-11/70 retrieval software is a multi-channel program responsible for receiving and interpreting results from the VRS computer and honoring those requests by supplying weather information from the weather data base. The inputs from the VRS computer take the form of specific requests for message unit elements of the weather data base (demand response), or of supplying the parametric information defining the briefing requested by the user (briefing request message Section 2.1.2.1).

It is the responsibility of the retrieval task to access the weather data base independently, building briefing tables for asynchronous access for the VRS computer. The process of constructing briefing tables may occur several times during each user session (briefing) in order to progress through briefing phases. Each briefing phase (sub-briefing) is delineated by a briefing request message #2 (Section 2.1.2.1). The VRS computer employs the briefing request message #2 to cause the retrieval to build a sub-briefing. When the VRS computer has requested all of the message units it requires (dependent upon user Touch-Tone® interactions) as a result of briefing request message #2, it may issue a subsequent briefing message #2, to cause the retrieval program to build another briefing table. During a channel briefing, there is only one briefing table, the progressions from sub-briefing to sub-briefing are conducted only in a forward-going manner. That is, the VRS computer may not request message units from the briefing table for any briefing request message #2 prior to the briefing request message #2 currently being processed. Figure 2-18 shows a baseline structure for the PDP-11/70 retrieval task.

2.4.4.1 Retrieval Task Organization - In order to take advantage of the RSX11D/V6B, event-driver, multi-programming system, the PDP-11/70 retrieval task is comprised of three basic components: an executive level; an interrupt level; and an internal data base used

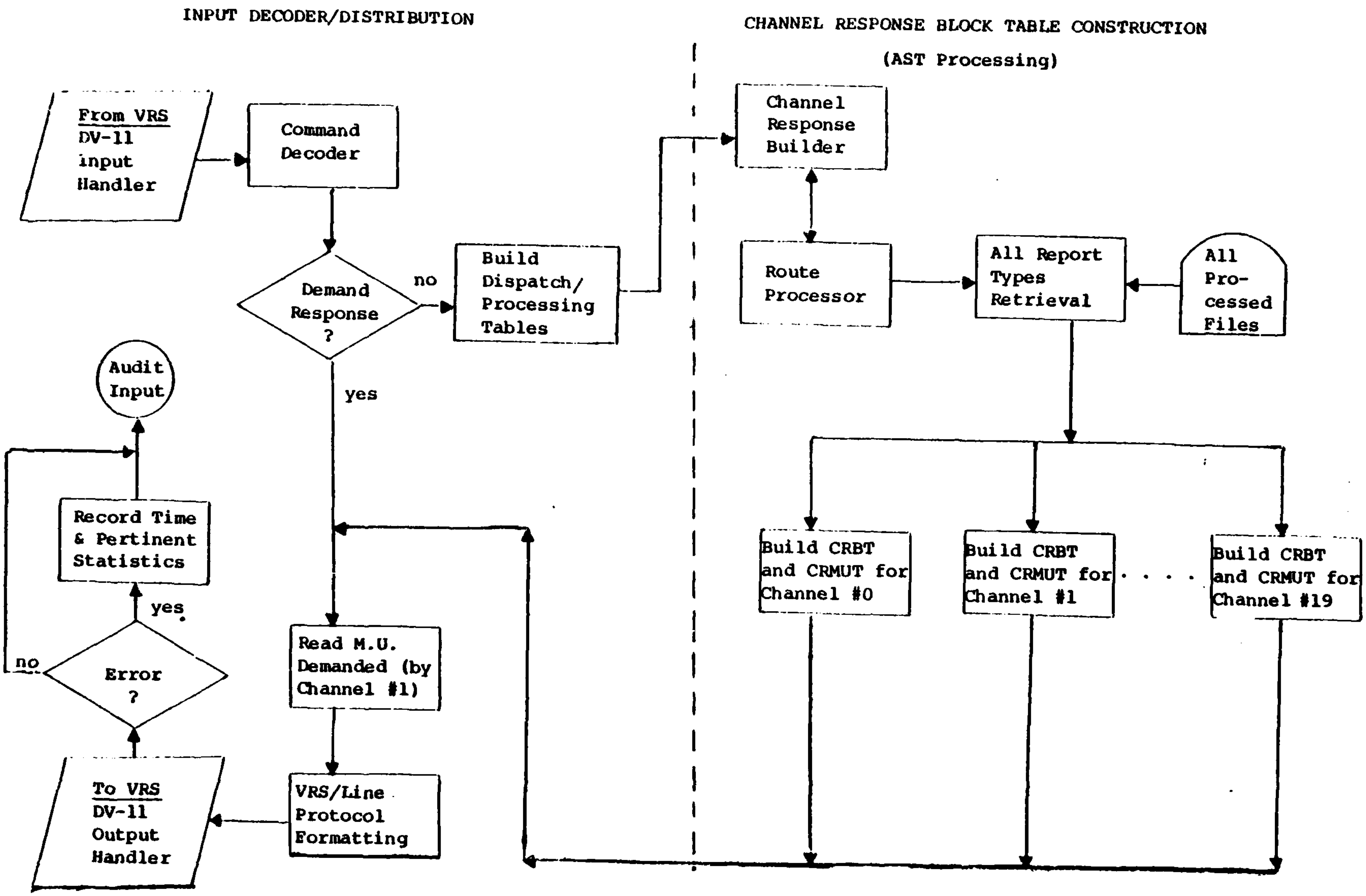


FIGURE 2-18 : PDP-11/70® Weather Retrieval Software for communication between the executive and interrupt levels, and also used for inter-computer communication, disk transfers, tables, flags, and variables of processing. The interrupt level will be defined as asynchronous trap (AST) processing. With reference to Section 2.2, the executive level may be considered as analogous to the VRS computer background processing and the AST level may be considered as analogous to the VRS computer completion routine processing.

2.4.4.1.1 Retrieval Task Data Base - To maintain channel independence and integrity, a data base consisting of eight hundred words per channel is used for all channel dependent variables, flags, I/O areas, tables, etc. In addition, another area consisting of twenty buffers of sixty-four bytes is maintained as a queued input buffer, for receiving VRS computer commands.

2.4.4.1.1.1 Input Buffer Queue - The input buffer, labeled BUFFER, consists of forty elements. Each element contains sixty-four characters, where the first two bytes are used as a linkage thread, and the last sixty-two are used for storing the commands received from the VRS computer.

The threads are used to maintain information as to the logical assignment of the elements. Two list headers (queues) are maintained. Each list header contains two words, where the first word is used to point to the top of the list, and the second word is used to point to the tail (end) of the list. The two list headers are used for maintaining a queue of "in use" elements, and for maintaining a queue of "available" elements.

By the process of maintaining the elements' threads, buffer elements may be accessed in the order in which the VRS computer transmits commands, thereby ensuring that the PDP-11/70® retrieval program services the VRS computer requests in the order presented.

This does not assure responses to the VRS computer will be in the order of received requests. Because of the length of time of command, services will not, in general, be uniform.

Figure 2-19 is a representation of the input buffer, and the two list headers. The figure assumes that the queue for "in-use" elements is labeled RETQUE and the queue for "available" elements is labeled FREEPL. The linkage threads are the element identifiers, and the thread ends with the element whose linkage is zero. The figure shows that elements 2, 3, and 4 are "in-use", element 5 is currently assigned as the input area for the current outstanding read function, and the remaining elements are "available." They will be assigned in the order: element 6 through element 20 in order, then element 1. If any "in-use" element were to be released, it would be placed at the tail of the FREEPL queue and element 1's linkage thread would be replaced with the freed element's identifier, whose linkage thread would be zero.

2.4.4.1.1.2 Channel Status Block - In order to maintain complete channel independence, and to maintain briefing state information for each channel, a sixteen thousand word block of memory is allocated, eight hundred words per channel. The channel status block (CSB) is used for maintaining all the information relative to the operation of the channel.

All flags, status indicators, disk transfer buffers, VRS output buffers, etc., are contained in this area. In addition, all driver tables and parametric information required for constructing the desired briefing are in this area.

The retrieval program constructs the briefing directly onto the CSB. It consists of a list of virtual disk blocks of the weather data base. The following items are entries in the CSB.

Linkage Thread	Received Characters	Element
0	$C_1, C_2, \ldots C_{n}$	1
3		2
4		3
0		4
0		5
7		6
8		7
9		8
10		9
11		10
12		11
13		12
14		13
15		14
16		15
17		16
18		17
19		18
20		19
1		20

RETQUE: 2 (head)
4 (tail)

FREEPL: 6 (head)
1 (tail)

FIGURE 2-19: BUFFER, RETQUE, FREEPL

- **DIOA** Disk I/O Area
  This area occupies 256 words and is used as the block transfer area from disk into memory.

- **QB** This word contains the number of the BUFFER element currently in use for the channel. It is saved for the requirement that element numbers must be retrievable so that they can be used in the buffer release call.

- **MODE** This word is used to save the mode under which the current briefing is operating.

- **DIAGP** This word is used to maintain the next available byte position in the diagnostic buffer for the channel.

- **CRBT** Channel Response Block Table (Briefing Table). This is a table which contains the UDF virtual block number of each block required for the briefing currently in progress. Every block is entered regardless of whether it is the start of a linked-block indicating report continuation. The table is constructed in a top-down manner in which each succeeding entry logically follows its predecessor for purposes of the briefing presentation. There is no relationship of the virtual block numbers to other virtual block numbers, other than briefing order. (Size 300 words.)

- **CRMUT** Channel Response Message Unit Table. Because of the requirement to deliver message units by number and because of the construction of the data base in which each block may contain either one, two, three or four message units, a table of cumulative count of message units must be maintained. The CRMUT contains the least message unit (LM) number and the greatest message unit (GMU) number in the briefing message unit sequence for the current block. A demand message unit, not within the range of the CRMUT, will cause the appropriate block to be read.

- DIAGB This is a sixty-four word area into which diagnostic messages are constructed. These are the messages which are transmitted to the VRS computer for the purpose of either indicating command compliance or for indicating why compliance is not possible (Section 2.1.3).

- ALT This word contains the requested altitude for processing Winds Aloft Data and for determining the filtering of reporting points along a flight path.

- HOURS This word contains the "forecast-ahead" time for which Winds Aloft Data are required.

- LMUS This word contains the number of the last message unit sent.

- RPMSK This is a table of requested report types and is constructed from the information received in a BRM2 transmission.

- RLOCS This is a table of sixteen-word entries which are the locator index table (LIT) entries corresponding to the requested location identifiers. The entries are extracted from the locator table index at the time of location identifier confirmation. They are held in the channel's status block area in order to obviate the necessity for reading the disk each time a report isolation is required. That is, the function of reading a report requires only reading the report and not reading the locator index table again.

- **LOCPTR** This is a position indicator for accessing the RLOCS tables.

- **BRM1E** Error indicator for briefing request message 1. The indicator may be set for a variety of reasons: request out of format; improper mode; illegal location identifier(s); improper channel, etc. The indicator is used as a switch at the end of decoding, as to whether a confirmation message is required or a diagnostic message.

- **LSTLOC** Index to the number of location identifiers residing in the RLOCS tables.

- **STAGE** The briefing stage currently attained. Because the retrieval program operates mainly as a series of AST completions, the stage indicator is used as the director for the next function to be performed.

2.4.4.1.2 Command Decoder (COMDEC) - The executive level of the retrieval program, called the command decoder, is responsible for recognizing the existence of a command received from the VRS computer, and initiating appropriate action which will cause the command to be implemented.

In order to accomplish its function, COMDEC is required to parse the input commands (Section 2.1.2.1), checking for both form and content. During the process of scanning the input command, the tables, flags, and indicators of the channel status block (previous section) are initialized and constructed in conformance with the specified command. Also, the diagnostic area is initialized and its construction is started.

The command decoder remains in a suspended state until resumed by the asynchronous trap handler which receives the communications line inputs. The input is dequeued from the input buffer area, BUFFER (Section 2.4.4.1.1.1), and the channel status block, CSB (Section 2.4.4.1.1.2), is constructed. The system is designed such that each input request causes a series of disk accesses which are processed on the AST level (Section 2.4.4.1.3). The command decoder is not required to take any further action upon an input request beyond causing the initial disk access. The disk access will in turn cause further disk accesses for the purpose of either accessing the locator index table (for location identifier verification), or accessing a block of data representing processed weather data (for demand response delivery).

After the disk access is initiated, the command decoder dequeues the next input command. If no input command has been received, the command decoder suspends itself (to be resumed by the communications line AST handler).

2.4.4.1.3 AST Processing - This level of processing may be considered as analogous to the RT-11 completion routines described in Section 2.2.4.

There are two asynchronous traps (AST) which the retrieval task is required to implement--one to handle input requests from the VRS computer via the communications line, and one to handle disk read completions.

The AST logic required for handling the communications line consists of linking the current input buffer element to the "in-use" list header (Section 2.4.4.1.1.1), acquiring the next available input buffer element from the "available" list header, resuming the command decoder, and issuing a communications line read request. In this manner, there is always an outstanding read request, which ensures that no requests issued by the VRS computer will be missed.

The function of resuming the command decoder is an RSX-11D operating system directive which will cause the command decoder to re-start if it is suspended when the directive is issued, or will not cause any action if the command decoder is not suspended when the directive is issued.

The AST logic required for handling disk read completions is dependent upon the original reason for generating the read. The final function of the disk read AST logic may be to issue another I/O request, either another disk read (which will cause another AST) or a communications line response to the VRS computer, or simply to exit, without initiating further I/O action.

There are essentially three distinct stages during a briefing session which require disk access. When the briefing request message #1 is received, it is necessary to verify that all locations requested exist in the weather data base. Each identifier verification read completion AST will start the read for the next identifier, until the final identifier is verified. The final AST will cause the AST logic to issue a message to the VRS computer.

During message unit delivery in response to VRS computer demands, the disk block containing the message unit is read. When the AST occurs, the proper message unit within the disk block must be extracted and the AST logic terminates by issuing the message unit to the VRS computer via the communications line.

2.4.4.1.4 PDP-11/70® Retrieval Task Inputs - The inputs required for the retrieval task are the VRS computer command messages and the processed weather data base.

The briefing request messages are used to construct channel dependent directive tables and parameters which become secondary inputs for locating the required weather data. The tables and parameters are discussed in Section 2.4.4.1.1.2.

The demand response messages are used to retrieve specific message units from the weather data base and send the message units to the VRS computer. The message units may be recovered and delivered to the VRS computer either in sequence (that is, in the order requested) or out of sequence in the case of repeat and skip functions. The VRS computer controls the briefing presentation order by demanding which message unit to skip ahead from. In addition, demand response messages are used to indicate channel activity, such as end-briefing, hang-up, etc.

2.4.4.1.5 PDP-11/70® Retrieval Task Outputs - The primary output of the retrieval task is message units of processed weather. The message unit information is transmitted to 11/34 VRS in response to the 11/34 demands.

In addition to the primary output there are required a series of secondary outputs which are constructed as a function of compiling the specific briefing requested.

The secondary outputs are two tables which are channel dependent and reside in the CSB. They are the channel response briefing table (CRBT) and the channel response message unit table (CRMUT).

The CRBT is an ordered list of weather data base virtual block numbers. The order is determined by compiling the list in the same order as requested by the VRS computer. That is, for each weather report type requested, the block numbers containing the weather data are written to the table in location identifier order. For example, if Hourly Surface Observations (SA) and Terminal Forecasts (FT) were to be requested for Boston (BOS), Albany (ALB) and Washington National (DCA), the CRBT would consist of the virtual block numbers of the weather data base, containing, in order, the BOS SA, the ALB SA, the DCA SA, the BOS FT, the ALB FT, and the DCA FT.

Corresponding to each block number is a "flag" word containing flag bits for new report type, skip type, and report location in the Location Index Table. As the briefing message units are demanded by the VRS computer, the block message units are sequenced. The sequence number of the first message unit of each block is entered into the corresponding message unit number (MU#) of the CRBT as the block is read. This number is also entered into the CRMUT as the least message unit (LMU). The sum of this number and the number of message units contained in the block is the greatest message unit (GMU). When a message unit is demanded that is greater than the current GMU, the next block of the briefing is read. If a message unit is demanded that is less than the LMU, the appropriate block is found by the previous MU#.

Figure 2-20 shows the construction process for the CRBT and CRMUT. The blocks are listed in briefing order with their appropriate "flag" values. For example, block 256 contains the BOS SA weather data. The flag values are:

Bit 1 = 1 BOS is the first SA report

Bit 2 = 1 SA skip protocol - skip to next report type

Last 4 bits = 1 SA is the first report in the Location Index Table.

In this example, block 466 has been read into the buffer. Its first message unit is the eighth message unit of the briefing. Since block 466 contains three message units, the eighth through tenth message unit is currently in the buffer. This is indicated by the CRMUT values.

In addition to the outputs required to satisfy the briefing (message units and briefing tables), an Error and Diagnostic File is generated. This file maintains a history of activity of the

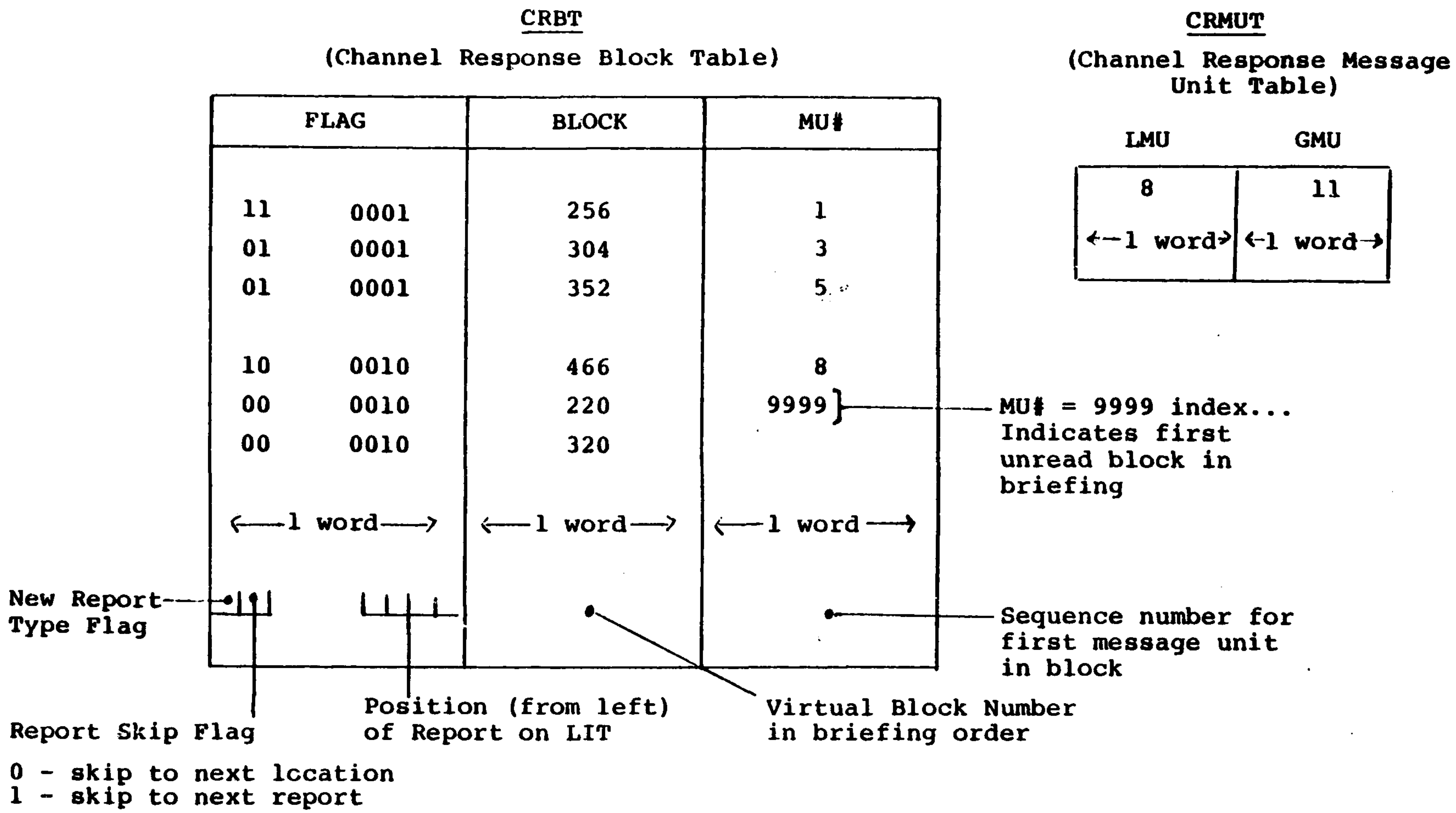
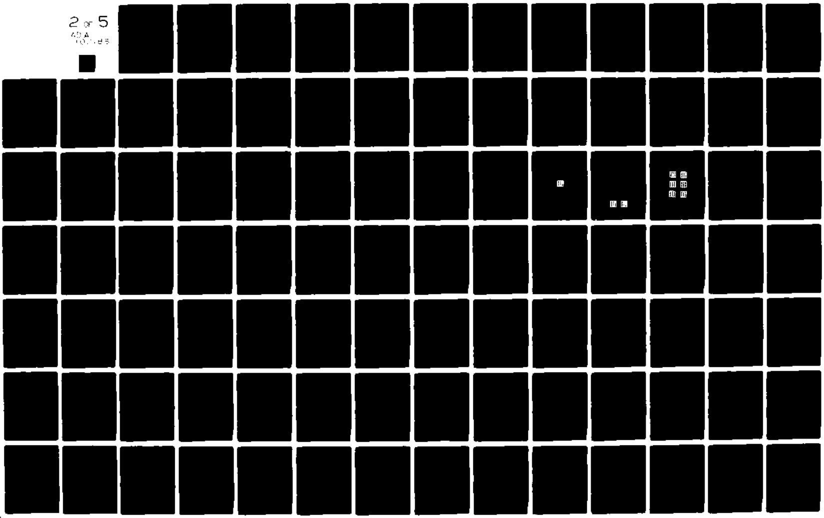


FIGURE 2-20: CRBT and CRMUT

AD-A102 185 INPUT OUTPUT COMPUTER SERVICES INC WALTHAM MA F/G 17/2
TWENTY-CHANNEL VOICE RESPONSE SYSTEM.(U)
JUN 81 DOT-TSC-1313
UNCLASSIFIED FAA-RD-81-51 NL

2 of 5
AD A
102185



retrieval task. Additional outputs of the retrieval task could be accounting information files allowing an analysis of system resource use.

2.4.4.1.5.1 Message Unit Transmission Format - The message units are transmitted according to a fixed communications protocol (Appendix B). The message units are buffered directly from the channel status block area into which they are read from disk (DIOA). That is, the address presented to the DV-11 handler is the one representing the correct message unit position of the block of data residing in the CSB.

2.4.4.2 Winds Aloft Retrieval - When a briefing request for Winds Aloft data is received by Retrieval, it, in turn, must request the data from a special, installed task, Winds Aloft Retrieval (FDRTRV). This is because Winds Aloft information must be dynamically interpolated for each location from a grid of winds data stored in the UDF (see Section 2.4.3.2.4).

FDRTRV receives and processes requests for Winds Aloft information for a given location, altitude, and time period. Restrictions on the input to the program are that the altitude cannot be greater than 45,900 feet and the time period cannot be more than 30 hours beyond the effective date and time of the winds aloft data. Blocks numbers returned by FDRTRV contain message unit data for the given altitude, an altitude 4,000 feet higher, and an altitude 4,000 feet lower (unless the given altitude was equal to or less than 6,000 feet, in which case an altitude 2,000 feet lower is given). If the altitude given is determined to be less than the estimated terrain height for the location given, then the values returned are for an altitude equal to the terrain height plus 2,000 feet and a higher altitude equal to the previous value plus 2,000 feet and a higher altitude equal to the previous altitude value plus 4,000 feet. If the altitude given plus 4,000 feet is greater than

45,900 feet, then the higher altitude values are not returned by FDRTRV. Alternatively, if the lower altitude calculated for the given altitude is lower than the terrain height, no values are returned for the lower altitude.

The values which are returned by FDRTRV for each altitude are the wind direction in degrees, the wind speed in knots and the temperature in whole degrees Celsius. Since these values are determined by interpolation from retrieved data values, if critical data are missing or have become too old, (more than 30 hours) a message of "data not available" is returned.

After FDRTRV has calculated the Winds Aloft Data and stored them in message units in the UDF, it then returns the block numbers to the Retrieval program. These block numbers are inserted into the appropriate Channel Response Briefing Table for use during the weather briefing.

3. SUPPORT SOFTWARE

In addition to the operating systems, there are programs required to create and initialize the VRS data base.

3.1 UDFPRG

Using a file (NLC.DAT) containing the name, region, and geographic coordinates of each weather reporting station, UDFPRG creates the file UDF.DAT where VRS processed weather reports are stored (see Section 2.3.2.2).

3.2 ERRCRT

When raw weather reports read from the KCW.DAT file contain errors, they are stored by VRS in an error file (ERR.DAT) where they are accessible by the editor. ERRCRT creates ERR.DAT (see Section 2.4.3.5).

3.3 DEPTT

The Data Edit Position Terminal Tasks, in conjunction with DEPST, constitute the editor used to correct erroneous raw weather reports (see Section 2.4.3.5).

3.4 VRINIT

Before VRS can be executed, certain initialization functions must be performed. The subroutine VRSMAP initializes the UDF block allocation map by flagging all table blocks as being in use and the remaining report blocks as being free. It then scans the Locator Index Table for any report blocks in use and sets the corresponding map bytes in the UDF block to one, signalling the blocks in use.

Also if there are any duplicate report blocks for locations, signifying an error has occurred in block allocation, the blocks in question are zeroed thus preventing invalid reports for location.

There exists a file, SFI.DAT, which is used by the VRS subroutine VRPAOV to determine if any new reports have been recently added to KCW.DAT. SFI.DAT contains the same subfile pointers that are contained at the beginning of KCW.DAT itself. If new reports have been added, the data will not be the same and VRS then knows it must invoke the report processors. The VRINIT subroutine, VRSPTR, initializes SFI.DAT to point to the most recent set of weather reports so that the VRS will process them as soon as execution has begun.

3.5 VRSTOP

To safely stop the VRS execution in a coordinated way that insures all files are closed and an I/O function is not interrupted before completion, VRSTOP is executed. A message is sent to the VRS executive. When the VRS sees it, an acknowledgment is sent and both the VRS and the VRSTOP exit.

3.6 NLCUPD

The file NLC.DAT, containing identifying information on each weather reporting station, is used by UDFPRG to create the UDF (see Section 3.1). NLC.DAT is built and modified by program NLCUPD, which provides editing capabilities.

3.7 SENDIC

The "dictionary" portion of the 11/34 vocabulary disk file, DIRECT.DVF, is needed by the 11/70 dictionary task. SENDIC sends it to the VRS disk area on the 11/70.

3.8 WRDICT

Once SENDIC (above) has been executed, the file created at the 11/70 is made into a common block within the 11/70 dictionary task by executing this utility.

4. VRS MAINTENANCE--11/34

For discussion of the 11/34 maintenance procedures the reader should be familiar with the RT-11V03 Extended Memory Monitor and MACRO-11 programming. The reader should have a thorough understanding of the functional flow of completion routines before attempting to modify the 11/34 software (see Reference 9).

4.1 PROGRAM CREATION PROCEDURE

The RT-11V03 indirect command file capability is used to create the 11/34 VRS software. The indirect command file ASMVRS.COM assembles the software from the MACRO sources. The following modules must be present to assemble the system:

- BACKGR.MAC
- DAP.MAC
- DICT.MAC
- SPEC.MAC
- SPEAK.MAC
- SEND.MAC
- CLOCK.MAC
- PURGE.MAC
- QUEUE.MAC
- TRAP.MAC
- TABLE.MAC
- TRAC.MAC
- PREFIX.MAC.

The following four modules must be present to generate the specialized data handlers for insertion into the RT-11 operating system:

- ADX.MAC
- LCX.MAC
- LIX.MAC
- LOX.MAC.

By typing "@ASMVRS" all object modules listed above will be generated. The object modules must be linked together to create the VRS save image file. The command file VRSLNK performs this operation. To list the software package, the users can type @ASMLST and the sources of all seventeen modules will be listed on the line printer. To generate the specialized handlers needed by the software, the command file VRSHND should be invoked.

Figure 4-1 is a subroutine tree of the 11/34 modules. Since the software is a Macro-11 asynchronous event-driven program, the tree does not depict logical program flow. It is meant to depict possible modular interface. See Appendix A for a more detailed description of the modules.

4.2 SYSTEM REQUIREMENTS

To generate a twenty channel voice response system the following assumptions are made:

- <u>Hardware</u>

    a. PDP-11 with extended memory management

    b. 64K words 16-bit memory

    c. Fast Random Access Disk with a capacity of at leaat 3.5 Megabytes

    d. Specialized DMA ADPCM Module

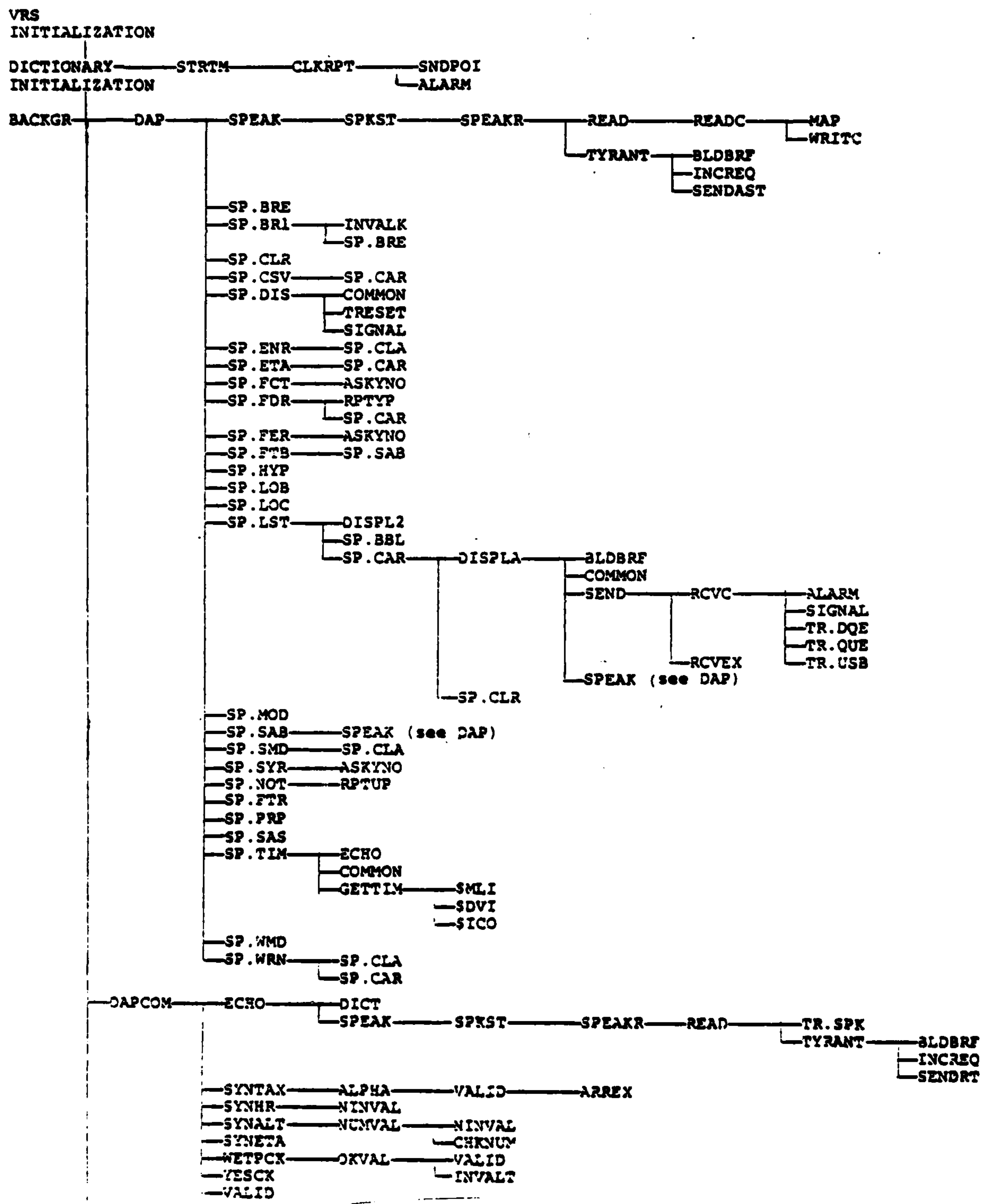


FIGURE 4-1: 11/34 Software Subroutine Tree

```
BACKGR (continued)
   |--DISABL------DISCON----+--BLDBRF-----SP.CLA
   |                        |--CHKREQ-----TR.DQE
   |                        |--COMMON
   |                        |--ECHDON-----TR.MOD
   |                        |--ENABLE-----RPTREQ
   |                        |--INCREQ
   |                        |--MRKTIM----+--MRKCON
   |                        |            |--MORSPK----+--READ------READC----+--MAP
   |                        |            |            |                    └--WRITC------MAP
   |                        |            |            |--TR.USB
   |                        |            |            └--TYRANT----+--BLDBRF
   |                        |            |                         |--INCREQ
   |                        |            |                         └--SENDRT
   |                        |            └--SIGNAL
   |                        └--RTNQUE
   |--EXIT-------+--DISABL-(see above)
   |             |--MRKTIM------SEND------+--RCVC------+--ALARM
   |             |                        |            |--SIGNAL
   |             |                        |            |--TR.DQE
   |             |                        |            |--TR.QUE
   |             |                        |            └--TR.USB
   |             |                        └--RCVEX
   |             |--STRT
   |             └--TRESET
   |--GO---------TR.MOD
   |--INVALK-----+--CLRTTK
   |             |--SPEAK        (see DAP)
   |             └--TR.MOD-------TRACE
   |--NO
   |--NORMAL
   |--NXTCAR-----+--NXTEXT
   |             |--PROCA
   |             |--PROCCR
   |             |--PROCD--------SIGNAL
   |             |--PROCLF
   |             |--PROCR
   |             |--PROCT
   |             |--PROCX
   |             └--TR.USB
   |--RECYCLE
   |--REPEAT-----PRTSKP----+--BLDBRF-----SP.CLA
   |                       |--CHKREQ-----TR.QUE
   |                       |--CLRTLK
   |                       |--DECRM
   |                       |--INCREQ
   |                       |--RTNQUE----+--TR.DQE
   |                       |            └--TR.QUE
   |                       |--SEND-------(see EXIT)
   |                       |--SPEAK------(see DAP)
   |                       |--TSTRCV----+--BLDBRF------SP.CLA
   |                       |            └--SEND
   |                       └--TR.QUE
   |--SKIP-------PRTSKP------(see REPEAT)
   |--STOP-------TR.MOD
   |--TIMOUU-----+--TR.USB
   |             └--SIGNAL
   |--TOGO-------+--MRKTIM------(see EXIT)
   |             |--PUTTR
   |             |--RTNQUE
   |             |--SPEAK-------(see DAP)
   |             |--SP.DIS------COMMON
   |             |--TR.DQE
   |             └--TR.QUE
   |--TR.SIG
   └--YES
```

FIGURE 4.1. 11/34 SOFTWARE SUBROUTINE TREE (continued)

e. 2 asynchronous line units

f. 1 20-channel Votrax MC-I

g. 1 TCU-100 Timing Control Unit

- <u>Software</u> -

RT-11 V03 XM generated for use with the specified disk.

- <u>Data Bases</u> -

DIRECT.DVF - this file (5000 blocks long) contains all utterances spoken by the system. It is created using the ADPCM encoder and programs VEDIT and RECORD (see Reference 6, Chapter 8).

VRDATA.DAT - this file (1000 blocks long) is created by the VRS software and contains all statistics data generated in system operations.

5. VRS MAINTENANCE--11/70

For the discussion of 11/70 maintenance procedures, the reader should be familiar with FORTRAN-IV PLUS and MACRO-11 programming languages under the RSX-11D monitor and with the RSX-11D utilities, special subroutines, overlay capabilities, event flags, priority levels, and asynchronous system traps.

5.1 TASK CREATION CONVENTIONS

The RSX-11D command file capability is used to assemble, compile, taskbuild, and install or remove most tasks. The command files are named AAABBB.CMD, where AAA is the task name abbreviation (e.g., VRS) and BBB is LST if a compiling command file, INS if an installing command file and REM if a removing command file. BBB is omitted if the command file is for taskbuilding. For example, if a task were to be built from the FORTRAN source file VRS.FTN, the procedures would be as follows:

- o MCR F4P @VRSLST - to compile, then
- o MCR TKB @VRS - to taskbuild.

If VRS.CMD used the TKB overlay switch an overlay definition file must exist and would be named VRS.ODL.

The command files are written to create object files the same name as the source file and to create nonspooled compiler listings on disk.

5.2 SOFTWARE CONVENTIONS

The following items are miscellaneous practices in the 11/70 VRS software. The 11/70 program written in MACRO-11 are DICT, RETREV, VRSTIM, and VRSGLB. These programs require the special capabilities available only with MACRO-11, such as the asynchronous system traps. The rest were written in FORTRAN-IV PLUS: VRINIT, VRS, VRSOUT, VRSFD, FDRTRV, VRSTOP, UDFPRG, and ERRCRT.

Many of the subroutines of the FORTRAN programs reference by means of an INCLUDE statement the file VRPARAM.FTN which contains ubiquitous VRS parameters in common. The parameters are:

- ITI - Terminal logical unit number
- LPU - Line printer logical unit number
- LUNERR - ERR.DAT logical unit number
- LUNKCW - KCW.DAT logical unit number
- LUNUDF - UDF.DAT logical unit number
- LUNHIS - SFI.DAT logical unit number
- MAXIN - Raw weather report buffer size (from KCW.DAT)
- MAXOUT - Processed weather buffer size (to UDF.DAT)
- ISLOTS - Location Index Table size in blocks
- IESTEDT - EST or EDT time indicator.

The VRS software makes use of the RSX-11D special subroutines to handle inter-task communications. A variable number of parameters pertinent to the transaction are transmitted using VSNDRR and responses received using VRECRR.

All disk files are referenced within the software as residing on disk structure DB7. An assignment can be made with the RSX-11D monitor that would define DB7 as being any other single disk structure.

Task priorities are fine-tuned through experience with the system, but in general it can be said that the device handlers must run under the highest priority used and that RETREV and FDRTREV must run at a higher priority than the VRS processor to insure good response time.

5.3 SUPPORT SOFTWARE TASK CREATION

The programs used to create and initialize data base files and perform other auxiliary functions are discussed in Section 3.0. This section will discuss how to create the executable file for each.

5.3.1 UDFPRG

The Universal Data File, UDF.DAT, is created with UDFPRG which requires as input the file NLC.DAT containing the identifying data for each weather reporting station and airport. UDFPRG is comprised of five source files: UDFPRG, BLCR8, IOBLCK, VRSLIB, and NOMESG. They are compiled and listed using the command file UDFLST.CMD and taskbuilt using UDFPRG.CMD.

5.3.2 ERRCRT

Raw weather reports containing format errors are sent to the file ERR.DAT which is created using program ERRCRT. ERRCRT is contained on a single source file, ERRCRT.FTN, and so compile command file is used. The compiler command line is as follows:

- MCR F4P ERRCRT, ERRCRT 1-SP = ERRCRT.
- For taskbuilding, the command file ERRCRT.CMD is used.

5.3.3 VRSGLB

A VRS global common area is created with VRSGLB. The source file, VRSGLB.MAC, is assembled using the MACRO Command File GLBLST.CMD. Taskbuilding is accomplished when the DICT module is taskbuilt with DICT.CMD.

5.3.4 VRINIT

SFI.DAT is a file containing the KCW.DAT pointers existing at the time VRS last processed the raw weather reports. When SFI.DAT and the KCW pointers no longer match, VRS knows new reports have been entered. SFI.DAT is created or initialized by a subroutine of VRINIT, VRSPTR. VRINIT also initializes the map array in the GCA.

VRINIT is comprised of 6 source files: VRINIT, VRSMAP, ZULUTIM, DTELAP, EXTHED, and VRSLIB. They are compiled using VRINLST.CMD and taskbuilt using VRINIT.CMD.

5.3.5 VRSTOP

The only safe way to stop the 11/70 VRS executive is to run VRSTOP, which insures that the UDF block usage control array will be in order. Any other method such as ABORT or a system crash will require running VRINIT before execution could be resumed. The F4P command lines needed to compile the VRSTOP modules are as follows:

- MCR F4P VRSTOP=VRSTOP
- MCR F4P VRSLIB=VRSLIB

The TKB command file, VRSTOP.CMD is used for taskbuilding.

5.3.6 NLCUPD

An editor is required to modify and add to NLC.DAT, the file containing the weather reporting station identification data. NLCVPD is compiled as follows:

MCR F4P NLCVPD=NLCVPD.

Taskbuilding is done with TKB command file NLC.CMD.

5.4 VRS WEATHER PROCESSOR

The VRS Processor executive is an overlaid task with the tree structure shown in Figure 5-1. The VRS root contains the only MACRO-11 routine for the task, VRSTIM.MAC. The second level of overlays constitute the primary VRS functions:

- OPEND opens and closes files and check subfile pointers for KCW.DAT, SFI.DAT, and ERR.DAT.

- SA is the surface observations processor.

- SARMK is the surface observations remarks processor.

- FT is the Terminal Forecast processor.

- ERR is the erroneous report handler.

The names given are those used in the Overlay Definition Files.

Five other tasks also called by the VRS processor executive, differ from the above in that they are independently executing programs, not just subroutines of VREXEC.

1. VRSFD is the Winds Aloft processor. The compiler command lines are as follows:

   - MCR F4P VRSFD=VRSFD
   - MCR F4P VRSLIB=VRSLIB.

Taskbuilding and installation are accomplished with the command files VRSFD.CMD and FRSINS.CMD, respectively.

2. VRSOUT, the VRS I/O task, is comprised of eight source modules which are compiled by means of the F4P command file

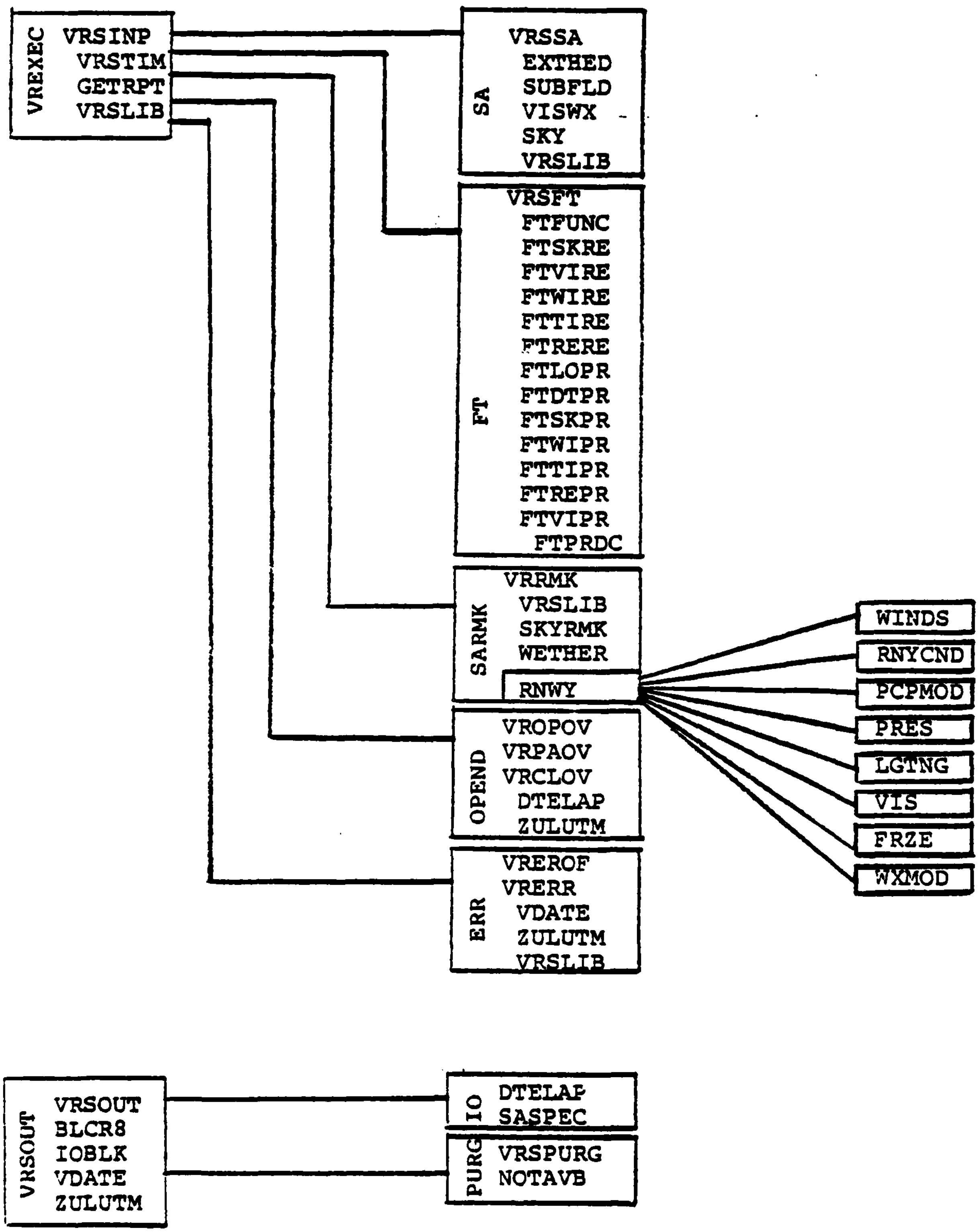


FIGURE 5-1: PDP-11/70® VRS Task/Overlay/Subroutine Tree

VRSLST.CMD. Taskbuilding is done with VRSOUT.CMD and the overlay definition file VRSOUT.ODL. Installation is done with VRSINS.CMD.

3. DICT, the module that translates raw weather reports to dictionary pointers, is comprised of the two modules DICT.MAC and VOCAB.MAC (Plus assembly contents contained on PREFIX.MAC) which as assembled with the following MACRO command lines:

- MCR MAC DICT = PREFIX, DICT
- MCR MAC VOCAB = PREFIX, VOCAB.

Taskbuilding is done with TKB command file DICT.CMD and installation with FRSINS.MD.

4. RETREV, the VRS weather data retrieval program, is comprised of 10 MACRO source files which are assembled with MACRO command file RETASM.CMD. To taskbuild, RETREV.CMD is used. See Figure 5-2.

5. FDRTREV, which calculates Winds Aloft data, consists of 5 source files compiled with F4P command file FDRLST.CMD. Taskbuilding is done with FDRTRV.CMD. Installation is done with VRSINS.CMD. See Figure 5-3.

5.5 PERIODIC SOFTWARE CHANGES

The PDP-11/70® system time is set to Eastern Standard or Eastern Daylight Time. VRS, however, runs under Greenwich Mean Time and three routines must be changed biannually: RETVER.MAC, a subroutine of RETREV, DTELAP.FTN, and ZULUTM.MAC, subroutines of VRSOUT. The changes to the FORTRAN programs DTELAP, and ZULUTM may be made to a change to include parameter IESTEDT.

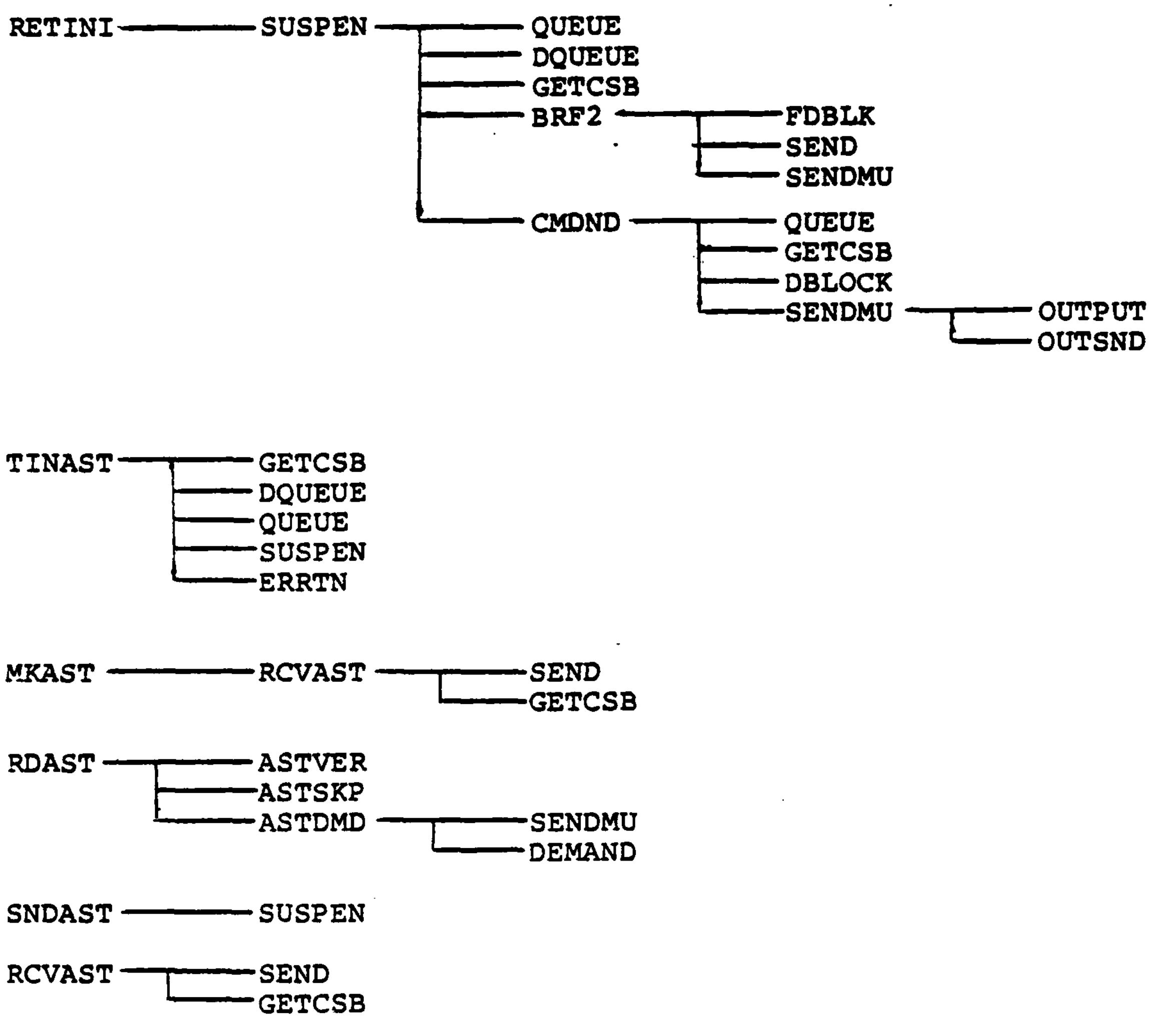


FIGURE 5-2: RETREV Subroutine Tree

- FDRTRV
  - RECEV
    - VRECSP
    - ERRPRC
  - SUMMIT
    - IOBLCK
    - WTFOR3
  - ACTIV
    - VSNDRR
    - EXIT
    - RECEV
      - VRECSP
      - ERRPRC
  - R5ØASC
  - IDATE
  - TIME
  - IOBLCK
  - BLCR8
  - VRECEX

- VRSFD
  - RECEV
    - VRECSP
    - ERRPRC
  - GTRPT
    - BLOCK
    - ALTSTR
  - EXTSTR
  - IDATE
  - IOBLCK

FIGURE 5-3: FDRTRV and VRSFD Subroutine Tree

6. OPERATIONS MANUAL

The following is a summary of steps required to start up and shut down the VRS system:

- Start Up 11/70 Subsystem
  - a. Log On Terminal
  - b. Bring Up Subsystem
- Start Up 11/34 Subsystem
  - a. Power Up System
  - b. Boot 11/34
  - c. Bring Up Subsystems
- "Abort RETREV" Line Clean Up
- Shut down 11/70 Subsystem
- Shut down 11/34 Subsystem
- "Barge In" On
- "Barge In" Off
- System Test.

Details of these procedures are given next in this section. If there is a problem, refer to Figure 6-1 which outlines in flow-chart form procedures for handling problems.

6.1 START UP 11/70 SUBSYSTEM

6.1.1 Log-on Terminal

Enter on the Terminal:

CTRL/Z

CTRL/C

MCR HEL [300,100][CR]

PASSWORD (password)[CR]

MCR

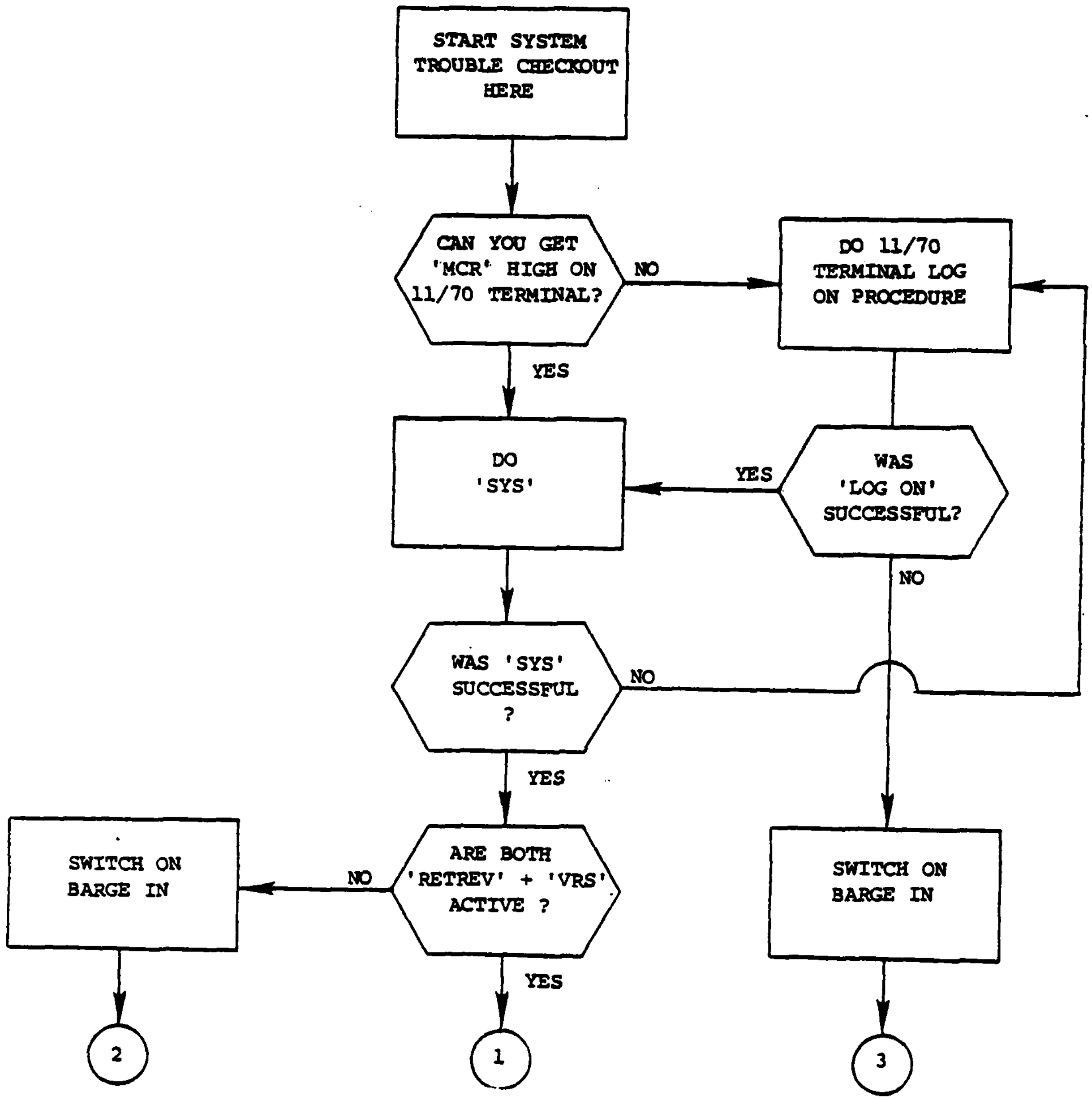


FIGURE 6-1: VRS System Trouble Chart

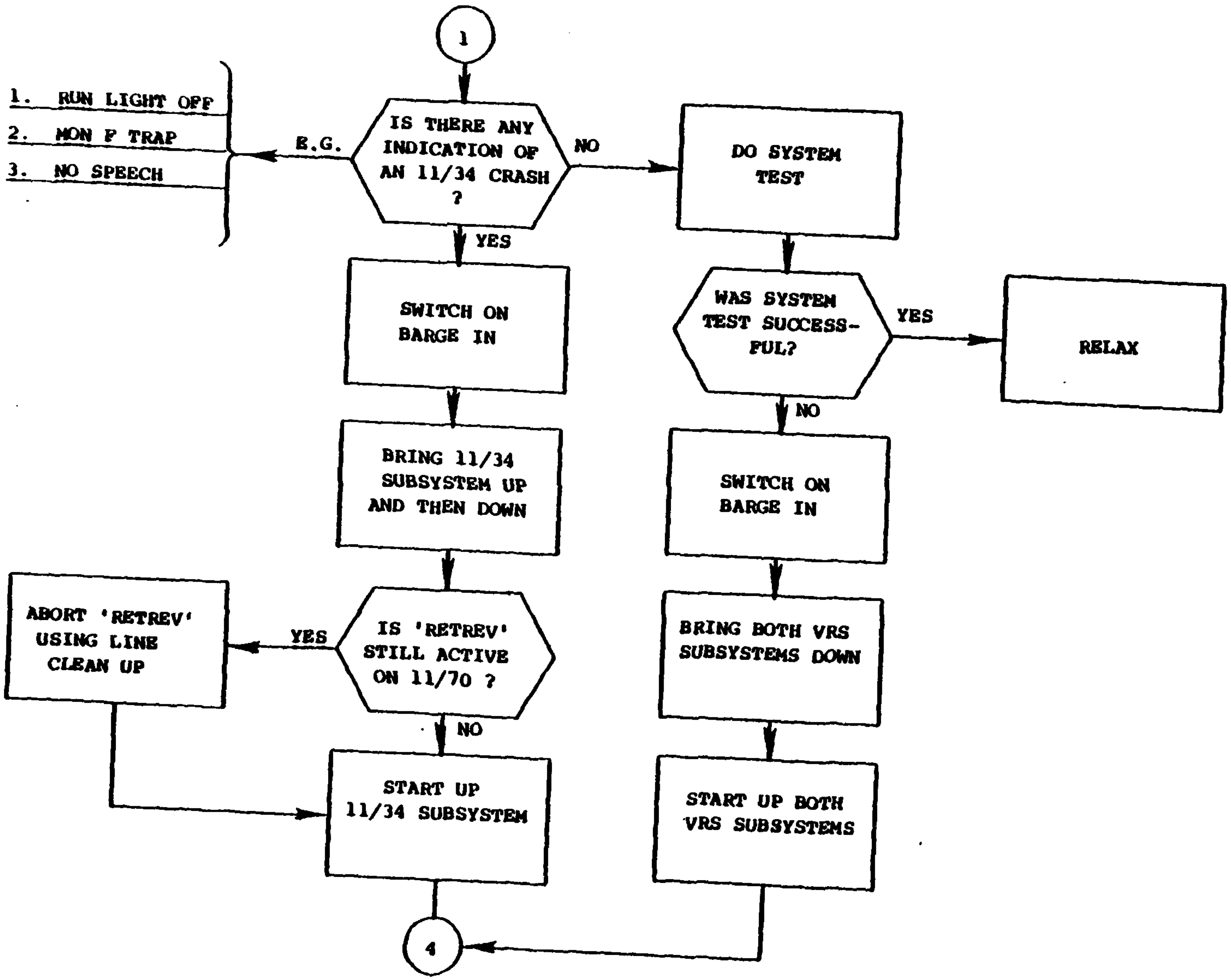


FIGURE 6-1: VRS System Trouble Chart (Cont'd.)

