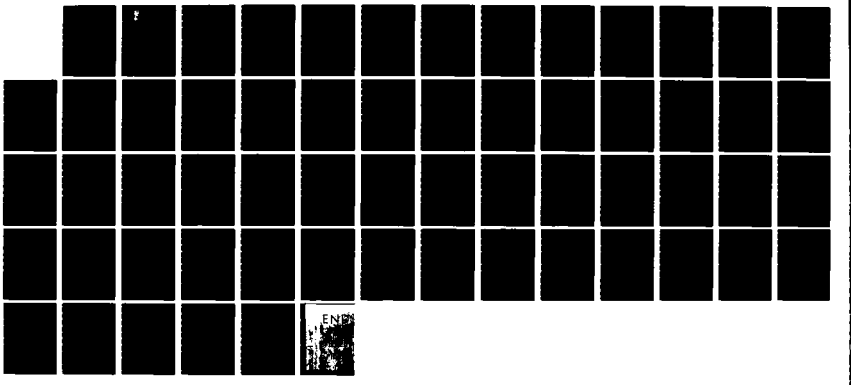
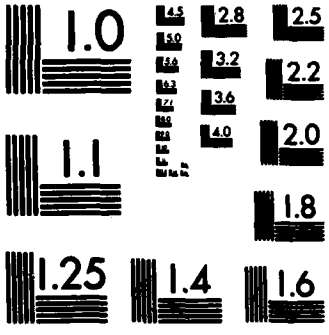
AD-A133 431 THE AXIS TEST BOX SOFTWARE REPORT(U) ROYAL SIGNALS AND RADAR ESTABLISHMENT MALVERN (ENGLAND) A L SIMCOCK APR 83 RSRE-MEMO-3585 DRIC-RR-89152 1/1

UNCLASSIFIED F/G 14/2 NL





RSRE
MEMORANDUM No. 3585

ROYAL SIGNALS & RADAR ESTABLISHMENT

THE AXIS TEST BOX SOFTWARE REPORT

Author: A L Simcock

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
RSRE MALVERN,
WORCS.

ROYAL SIGNALS AND RADAR ESTABLISHMENT

Memorandum 3585

Title: THE AXIS TEST BOX SOFTWARE REPORT

Author: A L Simcock

Date: April 1983

SUMMARY

This memorandum describes in detail the software program which the M68Ø9 microprocessor obeys to perform the functions of the AXIS Test Box.

Extensive use is made of Flow Diagrams where these either enhance or replace wordy descriptions.

RSRE MEMORANDUM No 3585

THE AXIS TEST BOX SOFTWARE REPORT

A L Simcock

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

MMI	Man Machine Interface
PPI	Programmable Peripheral Interface
VDU	Visual Display Unit
GTU	Group Terminating Unit
o/p	Output
i/p	Input
Ø	The symbol Ø is used to distinguish zero from the letter O.
HEX	Hexadecimal
TX	Transmit
RX	Receive
EOT	End of Text
CC	Check Code
RA	Register Address

1 INTRODUCTION

This is the third in a series of 4 reports on the AXIS Test Box. The first report (Ref 1) is the Introduction to the Test Box and describes the role of the Test Box within the Axis experimental project. The second report (Ref 2) is the Operating Guide, and details the rationale for the nature of some of the commands/functions provided. The third report (Ref 3) is the Hardware report.

This software report describes briefly the program which controls the hardware and provides the MMI described in Ref 2.

This report is brief since it is felt that the commented program listing is almost self explanatory. The report mainly details program flow and structure and uses diagrams, wherever possible, so that program construction may easily be followed. Data and data flow are also described in Sections 17 to 19. Annex B is a list of the drawing symbols used.

The program listing is held in a file called FINALTB on floppy disc for the FUTUREDATA Microprocessor Development System in use in T24 (Ref 4).

It can be seen from the diagrams illustrating program flow that the body of each of the major routines consists mainly of a series of subroutine calls.

The body of each of the major routine will be described very briefly using the illustrations as a guide to the subroutines being used.

The subroutines themselves will be described in greater detail, since this is where the majority of the work of the program is carried out. The preparation of data for tramsmission, and the hardware handlers are also described separately.

The report is written in this way in order that the 'casual' reader should be able to obtain a superficial understanding of the program and the 'determined' reader should find the detail necessary for a full understanding. Ref 2 desscribes fully the functions provided by the Test Box, this report describes how those functions are provided.

2 INITIALISATION

The initialisation routine sets the Programmable Peripheral Interface devices (PPIs) to their initial state and programs them for input or output; some internal flags are also initialised.

3 BASIC COMMAND MODE

The Basic Command Mode routine allows the user to select any one of the 5 functional routines (see Fig 1).

4 ASSEMBLE MESSAGE

The Assemble Message routine takes the user step by step through the input of the parameters necessary to assemble a message ready for transmission (see Fig 2). Individual parameter entry is controlled by subroutines and the Assemble Message routine calls these subroutines in the correct order to enable a complete message to be assembled.

5 DISPLAY ON REQUEST

The Display on Request routine allows the user to display input messages which have been stored. The user should enter the segmentation format required for the display, then the user should enter the number of the block to be displayed. Fig 3 illustrates the flow of the program.

6 AUTO DISPLAY

The Automatic Input Display routine is very simple. The user is allowed to select the display options required for display of the input messages and is then able to jump to the hardware handlers to receive the input messages (see Fig 4).

7 REPEAT

The Repeat routine firstly causes the current Menu to be displayed on the VDU and then allows the user to select parts of the message to change, the user also has the option to display the Menu, Quit the routine or Go to the hardware handlers (see Fig 5).

8 DISPLAY CURRENT MENU

This routine is itself written as a subroutine which may be called from the Repeat routine. The program flow is illustrated in Fig 6.

9 PREPARE DATA FOR TRANSMISSION

The entry to this routine is either from the Assemble Message or Repeat routines.

Firstly, the hardware interfaces (PPIs) are set to the transmit mode. The Message Type is then checked. If the Type indicates an A message or T (for both Messages) then a pointer to the A message is passed to the FMFORP (Format for Parity) subroutine. If the Type is B then the pointer is set to the B message before calling the FMFORP subroutine. The Format of the message is then determined in order that the correct number of parity bits may be generated. The CALCP (Calculate Parity) subroutine is then called:-

a Once for a GTU message

b Twice for a MATRIX Command message

c Three times for a MATRIX Connection message

When the necessary parity bits have been generated the whole message with its newly generated parity bit(s) is transformed by the TFTOTX (Transfor to Transmit Format) subroutine into a format directly applicable to loading into the hardware output registers.

The type is checked again and if TYPE = T (for both messages) the whole procedure is repeated using a pointer to the B message. Figure 7 illustrates this flow in diagrammatical form.

9.1 Prepare for Hardware Handlers

This is a continuation of the above routine but is also the entry point from the Auto Receive Mode. Figure 8 illustrates the program flow in diagrammatical form.

Firstly the last valid input stores (for A and B message responses) are preset and then the start address of the input message store is initialised. At this point if this routine was entered from the Auto Receive Mode the Programmable Peripheral Interface (PPI) devices, which interface directly to the hardware input and output boards, are set to receive inputs and the program control transfers to the hardware handling routines. If entry was not via Auto Receive the output message(s) must be loaded into the hardware output register (S).

If the message Type is A or B then O/P register 1 is loaded with the A or B message and if more than one message is to be sent (ie Transmit Options Continuous or Multiple have been chosen) then O/P Register 2 is also loaded with the A or B message. If the Type is T (for both messages Together) then O/P register 1 is loaded with the A message and O/P register 2 is loaded with the B message.

The time interval between O/P transfers, from the hardware output transfer board, is determined either by a fixed hardware delay or by software. The hardware delay will only be activated if the input message response to each output is not to be interrogated, ie delay is only activated if:-

a No Wait condications are set

b Display Option chosen is NONE

under these circumstances the number of output transfers is loaded into the hardware counters and program control transfers to the hardware handlers.

If the above conditions are not met a single output will be sent and the input response will be interrogated by the hardware handlers, corresponding action will be taken, and control returned to this routine in order that the next output may be sent. In this way the time interval between output transfers is determined by the software.

If the Transmission Option selected is Multiple the number of outputs required is copied from MOPTNUM into OPTXNUM and this decremented at each output in order to call the hardware handlers to send the correct number of output transfers.

1Ø DATA MANIPULATION SUBROUTINES

1Ø.1 FMFORP ie Format for Parity Generation

This generation takes the dats identified as Message 1 or Message 2 in the internal message structure (Fig 25) and re-organises the data in order into a 42 byte block of data, so that it is in a suitable format for parity generation. (Fig 26A).

This routine should be called as a subroutine and given the address of the check code of the message (ie 1 or 2) for transformation. This address parameter should be stored in TEMP.

10.2 CALCP ie Calculate Parity

Three types of message each requiring different parity calculations have been identified in the User guide.

These are:

G = GTU

M1 = Matrix Command

M2 = Matrix Connection

The identification of the type of message is stored in AFORMAT for the A Message and BFORMAT for the B Message.

0 = GTU

1 = M1

2 = M2

The parity generating routine requires 3 parameters:

a Start bit number for parity generation in STAP

b End bit number for parity generation in ENDP

c Bit number to store the generated parity in PUTP

The routine should be called as a subroutine with the above 3 parameters defined. Setting these parameters allows the CALCP routine to be used to calculate parity bits over any data area within a message. (See also Fig 26 A).

10.3 TFTOTX ie Transform to Transmit Format

This routine takes the information from the 42 byte data area used for parity generation and transforms this into a format which will make it immediately usable by the Tx hardware handlers, ie the format illustrated in the Tx message structure, Fig 26B.

This routine should be called as a subroutine and requires the address of the first byte of the first or second (whichever is required) Tx Message Structure (see Fig 26B). The address parameter should be stored in TEMP 2 before the routine is called.

10.4 LOAD HARDWARE REGISTERS

There are 5 subroutines under this heading:

OPAMS1 Loads A message into O/P Register 1

OPAMS2 Loads B message into O/P Register 2

OPBMS1 Loads B message into O/P Register 1

OPPMS2 Loads B message into O/P Register 2

PUTBYOU Generates the parallel load pulse and selects the shift register to be loaded.

The A or B message stored in the Transmit Message (Fig 26B) is loaded byte by byte into successive 8 bit shift registers which form the O/P registers. Six load sequences are necessary to complete the loading of each O/P register.

11 BASIC INPUT SUBROUTINES (See also Fig 9)

11.1 GETIP Get Input

This subroutine gets characters from the VDU, checks the character is not a back space and counts the number of input characters. It also checks that the number of input characters does not exceed the maximum required by the calling routine.

The calling routine should set the maximum number of characters required in the variable called MAXNUM and call GETIP as a subroutine.

The GETIP subroutine will return to the calling routine upon the entry of a Carriage Return on the VDU. The actual number of characters entered will be in the variable COUNT and the characters will be stored on the U stack. The PULU A instruction used by the calling routine will extract input character from the stack and put it in the A register. The first character available at the top of the stack is the first character enetred on the VDU.

eg If the User entered A B C D, these characters would be stored on the U stack and successive PULU A instructions would retrieve A, B, C and D respectively.

11.2 GETHEXIP Get Hex Input

This subroutine gets Hex input from the VDU. It uses the GETIP subroutine previously defined. The GETHEXIP subroutine checks the validity of the input characters (to ensure hex input) and also counts the number of input characters and checks also that the number of input characters does not exceed the maximum required by the calling routine.

The calling routine should set the maximum number of characters required in the variable MAXNUM and call GETHEXIP as a subroutine.

The GETHEXIP subroutine will transform the input characters into HEX bytes, ie 2 input characters constitute one HEX byte. GETHEXIP assumes leading zeroes, ie if the calling routine set a maximum number of input characters as 4 requiring an input between ØØØØ and FFFF, and the User enter[illegible] 123 carr[illegible]age return the GETHEXIP subroutine would transform this into [illegible]

This function as defined as NORMALISING the hex input since a normal hex byte contains 2 input characters.

Normalising is only carried out in the event of an odd number of characters being entered on the VDU,

ie if the User was to enter 23 or this would remain as 23.

The GETHEXIP subroutine returns to the calling routine when a carriage return is entered on the VDU. The actual number of input characters is returned in COUNT. The number of Hex bytes is in REVCO and the hex bytes are stored on the U stack.

Examples 1 and 2 below illustrate more precisely the values returned by GETHEXIP. In both examples the maximum number of allowable input characters set by the calling routine is 6.

Example 1

User enters 1 2 3 4 5 cr

GETHEXIP returns with COUNT = 5

REVCO = 3

USTACK = Ø1, 23, 45

Example 2

User enters 1 2 3 4 cr

GETHEXIP returns with COUNT = 4

REVCO = 2

USTACK = 12, 34

11.3 INDECNU Input Decimal Number

This subroutine gets decimal characters from the VDU and transforms them into a hex value. The calling routine should set the maximum number of input characters in MAXNUM. The absolute maximum allowed is 5 decimal characters representing FFFF Hex or 65,535 decimal. INDECNU checks the validity of the input characters, also checks that the count does not exceed the maximum number set by the calling routine and also checks that the input does not exceed 65,535.