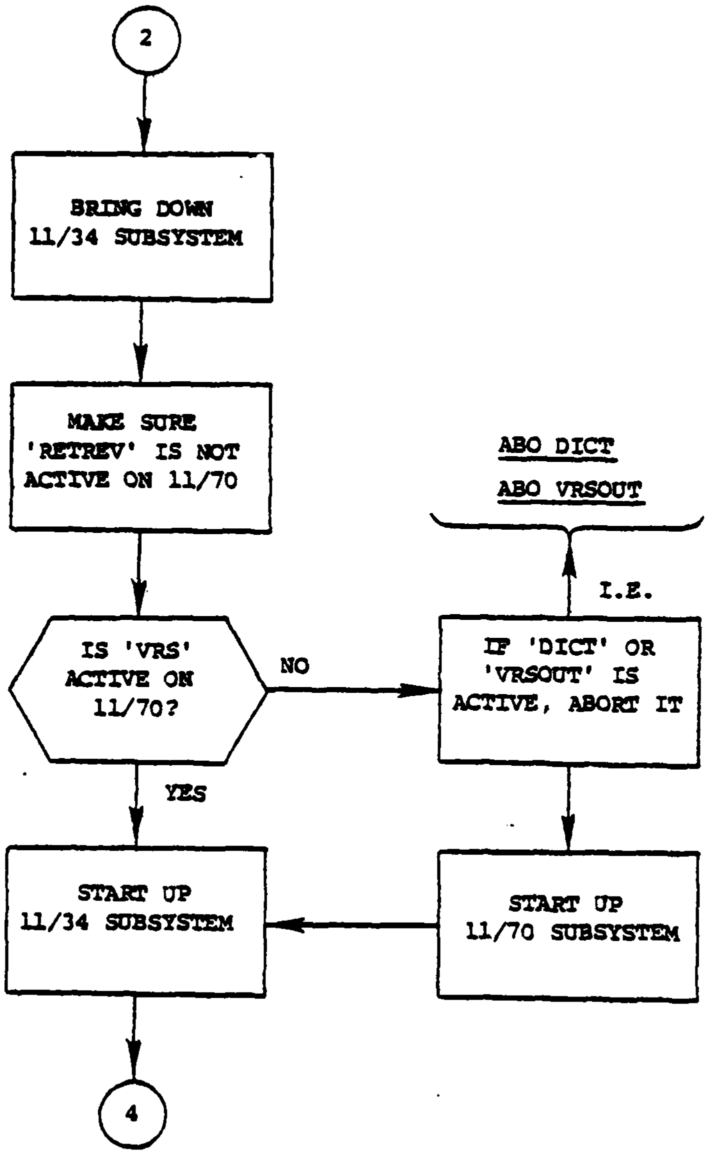


FIGURE 6-1: VRS System Trouble Chart (Cont'd.)

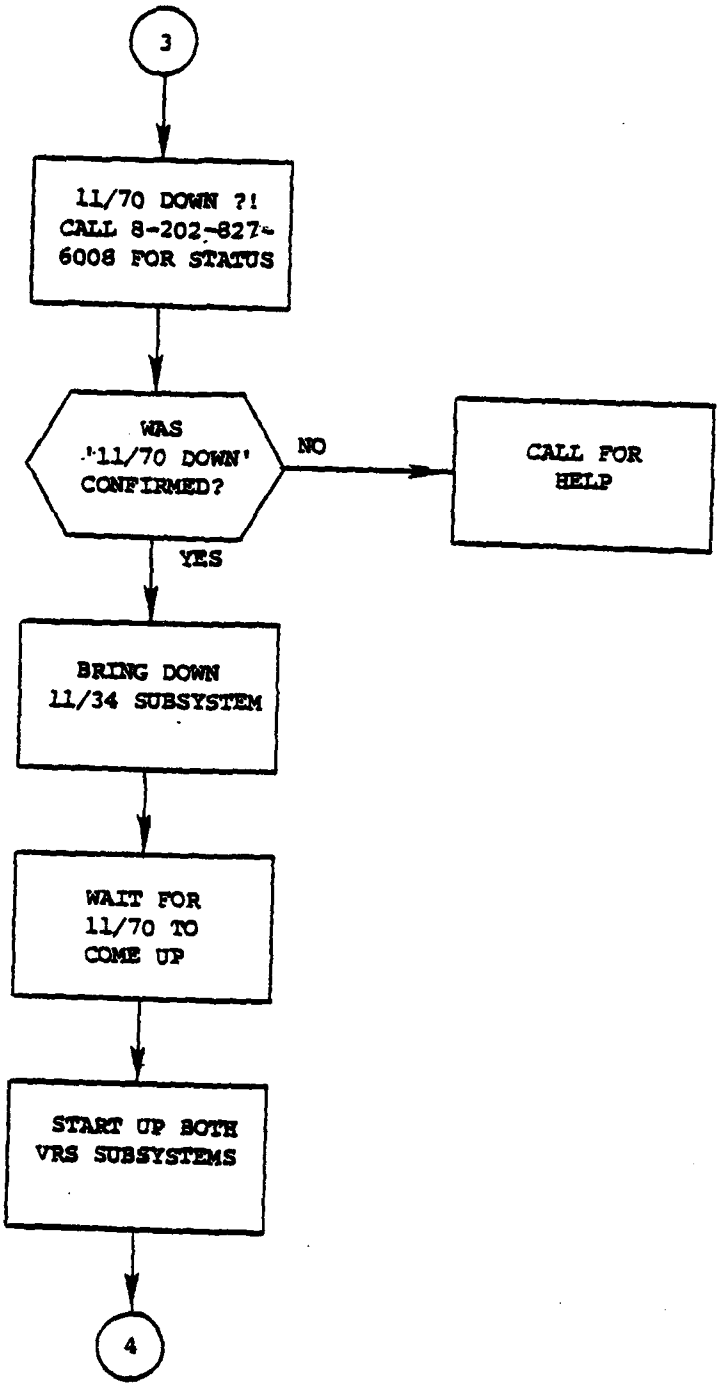


FIGURE 6-1: VRS System Trouble Chart (Cont'd.)

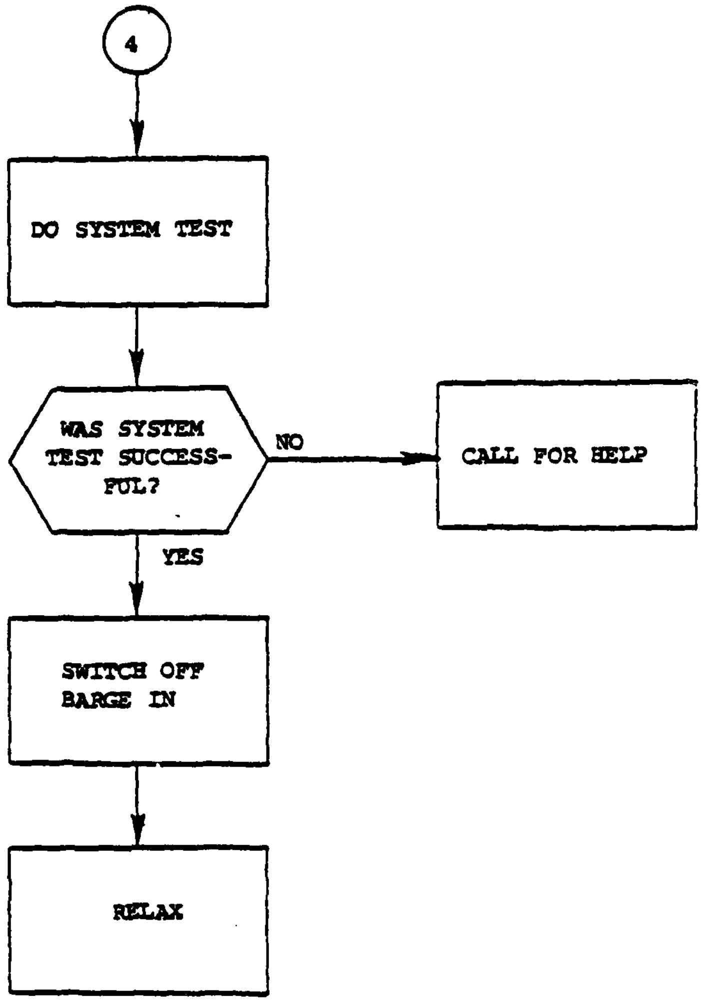


FIGURE 6-1: VRS System Trouble Chart (Cont'd.)

6.1.2 Bring Up Subsystem

6.1.2.1 Initial Procedure

```
MCR  RUN DB7:VRINIT[ESC]
     INITIALIZE VRS - START HH:MM:SS EST
     CALLING VRSMAP
     CALLING VRSPTR
     INITIALIZATION COMPLETE:  HH:MM:SS EST
     CTRL/C

MCR  RUN DB7:VRS[ESC]

     DD-MMM-YY VRXEC HAS RESTARTED HH:MM:SS EST
     AT 1 HH:MM:SS EST
     etc.
```

6.1.2.2 Recovery Procedure

```
MCR  RUN DB7:RECOVER[ESC]
     RECOVER VRS - START; HH:MM:SS EST
     CALLING VRSMAP
     VRS RECOVER COMPLETE:  HH:MM:SS EST

CTRL/C

MCR  RUN DB7:VRS[ESC] etc.
```

6.1.3 Start Up 11/34 Subsystem

6.1.3.1 Power Up System

a) 11/34 Computer
   Switch to DC ON

b) Teleterm
   Set switches: LOCAL #0-, ON

c) Upper two VOTRAX units
   Switch ON.

6.1.3.2 Boot 11/34

6.1.3.2.1. From Fixed Head Disk

Depress panel buttons: CTRL/HALT, CTRL/BOOT
Should print 4 octal numbers on terminal)

```
$L 177462[CR]
$D 177400[CR]
$L 177460[CR]
$D 5[CR]
$L 0[CR]
$S[CR]

.RT-11XMV03-02
.INS MC,AD,LI,LO
.LOA MC,AD,LI,LO,DP
```

```
.D 56=2012
.DATE DD-MMM-YY[CR]
.TIME HH:MM:SS(CR]          (GMT)
.DATE[CR]                   (Verification)
.TIME[CR]                   (Verification).
```

6.1.3.2.2 From CDC Backup Disk

Depress panel buttons: CTRL/HALR, CTRL/BOOT$L 1000[CR]
(Should print 4 octal numbers on terminal)

```
$L 1000[CR]
$D 12700[CR]
$D 176712[CR]
$D 12760[CR]
$D 1[CR]
$D 12 [CR]
$D 105760[CR]
$D 12[CR]
$D 100375[CR]
$D 5040[CR]
$D 5040 [CR]
$D 5040 [CR]
$D 12740[CR]
$D 400[CR]
$D 12740[CR]
$D 5[CR]
$D 105710[CR]
$D 100376[CR]
$D 5007 [CR]
$L 1000[CR]
$S [CR]
.RT-11XMV03
.INS MC,RF,AD,LI,LO
```

.LOA MC,AD,LI,RF,LO

.D 56=2012

.TIME HH:MM:SS[CR] (GMT)

.DATE[CR] (Verification)

.TIME[CR] Verification).

6.1.3.3 Bring Up Subsystem

6.1.3.3.1 Initial Procedure

```
. DEL VRDATA.DAT[CR]
FILES DELETED :
DK:VRDATA.DAT ? Y[CR]
.R VRS[CR]
VRS VERSION-03X-00
     (If the remaining print out does not appear as listed
     below, enter "EXIT[CR]" on the 11/34 terminal and try
     "R VRS[CR]" again.)
MCR
MCR HEL [300,100]
PASSWORD
MCR RUN RETREV $
INITIALIZATION COMPLETE
     (At this point, do a "SYS" command on the 11/70
     terminal anc check that "RETREV" is running.)
```

6.1.3.3.2 Recovery Procedure

Same as above (i.e., Section 6.1.2.3.1) except do not delete VRDATA.DAT file.

6.1.3.4 Console Commands

There are six console commands available to the operator which affect the operation of VRS on a particular channel. The commands are typed on the VRS console in the following format:

.CnnX cr where

nn is the two digit channel specifier (single digit channels must be preceded by a zero) and X is the command letter identifier as listed below.

6.1.3.4.1 CnnN

The command turns off the trace function on the channel nn.

6.1.3.4.2 CnnT

This command allows the trace functions to be performed for the channel nn.

6.1.3.4.3 CnnD

This command disables the channel nn; that is, no calls will be received on that line.

6.1.3.4.4 CnnR

This command re-enables the channel nn; that is, a channel that has been disabled will now be able to receive calls.

6.1.3.4.5 CnnX

This command de-activates the fifteen-minute time-out on the line nn.

6.1.3.4.6 CnnA

This command activates the fifteen-minute time-out on the line nn.

6.1.4 <u>Shut Down 11/70 Subsystem</u>

Type the following in the 11/70 terminal:

CRTL/Z
CRTL/C

<u>MCR</u> RUN VRSTOP[ESC]

<u>****VRS EXEC TERMINATING</u>
<u>VRS--STOP</u>

(NOTE: It may take up to 5 minutes to obtain the last line.)

6.1.5 <u>Shut Down 11/34 Subsystem</u>

6.1.5.1 Temporary Procedure

Enter the following on the 11/34 terminal:

_EXIT[CR]

.

(All the channel lights should go out.)

6.1.5.2 Final Procedure

```
_EXIT[CR]
.COPY VRDATA.DAT DP:TRmmdd.yyV[CR]
.DIR *.yyV[CR]
.DEL VRDATA.DAT[CR]
FILES DELETED:
DK:VRDATA.DAT ? Y[CR]
```

The intention is to save the trace file on the CDC disk under the file name TRmmdd.yyV where "mm" is the number of the month, "dd" is the day of the month, and "yy" is the year. It is suggested that these trace files be periodically archived to magnetic tape.

6.1.6 <u>"Barge In" On</u>

1. Set switch on "barge in" phone to activate the message of interest, i.e., either the "temporary down" or "overnight" message.

2. Switch on the "barge in" to activate the "barge in" unit.

3. Call 8-202-347-3222 to check the "barge in" message.

6.1.7 "Barge In" Off

1. Switch off "barge in".

2. Call 8-202-347-3222 to check on system response.

6.1.8 System Test

1. Call into system on a local line.

2. Enter "DCA" loc ID and check out all the weather products.

6.1.9 System Trouble Chart

The intention of this section is to direct the operator to the appropriate action that should be taken for various system malfunctions.

7. USERS' MANUAL

Any public, business, or home telephone with a 12-key signalling system can be used to access the system. The conventional rotary dial telephone may be used only for dialing the access numbers, however, an acoustically-coupled tone signalling device (in lieu of a Touch-Tone® telephone) can be employed in conjunction with the rotary dial telephone to enter the information requests.

7.1 ENTERING DATA

To communicate with the computer you must use the keypad in a way that the computer "understands." Locations (weather reporting stations and airports) are uniquely identified by three-letter combinations and you enter these three-letter identifiers to delineate a single location or a series of locations (e.g., a proposed flightpath) for which you desire to know the weather.

The keypad does not have enough keys to allow the entry of an alphabetic character (letter) with a single keystroke. But it is possible to make an unambiguous entry by depressing two keys. You can enter a particular letter by depressing the key on which that letter appears and another key to indicate which of the three letters, 1st, 2nd, or 3rd. The numeral "1" key indicates the 1st letter, the numeral "2" key indicates the 2nd and the numeral "3" key indicates the 3rd. Thus the letter B is signalled by depressing the key on which B appears (the number "2" key) and then the numeral "2" key (2nd letter in the group, ABC). The letter C is signalled by depressing the key on which "C" appears and the numeral "3" key (3rd letter in group ABC). For example, DCA is entered as D-1, C-3, A-1, as shown below.

D

1	ABC 2	DEF 3
GHI 4	JKL 5	MNO 6
PRS 7	TUV 8	WXY 9
*	OPER 0	#

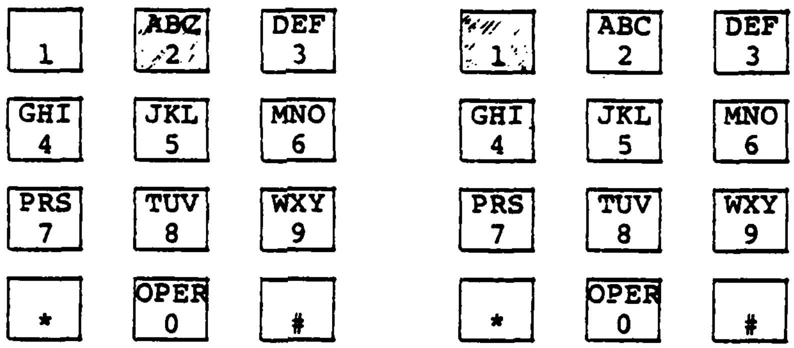
1	ABC 2	DEF 3
GHI 4	JKL 5	MNO 6
PRS 7	TUV 8	WXY 9
*	OPER 0	#

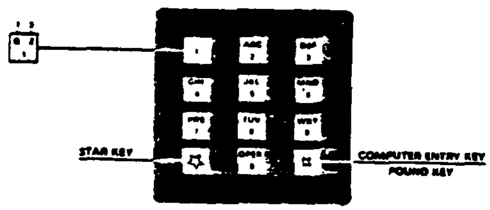
C

1	ABC 2	DEF 3
GHI 4	JKL 5	MNO 6
PRS 7	TUV 8	WXY 9
*	OPER 0	#

1	ABC 2	DEF 3
GHI 4	JKL 5	MNO 6
PRS 7	TUV 8	WXY 9
*	OPER 0	#

A





As shown above, the letters Q and Z and the blank character are assigned to the numeral "1" key. Q is 1-1, "Blank" is 1-2 and Z is 1-3. Each of the twenty-six letters of the alphabet can be entered in this fashion (two keystrokes) and no confusion will result. The 'blank' is not used.

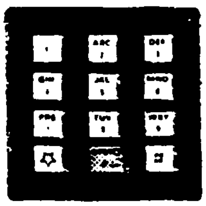
NOTE: In addition to the 1- 2- 3- keys for second keystroke denoting the letter position, left-middle-right keys of the same row may also be used for a faster keystroke. For example, the letter 'S' is contained on key seven as shown.

PRS	TUV	WXY
7	8	9

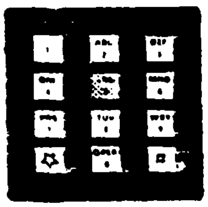
The user may use the keystrokes 7-9 to denote 'S' since 'S' is on key seven in the right position thus 7-9 may be used instead of 7-3. However, the left, middle, or right second keystroke must be in the same row.

It does not suffice just to be able to communicate a string of letters of the alphabet to the computer. You must be able to tell the computer what you want done with the information you have provided. At the lower right-hand corner of the keypad, there is a key imprinted with a "#" symbol. We call this the 'computer entry' key or, for conciseness, the 'pound' key. Since this key is not used to transmit letters or numbers, it creates no confusion to employ it as a control key to signal an action or a request. Used in conjunction with other keys, a number of different actions can be signalled. Other control functions will be explained later.

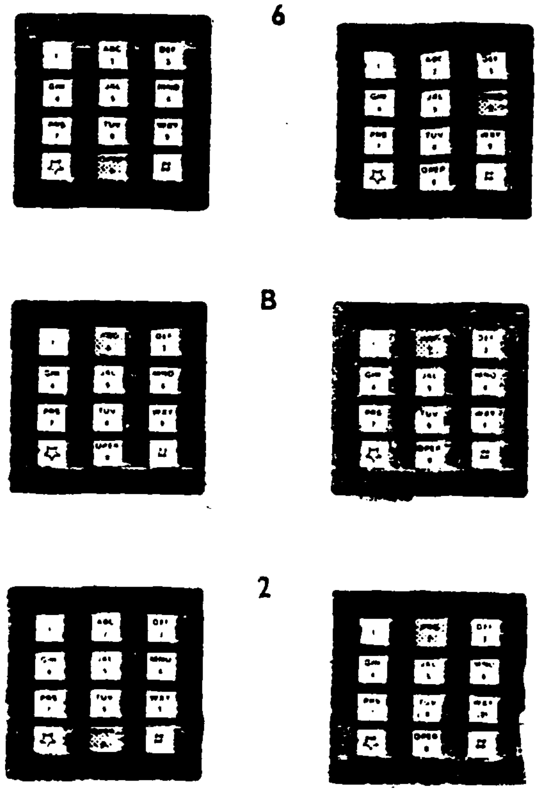
Some location identifiers use both letter and numerals. For these entries, it is necessary to use two keystrokes for each letter or numeral. The context of the pilot-computer dialogue will often preclude ambiguities and permit simpler data entry. Numbers can be entered unambiguously by depressing the 'OPER' key and the appropriate numeral key. The 'OPER' key is the key representing the numeral '0' (or zero) so that entry of the numeral '0' involves two actuations of the 'OPER' key. The numeral '5' is communicated by depressing 'OPER' and '5' (as shown below) and the other numerals are similarly communicated.



5



The procedure described is used only for entering numbers in three-letter location identifiers with mixed letters and numbers. For all other numeric entries, single keystrokes for numbers are required. For example, if the computer 'voice' requests an altitude or a number of hours (from the present time), then the numeric entries for these fields may be made via a single keystroke for each digit of the entry.



7.2 DATA NOT AVAILABLE

When data are not available, one of the following will occur:

- Wrong Identifier - If a three-character entry which does not constitute a valid location identifier is made (e.g., ABC), the VRS will read back the characters as entered. However, when the report requested is to be read out, the VRS will say "ALPHA-BRAVO-CHARLIE... is not a location identifier."

- No Report for a Given Location - If the location identifier is a valid one but not a reporting station for the type of report requested, the VRS will say "ALPHA-BRAVO-CHARLIE... is not an Hourly Observation Station" or "... is not a Terminal Forecast location."

- Noncurrent Data - If the location identifier is a valid one but the current data are not available, the VRS will say (e.g., SBY), "SIERRA-BRAVO-YANKEE... report not available" for report type requested.

NOTE:

- Hourly Observations: Only the latest available observation will be given provided that the observation is not more than 2 hours old. Special observations will be appended to last hourly.

- In this system all reporting stations for weather observations within the continental United States are contained in the data base.

- Minimum altitude for forecasted Winds Aloft is approximately 2,000 feet above terrain level.

- The system has some time-out functions which limit the amount of time an individual can use the system. This feature has been incorporated to preclude an individual from tying up the phone lines for an extended period.

The computer must be able to recognize the end of an entry (i.e., a string of alphabetic, numeric or mixed characters) and the request that it respond. The computer entry key ('#' key) is depressed twice to provide the end-of-entry signal immediately following each and every field. Thus, to request weather data for Martinsburg, W. Va. (and vicinity) the keystroke sequence 'M-1', 'R-2', 'B-2', '#''#' is generated .

The computer will 'read back' each item entered so that the correctness of the entry may be verified . The phonetic alphabet will generally be used so that the identifier MIV will be read back as "MIKE" "INDIA" "VICTOR"; CHO will be read back as "CHARLIE" "HOTEL" "OSCAR". For some locations, the actual name of the airport will be read back. For example, DCA (Washington National Airport) will be read back as "Washington National."

7.3 CONTROL FUNCTIONS

The use of the '#' key was discussed previously in section 7.2. The '*' (STAR) key is used to stop the computer response. While in the response mode, if it is necessary to interrupt the computer voice response, depress the '*' key. This will halt the voice response until the operator is ready to proceed. The operator may then order a resumption of voice response, a repeat, a jump ahead (skip) or a begin over, by selecting the appropriate keystroke sequence shown below. Notice that the enter command '#'-'#' is not required after the control functions containing the '*' (STAR) keystroke.

ENTER	# #	REPEAT	* 7
YES	9 # #	JUMP AHEAD	* 5

NO______________	6 # #	DELETE__________	* 3
STOP______________	*	BEGIN OVER________	* 2
GO______________	* 4		

NOTE: "YES" or "NO" may be entered with a single pound sign.

7.4 EXAMPLE OF TYPICAL VRS DIALOGUE

PILOT - pilot dials.

SYSTEM - "HELLO", Greenwich Time is XXXX."

SYSTEM - "Enter Location Identifier."

PILOT - (Desired location - PIT) P-1; I-3; T-1; # #

SYSTEM - "PAPA", "INDIA", "TANGO" "ENTER NEXT LOCATION"

PILOT (Desired location - ILG) I-3, L-3, G-1; # #

SYSTEM - "INDIA", "LIMA", "GOLF" "ENTER NEXT LOCATION"

PILOT (If no additional entries, enter ##)

SYSTEM - "DO you want hourly surface observations? Answer yes or no."

PILOT - Y; # #

SYSTEM - reads hourlys for PIT, ILG, etc.

SYSTEM - "Do you want terminal forecasts? Answer yes or no"

PILOT - Y; # #

SYSTEM - reads forecasts for PIT and ILG

SYSTEM - "Do you want winds aloft forecasts? Answer yes or no."

PILOT - Y; # #

SYSTEM - "How many hours from now? The maximum is 30 hours.

PILOT - 6; # #

SYSTEM - "six"

SYSTEM - "At what altitude?"

PILOT - 85; (or 8500; no matter) # #

SYSTEM - "eight five"

SYSTEM - reads winds aloft at requested altitude, +4000 feet and -4000 feet for each location.

SYSTEM - "Do you need more information? Answer yes or no."

PILOT - Y; # #

SYSTEM - "Enter location identifier, etc."

8. REFERENCES

1. "PDP-11 Peripherals Handbook,"© 1975, Digital Equipment Corp., Maynard MA.

2. "Ten-Channel Voice Response System, Systems Design Report," Unpublished material on file at DOT/TSC. June 1977.

3. Bell System Technical Reference-Data Set 201C Interface Specification. Nov. 1973, AT&T NY, NY.

4. "Bell System Data Communications - Technical Reference - Data Set 407A - Interface Specifications," Nov. 1973. AT&T NY, NY PUB41408.

5. "RT-11 Software Support Manual," DEC Order No. DEC-11-ORPGA-B-D© 1973, 1975, Digital Equipment Corp., Maynard MA.

6. "Single-Channel Voice Response System Program Documentation, Final Report," FAA-RD-77-177, Vols 1-3, Dec. 1977.

7. "Ten-Channel VRS Processor Design Report (SA, SA Remarks, FT, FD)," Unpublished material on file at DOT/TSC., Nov. 1977.

8. "Design Document for the Data Edit Position Software," Unpublished material on file at DOT/TSC, Aug. 1977.

9. RT-11 Advanced Programmers Guide,© 1977, DEC Order No. AA-5280B-TC, Digital Equipment Corp., Maynard MA.

APPENDIX A

PDP-11/34® and PDP-11/70® Software Module Descriptions

A.1 PDP-11/34® VRS

MODULE NAME: ADX.SYS

PROGRAM: 11/34 VRS

SOURCE FILE: ADX.MAC

PURPOSE: ADPCM output device driver for 20 channels

CALLING ROUTINES:

CALLING SEQUENCE: Called by a WRITE request in speak module QUEUE. QUEUE pointers are arranged by a trap call which executes some code in trap handler, then jumps to subroutine in handler which links QUEUE pointers.

COMMON: ADCQE
ADLQE

SUBROUTINES CALLED:

DQUEUE - DE-QUEUE an element
OFF - take element off ADX QUEUE list
EQUEUE- QUEUES an element
PUT - put element onto ADX QUEUE list
SETRPT - turn on interrupts

FUNCTION DESCRIPTION:

Output: Upon - WRITE request:
1. DEQUEUES FROM RT-11 QUEUE
2. QUEUES internally one-QUEUE per channel
3. Initiates NPR output

On completion of ADPCM write:
1. DEQUEUES from internal QUEUE
2. Transfers element back to RT-11 QUEUE
3. Requests write completion on ADPCM.

COMMENTS: This driver handles data synchonously for each user by maintaining a separate output queue for each user. When a write request is issued, the element is removed (unlinked) from the RT-11 queue and held until completion of the write (speech), when it gets re-linked to RE-11 queue. Therefore, RT-11 never sees more than 1 write on the channel at any point in time.

MODULE NAME:	LIX.SYS
PROGRAM:	11/34 VRS
SOURCE FILE:	LIX.MAC
PURPOSE:	Input driver for communication between 11/70 and 11/34 by serial line
CALLING ROUTINES:	
CALLING SEQUENCE:	Called by .READC in background routine during INIT Called by .READC in send/receive when communicating
COMMON:	LICQE: LILQE:
SUBROUTINES CALLED:	$INPTR Monitor CUR's $PUTBYT
FUNCTION DESCRIPTION:	Input: Receives characters from 11/70 and stores them in user buffer space associated with channel to which data applies. <CR> is treated as an end-of-file.
COMMENTS:	At initialization time, a series of 10 .READC requests are issued for synchronization.

MODULE NAME:	LOX.SYS
PROGRAM:	11/34 VRS
SOURCE FILE:	LOX.MAC
PURPOSE:	SLU device driver for output side of 11/34 to 11/70 communication
CALLING ROUTINES:	
CALLING SEQUENCE:	Called by WRITE in BACKGROUND module Called by WRITE in SEND/RECEIVE module
COMMON:	LOLQE LOCQE
SUBROUTINES CALLED:	$INPTR RT-11 System Functions $GTBYT
FUNCTION DESCRIPTION:	Output: Functions like a DL-11 Receives characters from user buffer or text string. Transfers one character at a time under interrupt control at priority 4.
COMMENTS:	This driver treats <CR> as an end-of-file.

MODULE NAME: MCX.SYS

PROGRAM: 11/34 VRS

SOURCE FILE: MCX.MAC

PURPOSE: Touch-Tone® input handler for 20 channels

CALLING ROUTINES AND CALLING SEQUENCE:

Output - Called by .WRITE in background. This occurs in response to reception of STATUS CHARS from data set.

Input - Enabled by setting interrupt enable bit (BIS #100, @#175630) after initialization in background routine

COMMON:

MCICQE
MCILQE
MCOCQE
MCOLQE

SUBROUTINES CALLED:

DEFUSB - Define user status block
LVMCON - input character decoder
SIGNAL - signal significant event

FUNCTION DESCRIPTION:

Input:
1. Accept chars from VOTRAX unit, check for and remove SYNC CHAR, separate control CHARS from data CHARS, if data numeric, check for legality of numeric data. Convert 2 numbers into a letter. If control or status CHAR, signal the event, if just data, stash in channel buffer

Output:
2. Produces line status changes (answer, hang-up, disconnect)

COMMENTS:

MCX never issues READ completions to RT-11. Instead, it writes the data word directly into the user buffer, then gives a completion signal to the background. Causes interrupt whenever a digit is received.

MODULE NAME:	INITIALIZATION ROUTINES
PROGRAM:	11/34 VRS
SOURCE FILE:	BACKGR.MAC
PURPOSE:	To allocate memory set up I/O QUEUES
CALLING ROUTINES:	This is first routine in VRS. entered thru start address START. This code is executed once only.
CALLING SEQUENCE:	
COMMON:	All TR.*** US.*** SP.*** FL.*** DP.*** Parameters defined by PREFIX.MAC
SUBROUTINES CALLED:	None
FUNCTION DESCRIPTION:	1. Allocates extra QUEUE elements. 2. Allocates space in extended memory for dictionary. 3. Allocates space in extended memory for buffers. 4. Defines extra I/O channels. 5. Prints version ID. 6. Creates USB's one per line. Then continues to dictionary initialization
COMMENTS:	

MODULE NAME:	DICTIONARY INITIALIZATION
PROGRAM:	VRS
SOURCE FILE:	BACKGR.MAC
PURPOSE:	To open channels, read in dictionary and assure proper communication with 11/70
CALLING ROUTINES:	Entry point $FA001 Code is executed once only.
CALLING SEQUENCE:	
COMMON:	User Status block parameters
SUBROUTINES CALLED:	DICT STRTIM TRAP TR.QUE TR.DQE TR.USB

FUNCTION DESCRIPTION:

1. Opens TTy handler.
2. Opens one file per channel for dictionary reads.
3. Reads dictionary directory blocks into core.
4. Starts VRS clock by loading RT-11 time.
5. Assigns I/O channel numbers to ADPCM. devices, Touch-Tone® receiver, 11/70 input, and 11/70 output.
6. Logs into 11/70 RSX system and runs RETREV.
7. Prints initialization complete message.
8. Jumps to BACKGR to await significant events.

COMMENTS: During 11/70 log on, all messages from 11/70 are echoed on TTY.

MODULE NAME: BACKGR

PROGRAM: VRS

SOURCE FILE: BACKGR.MAC

PURPOSE: Polling loop to check for significant events

CALLING ROUTINES: Program returns to this module at completion of any function.

CALLING SEQUENCE: JMP BACKGR

COMMON: Parameters defined by PREFIX.MAC

SUBROUTINES CALLED: TRAP TR.SIG
TRAP TR.USB

FUNCTION DESCRIPTION:

1. Checks BITMSK and BITMSK+@ FOR DEVICES COMPLETIONS. If no completions, continues checking.
2. When completion occurs, determines which channel it was.
3. Uses channel # to determine USB address.
4. Jumps to proper completion routine by vectoring from DONVEC table.

Also prints appropriate error messages upon detection of errors

COMMENTS:

MODULE NAME: DISABL

PROGRAM: 11/34 VRS

SOURCE FILE: BACKGR.MAC

PURPOSE: Disables a channel

CALLING ROUTINES: DAP

CALLING SEQUENCE:
R1 → channel #
R0 → USB ADDR
JSR PC, DISABL

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1. Pushes R0 onto the stack.
2. Puts channel # into .WRITE parameter block DISADW.
3. Does a .WRITE to MCX which puts selected channel out of service.
4. Restores R0 and returns via RTS PC.

COMMENTS:

MODULE NAME: ENABLE

PROGRAM: 11/34 VRS

SOURCE FILE: BACKGR.MAC

PURPOSE: Enables Datasets in use by system.

CALLING ROUTINES: DISCON

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:

1. Pushes R0 onto the stack.
2. Clears the line timeout flag.
3. Puts channel number into .WRITE parameter block ENABDW.
4. Does a .WRITE to MCX, which enables one channel.
5. Restores R0 and returns via RTS PC.

COMMENTS:

MODULE NAME:	NXTCAR
PROGRAM:	11/34
SOURCE FILE:	BACKGR.MAC
PURPOSE:	Routine decodes console commands of the format C NN X where NN is a 2-digit channel number. X is one of the following: N, T, D, R, A, X
CALLING ROUTINES:	This is a read completion routine from TT.
CALLING SEQUENCE:	
COMMON:	TR.VSB TTPAR
SUBROUTINES CALLED:	TRAP TR.USB PROCN PROCT PROCD PROCR PROCA PROCX PROCCR PROCLF
FUNCTION DESCRIPTION:	1. Pushes R2, R3, R4, and R5 onto the stack. 2. Checks for exit command if so, restores registers and exits to NXTEXT. 3. Checks for legal channel number. If OK, resolves USB address; if error, prints message and exits to NXTEXT. 4. Checks for legal character from list at CARCK. Ignores character if not valid. 5. If valid character, vectors to proper servicing routine. All service routines exit thru NXTEXT.
COMMENTS:	

MODULE NAME:	NXTEXT
PROGRAM:	11/34 VRS.
SOURCE FILE:	BACKGR.MAC
PURPOSE:	Exit routine for NXTCAR
CALLING ROUTINES:	NXTCAR PROCR PROCN PROCA PROCT PROCX PROCD PROCLF
CALLING SEQUENCE:	JMP NXTEXT
COMMON:	NXTBUF
SUBROUTINES CALLED:	None
FUNCTION DESCRIPTION:	1. Issues another .READC to TT: 2. Restores saved registers. 3. Exits from completion via RTS PC
COMMENTS:	

MODULE NAME: PROCA

PROGRAM: 11/34 VRS.

SOURCE FILE: BACKGR.MAC

PURPOSE: Turns on line timeout for channel specified if not already on.

CALLING ROUTINES: NXTCAR

CALLING SEQUENCE: JMP @ VCT-2 (R1)

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1. Sets timeout bit in USB.
2. If user on that line, starts a marktime.
3. Exits to NXTEXT.

COMMENTS:

MODULE NAME: PROCCR

PROGRAM: 11/34 VRS.

SOURCE FILE: BACKGR.MAC

PURPOSE: Treats <CR> as a valid character, but ignores it.

CALLING ROUTINES: NXTCAR

CALLING SEQUENCE: JMP @ VECT-2 (R1)

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: 1. Returns immediately to NXTEXT.

COMMENTS:

MODULE NAME: PROCD

PROGRAM: 11/34 VRS.

SOURCE FILE: BACKGR.MAC

PURPOSE: Disconnects user of channel specified and disables line.

CALLING ROUTINES: NXTCAR

CALLING SEQUENCE: JMP @VECT-2 (R1)

COMMON:

SUBROUTINES CALLED: COMMON
TRESET
SIGNAL

FUNCTION DESCRIPTION:

1. Causes a hard hang-up.
2. Clears the USB.
3. Resets the Touch-Tone® line.
4. Signals the event via BITMSK.
5. Exits to NXTEXT.

COMMENTS:

MODULE NAME: PROCLF

PROGRAM: 11/34 VRS

SOURCE FILE: BACKGR.MAC

PURPOSE: Treats <LF> as a valid character but ignores it.

CALLING ROUTINES: NXTCAR

CALLING SEQUENCE: JMP @ VECT-2 (R1)

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: 1. Returns immediately to NXTEXT.

COMMENTS:

MODULE NAME: PROCN

PROGRAM: 11/34 VRS.

SOURCE FILE: BACKGR.MAC

PURPOSE: Turns off trace for channel specified.

CALLING ROUTINES: NXTCAR

CALLING SEQUENCE: JMP @ VECT-2 (R1)

COMMON: All FL.*** US.*** as defined in PREFIX.MAC

SUBROUTINES CALLED: MTCLOS

FUNCTION DESCRIPTION:
1. Turns off trace.
2. Closes trace statistics file.
3. Exit thru NXTEXT.

COMMENTS:

MODULE NAME: PROCR

PROGRAM: 11/34 VRS.

SOURCE FILE: BACKGR.MAC

PURPOSE: Resets and enables data set for channel specified.

CALLING ROUTINES: NXTCAR

CALLING SEQUENCE: JMP @ VECT-2 (R1)

COMMON:

SUBROUTINES CALLED: COMMON
TRESET
ENABLE

FUNCTION DESCRIPTION:
1. Initializes the buffers.
2. Puts a hang-up indicator in status field.
3. Resets channel.
4. Enables the line.
5. Exits to NXTEXT.

COMMENTS:

MODULE NAME: PROCT

PROGRAM: 11/34 VRS.

SOURCE FILE: BACKGR.MAC

PURPOSE: Turns on trace for specified channel

CALLING ROUTINES: NXTCAR

CALLING SEQUENCE: JMP @ VECT-2 (R1)

COMMON: All FL.*** as defined in PREFIX.MAC
US.***

SUBROUTINES CALLED: OPNTR

FUNCTION DESCRIPTION:
1. Sets trace but.
2. Opens trace file.
3. Exits to NXTEX.

COMMENTS:

MODULE NAME: PROCX

PROGRAM: 11/34 VRS.

SOURCE FILE: BACKGR.MAC

PURPOSE: Turns off line timeout for channel specified

CALLING ROUTINES: NXTCAR

CALLING SEQUENCE: JMP @ VECT-2 (R1)

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:

1. If timeout is already disabled, exits immediately.
   ELSE:
2. If timeout is not disabled, timeout by setting a bit in USB. If channel in use, cancels marktime and exits else exits immediately.

COMMENTS:

MODULE NAME: SIGNAL

PROGRAM: 11/34 VRS

SOURCE FILE: BACKGR.MAC

PURPOSE: Given channel number, sets appropriate bit in BITMSK or BITMSK+2.

CALLING ROUTINES: PROCD
MRKTIM TIMOUU
SP.DIS MCX.SYS

CALLING SEQUENCE: JSR PC, SIGNAL

COMMON: US.CHN

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1. Pushes R1, R2, R3 onto the stack.
2. Shifts a 1 into R1 and R2 the same number of places as the channel number.
3. Puts R1 into BITMSK+2 and R2 into BITMSK via BIS instruction.
4. Restores R1, R2, R3, and returns.

COMMENTS:

MODULE NAME: STRTIM

PROGRAM: 11/34 VRS

SOURCE FILE: BACKGR.MAC

PURPOSE: Starts VRS clock

CALLING ROUTINES: DICTIONARY INIT.

CALLING SEQUENCE: JSR PC, STRTIM

COMMON: TIME, TIME +2

SUBROUTINES CALLED: $MLI (Multiply Routine)

FUNCTION DESCRIPTION:

1. Gets GMT from TCU-100.
2. Converts to seconds since midnight.
3. Stores 2-word result in TIME and TIME+2.
4. Issues a 1-second marktime so next event occurs as a completion routine.

COMMENTS:

MODULE NAME: TRESET

PROGRAM: 11/34 VRS

SOURCE FILE: BACKGR.MAC

PURPOSE: Unconditionally resets all Touch-Tone® lines.

CALLING ROUTINES: DISCON PROCD PROCR SP.DIS

CALLING SEQUENCE: JSR PC, TRESET

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1. Pushes R0 onto the stack.
2. Puts channel number into write parameter block TRESDW.
3. Does a .WRITE to MCX which resets all channels.
4. Restores R0, then returns via RTS PC.

COMMENTS:

MODULE NAME: CANCEL

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: Deletes last Touch-Tone® input in response to user command *3

CALLING ROUTINES: BACKGR

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: TRAP TR.MOD
SPEAK
CLRTTK

FUNCTION DESCRIPTION:

1. Ignores command if user in briefing mode or being disconnected.
2. Removes one locid from list if in entry mode.
3. Deletes response if yes/no.
4. Speaks "RE-ENTER" to user.
5. Returns,

COMMENTS:

MODULE NAME: CLRTTK

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Enables Touch-Tone® key-ins for specified channel.

CALLING ROUTINES: CANCEL RPTSKP INVALK SKIP

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: None

FUNCTION SEXCRIPTION:
1. Enables Touch-Tone inputs by setting appropriate bits in USB flag word (US.FLG).
2. Exits via RTS PC.

COMMENTS:

MODULE NAME: COMMON

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: Initializers USB for new user

CALLING ROUTINES: RING

CALLING SEQUENCE: JSR PC, COMMON

COMMON:

SUBROUTINES CALLED: ECHDON
TRAP TR.QUE

FUNCTION DESCRIPTION:
1. Checks if ECHO buffer is in use.
2. Queues an element onto RDQUE.
3. Initializes USB PARAMETERS.

COMMENTS:

MODULE NAME: DAP

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Dialogue prompt speaking routine.

CALLING ROUTINES: DAPCOM, BACKGR

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: SPEAK
All SP.*** special functions, using routine specified in TABLE (VECTOR)

FUNCTION DESCRIPTION:

1. Gets pointer to next protocol field.
2. Executes special function before prompt is specified.
3. Speaks prompt.
4. Jumps to DAP if cycle request else to BACKGR.

COMMENTS:

MODULE NAME: DAPCOM

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: Dialogue protocol cycling routine

CALLING ROUTINES: BACKGR, DAP

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: SYNTAX
ECHO
All SP.*** via dialogue TABLE pointers, at vector

FUNCTION DESCRIPTION:

1. Gets cycle pointer from USB.
2. Performs special function if any in table before SYNCHK.
3. Performs syntax check.
4. Performs special function before echo if entry in table.
5. Echos response if required.
6. Performs special function before branching if entry in table
7. Gets pointer to next dialogue table. depending on yes, no or normal response.
8. Continues to DAP.

COMMENTS:

MODULE NAME: DECRM

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Decrements message unit number during repeat and recycle.

CALLING ROUTINES: RPTSKP

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1. Adds USB BASE ADDRESS TO OFFSET IN R5.
2. Decrements message unit number.
3. If resulting message unit number is less than 0, repleces that with 9.

COMMENTS:

MODULE NAME:	DISCON
PROGRAM:	11/34 VRS.SAV
SOURCE FILE:	DAP.MAC
PURPOSE:	Disconnects user at end of briefing
CALLING ROUTINES:	BRIEFR
CALLING SEQUENCE:	
COMMON:	

SUBROUTINES CALLED:

ECHDON	RTNQUE	COMMON
MRKTIM	CHKREQ	RPTREQ
BLDBRF	TRESET	REPDEC
SEND	ENABLE	TR.MOD
		TR.QUE

FUNCTION DESCRIPTION:

1. Cancels channel's TIMEOUT marktime.
2. Interrupts speech in progress.
3. Returns ECHO buffers.
4. Returns QUEUE elements.
5. Informs 11/70 of disconect.
6. Performs disconnect.
7. If not a console disconnect (see section 6.1.3.4), enables line.
8. Exits to BACKGR.

COMMENTS:

MODULE NAME: ECHO

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Echoes user response

CALLING ROUTINES: DAPCOM

CALLING SEQUENCE: JSR PC, ECHO

COMMON: PREFIX.MAC defined parameters

SUBROUTINES CALLED: TRAP TR.DQE
DICT
SPEAK

FUNCTION DESCRIPTION:
1. Resolves input string.
2. Dequeues an element from RDQUE.
3. ADDS "..." before phrase for short delay checks for phonetic echo.
4. Translates phrase by call to DICT.
5. Busy's out echo buffer.
6. Speaks.

Exits via RTS.

COMMENTS:

MODULE NAME: ECHDON

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: Returns dynamic buffers used in echo function

CALLING ROUTINES: COMMON

CALLING SEQUENCE: JSR PC, ECHDON

COMMON: PREFIX.MAC defined parameters

SUBROUTINES CALLED: TRAP TR.QUE

FUNCTION DESCRIPTION:

1. If in briefing mode echo done flag is cleared, then QUEUE ELEMENT AT US.SPK is returned to RDQUE.
2. If in correction mode, correction flag is cleared, then QUEUE element at US. RCV is returned to RDQUE.
3. Return via RTS PC.

COMMENTS:

MODULE NAME: GO

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Resumes briefing in response to user command *4

CALLING ROUTINES: BACKGR

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: TRAP TR.MOD

FUNCTION DESCRIPTION:
1. Take a Trace.
2. Resume speech only if interrupted by stop command.
3. Exit to BACKGR.

COMMENTS:

MODULE NAME: INVALK

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: Handles invalid keystroke entries

CALLING ROUTINES: BACKGR

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: TRAP TR. MOD
SPEAK
CLRTTK

FUNCTION DESCRIPTION:
1. Puts invalid keystroke flag in status word of USB.
2. Resets input buffer/.
3. Speaks message "invalid entry".
4. Enables more Touch-Tone® inputs.
5. Exits to BACKGR.

COMMENTS:

MODULE NAME: MORSPK

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Checks if more inputs to speak

CALLING ROUTINES: MRKTIM

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: TRAP TR. USB
READ
TYRANT

FUNCTION DESCRIPTION:
1. Saves R2, R3, R4, and R5 on stack.
2. Gets USB address.
3. If more inputs
   READS inputs to double buffers
   Restores registers
   Exits completion routine
   If no move inputs, it exits to Backgr.

COMMENTS:

MODULE NAME: MRKCOM

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: Marktime completion routine for MRKTIM

CALLING ROUTINES: Entered at completion of marktime request issued by MRKTIM routine

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: SIGNAL

FUNCTION DESCRIPTION:
1. REsolves USB address.
2. Sets up RETRVN FLAG IN VS.FLG of USB.
3. Signals event by JSR PC, signal.
4. Returns via RTS PC.

COMMENTS:

MODULE NAME: MRKTIM

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: To wait an interval of time specified by R4

CALLING ROUTINES: DISCON

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1. Pops a word off the stack to save in USB for return address.
2. Stores R1 in USB save area.
3. Gets time parameter from R4 and issues MRKT request.
4. Returns to polling loop (JMP BACKGR).

COMMENTS:

MODULE NAME: NO

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: Sets no response indication in USB permanent flag bits and line status word.
This occurs as a result of user answering a yes/no query with a no.

CALLING ROUTINES: BACKGR

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1 Sets appropriate bits in US. PER and in R1.
2. Branches to CHUSB.
3. CHUSB puts R1 into US.STA and returns to DAPCOM.

COMMENTS:

MODULE NAME: NORMAL

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: Sets normal response indication in USB

CALLING ROUTINES: BACKGR

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1. Sets appropriate bits in R1.
2. Puts R1 into VS.STA and returns to DAPCOM.

COMMENTS:

MODULE NAME: PUTTR

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Clears out talk required list (TRL)

CALLING ROUTINES: RING

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: TRAP TR.DQE
TRAP TR.QUE

FUNCTION DESCRIPTION:
1. Calculates TRL list head ADDR.
2. Dequeues an element.
3. Queues element onto RDQUE.
4. Loops until no elements in TRL, then returns to BACKGR.

COMMENTS:

MODULE NAME: RECYC

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: In briefing mode, restarts briefing from beginning in prompt mode, restarts from "hello"

CALLING ROUTINES: BACKGR

CALLING SEQUENCE:

COMMON: All FL.*** as defined in PREFIX.MAC
US.***
TR.***
ST.***

SUBROUTINES CALLED: TRAP TR.MOD

FUNCTION DESCRIPTION:

1. Puts beginning of protocal indication in line status field.
2. If in briefing mode, starts at beginning of briefing by putting message unit #00 in US.SPK and executing the repeat function (JMP REPEAT).
3. If not in briefing mode, re-starts the session by executing the disconnect logic (BR DISCON).

COMMENTS:

MODULE NAME: REPEAT

PROGRAM: 11/34 VRS

SOURCE FILE: DAP.MAC

PURPOSE: Repeats last message unit

CALLING ROUTINES:

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: RPTSKP
TRAP TR.MOD

FUNCTION DESCRIPTION:

1. Modifies line status field of USB.
2. If in briefing mode, goes to RPTSKP. If not, waits for completion of speech before repeating last prompt.
3. Exits to BACKGR

COMMENTS:

MODULE NAME:	RING
PROGRAM:	11/34
SOURCE FILE:	DAP.MAC
PURPOSE:	Ring indication routine for all channels.
CALLING ROUTINES:	BACKGR
CALLING SEQUENCE:	
COMMON:	
SUBROUTINES CALLED:	COMMON PUTTR TR.MOD

FUNCTION DESCRIPTION:

1. Executes common setup routines.
2. Sets ring indication in USB via tR.MOD.
3. Sets up line timeout if not disabled (15 min).
4. Sets briefing mode to prompt.
5. Clears out TRL.
6. Exits to DAP.

COMENTS:

MODULE NAME: RPTREQ (Also REPDEC)

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Returns elements to RDQUE

CALLING ROUTINES: DISCON

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: TRAP TR.QUE

FUNCTION DESCRIPTION:
1. If entered thru RPTREQ, queues one element, address of which is in R5, to RDQUE and exits to BACKGR.
2. If entered thru REPDEC, queues one element, address of which is in R4, to RDQUE and exits to BACKGR.

COMMENTS:

MODULE NAME:	RPTSKP
PROGRAM:	11/34 VRS
SOURCE FILE:	DAP.MAC
PURPOSE:	Routine common to SKIP and REPEAT commands in briefing mode only
CALLING ROUTINES:	REPEAT SKIP
CALLING SEQUENCE:	JMP RPTSKP or BR RPTSKP
COMMON:	All TR.*** US.*** FL.*** as defined in PREFIX.MAC
SUBROUTINES CALLED:	BLDBRF, SEND, RTNQUE, CHKREQ, CLRTTK, INCREQ, TSTRCV, SENDRT, SPEAK, TR.QUE
FUNCTION DESCRIPTION:	1. If briefing done flag is high, ignores repeat skip, and exits to GO. 2. If repeat request, backs up to beginning of message unit and returns to BACKGR. 3. If skip request, dumps message unit pointers, returns QUEUE elements, re-enables Touch-Tone® inputs and exits by JMP BRIEFR.
COMMENTS:	

MODULE NAME:	RTNQUE
PROGRAM:	11/34 VRS
SOURCE FILE:	DAP.MAC
PURPOSE:	Dequeues all QUEUE elements from speak QUEUE and returns them to reads QUEUE
CALLING ROUTINES:	RPTSKP DISCON TOGO
CALLING SEQUENCE:	JSR PC, RTNQUE
COMMON:	All TR.*** US.*** SP.*** FL.*** DP.*** defined in PREFIX.MAC
SUBROUTINES CALLED:	TRAP TR.DQE TRAP TR.QUE
FUNCTION DESCRIPTION:	1. Determine speak Q address from USB address. 2. Dequeues an element. 3. If no element, exit. 4. Queues the element to RDQUE. 5. Go back to step 1.
COMMENTS:	

MODULE NAME: SKIP

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Skips to next message unit in response to user command *5

CALLING ROUTINES: BACKGR

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: CLRTTK, GO, RPTSKP, TR.MOD

FUNCTION DESCRIPTION:

1. Modifies line status block.
2. Checks if user is in briefing mode. If not, enables Touch-Tone® and exits to BACKGR inputs.
3. Checks if briefing is done, if so ignore command.
4. Jumps to RPTSKP to skip report being spoken.

COMMENTS:

MODULE NAME: STOP

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Stops briefing in response to user command *

CALLING ROUTINES: BACKGR

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: TRAP TR.MOD

FUNCTION DESCRIPTION:
1. Takes a trace.
2. Interrupts speech if in briefing mode.
3. Exits to BACKGR.

COMMENTS:

MODULE NAME: TIMOUU

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Line timeout completion routine

CALLING ROUTINES: RING issues a .MRKT which calls TIMOUU upon completion

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: TRAP TR.USB
SIGNAL

FUNCTION DESCRIPTION:
1. Determines USB addr of offending channel.
2. Sets exit bit in USB.
3. Signals event to BACKGR.
4. Returns from completion routine.

COMMENTS:

MODULE NAME: TOGO

PROGRAM: 11/34 VRS.SAV

SOURCE FILE: DAP.MAC

PURPOSE: Waits for end of current message, then speaks timeout message.

CALLING ROUTINES: BACKGR

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: TRAP TR.DQE
TR.QUE
MARKTIM
RTNQUE
PUTTR
SPEAK
SP.DIS

FUNCTION DESCRIPTION:
1. Turns off briefing mode.
2. Waits 3 seconds.
3. Dequeues any talk header elements and returns them to free element pool.
4. Also returns user's read header elements to free pool.
5. Returns speak Queue elements.
6. Returns TRL Queue elements.
7. Speaks timeout message.
8. Waits 3 seconds.
9. Hangs up on user.
10. Returns to polling loop (BACKGR).

COMMENTS:

MODULE NAME:	YES
PROGRAM:	11/34 VRS
SOURCE FILE:	DAP.MAC
PURPOSE:	Sets YES response bits in permanent flag and line status words of USB
CALLING ROUTINES:	BACKGR
CALLING SEQUENCE:	
COMMON:	
SUBROUTINES CALLED:	None
FUNCTION DESCRIPTION:	1. Sets appropriate bits in USPER and in R1. 2. Branches to CHUSB. 3. CHUSB puts R1 into VS.STA and returns to DAPCOM.
COMMENTS:	

MODULE NAME: DICT-DICTST

PROGRAM: 11/34 VRS

SOURCE FILE: VOCAB.MAC

PURPOSE: Translate ASCII text into VRS code pairs

CALLING ROUTINES: Dictionary initialization in BACKGR.MAC and ECHO in DAP.MAC

CALLING SEQUENCE: Call DICTST, which calls DICT as a marktime completion routine

COMMON:

SUBROUTINES CALLED: SIGNAL

FUNCTION DESCRIPTION:
1. R2 -- Address of text string to be translated.
2. R3 -- Address of word pair
   1 word - byte length of translation
   2 word - address of translation.

COMMENTS: DICTST is called to set a one second marktime which will call DICT as a completion routine.

AD-A102 185 INPUT OUTPUT COMPUTER SERVICES INC WALTHAM MA F/G 17/2

TWENTY-CHANNEL VOICE RESPONSE SYSTEM.(U)

JUN 81 DOT-TSC-1313

UNCLASSIFIED FAA-RD-81-51 NL

3 of 5

AD A
102185

MODULE NAME:	ALPHA
PROGRAM:	11/34 VRS
SOURCE FILE:	SPEC.MAC
PURPOSE:	Check input buffer characters for proper locid syntax - alpha-numeric
CALLING ROUTINES:	SYNTAX
CALLIN SEQUENCE:	
COMMON:	SYNFLG: Flag for 1st character check - then '/' will be allowed
SUBROUTINES CALLED:	VALID, INVALA (SYNTAX), ANEX
FUNCTION DESCRIPTION:	1. Input: R3 - input buffer pointer. 2. Output: C-Bit set for invalid format.
COMMENTS:	

MODULE NAME: ASKYNO

PROGRAM: VRS (11/34)

SOURE FILE: SPEC.MAC

PURPOSE: Sets error flag if last response not yes.

CALLING ROUTINES: SP.FCT SP.NOT
SP.FER SP.FTR
SP.LOB SP.PRP
SP.SYR SP.SAS

CALLING SEQUENCE:

COMMON: FL.YER
UR.PER

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: 1. Input: R0 - USB address.
2. Output: C-bit set for error return.

COMMENTS: The return address is popped off stack if error, that is, not a yes response.

MODULE NAME: BRIEFR

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: Check for phone hang up; if so jumps to disconnect logic. If not, gets next protocol address and puts the return address on stack.

CALLING ROUTINES: BACKGR

CALLING SEQUENCE:

COMMON: Prefix parameters:
FL.TRN
US.PER
US.DAP
VECTOR
US.SA1
US.SA2

SUBROUTINES CALLED: DISCON

FUNCTION DESCRIPTION:
1. Input: R0 - USB address.
2. Output: R1 - protocol vector address
SP - saved return address.

COMMENTS:

MODULE NAME: CKHNUM

PROGRAM: 11/34 VRS

SOURCE FILE: SPEC.MAC

PURPOSE: To check input characters are numeric

CALLING ROUTINES: NUMINP

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: NINVAL

FUNCTION DESCRIPTION:
1. Input: R3 - pointer to character to be checked.
2. Output: Calls 'NINVAL' if character not number.

COMMENTS:

MODULE NAME:	NUMBER
PROGRAM:	11/34 VRS
SOURCE FILE:	SPEC.MAC
PURPOSE:	Count number of characters process and check that character is numeric
CALLING ROUTINES:	DAP
CALLING SEQUENCE:	
COMMON:	SYNFLG: used as character processed flag NUMFLG: count of characters processed
SUBROUTINES CALLED:	INVALN (see SYNTAX)
FUNCTION DESCRIPTION:	Input: R3 - input buffer pointer.
COMMENTS:	

MODULE NAME: CKHNUM

PROGRAM: 11/34 VRS

SOURCE FILE: SPEC.MAC

PURPOSE: To check input characters are numeric

CALLING ROUTINES: NUMINP

CALLING SEQUENCE:

COMMON:

SUBRUTINES CALLED: NINVAL

FUNCTION DESCRIPTION:
1. Input: R3 - pointer to character to be checked.
2. Output: Calls 'NINVAL' if character not number.

COMMENTS:

MODULE NAME: SP.BBL

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: Enter channel identifier, and briefing mode into buffer and initialize location flags and counters

CALLING ROUTINES: SP.LST

COMMON:

US.BEG	FL.LST
US.TRM	FL.FIR
US.BRF	
US.CUR	
US.RCV	
US.PER	

SUBROUTINS CALLED: None

FUNCTION DESCRIPTION:
1. Input: R0 - USB address.
2. Output: Channel identifier and briefing mode entered into buffer.

COMMENTS:

MODULE NAME: SP.BRE

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: Moves the briefing mode into the buffer

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON:
US.BEG
US.IND
US.BRF
US.CUR

SUBROUTNES CALLED: None

FUNCTION DESCRIPTION:
1. Input: R0 - USB address
US.BEG - contains beginning point for buffer
US.BRF - contains briefing mode.
2. Output: buffer now contains briefing mode.

COMMENTS:

MODULE NAME: SP.BR1

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: Check briefing mode input against table of valid modes ('Prompt,' 'Enmode,' 'local') and inputs valid mode into buffer

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: US.INP
US.CUR
US.BRF

SUBROUTINES CALLED: INVALK, SP.BRE

FUNCTION DESCRIPTION: Input: R0 USB address.

COMMENTS:

MODULE NAME: SP.CAR

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE:
1. Calculates number of characters in the input buffer
2. Saves the return addresses in the USB JMPS to NSPLA to send data

CALLING ROUTINES:
SP.CSV
SP.ETA
SP.FTR
SP.LST
SP.WRN

CALLING SEQUENCE:

COMMON:
US.CUR
US.BEG
US.SA1
US.SA2

SUBROUTINES CALLED: DISPLA

FUNCTION DESCRIPTION:
1. Input: R0 - USB address.
2. Output: R4 - number of characters.

COMMENTS:

MODULE NAME: SP.CLA

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: Places the terminal identifier in 1st buffer position and saves the next position as current location pointer and last valid input pointer

CALLING ROUTINES: BLDBRF
SP.ENR
SP.LST
SP.SMD
SP.WRN

CALLING SEQUENCE:

COMMON: US.BEG
US.TRM
US.CUR
US.IND

SUBROUTINES CALLED: SP.CLR

FUNCTON DESCRIPTION: Input: R0 USB address.

COMMENTS:

MODULE NAME: SP.CLR

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: Clear the buffer positions not used, that is, those following the current buffer position as defined by US.CUR.

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: .LVBUF
US.CUR
US.BEG

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: Input: R0 - USB address.

COMMENTS:

MODULE NAME: SP.CSV

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: To call SP.CAR for preparation to send message

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: SP.CAR

FUNCTION DESCRIPTION: INPUT: user status block address.

COMMENTS:

MODULE NAME: SP.DIS

PROGRAM: 11/34 VRS

SOURCE FILE: SPEC.MAC

PURPOSE: Initialize USB, reset Touch Tone® line, and disconnect line

CALLING ROUTINES: TOGO

CALLING SEQUENCE:

COMMON: US.PER
US.CHN
FL.DID
US.FLG

SUBROUTINES CALLED: COMMON, TRESET, SIGNAL, BACKGR

FUNCTION DESCRIPTION:
1. Input: R0 - USB address.
2. Output: R1 - channel number.

COMMENTS: SP.DDD is same as SP.DIS except for 'excessive time' terminator signal is first set.

MODULE NAME: SP.ENR

PROGRAM: VRS 11/34

SOURCE FILE: SPEC.MAC

PURPOSE: Clear out input buffer, insert 'SD0' for a 'scan data' request

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: FL.YER
US.PER
SP.CLA

SUBROUTINES CALLED: SP.CLA

FUNCTION DESCRIPTION: Input: R0 - USB address.

COMMENTS:

MODULE NAME: SP.ETA

PROGRAM: 11/34 VRS

SOURCE FILE: SPEC.MAC

PURPOSE: Clears 6 characters in input buffer and update USB input buffer pointer, US.CUR.

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: US.CUR

SUBROUTINES CALLED: SP.CAR

FUNCTION DESCRIPTION: Input: R - USB address.

COMMENTS:

MODULE NAME: SP.FCT

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: For En route mode, enters FT's and synopsis into input buffer

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: ASKYNO, RPTYP

FUNCTION DESCRIPTION: Input: R3 input buffer pointer.

COMMENTS:

MODULE NAME: SP.FDR

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: To determine if FD's requested, clears C-bit if yes sets C-bit and sends data if not.

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: FL.PHE
US.FLG

SUBROUTINES CALLED: SP.CAR

FUNCTION DESCRIPTION:
1. Input: R0 - USB address.
2. Output: C-bit set if FD's not requested cleared otherwise.

COMMENTS:

MODULE NAME: SP.FER

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: To add FD request to output buffer

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: US.CUR
US.INP

SUBROUTINES CALLED: ASKYNO

FUNCTION DESCRIPTION: Input: R0 - USB address,
R3 - output buffer pointer.

COMMENTS:

MODULE NAME:	SP.FTB
PROGRAM:	11/34 VRS
SOURCE FILE:	SPEC.MAC
PURPOSE:	Sets report value to FT, then calls Check B to check for reports available, none available species none in effect message
CALLING ROUTINES:	DAP
CALLING SEQUENCE:	
COMMON:	
SUBROUTINES CALLED:	CHECKB (in SP.SAB)
FUNCTION DESCRIPTION:	1. Input: R2- FT value. 2. Output: R3 - pointer to none in effect message.
COMMENTS:	

MODULE NAME: SP.HYP

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: Insert a hyphen into input data

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: US.CUR

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1. Input: R0 - USB address.
2. Output: R3 - points to current location pointer (before hyphen).

COMMENTS:

MODULE NAME: SP.LOB

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: For En route mode; enters SA's, UA's, NO's into output buffer

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: ASKYNO, RPTYP

FUNCTION DESCRIPTION:
1. Input: R3 - output buffer pointer.
2. Output: R3 - output buffer pointer.

COMMENTS:

MODULE NAME: SP.LOC

PROGRAM: VRS - 11/34

SOURCE FILE: SPEC.MAC

PURPOSE: To check if loc entered is valid format and if 10 locs entered.

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON:

```
US.INP     FL.LST
US.CUR     FL.LOC
US.RCV
US.PER
```

SUBROUTINES CALLED: INVALK

FUNCTION DESCRIPTION:

```
1. Input:  R0 - USB address.
2. Output:  US.PER - last loc flag set on 10th
                     loc
                   - loc entered flag set if
                     format valid
          US.RCV+2 - increment total of locs
                     entered
             C-bit - set for abnormal exit -
                     invalid loc or 10th loc.
```

COMMENTS:

MODULE NAME:	SP.LST
PROGRAM:	VRS (11/34)
SOURCE FILE:	SPEC.MAC
PURPOSE:	Checks if loc entered was last loc and/or correction mode if not: normal return to DAP, if yes, the data are sent. If select mode, the report types are also sent.
CALLING ROUTINES:	DAP
CALLING SEQUENCE:	
COMMON:	FL.LST RDQUE US.PER TR.QUE - QUEUE trap address FL.COR US.DAP FL.YER US.BRF US.RCV US.CUR US.BEG
SUBROUTINES CALLED:	DISPL2, SP.BBL. SP.CAR, SP.CLA
FUNCTION DESCRIPTION:	1. Input - R0 - USB address. 2. Output - C-bit set if not local mode briefing when last location processed.
COMMENTS:	

MODULE NAME:	SP.MOD
PROGRAM:	VRS - 11/34
SOURCE FILE:	SPEC.MAC
PURPOSE:	Checks if last response a 159 - '!' if yes sets up for briefing mode query
CALLING ROUTINES:	DAP
CALLING SEQUENCE:	
COMMON:	US.CUR US.DAP
SUBROUTINES CALLED:	None
FUNCTION DESCRIPTION:	1. Input: R0 USB address, 2. Output: #2 in dialogue protocol US.DAP.
COMMENTS:	This is not used (commented out) while in prompt mode only.

MODULE NAME: SP.SAB

PROGRAM: 11/34 VRS

SOURCE FLE: SPEC.MAC

PURPOSE: Check for SA's available, if not, speak 'none in effect' message

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON:
US.RPT
NONEFF
DP.ABN
NS.DAP
FL.DIS
US.FLG

SUBROUTINES CALLED: SPEAK

FUNCTION DESCRIPTION:
1. Input: R0 - USB address.
2. Output: R3 - pointer to message to be spoken.

COMMENTS:

MODULE NAME: SP.SMD

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: To determine if briefing mode is 'En route' or 'Prompt' and points to proper dialogue.

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: US.BRF
US.DAP

SUBROUTINES CALLED: SP.CLA

FUNCTION DESCRIPTION: Input: R0 - USB address.

COMMENTS:

MODULE NAME: 1. SP.SYR
2. SP.NOT
3. SP.FTR
4. SP.PRP
5. SP.SAS

PROGRAM: 11/34 VRS

SOURCE FILE: SPEC.MAC

PURPOSE: To put request in output buffer for:
1. Synopsis
2. NOTAMS
3. Terminal Forecasts (FT)
4. Pilot REports (UA)
5. Surface observations (SA's)

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: ASKYNO, RPTUP

FUNCTION DESCRIPTION: 1. Input: R3 - output buffer pointer.
2. Output: R3 - updated output buffer pointer past inserted request.

COMMENTS:

MODULE NAME: SP.TIM

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: Gets present time, disables Touch-Tone® input, speaks time, and initializes users buffer.

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: US.SA1 FL.DIS
US.FLG
US.CUR

SUBROUTINES CALLED: ECHO, COMMON, GETTIM

FUNCTION DESCRIPTION: Input: R0 - USB address.

COMMENTS:

MODULE NAME: SP.WMD

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: Checks if briefing mode local if not, returns. If yes, pops return address of stack, sets dialogue protocol for local and jumps to DAP,

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: US.DAP
US.8RF

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1. Input: R0 - USB address.
2. Output: US.DAP set to 6 if briefing mode local.

COMMENTS:

MODULE NAME: SP.WRN

PROGRAM: VRS (11/34)

SOURCE FILE: SPEC.MAC

PURPOSE: Puts briefing mode (En route, Select, or Prompt: into output buffer

CALLING ROUTINES: DAP

CALLING SEQUENCE:

COMMON: US.CUR
US.DAP
US.BRF

SUBROUTINES CALLED: SP.CLA, SP.CAR

FUNCTION DESCRIPTION: Input: RO - USB address.

COMMENTS:

MODULE NAME: SYNALT

PROGRAM: (11/34) VRS

SOURCE FILE: SPEC.MAC

PURPOSE: Check altitude input for proper format and value alt - either greater than 1000 ft or less than 45999 with either two digit or 4 digit input

CALLING ROUTINES: SYNTAX (SPEC.MAC)

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: NUMIN, NINVAL, OKVAL

FUNCTION DESCRIPTION:
1. Input: R3 - input buffer pointer
   R4 - No. of characters
2. Output: Either clear or set C-bit for valid or invalid syntax.

COMMENTS:

MODULE NAME:	SYNETA
PROGRAM:	(11/34) VRS
SOURCE FILE:	SPEC.MAC
PURPOSE:	Check syntax of ETA (winds) time characters in input buffer and adds 'Z' for zulu time
CALLING ROUTINES:	SYNTAX (SPEC.MAC)
CALLING SEQUENCE:	
COMMON:	US.CUR - current input pointer
SUBROUTINES CALLED:	NUMIMP, NINVAL, OKVAL
FUNCTION DESCRIPTION:	1. Input: R4 - No. of characters. R3 - pointer to input array. 2. Output: US.CUR is updated.
COMMENTS:	

MODULE NAME: SYNHR

PROGRAM: (11/34) VRS

SOURCE FILE: SPEC.MAC

PURPOSE: Check hour value input for winds report must be numeric and less than or equal to 30 hours

CALLING ROUTINES: SYNTAX

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: NINVAL, OKVAL, NUMIMP

FUNCTION DESCRIPTION:
1. Input: R3 - input buffer pointer
   R4 - No. of characters.
2. Output: C-bit: Cleared for valid format or value set for invalid.

COMMENTS:

MODULE NAME:	SYNTAX
PROGRAM:	(11/34) VRS
SOURCE FILE:	SPEC.MAC
PURPOSE:	Check input buffer characters for appropriate subroutine to call to check format
CALLING ROUTINES:	ALPHA DAPCOM YESCK
CALLING SEQUENCE:	INVALN - called by NUMBER (SPEC.MAC) VALID, INVALY - call by YESCK (SPEC.MAC) INVALT - call by WETPCK
COMMON:	SYNFLG - first pass flag NUMFLG - numeric flag USINP
SUBROUTINES CALLED:	ALPHA, SYNHR, SYNALT, SYNETA, WETPCK, YESCK, VALID
FUNCTION DESCRIPTION:	1. Input: R2 - buffer pointer. 2. Output: C-bit set for invalid format.
COMMENTS:	Following are 'mini' - routines contained in Syntax INVALA sets invalid alpha flag in ST.SNV - into R3 INVALN sets invalid number flag in ST.SNV - into R3 INVALT sets invalid type flag in ST.SNV - into R3 INVALY sets invalid Y/N flag in ST.SNY - into R3 INVALU - modifies the line status flag according to the above flags that had been set.

MODULE NAME: WETPCK

PROGRAM: (11/34) VRS

SOURCE FILE: SPEC.MAC

PURPOSE: Check input buffer for valid weather type

CALLING ROUTINES: SYNTAX (SPEC.MAC)

CALLING SEQUENCE:

COMMON: FL.DHE
US.FLG
SYNFLAG - hold weather type characters

SUBROUTINES CALLED: VALID, INVALT

FUNCTION DESCRIPTION: Input: R3 - input buffer pointer.
Output: If winds report, 'ED'; sets FD flag in US.FLG.

COMMENTS:

MODULE NAME: YESCK

PROGRAM: (11/34) VRS

SOURCE FILE: SPEC.MAC

PURPOSE: Check input buffer for valid yes or no response. Prompt must call for 4/N and 4/N must be in right format.

CALLING ROUTINES: SYNTAX (SPEC.MAC)

CALLING SEQUENCE:

COMMON: USB parameters:
FL.YES
US.FLG
FL.YER
US.PER
FL.NO

SUBROUTINES CALLED: VALID, INVALY (SYNTAX)

FUNCTION DESCRIPTION:

1. Input: R0 - USB address
   R3 - input buffer pointer
   R1 - protocol mask pointer,
2. Output: R2 = 50 for no response
   R2 = 47 for a yes response.

COMMENTS:

MODULE NAME: MAP

PROGRAM: (11/34) VRS

SOURCE FILE: SPEAK.MAC

PURPOSE: Maps 4K memory segments

CALLING ROUTINES: READC

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: HALT

FUNCTION DESCRIPTION:
1. Saves R0 on the stack.
2. Sets up window offsets and maps the region.
3. If error, calls HALT routine which bolts the processor.
4. Restores R0 and exits.

COMMENTS:

MODULE NAME:	READ
PROGRAM:	11/34 VRS
SOURCE FILE:	SPEAK.MAC
PURPOSE:	Reads data from vocabulary disk
CALLING ROUTINES:	SPEAKR
CALLING SEQUENCE:	JSR PC, READ
COMMON:	All TR.*** US.*** FL.*** BQ.*** as defined in PREFIX.MAC
SUBROUTINES CALLED:	TRAP TR.DQE, TRAP TR.QUE, HALT, MAP
FUNCTION DESCRIPTION:	1. Gets a queue element from fill pool and puts it on read list head. 2. Performs mapping if necessary. 3. Issues a .READC request to disk.
COMMENTS:	

MODULE NAME:	READC
PROGRAM:	(11/34) VRS
SOURCE FILE:	SPEAK.MAC
PURPOSE:	Read completion routine for disk (reading speech file)
CALLING ROUTINES:	READ
CALLING SEQUENCE:	Called at completion of a .READC request
COMMON:	All TR.*** US.*** FL.*** as defined in PREFIX.MAC
SUBROUTINES CALLED:	MAP HALT
FUNCTION DESCRIPTION:	1. If error on previous read, prints error message. 2. Calculates USB address. 3. Saves R2, R3, R4, R5 on the stack. 4. Moves Queue element from read queue to talk list head. 5. Maps user into extended memory buffer. 6. Issues a .WRITE request to ADPCM output device. 7. Restores USB address and saved registers, enables Touch-Tone® and exits.
COMMENTS:	

MODULE NAME: SPEAKR

PROGRAM: (11/34) VRS

SOURCE FILE: SPEAK.MAC

PURPOSE: Queue speak buffer and issue reads to disk for speech data

CALLING ROUTINES: SPEAKST

CALLING SEQUENCE: JSP PC, SPEAKR

COMMON:

All	ST.***	as defined in PREFIX.MAC
	FL.***	
	US.***	

SUBROUTINES CALLED: READ
TYRANT

FUNCTION DESCRIPTION:
1. Records speak indication in USB.
2. Queues element onto speak queue.
3. Extracts message fields
4. Initiates double-buffered disk reads.
5. Exits.

COMMENTS:

MODULE NAME: SPKST

PROGRAM: 11/34 VRS

SOURCE FILE: SPEAK.MAC

PURPOSE: Sets up speech buffers

CALLING ROUTINES: Completion routine from MARKTIME issued in speak module

CALLING SEQUENCE:

COMMON: All TR.*** as defined in PREFIX.MAC
US.***
FL.***

SUBROUTINES CALLED: TRAP TR.USB
SPEAKR

FUNCTION DESCRIPTION:
1. Saves R2, R3, R4, R5 on the stack.
2. Gets USB address.
3. Sets speak indicator in USB and executes speak routine.
4. Clears speak indicator.
5. Restores saved registers and returns.

COMMENTS:

MODULE NAME: TYRANT

PROGRAM: (11/34) VRS

SOURCE FILE: SPEAK.MAC

PURPOSE: Controls speaking process. Sets lst block address, number of blocks and last words. Returns if end of message and not hanging up. Dequeues element from message queue, queues the last message buffer to free pool queue and requests next message if end of briefing or hang up, indicates end of briefing and enables Touch Tone® Input.

CALLING ROUTINES: MORSPK WRITC SPEAKR

CALLING SEQUENCE:

COMMON:

US.lst FL.INT TR.DQE
US.FLG
US.NUM
US.BLK
US.MSG
US.PER
US.DMB

SUBROUTINES CALLED: INCREQ
BLDBRF
SENDRT

FUNCTION DESCRIPTION:
1. Input: R0 - USB address.
2. Output: US.NUM (R0) number of consecutive blocks.
US.LST (R0) number of words in last block.
US.BLK (R0) address of lst block.
US.MJG (R0) updated pointer for next speak pass.
US.FLG (R0) end of talk mode flag set if end.

COMMENTS:

MODULE NAME:	WRITC
PROGRAM:	(11/34) VRS
SOURCE FILE:	SPEAK.MAC
PURPOSE:	Write completion routine for ADPCM output
CALLING ROUTINES:	READC
CALLING SEQUENCE:	This is completion routine for .WRITC in READC module.
COMMON:	All TR.*** US.*** FL.*** ST.*** as defined in PREFIX.MAC
SUBROUTINES CALLED:	TRAP TR.QUE TRAP TR.DQE TYRANT READ SIGNAL
FUNCTION DESCRIPTION:	1. If error on write, prints error message. 2. Saves R2, R3, R4, R5 on the stack. 3. Calculates USB address if illegal USB address, prints a message. 4. Returns speech element to free pool. 5. Gets next message field and reads from disk. 6. Restores saved registers and exits.
COMMENTS:	

MODULE NAME: ALARM/ALARMP

PROGRAM: 11/34

SOURCE FILE: SEND.MAC

PURPOSE: Alerts the operator if task RETREV or VREXEC is not running

CALLING ROUTINES: RCVER, CLKRPT

CALLING SEQUENCE: If a processor (VREXEC) alarm, jump to ALARMP.
If a RETREV alarm, jump to ALARM.

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: Rings the terminal bell 10 times and types one of the following messages:

1. RETREV NOT RUNNING. VRS ABORTING.
2. PROCESSORS NOT RUNNING.

The system exits if message #1 was typed.

COMMENTS:

MODULE NAME: BLDBRP

PROGRAM: 11/34 VRS

SOURCE FILE: SEND.MAC

PURPOSE: Composes a demand request

CALLING ROUTINES: SPEAK, DISCON, DISPLA, RPTSKP

CALLING SEQUENCE: R0 - User Status Block pointer
R2 = Demand request type

COMMON:

SUBROUTINES CALLED: SP.CLA

FUNCTION DESCRIPTION: Composes a demand request, storing it in the "current input location" pointed to by word 2 of the USB, and getting the channel and demand request number from the USB.

COMMENTS:

MODULE NAME: CHKREQ

PROGRAM: (11/34) VRS

SOURCE FILE: SEND.MAC

PURPOSE: Check ASCII Channel Number.

CALLING ROUTINES: DISCON, RPTSKP

CALLING SEQUENCE: R0 = points to USB.

COMMON:

SUBROUTINES CALLED: TRAP TR.DQE

FUNCTION DESCRIPTION: Compares the ASCII channel, number in the USB with the one in an 11/70 receive QUEUE element.

COMMENTS:

MODULE NAME: DISPLA

PROGRAM: (11/34) VRS

SOURCE FILE: SEND.MAC

PURPOSE: Initiates sends to the 11/70 and fields the responses

CALLING ROUTINES: SPEC

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: SPEAK, SEND, BLDRF, COMMON

FUNCTION DESCRIPTION: Briefing requests are sent and the address of the start of the coding which fields the responses is stored in U.S. RTN (by SEND) for the channel. This address is returned to from BACKGR when a read completes later on. When that happens, the various response formats are checked for: the message acceptable response, the diagnostic responses, and the type 2 message unit responses.

COMMENTS:

MODULE NAME: RCVC

PROGRAM: (11/34) VRS

SOURCE FILE: SEND.MAC

PURPOSE: Fields data sent from 11/70

CALLING ROUTINES: Completion routine for the .READC issued in module RCVEX

CALLING SEQUENCE: R4 points to FWA of data buffer

COMMON:

SUBROUTINES CALLED: SIGNAL, ALARM, DEFUSB, TR.DQE, TR.QUE

FUNCTION DESCRIPTION: Handles the two types of 11/70 messages queueing them for the appropriate processing. A validity check is performed and if the message is not a valid briefing request acknowledgment not a briefing message unit, the error path checks for RETREV log-on echoes, which are sent to the terminal, or for *1, indicating a response by RETREV to a poll message sent by the 11/34 every 7 seconds, or for *2, sent by RETREV if the weather processors do not wake up every 15 minutes. A branch is made to ALARM when *2 is received. When *1 is received a new 20-second MKTM issued (after cancelling the one in effect).

COMMENTS:

MODULE NAME: RCVEX

PROGRAM: (11/34) VRS

SOURCE FILE: SEND.MAC

PURPOSE: Receive protocol for 11/70 to 11/34 communication

CALLING ROUTINES: RCVC

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: RCVC completion routine, TR.DQE, TR.QUE

FUNCTION DESCRIPTION: Fetches an available QUEUE address and issues a read with completion on Channel 3.

COMMENTS:

MODULE NAME:	SEND - SENDRT
PROGRAM:	(11/34) VRS
SOURCE FILE:	SEND.MAC
PURPOSE:	Sends a byte string to the 11/70
CALLING ROUTINES:	DISPLA RPTSKP DISCON TSTRCV
CALLING SEQUENCE:	R3 = Data buffer start address R4 = Data buffer length
COMMON:	SENDC, the completion routine.
SUBROUTINES CALLED:	None
FUNCTION DESCRIPTION:	Writes a string of bytes to the 11/70 on channel 4. A checksum is computed and appended to the data.
COMMENTS:	

MODULE NAME: INCREQ

PROGRAM: (11/34) VRS

SOURCE FILE: SEND.MAC

PURPOSE: Increment the ASCII message unit number by one.

CALLING ROUTINES: RPTSKP, SPEAK

CALLING SEQUENCE: R0 = User status block pointer
R5 = Message unit number USB offset

COMMON:

SUBROUTINES CALLED:

FUNCTION DESCRIPTION: Increments the 4-character ASCII message unit number by one.

COMMENTS:

MODULE NAME: TSTRCV

PROGRAM: (11/34) VRS

SOURCE FILE: SEND.MAC

PURPOSE: Validity check on message unit data

CALLING ROUTINES: DAP

CALLING SEQUENCE: R4 points to start of input buffer.

COMMON:

SUBROUTINES CALLED: BLDBRF, SEND (SENDRT)

FUNCTION DESCRIPTION: Checks message unit pairs for validity. If the block number of a pair is invalid, the briefing request is rebuilt and sent to the 11/70 again.

COMMENTS:

MODULE NAME: EXIT

PROGRAM: (11/34) VRS

SOURCE FILE: PURGE.MAC

PURPOSE: Exit routine for 11/34 VRS

CALLING ROUTINES: BACKGR

CALLING SEQUENCE: NXTEXT sets EXITFL signal for BACKGR when a Terminal input of 'EXIT' received

COMMON:

SUBROUTINES CALLED: TRESET, MRKTIM, DISABLE, STRT

FUNCTION DESCRIPTION:
1. Closes
   - o each line channel to ADPCM hardware and disable each Touch Tone® line
   - o Dictionary file.
2. Sends exit message to 11/70 program RETREV
   - o closes input channel to 11/70
   - o closes output channel to 11/70
   - o closes Touch-Tone (MCX) channel
   - o closes ADPCM channels.

COMMENTS:

MODULE NAME: CLKRPT

PROGRAM: (11/34) VRS

SOURCE FILE: CLOCK.MAC

PURPOSE: Tics the VRS clock and attends to certain real-time scheduled functions

CALLING ROUTINES: Completion routine to a 1-sec MRKT, issued by STRTIM and issued each time thereafter by itself

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: SNDPOI
ALARM

FUNCTION DESCRIPTION: When a 1-sec MRKT expires, a second is added to the seconds-past-midnight counter. Every 7 seconds, a poll message (ESC NULL) is sent to RETREV. Also, a check is made for delays in 11/70 responses (in SNDPOI).

COMMENTS:

MODULE NAME: GETTIM

PROGRAM: (11/34) VRS

SOURCE FILE: CLOCK.MAC

PURPOSE: Put current time of day into LVM50 Touch-Tone® input buffer.

CALLING ROUTINES: SP.TIM

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED: $MLI, $DVI, $ICO

FUNCTION DESCRIPTION: Converts time to ASCII (hhmm) and stores in Touch-Tone input buffer.

COMMENTS:

MODULE NAME: $DVI

PROGRAM: (11/34) VRS

SOURCE FILE: CLOCK.MAC

PURPOSE: Integer divide routine

CALLING ROUTINES: GETTIM

CALLING SEQUENCE:
R4 = HI order dividend
R3 = LO order dividend
R1 = divisor

RETURNS:
R4 = HI order quotient
R3 = LO order quotient

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: Divides a 32-bit dividend by a 16-bit divisor for a 32-bit quotient.

COMMENTS:

MODULE NAME: $MLI

PROGRAM: 11/34 VRS

SOURCE FILE: CLOCK.MAC

PURPOSE: Integer multiply routine

CALLING ROUTINES: GETTIM

CALLING SEQUENCE:
R4 = HI order multiplicand
R3 = LO order multiplicand
R1 = multiplier

RETURNS: R4 = HI order product
R3 = LO order product

COMMON:

SUBROUTINES CALLED:

FUNCTION DESCRIPTION: Multiplies a 32-bit multiplicand by a 16-bit multiplier for 32-bit product.

COMMENTS:

MODULE NAME: TR.HAN

PROGRAM: 11/34 VRS

SOURCE FILE: TRAP.MAC

PURPOSE: Handles entry to all TRAP routines

CALLING ROUTINES: BACKGR DAP SPEC

CALLING SEQUENCE: TRAP TR.***

COMMON: TR.LST

SUBROUTINES CALLED: All TRAP routines (TRAP.TR.***)

FUNCTION DESCRIPTION:
1. Gets TRAP code from stack.
2. Checks for legal TRAP code.
3. Resolves address of desired TRAP routine.
4. Enters routine via JSR.
5. On return from routine does error checking.
6. Returns via RTI.

COMMENTS:

MODULE NAME:	TR.MOD (MODLSB)
PROGRAM:	11/34 VRS
SOURCE FILE:	TRAP.MAC
PURPOSE:	Modifies line status field of USB.
CALLING ROUTINES:	RING
CALLING SEQUENCE:	TRAP TR.MOD
COMMON:	ALL TR.*** As defined in PREFIX.MAC US.*** FL.*** SP.*** DP.***
SUBROUTINES CALLED:	TRACE
FUNCTION DESCRIPTION:	1. Places R1 in line status field. 2. If input received from 11/70, clears line timeout flag in clock. 3. Performs a trace. 4. Returns.
COMMENTS:	This routine is entered thru a TRAP vector in order to change processor priority to 7, thus preventing device interrupts from changing vital parameters.

MODULE NAME:	TR.SIG (SIGMAN)
PROGRAM:	11/34 VRS
SOURCE FILE:	TRAP.MAC
PURPOSE:	Signal flag modification routine
CALLING ROUTINES:	BACKGR
CALLING SEQUENCE:	TRAP TR.SIG
COMMON:	
SUBROUTINES CALLED:	None
FUNCTION DESCRIPTION:	1. Moves BITMSK into R1 and clears BITMSK. 2. Moves BITMSK+2 into R2 and clears BITMSK+2 3. Returns.
COMMENTS:	This routine is entered thru a TRAP vector in order to change processor priority to 7, thus preventing device interrupts from changing vital parameters.

MODULE NAME:	TR.SPK
PROGRAM:	11/34 VRS
SOURCE FILE:	TRAP.MAC
PURPOSE:	Executives SPEAK routine
CALLING ROUTINES:	SPEAKR
CALLING SEQUENCE:	TRAP TR.SPK
COMMON:	ALL TR.*** as defined in PREFIX.MAC US.*** FL.*** SP.*** DP.***
SUBROUTINES CALLED:	TRAP TR.QUE
FUNCTION DESCRIPTION:	1. QUEUES message pointer into SPEAK QUEUE. 2. Checks to see if done talking. If so, returns with carry bit clear. If still talking, returns with carry bit set.
COMMENTS:	This routine is entered thru a TRAP vector in order to change processor priority to 7, thus preventing device interrupts from changing vital parameters.

MODULE NAME: TR.USB (DEFUSE)

PROGRAM: 11/34 VRS

SOURCE FILE: TRAP.MAC

PURPOSE: Calculates USB address from channel # in R0

CALLING ROUTINES: MCX.SYS

CALLING SEQUENCE: TRAP TR.USB

COMMON: All TR.*** as defined in PREFIX.MAC
US.***
FL.***
SP.***
DP.***

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:

1. Checks for legal channel #.returns with C-bit set if error.
2. Multiples channel # by 64 and adds base address of USB.
3. Returns.

COMMENTS:

MODULE NAME: TR.DQE (DQUEUE)

PROGRAM: 11/34 VRS

SOURCE FILE: QUEUE.MAC

PURPOSE: Removes one element from AQUEUE list

CALLING ROUTINES: BACKGR,DAP,SPEC

CALLING SEQUENCE:
```
MOV #QLIST, R3
TRAP TR.DQE
```

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:

1. Address of a queue list header is placed in R3.
2. Routine exits with carry bit set if no elements in list.
3. List header and tail pointer are adjusted.
4. Routine exits with R4 containing address of QUEUE element.

COMMENTS: This routine is entered thru a TRAP vector in order to change processor priority to 7, thus preventing device interrupts from changing vital parameters.

MODULE NAME: TR.QUE (EQUEUE)

PROGRAM: 11/34 VRS

SOURCE FILE: QUEUE.MAC

PURPOSE: Inserts one element into QUEUE list

CALLING ROUTINES: BACKGR,DAP,SPEC

CALLING SEQUENCE:

```
MOV #QLIST, R3
MOV #ELADDR, R4
TRAP TR.QUE
```

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:

1. Address of QUEUE list reader is placed in R3.
2. Address of QUEUE element is placed in R4.
3. List reader and tail pointer are adjusted.
4. Routine exits with carry bit clear.

COMMENTS: This routine is entered thru a TRAP vector in order to change processor priority to 7, thus preventing device interrupts from changing vital paramenters.

MODULE NAME: TRACE

PROGRAM: 11/34 VRS

SOURCE FILE: TRACE.MAC

PURPOSE: Creates a statistical data file VRDATA.DAT.

CALLING ROUTINES: TR.MOD (MODLSB)

CALLING SEQUENCE:

COMMON:

SUBROUTINES CALLED:

FUNCTION DESCRIPTION: Fills a buffer with selected data from the User Status Block for each briefing performed and writes it to a revolving file, VRDATA.DAT, along with a record pointer block in block 0 and data record definitions prepended to each briefing's record. Upon initialization of VRS, if no file exits on disk, it is created. If one exits but was not concluded during a normal exit, the file is scanned and a record pointer block constructed.

COMMENTS:

MODULE NAME: TABLE

PROGRAM: 11/34 VRS

SOURCE FILE: TABLE.MAC

PURPOSE: Steps each user channel through the system dialogue.

CALLING ROUTINES: DAP

CALLING SEQUENCE: Twice the value in US.DAP (RO) added to the top address of TABLE (VECTOR) yields the address of the desired table.

COMMON: The special function entry points, SP.xxx.

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: For each step of the dialogue protocol there is a table of pointers and flags as follows:

1. A word of flags indicating certain temporary conditions, and expectations.
2. Address of any special function. necessary before speaking a prompt.
3. Wait interval before speaking prompt.
4. Wait interval before speaking echo.
5. Flag if to repeat same utterance after response.
6. Address of the prompt message units.
7. Address of any special function necessary to user syntax analysis.
8. Address of masks used in syntax checking.
9. Address of any special function necessary before speaking an echo.
10. Address of special function necessary before branching to next function in DAP.
11. Yes or normal response branch vector.
12. No or abnormal response branch vector.

The elements of the tables are accessed as follows: A constant stored in some address DP.XXX is added to current value of R1 to point to the right table. Another DP.XXX value is added to point to the desired element of the table.

COMMENTS:

A.2 PDP-11/70® VRS

MODULE NAME:	DICT
PROGRAM:	VREXEC
SOURCE FILE:	VOCAB.MAC
PURPOSE:	To translate ASCII text to Speech File Pointers
CALLING ROUTINES:	START (DICT.MAC) interface module
CALLING SEQUENCE:	FORTRV - ASCII text in ATADII VSNDRR DICT
COMMON:	Requires VRSDIC for Global Common
SUBROUTINES CALLED:	
FUNCTION DESCRIPTION:	Given the ASCII weather report text, a binary search is done on a list for each word to obtain the vocabulary file pointers and record lengths to be sent to the 11/34 VRS.
COMMENTS:	

<u>MODULE NAME</u>: EXTHED

<u>PROGRAM</u>: VREXEC

<u>SOURCE FILE</u>: EXTHED.FTN

<u>PURPOSE</u>: This subroutine extracts the date/time group from a header report.

<u>CALLING ROUTINES</u>: VRSSA, VRSPTR

<u>CALLING SEQUENCE</u>: Call EXTHED (A, ILEN)
where: A = raw data input array
ILEN = length in bytes of raw data array

<u>COMMON</u>:

<u>SUBROUTINES CALLED</u>: None

<u>FUNCTIONAL DESCRIPTION</u>: To extract the six-digit header date and time from the report header passed to it.

Input:

A = A byte array containing the report header.

ILEN = The length, in bytes, of the report header contained in the array A.

COMMON/ZULU/HTIME, IRTIM, STIME where HTIME, IRTIM, and STIME are all six-byte arrays.

Output:

The six-digit header date and time group is placed into the six-byte array HTIME in the labeled common ZULU.

<u>COMMENTS</u>:

MODULE NAME: LGTNG

PROGRAM: VREXEC

SOURCE FILE:

PURPOSE: This subroutine decodes lighting SA remarks.

CALLING ROUTINES: VRRMK

CALLING SEQUENCE: Call LGTNG (WORK, WLEN, RMK, RLEN, INDX, IERR)

where:
- WORK = raw data word
- WLEN = length in bytes of raw data word
- RMK = raw Remarks data array
- RLEN = length in bytes of Remarks raw data array
- INDX = current index position in Remarks raw data array
- OERR = error flag

COMMON:

SUBROUTINES CALLED: None

SYSTEM ROUTINE REQUIRED: INDSTR

FUNCTION DESCRIPTION: To decode lighting remarks which occur in the Remarks portion of SA reports.

Input:
- WORD = A byte array containing the data word to be decoded.
- WLEN = The length, in bytes, of the data word.
- RMK = A byte array containing the SA Remarks data.
- RLEN = The length, in bytes, of the SA Remarks data.
- INDX = The current pointer position within the SA Remarks data.

COMMON/RSTUFF/RLIST,IRNDS, NWX

where
- RLIST = A byte array containing the decoded Remarks
- IRNDX = The current pointer position within the decoded remoars data.
- NWX = A flag indicating if weather data were decoded in the subroutine VISWX.

Output:

The decoded lighting phrase is placed into the RLIST array and IRNDX is appropriately incremented.

IERR = An error flag which is set if the lighting remark cannot be decoded.

COMMENTS:

MODULE NAME: PCPMOD

PROGRAM: VREXEC

SOURCE FILE:

PURPOSE: This subroutine decodes precipitation SA remarks relating to hail stone size, ground fog depth, snow increasing, and precipitation in inches.

CALLING ROUTINES: VRRMK

CALLING SEQUENCE: Call PCPMOD (WORD, WLEN, RMK, RLEN, INDX, IERR)

where:
- WORD = raw data word
- WLEN = length in bytes of raw data word
- RMK = raw Remarks data array
- RLEN = length in bytes of Remarks raw data array
- INDX = current index position in Remarks raw data array
- IERR = error flag

COMMON:

SUBROUTINES CALLED: none

SYSTEM ROUTINE REQUIRED: INDSTR, INUM

FUNCTION DESCRIPTION: To decode precipitation remarks which occur in the Remarks portion of SA reports.

Input:
- WORD = A byte array containing the data word to be decoded.
- WLEN = The length, in bytes, of the data word.
- RMK = A byte array containing the SA Remarks data.
- RLEN = The length, in bytes, of the SA Remarks data.
- INDX = The current pointer position within the SA Remarks data.

COMMON/RSTUFF/RLIST, IRNDX, NWX

where
- RLIST = A byte array containing the decoded Remarks.
- IRNDS = The current pointer position within the decoded Remarks data.
- NWX = A flag indicating if weather data were decoded in the subroutine VISWX.

Output:

IERR = An error flag which is set if the precipitation remark cannot be decoded.

The decoded precipitation phrase is placed into the RLIST array and IRNDX is appropriately incremented.

COMMENTS:

MODULE NAME: INCREQ

PROGRAM: 11/34 YRS

SOURCE FILE: SEND.MAC

PURPOSE: Increment the ASCII message unit number by one.

CALLING ROUTINES: RPTSKP, SPEAK

CALLING SEQUENCE: R0 = User Status Block pointer
R5 = Message Unit Number USB offset.

COMMON:

SUBROUTINES CALLED:

FUNCTION DESCRIPTION: Input:
R7 - USB pointer.
Output:
US.DMB incremented by one.

COMMENTS:

MODULE NAME: PRES

PROGRAM: VREXEC

SOURCE FILE:

PURPOSE: This subroutine decodes SA remarks relating to pressure.

CALLING ROUTINES: VRRMK

CALLING SEQUENCE: Call PRES (WORD, WLEN, RMK, RLEN, INDX, IERR)

where:
- WORD = raw data word
- WLEN = length in bytes of raw data word
- RMK = raw Remarks data array
- RLEN = length in bytes of remarks raw data array
- INDX = current index position in remarks raw data array
- IERR = error flag

COMMON:

SUBROUTINES CALLED: None

SYSTEM ROUTINE REQUIRED: INDSTR, INUM

FUNCTION DESCRIPTION: To decode pressure remarks which occur in the Remarks portion of SA reports.

Input:
- WORD = A byte array containing the data word to be decoded.
- WLEN = The length, in bytes, of the data word.
- RMK = A byte array containing the SA Remarks data.
- RLEN = The length, in bytes, of the SA Remarks data.
- INDX = The current pointer position within the SA Remarks data.

COMMON/RSTUFF/RLIST, IRNDX, NWX

where
- RLIST= A byte array containing the decoded Remarks
- IRNDX= The current pointer position within the decoded Remarks data.
- NWX= A flag indicating if weather data were decoded in the subroutine VISWX.

Output:

The decoded pressure phrase is placed into the RLIST array and IRNDX is appropriately incremented.

IERR= An error flag which is set if the pressure remark cannot be decoded.

COMMENTS:

MODULE NAME: RNWY

PROGRAM: VREXEC

SOURCE FILE: RNWY.FTN

PURPOSE: This subroutine decoded runway visibility and visual range SA remarks.

CALLING ROUTINES: VRRMK

CALLING SEQUENCE: Call RNWY (INDX, WORD, LENGTH, ICALL, IKEY, ING)

where INDX = current position in raw data array

WORD = current raw data word

LENGTH = length in bytes of data word

ICALL = 1 for runway visibility decode, 2 for runway visual range decode

ING = error flag

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: To decode runway visibility and visual range remarks which occur in the REmarks portion of SA reports.

Input:

INDX = The current pointer position within the SA Remarks data.

WORD = A byte array containing the data word to be decoded.

LENGTH = The length, in bytes, of the data word.

ICALL = 1 for visibility decode, 2 for visual range decode.

IKEY = Points to position of 'VV' or 'VR' within the data work being decoded.

COMMON/RSTUFF/RLIST, IRNDX, NWX

where RLIST = A byte array containing the decoded Remarks.

IRNDX = The current pointer position within the decoded Remarks data.

NWX = A flag indicating if weather data were decoded in the subroutine VISWX.

Output:

The decoded runway phrase is placed into the RLIST array and IRNDX is appropriately incremented.

ING = An error flag which is set if the runway remark cannot be decoded.

COMMENTS:

MODULE NAME: RNYCND

PROGRAM: VREXEC

SOURCE FILE:

PURPOSE: This subroutine decodes runway condition SA remarks.

CALLING ROUTINES: VRRMK

CALLING SEQUENCE: Call RNYCND (WORD, WLEN, RMK, RLEN, INDX, IERR)

where:
- WORD = raw data word
- WLEN = length in bytes of raw data word
- RMK = raw remarks data array
- RLEN = length in bytes of remarks raw data array
- INDX = current index position in remarks raw data array
- IERR = error flag

COMMON:

SUBROUTINES CALLED: None

SYSTEM ROUTINES REQUIRED:

FUNCTION DESCRIPTION: To decode runway condition remarks which occur in the Remarks portion of SA reports.

Input:
- WORD = A byte array containing the data word to be decoded.
- WLEN = The length, in bytes, of the data word.
- RMK = A byte array containing the SA Remarks data.
- RLEN = The length, in bytes, of the SA Remarks data.
- INDX = The current pointer position within the SA Remarks data.

COMMON/RSTUFF/RLIST, IRNDX, NWX

where
- RLIST = A byte array containing the decoded Remarks.
- IRNDX = The current pointer position within the ddecoded Remarks data.
- NWX = A flag indicating if weather data were decoded in the subroutine VISWX.

Output:

The decoded runway condition phrase is placed into the RLIST array and IRNDX is appropriately incremented.

IERR = An error flag which is set if the runway condition remark cannot be decoded.

COMMENTS:

MODULE NAME: SKY

PROGRAM: VREXEC

SOURCE FILE: SKY.FTN

PURPOSE: This subroutine extracts and decodes sky cover data.

CALLING ROUTINES: VRSSA

CALLING SEQUENCE:

```
Call SKY (A, SKYCVR, ISKILL)
where     A =  raw data input array
     SKYCVR =  decoded sky cover data
     ISKILL =  flag indicating error in sky
               over field.
```

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:

```
To extract and decode sky cover data
occurring in the main body of an SA report.
Input:
     A = A byte array containing the SA
     report being decoded.
     COMMON/INDS/IVSTART,IVEND,ISKSTR,ISKEND
     where IVSTART = Points to beginning of
                     the visibility field
                     in the SA report.
             IVEND = Points to the end of
                     the visibility field
                     in the SA report.
            ISKSTR = Points to the begin-
                     ning of the sky cover
                     field in the SA report
            ISKEND = Points to the end of
                     the sky cover field in
                     the SA report.
Output:
     SKYCVR = A byte array containing the
              decoded sky cover data.
     IKILL =  An error flag which is set if
              the sky cover data cannot be
              decoded.
     COMMON/ERROR/IERROR (10)
     where: IERROR is an integer array
            pointing to any errors in the
            SA report.
     COMMON/ERRPTS/NDXERR, NDXTEX
     where: NDXERR = Number of errors in
                     IERROR array
            NDXTERX =   Number of free text
                        items
```

COMMENTS:

MODULE NAME: SKYRMK

PROGRAM: VREXEC

SOURCE FILE:

PURPOSE: This subroutine decodes SA remarks relating to sky cover, compass directions, and miscellaneous words.

CALLING ROUTINES: VRRMK

CALLING SEQUENCE: Call SKYRMK (WORD, LENGTH, RMK, LNRMKS, INDX, IBAD)

```
where:   WORD =  raw data word
       LENGTH =  length in bytes of raw
                 data word
          RMK =  raw remarks data array
       LNRMKS =  length in bytes of remarks
                 raw data array
         INDX =  current index position in
                 remarks raw data array.
         IBAD =  error flag
```

COMMON:

SUBROUTINES CALLED: None

SYSTEM ROUTINE REQUIRED: ILET, INUM

FUNCTION DESCRIPTION: To decode SA Remarks relating to sky cover and compass directions.

```
Input:
     WORK = A byte array containing the
            data word to be decoded.
   LENGTH = The length, in bytes, of the
            data word.
      RMK = A byte array containing the SA
            Remarks data.
   LNRMKS = The length in bytes, of the SA
            Remarks data.
     INDX = The current pointer position
            within the SA Remarks data.
     COMMON/RSTUFF/RLIST, IRNDX, NWX
     where: RLIST = A byte array containing
            the decoded Remarks
            IRNDX =  The current pointer
                     position within the
                     decoded Remarks data.
              NWX =  A flag indicating if
                     weather data were
                     decoded in the
                     subroutine VISWX.
```

Output:

The decoded skycover phrase is placed into the RLIST array and IRNDX is appropriately incremented.

IBAD = An error flag which is set if the sky cover remark cannot be decoded.

COMMENTS:

MODULE NAME:	START
PROGRAM:	VREXE:
SOURCE FILE:	DICT.MAC
PURPOSE:	Interface between the main dictionary translator, VOCAB.MAC, and VRS
CALLING ROUTINES:	VRINP
CALLING SEQUENCE:	VRINP performs a SEND with R (4) set to indicate weather, winds, or exit (see below)
COMMON:	
SUBROUTINES CALLED:	DICT
FUNCTION DESCRIPTION:	1. Performs a VRCS$ and VSDR$ to receive and send data stored in array R: R (4) = Process identifier: exit, winds, weather. R (6) = Returned error indicator. R (7) = Returned data length. 2. Calls DICT, which does the translating.
COMMENTS:	

MODULE NAME: SUBFLD

PROGRAM: VREXEC

SOURCE FILE: SUBFLD.FTN

PURPOSE: This subroutine extracts the following items from an SA report:

1. Report location identifier
2. Beginning and end points of sky and visibility/weather fields
3. Temperature, dew point, wind direction, and speed.
4. Altimeter Setting
5. Remarks starting point

CALLING ROUTINES: VRSSA

CALLING SEQUENCE: Call SUBFLD (A, ILEN, TEMP, DP, WIND, DIR, SQLL, GUST, ALTIM, LOC, IGNORE, IK, IRMK)

where:

A	=	raw data input array
ILEN	=	length in bytes of raw data array
TEMP	=	extracted temperature
DP	=	extracted dew point
WIND	=	extracted wind velocity
DIR	=	extracted wind direction
SQLL	=	extracted wind squall velocity
GUST	=	extracted wind gust velocity
ALTIM	=	extracted altimeter setting
LOC	=	location identifier
IGNORE	=	flag indicating insufficient data to process
IK	=	flag indicating error in report
IRMK	=	start position of Remarks in raw data array

COMMON:

SUBROUTINES CALLED: None

FUNCTIONAL DESCRIPTION: Besides extracting the items listed above in the calling sequence, SUBFLD also sets the following flags in the common area FLGS:
COMMON/FLGS/IWXFLG, IGFLG, IQFLG, ITFLG, IDFLG, IWFLG, IAFLG, ISPFLG, ICOFLG, IAMFLG, IAEST, IWEST, IFRAC, IVIS
of which the following are output in SUBFLD:

IGFLG = A flag which is set if wind gusts are present.

IQFLG = A flag which is set if squalls are present.

ITFLG = A flag which is set if temperature is present.

IDFLG = A flag which is set if dew point is present.
IWFLG = A flag which is set if wind speed is present.
IAFLG = A flag which is set if altimeter setting is present.
ISPFLG = A flag which is set if the report is a SA Special.
ICOFLG = A flag which is set if the report is a SA correction.
IAMFLG = A flag which is set if the report is a SA AMOS or AUTOB report.
IAEST = A flag which is set if the altimeter setting is estimated.
IWEST = A flag which is set if the wind speed is estimated.
IFRAC = A flag which is set if a fractional visibility is present.

COMMON/INDS/IVSTRT,IVEND, ISKSTR, ISKEND

where IVSTRT = Points to beginning of the visibility field in the SA report.
IVEND = Points to the end of the visibility field in the SA report.
ISKSTR = Points to the beginning of the sky cover field in the SA report.
ISKEND = Points to the end of the sky cover field in the SA report.

COMMENTS:

MODULE NAME: VDATE

PROGRAM: VREXEC

SOURCE FILE: VDATE.FTN

PURPOSE: Converts the report date (day of month) into a four digit number representing the report date in terms of year and day of year.

CALLING ROUTINES: VRSOUT, VRERR, VRSPURG

CALLING SEQUENCE:

```
Call VDATE (DAY, DATE)
where:  DAY = report day of the month date
              in byte format
       DATE = 4 digit integer value
              representing report date by
              year and day of year.  Last 3
              digits = day of year, First
              digit = last digit of current
              year, i.e. 1 = 1981
```

COMMON:

SUBROUTINES CALLED:

FUNCTION DESCRIPTION: To convert a given day of the month value into a four digit number representing the day of the year and year.

```
Input:
     DAY =  A 2-byte array containing the
            day of the month.
Output:
     DATE = An integer variable containing
            the 4-digit value representing
            the year and day of the year
            for the given day of the month.
```

COMMENTS:

MODULE NAME: VIS

PROGRAM: VREXEC

SOURCE FILE:

PURPOSE: This subroutine decodes visibility SA remarks

CALLING ROUTINES: VRRMK

CALLING SEQUENCE: Call VIS (RMK, WORK, LNRMKS, LENGTH, INDX, ING, IAEND, IRMK)

where:
- RMK = raw Remark data array
- WORD = raw data word
- LNRMKS = length in bytes of Remarks raw data array
- LENGTH = length in bytes of raw data word
- INDX = current index position in Remarks raw data array
- ING = error flag
- IAEND = length in bytes of translated SA report contained in byte array ALIST.
- IRMK = start position of Remarks in raw SA report.

SUBROUTINES CALLED: None

SYSTEM ROUTINE REQUIRED: INUM, ILET

FUNCTION DESCRIPTION: To decode visibility remarks which occur in the Remarks portion of SA report.

Input:
- RMK = A byte array containing the SA Remarks data.
- WORD = A byte array containing the data word to be decoded.
- LNRMKS = The length, in bytes, of the SA Remarks data.
- LENGTH = The length, in bytes, of the data word
- INDX = The current pointer position within the SA Remarks data.
- IAEND= The length, in bytes, of the translated main body SA report.
- IRMK = Points to the beginning of Remarks in the SA report.

COMMON/RSTUFF/RLIST, IRNDX, NWX

where: RLIST = A byte array containing the decoded Remarks.

```
IRNDX = The current pointer
        position within the
        decoded Remarks data.
NWX   = A flag indicating if
        weather data were
        decoded in the
        subroutine VISWX.
Output:
    The decoded visibility phrase is
    placed into the RLIST array and IRNDX
    is appropriately incremented..
    ING =  An error flag which is set if
           the visibility remark  cannot
           be decoded.
    COMMON/ERRPTS/NDXEER, NDXTEX
    where: NDXERR = Number of errors in
                    IERROR array
           NDXTEX = Number of free text
                    items.
    COMMON/FRTEXT/FRTEXR (40), FRTEXP (40)
    where:  FRTEXR =   An integer array
                       which points to
                       each free text word
                       in the decoded SA
                       report data.
            FRTEXP =   An integer array
                       which points to
                       each free text word
                       in the decoded SA
                       report data.
```

<u>COMMENTS:</u>

MODULE NAME: VISWX

PROGRAM: VREXEC

SOURCE FILE: VISWX.FTN

PURPOSE: This subroutine extracts and decodes the SA visibility and weather data.