The calling routine should set the maximum character count acceptable in MAXNUM and call INDECNU as a subroutine. INDECNU will return to the calling routine when a carriage return is entered on the VDU. The hex representation of the decimal input is in the D register.

11.4 INASC

This subroutine in the ALS M68Ø9 Monitor program is used by GETIP and INDECNU (see Fig 9) to obtain characters from the VDU. It interacts directly with the VDU interface devices and returns the input character in the A register.

12 BASIC OUTPUT SUBROUTINES

12.1 OUTASC

This subroutine takes as its parameter the start address of an ASCII string and prints the string on the VDU. The string should be terminated by the End of Text (EOT) character (Hex Ø4).

12.2 OUTHEX

This subroutine takes as its parameter the address of an 8 bit byte which is printed on the VDU as a pair of Hex digits.

12.3 OUTDEC

This subroutine takes the address of a 16 bit number to be printed on the VDU as a decimal number.

The INASC, OUTASC, OUTHEX and OUTDEC subroutines are all part of the ALS M68Ø9 Monitor program. For a full description of these subroutines see Ref 5.

13 DATA ENTRY SUBROUTINES

Data entry subroutines 13.2 to 13.6 require the address of either the A or B internal message structure (CCIMS or CC2MS in Fig 25) to be stored in a parameter passing location TEMP4.

13.1 Input Message Type GETTYPE (See also Fig 1Ø)

Firstly the request to enter message type is output to the VDU. The type is entered via the GETIP subroutine. The input is validated and if invalid an error message is sent and the routine restarted. The internal identifier MSTYPE (See Fig 25) is loaded with Ø, 1 or 2 corresponding respectively to A, B or T input. If T is input the Transmission Option field is interrogated and if a single transmission has been selected the number of transmission in MOPTNUM is increased from 1 to 2, in order that a single A plus B Message may be output. Program control is then returned to the calling routine.

13.2 Input Check Code INPCC (See also Fig 11)

This subroutine firstly prints a request to enter the check code on the VDU. The check code input is then collected via the GETHEXIP routine and then validated. ie $Ø \leq CC \leq F$. If the check code is invalid the 'Invalid Input' error message is printed and the input request repeated. If the check code is valid the input is stored in the check code field of the correct message (See Fig 25).

13.3 Input Register Address INPRA (See also Fig 12)

Firstly this subroutine prints a request to enter the register address. The register address is entered via the GETHEXIP subroutine and then validated ie $Ø \leq RA \leq 7$. If invalid the 'Invalid Input' error message is printed and the subroutine restarted. If the RA is valid it is stored in the RA field of the correct message (See Fig 25).

13.4 Input Data GETDAT (See also Fig 13)

The enter data prompt is output to the VDU. Up to 9 hex digits of data may be input (via the GETHEXIP subroutine), this corresponds to up to 5 bytes of data. The number of bytes of data input is in the counter scratchpad REVCO. Data is removed from the U stack and stored in the correct data structure with the least significant byte being stored in either D11MS or D21MS (See Fig 25) (depending on whether data is for an A or B message respectively). When all the data has been pulled from the stack, the data structure up to and including the last byte of the data field D15MS or D25MS is filled with zeroes. Program control is then returned to the calling routine.

13.5 Input Parity GETPART (See also Fig 14)

Firstly the Enter Parity prompt is issued. The user response to the prompt "Parity True Y or N?" should be N to ensure false parity, any other character entered gives true parity. True parity is therefore the default value. The subroutine checks the response obtained via GETIP and sets the A or B message parity field (see Fig 25) to be either true or false.

13.6 Input Message Format/Structure MSSTR (See also Fig 15)

The request for Format prompt is first output to the VDU. The input is then obtained via the GETIP subroutine. The response is checked and if G the Message Structure field (see Fig 25) (of the correct message) is set to Ø ie GTU. If the input is neither G nor M an error message is printed and the subroutine re-started. If the input is M a check is made to see if a second parameter has been entered. If no second parameter has been entered it is requested and obtained via GETIP. This second parameter is checked and if neither 1 or 2 an error message is printed and the subroutine re-started. If the second parameter is 1 (ie Matrix Command) the Message Structure field is set to 1. If the second parameter is 2 (ie Matrix Connection) the Message Structure field is set to 2.

After the setting of the Message Structure field, program control is returned to the calling routine.

13.7 Get Transmission Option GETTXOPT (See also Fig 16)

The prompt for input is displayed on the VDU. The input obtained via GETIP is validated. If not S, C or M an error message is printed and the subroutine re-started. If the input is S for single, TXOPT is loaded with Ø and the message type checked. If the type is A or B then the number of outputs (MOPTNUM) is loaded with 1. If type is T then MOPTNUM is loaded with 2.

If the input is C (for continuous) then TXOPT is loaded with 1 and MOPTNUM is loaded with 1. Then the ON/OFF line variable ONOFFLI is interrogated. If ON line then Comfort Display information is requested (see Section 13.9) - control is returned to the calling routine after this information has been input - if OFF line control is returned to the calling routine. If the input is M for Multiple TXOPT is loaded with 2 and then the number of output transfers (in Decimal) must be input via INDECNU. This number if not zero is stored in MOPTNUM. If the number

is greater than 1000 then ONOFFLI is interrogated and, if ON line, comfort display information requested as above. If the number is less than 1000 or the display is OFF line then control is returned to the calling routine.

13.8 Get Wait Conditions GETWAIT (See also Fig 17)

The request for input is displayed on the VDU. The input is collected via the GETIP subroutine. If the input is neither Y or N then an error message is printed and the subroutine re-started. If the input is N for NO then WAITY is set to zero and program control is immediately returned to the calling routine.

If the input is Y for YES then WAITY is set to 1 and the Wait Register address is collected via the GETHEXIP subroutine. The input is validated ie it should be greater than or equal to zero and less than 8. If invalid an error message is printed and the Wait Register Address should be re-entered. If valid the wait register address is stored in WAITRA (see Figure 25) and program control returned to the calling routine.

13.9 Get Display Option GETDISOP (See also Fig 18)

The Display Option prompt is issued. The input is collected via the GETIP subroutine. If the input is not N, A or C then an error message is printed and the subroutine re-started.

If the input is N for None then DISOPT is set to zero and control returned to the calling routine.

If the input is A for All or C for Changes DISOPT is set to either 1 or 2 respectively. ON/OFF line selection is then required. If Off line display is selected ONOFFLI is set to zero and control returned to the calling routine. If ON line display is selected Segmentation entry is required (see Section 14.1) then DISOPT is again inspected and if ALL display has been selected control will return to the calling routine. If the user has selected to display Changes of Message TXOPT is inspected and if Single transmission has been selected control will immediately return to the calling program. If Multiple outputs have been selected the number of outputs (stored in MOPTNUM) must be greater than 1000 or the calling routine will be re-entered. At this point the user has the option of displaying 'comfort' messages every 1000th input. Selecting to display 'comfort' messages will cause COMFSTOR to be set to 1 (otherwise it will be cleared) and program control will then return to the calling routine.

14 DISPLAY ON REQUEST SUBROUTINES

14.1 Input Segmentation SEGMENT (See also Fig 19 and Para 17.2)

The prompt requesting Segmentation information is printed on the VDU. (For full description of Segmentation see Ref 2). The input, obtained via GETIP, is then checked. If the segmentation chosen is 1, 2 or 3 (corresponding to GTU, MATRIX Command and MATRIX Status respectively) then the correct pre-set segmentation values are copied into the segmentation working stack and control returned to the calling routine.

If the segmentation requested is 4, ie Input Dependent, a flag is set (which will be used in the Print Message Subroutine see para 14.3) and control is immediately returned to the calling program. If the Segmentation is not 4 the flag is cleared.

If the segmentation selected is Ø(SEGØ) ie User Defined then a second segmentation prompt is displayed on the VDU. The user should then decide whether or not to change the existing segmentation Ø specification. If not the existing specification is copied into the segmentation working stack and control returned to the calling program. If the user elects to change the previously defined segmentation Ø specification then segmentation specification input information is displayed on the VDU and user input enabled. (Up to 7 user defined segmentation spaces may be entered, input should be terminated by entering the letter 'E').

As each input is entered it is validated to ensure that it is either the letter 'E' or a decimal number (parts of INDECNU, see section 11.3, are used to validate the decimal number). If the input is a valid decimal number it is stored in the User Segmentation stack and the next input requested. If the input is invalide the whole Segmentation Ø section is restarted, this also happens if the user tries too many segmentation spaces. If the input is the termination character 'E' the segmentation specification built up in that part of memory allocated to user defined segmentation is copied in the segmentation working stack and program control returned to the calling routine.

14.2 Print Block PRINBLK (See also Fig 2Ø and Para 17.2)

The Print Block subroutine calculates the start address of each message (the datum being the start address of the Block) and calls the Print Message (PRINMES) subroutine (with the message start as the parameter) to print the message. It then calculates the start address of the next message and continues printing messages until every message in the block has been printed. Control is then returned to the calling routine.

14.3 Print Message PRINMES (See also Fig 21).

The automatic segmentation flag is checked and if set the Message Type (MSTYPE) is checked, if the Type is T (for both) then the input message number is checked. An odd input number will correspond to a response from an A message, similarly an even input number to a B message. The Format field of the structure (See Fig 25)of the correct message is then interrogated. If the format field is set to Ø then segmentation 1 is used, otherwise the Register Address parameters are extracted from the input message. If the input Register Address is equal to 2 (ie a MATRIX Status input) then segmentation 2 is selected otherwise segmentation 3 is selected. The selected segmentation specification is then copied into the segmentation working stack. At this point either the automatic segmentation flag was clear, and the segmentation Ø, 1, 2 or 3 identified or the automatic segmentation has been determined as above.

The message start address parameter passed by the calling routine is then collected. The first two bytes contain the message number. This is printed on the VDU in decimal format using OUTDEC.

The remaining 6 bytes of the message contain the 42 bits of the input message. Each bit is collected in turn and stored in ASCII format (ASCII Ø being used for a bit equal to Ø and ASCII being used for bit equal to 1) in a vertical stack ASMSST. As each bit is stored the bit number is checked against the segmentation specification and if the bit number corresponds to a segmentation space an ASCII space character is inserted into ASMSST before the actual bit.

When all 42 bits of the message have been processed ASMSST contains a series of ASCII Øs, 1s and spaces. This whole stack is printed on the VDU using the OUTSAC subroutine. Control is then returned to the calling routine.

15 THE HARDWARE INTERFACES

15.1 Programming the PPIs

The Hardware Handlers control the input and output peripheral circuits via four Programmable Peripheral Interface devices (PPIs). The input/output functions to these devices are programmed initially in the Initialisation routine which is entered after the system is reset.

Tables 1 to 4 list the PPI addresses and their internal identifiers and Figure 22 illustrates the nature (ie input or output) of their individual programming and details the function of the bits of each of the PPI ports.

HEX ADDRESS	PORT	FUNCTION	IDENTIFIER
8Ø8Ø	A	OUTPUT	POPDAT
8Ø81	B	OOUTPUT	POPADD
8Ø83	CONTROL	CONTROL	POPCON1

CONTROL BYTE HEX 89

Table 1 PPI 1 Notations

Loading the Control byte (Hex 89) into the control port POPCON1 programs Ports A and B to be outputs.

HEX ADDRESS	PORT	FUNCTION	IDENTIFIER
8Ø84	A	OUTPUT	POPCOM
8Ø85	B	INPUT	POPSTA
8Ø87	CONTROL	CONTROL	POPCON2

CONTROL BYTE HEX 8B

Table 2 PPI 2 Notations

Loading the control byte (Hex 8B) into the control port POPCON2 programs Port A to be an output and Port B to be an input.

PPIs 1 and 2 are used to interface to the Axis Test Box peripheral card controlling OUTPUT TRANSFERS.

HEX ADDRESS	PORT	FUNCTION	IDENTIFIER
8Ø88	A	INPUT	PIPDAT
8Ø89	B	OUTPUT	PIPADD
8Ø8B	CONTROL	CONTROL	PIPCON1

CONTROL BYTE HEX 99

Table 3 PPI 3 Notations

Loading the control byte (Hex 99) into the control port PIPCON1 programs Port A to be an input and Port B to be an output.

HEX ADDRESS	PORT	FUNCTION	IDENTIFIER
8Ø8C	A	OUTPUT	PIPCOM
8Ø8D	B	INPUT	PIPSTA
8Ø8F	CONTROL	CONTROL	PIPCON2

CONTROL BYTE HEX 8B

Table 4 PPI 4 Notations

Loading the control byte (Hex 8B) into the control port PIPCON 2 programs Port A to be an output and Port B to be an input.