CALLING ROUTINES: VRSSA

CALLING SEQUENCE: Call VISWX (A, MILES, WX, IVKILL)

where:
- A = raw data input array
- MILES = decoded visibility value
- WX = decoded weather data
- IVKILL = flag indicating error in visibility/weather field

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: To extract and decode visibility and weather data occuring in the main body of an SA report.

Input:

A = A byte array containing the SA report being decoded.

COMMON/INDS/IVSTRT, IVEND, ISKSTR, ISKEND

where:
- IVSTRT = Points to beginning of the visibility field in the SA report.
- IVEND = Points to the end of the visibility field in the SA report.
- ISKSTR = Points to the beginning of the sky cover field in the SA report.
- ISKEND = Points to the end of the sky cover field in the SA report.

Output:
- MILES = Decoded visibility value
- WX = A byte array containing the decoded weather data.
- IVKILL = An error flag which is set if the visibility/weather data field cannot be decoded.

COMMON/FLGS/IWXFLG, IGFLG, IQFLG, ITFLG, IDFLG, IWFLG, IAFLG, ISPFLG, ICOFLG, IAMFLG, IAEST, IWEST, IFRAC, IVIS......of which the following are output in VISWX:

IWXFLG = A flag which is set if weather data were decoded.

IVIS = Points to visibility mileage position.

COMMON/ERROR/IERROR (10)

where: IERROR is an integer array pointing to any errors in the SA report.

COMMON/ERRPTS/NDXERR, NDXTEX

where: NDXERR = Number of errors in IERROR array.

NDXTEX = Number of free test items.

COMMENTS:

MODULE NAME: VRRMK

PROGRAM: VREXEC

SOURCE FILE: VRRMK.FTN

PURPOSE: This subroutine extracts SA Remarks and, based upon Keyword analysis, calls appropriate subroutines for decoding. If no Keyword is found, it then determines whether the data are free text items, additive data item, PIREP, NOTAM, garbage, or error.

CALLING ROUTINES: VREXEC

CALLING SEQUENCE: Call VRRMK (A, ILEN, IRMK, ALIST, IAEND, IRKILL, NWXPASS

where:
- A = raw data input array
- ILEN = length in bytes of raw data array
- IRMK = start position of Remarks in raw data array
- IRKILL = flag indicating error in Remarks
- IAEND = length in bytes of translated message in output array ALIST

COMMON:

SUBROUTINES CALLED: RNWY, WINDS, VIS, SKYRMK, RNYCND, PCPMOD, WXMOD, PRES, LGTNG, WETHER

FUNCTION DESCRIPTION: To extract SA Remarks and, based upon Keyword analysis, call the appropriate subroutine for decoding.

Input:
- A = A byte array containing the SA report being decoded.
- ILEN = The length, in bytes, of the SA report contained in the array A.
- IRMK = Points to the beginning of Remarks in the SA report.
- NWXPASS = A flag indicating if weather data were decoded in the subroutine VISWX.

COMMON/CHKLOC/LOC

where: LOC = A byte array containing the report location identifier

Output:

ALIST = A byte array containing the decoded SA report, including Remarks.

IAEND = The length, in bytes, of the decoded SA report contained in ALIST.

IRKILL = An error flag which is set if the Remarks cannot be decoded.

COMMON/RSTUFF/RLIST, IRNDX, NWX

where: RLIST = A byte array containing the decoded Remarks.

IRNDX = The current pointer position within the decoded Remarks data.

NWX = A flag indicating if weather data were decoded in the subroutine VISWX.

COMMON/ERROR/IERROR (10)

where: IERROR is an integer array pointing to any erors in the SA report.

COMMON/ERRPTS/NDXERR, NDXTEX

where: NDXERR= Number of errors in IERROR array.

NDXTEX= Number of free text items.

COMMON/FRTEXT/FRTEXR (40), FRTEXP (40)

where: FRTEXR = An integer array containing pointers to free text items in the raw SA report.

FRTEXP = An integer array containing pointers to free text items in the decoded SA report.

COMMENTS:

MODULE NAME: VRSSA

PROGRAM: VREXEC

SOURCE FILE: VRSSA.FTN

PURPOSE: This subroutine receives a SA report from VREXEC and determines whether or not it is a SA header or a valid SA report. If it is a valid report, VRSSA calls the appropriate routines to decode it, and returns the decoded SA (excluding SA Remarks) to VREXEC. It also identifies whether or not the SA is a Special and identifies the position in the report where Remarks begin, if any exist.

CALLING ROUTINES: VREXEC

CALLING SEQUENCE: call VRSSA (ARRAY, ILEN, ALIST, IAEND, LOC, IHEAD, IGNORE, IKILL, IRMK, XWX, SPCLSA)

where:
- ARRAY = raw data input array
- ILEN = length in bytes of raw data array
- ALIST = translated message output array
- IAEND = length in bytes of translated message
- LOC = location identifier
- IHEAD = flag indicating whether or not report was a header
- IGNORE = flag indicating insufficient data to process
- IKILL = flag indicating error in report
- IRMK = start position of Remarks in raw data array
- XWX = flag indicating whether or not report contained weather data
- SPCLSA = flag indicating whether or not report was a Special SA.

COMMON:

SUBROUTINE CALLED: EXTHED, SUBFLD, VISWX, SKY

FUNCTION DESCRIPTION: Input:
- ARRAY = A byte array containing the SA report to be analyzed.
- ILEN = The length, in bytes, of the SA report contained in ARRAY.

Output:

ALIST = A byte array containing the decoded SA report, not including Remarks however.

IAEND = The length, in bytes, of the decoded SA report contained in ALIST.

LOC = A byte array containing the location identifier for the SA report.

IHEAD = A flag which is set if the report was a header.

IGNORE = A flag which is set if there were insufficient data to process.

IKILL = An error flag which is set if the SA report cannot be decoded.

IRMK = Points to the beginning of Remarks in the SA report.

XWX = A flag indicating if weather data were decoded in the subroutine VISWX.

SPCLSA = A flag indicating if the report was a Special SA.

COMMON/ZULU/HTIME, IRTIM, STIME

where: HTIME = A byte array containing the header time.

IRTIM = A byte array containing the report time.

STIME = A byte array containing the output message time.

COMMON/ERROR/IERROR (10)

where: IERROR is an integer array pointing to any errors in the SA report.

COMMON/ERRPTS/NDXERR, NDXTEX

where: NDXERR = Number of errors in IERROR array

NDXTEX = Number of free text items.

COMMON/FRTEXT/FRTEXR (40), FRTEXP (40)

where: FRTEXR = An integer array containing pointers to free text items in the raw SA report.

FRTEXP = An integer array containing pointers to free text items in the decoded SA report.

COMMENTS:

AD-A102 185 INPUT OUTPUT COMPUTER SERVICES INC WALTHAM MA F/G 17/2
TWENTY-CHANNEL VOICE RESPONSE SYSTEM.(U)
JUN 81 DOT-TSC-1313
UNCLASSIFIED FAA-RD-81-51 NL

4 of 5
ADA
102185

MODULE NAME: WETHER

PROGRAM: VREXEC

SOURCE FILE:

PURPOSE: This subroutine decodes weather SA remarks.

CALLING ROUTINES: VRRMK

CALLING SEQUENCE: Call WETHER (WORK, LN, INDX, LNRMKS, ING)

where:
- WORD = raw data word
- LN = length in bytes of raw data word
- INDX = current index position in remarks raw data array
- LNRMKS = length in bytes of remarks raw data array
- ING = flag indicating whether or not a successful weather decode occurred.

COMMON:

SUBROUTINES CALLED: None

SYSTEM ROUTINE REQUIRED: INUM, INDSTR

FUNCTION DESCRIPTION: To decode weather remarks which occur in the Remarks portion of SA reports.

Input:
- WORD = A byte array containing the data word to be decoded.
- LN = The length, in bytes, of the data word
- INDX = The current pointer position within the SA Remarks data.
- LNRMKS = The length, in bytes, of the SA Remarks data.

COMMON/RSTUFF/RLIST, IRNDX, NWX

where:
- RLIST = A byte array containing the decoded Remarks.
- IRNDX = The current pointer position within the decoded Remarks data.
- NWX = A flag indicating if weather data were decoded in the subroutine VISWX.

Output:

The decoded weather phrase is placed into the RLIST array and IRNDX is appropriately incremented.

ING = An error flag which is set if the weather remark cannot be decoded.

COMMENTS:

MODULE NAME: WINDS

PROGRAM: VREXEC

SOURCE FILE:

PURPOSE: This subroutine decodes wind SA remarks.

CALLING ROUTINES: VRRMK

CALLING SEQUENCE:
```
Call WINDS (WORD, LENGTH, ING, INDX, RMK,
LNRMKS)
where:   WORK   = raw data word
         LENGTH = length in bytes of raw
                  data word
          ING   = error flag
          INDX  = current index position in
                  Remarks raw data array
          RMK   = raw REmarks data array
         LNRMKS = length in bytes of Remarks
                  raw data array
```

COMMON:

SUBROUTINES CALLED: None

SYSTEM ROUTINE REQUIRED: INDSTR, INUM

FUNCTION DESCRIPTION:
```
To decode wind remarks which occur in the
Remarks portion of SA reports.
Input:
     WORD   =  A byte array containing the
               data word to be decoded.
     LENGTH = The length, in bytes, of the
               data word.
     INDX   = The current pointer position
               within the SA Remarks data.
     RMK    = A byte array containing the
               SA Remarks data,
     LNRMKS = The length, in bytes, of the
               SA Remarks data.
     COMMON/RSTUFF/RLIST, IRNDX, NWX
     where RLIST = A byte array containing
                   the decoded Remarks
           IRNDX = The current pointer
                   position within the
                   decoded Remarks data.
           NWX   = A flag indicating if
                   weather data were
                   decoded in the
                   subroutine VISWX.
Output:
     The decoded wind phrase is placed into
     the RLIST array and IRNDX is
     appropriately incremented.
     ING =  An error flag which is set if
            the wind remark cannot be
            decoded.
```

COMMENTS:

MODULE NAME: WXMOD

PROGRAM: VREXEC

SOURCE FILE:

PURPOSE: This subroutine decodes dispersal SA remarks such as dispersal schedule to begin/end at [time] and dispersal began/ended at [time].

CALLING ROUTINES: VRRMK

CALLING SEQUENCE: Call WXMOD (WORD, WLEN, RMK, RLEN, INDX, IERR)

where:
- WORD = raw data word
- WLEN = length in bytes of raw data word
- RMK = raw remarks data array
- RLEN = length in bytes of remarks raw data array
- INDX = current index position in remarks raw data array
- IERR = error flag

COMMON:

SUBROUTINES CALLED: None

SYSTEM ROUTINE REQUIRED: INDSTR, INUM, ILET

FUNCTION DESCRIPTION: To decode dispersal remarks which occur in the Remarks portion of SA reports.

Input:
- WORD = A byte array containing the data word to be decoded.
- WLEN = The length, in bytes, of the data word.
- RMK = A byte array containing the SA Remarks data.
- RLEN = The length, in bytes, of the SA Remarks data.
- INDX = The current pointer position within the SA Remarks data.

COMMON/RSTUFF/RLIST, IRNDX, NWX

where:
- RLIST = A byte array containing the decoded Remarks
- IRNDX = The current pointer position within the decoded Remarks data.
- NWX = A flag indicating if weather data were decoded in the subroutine VISWX.

Output:

The decoded dispersal phrase is placed into the RLIST array and IRNDX is appropriately incremented.

IERR = An error flag which is set if the dispersal remark cannot be decoded.

COMMENTS:

A.3 PDP-11/70® RETREV

MODULE NAME: ASTDMD

PROGRAM: RETREV

SOURCE FILE: RETVER.MAC

PURPOSE: Gets the first M.U requested from Block read into CSB ADDS in 'previous report' message if report old

CALLING ROUTINES:

CALLING SEQUENCE:

COMMON: CSB PARAMETERS:

$CRMUT+LMU	$BKVB
CMU	BLOCK
BRM.LN	.BKHDR
$BRMIE	.MUHDR
SAB	$DIAGB
$CRBT	
FLAG	
PMAD	
$CRBTPT	

SUBROUTINES CALLED: SENDMU
STIM
DEMAND (DMNDMU RETDMD.MAC)

FUNCTION DESCRIPTION:
1. Input: RI-CSB Address.
2. Output: MU requested is put into 11/34 send buffer.

COMMENTS: Must change EMT time addition until system value given as Greenwich mean time.

MODULE NAME:	ASTVER
PROGRAM:	RETREV
SOURCE FILE:	RETVER.MAC
PURPOSE:	Subroutine to verify requested loc from lit block - set report's available mask
CALLING ROUTINES:	RDAST
CALLING SEQUENCE:	The AST address after a read complete
COMMON:	CSB PARAMETERS: $LOCPTR LOCSIZ $CRMUT + LMU (R1) (used as count of locs at this pt must be less than 10) SAB BRM.ER .UDMOD $BRIME. $RPMSK UDBAS $DIAGP $BKBV
SUBROUTINES CALLED:	
FUNCTION DESCRIPTION:	1. Input: Address of CSB - R1. 2. Output: location verification - @ sign replaces proper loc report mask -. RPMSK - bits set for report types available. Buffer sent to last loc - next read issued if not.
COMMENTS:	

MODULE NAME: BRF 2

PROGRAM: RETREV

SOURCE FILE: RETBRF

PURPOSE: Process 11/34 Briefing Request #2; Build a Channel Response Briefing Table (CRBT) of Blocks for each report per location requested; send request accepted or error w/request acknowledgment back to 11/34.

CALLING ROUTINES:

CALLING SEQUENCE: SUSPEN (RETMAN.MAC)

COMMON: CSB PARAMETERS:

$BRMIE	$ALT
$CRMUT	$LOST
LMU	$RLOCS
GMU	FLAG, BLOCK, MUNUM
$DIAGB	$SAVCB
$DIAGP	LOCSIZ
$CRBT	$OB
$CRBTPT	FREEPL - free pool (of buffers) list head
$HOURS	

SUBROUTINES CALLED:
FDBLK
SEND
System: CDTB convert data to binary BSDR$S

FUNCTION DESCRIPTION:

1. Input: Briefing Request #2 from 11/34
   $x\ F_1/F_2/F_3\ -n_1\ -n_2\ cr$
   x = Channel #
   FI = report type 1 F_3 = FD request, n_1 = hours, n_2 alt
2. Output: CRBT the FLAG bits for SKIP type, start of report type, the BLOCK containing report requested for loc; the message unit no. slot (only 1st filled in). These three words (FLAG, BLOCK, MUNIM) are filled for each loc per report block requested.
   R1 - CSB address
   R3 - input buffer address

COMMENTS:

MODULE NAME: DBLOCK

PROGRAM: RETREV

SOURCE FILE: RETSUB.MAC

PURPOSE: Decrement map for all report blocks listed in previous briefing table for channel then clears out the RLOCS table.

CALLING ROUTINES:

CALLING SEQUENCE: SUSPEN (RETMAN.MAC)
DEMAND (RETDMD.MAC)

COMMON: CSB Parameters:
$CRBT
BLOCK
$LSTLOC
$RLOCS
.NUM No. of report types
#SA SA offset

SUBROUTINES CALLED: FDBLK

FUNCTION DESCRIPTION:
1. Input R1 - CSB Address.
2. Output Map decremented for each block in RLOCS table RLOCS table cleared.

COMMENTS:

MODULE NAME: DEMAND

PROGRAM: RETREV

SOURCE FILE: RETDMD.MAC

PURPOSE: Process all 11/34 demands for message unit data

CALLING ROUTINES:

CALLING SEQUENCE:

SUSPEN: (RETMAN.MAC) - after 1st input buffer character is decoded as '&' a demand directive
ASTDMD: (send to DMNDMU) RETREV.MAC

COMMON:

CSB PARAMETERS:

$QB	$STAG
$DIAGB	$BKVB
DIAGP	$CRBTPT
$CRBT	GMU
BLOCK	LMU
ERR.DM	MUNUM
$IOST	$MURQ
BRM.CE	CRBTSZ
$BRMIE	FLAG

SUBROUTINES CALLED:

GETCSB	SYSTEM ROUTINES
QUEUE	READ
SUSPEN	$CDTB-ASCII-to-BINARY conversion
DBLOCK	$CBDMG-Binary-to ASCII conversion
SENDMU	$CBDSG-Binary-to signed decimal magnitude

FUNCTION DESCRIPTION:

1. Input: Input buffer address.
2. Output: Check buffer for channel number and demand type key:
   A. Hang up demand,
   B. Send message unit,
   C. 'jump ahead' to message unit and send,
   D. repeat message unit demand.
   A. Decrements map values and returns to 11/34 hangup acknowledge 'A'.
   B. If message unit requested in core - send 1) channel #, 2) B-demand type, 3) message unit data; if message unit not in core, proper block is read, (AST) the stage indicator is set to 1, and message is requeued until read completed.

C. Checks if MU requested less than least message unit (LMU) in core, output same as for B - demand. If MU requested greater or equal, then skip ahead flag is checked, link flag checked and proper block read.

D. Back-up in CRBT to proper block requested and block read (AST), message requested, stage indicator set to 1.

COMMENTS: Any error in format of demand from 11/34 is sent back with error diagnostic (ERRTN).

MODULE NAME: DQUEUE

PROGRAM: RETREV

SOURCE FILE: RETSUB.MAC

PURPOSE: DEQUEUES an element from the CSB QUEUE list-head.

CALLING ROUTINES:

CALLING SEQUENCE:
RETINI (MAC)
SUSPEN - (RETMAN.MAC)
TINAST (RETAST.MAC)

COMMON: None

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:

1. Input: R3-CSB-QUEUE hold location.
2. Output: R3-CSB QUEUE address which now holds the next QUEUE link - it no more QUEUE elements CSB head and tail QUEUE list head is zero R4-QUEUE address link.
   Sets carry bit if no elements QUEUED on list head.

COMMENTS:

MODULE NAME:	ERRTN
PROGRAM:	RETREV
SOURCE FILE:	RETDMD.MAC
PURPOSE:	Routine for processing error conditions
CALLING ROUTINES:	
CALLING SEQUENCE:	DEMAND (RETDMD.MAC) RETINI - (RETINI.MAC) RDAST - (RETAST.MAC) TINAST (RETAST.MAC)
COMMON:	
SUBROUTINES CALLED:	Send system $CBDMG. Binary to ASCII decimal magnitude
FUNCTION DESCRIPTION:	1. Input: R1 - CSB address R4 - Error code buffer R5 - Error code number. 2. Output: RO - address of translation of error code.
COMMENTS:	

MODULE NAME: EXIT

PROGRAM: RETREV

SOURCE FILE: RETMAN.MAC

PURPOSE: Performs retrieval exit tasks

CALLING ROUTINES:

CALLING SEQUENCE: SUSPEN - if exit flag has been set by TINPUT upon receiving 11/34 exit directive
RETINI - if error opening or reading UDF file

COMMON:
.LINE - CSB parameter
INPFDB - UDF-DAT file descriptor block

SUBROUTINES CALLED:
GETCSB - get CSB address
DBLOCK - free blocks in RLOCS
FDBLK - free block allocate for winds. Data in CRBT - channel response block table
TINPUT - detach terminal directive

FUNCTION DESCRIPTION:
1. Input: None required.
2. Output:
   1) A send directive to 'FDRTRV' task to exit.
   2) Map decremented to free report blocks for all channels.
   3) Close UDF.DAT file.
   4) Cancel all mark-time requests.
   5) Detach terminal.

COMMENTS:

MODULE NAME: FDBLK

PROGRAM: RETREV

SOURCE FILE: RETBRF.MAC

PURPOSE: To decrement map values for FD - winds data blocks in the CRBT

CALLING ROUTINES:

CALLING SEQUENCE:
EXIT (RETMAN.MAC)
DBLOCK (RETSUB.MAC)
BRF2 (RETBRF.MAC)

COMMON:
CSB Parameters
$CRBT
BLOCK
FLAG
CRBTSZ

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION:
1. Input: R1 - CSB Address.
2. Ouptut: Map values corresponding to FD Blocks in CRBT are decremented.

COMMENTS:

MODULE NAME:	GETCSB
PROGRAM:	RETREV
SOURCE FILE:	RETSUB.MAC
PURPOSE:	Translates binary or ASCII channel number to its channel status block address
CALLING ROUTINES:	
CALLING SEQUENCE:	RETINI.MAC SUSPEN (RETMAN.MAC) EXIT (RETMAN.MAC) DEMAND (RETDMD.MAC) RCVAST (RETAST.MAC) TINAST (RETAST.MAC)
COMMON:	None
SUBROUTINES CALLED:	None
FUNCTION DESCRIPTION:	1. Input: R1 - the binary or ASCII channel #. 2. Output: R1 - the CSB address.
COMMENTS:	R1 is reserved throughout RETREV to hold this CSB address. (unless it must be changed when calling a system routine requiring R1).

MODULE NAME: MRKAST

PROGRAM: RETREV

SOURCE FILE: RETAST.MAC

PURPOSE: Set timer to check for data received for FDRTRV (this is a precautionary measure to insure all sends from FDRTRV are received since there are some 11/70 system problems with the receive AST logic)

CALLING ROUTINES:

CALLING SEQUENCE: System traps to this routine when the mark time elapses

COMMON: MARK FLAG

SUBROUTINES CALLED: RCVAST

FUNCTION DESCRIPTION:
1. Input: None.
2. Output: Resets new mark time.

COMMENTS: Uses mark time AST routines MRKT$S to continuously check for data received from 'FDRTRV'.

MODULE NAME: OUTSEND

PRORAM: RETREV

SOURCE FILE: RETBRF.MAC

PURPOSE: Perform check sum logic on buffer to be sent to 11/34 and QUEUE the buffer to be sent

CALLING ROUTINES:

CALLING SEQUENCE:
SEND (RETBRF.MAC)
SENDMU (RETBRF.MAC)

COMMON:
$IOST - CSB parameter
TINPUT

SUBROUTINES CALLED:

FUNCTION DESCRIPTION:

1. Input: R2 - Buffer address for data to be sent.
2. Output: Performs check sum logic and adds check sum characters to output buffer.

COMMENTS:

Outsend kills any pending reads to the terminal, then outputs the buffer. A terminal read is then reissued in order to receive input continuously. The checksum logic is as follows:

EXAMPLE:

&	=	46
A	=	101
CR	=	15

A) initial output buffer

46
101
15
0
0
164
7

B) output buffer with check sum characters

Figure A is the initial output buffer, with each character inserted at a byte location. The output buffer is an acknowledge of a hangup demand to 11/34. The check sum logic then appends the two null characters, the binary sum of the characters, followed by the number of characters sent, including the check sum characters - as shown in Example B.

MODULE NAME:	QUEUE
PROGRAM:	RETREV
SOURCE FILE:	RETSUB.MAC
PURPOSE:	Add buffer to QUEUE
CALLING SEQUENCE:	SUSPEN (RETMAN.MAC) DEMAND (RETDMD.MAC) TINAST (RETAST.MAC)
COMMON:	None
SUBROUTINES CALLED:	None
FUNCTION DESCRIPTION:	1. Input: R3 - QUEUE list head address - (QUEUE head & tail pointer) R4 - $QB (R1) the buffer address R1 - the CSB address. 2. Output: The QUEUE tail pointer updated to addition of buffer QUEUED the last buffer tail pointer changed to point to added buffer.
COMMENTS:	

MODULE NAME: RCVAST

PROGRAM: RETREV

SOURCE FILE: RETAST.MAC

PURPOSE: AST location for data received from 11/70 programs currently (9/1/78) only from FDRTRV

CALLING ROUTINES:

CALLING SEQUENCE: RCVAST is trap location for data received from 11/70 programs FDRTRV but is also called by MRKAST. (RETAST.MAC)

COMMON:
CSB parameters
$BRMIE
$SAVCB
BLOCK
CRBTSZ
$DIAGB
FLAG

SUBROUTINES CALLED: SEND GETSSB

FUNCTION DESCRIPTION:

1. Input: Data block of 4 words queued by 11/70 program FDRTRV word
   1 RAD50 'FDR'
   2 RAD 50 'TRV' name of sender
   3 Channel # task
   4 Block # of FD report requested by RETREV.
2. Output: Fills block # received into CRBT BLOCK LOC as pointed to by $SAVCB
   if 1st FDBLOCK received, then the output buffer containing acknowledge to 11/34 is sent.

COMMENTS:

MODULE NAME: RDAST

PROGRAM: RETREV

SOURCE FILE: RETAST.MAC

PURPOSE: The AST address after a read completes, the program vectors either for an LIT read for LOC verification or an UDF report block read for message units.

CALLING ROUTINES:

CALLING SEQUENCE: AST address after a read on UDF completes

COMMON: CSB parameters:
$IOST
$STAGE

SUBROUTINES CALLED: ERRTN ASTSKP
ASTVER
ASTDMD

FUNCTION DESCRIPTION:

1. Input: SP contains # characters transferred on read and the IO status word in CSB.
2. Output: vectors program to either
   ASTVER - verify LOC IDS
   ASTDMO - DEMAND request
   ASTSKP - skip to next briefing block.

COMMENTS:

MODULE NAME: Retrieval Constant Area

PROGRAM: RETREV

SOURCE FILE: RETCON.MAC

PURPOSE: Storage area for retrieval program

CALLING ROUTINES:

CALLING SEQUENCE: All routine use the area

COMMON:

```
The storage areas are:
19:  Channel Status Blocks - a block for
each channel line the block is described in
template file prefix.max (3200 bytes - size
per CSB)
75600 -  Freepool list head
75602 -  Freepool buffers - (41 buffers)
         Free 1 - Free 41
         Each buffer has link pointer 1
         word plus 25 words
101730 - return QUEUE list head (head &
         tail pointer two words)
101736 - IO QUEUE list head
101740 - INPFDB - UDF file descriptor block
```

FUNCTION DESCRIPTION:

COMMENTS:

MODULE NAME: RETINI.MAC

PROGRAM: RETREV

SOURCE FILE:

PURPOSE: Initialization module for program RETREV

CALLING ROUTINES:

CALLING SEQUENCE: The VRS 11/34 logs onto the 11/70 and runs RETREV the start address for RETREV IS AT BEGINNING OF RETINI

COMMON:

Channel status block parameters

```
$BKVB    MRKAST - Mark time AST address
LOCSIZ   TINPUT - Terminal QIO address
.BLKHD   FREEPL - Free pool list head
$CSBIN   TINAST - Terminal input AST address
$EVMSK
INPFDB - File Descriptor Block UDF address
CSBADR - Channel status block
PMAD - 'previous message' address
RCVAST - receive AST address
```

SUBROUTINES CALLED:

```
EXIT     SYSTEM ROUTINES:
ERRTN    WAIT        FINIT    QIO
GETCSB   SRDA$$      OPNS$M   READ
```

FUNCTION DESCRIPTION:

1) Opens UDF.DAT.
2) Gets 'previous report' messasge from block number given at zero loc in UDF LIT, stores the messagae for future use at global address PMAD.
3) Sets receive AST address.
4) Attaches terminal for RETREV task.
5) Issues another terminal read.
6) Jumps to suspend address in main body code of RETMAN

COMMENTS: The channel status block offsets are defined in the prefix file RETINI.MAC, each module of RETREV must be compiled with this module.

MODULE NAME: RETURN

PROGRAM: 11/34 VRS.

SOURCE FILE: BACKGR.MAC

PURPOSE: Routine to return address specified in US.RTN

CALLING ROUTINES:

CALLING SEQUENCE:

COMMON: All FL.***
US.***
TR.***

SUBROUTINES CALLED: TRAP TR-QUE

FUNCTION DESCRIPTION:

1. If echo-done bit is set, return one element to RDQUE.
2. In any case, restore R1 from US.SA1.
3. Jumps to address specified in US.RTN of USB.

COMMENTS:

MODULE NAME: SEND

PROGRAM: RETREV

SOURCE FILE: RETBRF.MAC

PURPOSE: Count number of characters in buffer - insert two null characters insert character count and buffer address into QIO block

CALLING ROUTINES:

CALLING SEQUENCE:
RCVAST (RETAST.MAC)
ERRTN (RETDMD.MAC)
BRF 2 (RETBRF.MAC)

COMMON: Output:
address of Q10 parameter block for output to 11/34

SUBROUTINES CALLED: (Output - QIO$ Output) System: IOKILL - kill any pending I/O to terminal OUTSND

FUNCTION DESCRIPTION:
1. Input: R1, CSB address
   R2, the output buffer address,
2. Output: The character count and buffer address in the Q O output block.

COMMENTS:

MODULE NAME:	SENDMU
PROGRAM:	RETREV
SOURCE FILE:	RETBRF.MAC
PURPOSE:	1) Compute end-of-send buffer (without two null terminator) then 2) Call outsend to perform check sum and I/O to 11/34
CALLING ROUTINES:	
CALLING SEQUENCE:	BRF2 (RETBRF.MAC) ASTDMD. (RETVER.MAC) DEMAND (RETDMD.MAC)
COMMON:	Output - Address of QIO request block
SUBROUTINES CALLED:	Output - QIO for output to 11/34
FUNCTION DESCRIPTION:	1. Input: R2 - output buffer address R3 - no of characters to send. 2. Output: the output buffer with check sum characters to be sent by 11/34.
COMMENTS:	

MODULE NAME: SNDAST

PROGRAM: RETREV

SOURCE FILE: RETAST.MAC

PURPOSE: Send AST address to resume RETREVAL, and queue next event for channel

CALLING ROUTINES:

CALLING SEQUENCE: 11/70 system traps to this address after an 11/70 - 11/34 send completes

COMMON: CSB parameters:
$IOST
BRM.BY
$BRMIE
$EVNSK
EVENT - event word for channel activity bit flags

SUBROUTINES CALLED:

FUNCTION DESCRIPTION:

1. Input: IO status block from stack pointer (computes CSB from $IOST word).
2. Output: Event word with bit set for appropriate channel busy cleared in the channel busy word $BRMIE.

COMMENTS:

MODULE NAME: SUSPEN

PROGRAM: RETREV

SOURCE FILE: RETMAN.MAC

PURPOSE: Check event flag for channel activity if yes jump to briefing request routines or demand processing if not suspend

CALLING ROUTINES:

CALLING SEQUENCE: The initialization module calls suspend initially, after that it is the suspend address called after each channel activity has been completed. Demand (RETMAN.MAC)

COMMON: Channel status block parameters:

$DIAGB	BRM.BY	$MODE	$BRMIE
$EVMSK	.UDMOD	$QUEUE	.UDBAS
$QB	$BKVB	$RPMSK	$STAGE
$RLOCS	$LOCSPTR	BRM.ER	

EVENT - double word containing bits set for each channel to be serviced

FREEPL- address of free pool list head (head & tail pointer)

SUBROUTINES CALLED:

GETCSB	DEMAND	QUEUE	SYSTEM ROUTINES
DQUEUE	DBLOCK		$CATS

FUNCTION DESCRIPTION:

COMMENTS: Inhibits AST processing while checking event flags and dequeueing an element.

MODULE NAME: TINAST

PROGRAM: RETREV

SOURCE FILE: RETAST.MAC

PURPOSE: AST address for terminal read complete

CALLING ROUTINES;

CALLING SEQUENCE: AST address upon terminal input received from 11/34

COMMON: CSB paramenters:
$QUEUE
FREEPL
$EVMSK

Event - word of channel activity bit flags

Exit FL = flag word for exit directive

SUBROUTINES CALLED:	RSUM$	RETREV	GETCSB
	QUEUE	DQUEUE	ERRTN

FUNCTION DESCRIPTION:

1. Input: Buffer queued to terminal by 11/34
2. Output:
   1. DEQUEUES buffers for particular channel if receive is a hang up directive
   2. Sets exit flag if receive is an exit directive
   3. Issues next terminal receive for continuous terminal input.

COMMENTS: TINAST performs check sum logic on receive data and checks it against the received 11/34 check sum characters (see outsend module for description of check-sum logic).

A.4 PDP-11/70® VRSOUT

MODULE NAME:	BLCR8
PROGRAM:	VRSOUT
SOURCE FILE:	BLCR8.FTN
PURPOSE:	To format the report into message unit block format
CALLING ROUTINES:	VRSOUT
CALLING SEQUENCE:	BLCR8 (ITIM, NMUS, PDICO, IPNDX, IPAIRS, IFILE, BLOCK)
COMMON:	ITIM - time of report NMS - number of message units in Block PLICO - start address of the report in common IPNDX - pointer to the report array PDICO IPAIRS - number of PTR pairs in block IFITE - report type subfile number BLOCK - the Block Buffer
SUBROUTINES CALLED:	None
FUNCTION DESCRIPTION:	1. Input: The offset in the ARRAY PDICO to the format into block format. 2. Output: The report pointers in block format that is 4 message unit headers followed by the message unit of 27 pointer pairs.
COMMENTS:	

MODULE NAME:	IOBLCK
PROGRAM:	VRSOUT
SOURCE FILE:	IOBLCK.FTN
PURPOSE:	To read/write data to UDF.DAT
CALLING ROUTINES:	VRSOUT
CALLING SEQUENCE:	CALL IOBLCK (FUNC, BLMVM, BLCK)
COMMON:	FUNC - the function to perform 1 = Read 2 = Write BLNUM - Block number to be written BLCK - the buffer to receive the block read or to be written in the UDF.DAT depending on the function requested
SUBROUTINES CALLED:	System Routines : Read - Write
FUNCTION DESCRIPTION:	1. Input: Block number function to perform buffer for block. 2. Output: The block to UDF. or the block read into buffer an error flag is returned in the function parameter - FUNC.
COMMENTS:	

MODULE NAME: NOTAVB

PROGRAM: VRSOUT

SOURCE FILE: NOTAVB.FTN

PURPOSE:

CALLING ROUTINES: VRSPURG

CALLING SEQUENCE: Call NOTAVB (LOC, IFILE, NOTBLK)

where LOC = location identifier

IFILE = 1 value for SA purge, 2 value for FT purge

NOTBLK = block number where the purge message was written in the UDF

COMMON:

SUBROUTINES CALLED: BLCR8, IOBLCK, ACTIV, DICT

FUNCTION DESCRIPTION: This subroutine inserts a "Report Not Available" message for a given locid SA or FT report into the UDF and returns the block number where it was written to the calling program, VRSPURG, for insertion in the LIT.

COMMENTS:

MODULE NAME:	SASPEC
PROGRAM:	VRSOUT
SOURCE FILE:	SASPEC.FTN
PURPOSE:	To append SA specials to the SA report for the same hour
CALLING ROUTINES:	VRSOUT
CALLING SEQUENCE:	Call SASPEC (MAP, HDDR, KB, PDICO, NP, IOLD, ITIM)
COMMON:	MAP - the address of the (global common) amp array HDDR - buffer containing first block of current report KB - the first free block available (for ichain value) PDICO - the report array IOLD = the UDF block number of current report NP - the number of PTR pairs in report ITIM = the report time
SUBROUTINES CALLED:	None
FUNCTION DESCRIPTION:	1. Input: The SA special report. 2. Output: The report appended to the current SA report, the remaining report is returned to VRSOUT for regular processing by BLCR8 - and IOBLCK.
COMMENTS:	If a report currently contains an appended report, the time is checked. If the new report is more recent it is written over the old special, and any remaining linked blocks are freed - (map value decremented).

MODULE NAME: VRSOUT

PROGRAM: VRSOUT

SOURCE FILE: VRSOUT.FTN

PURPOSE: Receives directive from VRS (processor executive) to output data to UDF.DAT file

CALLING ROUTINES:

CALLING SEQUENCE: VRSOUT is an installed task which is loaded into memory upon initial send/request/resume directive from VRS.VRSOUT then remains suspended until it receives an exit directive.

COMMON: VRS global common area
MAP - index to UDF block usage
PDICN - processed report array (ASSCII)
PDICO - translated report array (integer ptrs)
ATADII - winds data (raw)
ATADIO - winds data (translated)

SEND BLOCK RECSND/R
R1 - sender name in RAD50
R2 - sender name in RAD50
R3 - Report type
R4 - LOC-in RAD50
R5 - Translated pairs
R6 - PDICIN length
R7 - Date (day of month in ASCII)
R8 - Date
R9 - Time (time - HH-MN in ASCI)
R10- Time
R11- Time
R12- Time

SUBROUTINES CALLED: BLCR8
LOBLCK
SASPEC

FUNCTION DESCRIPTION;
1. Input: The received send-block R16 integers the report to output in PDICO.
2. Output: The report in block format chained to addition blocks is necessary and output to UDF.

COMMENTS: VRSOUT is an installed task installed by VRSINS.CMD.

MODULE NAME: VRSPURG

PROGRAM: VRSOUT

SOURCE FILE: VRSPURG.FTN

PURPOSE:

CALLING ROUTINES: VRSOUT

CALLING SEQUENCE: Call VRSPURG

COMMON:

SUBROUTINES CALLED: ZULUTM, VDATE, R50ASC, NOTAVB, ACTIV, DICT

FUNCTION DESCRIPTION: This subroutine purges from the UDF those SA reports which are more than 2 hours old and those FT reports that are more than 8 hours old.

COMMENTS:

A.5 PDP-11/70® VRSFD

MODULE NAME: VRSFD (installed task)

PROGRAM: VRSFD

SOURCE FILE: VRSFD.FTN

PURPOSE: This program retrieves and processes Winds Aloft data from the KCW.DAT file and stores it, according to a record number calculation, in the UDF for later VRS retrieval by FDRTRV.

CALLING ROUTINES: VREXEC

CALLING SEQUENCE: Called through ACTIV

COMMON:

SUBROUTINES CALLED: GTRPT, IDATE, IOBLCK, EXTSTR, RECEV

FUNCTION DESCRIPTION: To extract Winds Aloft data from the KCW.DAT file and process and store it in the UDF for later VRS retrieval by FDRTRV.

Input:

PAR = A 7 integer array passed in the ACTIV send block containing the KCW.DAT file pointers for Winds Aloft.

Output:

None

COMMENTS:

A.6 PDP-11/70® FORTRV

MODULE NAME: FDRTRV (installed task)

PROGRAM: FDRTRV

SOURCE FILE: FDRTRV.FTN

PURPOSE: To retrieve ATA winds data requested by RETREV.

CALLING ROUTINES: RETREV

CALLING SEQUENCE: Called through ACTIV

COMMON:

SUBROUTINES CALLED: R50ASC, IDATE, TIME, IOBLCK, SUMMIT, RECEV, ACTIV, BLCR8, VRECEX, DICT, RETREV

FUNCTION DESCRIPTION: This program is activated upon a Winds Aloft request from RETREV. Data received from RETREV consist of the channel number of the request, altitude requested, number of hours to departure, RAD50 representation of the locid, latitude and longitude of the locid. The program then determines the appropriate data to obtain from the UDF, interpolates the data, and creates a voice response message containing the decoded results. It then stores the message in the UDF and returns to RETREV the block number where it was stored as well as the channel number of the request.

```
Input:
     R=A16 integer word array passed in
     RECEV where:
         R(4) = channel number
         R(5) = altitude
         R(6) = number of hours to departure
         R(7) = RAD50 locid
         R(8) = latitude
         R(9) = longitude
     COMMON/VRSGLB/MAP (10240), PDICIN
     (700), PDICO (350) ATADII (160),
     ATADIO (160)
     where: MAP =  A byte array
                   representing the status
                   of the UDF.
            PDICIN = A byte array
                     containing dictionary
                     input from VRSINP.
            PDICO =  An integer array
                     containing dictionary
                     output corresponding
                     to PDICIN.
```

ATADII = A byte array containing dictionary input from FDRTRV.

ATADIO = An integer array containing dictionary output corresponding to ATADII.

Output:

R = A 16 integer word array passed in ACTIV

where: R(4) = channel number

R(5) = Winds Aloft response message location in UDF.

COMMENTS:

MODULE NAME: IOBLCK

PROGRAM: FDRTRV, VRSOUT

SOURCE FILE: IOBLCK.FTN

PURPOSE: This subroutine reads or writes a block of data from or into the UDF.

CALLING ROUTINES: Call IOBLCK (FUNC, BLNUM, BLCK)

```
where:   FUNC =  1 for read operation, 2
                 for write operation
         BLNUM = block number of data to be
                 read or written
         BLCK =  data block
```

CALLING SEQUENCE: None

COMMON:

SUBROUTINES CALLED:

FUNCTION DESCRIPTION: This subroutine reads or writes a block of data from or into the UDF.

```
Input:
     FUNC = 1 for a read operation, 2 for a
            write operation
     BLNUM =  Block number of data to be
              read or written
     BLCK =   Data block to be written.
Output:
     BLCK =   Data block read.
```

COMMENTS:

MODULE NAME: SUMMIT

PROGRAM: FDRTRV

SOURCE FILE: SUMMIT.FTN

PURPOSE: Interpolate Winds Aloft data for a requested geographical position.

CALLING ROUTINES: FDRTRV

CALLING SEQUENCE: Call SUMMIT (LVL, NDAT, SUMT, SUMX, SUMY, MASTER)

where:
- LVL = data level required (1, 2 or 3 value)
- NDAT = pressure level required within data level
- SUMT = interpolated temperature value
- SUMX = interpolated X coordinate value of the wind vector
- SUMY = interpolated Y coordinate value of the wind vector
- MASTER = UDF record 9972 containing special flag and time values for diagnosing invalid data.

COMMON:

SUBROUTINES CALLED: IOBLCK, WTFOR3

FUNCTION DESCRIPTION: This subroutine retrieves Wind Aloft data for the data level, blocks, and subsquares given in the calling statement and FDSUM labeled common. It then interpolates the data for the geographical point requested according to calculated weighting factors and returns the results to the calling program FDRTRV.

Input:
- LVL = Winds Aloft data level required (1, 2 or 3 valve)
- NDAT = Pressure level required within the data level.
- MASTER = UDF record 9972 containing special flag and time values for diagnosing invalid data.

COMMON/FDSUM/ITIME, BK1, BK2, BK3, BK4, SQ1, SQ2, SQ3, SQ4, PT1, PT2, PT3, PT4, IFOLD, IFUNK, NREAD

where: ITIME = Forecast time period required

BK1	
BK2	Grid blocks required
BK3	
BK4	
SQ1	
SQ2	Subsquares required
SQ3	
SQ4	
PT1	
PT2	Weighting factors of subsquare points
PT3	
PT4	
IFOLD =	An error flag which is set if the current Winds Aloft data are too old.
IFUNK =	An error flag which is set if the Winds Aloft data required are missing or unknown.
NREAD =	Number of disk reads required in order to compute the Winds Aloft results.

Output:

SUMT = Interpolated temperature valve.

SUMX = Interpolated X coordinate of the wind vector.

SUMY = Interpolated Y coordinate of the wind vector.

<u>COMMENTS</u>:

MODULE NAME: WTFOR3

PROGRAM: FDRTRV

SOURCE FILE: WTFOR3.FTN

PUROSE: This subroutine re-apportions the weighting factor of a subsquare point having unknown wind data amongst the three other points in order to complete interpolation of wind data within this plane.

CALLING ROUTINES: SUMMIT

CALLING SEQUENCE:

```
call WTFOR3 (PT1K, PT2K, PT3K, PTUNK)
where:   PT1K =  weighting factor of point 1
         PT2K =  weighting factor of point 2
         PT3K =  weighting factor of point 3
        PTUNK =  weighting factor of point
                 having unknown data values
```

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: This subroutine re-apportions the weighting factor of a subsquare point having unknown wind data amongst the three other points in order to complete interpolation of wind data within this plane.

```
Input:
     PT1K = Weighting factor of point 1.
     PT2K = Weighting factor of point 2.
     PT3K = Weighting factor of point 3.
    PTUNK = Weighting factor of point
            having unknown data values.
Output:
     PT1K = New weighting factor of point 1.
     PT2K = New weighting factor of point 2.
     PT3K = New weighting factor of point 3.
```

COMMENTS:

A.7 PDP-11/70® UDFPRG

MODULE NAME: UDFPRG

PROGRAM: UDFPRG

SOURCE FILE: UDFPRG.FTN

PURPOSE: To create the VRS report data file UDF.DAT

CALLING ROUTINES: Run by user to re-create the Universal Data File

CALLING SEQUENCE: None

COMMON:

SUBROUTINES CALLED: NOMESG, GETADR, WTQIO, IDATE, TIME, GETLUN, ACTIV, DICT

FUNCTION DESCRIPTION: This program creates the Universal Data File (UDF) and stores the message, "Report Not Available" within each SA and FT report location. It also inserts the special message, "Current Report Not Available, Previous Valid Report Is...." for locid '$00'. This is a special locid used by VRS Retrieval.

Input:

COMMON/VRSGLB/MAP (10240), PDICIN (700), PDICO (350), ATADII (160), ATADIO (160)

where: MAP = A byte array representing the status of the UDF.

PDICIN = A byte array containing dictionary input from NOMESG.

PDICO = An integer array containing dictionary output corresponding to PDICIN.

ATADII = A byte array containing dictionary input from FDRTRV.

ATADIO = An integer array containing dictionary output corresponding to ATADII.

Output:

None

COMMENTS:

MODULE NAME: NOMESG

PROGRAM: UDFPRG

SOURCE FILE: NOMESG.FTN

PURPOSE: To create a 'report not available' report for given location.

CALLING ROUTINES: UDFPRG

CALLING SEQUENCE:

```
Call NOMESG (LOC, SAMESG, FTMESG)
Where:   LOC = location identifier
       SAMESG = block number of SA message
       FTMESG = block number of FT message
```

COMMON:

SUBROUTINES CALLED: BLCR8, IOBLCK, ACTIV, DICT

FUNCTION DESCRIPTION:

This subroutine, called by UDFPRG, creates the message "Report Not Available" for each SA and FT report locid and the message "Current Report Not Available, Previous Valid Report Is..." for locid '$00'. It returns the block number where each message is stored to UDFPRG for insertion into the Locator Index Table.

```
Input:
     LOC = Location identifier.
     COMMON/VRSGLB/MAP (10240), PDICIN
     (700), PDICO (350), ATADII (160),
     ATADIO (160)
     where: MAP =  A byte array
                   representing the status
                   of the UDF.
         PDICIN =  A byte array containing
                   dictionary input from
                   NOMESG.
          PDICO =  An integer array
                   containing dictionary
                   output corresponding to
                   PDICIN.
         ATADII =  A byte array containing
                   dictionary input from
                   FDRTRV.
         ATADIO =  An integer array
                   containing dictionary
                   output corresponding to
                   ATADII.
         COMMON/UBLOCK/UDFBLK
         where: UDFBLK = Number of Last UDF
                         block written.
Output:
     SAMESG =    Block number of SA message.
     FTMESG =    Block number of FT message.
```

COMMENTS:

A.8 PDP 11-70® VRINIT

MODULE NAME: VRINIT

PROGRAM: VRINT

SOURCE FILE: VRINIT.FTN

PURPOSE: To initialize the VRS processor data base map and pointers

CALLING ROUTINES: Run by user at start-up time

CALLING SEQUENCE: None

COMMON:

SUBROUTINES CALLED: TIME, VRSMAP, VRSPTR

FUNCTION DESCRIPTION: This program clears and re-initializes the VRS data base map based upon current report information within the LIT and re-sets the history file pointers for SA's, FT's and Winds Aloft to their last major transmission point in the KCW.DAT file.

Input:

COMMON/VRSGLB/MAP (10240), PDICIN (700), PDICO (350), ATADII (160), ATADIO (160) of which only MAP is used.

MAP = A byte array representing the status of the UDF.

Output:

None

COMMENTS:

MODULE NAME: VRSMAP

PROGRAM: VRINIT

SOURCE FILE: VRSMAP.FTN

PURPOSE: To initialize the VRS processor data base map.

CALLING ROUTINES: VRINIT

CALLING SEQUENCE:

```
call VRSMAP (MAP)
where:   MAP =   10240 byte map array of
                 VRS which will be stored
                 in the global common VRSGLB
```

COMMON:

SUBROUTINES CALLED: None

FUNCTION DESCRIPTION: This subroutine initializes the VRS global common map. The map contains a byte corresponding to each block in the UDF. For all pre-allocated blocks in the UDF, i.e., the map, the region table, the LIT, and the Winds Aloft data blocks, the corresponding bytes of the map are set to a value of one (1). All other bytes are initialized to -1 to indicate that the blocks are free. The subroutine then scans the Locator Index Table (LIT) and sets the bytes for each block containing a report, including blocks chained for a report. If there is a discrepancy for a report block, such as a block number out of range, then all the blocks for that locator index for the report are zeroed.

```
Input:
     MAP =  A byte array representing the
            status of the UDF.
Output:
     MAP =  A byte array representing the
            status of the UDF.
```

COMMENTS:

MODULE NAME: VRSPTR

PROGRAM: VRINIT

SOURCE FILE: VRSPTR.FTN

PURPOSE: To initialize the VRS processor data base pointers.

CALLING ROUTINES: VRINIT

CALLING SEQUENCE: Call VRSPTR

COMMON:

SUBROUTINES CALLED: DTELAP, ZULUTM, TIME, GTRPT, EXTHED, EXTSTR

FUNCTION DESCRIPTION: This subroutine re-sets the history file (SF1.DAT) pointers to the last major transmission points in KCW.DAT for SA's, FT's and Winds Aloft. The method used for each report type is to back-up half a file size from the current pointer position in the KCW.DAT file and sequentially read headers until the calculated desired starting point is found.

Input:
None

Output:
None

COMMENTS:

APPENDIX B

PDP-11/34® and PDP-11/70® Line Communication

B.1 PDP-11/34 and PDP-11/70 Communications Protocol

During communications among the VRS computer, the PDP-11/34, and the Processor computer the PDP-11/70, errors occur in transmitting information over the 1200 BAUD asynchronous dedicated line. In order to recognize and eliminate these errors, two validity checks are performed on all communications. Appended to each message from the 11/70 to the 11/34 are a check-sum of two digits followed by a character count of data characters to be transmitted. Before transmitting the message to the 11/34, Retrieval sums the value of each character to be transmitted. The sixteen bit check-sum is added to the transmitted message, along with an 8-bit count of the number of characters to be transmitted. As each character is received by the PDP-11/34, its sum is added to the value of the previous characters received in a particular message. When the message is complete, the check-sum is compared to the check-sum transmitted by the 11/70. The character count is also compared. If both tests pass, the 11/34 assumes the message is correct. If a heck fails, the message is dropped on the floor. The 11/34 line timeout routine would then request the information again as the VRS software on the 11/34 never sees the errant message.

The same procedure is followed on transmissions by the 11/34 to the 11/70 with one difference: The terminal handler recognizes some character values as special, which will initiate action by RSX-11D. As a result, the check-sum characters transmitted by the 11/34 contain none of these characters. Instead, the first ten bits of the check-sum are divided into two five-bit fields and added to octal 40.

Likewise, the character count is added to octal 40. This procedure insures that no control characters are passed to the RSX-11D operating system.

In the future, the software will use a 2400 band synchronous line using a DMC-11 on the PDP-11/34 and DECNET software on the PDP-11/70 . The following sections describe how that communication will proceed. When using DECNET-DDCMP, the error checks now performed will be deleted as redundant.

B.2 PDP-11/34® --PDP-11/70® DECNET (DDCMP)

Channel Type	-	Full Duplex Synchronous
Data Code	-	ASCII and Transparent Text
Line Speed	-	2400 Baud
Error Controls	-	CRC-16 Block Parity. Block ACK/NAK procedures
Block Size	-	194 characters (including framing characters). Last block is variable in length up to 194 characters.

DATA LINK CONTROL CHARACTERS
(ASCII)

ENQ - 00000101 Octal 5 - Enquiry
SPH - 00000001 Octal 1 - Start of Header
STX - 00000000 Octal 2 - Start of Text
ETB - 00010111 Octal 27 - End of Transmission Block
ETX - 00000011 Octal 3 - End of Text
SYN - 00020220 Octal 26 - Synchronous Idle

ACK - 00000110 Octal 6 - Affirmative Acknowledgment

NAK - 00010101 Octal 25 - Negative Acknowledgment

DLE - 00010000 Octal 20 - Data-Link Escape

The first character (ENQ) is an out-of-block (not framed) character while the remaining characters enable the hardware to detect the beginning and end of data transmission.

All data transmitted must be preceded by at least three SYN characters.

Message Formats

A. Data Messages (1st and intermediate blocks)

character #:	1	2	3	4	5	190	191	192	193	194
message:	0	SOH	N	DLE	STX	Transparent Text Data	DLE	ETB	BCC

Data Messages (last block)

character #:	1	2	3	4	5	K	K+1	K+2	K+3	K+4
message:	0	SOH	N	DLE	STX	Transparent Text Data	DLE	ETX		BCC

where K + 4 = 194

B. Acknowledgment Message

character #:	1	2	3	4	5	6
message:	0	SOH	N	ACK/NAK	ETX	BCC

C. Line Synchronization Messages

1
0 ENQ

where:

0	- Required number of SYN characters
SOH	- Start of header character
N	- Block sequence number (0-9)-1 ASCII character
DLE STX	- Start of Transparent text characters
DLE ETB	- End of intermediate transparent text characters
DLE ETX	- End of transparent text message characters
BCC	- Block check characters (CRC-16; 2 characters)
ACK	- Affirmative acknowledgment character
NAK	- Negative acknowledgment character
ENQ	- Enquiry character

The block check character (BCC) is used to provide a block data integrity check. It is a cyclic-redundancy check (CRC-16)* that uses an arithmetic accumulation that is reset

*See Section B.6.

with the SOH character in the transmission, and restarted with the character following. Thereafter, all characters in the transmission up to and including the ETB or ETX character are included in the CRC calculation. Within blocks of transparent text, the first DLE character of all two-character DLE sequences is excluded from the BCC.

B.3 Transparent-Text Mode

This mode permits greater versatility in the range of coded data that can be transmitted. This is because all data, including the normally restricted data-link line-control characters, are treated only as specific bit patterns when transmitted in transparent mode. Thus, unrestricted coding of data is permitted for transparent-mode operation. This mode is particularly useful for transmitting binary data and unique specialized codes.

Any data-link control characters transmitted during transparent mode and required to be effective must be preceded by a DLE. Thus, the following sequences are effective during transparent-mode operation:

SEQUENCE	USE
DLE STX	Initiates the transparent mode for the following block of data.
DLE ETB	Terminates a block of transparent data, returns the data link to ASCII mode, and calls for a reply.

DLE ETX — Terminates the transparent data, returns the data link to ASCII mode, and calls for a reply.

DLE ENQ — Indicates a "disregard this block of transparent data" and returns to ASCII mode.

DLE DLE — Used when a bit pattern equivalent to DLE appears with the transparent data to permit transmission of the DLE as data.

All replies, inquiries, and headers are transmitted in ASCII mode. Transparent data are received on a character-by-character basis; thus, character phase is maintained in the usual manner.

NOTE: ASCII data may also be transmitted in ASCII mode by omitting the DLE character from the data link control sequences - DLE STX, DLE ETB, DLE ETX, etc.

B.4 General Transmission Procedures

Each data block transmitted and received will be acknowledged when feasible. The acknowledgment may be a positive ACK or negative NAK. A positive ACK is sent if the following conditions are met:

1. The block size is correct.

2. The SOH/STX and ETB/ETX characters are proper (valid and expected).

3. The BCC is correct.

4. The block sequence number is correct.

Each time a center is forced into a cancel mode during a transmission regardless of the reason, the ENQ procedure will be initiated before the next transmission is started.

If the center receives an ENQ after the start of a data transmission (on input) and prior to an end transmission character (ETX) it will treat the ENQ as a cancel transmission request from the transmitting center.

B.4.1 Output Timing

A center establishes a timeout value of 5.9 seconds for every block transmitted. If the receiving center does not acknowledge receipt of the block before the timeout is detected, an automatic block return procedure is invoked. The timeout value increases to one minute for ETX blocks with the same block rerun procedure when a timeout is experienced.

If any of the above conditions are not met, the center will either transmit a negative acknowledgment (NAK) or refuse to respond, forcing the transmitting center to rerun the block when expected acknowledgment is overdue.

B.4.2 Block Acknowledge Procedures

A center will transmit an ACK or NAK reply block for every block received. The data block ACK/NAK format is the same as the ENQ response except for the content of the N field. That is, for data block acknowledgment the N field of the reply block contains the block number being acknowledged (ACK or NAK) whereas, for an ENQ response, the N field is always ASCII zero.

B.4.3 Block Rerun Procedures

Data blocks are retransmitted every time a center receives an NAK acknowledgment from the other center or when no acknowledgment is received within the allotted time (5.9 seconds NON-ETX blocks; 60 seconds for ETX blocks). If an NAK or data timeout occurs three times for the same data block, the center initiates a cancel and returns to the ENQ procedure. If a message is retransmitted three times without success, it is aborted. When a message abort procedures are used, the center will generate a printout (3NAK) and continue with the next message available for transmission.

B.4.4 Block Transmission Procedures

A center will stop transmitting when a persistent error condition has been detected. When a positive acknowledgment is received, the center will resume transmission.

B.5 Line Synchronization Procedures

A center will initiate an ENQ procedure to determine circuit viability an operational interface capability with the other center. The format for the ENQ transmission is:

character #: 1
message: 0 ENQ

where 0 represents the required SYN character sequence.

The SYN characters are followed by a single ASCII ENQ character. The ENQ sequence is sent at one second intervals until two consecutive positive replies are received. After 150 unanswered ENQ's have been transmitted, the center will generate a printout indicating a possible line problem exists. The center takes no other action at this time and continues to ENQ the other center. (It should be noted here that the other center has a similar responsibility regarding the transmission and acknowledgment of the ENQ procedure).