PPIs 3 and 4 are used to interface to the peripheral card controlling INPUT TRANSFERS.

15.2 Hardware/Software Boundary

When considering the boundary between the hardware and software in the AXIS Test Box it is important to consider the functions to be performed by the hardware. In the most basic terms these functions are:

a To accept data from the microprocessor.

b To transmit this data in the form of an output transfer to the system under test.

c To receive input transfers from the system under test.

d To present the received data to the microprocessor for investigation.

Since the AXIS Test Box is used in place of the System 25Ø (see Ref 1) it is necessary that the interface btween the Test Box and the system under test should be identical to that which the S25Ø would present. Of the 4 functions listed above items c) and d) must, therefore, be performed in real time, ie using the same transmissin rate as the S25Ø. Reference 5 gives full details of the interface requirements.

The time interval between transfers is not defined therefore the processing necessary to perform functions a) and d) above may be preformed in non-real time, ie at a rate determined only by the instruction speed of the M68Ø9 microprocessor controlling the Test Box.

15.3 Outputting Data to the Hardware Registers

Once data has been prepared for transmission, as described in Sections 9 and 1Ø, and is in the format shown in Figure 26B it must be loaded into the hardware registers. There are two hardware registers A and B. Each is 42 bits long and each is made up of five shift registers of 8 bits and one shift register of 2 bits. Each of the individual shift registers has been allocated a specific address (see Annex D). When a hardware register address is output via POPADD (Fig 22) the hardware decodes the address and generates a parallel load pulse for the selected shift register. Each byte of the data is output to the hardware via POPDAT (see Fig 22). The hardware register address is then output via POPADD this then causes a hardware parallel load pulse to be generated for the correct hardware register and the POPDAT data is latched into the hardware register (see para 1Ø.4).

When all the hardware data registers have been loaded the number of transmissions is loaded into the hardware counter in a similar manner.

Bit A_2 in POPCOM is set to Transmit, (ie logic Ø) and bit A_1 set to logic 1 if the Continuous option was selected. Bit A_3 is cleared if not in Auto Receive Mode, and finally a GO pulse sent on bit AØ. Once the GO pulse has been sent the hardward assumes control and the data is transmitted in real time to the system under test.

15.4 Reading Data from the Hardware Registers

The hardware input buffer is 42 bits in length, and is made up of five shift registers of 8 bits and one of 2 bits. Data is accessed via six tri-state buffers. (one for each shift register of the input buffer). Annex E lists the addresses of these tri-state buffers. When an input transfer has been received in real time from the system under test the polling routines described in para 16.1.2 inspect the input board status inputs (PIPSTA in Fig 22). If the Ready to Read bit (bit BØ) is set then the address of each byte of the hardware input buffer is loaded into PIPADD and that data byte read into the microprocessor via PIPDAT. This process is repeated until each byte of the hardware input buffer has been read into micro-processor store. The interrogation routines (see para 16.2) then assume control. The transfer of data from the hardware to the micro-processor is not carried out in real time.

16 HARDWARE HANDLERS

The Hardware Handling code can be divided into two sections:

a Handling hardware go/stop, and polling for hardware or VDU responses. These are the POLLING routines.

b Interrogation of inputs, storing, displaying etc. These are the INTERROGATION routines.

16.1 Polling Routines (See also Figs 22 and 23)

16.1.1 Starting Transfers

For an output transfer the hardware counter is loaded with the numbers of transfers to be sent. RX/TX is set to TX and a logic pulse generated on the GO line commands the output board to proceed with the transfer.

For an input, transfer circuitry on the output transfer board generates the timing necessary for inputs to be received. RX/TX is set to RX and a GO pulse generated.

16.1.2 Polling for Responses

Responses may arise from any of the following three areas:

a User response from VDU ie Stop command.

b Output transfers.

c Input requests, received etc.

Responses from a.above obviously come via the VDU interface, the Stop command has been set to be a specific sequence of the letter S followed by carriage return. Upon receipt of the letter S half the stop command has been received and the HALFSTOP flag set. A user input of carriage return will then cause a FULL-STOP, one more output and input sequence will then be accepted in order to synchronise the transmissions. An informtion message informing of user intervention will be printed on the VDU and control returned to the Basic Command Mode. (In the case of Automatic Input Display the next input will synchronise the termination).

If the character entered after the letter S is not a Carriage Return the HALFSTOP flag will be cleared and the whole Stop sequence must be re-entered before it will be accepted. This is done in order that the act of Stopping transfers should be a deliberate action and the recognition erroneous Stop commands minimised. Termination may also be caused by filling the input message store.

Once an output transfer has started it will continue until completed. Whenever the number of output transfers programmed into the hardware counter has been sent the Transmitting/Stopped bit in POPSTA will be raised to indicate completion of output sequence. When this has happened one more input will be accepted in order to synchronise with the system under test. If the Transmit Option selected is Continuous then the Continuous bit (see Figure 22) will be set thus disabling the hardware counters. In this case the user must intervene to stop outputs, or the message store must be filled.

When an output has been activated inputs should be expected and the polling routine will interrogate the status responses from the Peripheral Input Transfer card. These status responses (see Figure 22) are:

a Ready to read - this indicates that data has been received and is in the input buffer shift register awaiting interrogation. (See also para 15.4)

b Request i/p Transfer (TFR) - this indicates that the system under test has requested an input transfer. This request is granted by putting the RX/TX bit to RX and generating a GO pulse.

c End i/p transfer - this indicates the end of the input transfer. If this bit is set and the Ready to Read bit is clear it indicates that the system under test has failed to return the standard 42 bits of data. (The Matrix returns only 36 bits of data). A circuit on the Input peripheral card which generates 6 extra pulses, for the input buffer, is then activated. This is done so that standard interrogation routines described in section 16.2 may be used for both GTU and Matrix messages.

d Input timeout - this indicates that an output has been sent but the system under test has failed to respond within a specified period. After each Output transfer (or, in the case of Auto Display Mode, Input transfer) the timeout counter is reset. In order to avoid a possible 'Lock up' situation arising, if the system under test fails to respond, a timeout on Inputs is enabled. When the timeout status bit is set the peripheral hardware cards are re-set to Transmit mode and another Output sent.

16.2 Interrogation Routines (See also Fig 24)

The Interrogation Routines are entered when the Ready to Read Status bit is set. If the Display Option chosen is N for None and no Wait Conditions have been specified (ie the hardware delay is enabled) HALFSTOP is interrogated to check if this message should terminate the transfer sequence. If termination is indicated (ie HALFSTOP = 2) then the termination message is printed and control returned to the Basic Command Mode. If termination is not indicated (ie HALFSTOP $\neq$ 2) the hardward is re-set to Transmit, and the Polling Routines re-entered.

If the above conditions are not met the data in the input buffer is read and the message together with its message number is stored in a temporary storage area THISIP.

If a Wait Condition has been set the input Message Register Address is extracted and compared with the Wait Register Address. If a match is not achieved the termination check described is carried out and if no termination is indicated the hardware is re-set to Transmit and the Polling Routines re-entered.

If the Wait Register Address match is found, or no Wait conditions have been set, the Display Option chosen is inspected.

If All inputs are to be stored for display then the input message with its internally generated message number is copied from THISIP to the next message space available in the input message store. If ON Line display has been selected the message will then be displayed on the VDU using the Print Message subroutine.

The total number of input messages will then be checked and if the input message store is now full the input/output sequence will be terminated, otherwise the hardware will be re-set to Transmit and the polling routines re-entered.

If Changes of message are to be stored for display then this input message must be checked against the last message of the same type. If the Type is A then against the last A message preserved in LASTAIP. If the Type is B then the check is made against the last B message to be received and preserved in LASTBIP. If the type is T (for Both) then the input message number is inspected. If the input message number is ODD then the check is made against LASTAIP and if EVEN the check is made against LASTBIP.

If differences were found then the message in THISIP together with its message number is copied into the next space in the input message store and the relevant LAST A or B IP is overwritten with the new data only (excluding message number). If ON line display has been selected the message and message number will be printed on the VDU using the PRINMES subroutine; the termination checks will then be carried out and if no termination is indicated the hardware is re-set to Transmit and the Polling Routines re-entered.

If no differences were found then the 'comfort' message flag COMFSTOR is checked and if set the message number inspected. If this is the 1ØØØth message received since the last display it will be displayed on the VDU using the PRINMES subroutine.

The termination checks are then carried out and in the absence of termination indication the hardware is again re-set to Transmit and the Polling Routines re-entered.

17 THE DATA STRUCTURES

17.1 Message Assembly

When the Message Assembly has been completed the INTERNAL MESSAGE STRUCTURE (Figure 25) will be filled with the correct information.

Information is stored in this structure in a form which makes changing individual parameters relatively simple since each parameter is stored independently. This structure is not, however, the ideal for operations asuch as parity calculations and therefore, when it becomes necessary to calculate parity bits, the message information stored in the A or B Message Fields is transformed by the FMFORP (Format for Parity) subroutine into the structure shown in Figure 26A. In this structure each bit of the 42 bit message occupies a whole byte with ØØ representing a bit Ø and Ø1 representing a bit 1. With the information stored in this format

the CALCP subroutine may be called to generate parity over any data area within the 42 bit message and store the parity bit in whichever bit position is selected.

Once the parity bit(s) has been generated the information stored in the structure in Figure 26A is transformed by the TFTOTX subroutine into a format ideal for interfacing to the hardware by the hardware handlers. This format is illustrated in Figure 26B.

This is done in order that the user may select to transmit A, B or both when using the Repeat Command, and may also change individual elements of each message independently when using the Repeat Command.

17.2 Message Display

When input messages are received they are stored (in accordance with the display option selected) in the form illustrated in Figure 27A. Each input message is given a message number which is stored with the message.

Messages are stored in blocks of 16 with space available for up to 4Ø blocks. When the Display on request is selected the total number of messages stored is displayed and the user allowed to select the segmentation format required. This segmentation allows fields within the 42 bit message to be isolated for easier inspection. Three pre-set segmentations are available plus one which is user programmable. Figure 28 illustrates the segmentation storage area. The example shown in Figure 28 for segment 1 means that a space will be inserted in the bit pattern displayed after the 4th, 7th and 9th and 41st bits. This could isolate for example the check code, register address, data and parity fields of each message displayed.

In order to prepare each message for display and to include the spaces for segmentation the message is transformed from the format illustrated in Figure 27A to that illustrated in Figure 27B. In Figure 27B each bit of the 42 bit message occupies a byte, a bit Ø being represented by Hex 3Ø (ie ASCII Ø) and a bit 1 being represented by HEX 31 (ie ASCII 1). The segmentation spaces (Hex 2Ø) are inserted in the correct bit position thus when the message is output the message will appear as a series of Øs and 1s with spaces providing the segmentation isolation.

18 TRANSMIT DATA FLOW (See also Fig 29)

All input information is collected via the Data Entry Subroutines described in Section 13. At this point the information is stored in the internal message structure Fig 25. The Format for Parity subroutine (para 1Ø.1) translates the Data, Check Code and Register Address into a format which makes parity calculation easier (Fig 26A). When the parity bit(s) have been inserted the information is then transformed (see para 1Ø.3) into a format suitable for transmission (Fig 26B). This information is then loaded into the hardware registers as described in para 15.3.

19 RECEIVE DATA FLOW (See also Fig 3Ø)

The hardware input buffer register is read (see para 15.4) and the information stored in the format shown in Fig 27A. If the input transfer is to be displayed on the VDU, using the Print Message subroutine (see para 14.3) it is firstly transformed into the format illustrated in Fig 27B. Segmentation spaces are inserted and each data bit translated into its ASCII representation ready for output to the VDU. Reference 2 describes and illustrates quite clearly the type of display to be expected using different segmentation specifications.

INTERNAL IDENTIFIER CROSS REFERENCE

IDENTIFIER	INTERNAL LABEL	EXTERNAL VALUE	INTERNAL VALUE
MESSAGE TYPE	MSTYPE	A - A MESSAGE B - B MESSAGE T - BOTH	Ø 1 2
A PARITY	APARTRU	TRUE Y - YES N - NO	Ø 1
B PARITY	BPARTRU	TRUE Y - YES N - NO	Ø 1
A FORMAT	AFORMAT	G - GTU M1 - MATRIX COMMAND M2 - MATRIX CONNECTION	Ø 1 2
B FORMAT	BFORMAT	G - GTU M1 - MATRIX COMMAND M2 - MATRIX CONNECTION	Ø 1 2
TRANSMISSION OPTION	TXOPT	S - SINGLE C - CONTINUOUS M - MULTIPLE	Ø 1 2
WAIT CONDITION	WAITY	Y - YES N - NO	Ø 1
DISPLAY OPTION	DISOPT	N - NONE A - ALL C - CHANGES	Ø 1 2
ON OR OFF LINE DISPLAY	ONOFFLI	ON LINE Y - YES N - NO	Ø 1
RECEIVE OR TRANSMIT	RXTX	TRANSMIT RECEIVE	Ø 1
COMFORT MESSAGES	COMFSTOR	COMFORT MESSAGES Y - YES N - NO	 1 Ø

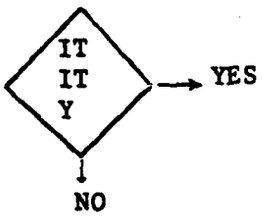
ANNEX A

ANNEX B

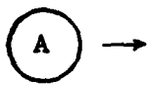
DRAWING SYMBOLS USED

Activity ie Code performs this task

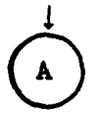
Subroutine



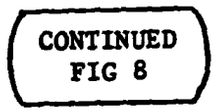
Decision Box



GOTO eg GOTO A



Entry point ie GOTO A would arrive at this point



Continuation ie Flow Diagram continuing at Fig 8.

Data Structure

ANNEX C

CROSS REFERENCE BY INTERNAL LABEL NAME

LABEL	CODE FUNCTION	PARAGRAPH	FIGURE NUMBER
ASMMES	Message Assembly Routine	4	2
AUTODIS	Auto Display Routine	6	4
CALCP	Calculate Parity Subroutine	10.2	-
COMMODE	Basic Command Mode Routine	3	1
DISOREQ	Display on Request Routine	5	3
ENDOFSUB	Prepare for Hardware Handlers Routine	9.1	8
FMFDRP	Format for Parity Generation Subroutine	10.1	-
GETDAT	Input Data Subroutine	13.4	13
GETDISOP	Get Display Options Subroutine	13.9	18
GETHEXIP	Get Hexadecimal Input Subroutine	11.2	9
GETIP	Get Input Character Subroutine	11.1	9
GETPART	Get Parity Subroutine	13.5	14
GETTXOPT	Input Transmission Options Subroutine	13.7	16
GETTYPE	Input Message Type Subroutine	13.1	10
GETWAIT	Input Wait Conditions Subroutine	13.8	17
GOANDTX	Prepare Data for Transmission Routine	9	7
INASC	Get ASCII Character from Keyboard Subroutine	11.4	9
INDECNU	Input Decimal Number Subroutine	11.3	9
INITIAL	Initialisation Routine	2	-
INPCC	Input Check Code Subroutine	13.2	11
INPRA	Input Register Address Subroutine	13.3	12

LABEL	CODE FUNCTION	PARAGRAPH	FIGURE NUMBER
MENUDIS	Display Current Menu Subroutine	8	6
MSSTR	Input Message Format/Structure Subroutine	13.6	15
OUTASC	Output ASCII character to keyboard Subroutine	12.1	9
OUTDEC	Output Decimal Subroutine	12.3	9
OUTHEX	Output Hexadecimal Subroutine	12.2	9
POLLING	Polling Routines	16.1.2	23
PRINBLK	Print Block Subroutine	14.2	2Ø
PRINMES	Print Message Subroutine	14.3	21
REAIPDAT	Interrogation Routines	16.2	24
REPEATX	Repeat Routine	7	5
SEGMENT	Input Segmentation Subroutine	14.1	19
TFTOTX	Transform to Transmit Format Subroutine	1Ø.3	-

ANNEX D

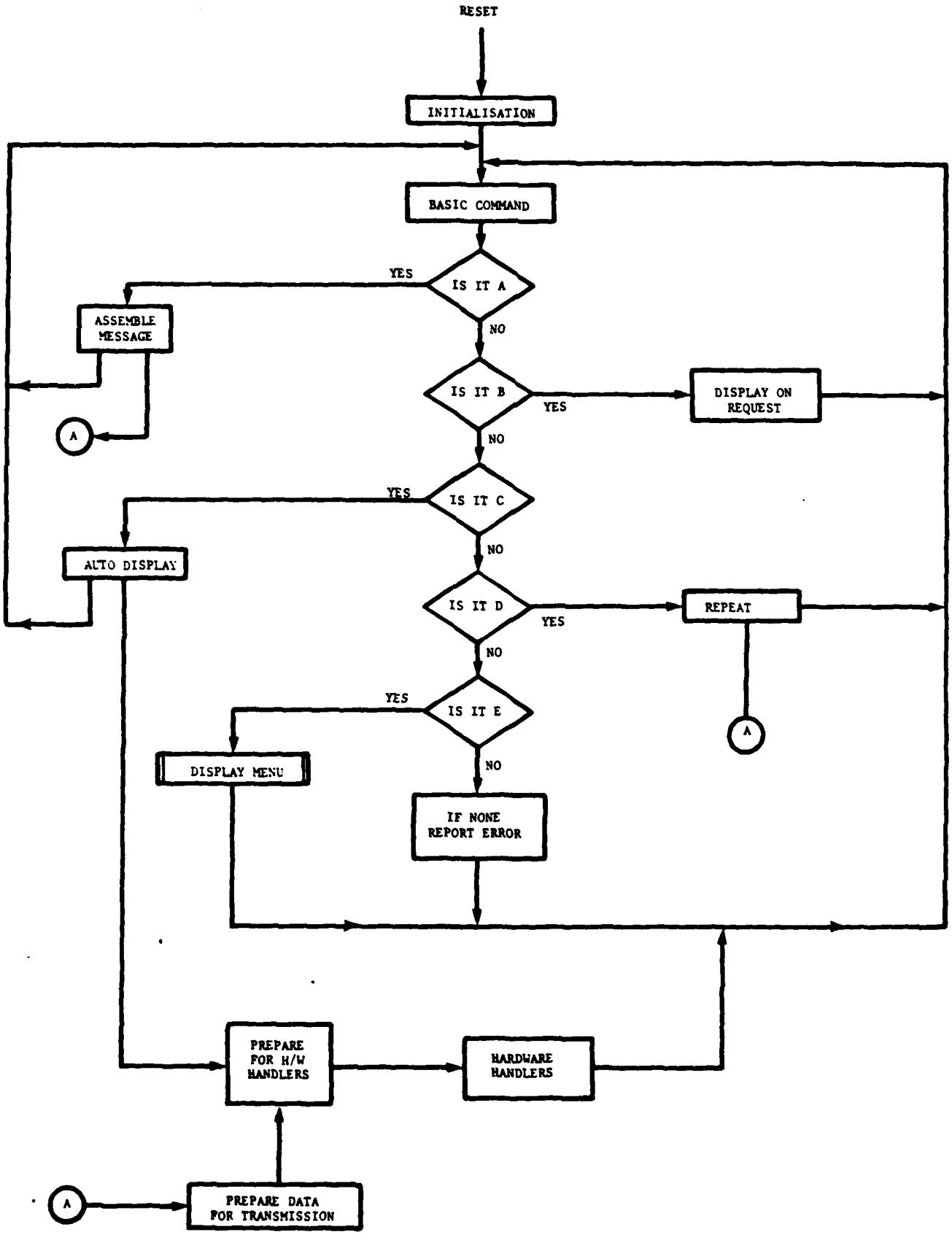
OUTPUT BOARD HARDWARE REGISTER ADDRESSES

ADDRESS (IN HEXADECIMAL)	REGISTER SELECTED
Ø	NONE - THIS IS THE QUIESCENT STATE
1	Register A Shift Register 1
2	Register A Shift Register 2
3	Register A Shift Register 3
4	Register A Shift Register 4
5	Register A Shift Register 5
6	Register A Shift Register 6
7	Register B Shift Register 1
8	Register B Shift Register 2
9	Register B Shift Register 3
A	Register B Shift Register 4
B	Register B Shift Register 5
C	Register B Shift Register 6
D	Hardware Counter Shift Register 1
E	Hardware Counter Shift Register 2
F	UNUSED

ANNEX E

INPUT BOARD HARDWARE REGISTER ADDRESSES

ADDRESS (IN HEXADECIMAL)	REGISTER SELECTED
Ø	NONE - THIS IS THE QUIESCENT STATE
1	Input Buffer Register 1
2	Input Buffer Register 2
3	Input Buffer Register 3
4	Input Buffer Register 4
5	Input Buffer Register 5
6	Input Buffer Register 6
7 → F	*UNUSED*


RESET
INITIALISATION
BASIC COMMAND
IS IT A
YES
NO
ASSEMBLE MESSAGE
A
IS IT B
YES
DISPLAY ON REQUEST
NO
IS IT C
YES
NO
ALTO DISPLAY
IS IT D
YES
REPEAT
NO
IS IT E
YES
NO
DISPLAY MENU
IF NONE REPORT ERROR
PREPARE FOR H/W HANDLERS
HARDWARE HANDLERS
A
PREPARE DATA FOR TRANSMISSION