The format for the response to the ENQ block is:

character #:		1	2	3	4	5,6
message:	0	SOH	N	ACK/NAK	ETX	BCC

All ENQ reply blocks are framed with SOH and ETX control characters. The rule which governs BCC generation for data blocks is also valid for reply blocks. The N field is always an ASCII zero when responding to an ENQ. If the center is not in an operational mode that would permit a large volume of data transfers on the circuit, a NAK responds is sent to the ENQ. The center receiving the NAK response must withhold the transmission of the next ENQ for thirty seconds.

B.6 Cyclic Redundancy Checking (CRC-16)

Cyclic Redundancy Checking (CRC-16) is a sophisticated method of block checking a data stream. This type of checking involves a polynomial division of the data stream by a CRC polynomial. The 1's and 0's of the data become the coefficients of the dividend polynomial while the CRC polynomial is present at X + X + dX + 1. The division uses subtraction modulo 2 (no carries) and the remainder serves as the Cyclic Redundancy Check. The receiving station compares the transmitted remainder with its own computed remainder and an equal condition indicates that no error has occurred.

APPENDIX B REFERENCES

1. MITRE document entitled "WMSC High Speed Interface Procedures," Dec. 1975.

2. Digital Data Communications Message Protocol, Dec. 10, 1974.

APPENDIX C

PDP-11/70® SOFTWARE FLOW DIAGRAMS

C.1 VREXEC

VRS PROCESSOR FLOWCHART

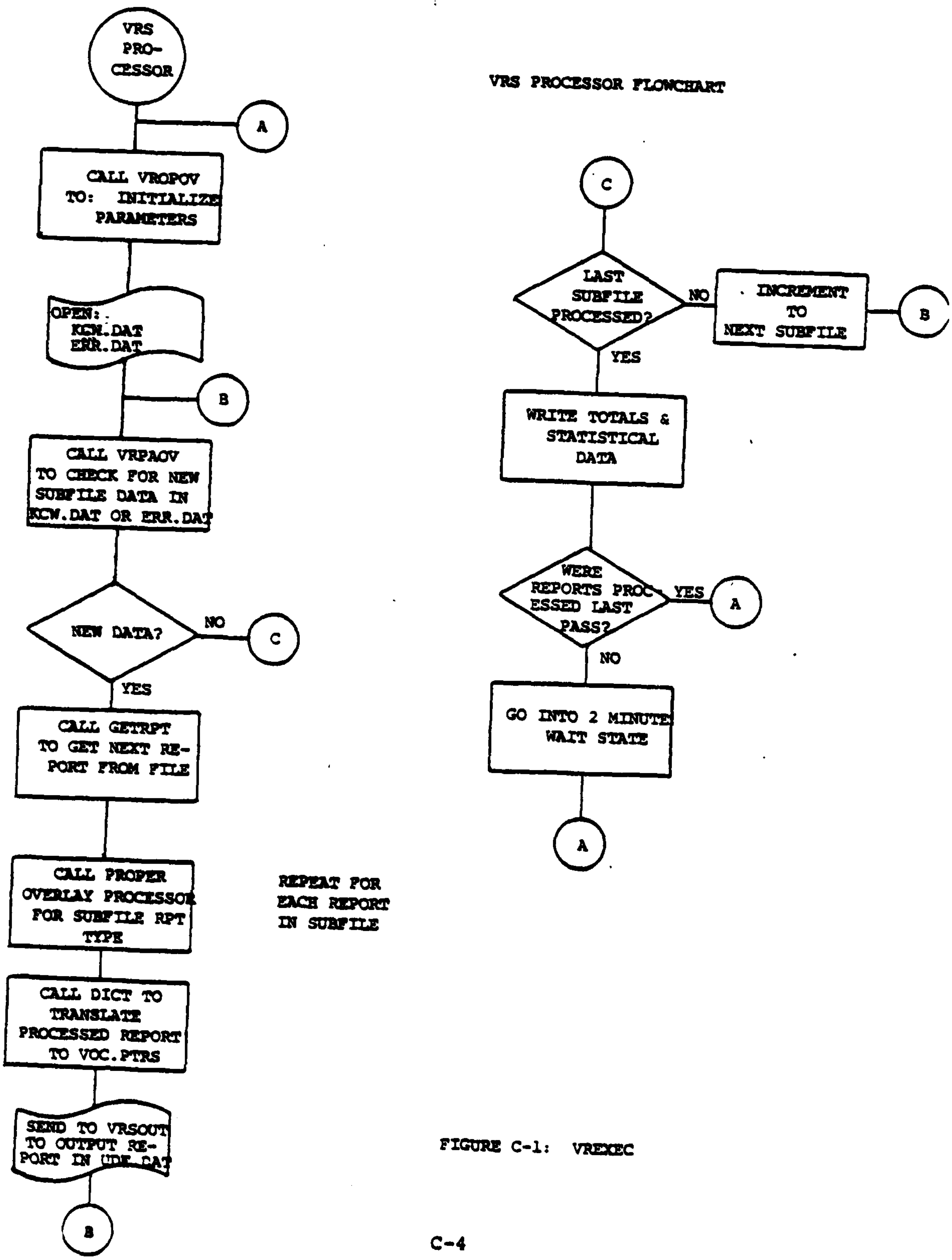


FIGURE C-1: VREXEC

C.2 VRSOUT

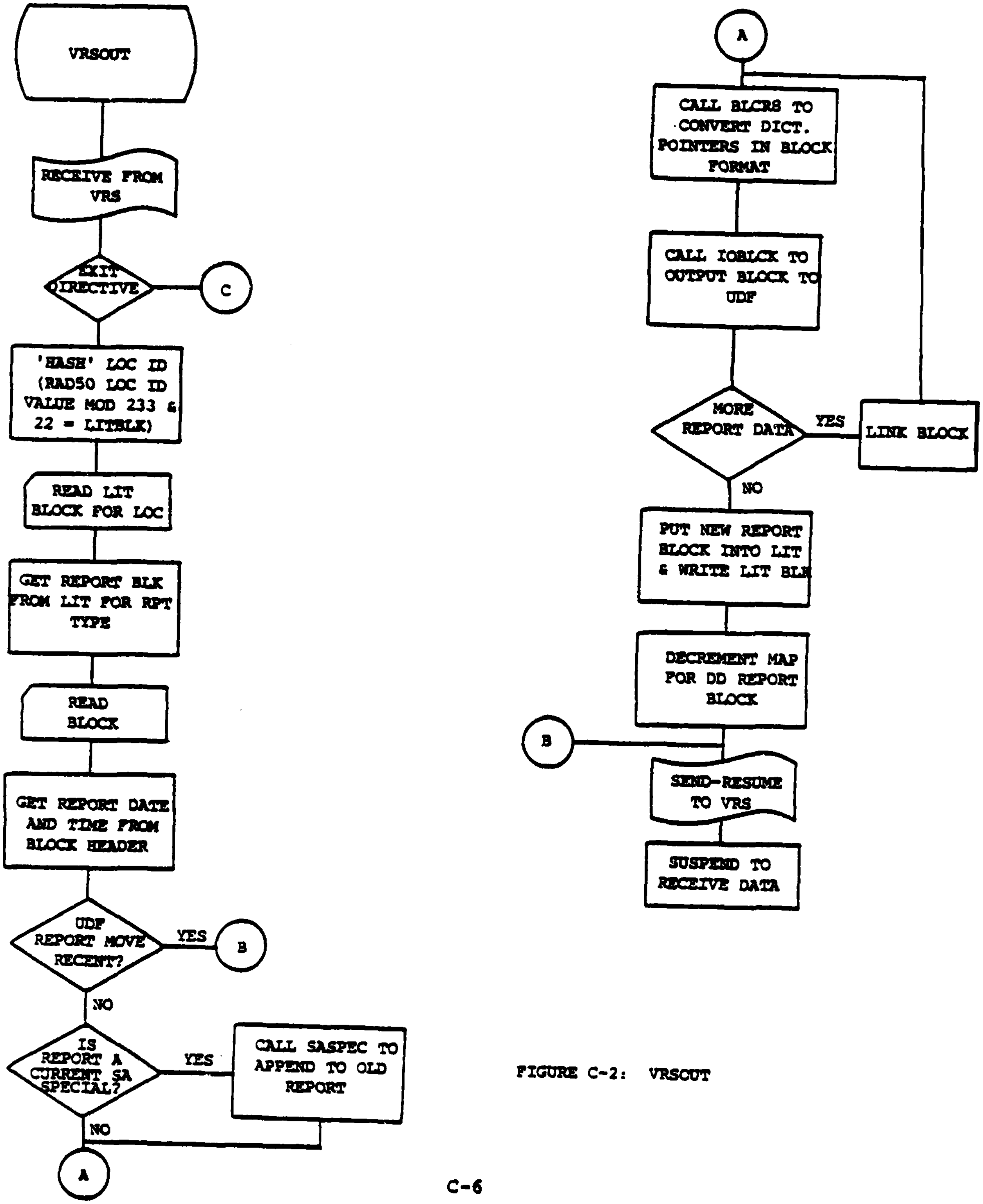


FIGURE C-2: VRSOUT

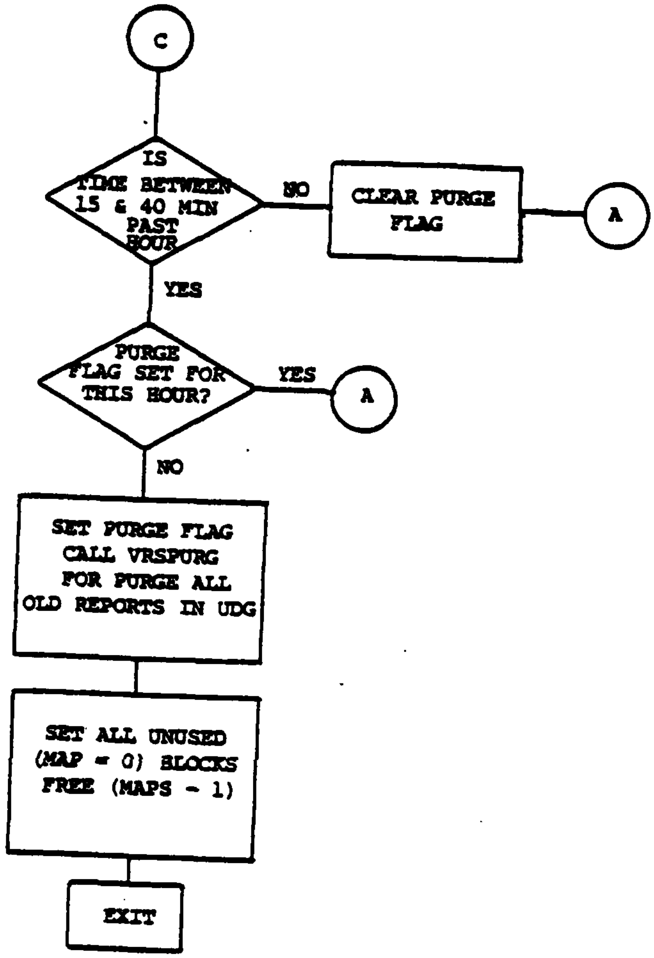


FIGURE C-2: VRSOUT (Cont'd.)

C.3 SA PROCESSOR

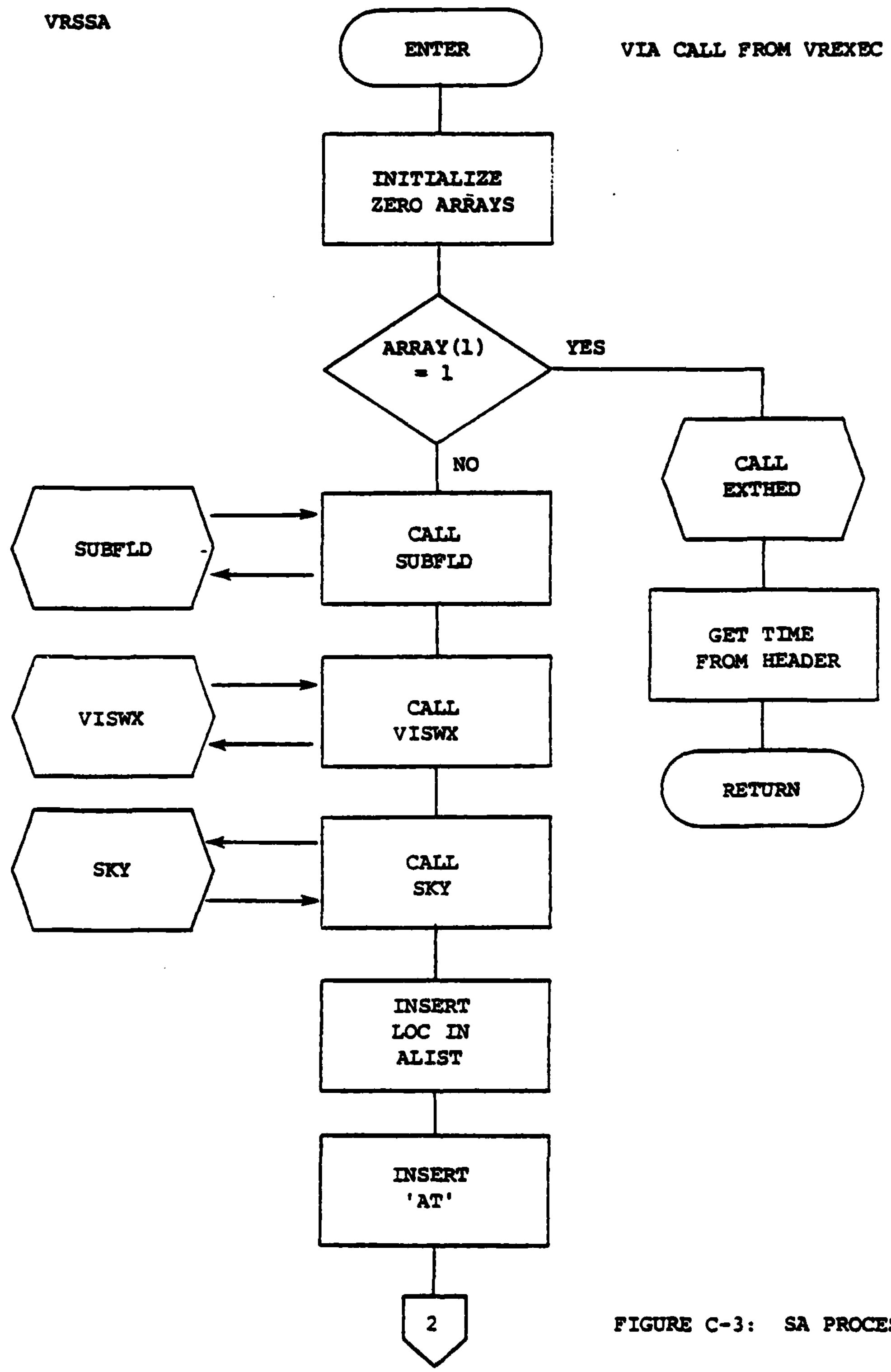


FIGURE C-3: SA PROCESSOR

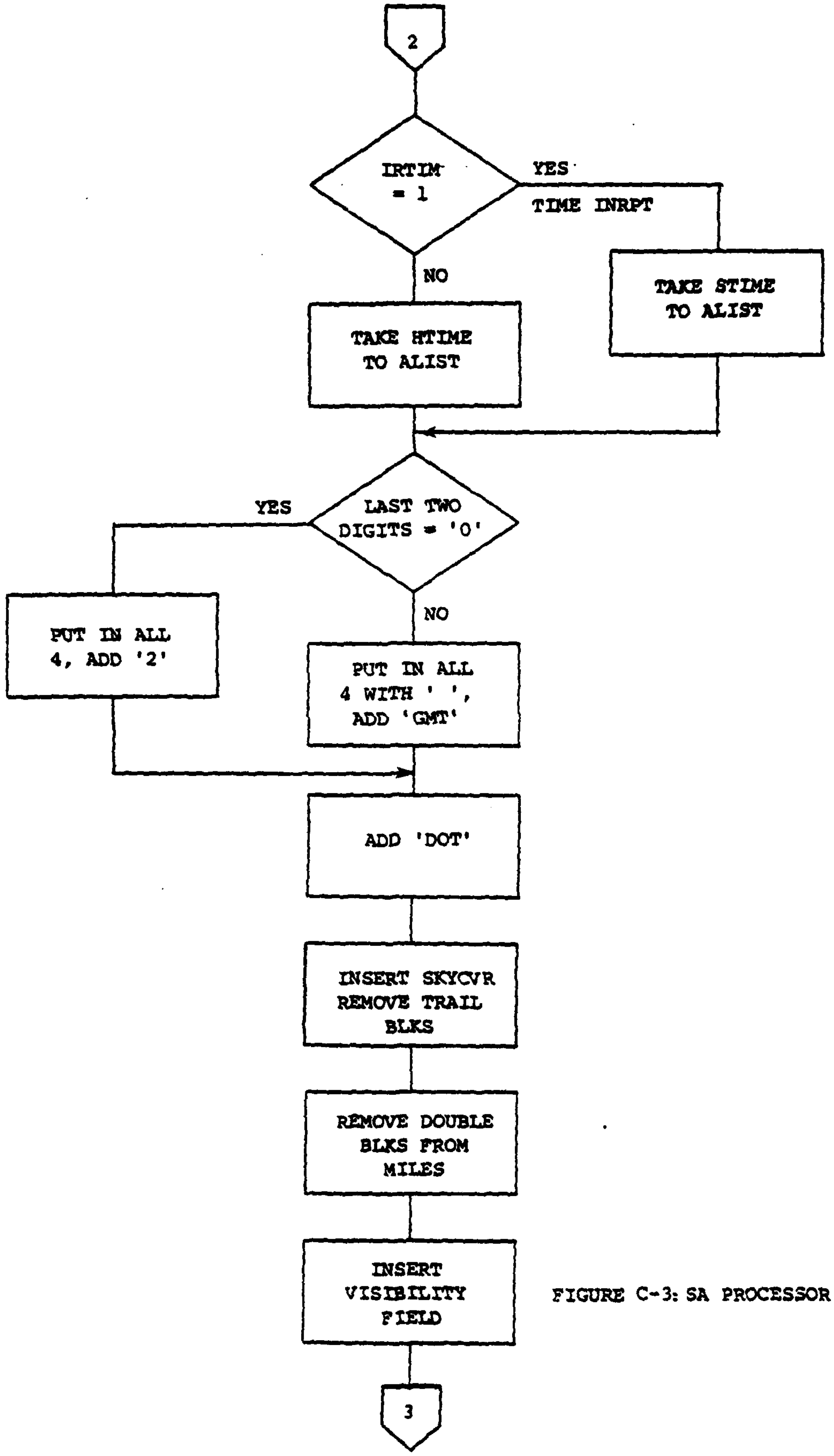


FIGURE C-3: SA PROCESSOR (Cont'd.)

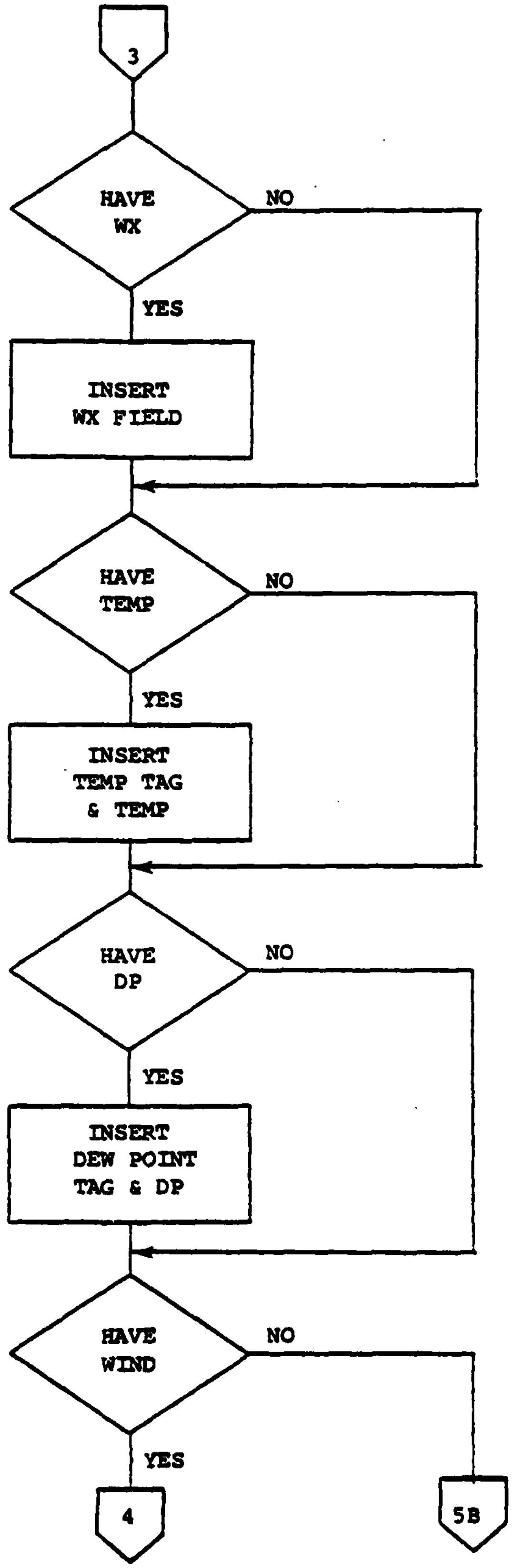


FIGURE C-3: SA PROCESSOR (Cont'd.)

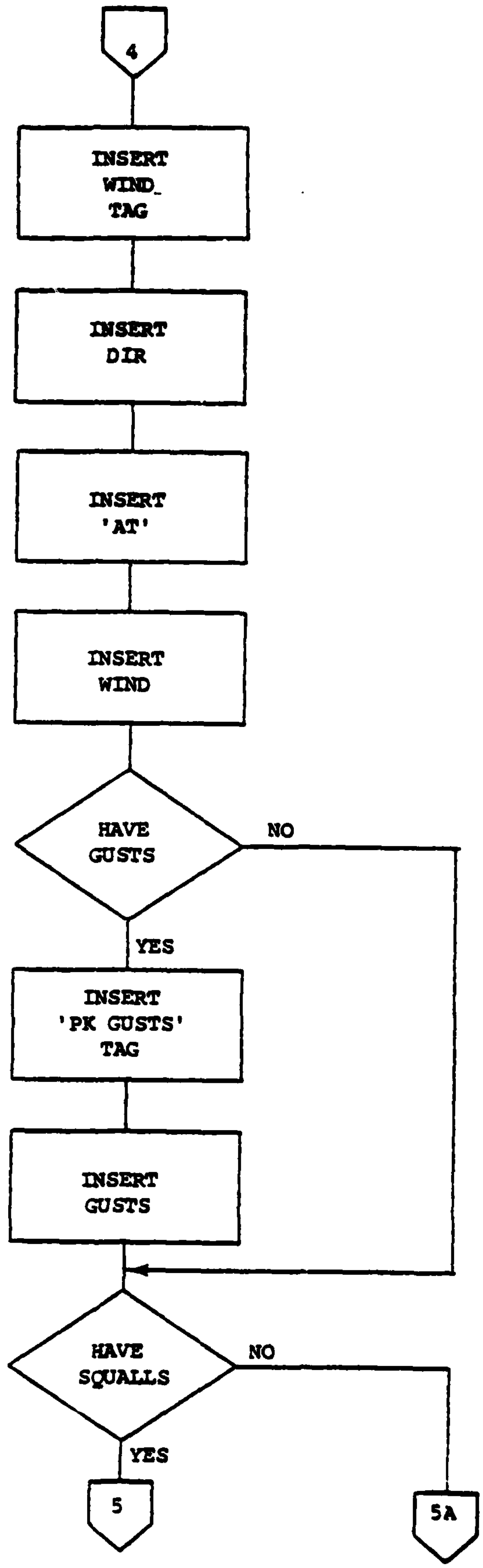


FIGURE C-3:
SA PROCESSOR (Cont'd.

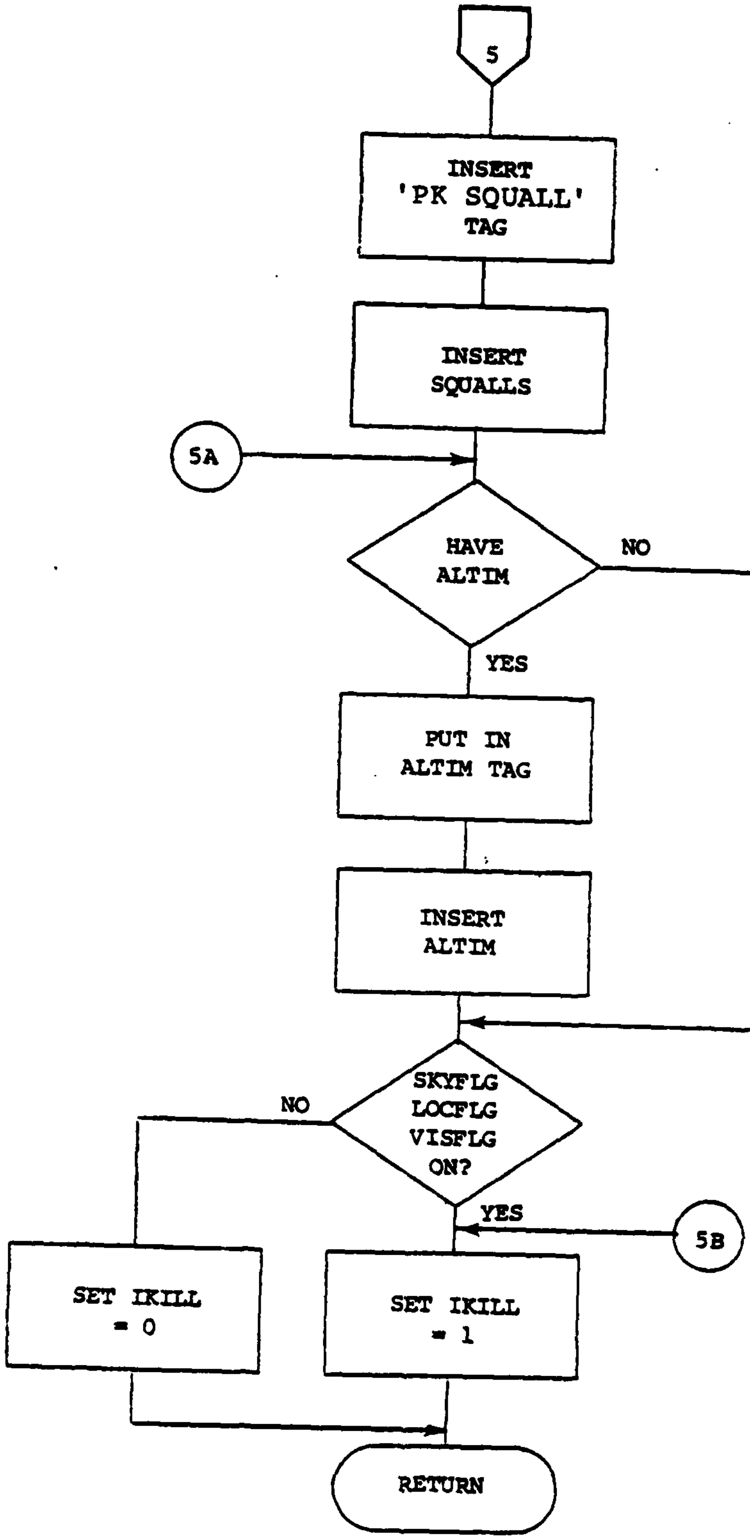


FIGURE C-3: SA PROCESSOR (Cont'd.)

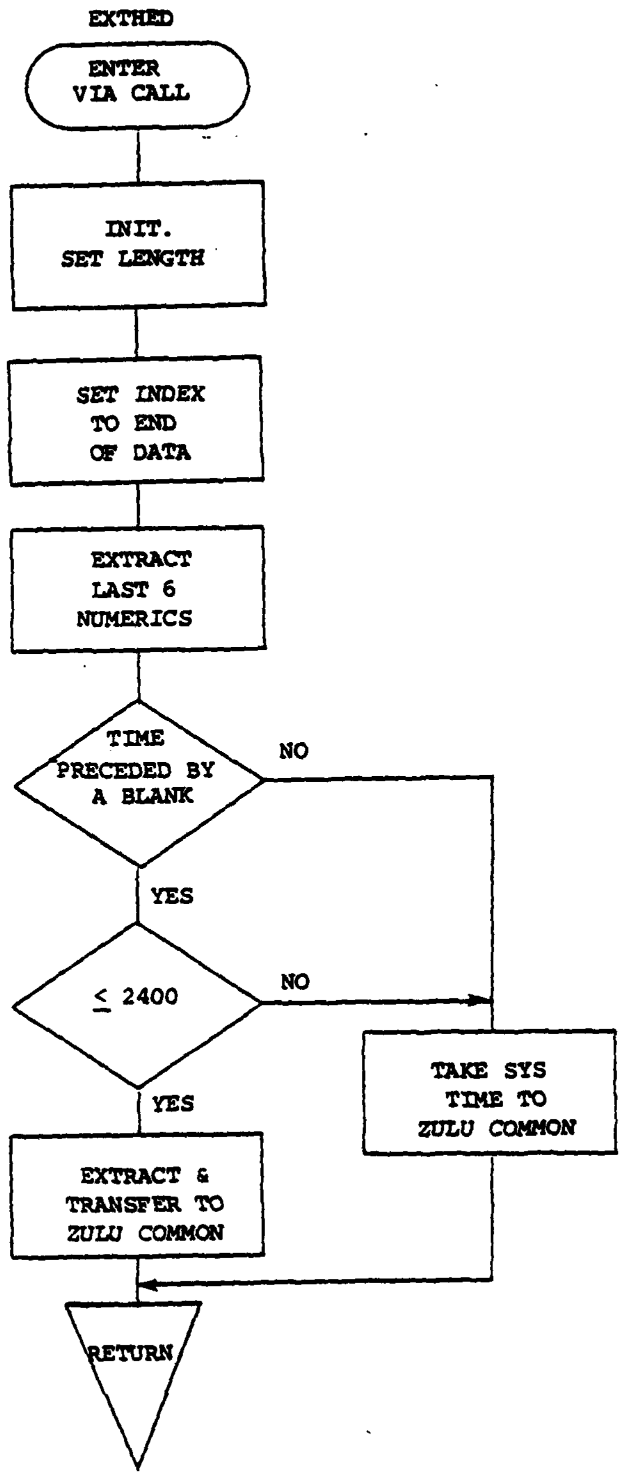


FIGURE C-3: SA PROCESSOR (Cont'd.)

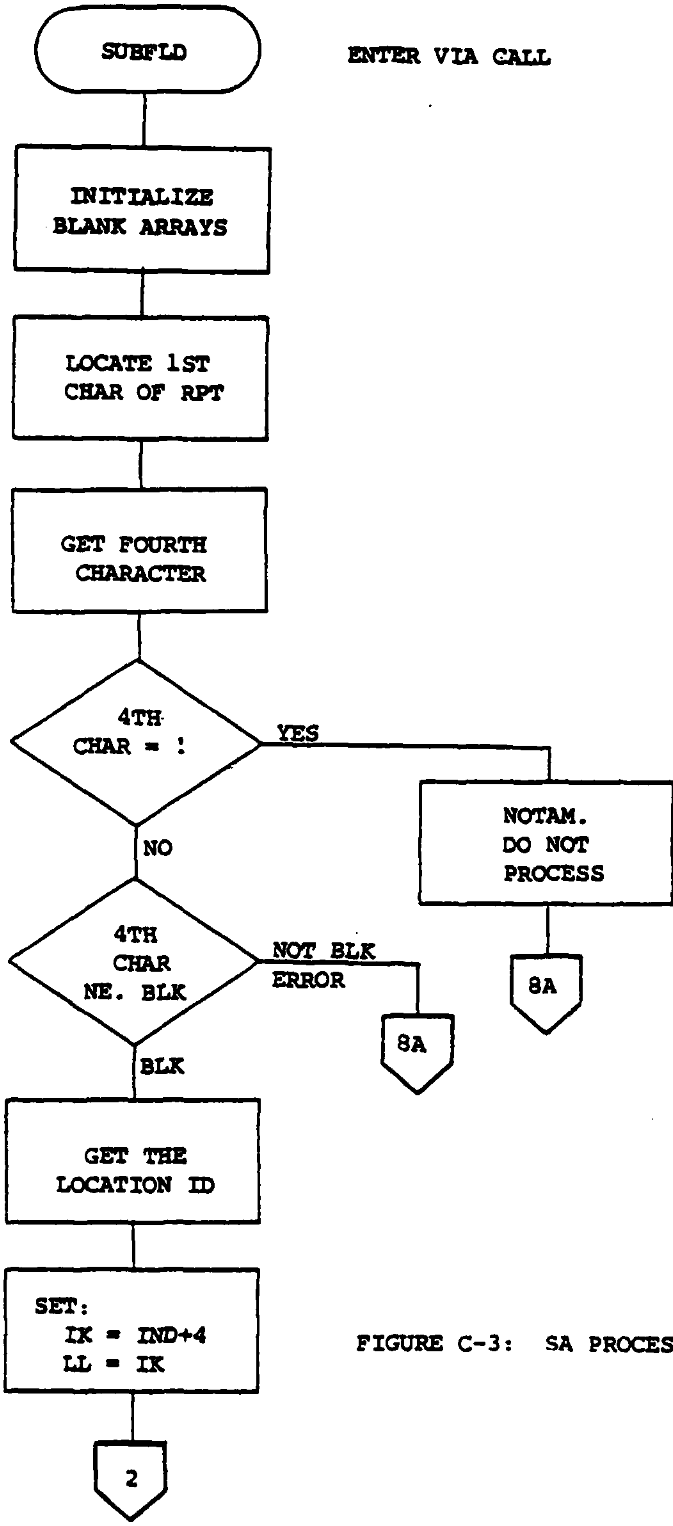


FIGURE C-3: SA PROCESSOR (Cont'd.)

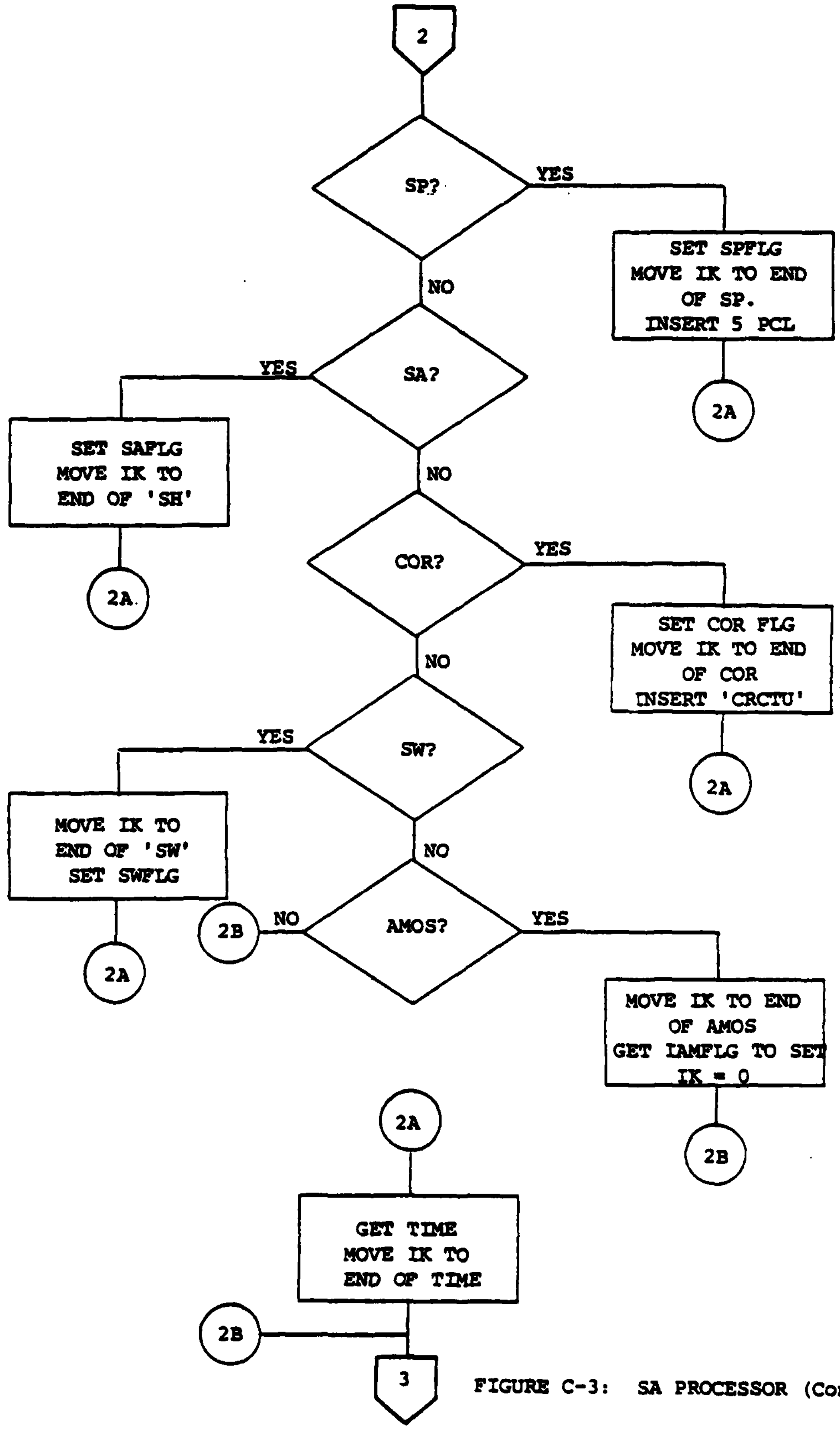


FIGURE C-3: SA PROCESSOR (Cont'd.)

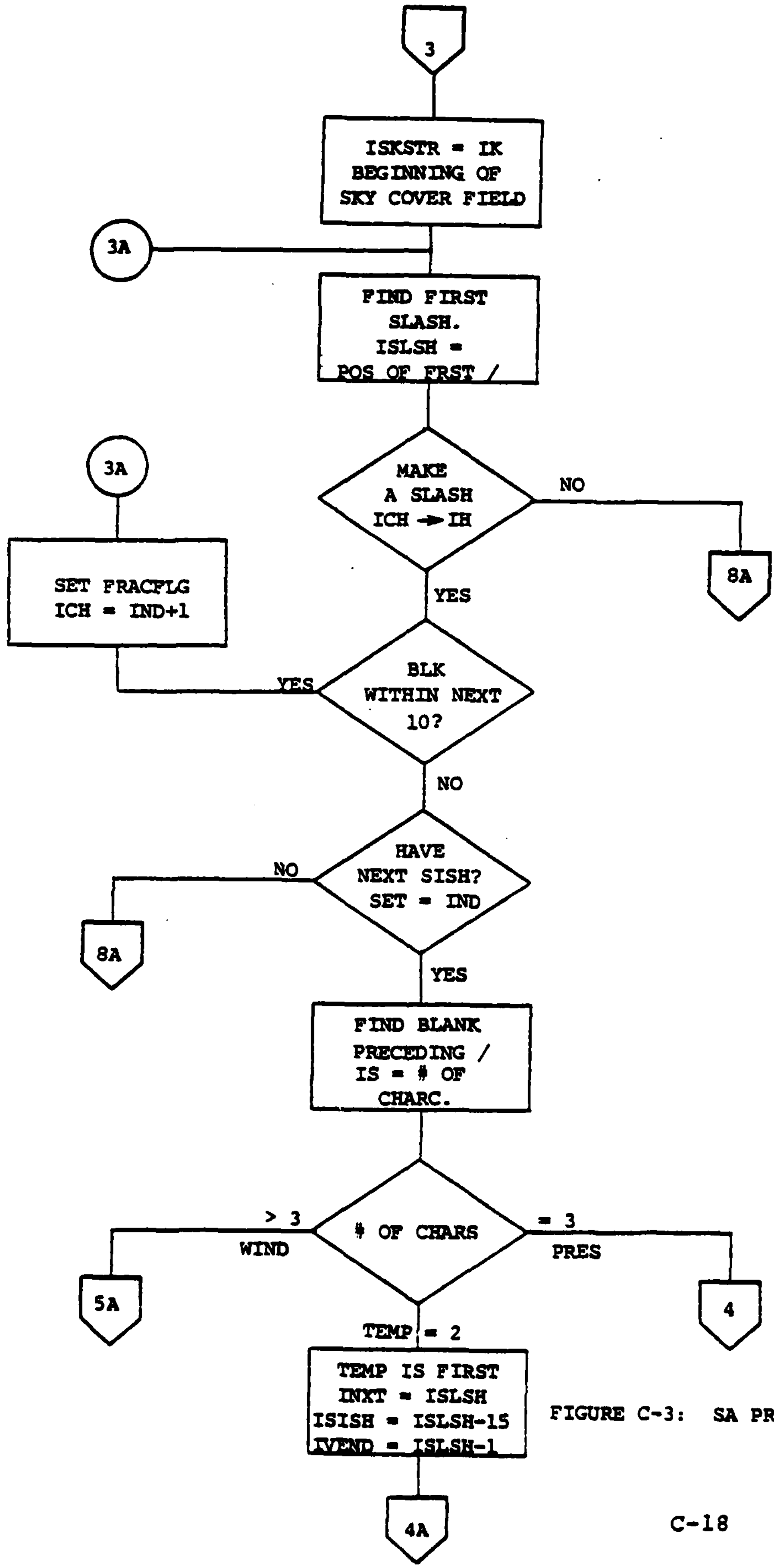


FIGURE C-3: SA PROCESSOR (Cont'd.)

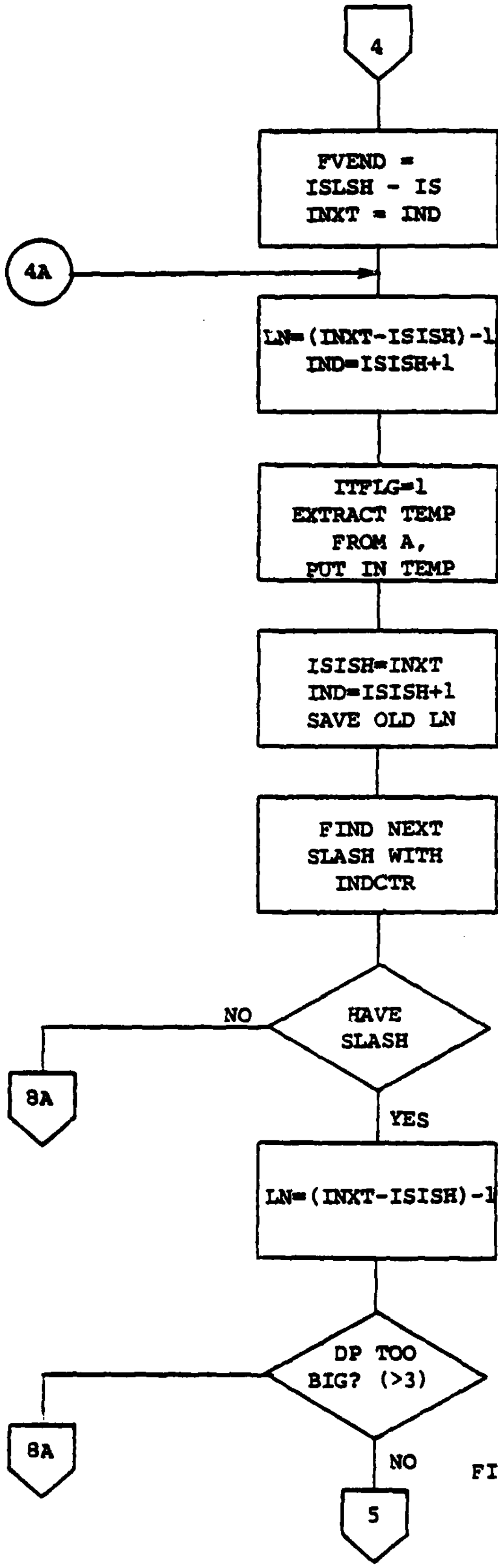


FIGURE C-3: SA PROCESSOR (Cont'd.)

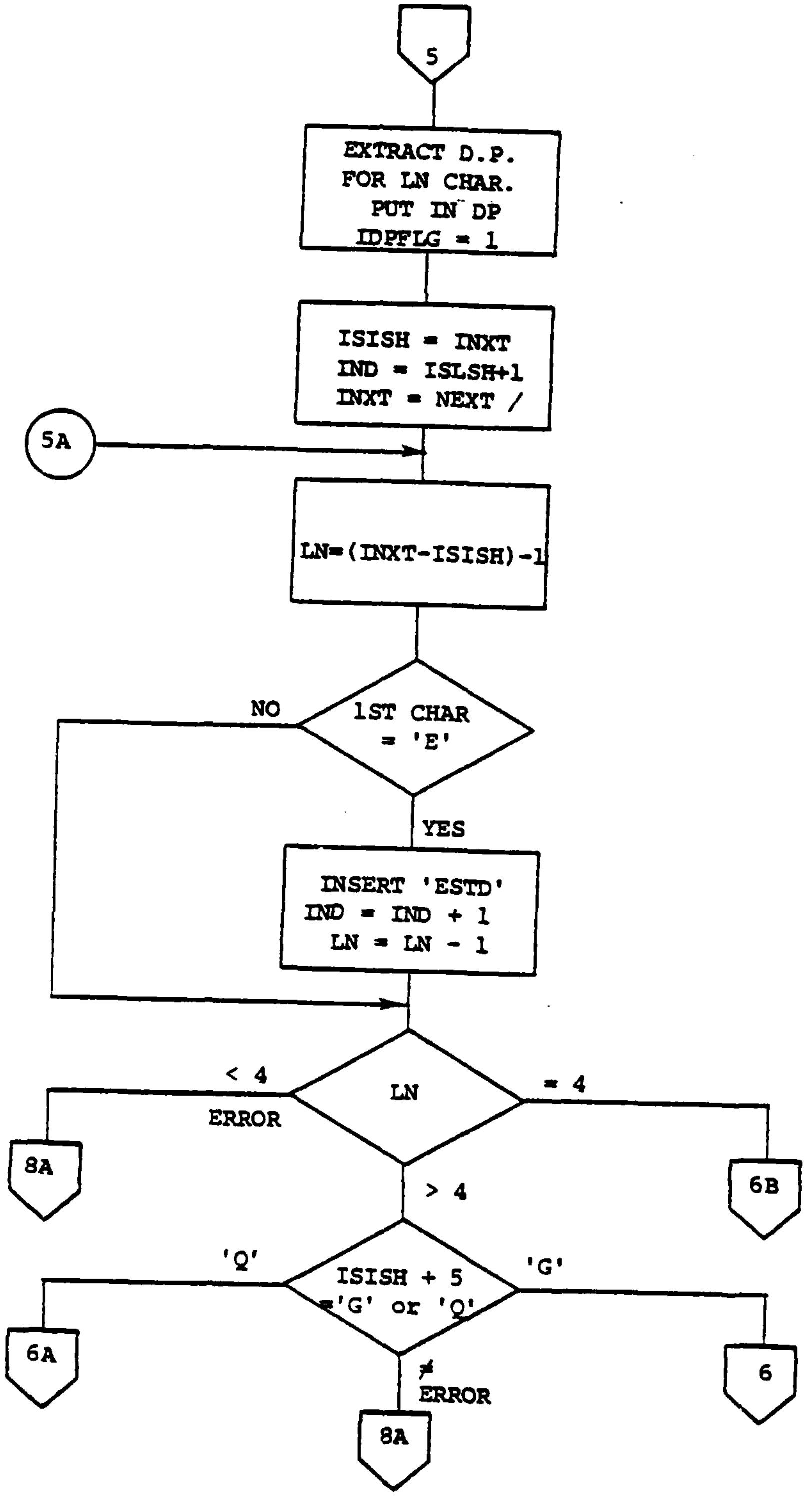


FIGURE C-3: SA PROCESSOR (Cont'd.)

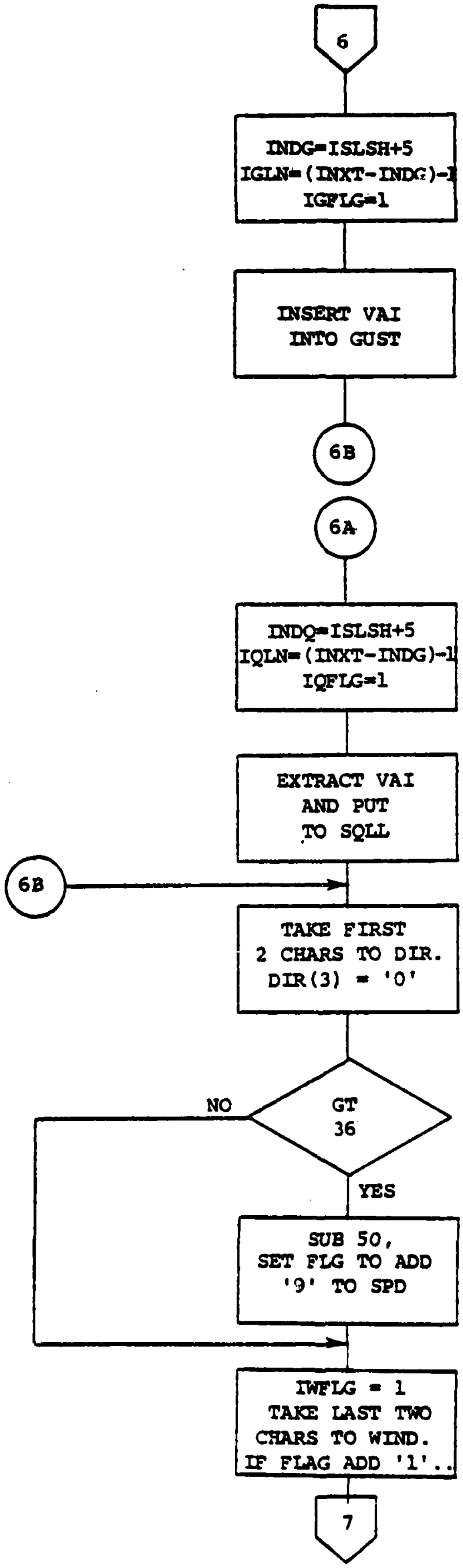


FIGURE C-3: SA PROCESSOR (Cont'd.)

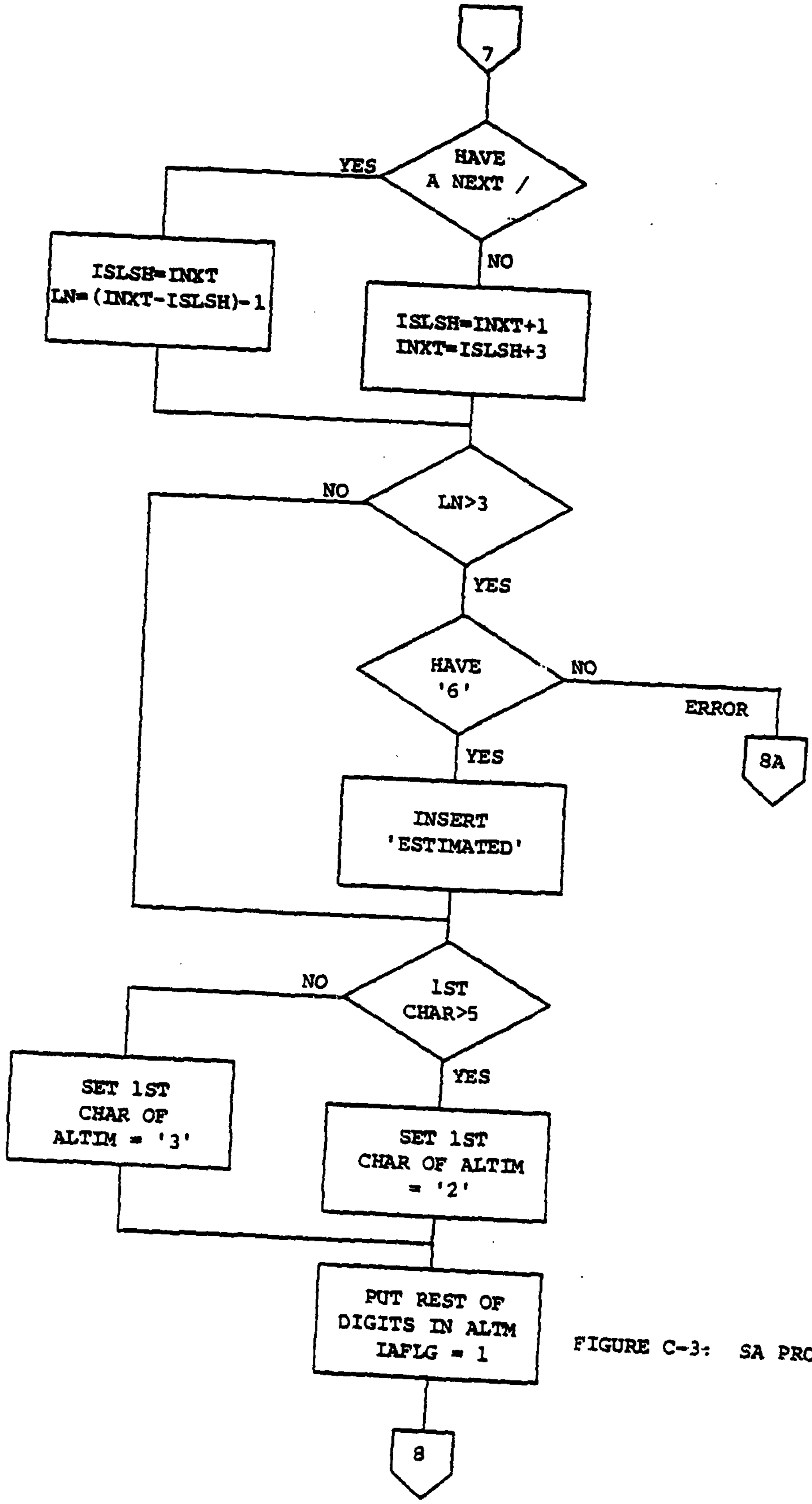


FIGURE C-3: SA PROCESSOR (Cont'd.)

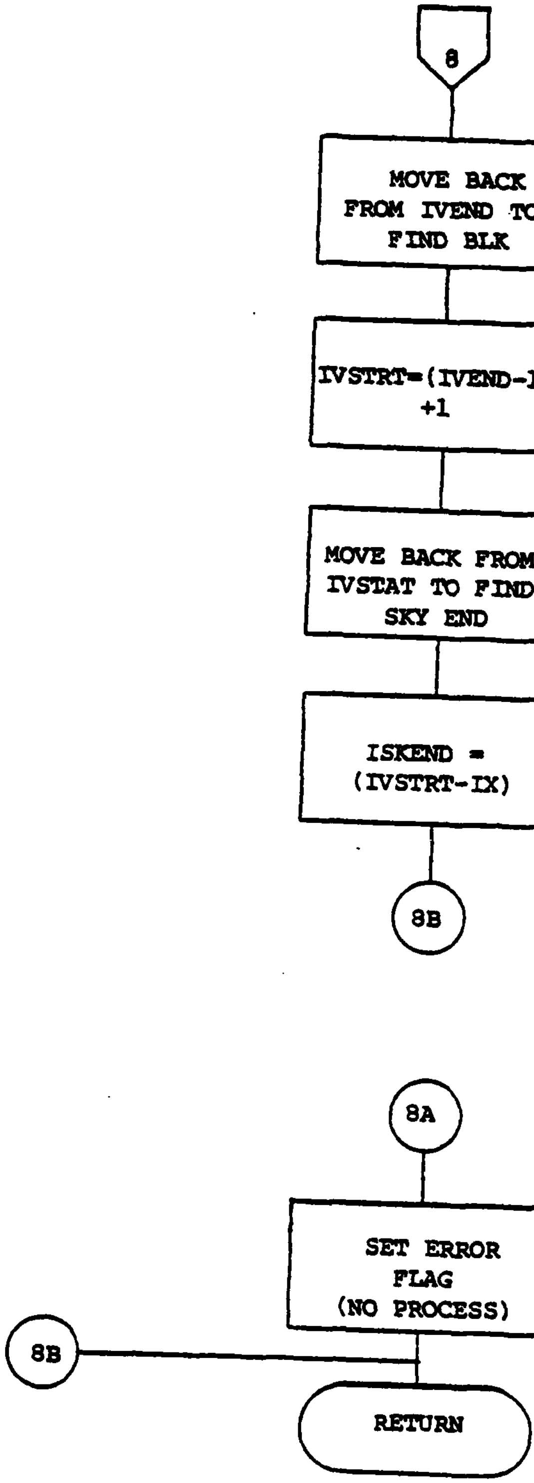


FIGURE C-3: SA PROCESSOR (Cont'd.)