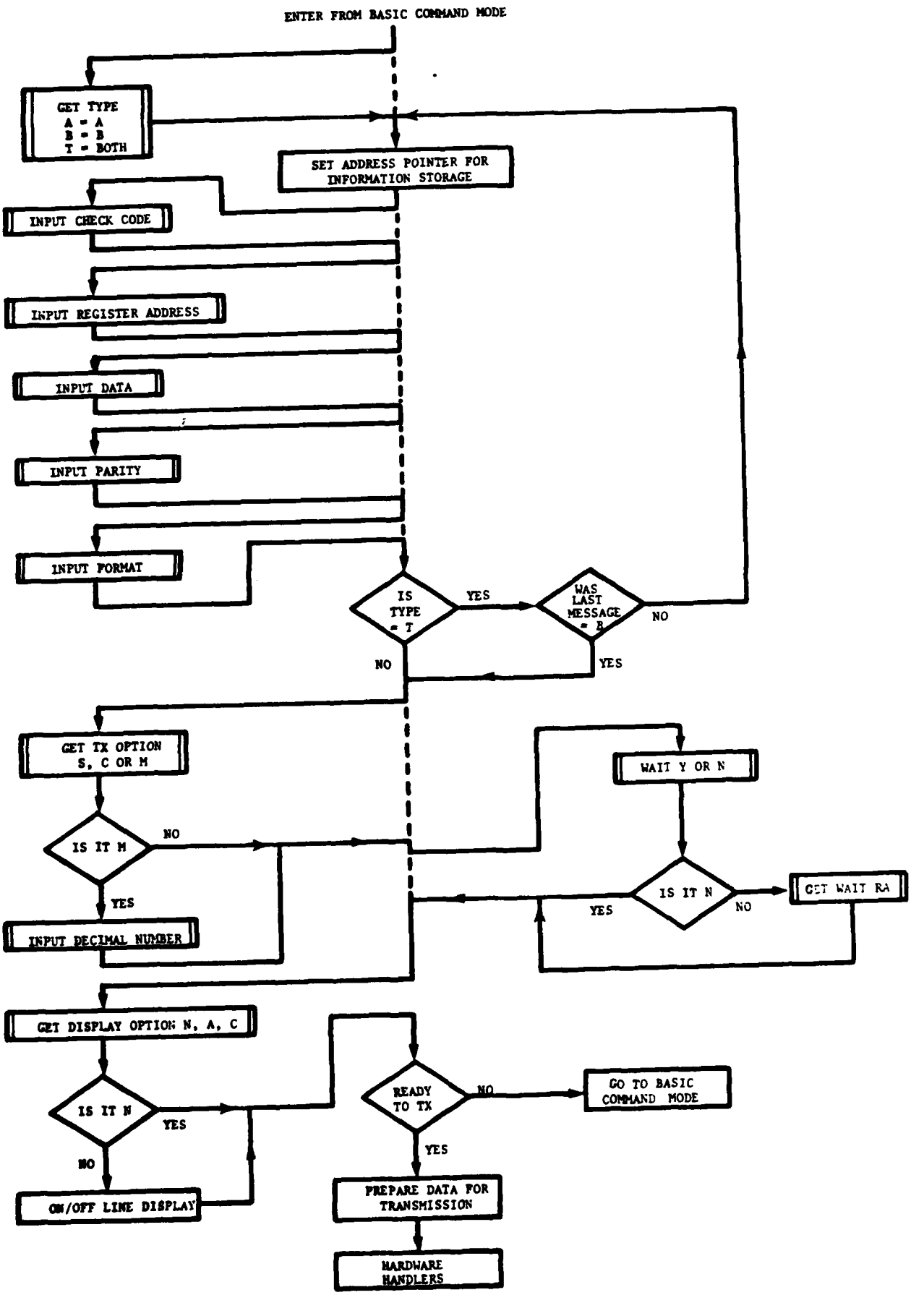


FIG 2 ASSEMBLE MESSAGE ROUTINE FLOW DIAGRAM

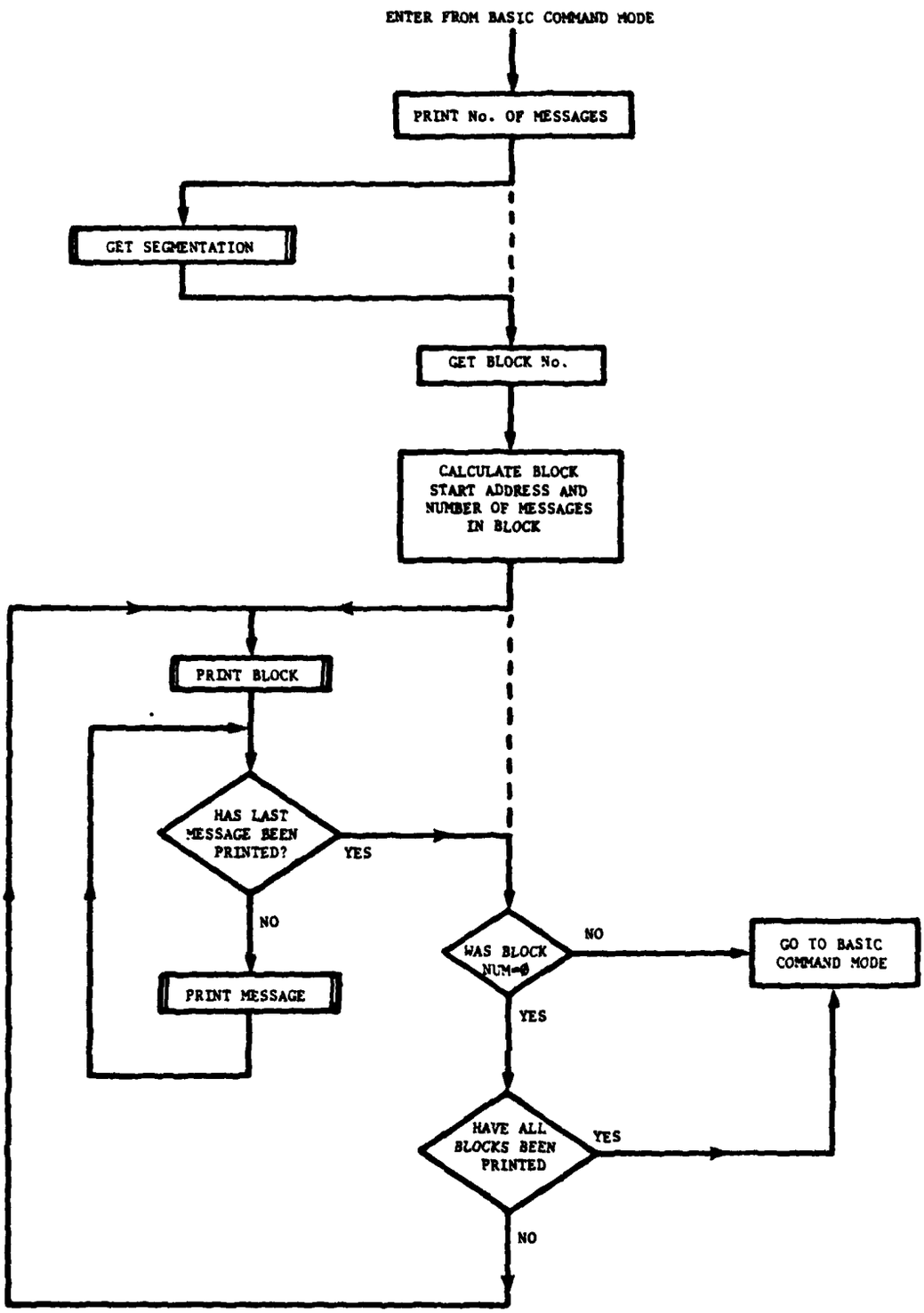


FIG 3 DISPLAY ON REQUEST ROUTINE FLOW DIAGRAM

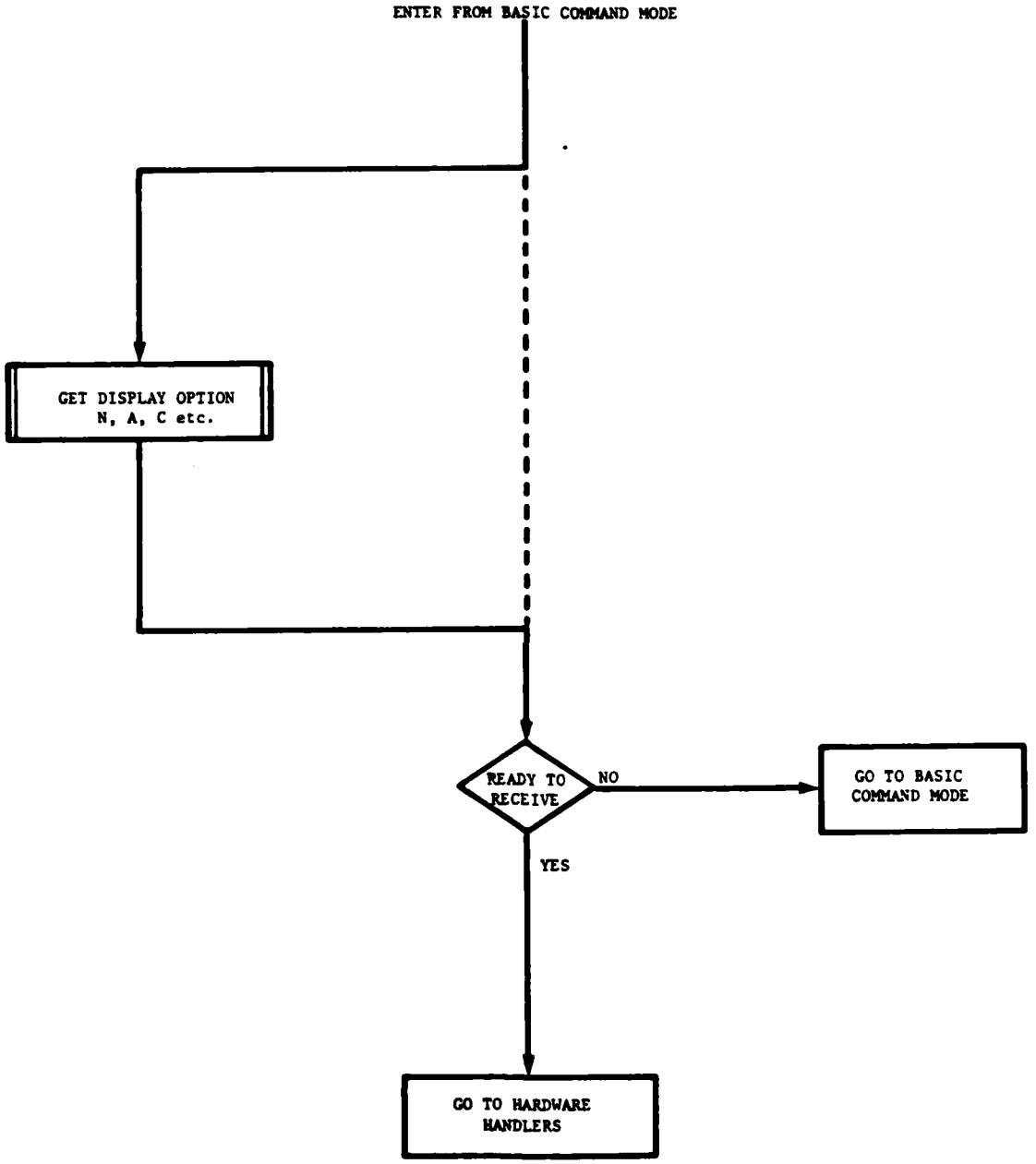
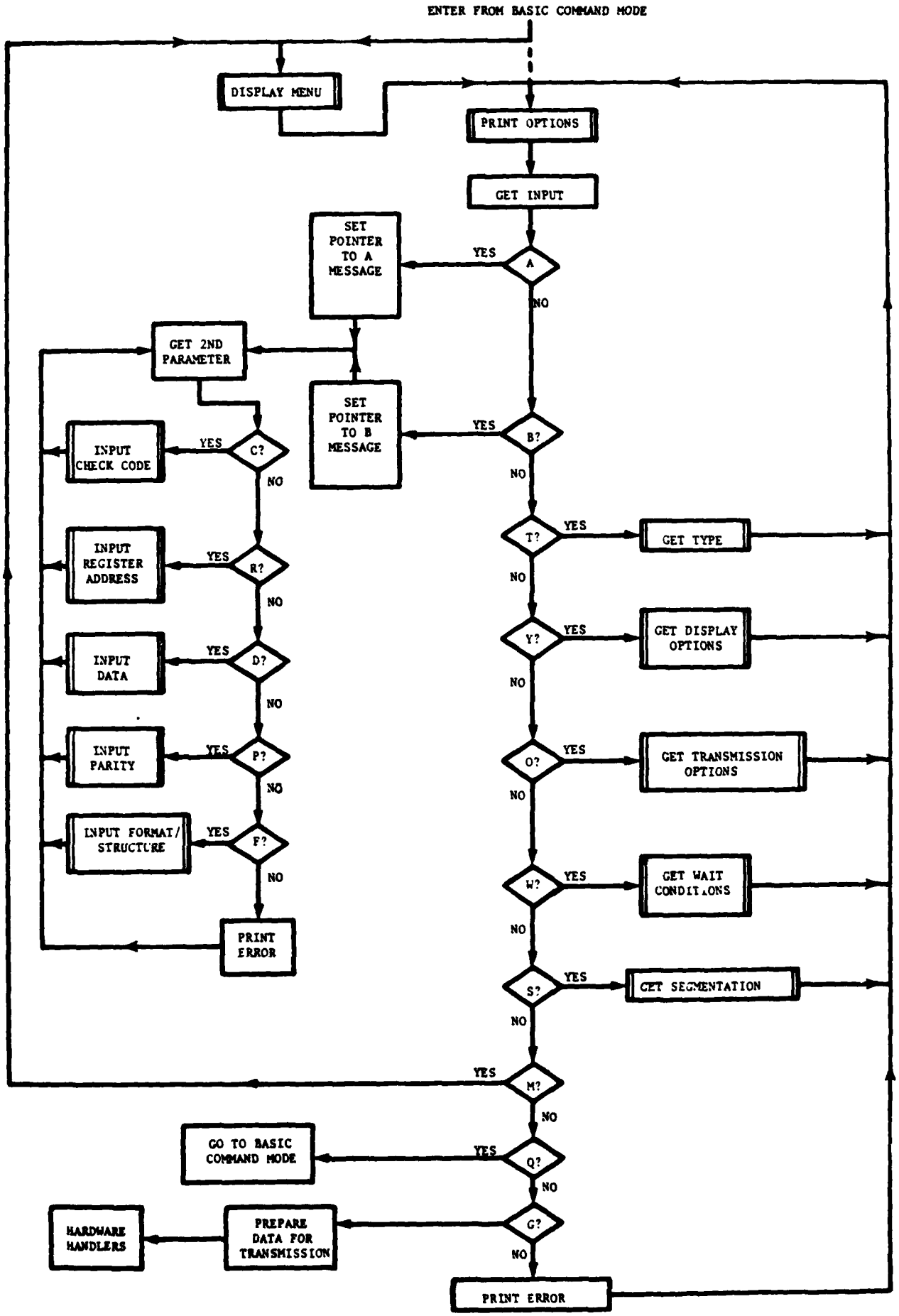


FIG 4 AUTO DISPLAY ROUTINE FLOW DIAGRAM


ENTER FROM BASIC COMMAND MODE
DISPLAY MENU
PRINT OPTIONS
GET INPUT
A
YES
SET POINTER TO A MESSAGE
NO
GET 2ND PARAMETER
B?
YES
SET POINTER TO B MESSAGE
NO
C?
YES
INPUT CHECK CODE
NO
R?
YES
INPUT REGISTER ADDRESS
NO
D?
YES
INPUT DATA
NO
P?
YES
INPUT PARITY
NO
F?
YES
INPUT FORMAT/ STRUCTURE
NO
PRINT ERROR
T?
YES
GET TYPE
NO
Y?
YES
GET DISPLAY OPTIONS
NO
O?
YES
GET TRANSMISSION OPTIONS
NO
W?
YES
GET WAIT CONDITIONS
NO
S?
YES
GET SEGMENTATION
NO
M?
YES
NO
Q?
YES
GO TO BASIC COMMAND MODE
NO
G?
PREPARE DATA FOR TRANSMISSION
HARDWARE HANDLERS
NO
PRINT ERROR