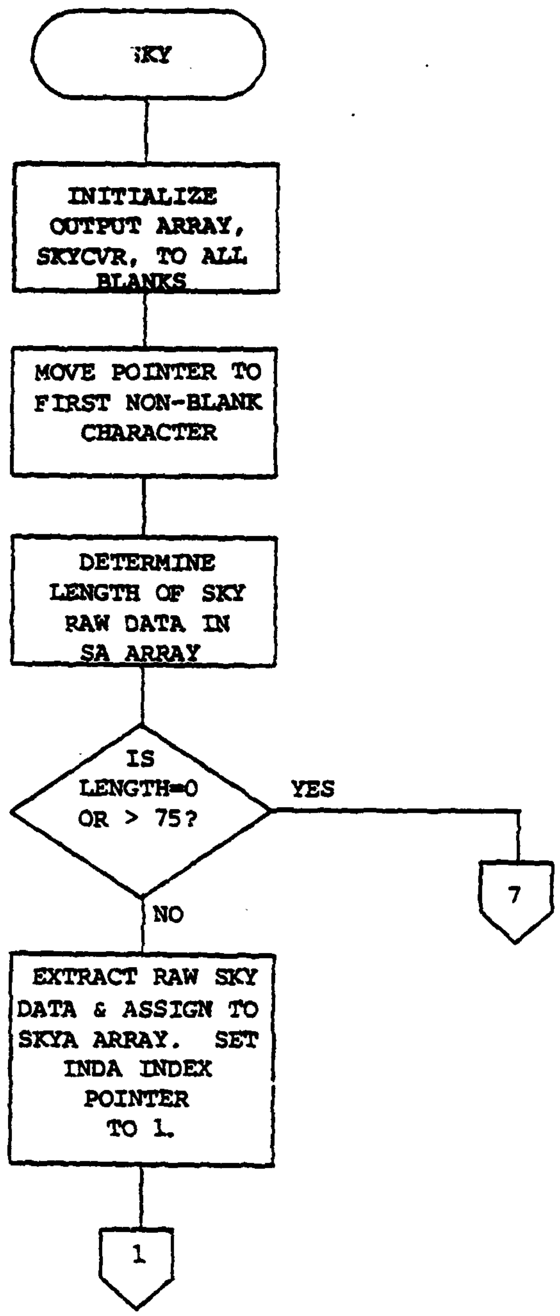


FIGURE C-3: SA PROCESSOR (Cont'd.)

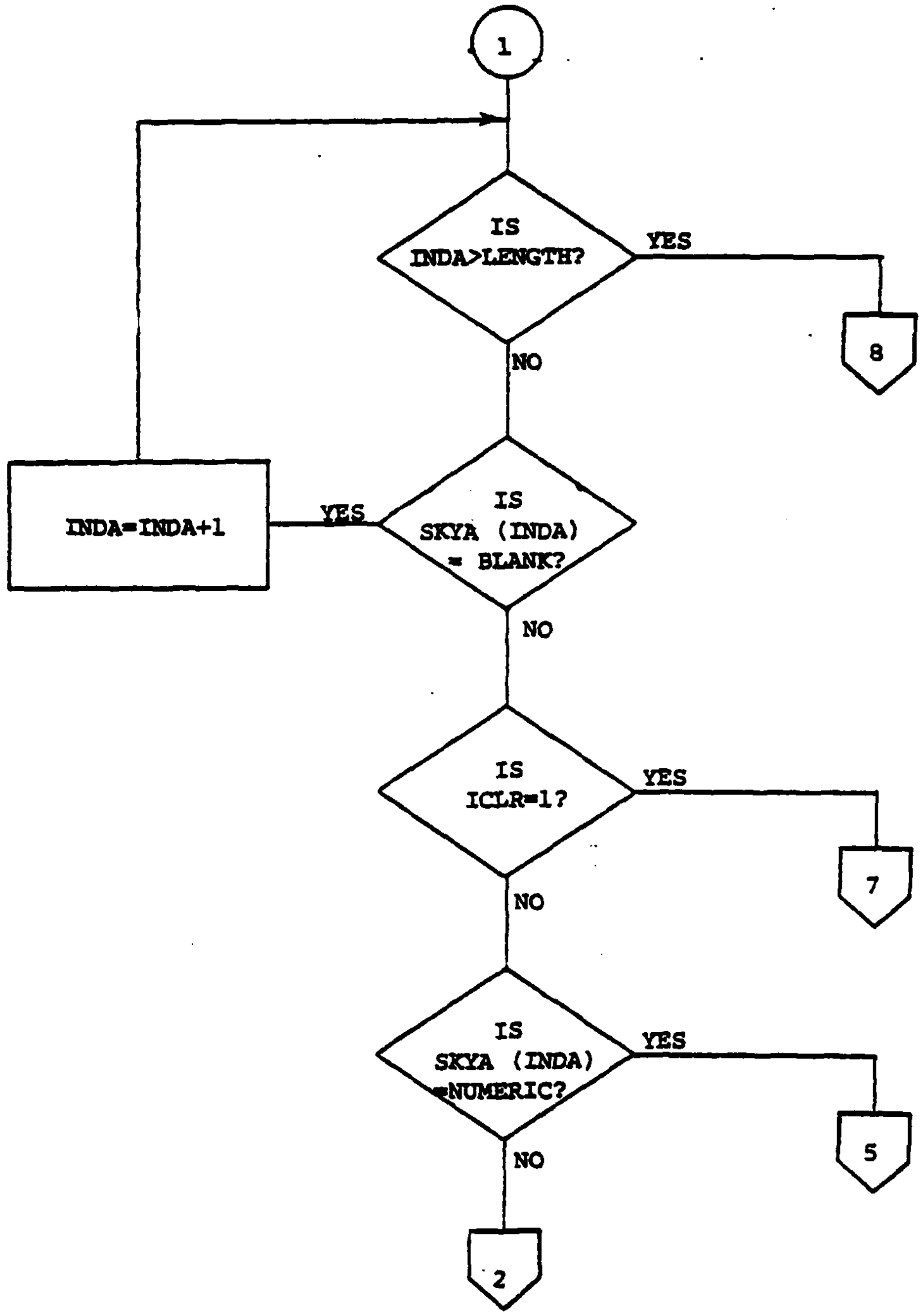


FIGURE C-3: SA PROCESSOR (Cont'd.)

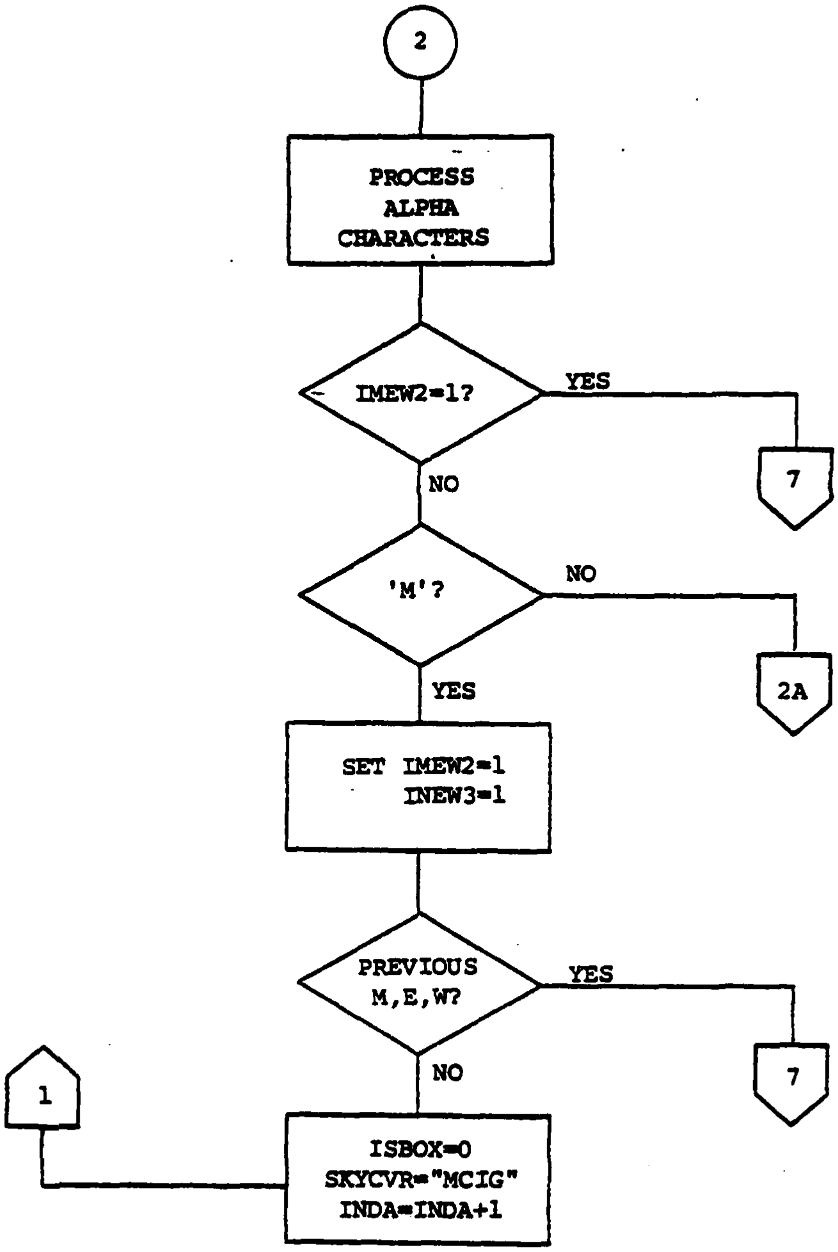


FIGURE C-3: SA PROCESSOR (Cont'd.)

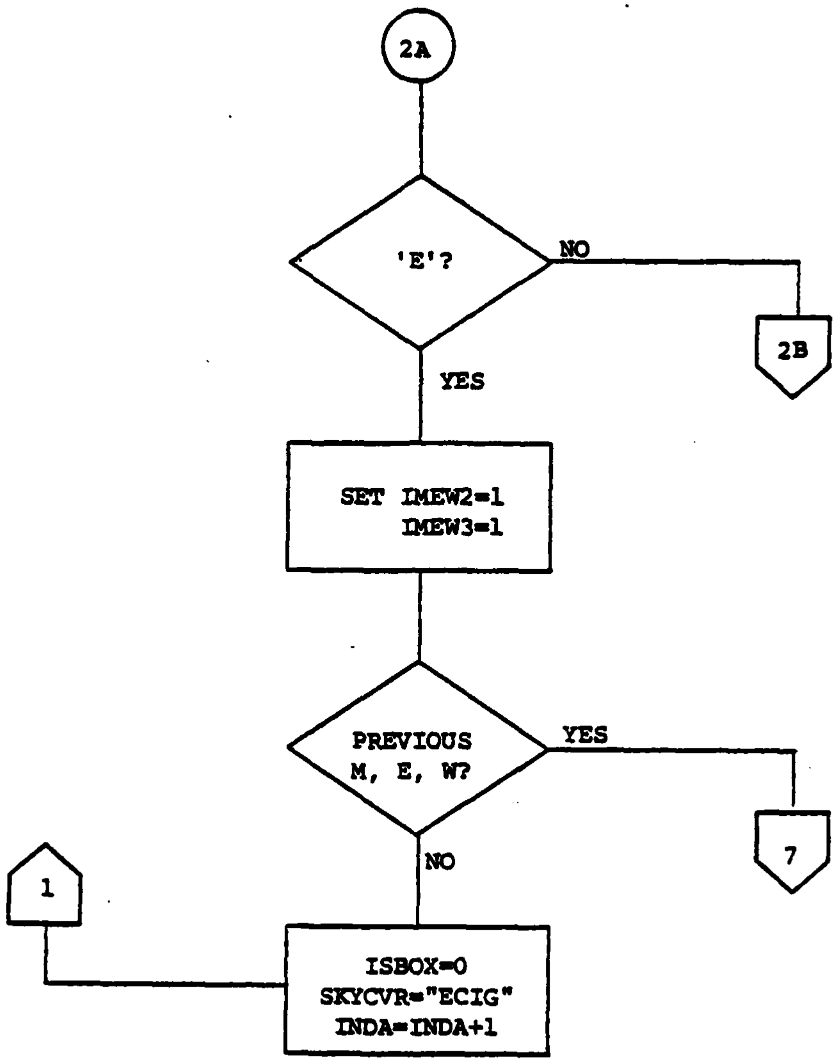


FIGURE C-3: SA PROCESSOR (Cont'd.)

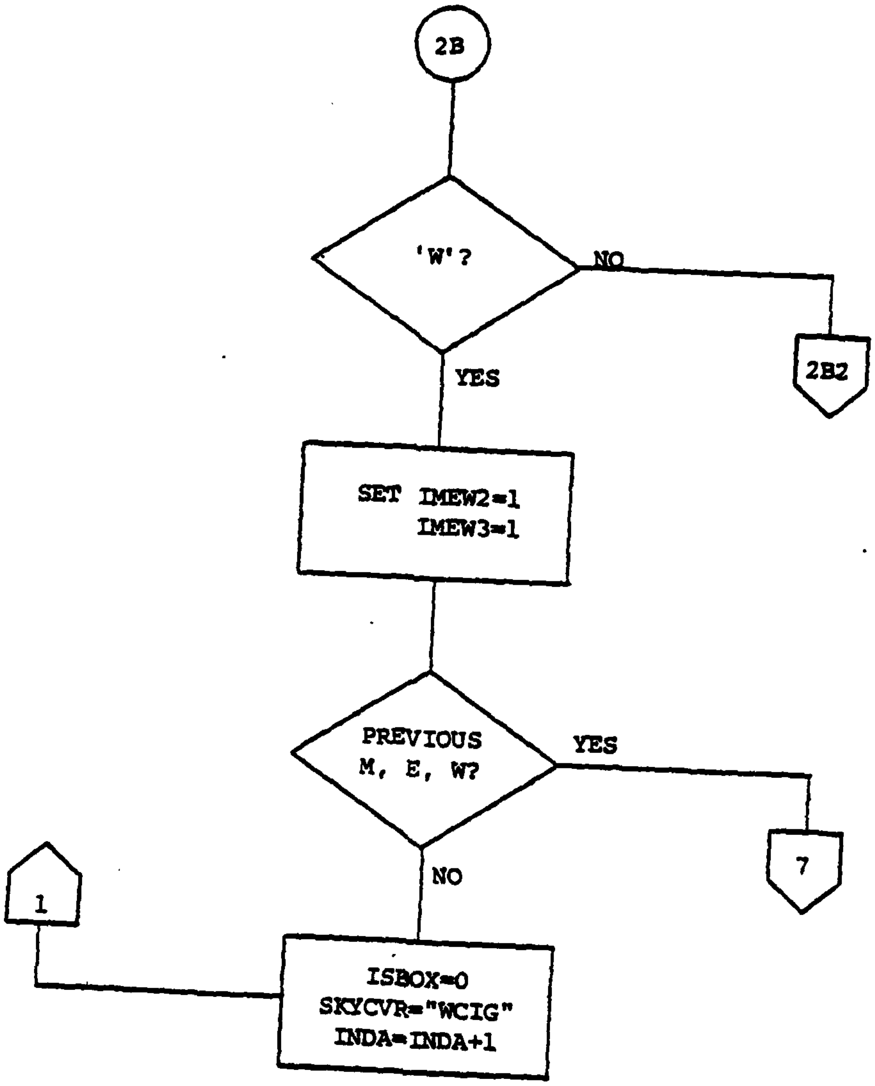


FIGURE C-3: SA PROCESSOR (Cont'd.)

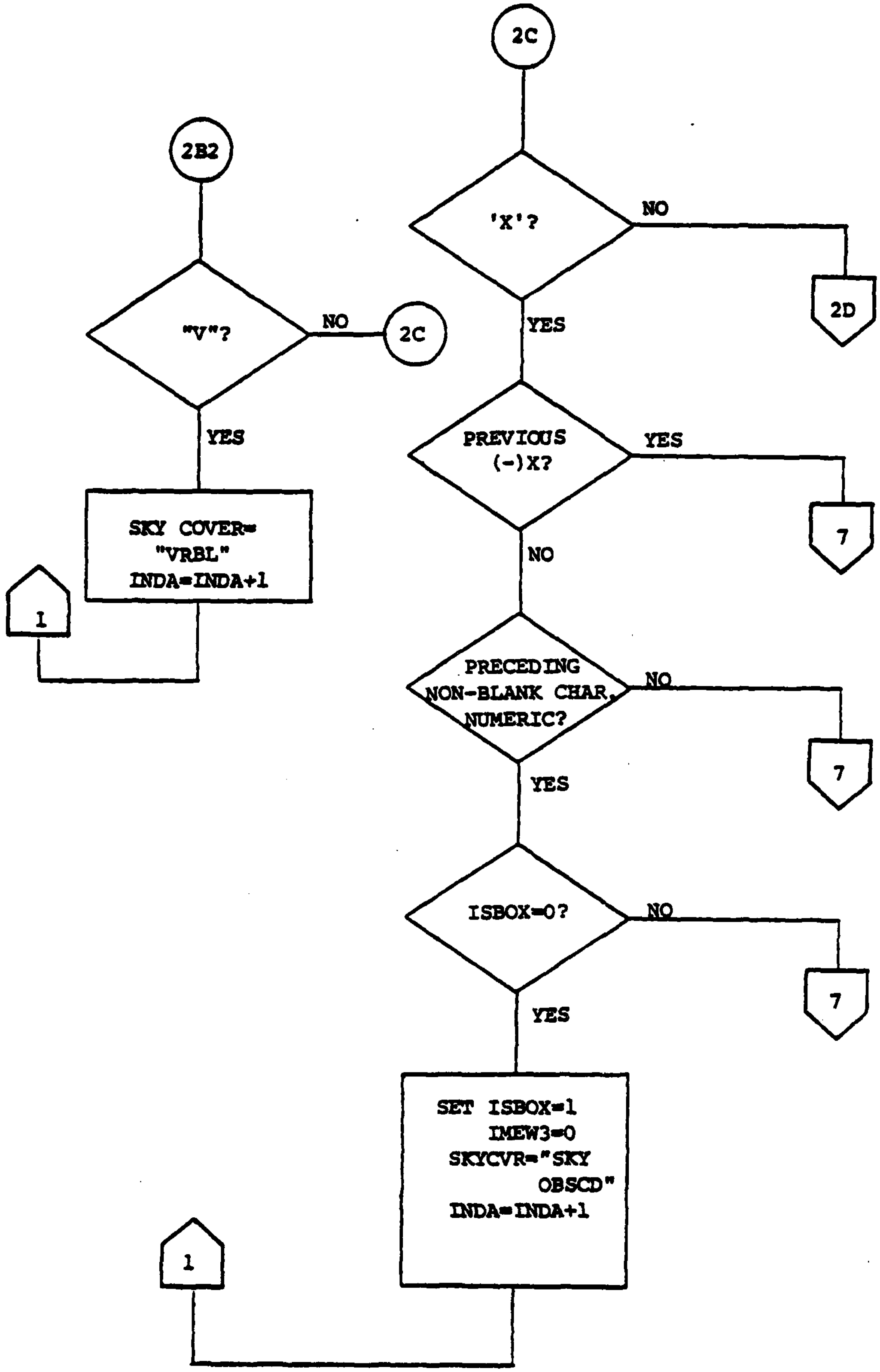


FIGURE C-3: SA PROCESSOR (Cont'd.)

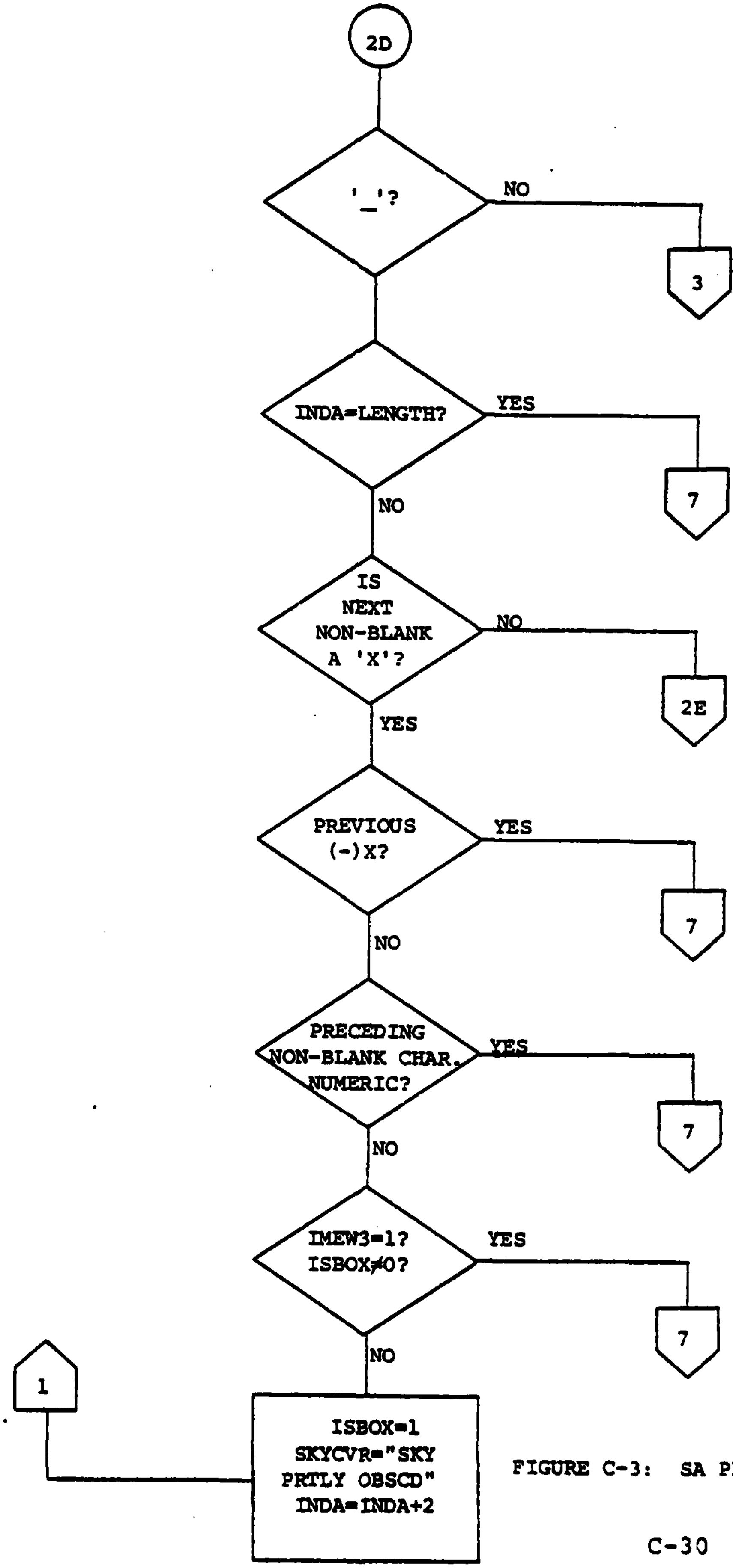


FIGURE C-3: SA PROCESSOR (Cont'd.)

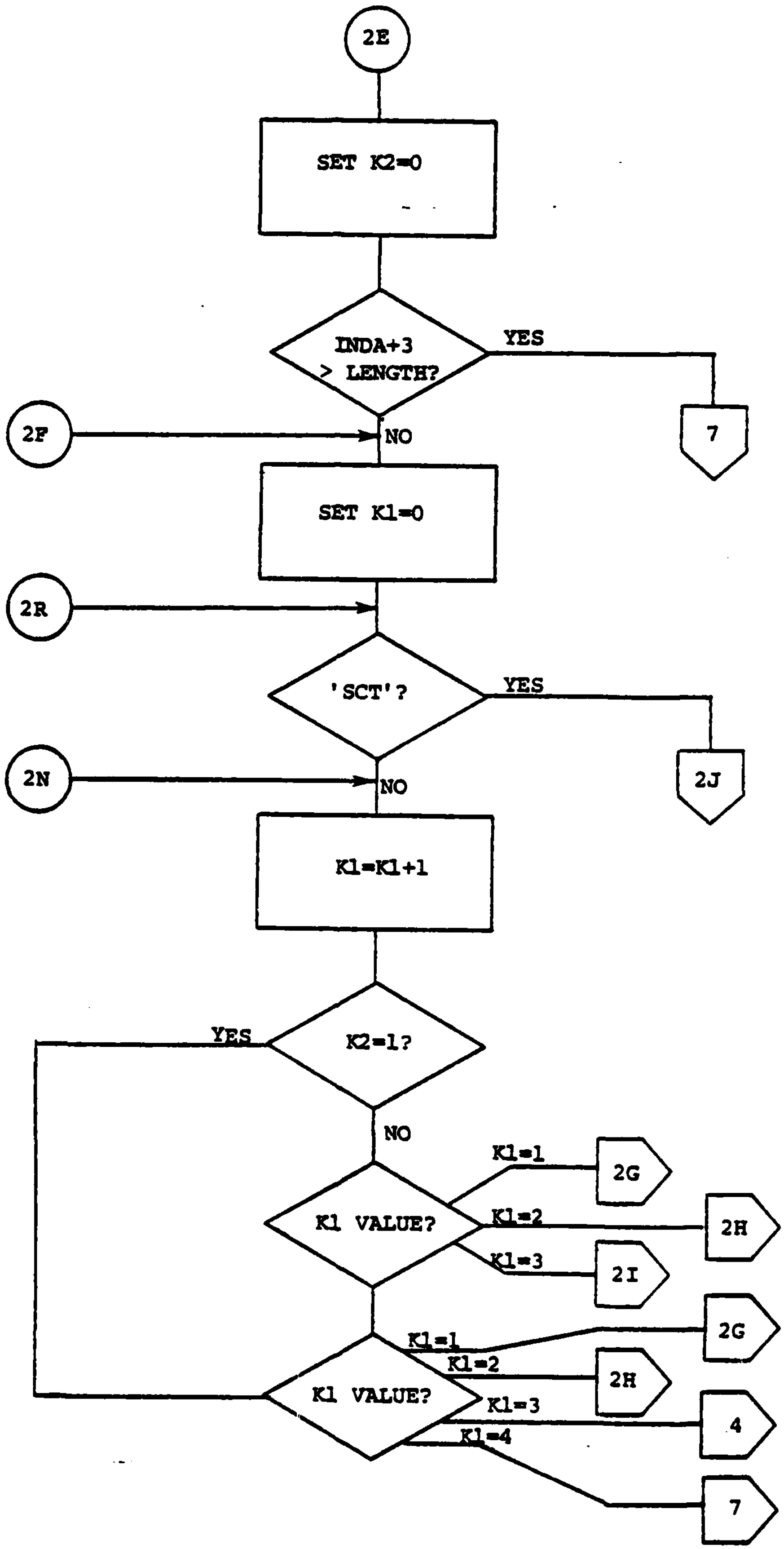


FIGURE C-3: SA PROCESSOR (Cont'd.)

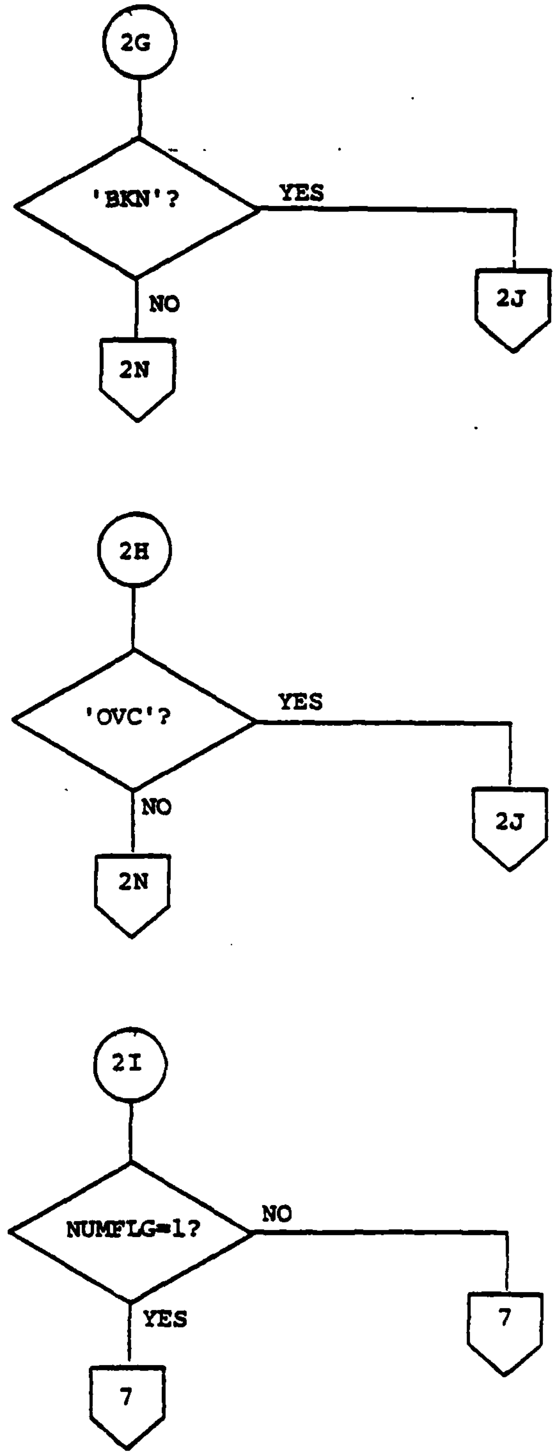


FIGURE C-3: SA PROCESSOR (Cont'd.)

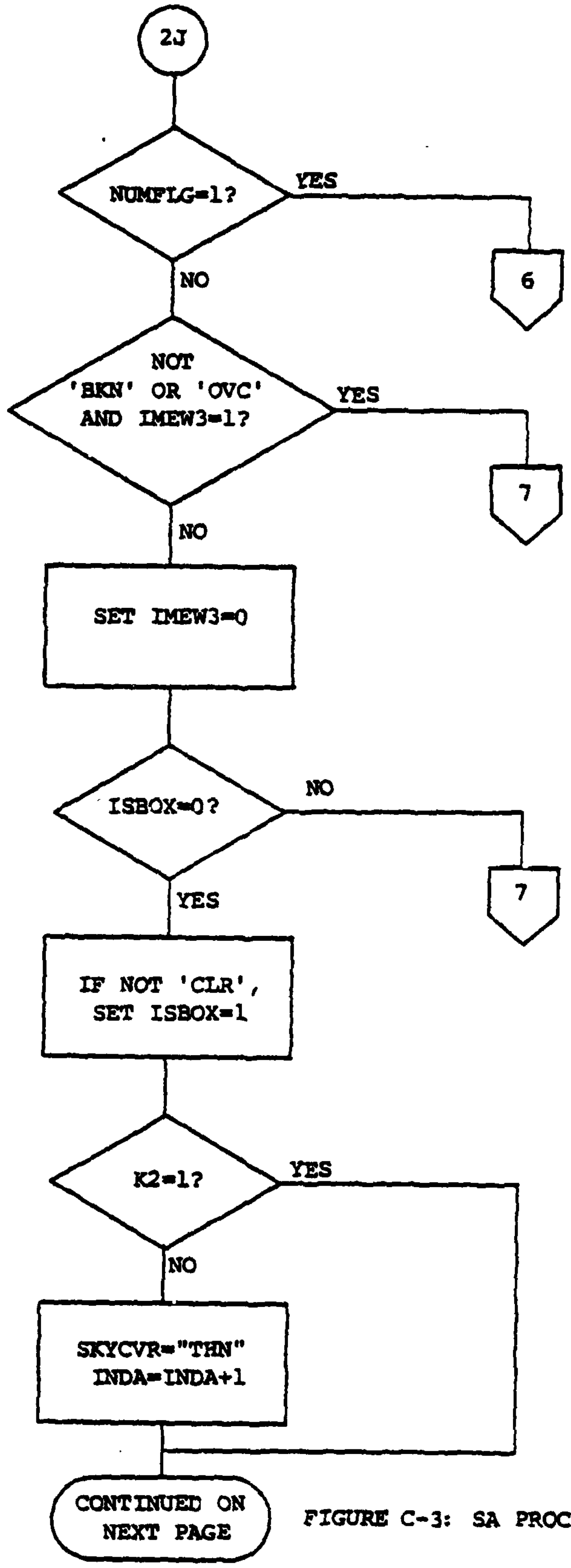


FIGURE C-3: SA PROCESSOR (Cont'd.)

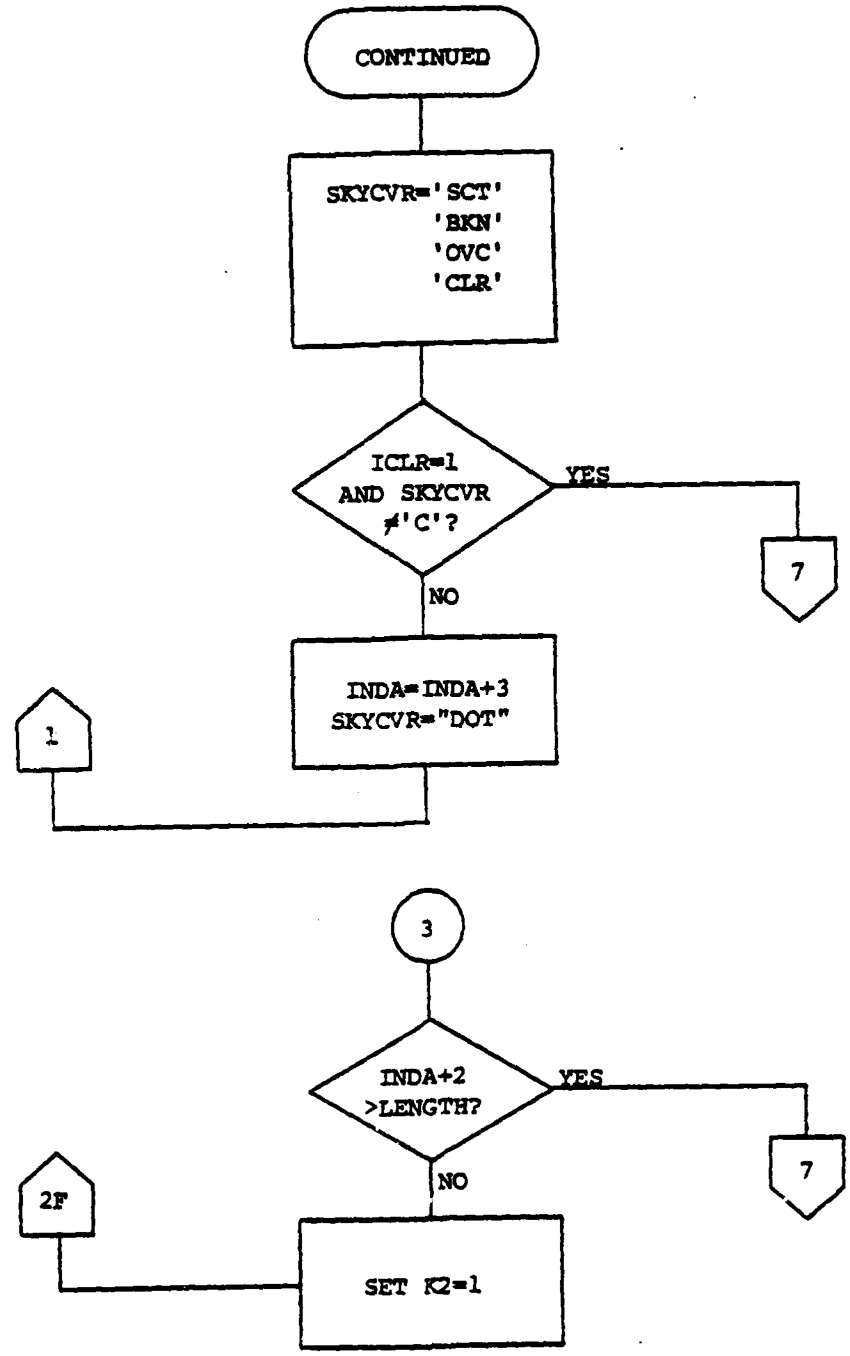


FIGURE C-3: SA PROCESSOR (Cont'd.)

AD-A102 185   INPUT OUTPUT COMPUTER SERVICES INC WALTHAM MA   F/G 17/2
TWENTY-CHANNEL VOICE RESPONSE SYSTEM.(U)
JUN 81   DOT-TSC-1313
UNCLASSIFIED   FAA-RD-81-51   NL

5 of 5
ADA
102185

END
DATE
FILMED
8-81
DTIC

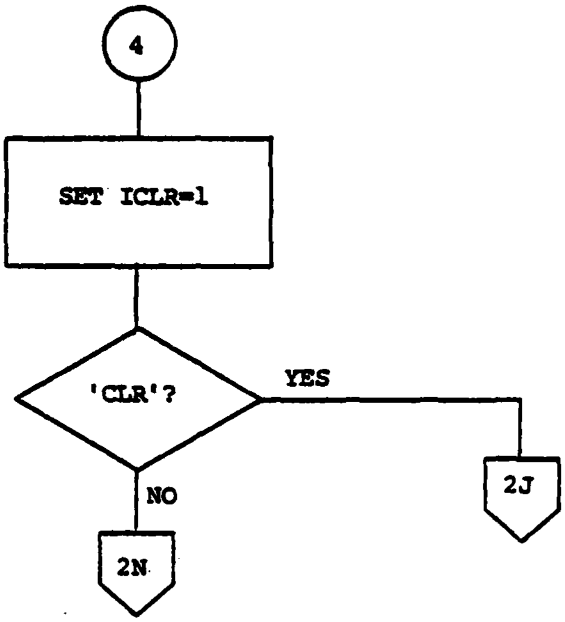


FIGURE C-3: SA PROCESSOR (Cont'd.)

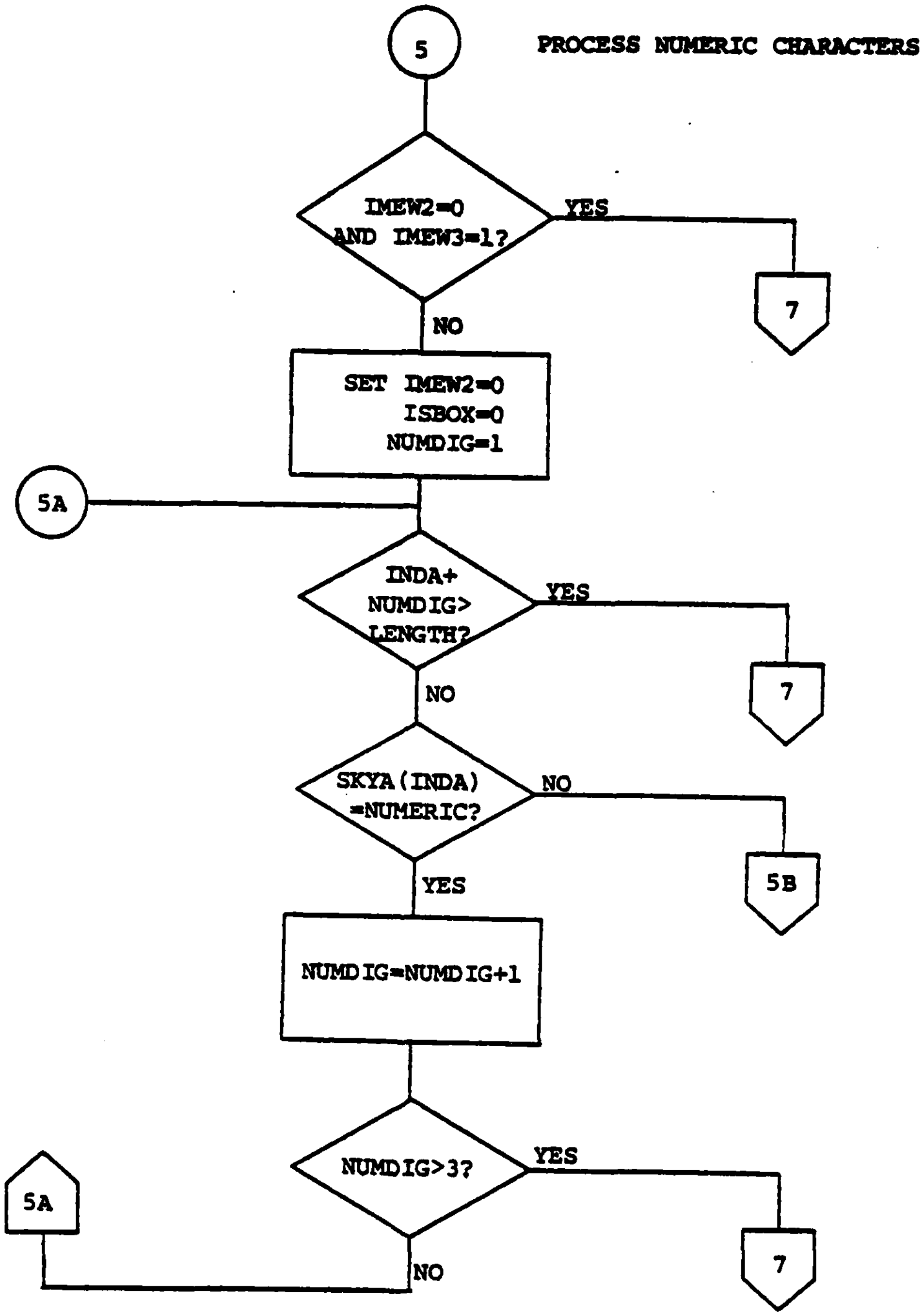


FIGURE C-3: SA PROCESSOR (Cont'd.)

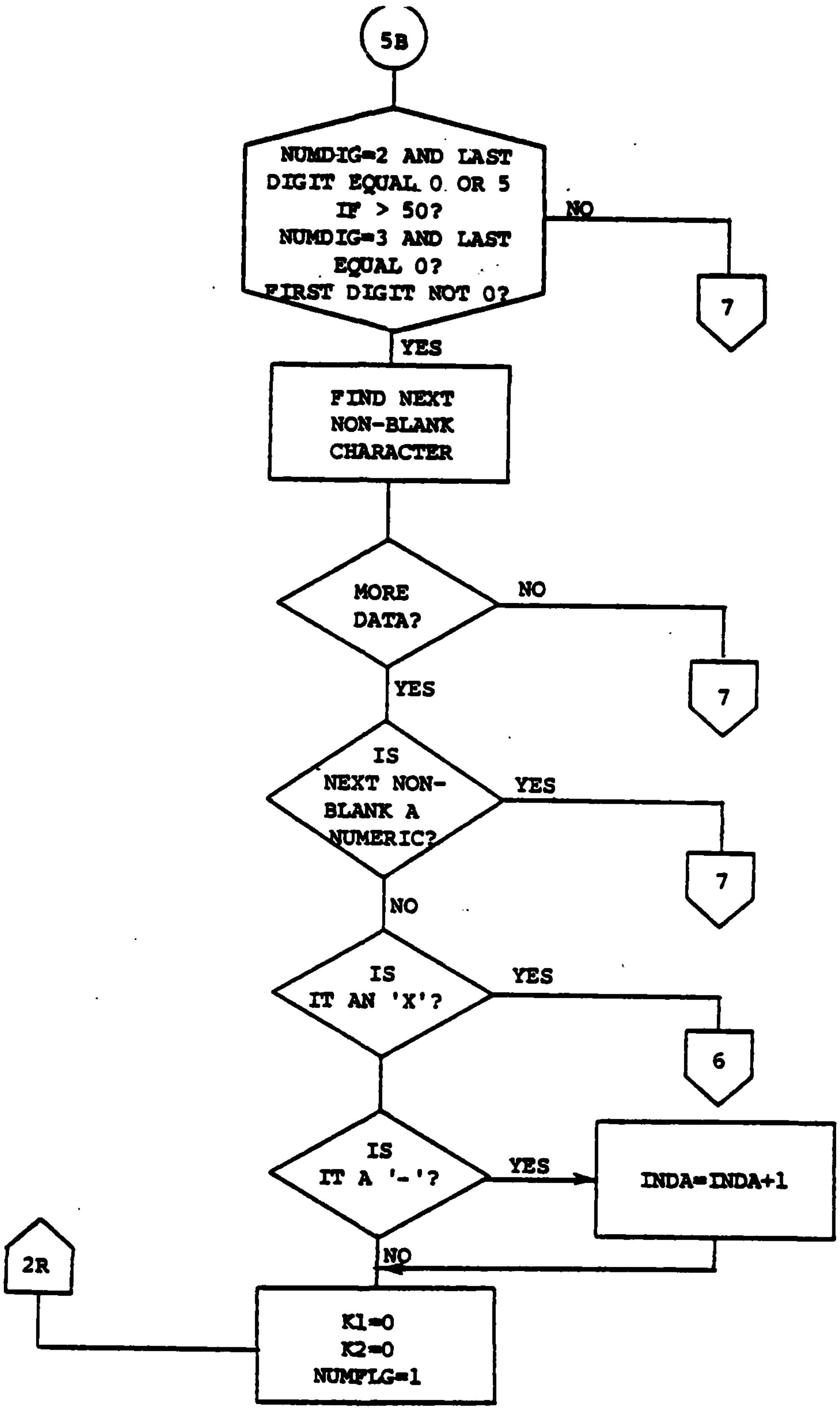


FIGURE C-3: SA PROCESSOR (Cont'd.)

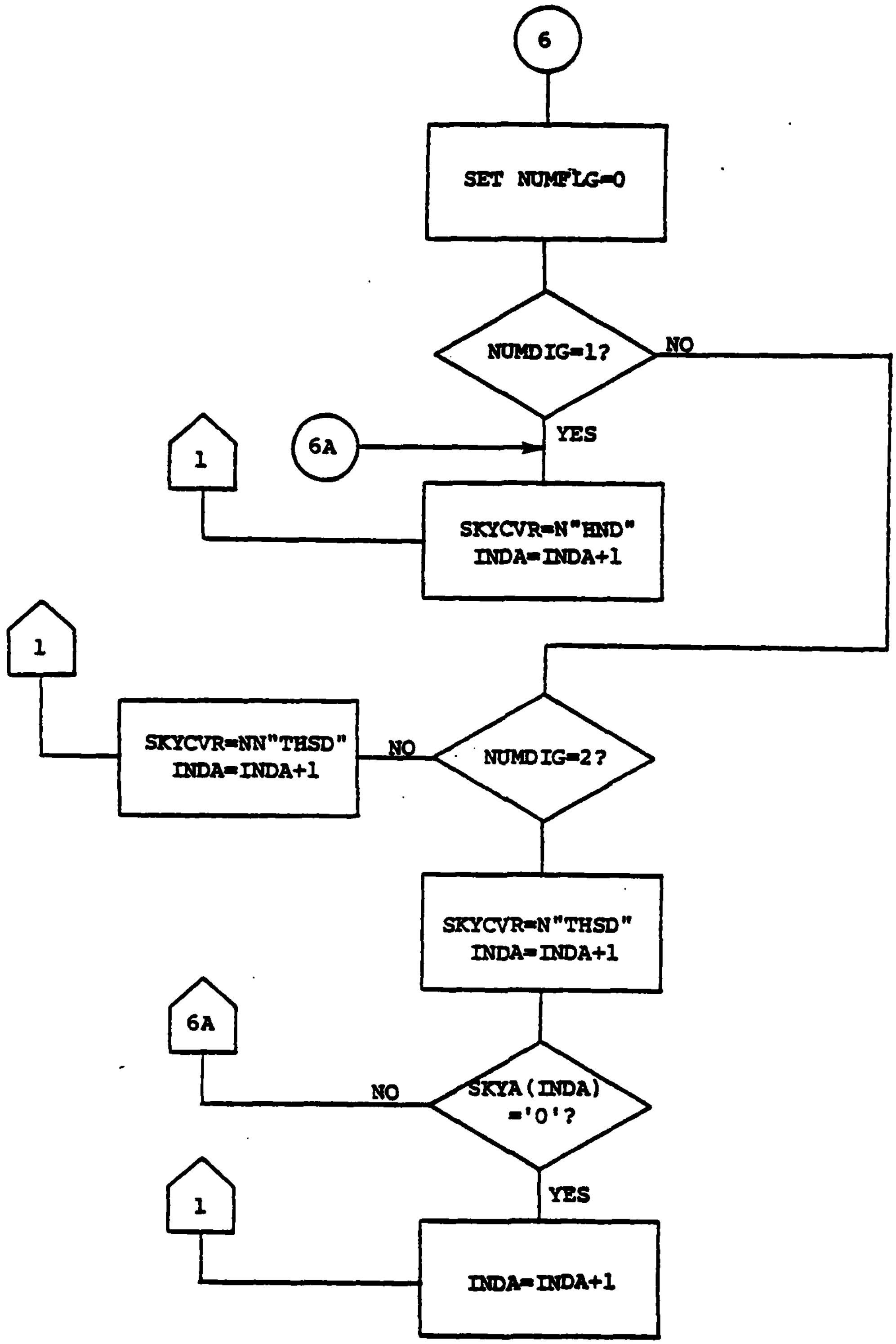


FIGURE C-3: SA PROCESSOR (Cont'd.)

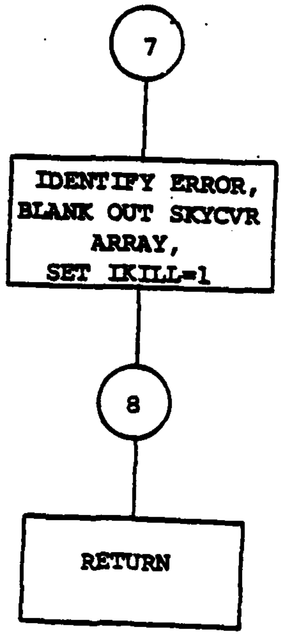


FIGURE C-3: SA PROCESSOR (Cont'd.)

C.4 SA REMARKS PROCESSOR

VRBMK REMARKS PROCESSOR MAIN

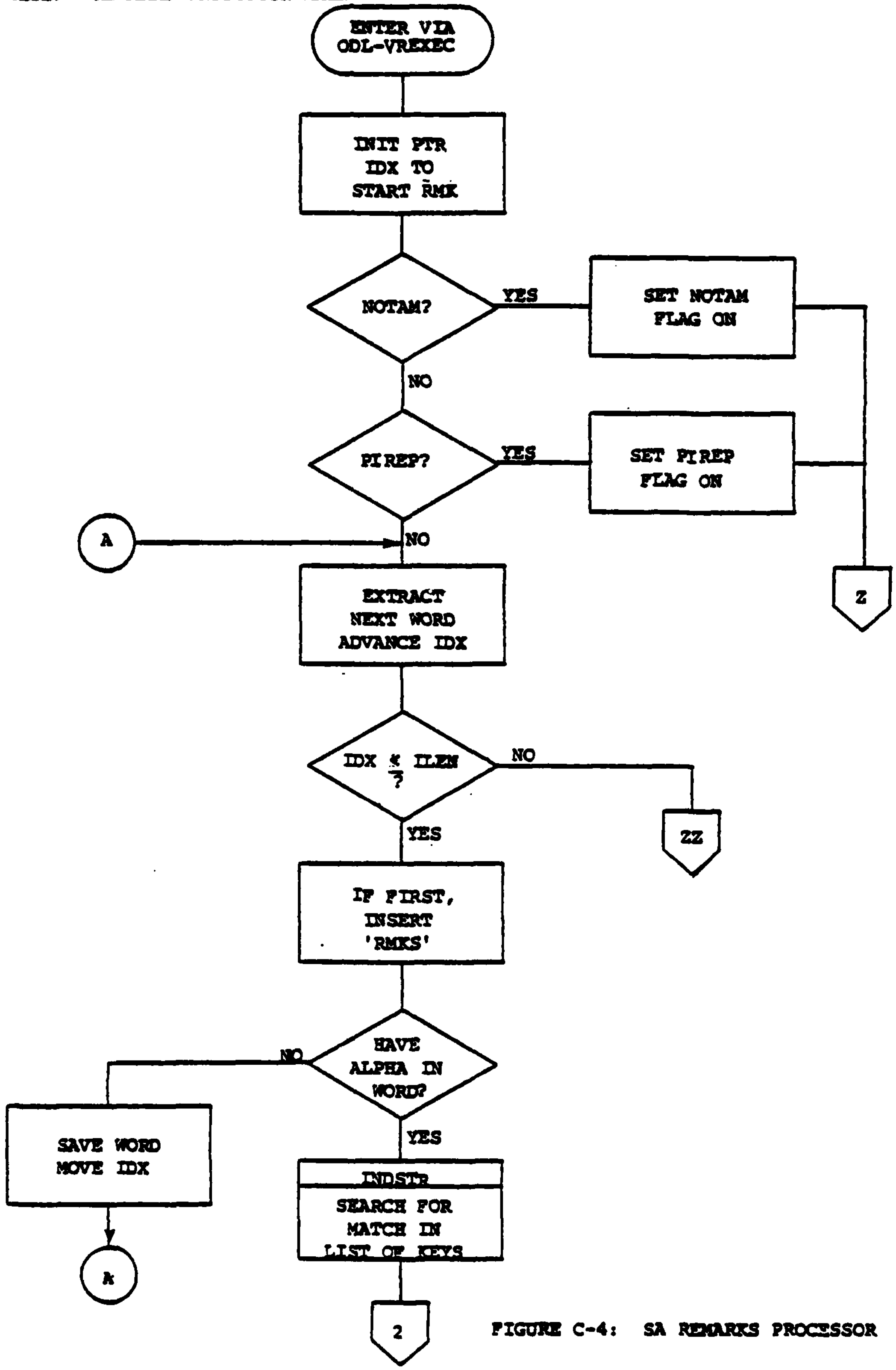


FIGURE C-4: SA REMARKS PROCESSOR

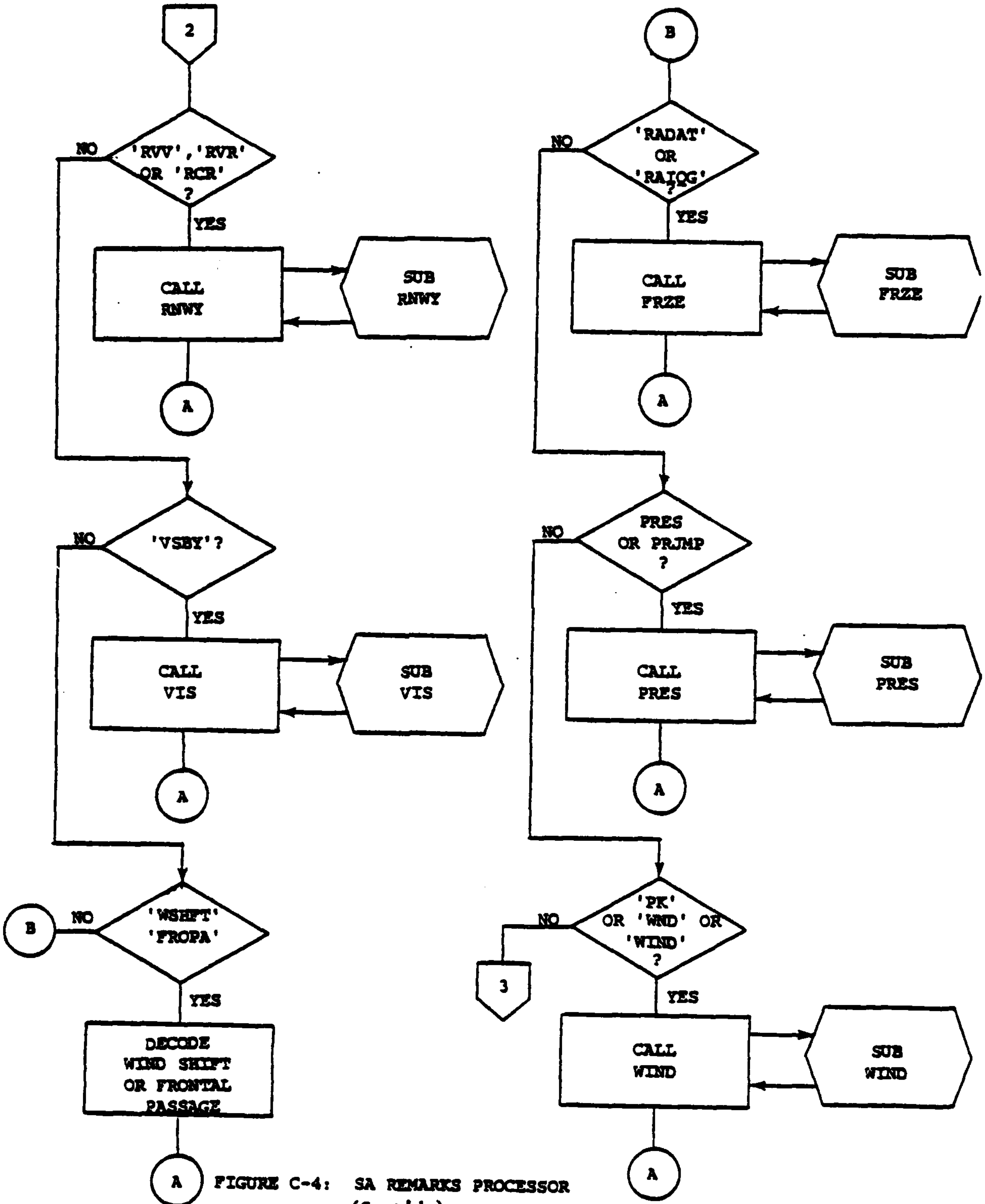


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

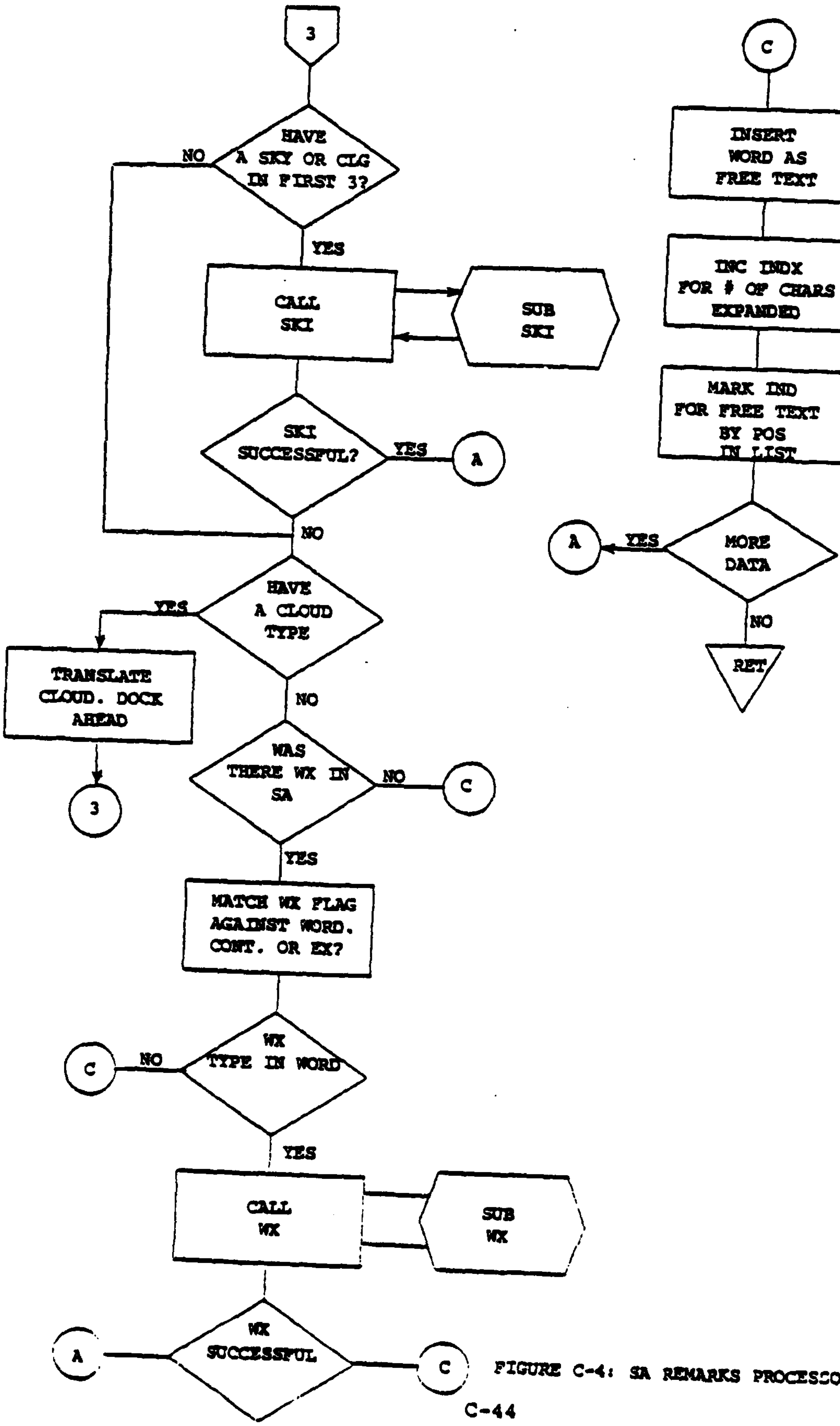


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

SUBROUTINE WIND (A, IND, RLIST, IRLEN, ILEN)

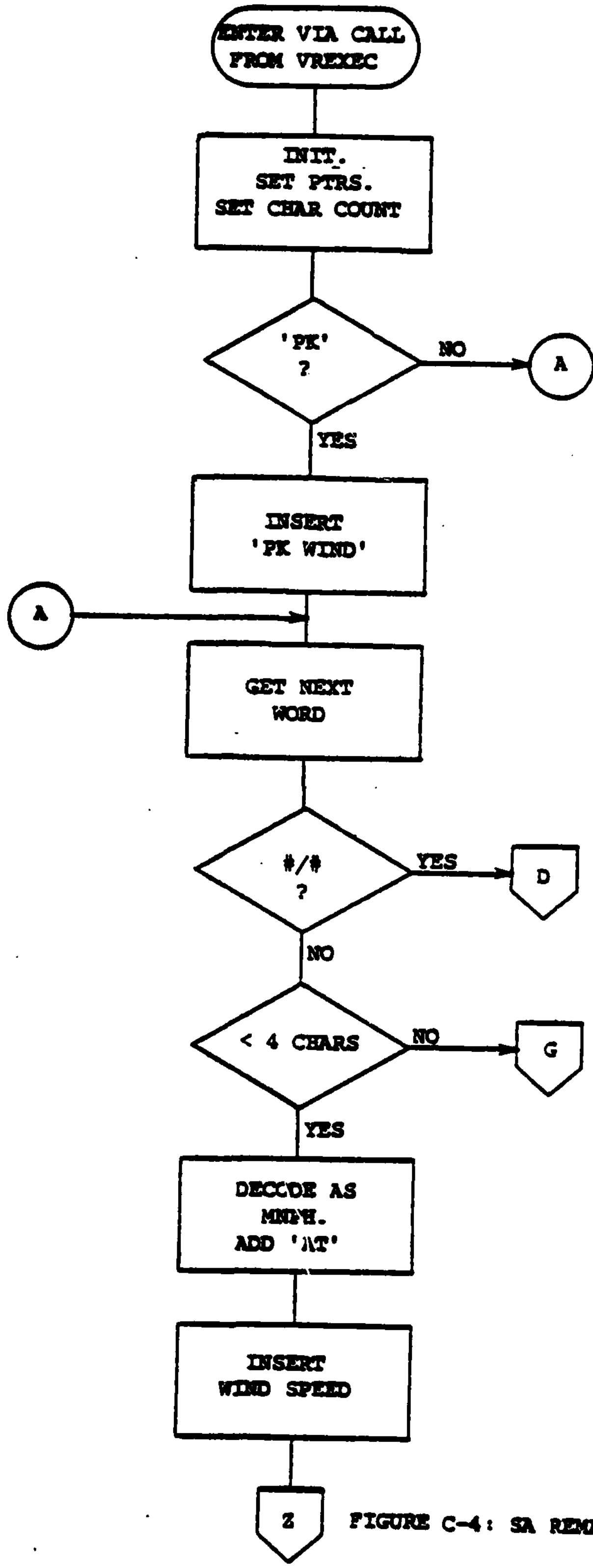


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

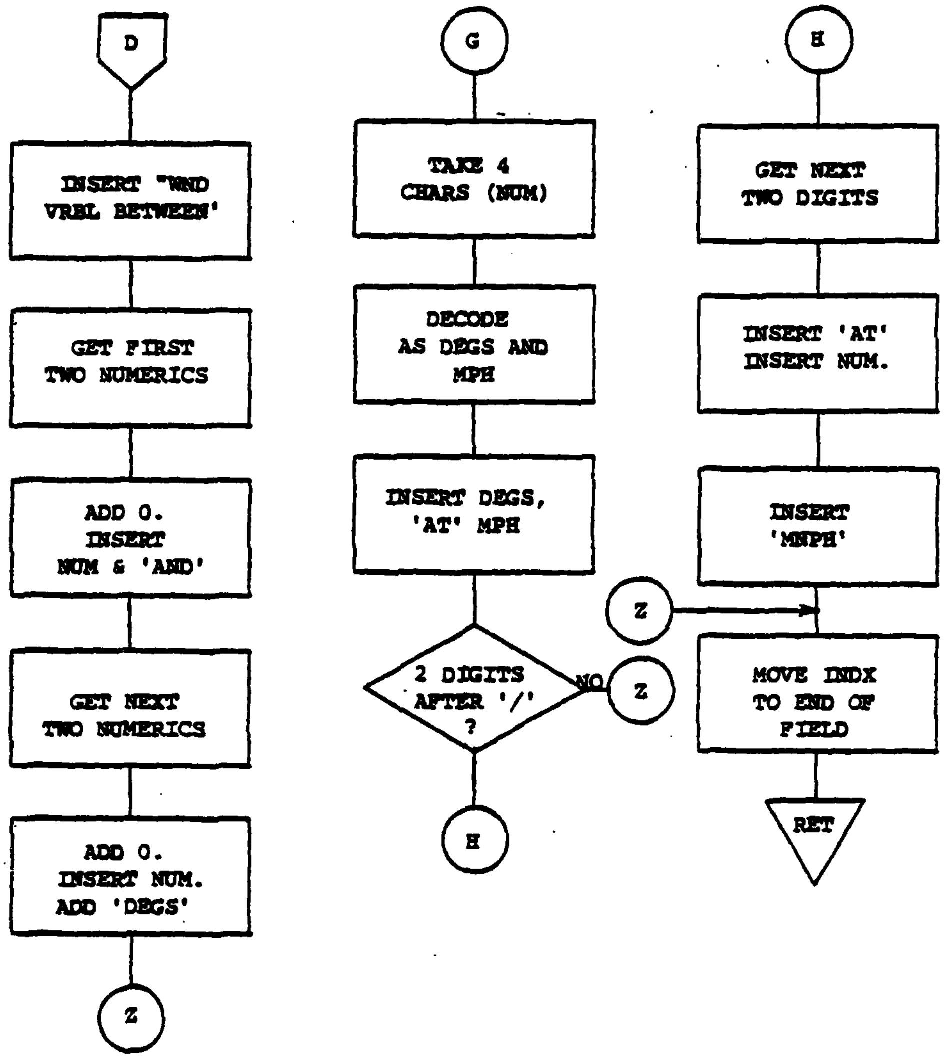


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

SUBROUTINE VIS (A, IND, RLIST, IRLEN, INDX)

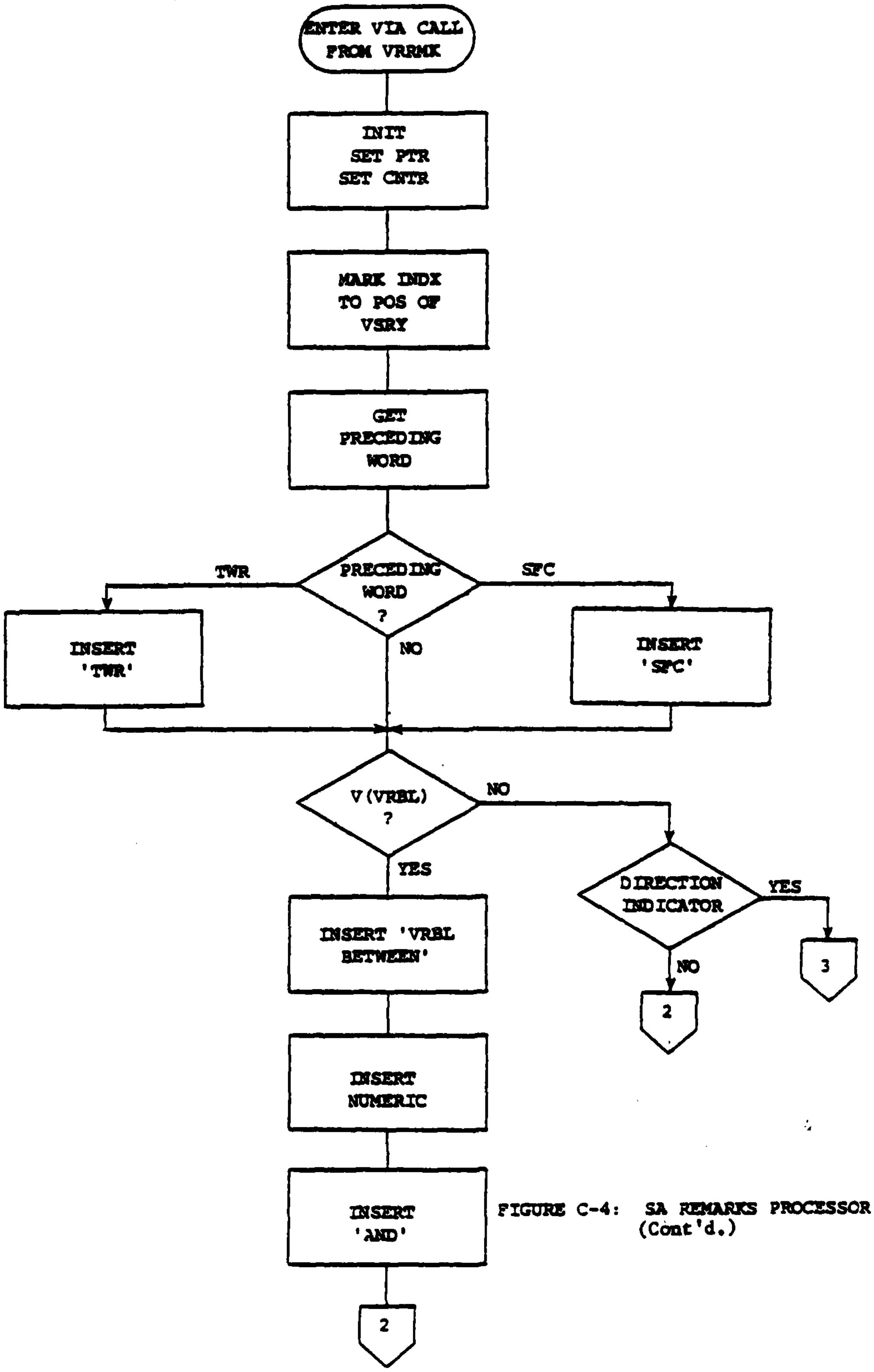


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

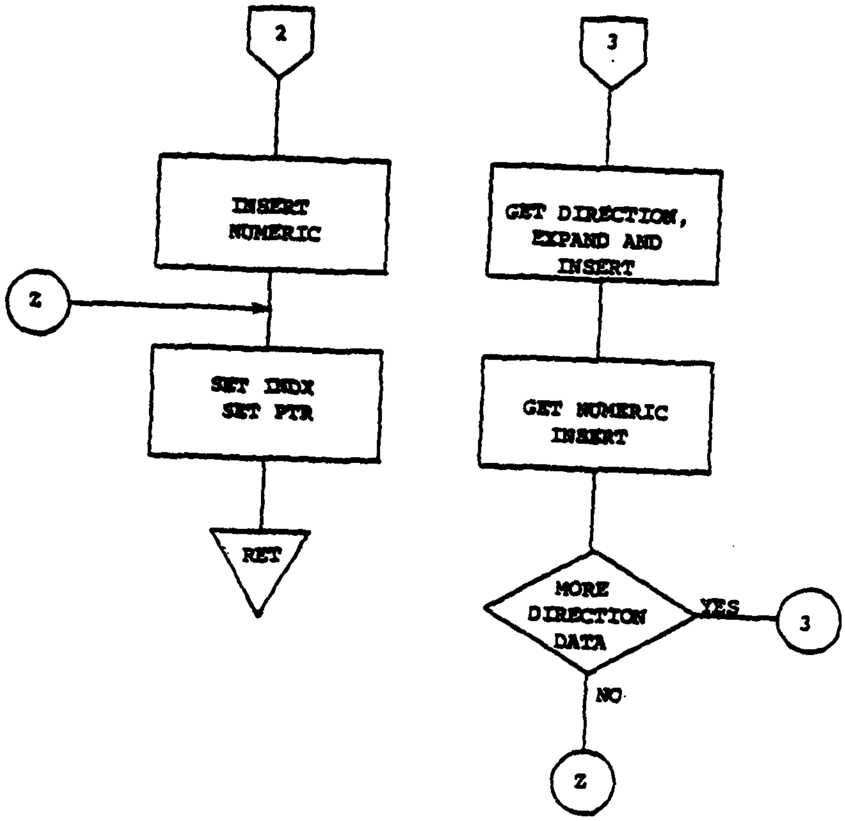


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

SUBROUTINE RNWY (A, IND, RLIST, ICALL, INDX)

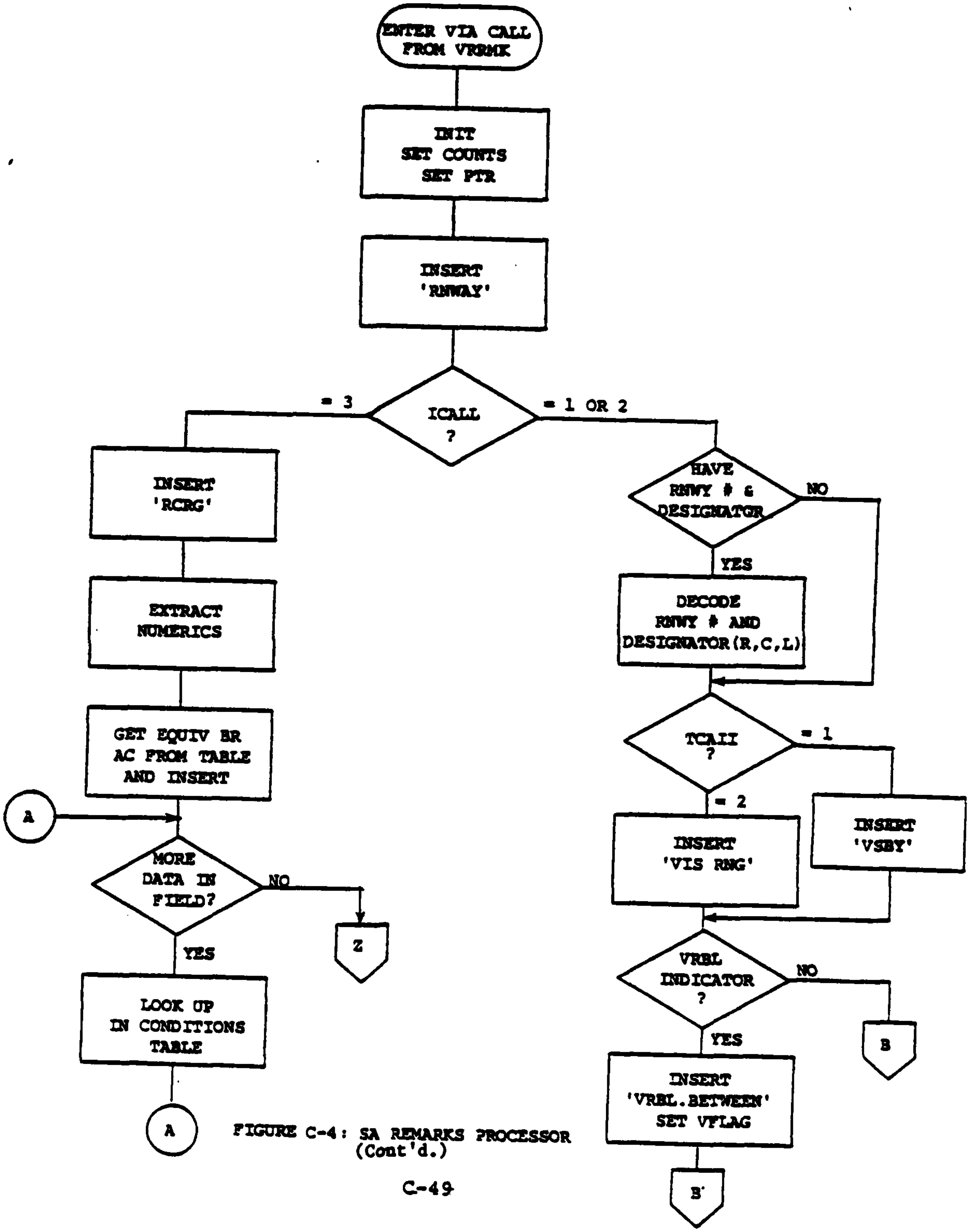


FIGURE C-4: SA REMARKS PROCESSOR
(Cont'd.)

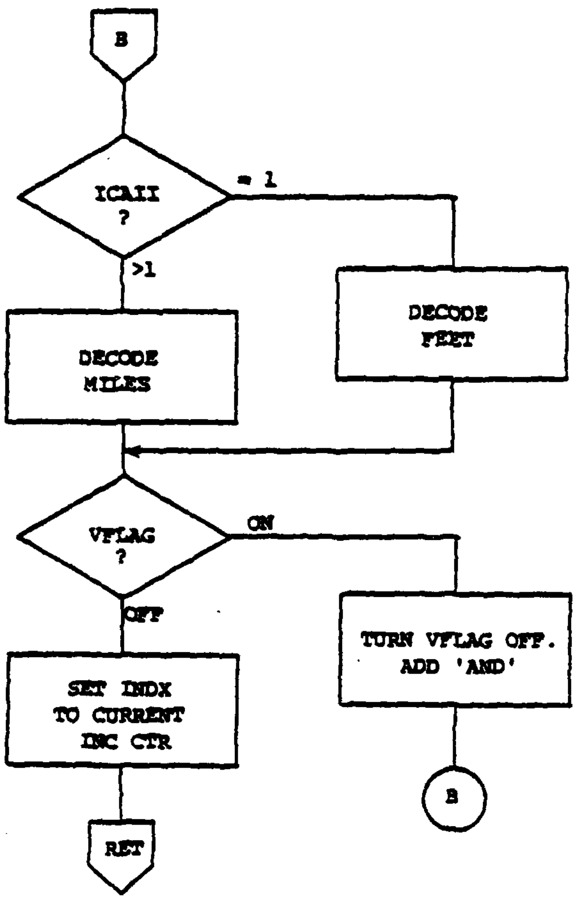


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

SUBROUTINE FRZE (A, RLIST, INDX, IRIND, ICALL)

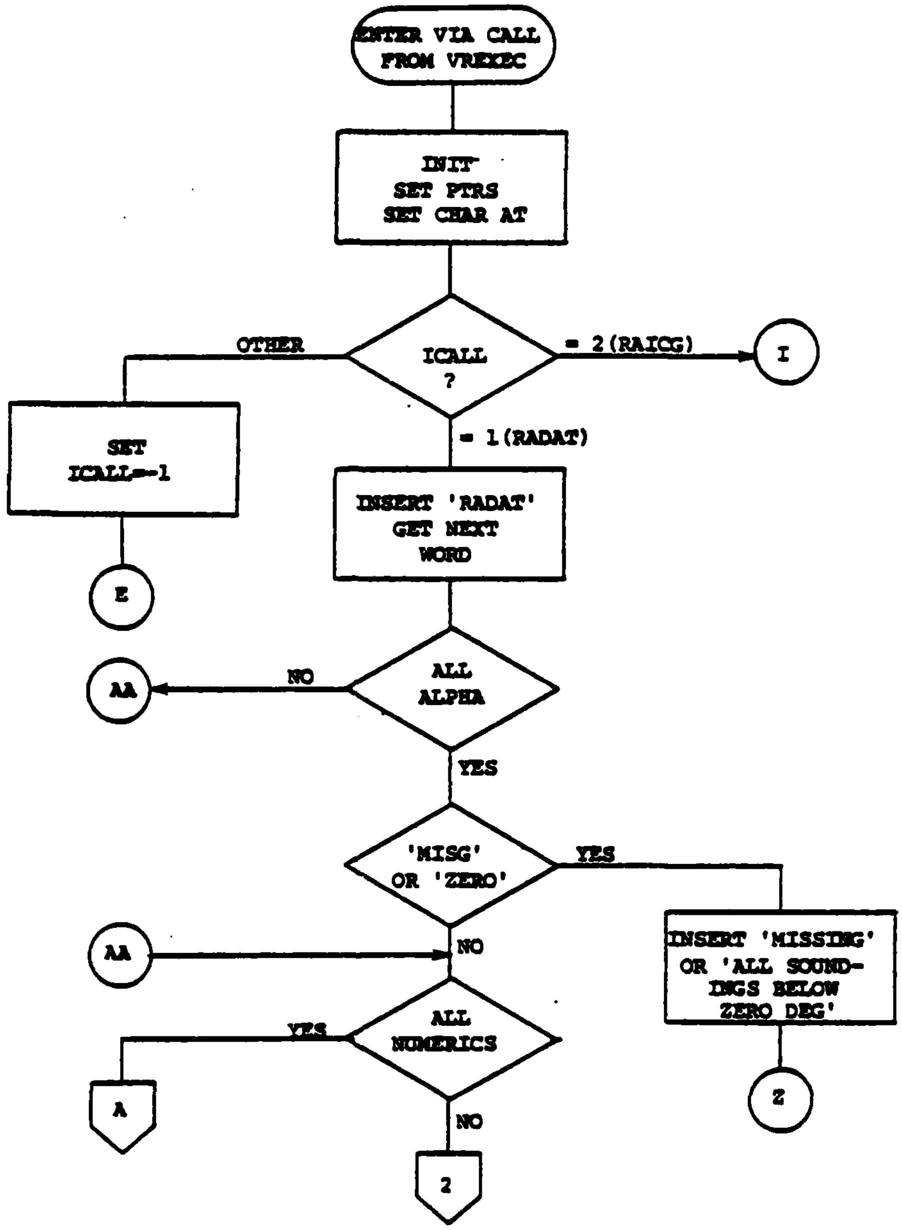


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

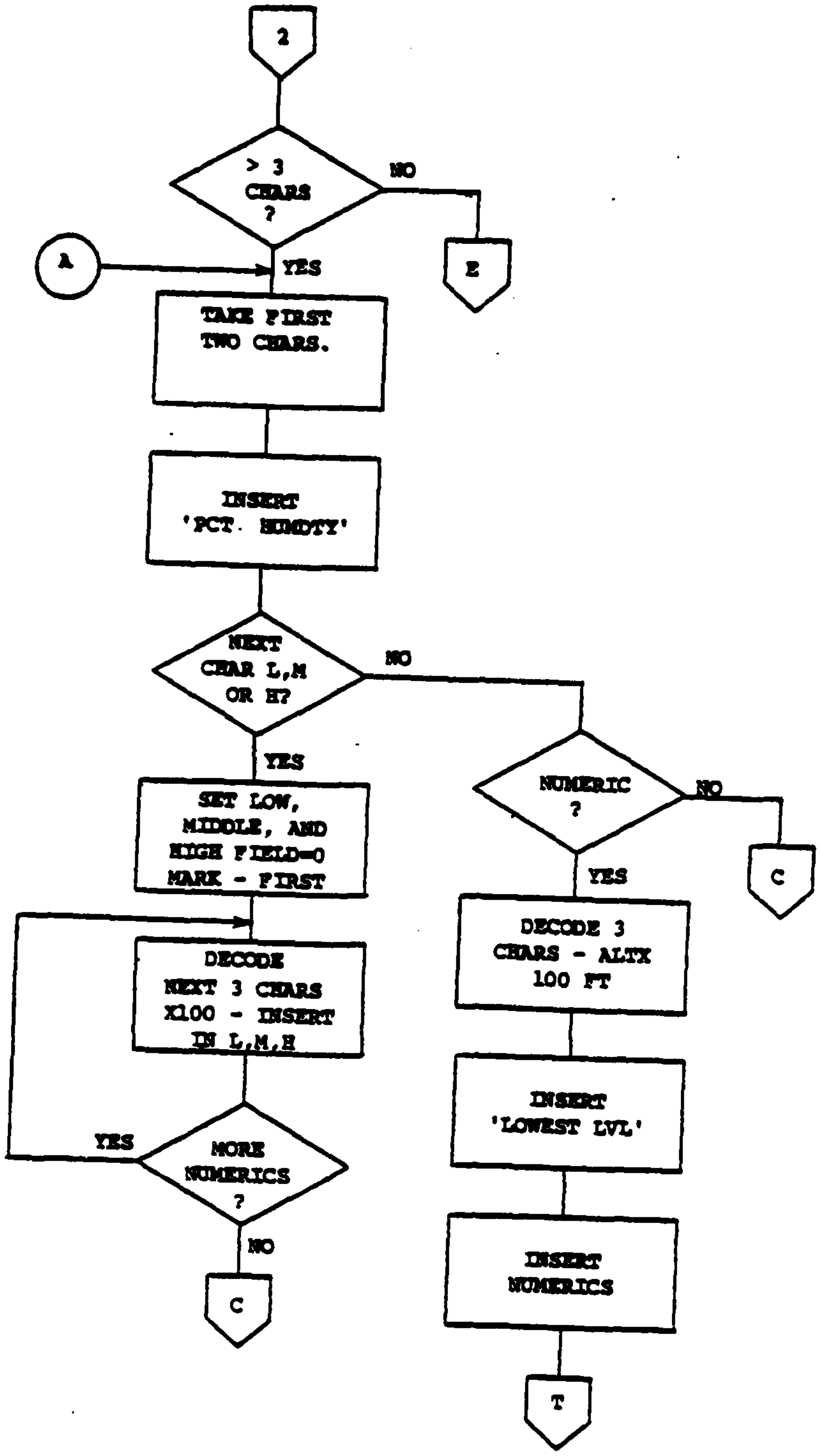


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

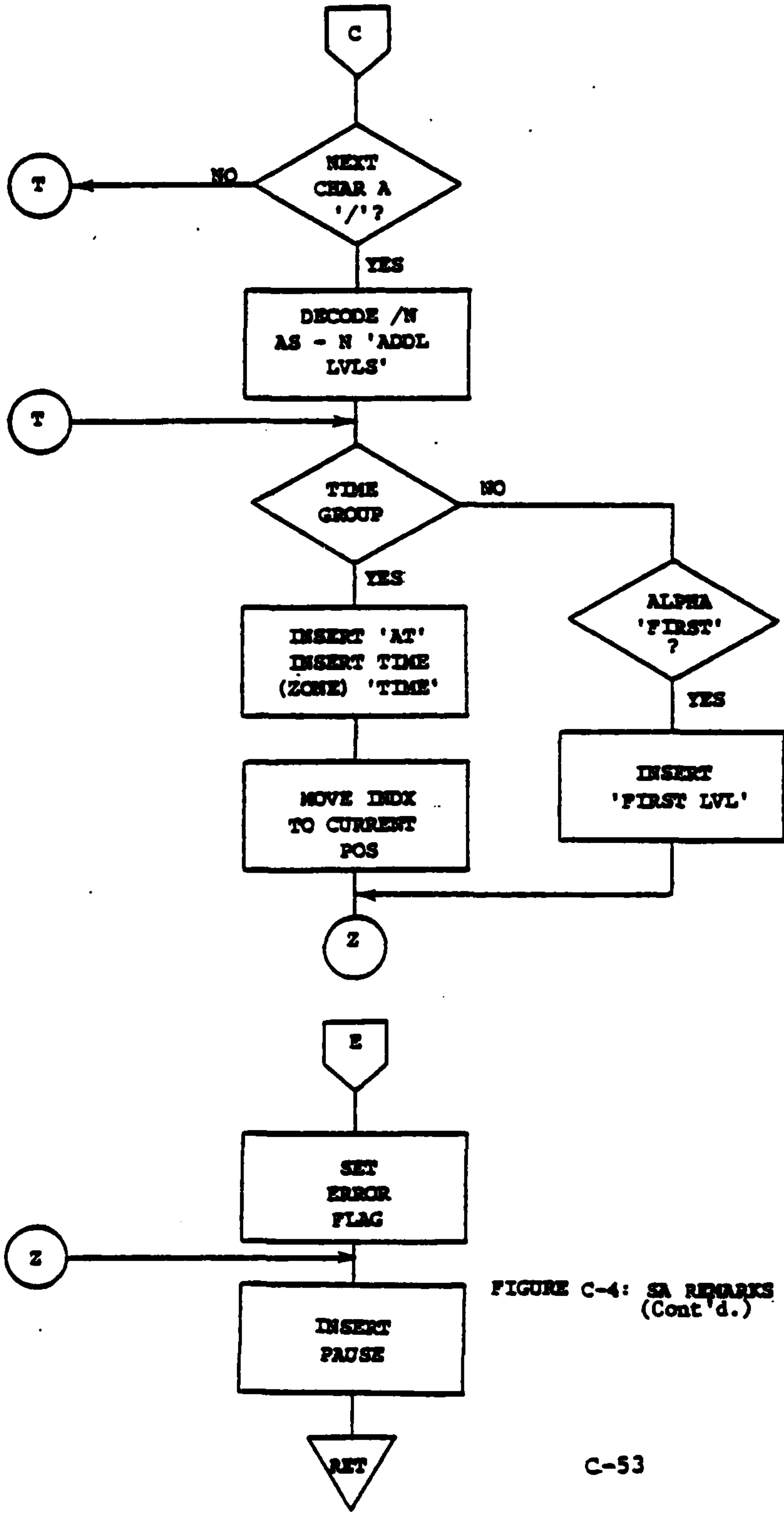


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

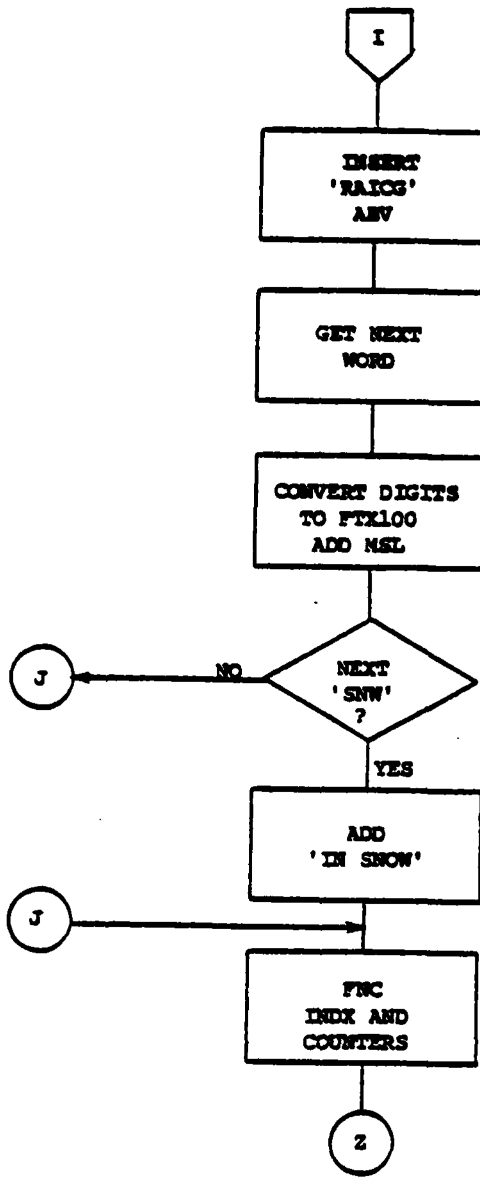


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

SUBROUTINE PRES (A, IND, RLIST, IRLEN, ILFN)

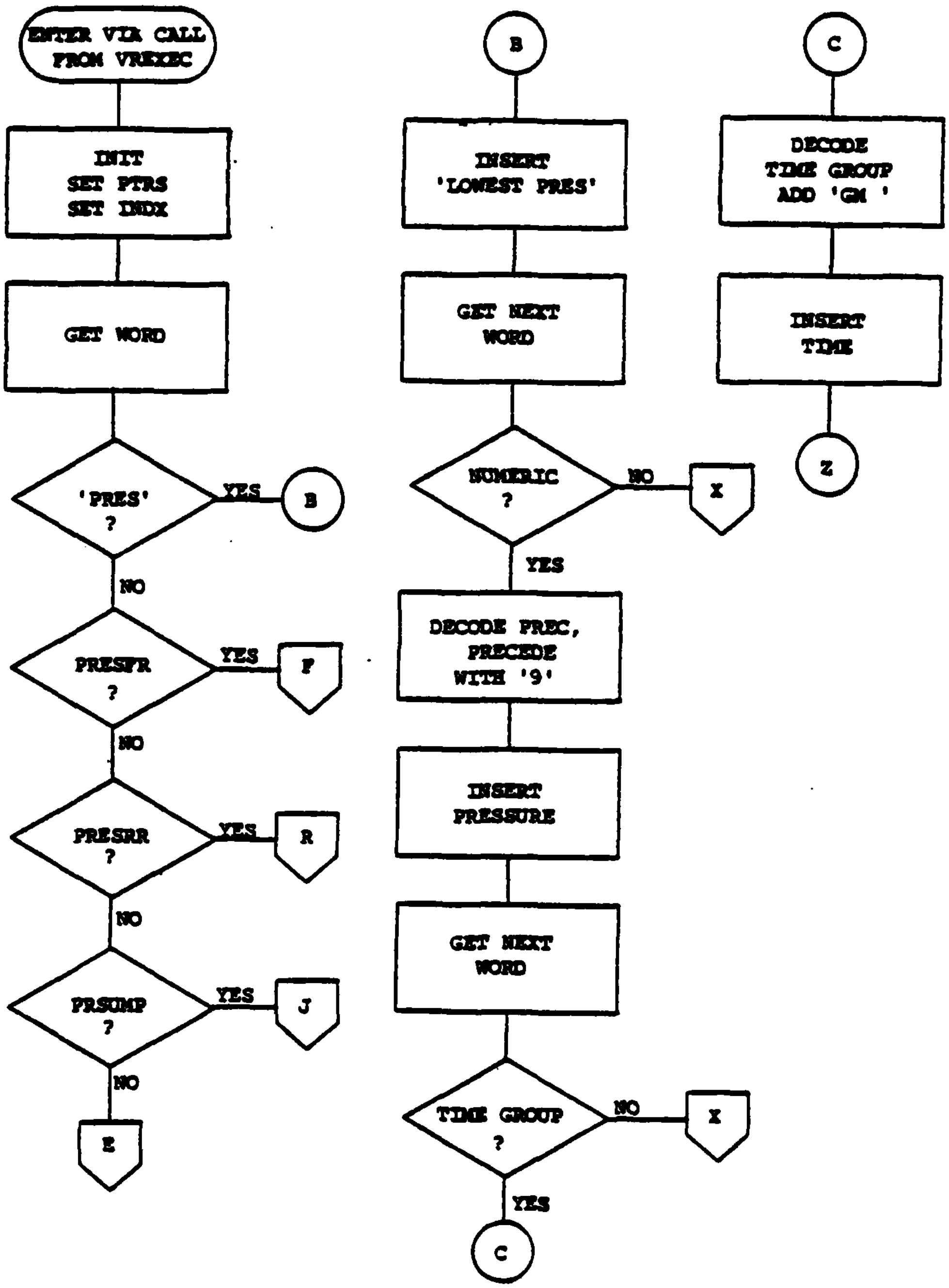


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

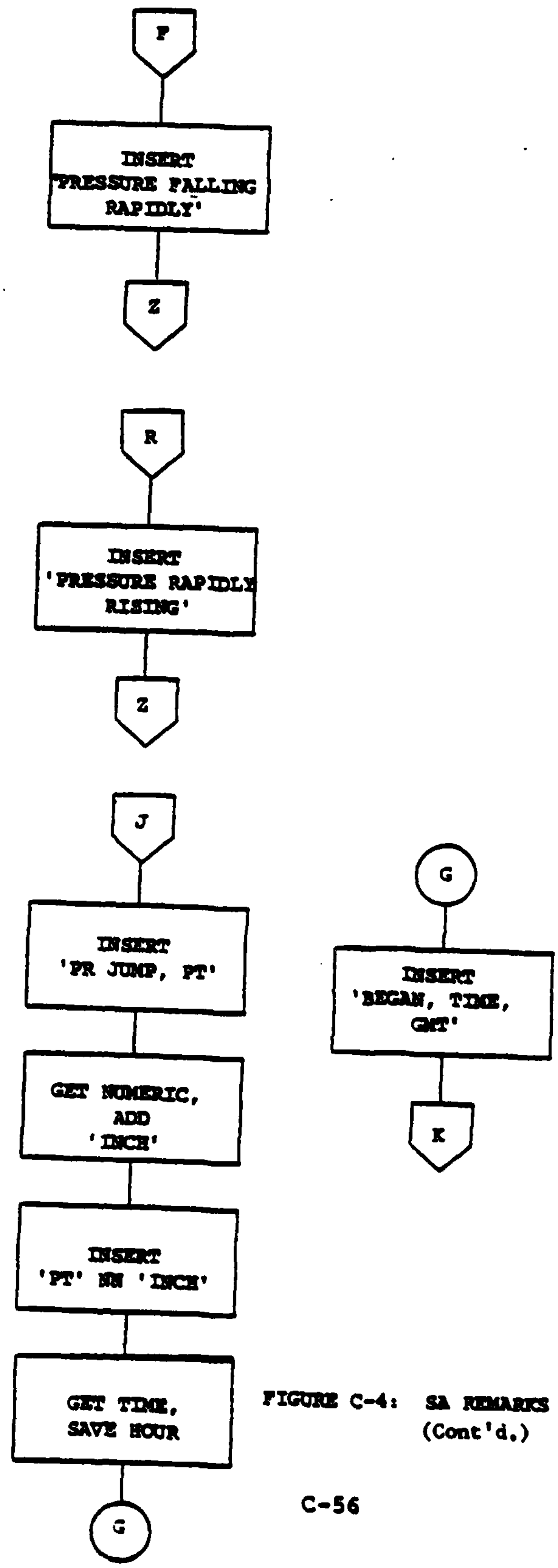


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

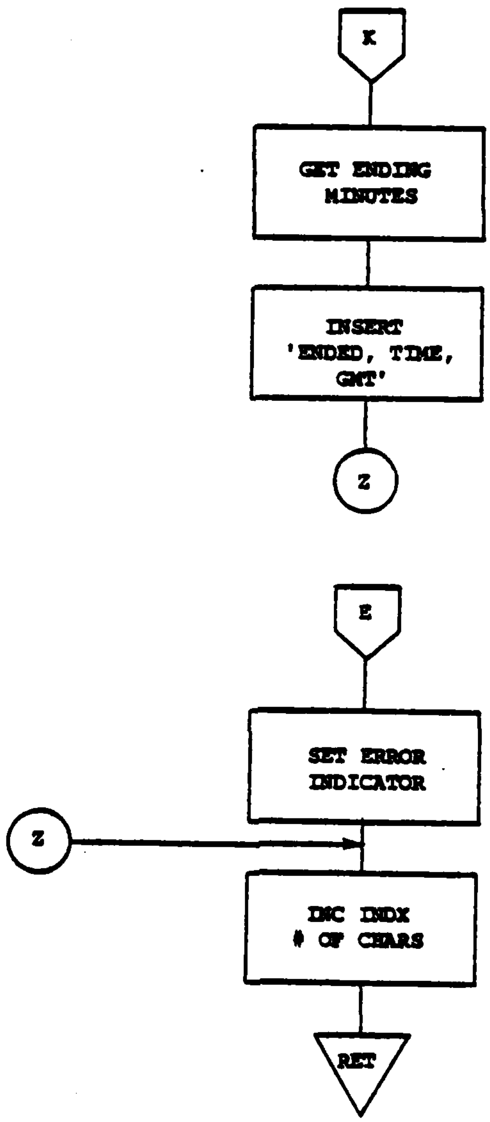


FIGURE C-4: SA REMARKS PROCESSOR (Cont'd.)

C.5 FT PROCESSOR

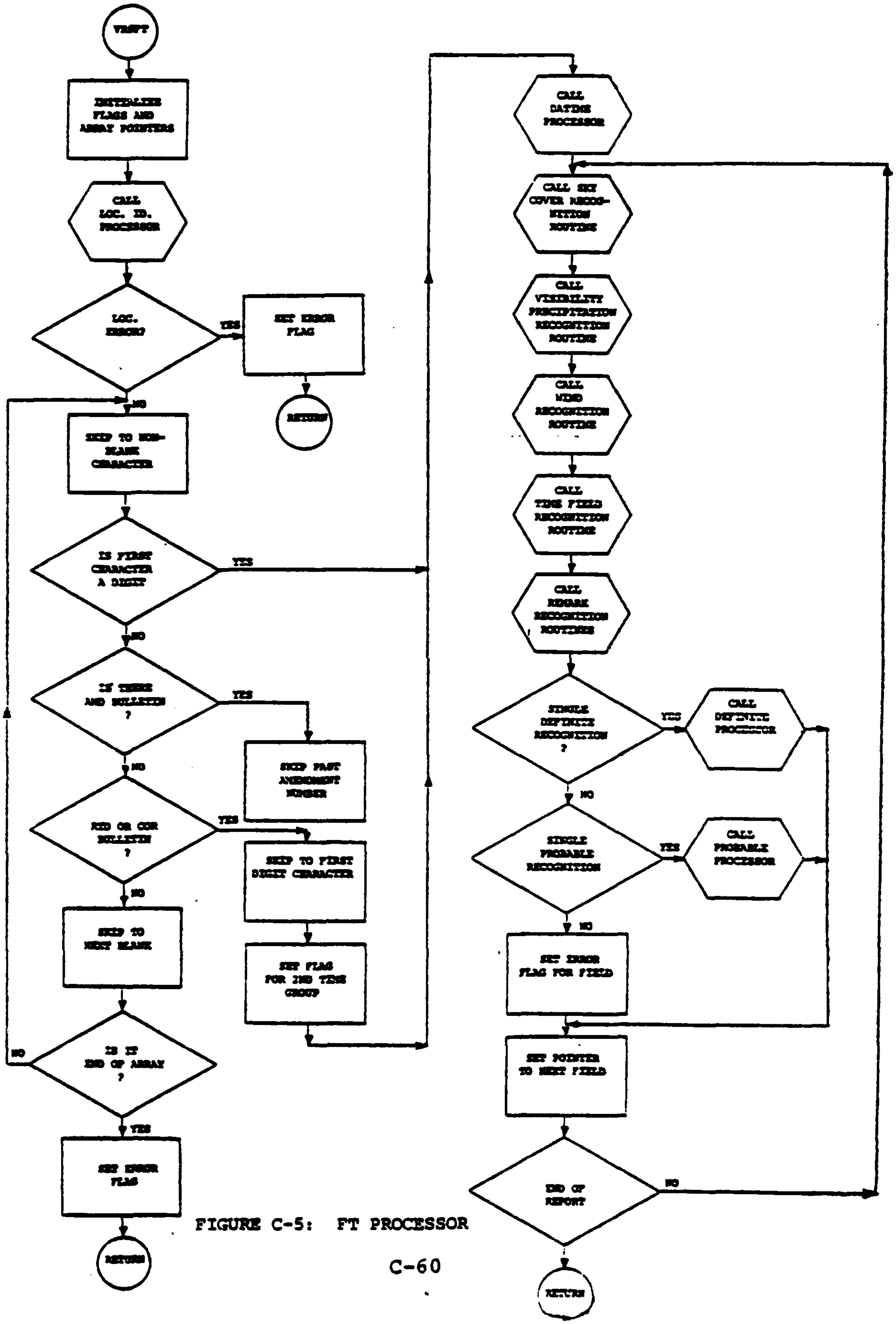


FIGURE C-5: FT PROCESSOR

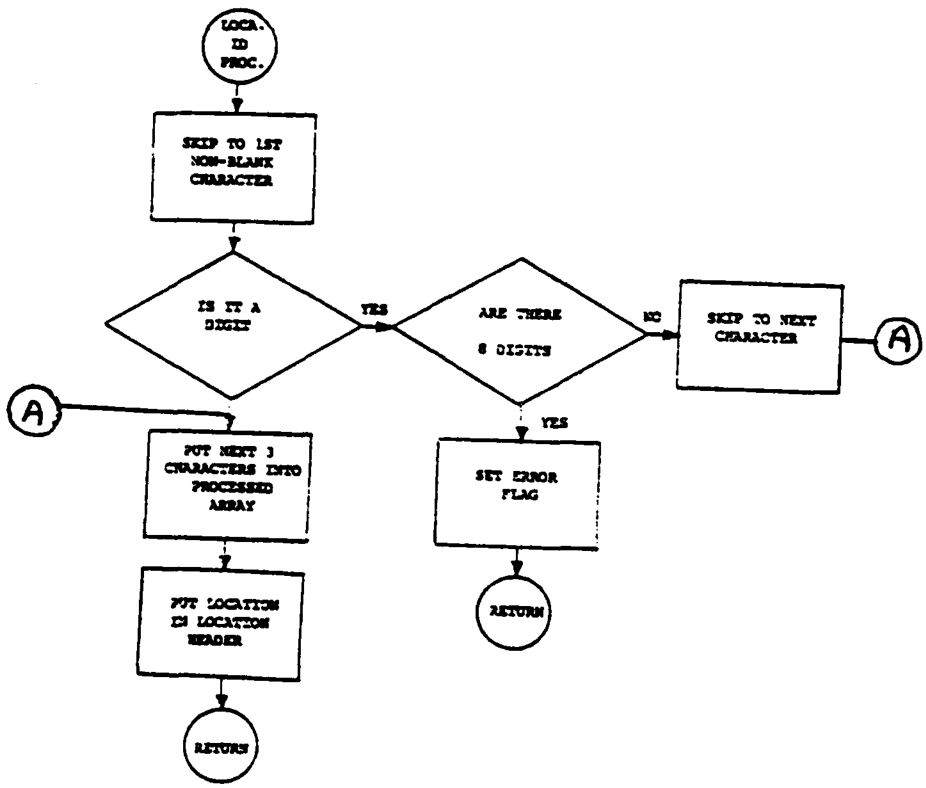


FIGURE C-5: FT PROCESSOR (Cont'd.)

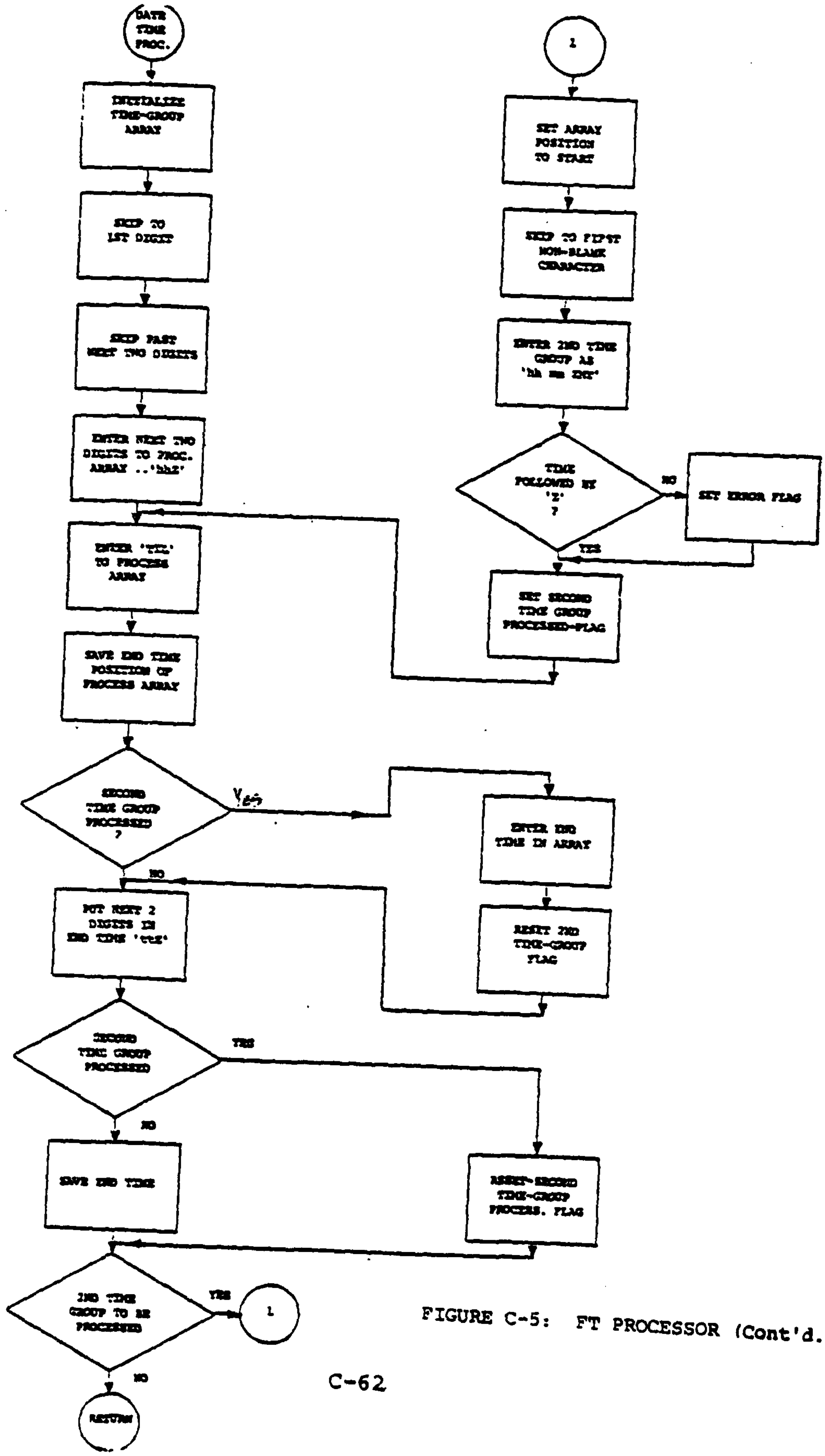


FIGURE C-5: FT PROCESSOR (Cont'd.)

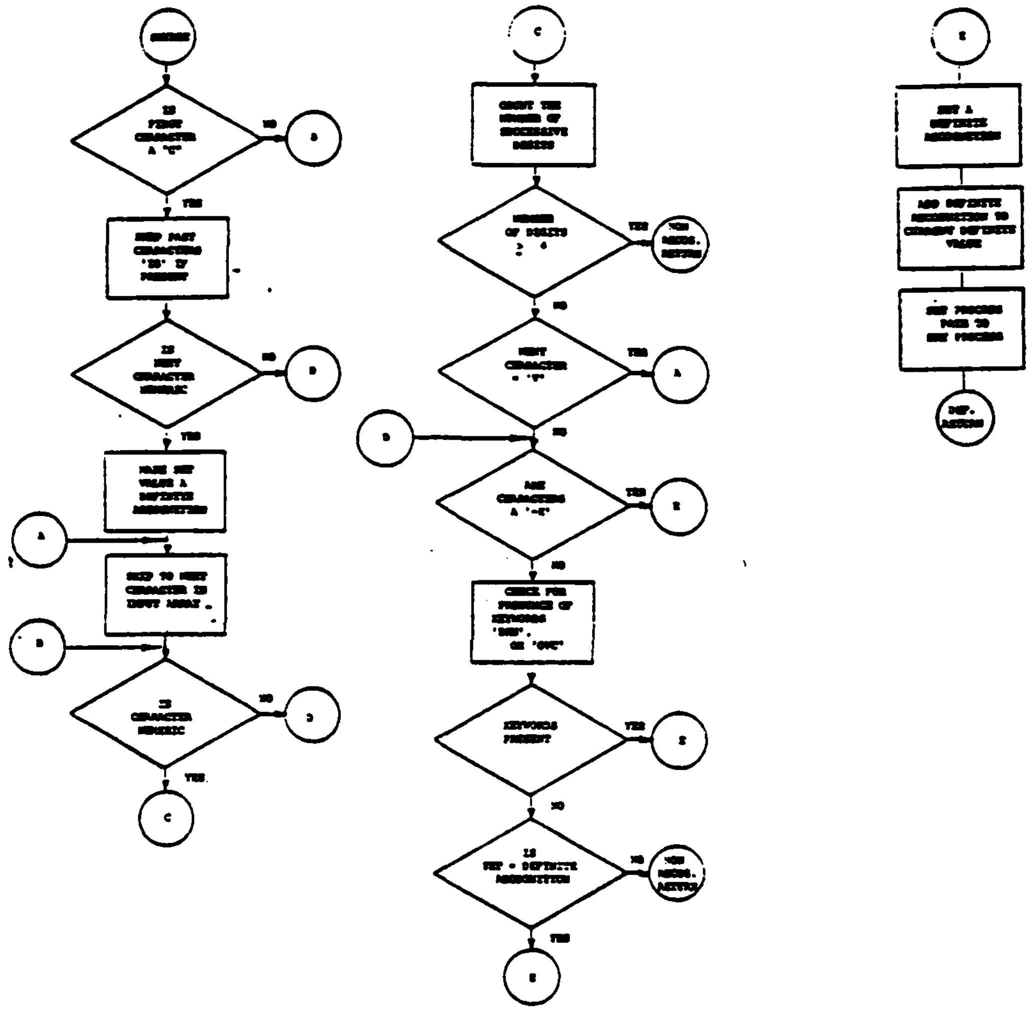


FIGURE C-5: FT PROCESSOR (Cont'd.)