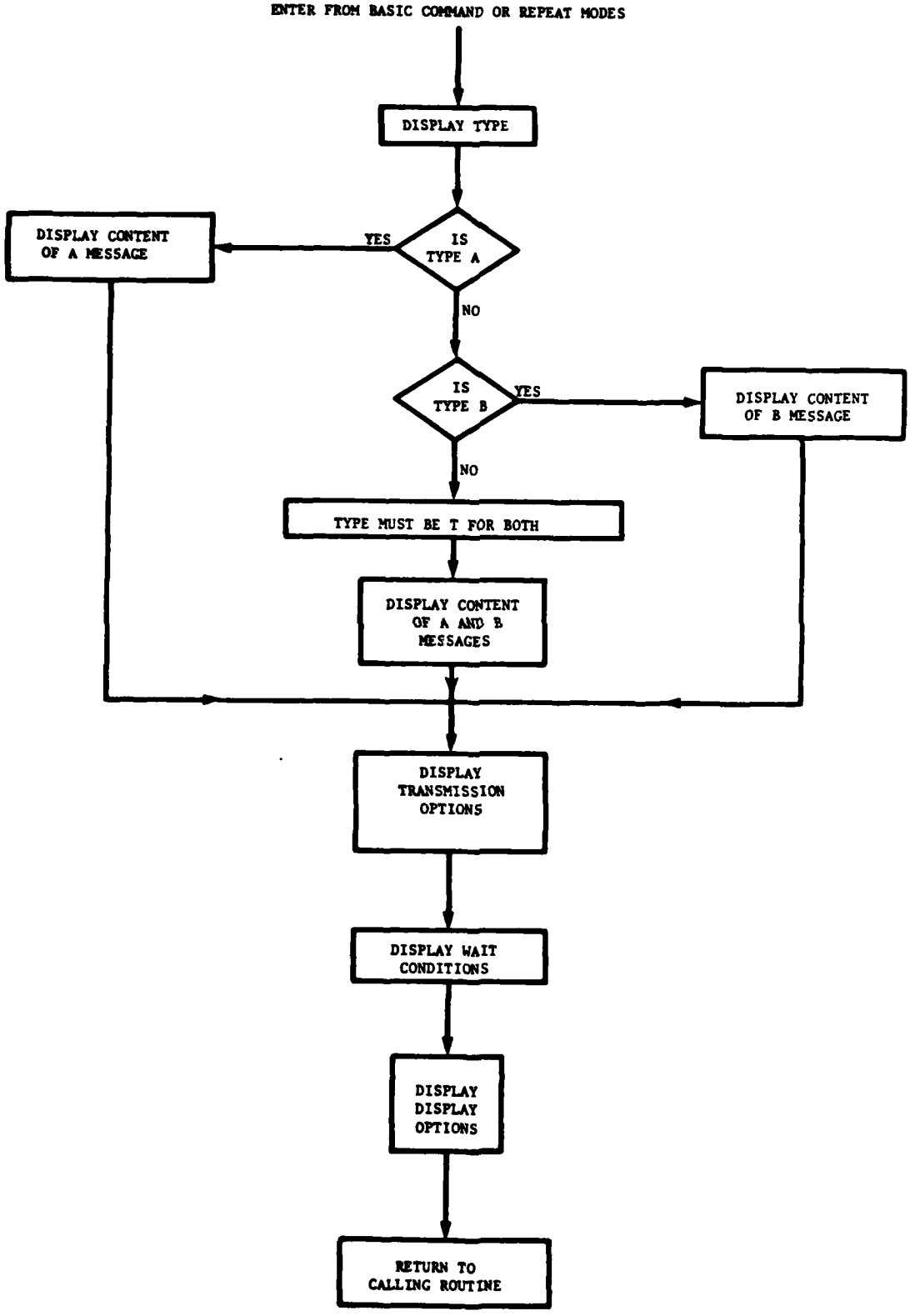


FIG 6 MENU DISPLAY ROUTINE FLOW DIAGRAM

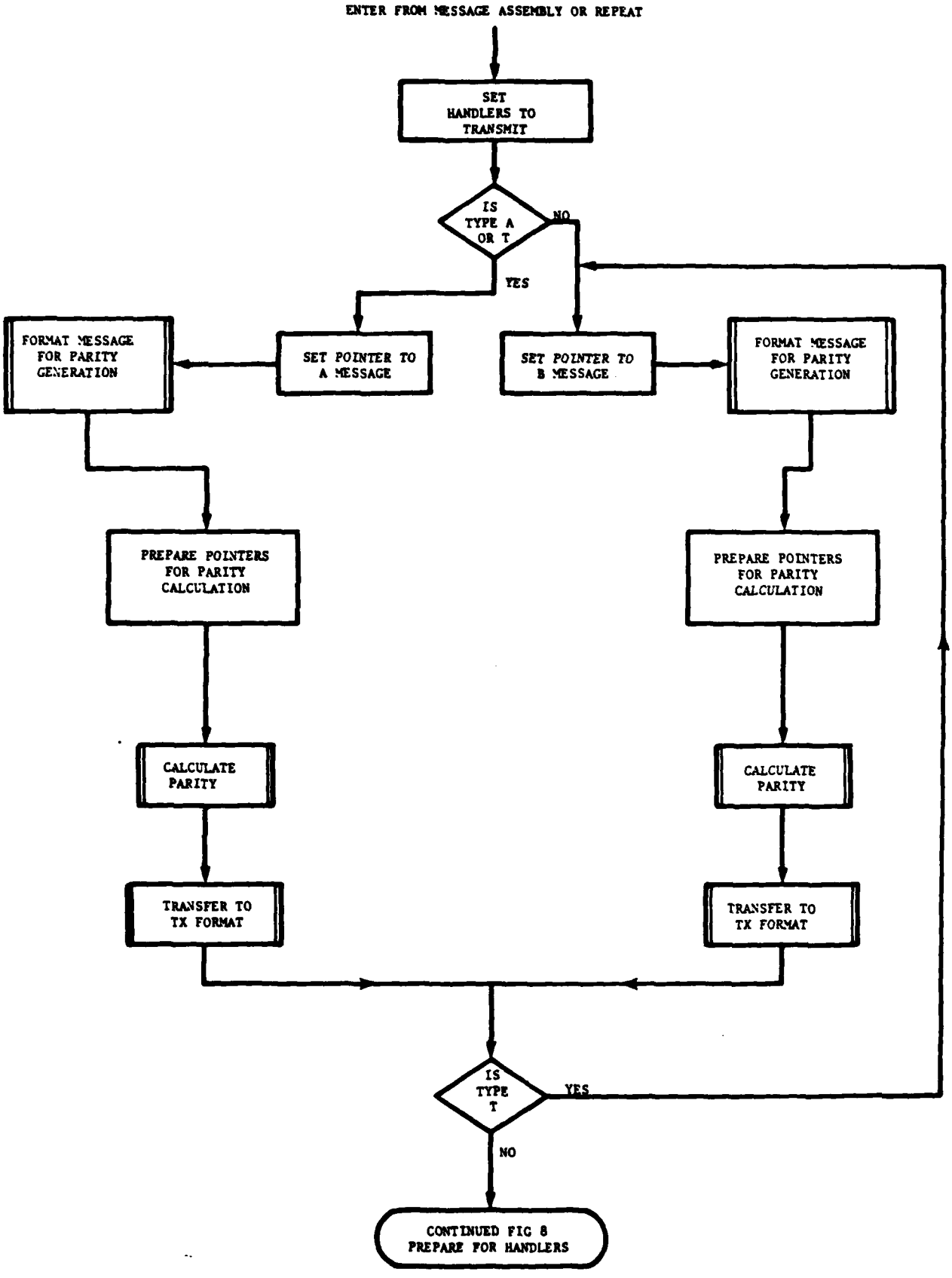


FIG 7 PREPARE DATA FOR TRANSMISSION FLOW DIAGRAM

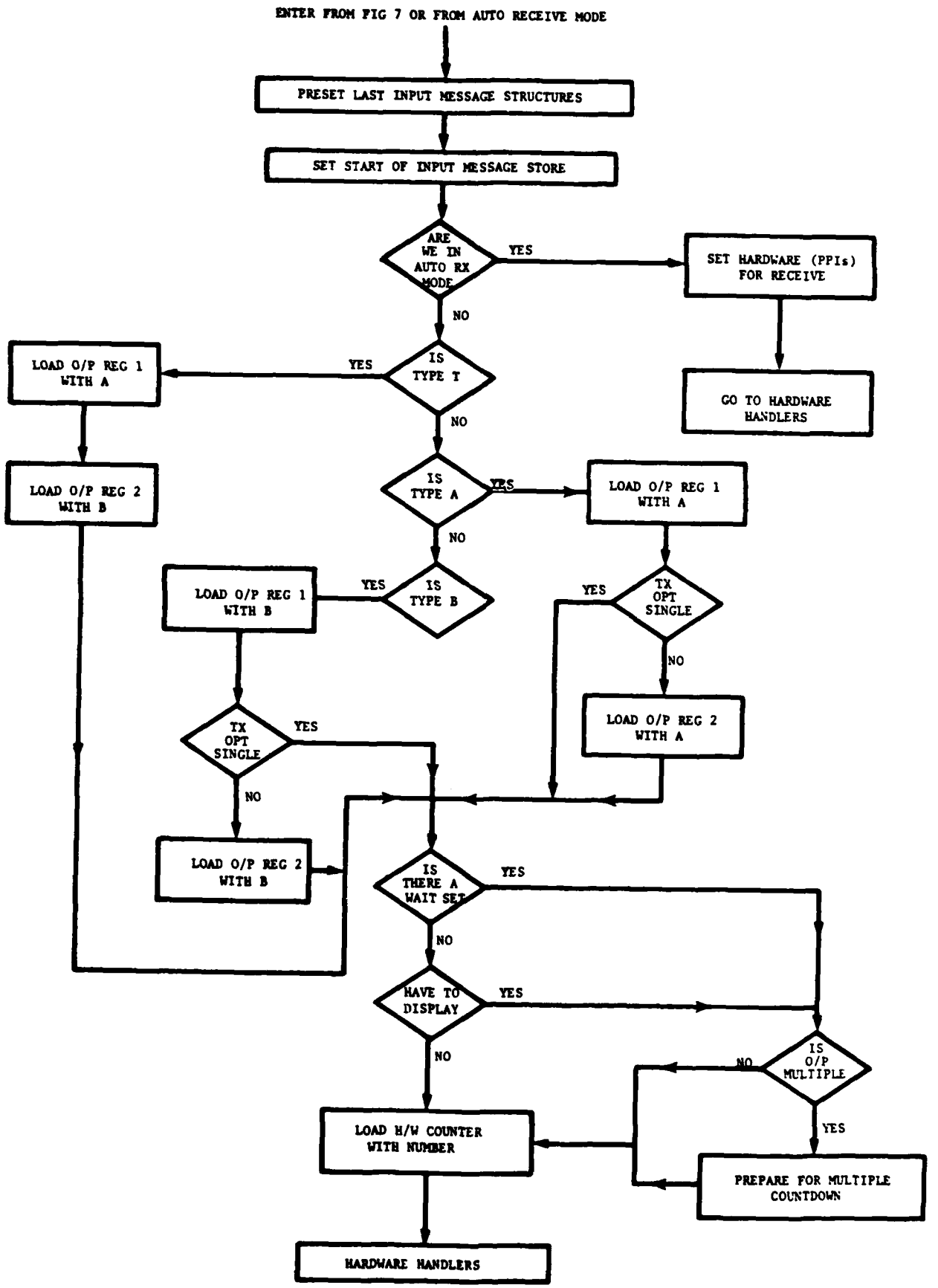


FIG 8 PREPARE FOR HARDWARE HANDLERS FLOW DIAGRAM

INPUT

11.1 GETIP

11.2 GETHEXIP

11.3 INDECNU

11.4 INASC

OUTPUT