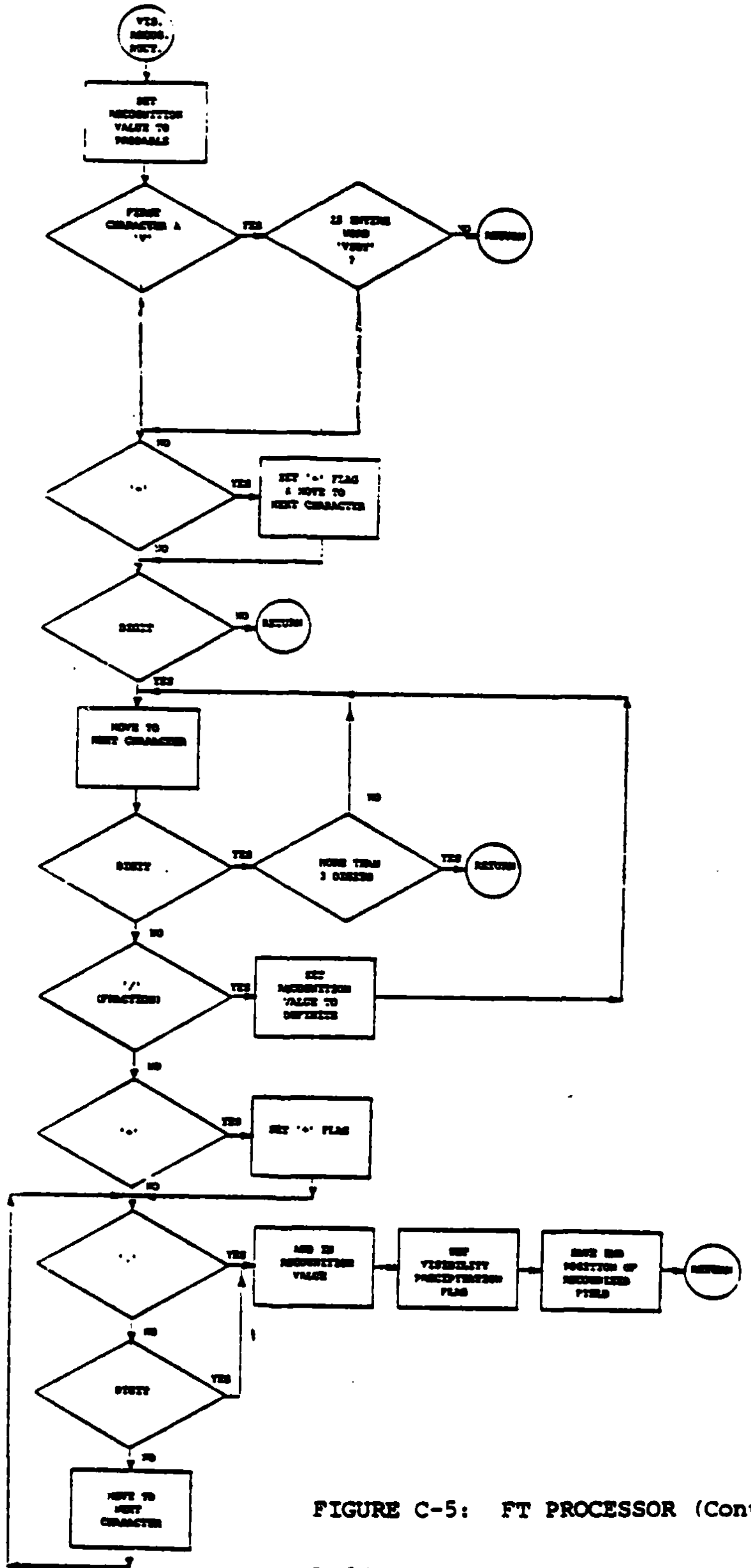


FIGURE C-5: FT PROCESSOR (Cont'd.)

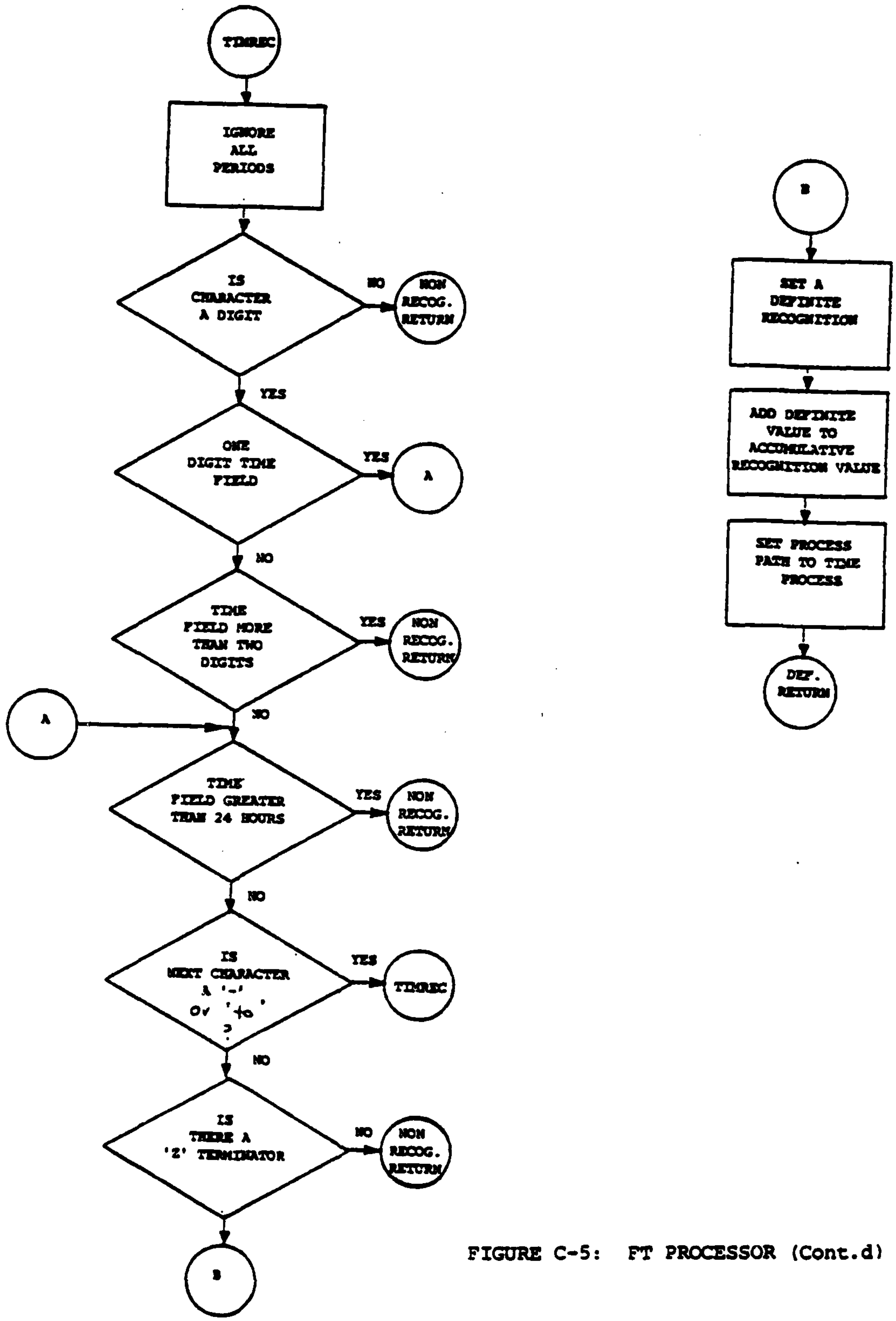


FIGURE C-5: FT PROCESSOR (Cont.d)

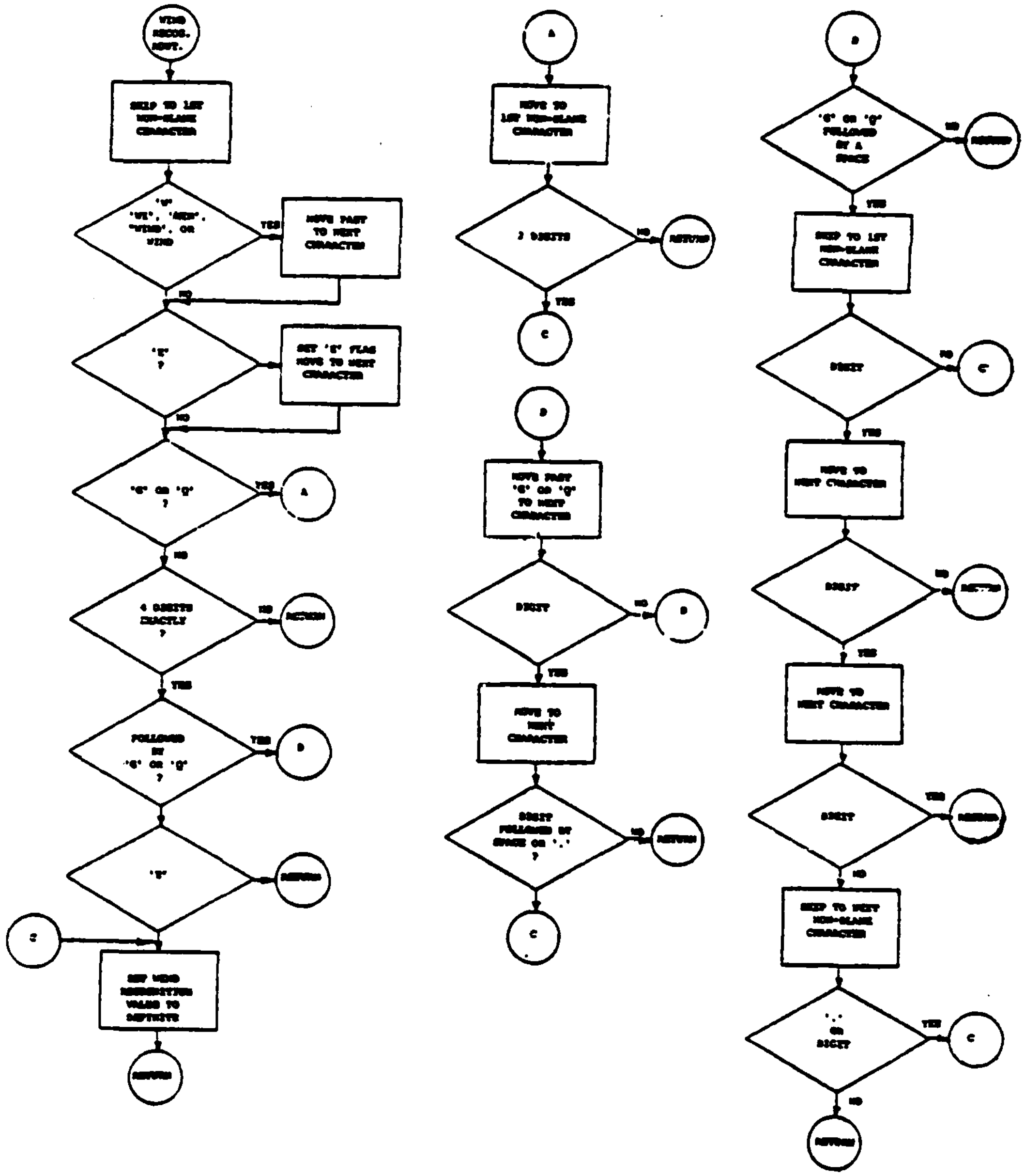


FIGURE C-5: FT PROCESSOR (Cont'd.)

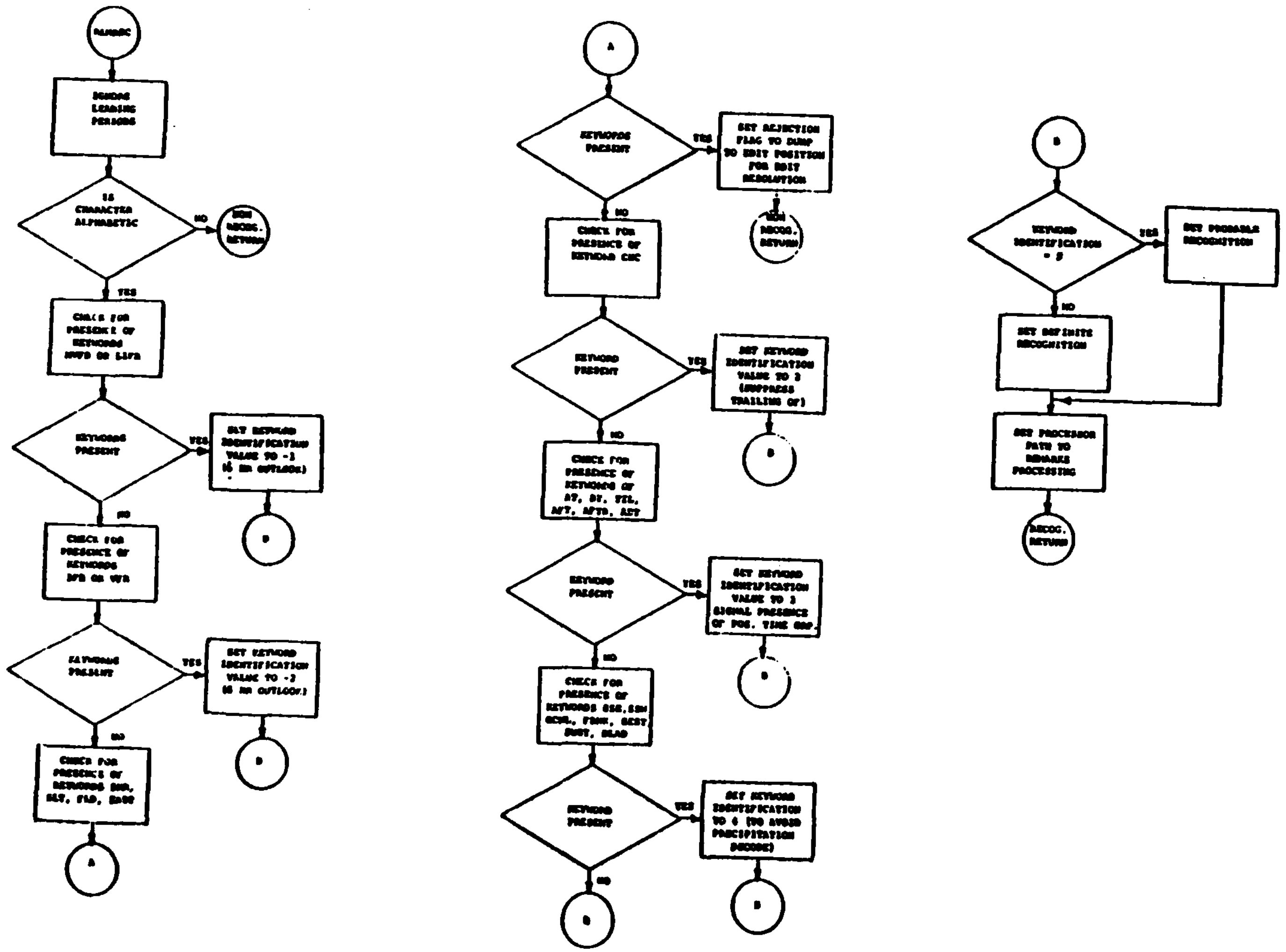


FIGURE C-5: FT PROCESSOR (Cont'd.)

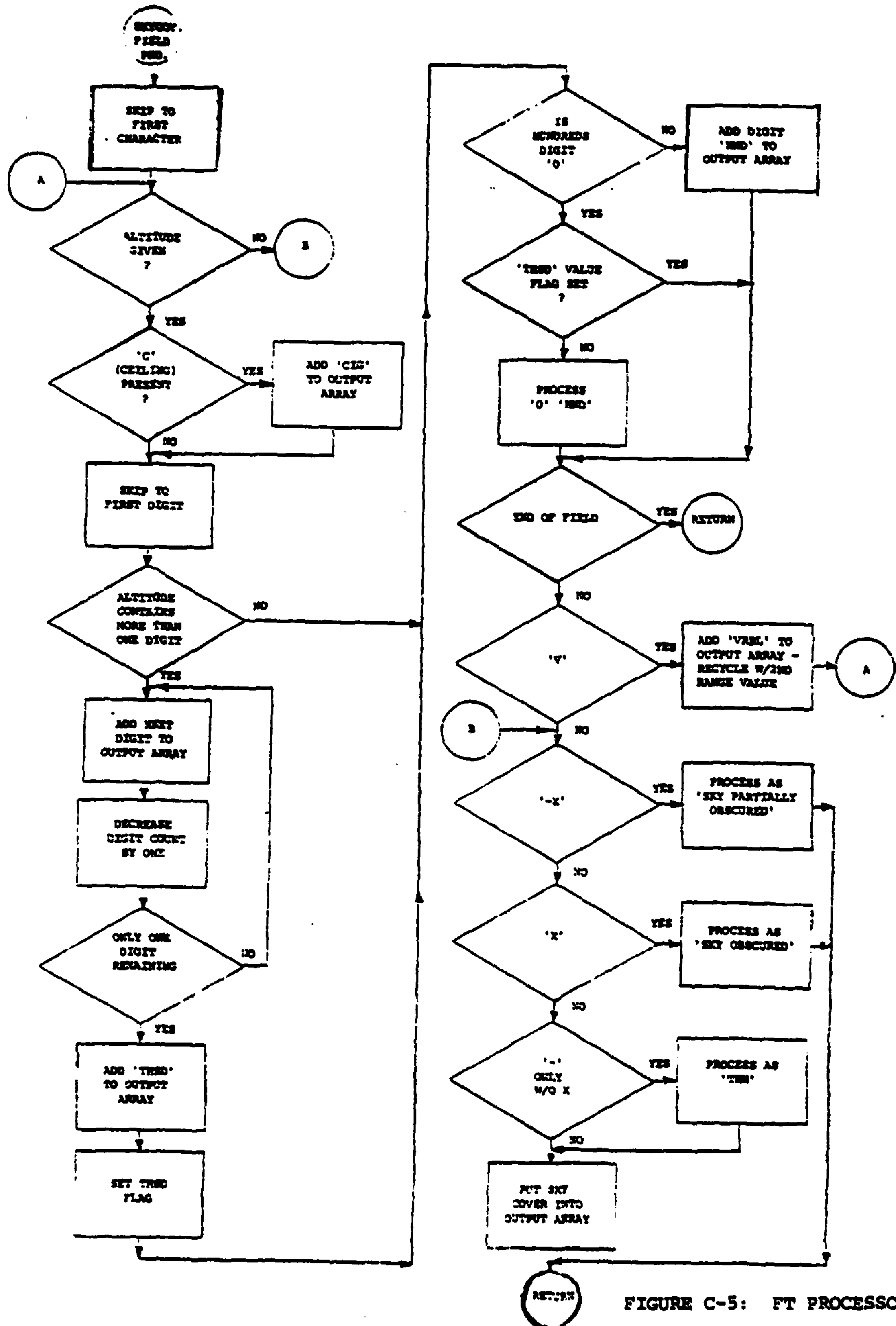


FIGURE C-5: FT PROCESSOR (Cont'd.)

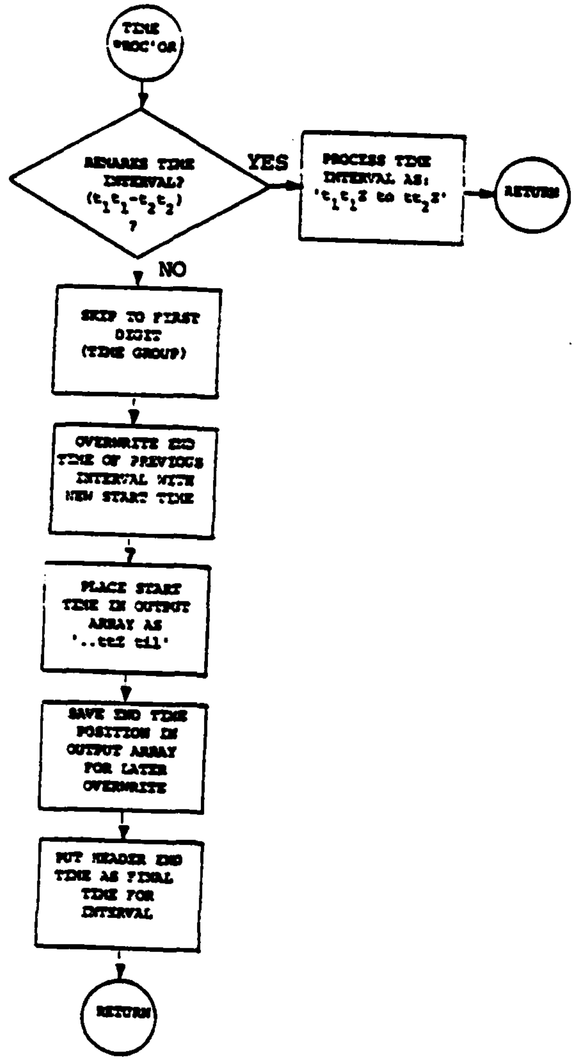


FIGURE C-5: FT PROCESSOR (Cont'd.)

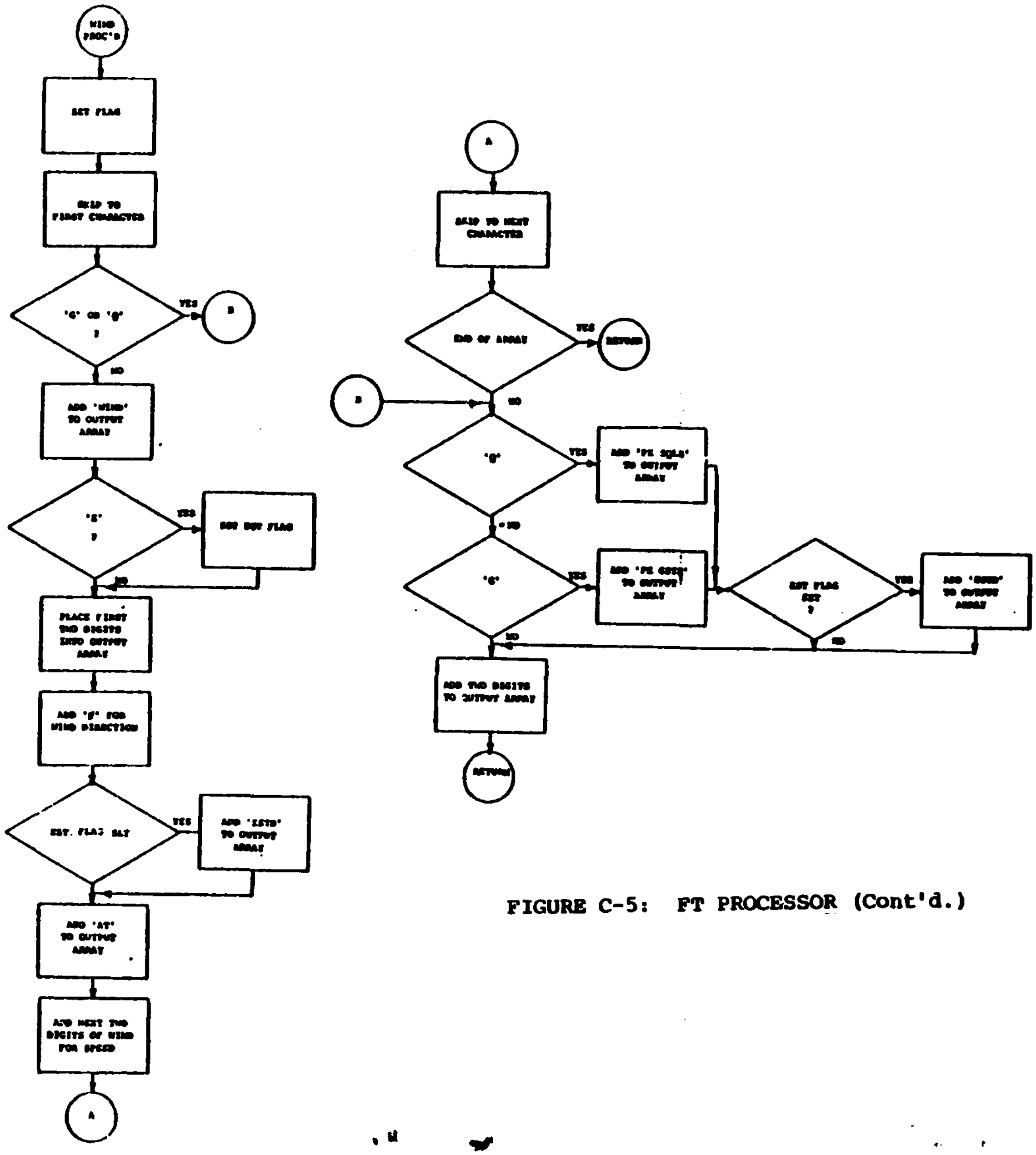


FIGURE C-5: FT PROCESSOR (Cont'd.)

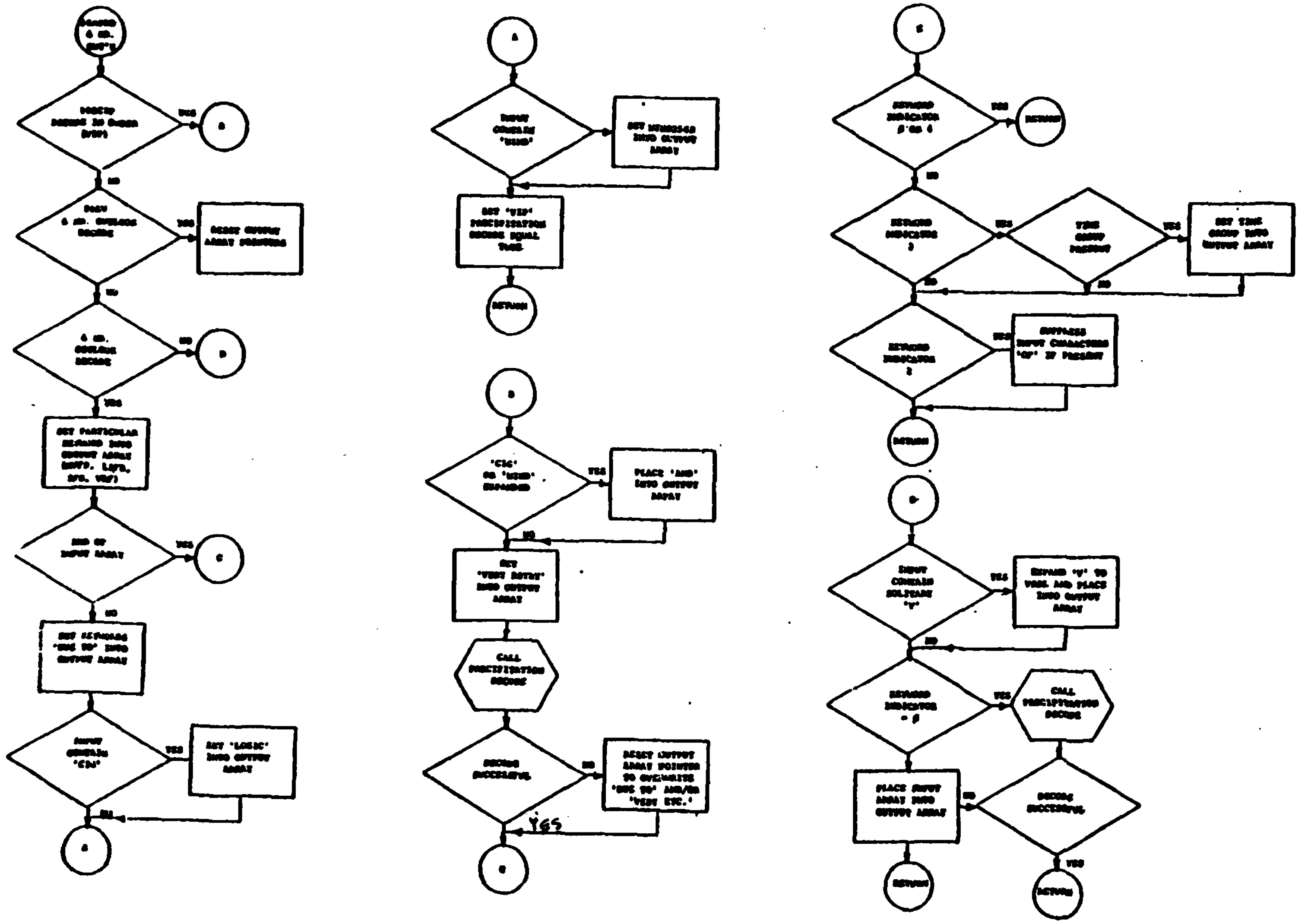


FIGURE C-5: FT PROCESSOR (Cont'd.)

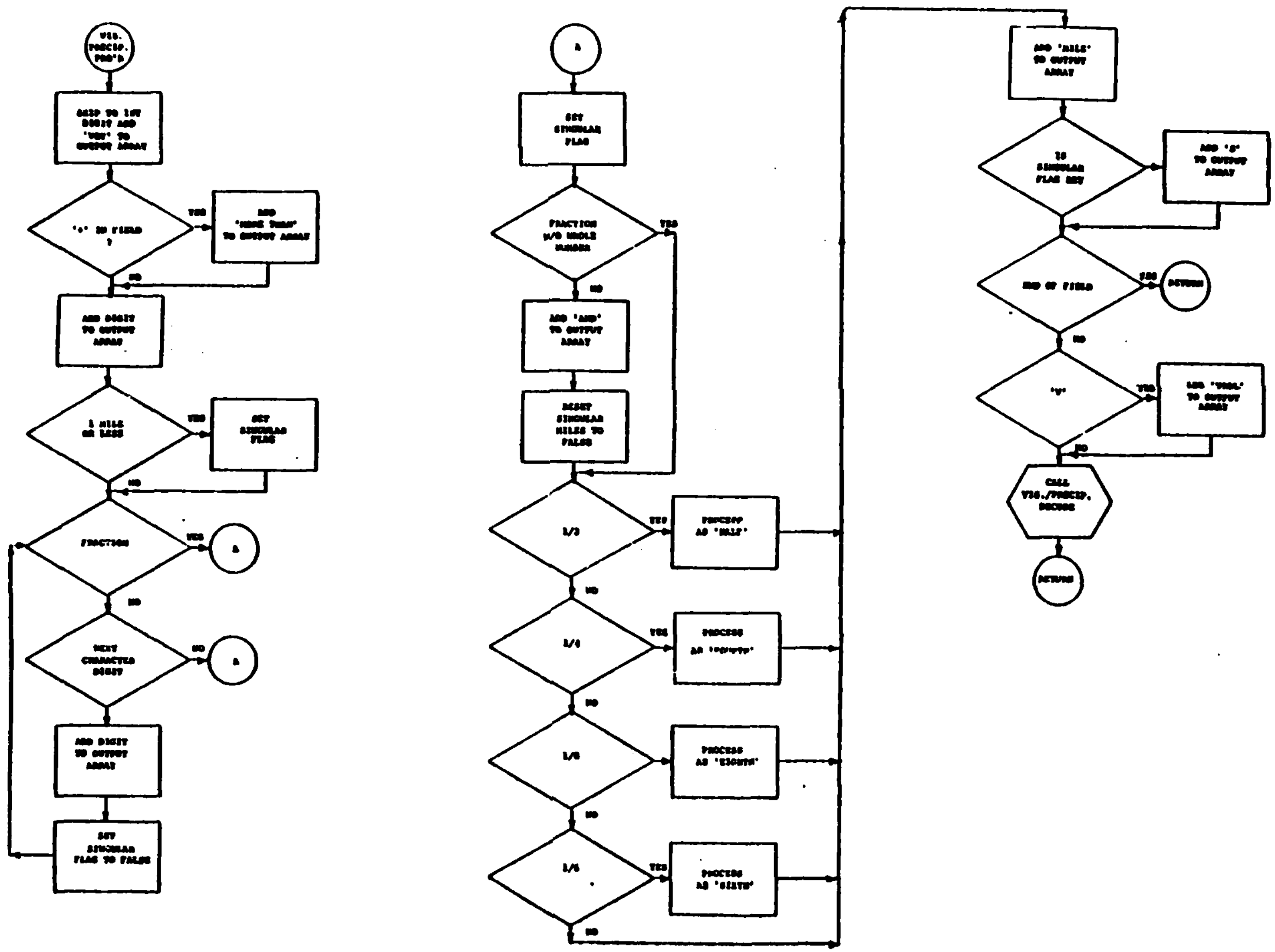


FIGURE C-5: FT PROCESSOR (Cont'd.)

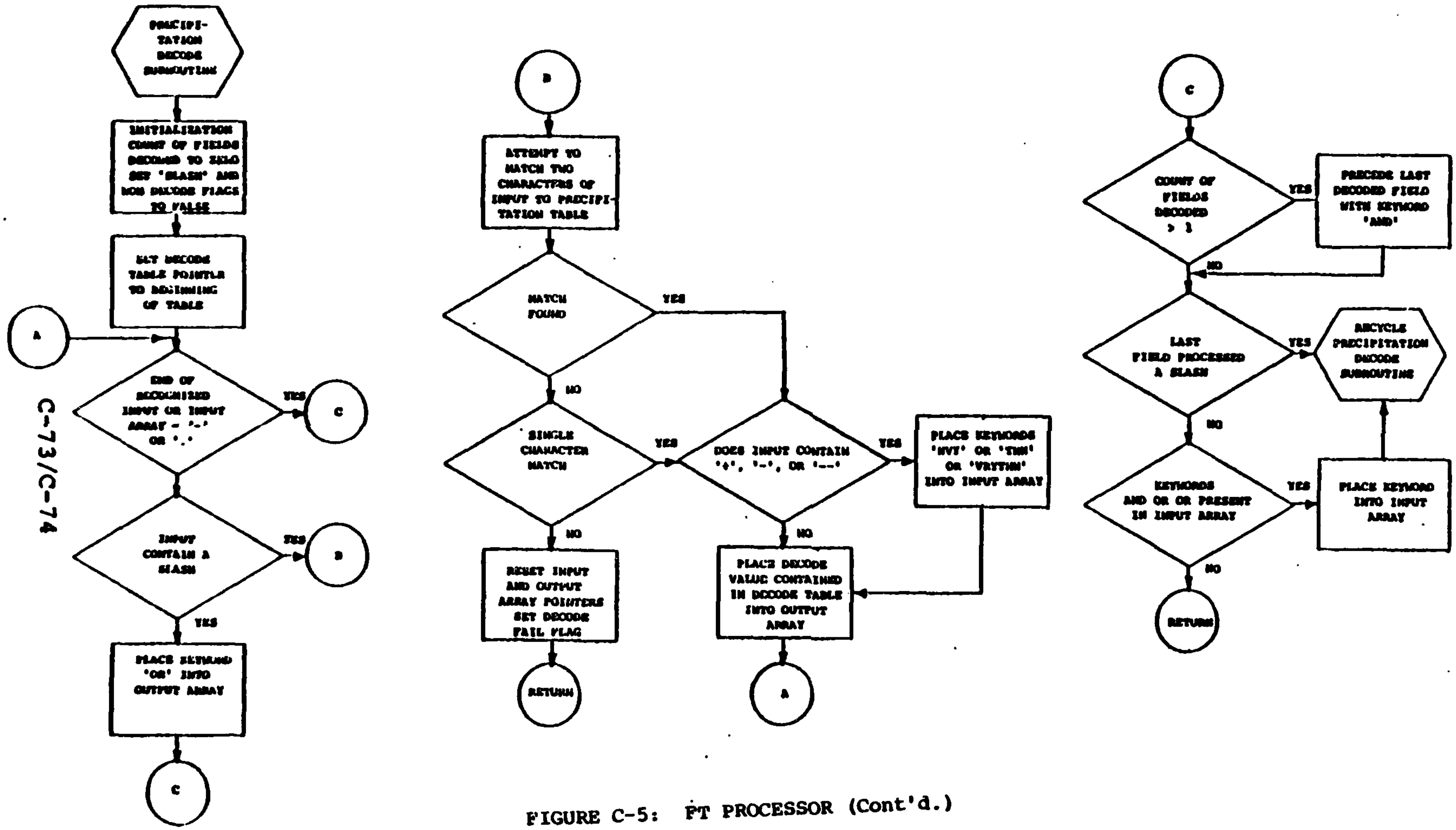
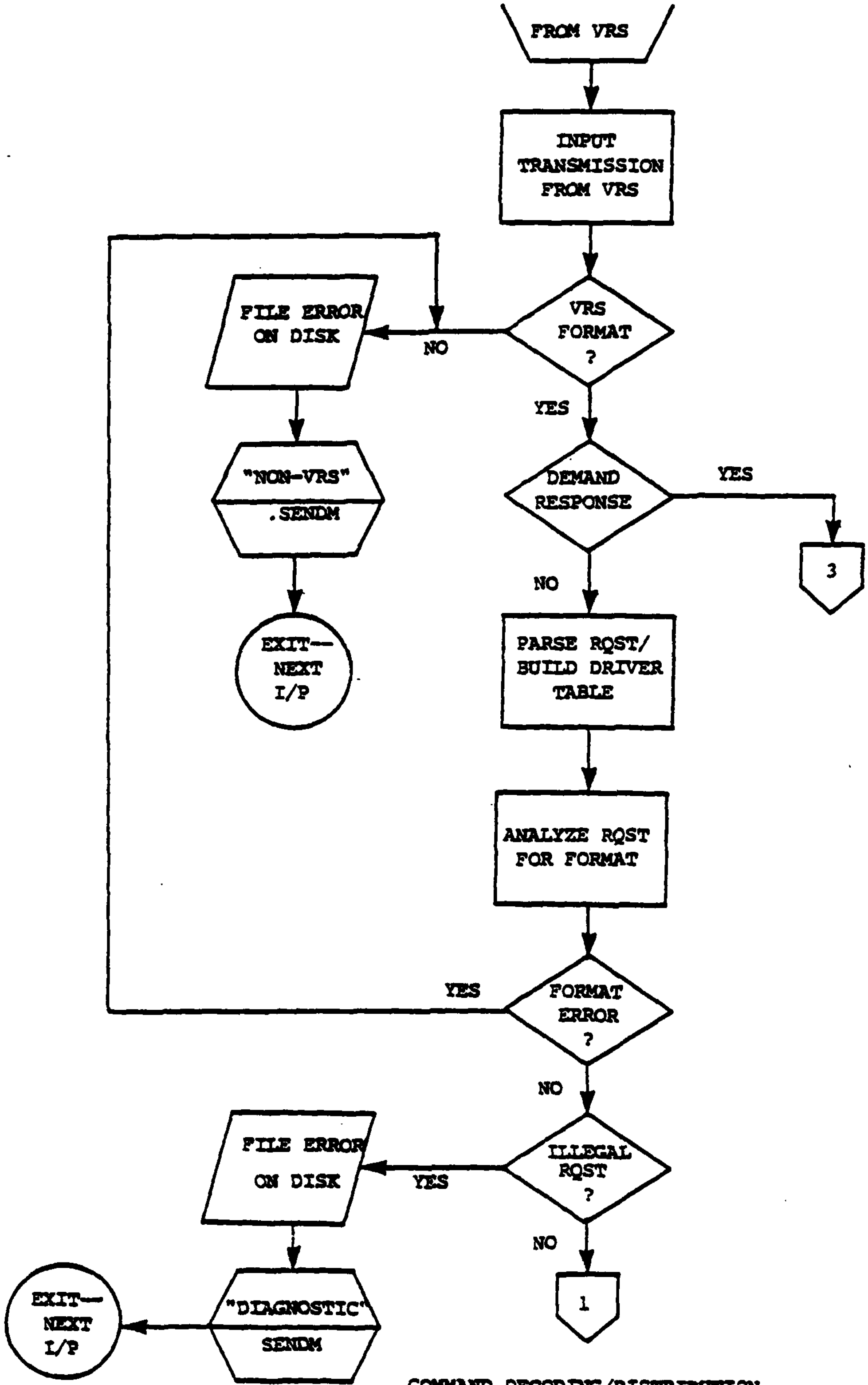


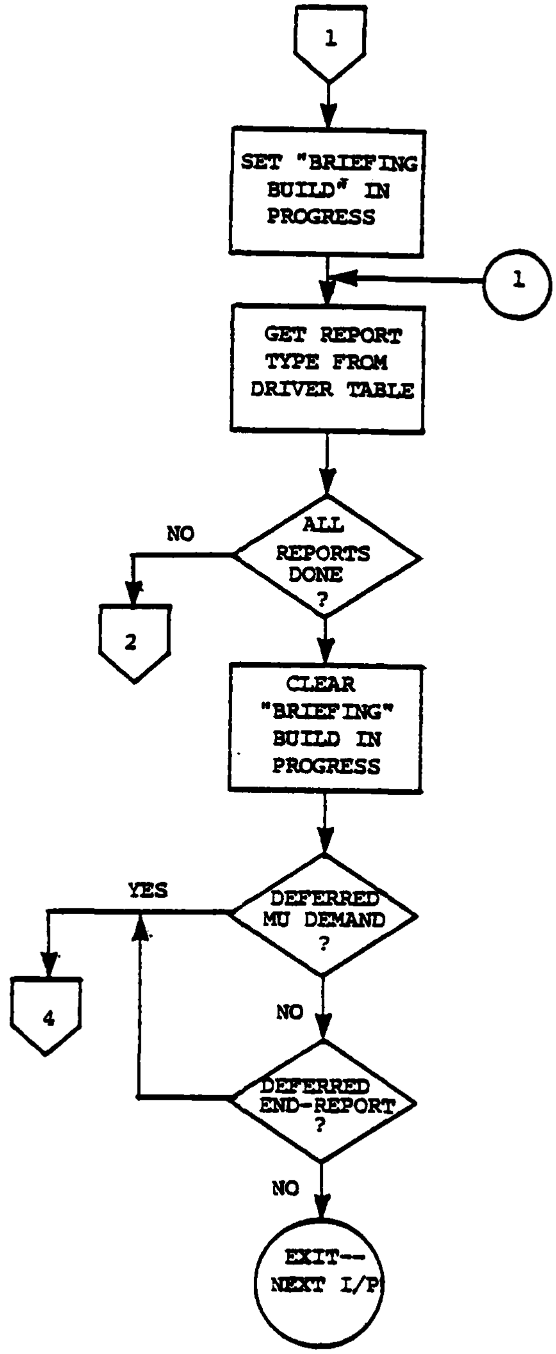
FIGURE C-5: FT PROCESSOR (Cont'd.)

C.6 RETREV



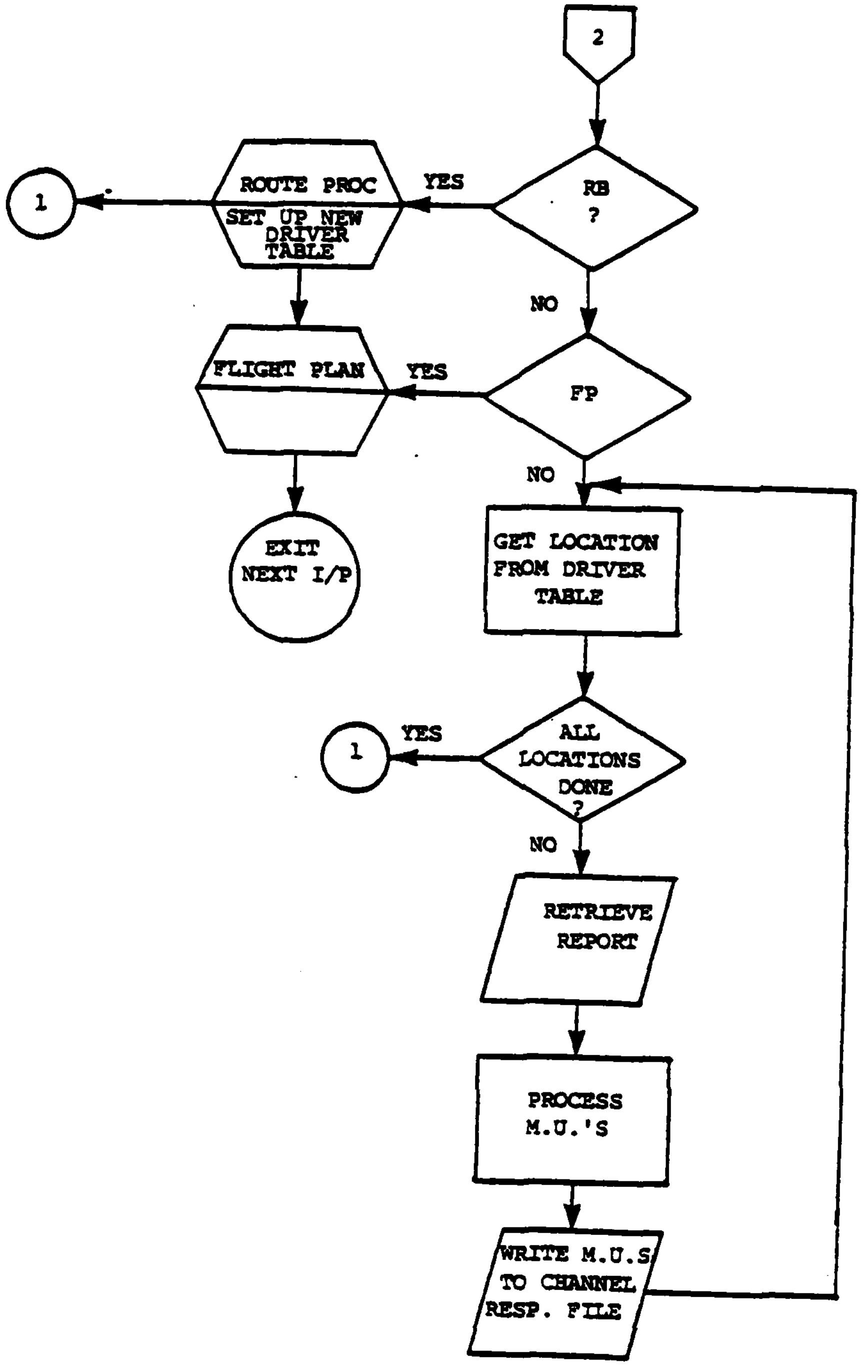
COMMAND DECODING/DISTRIBUTION

FIGURE C-6: RETREV



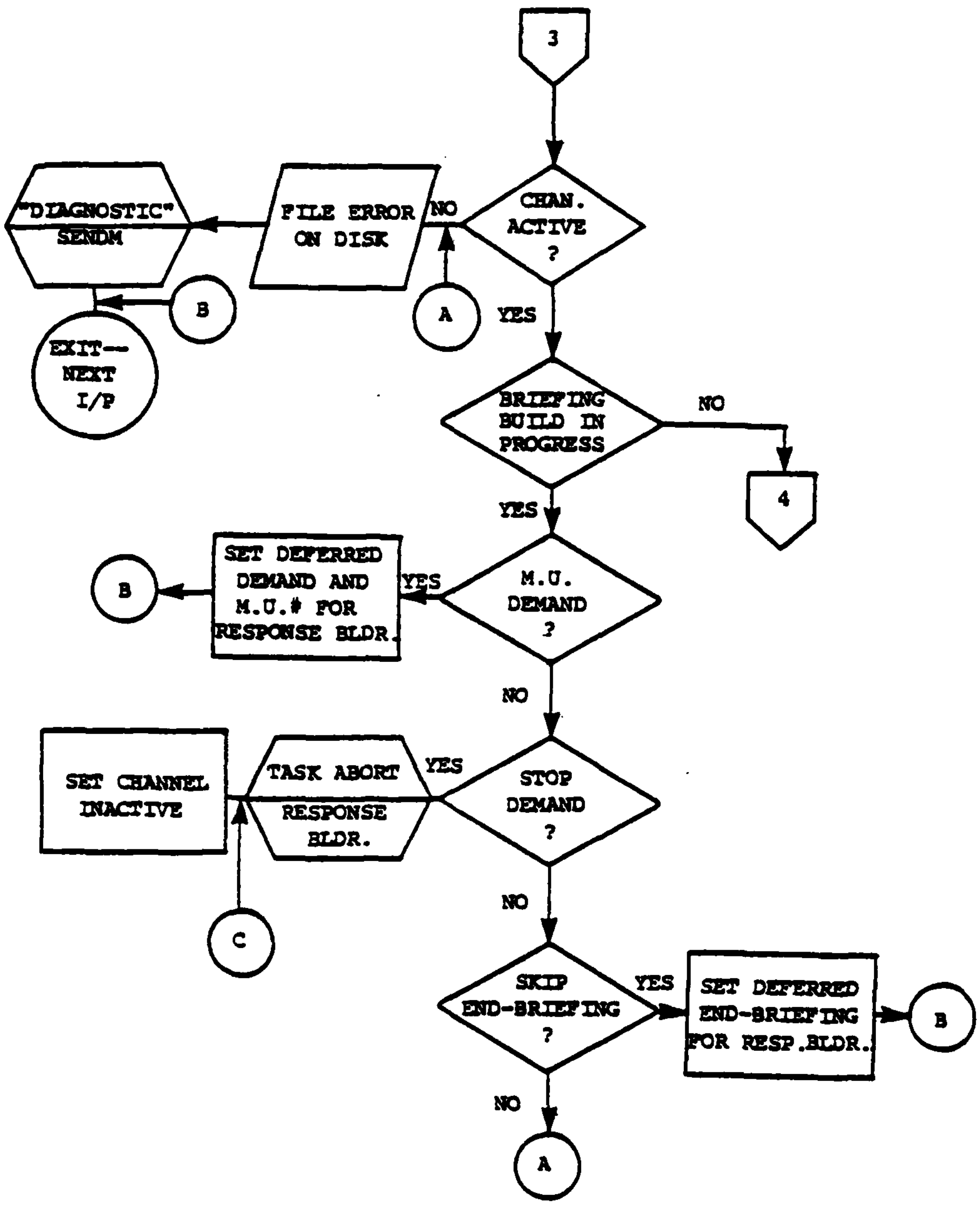
RESPONSE BUILDER

FIGURE C-6: RETREV (Cont'd.)



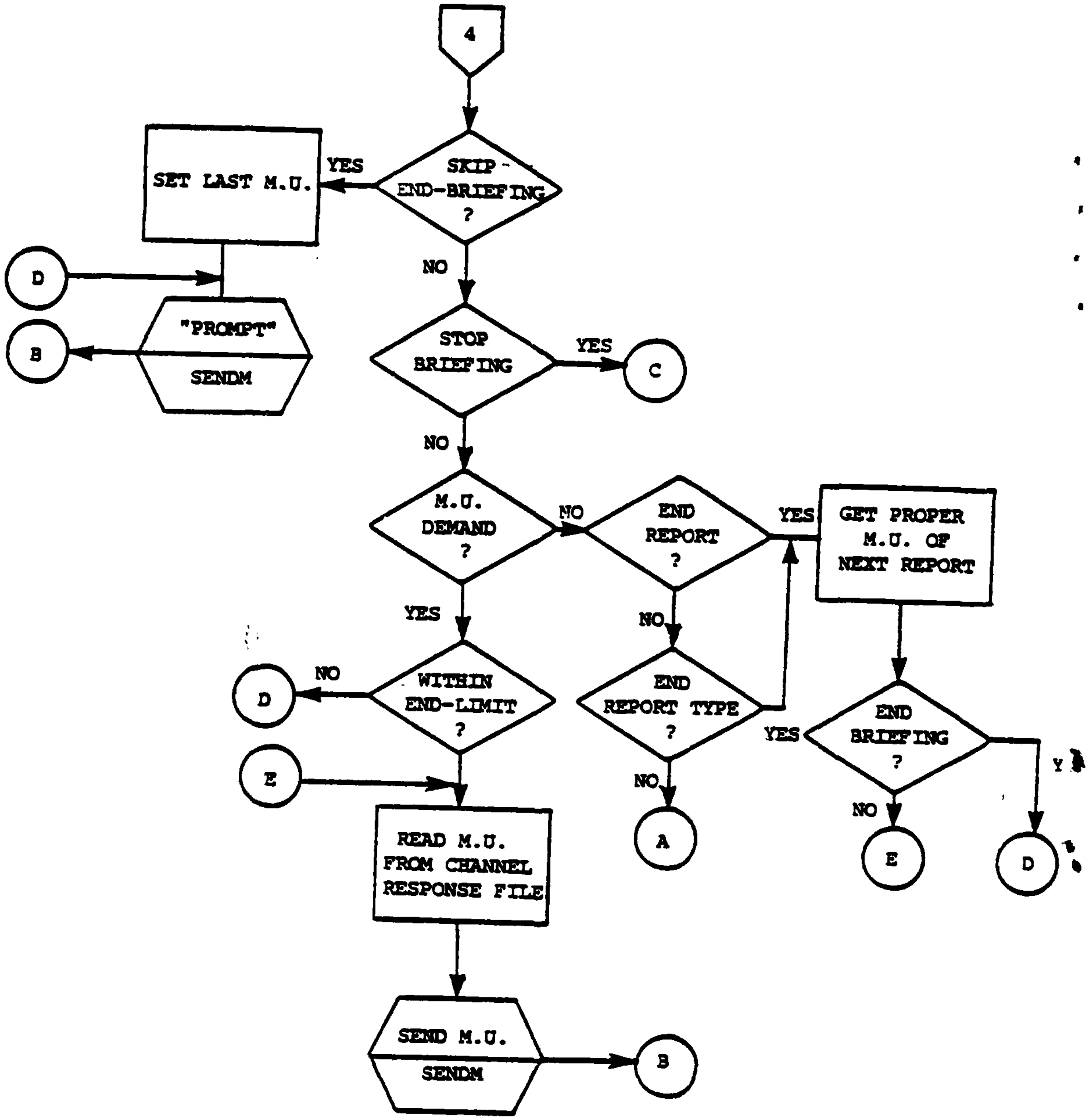
RESPONSE BUILDER

FIGURE C-6: RETREV



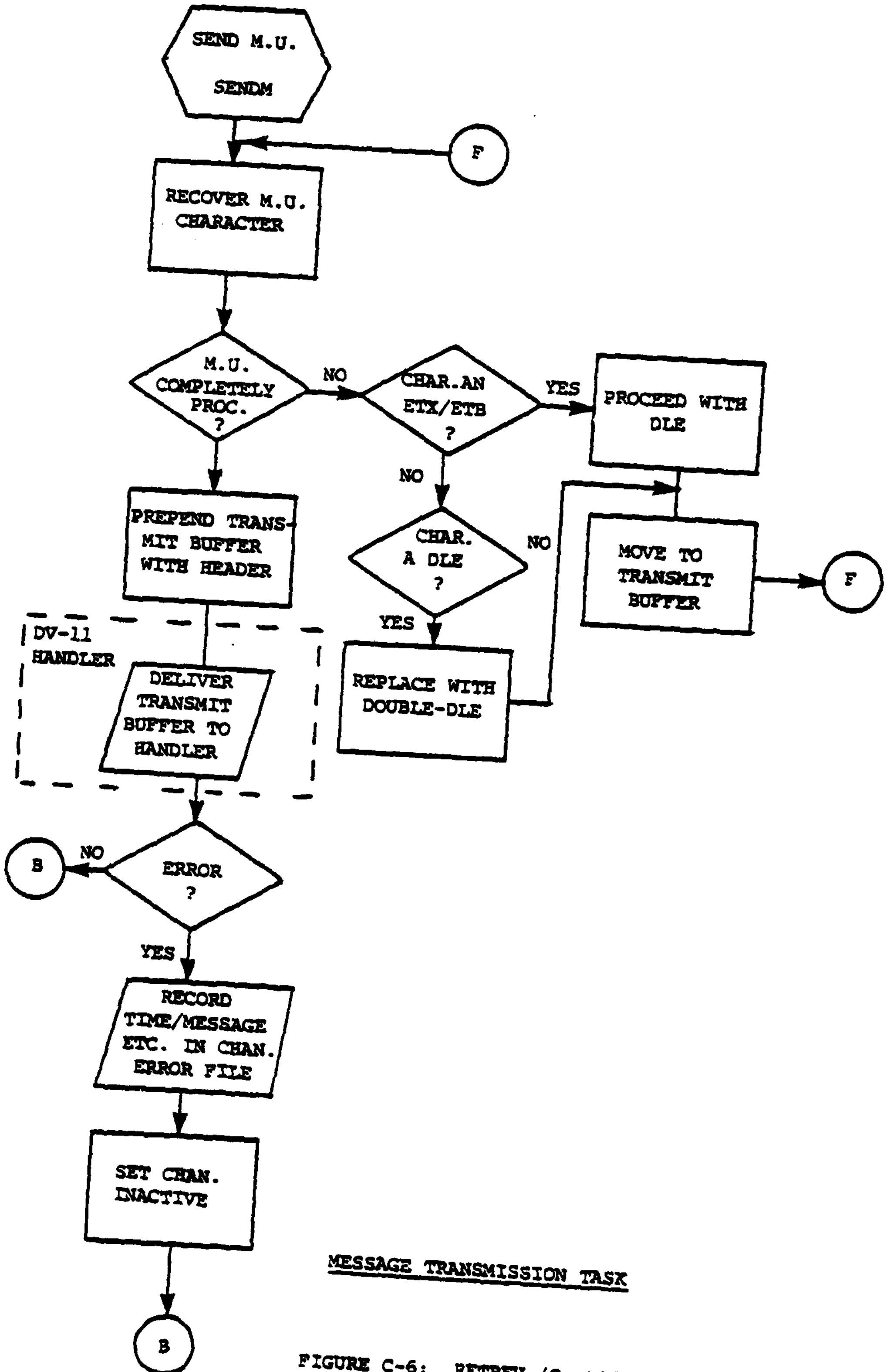
DEMAND RESPONSE

FIGURE C-6: RETREV (Cont'd.)



DEMAND RESPONSE

FIGURE C-6: RETREV (Cont'd.)



MESSAGE TRANSMISSION TASK

FIGURE C-6: RETREV (Cont'd.)

APPENDIX D

REPORT OF NEW TECHNOLOGY

There have been no inventions or important discoveries made during the performance of this contract. However, the Voice Response System has been implemented using a unique software design on both the PDP-11/34® and the PDP-11/70®.

The PDP-11/34 software was designed to run under the single-user operating system RT-11 and operationally to perform as a multi-user (20-channel) system. This was accomplished by using the RT-11 capability of asynchronous I/O with assigned priority. The priority assignment for each VRS I/O component was developed for uninterrupted speech on each channel.

Each channel follows a table-driven protocol using separate storage areas in memory to maintain channel status after asynchronous I/O completion. Improvements were made to the system in upgrading VRS from 10 to 20 channels by taking advantage of the extended memory management of RT-11 to utilize the 32K of memory added to the system. This involved the allocation and access of the speech buffers and dictionary in upper memory. See section 2.2 for the software description.

A single-user/20-channel design has been implemented for the PDP-11/70 weather retrieval program. See section 2.4.4. It employs separate storage areas for maintaining channel-briefing status upon completion of the asynchronous I/O. A unique file system has been designed for storage and retrieval of the weather reports processed on the PDP-11/70. This file system allows multi-task (processor and retrieval tasks) access and update without conflict. It exercises the RSX-11 operating system feature of shared global common areas in memory for the file block map and for multi-task communications. This system is described in section 2.4.

110 Copies

DA
FILM