12.1 OUTASC

12.2 OUTHEX

12.3 OUTDEC

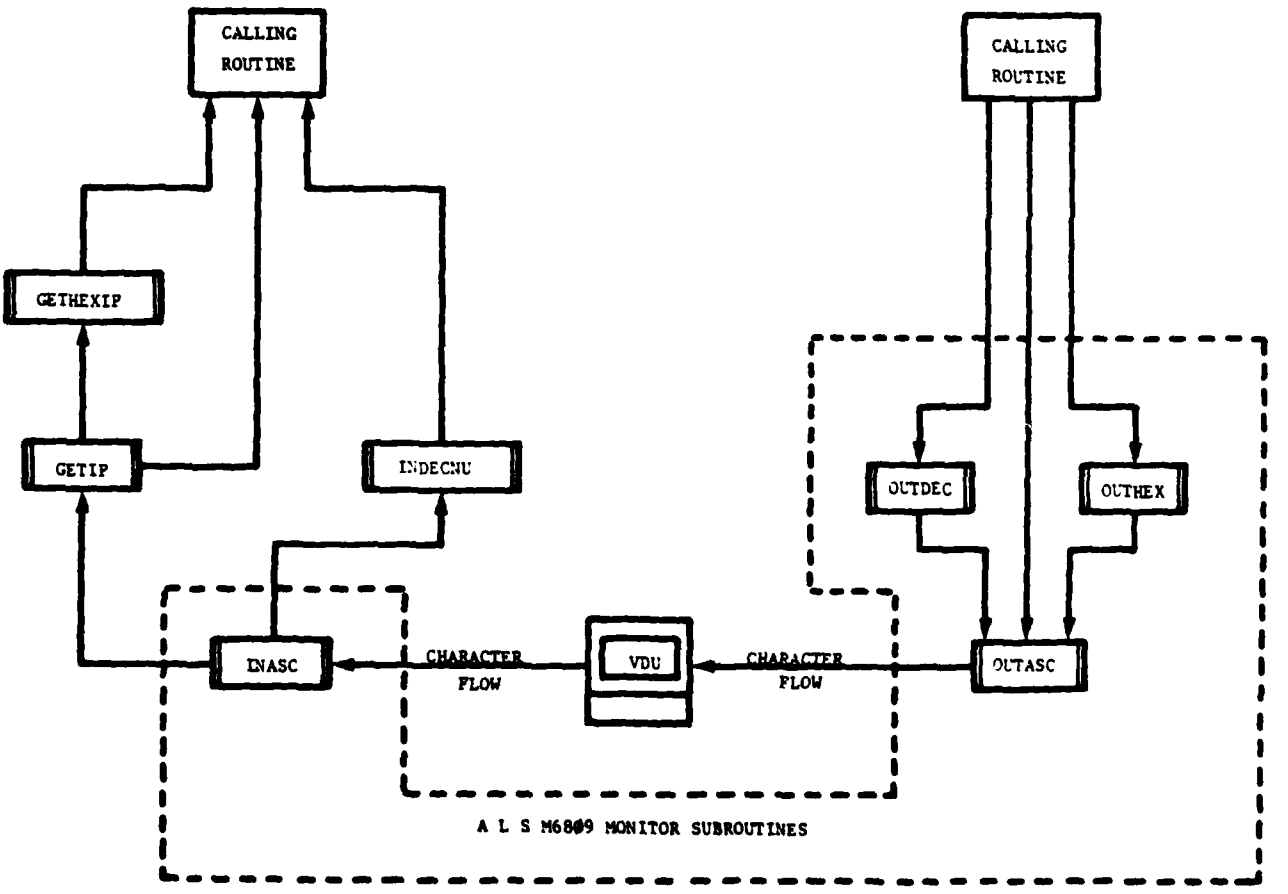


FIG 9 BASIC INPUT/OUTPUT SUBROUTINES

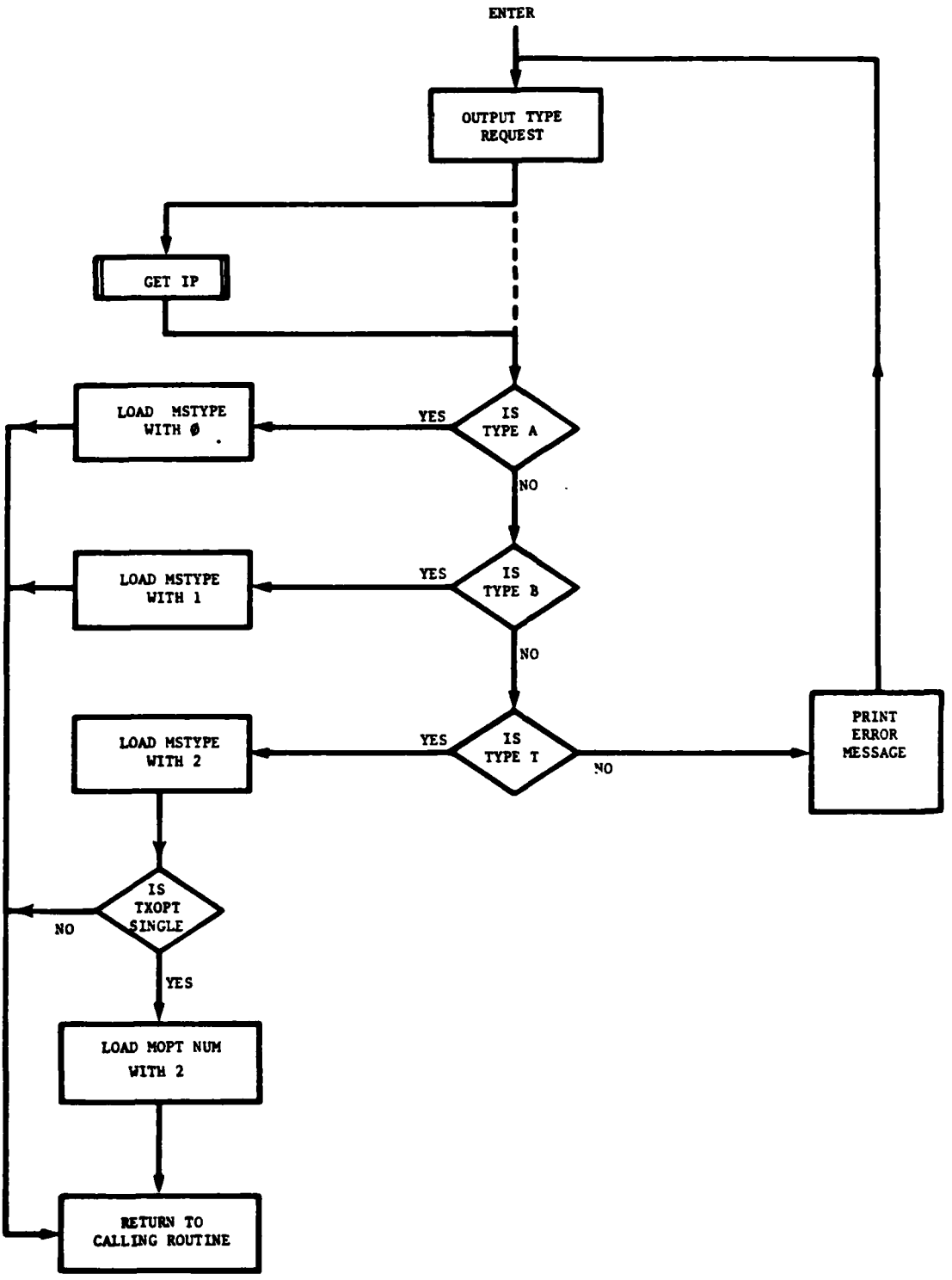


FIG 1Ø INPUT MESSAGE TYPE SUBROUTINE FLOW DIAGRAM

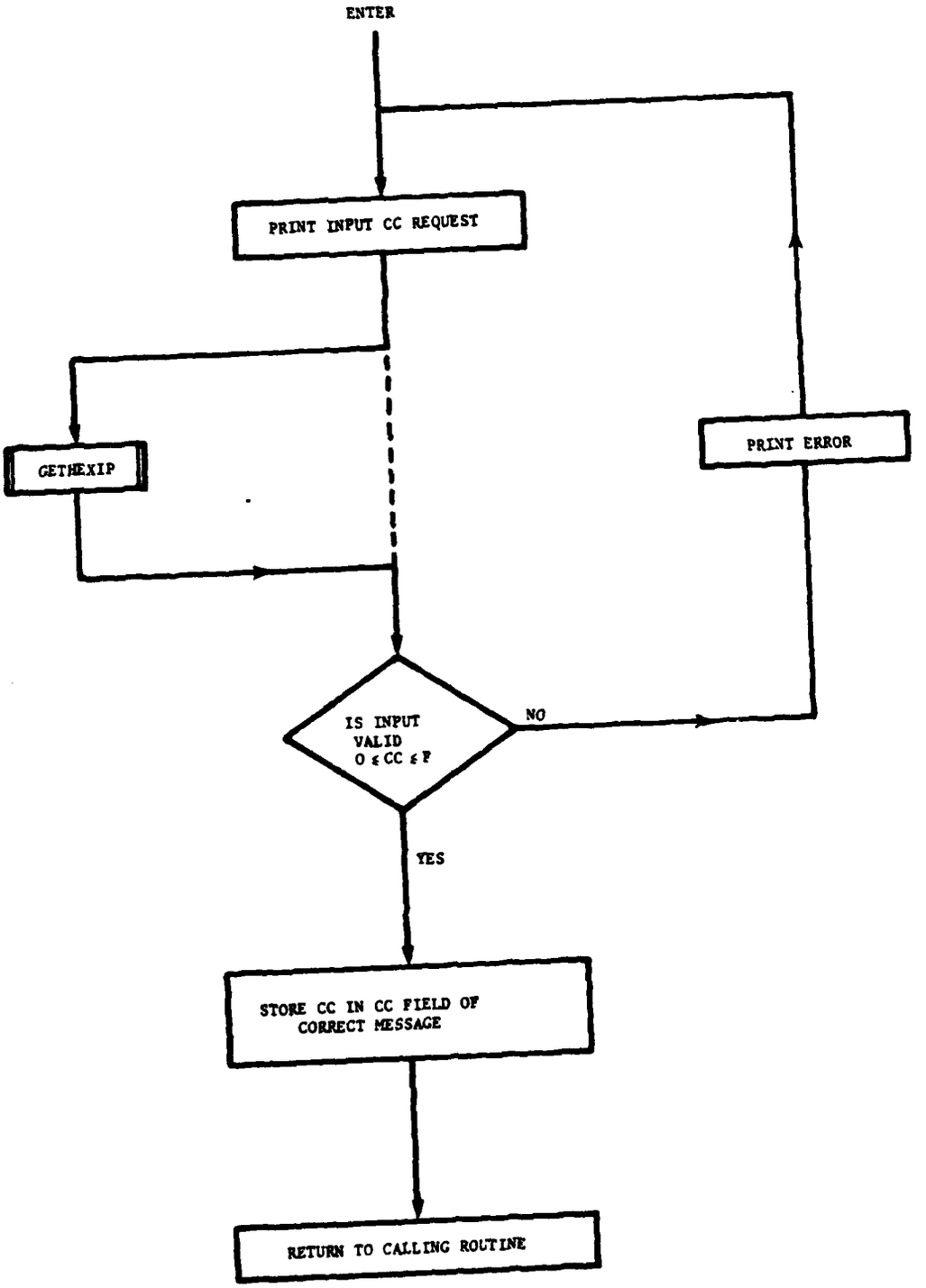


FIG 11 INPUT CHECK CODE SUBROUTINE FLOW DIAGRAM

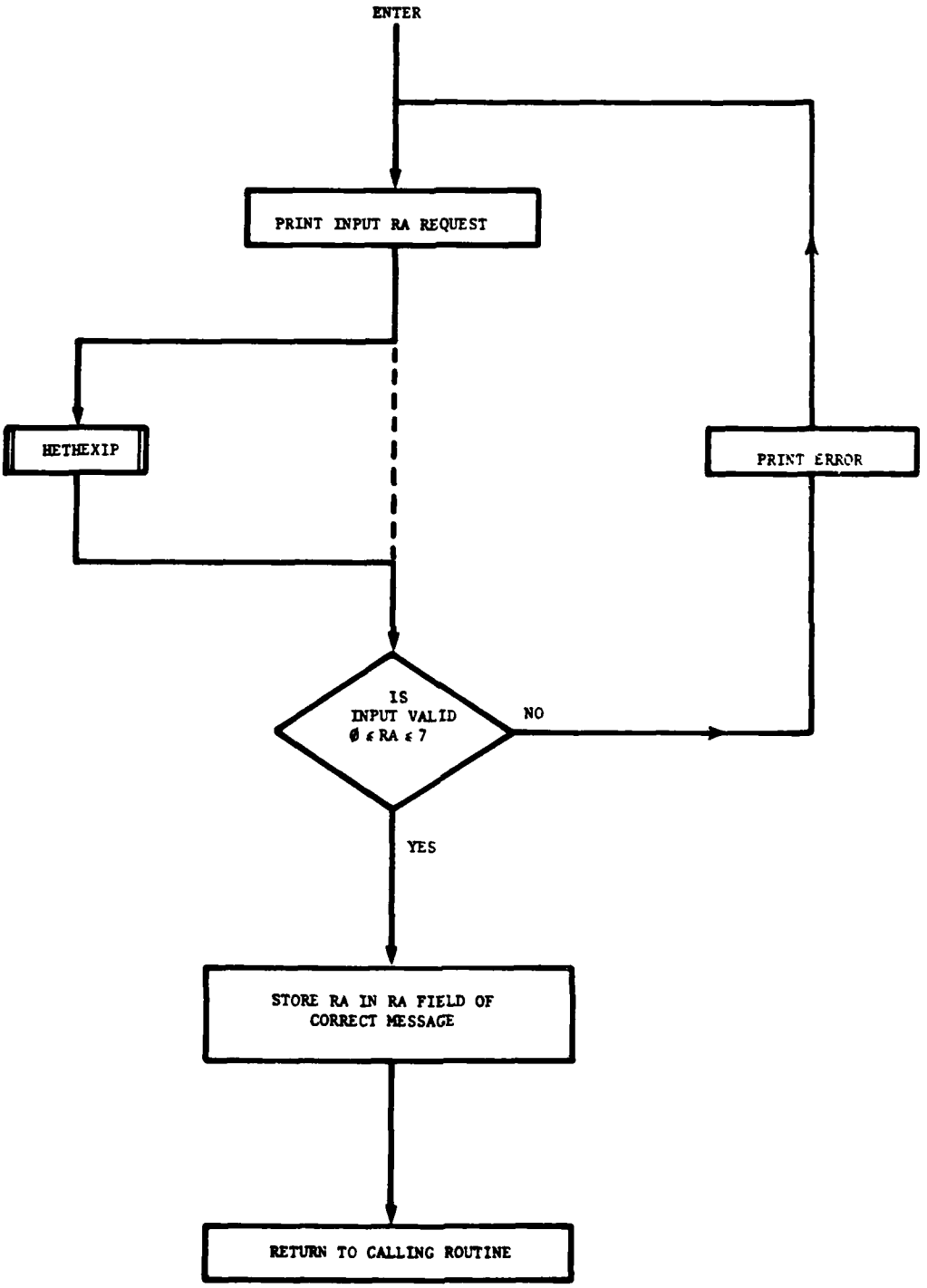


FIG 12 INPUT REGISTER ADDRESS SUBROUTINE FLOW DIAGRAM

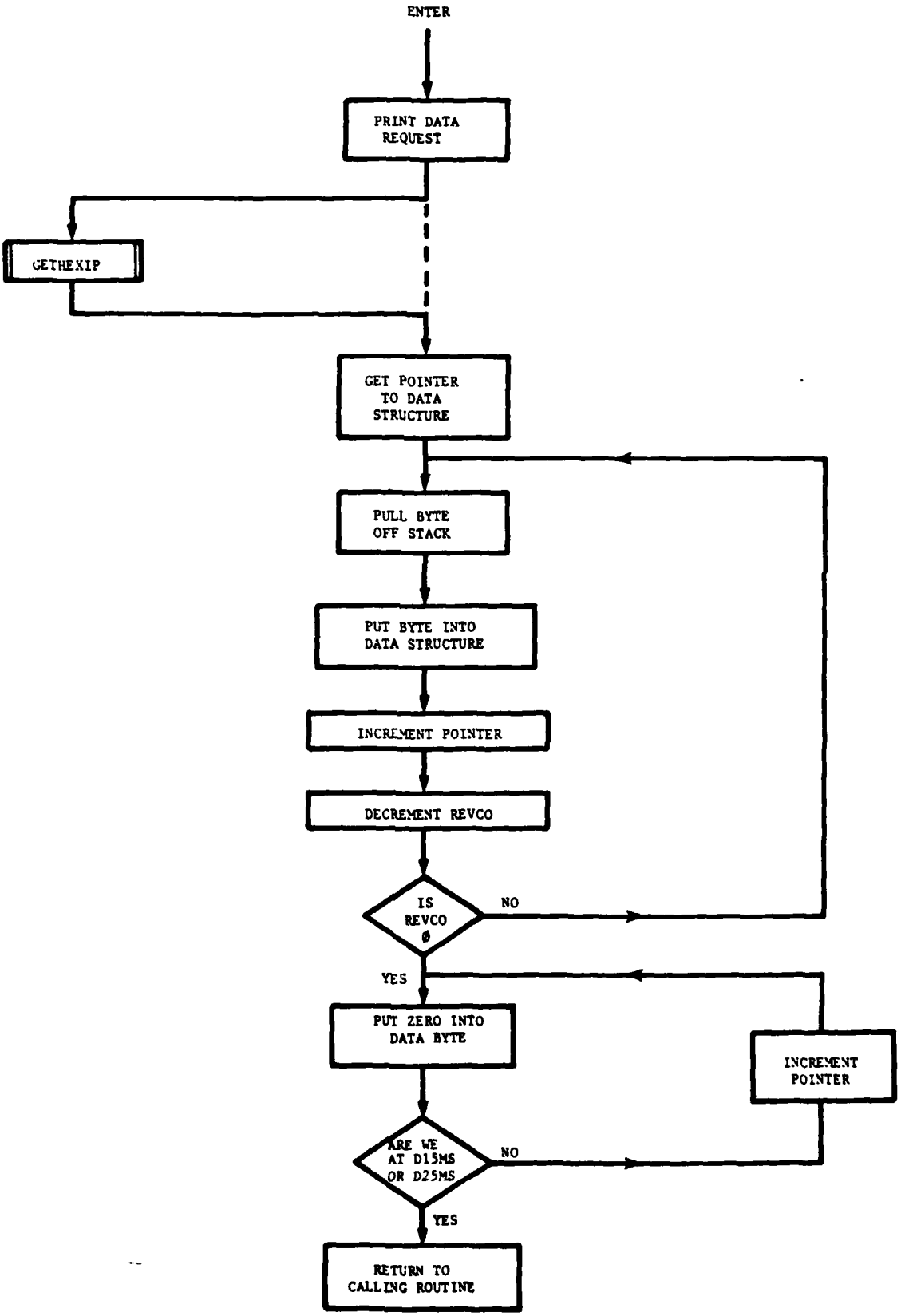


FIG 13 INPUT DATA SUBROUTINE FLOW DIAGRAM

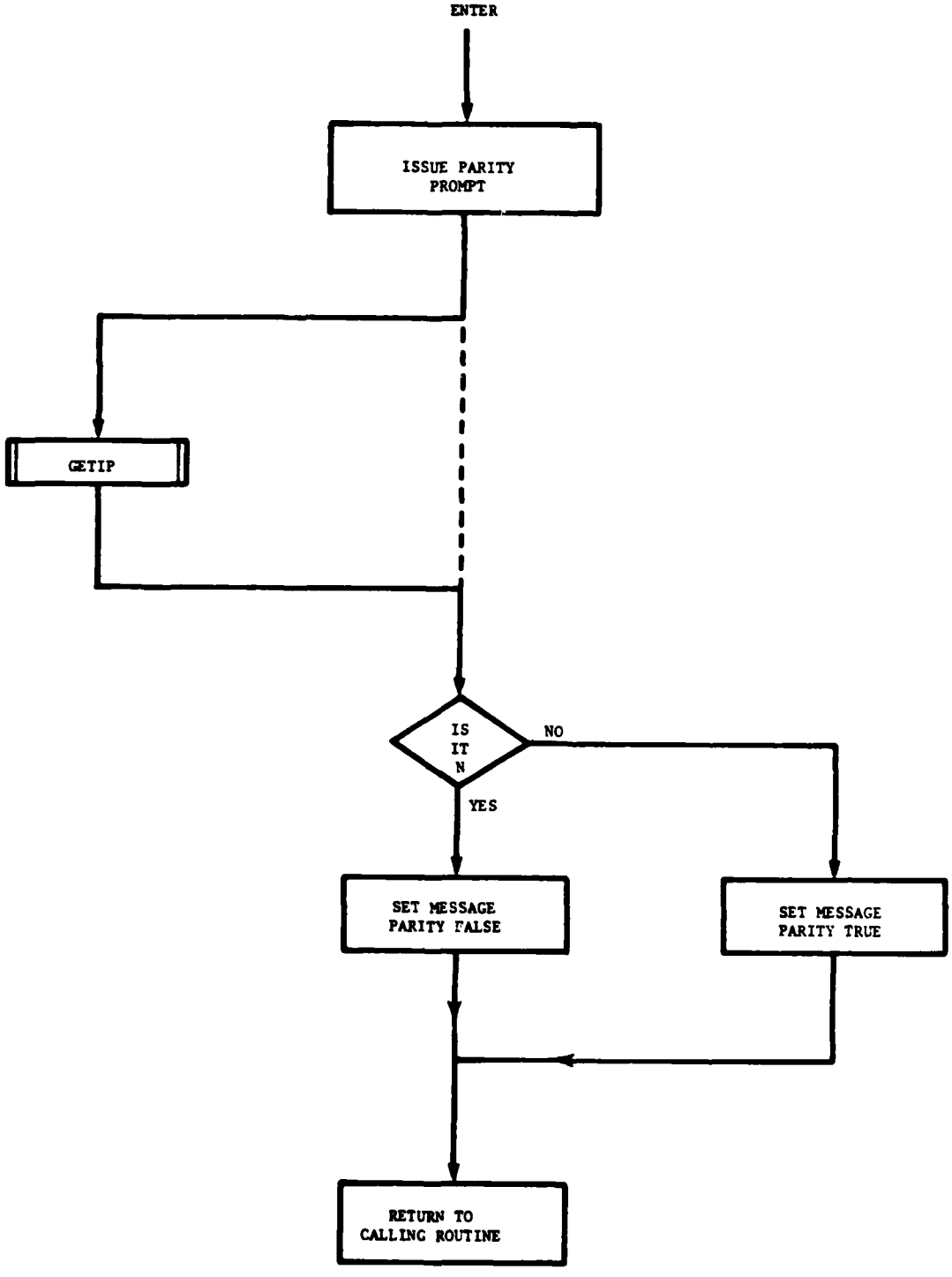


FIG 14 GET PARITY SUBROUTINE FLOW DIAGRAM

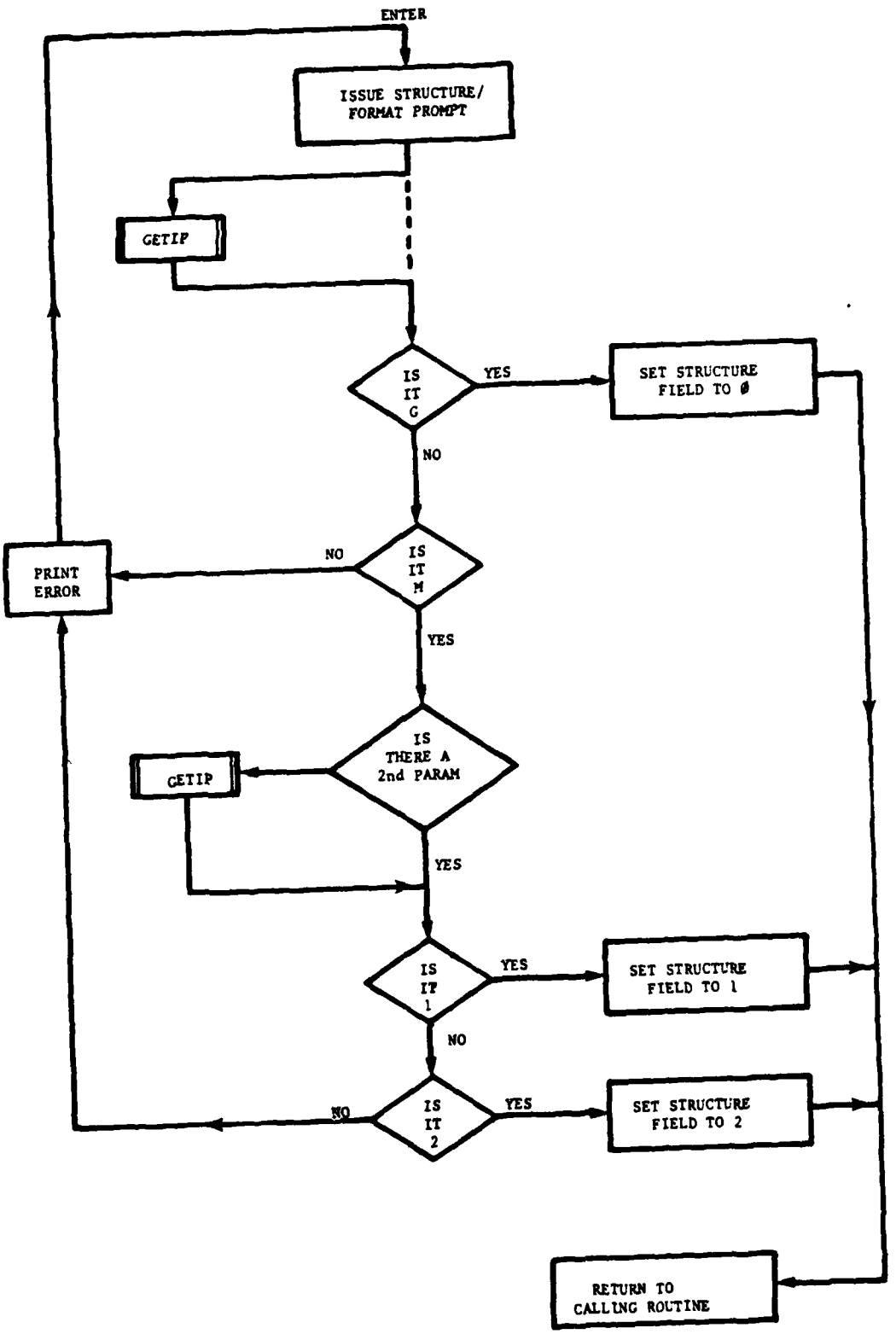


FIG 15 INPUT MESSAGE FORMAT/STRUCTURE SUBROUTINE FLOW DIAGRAM

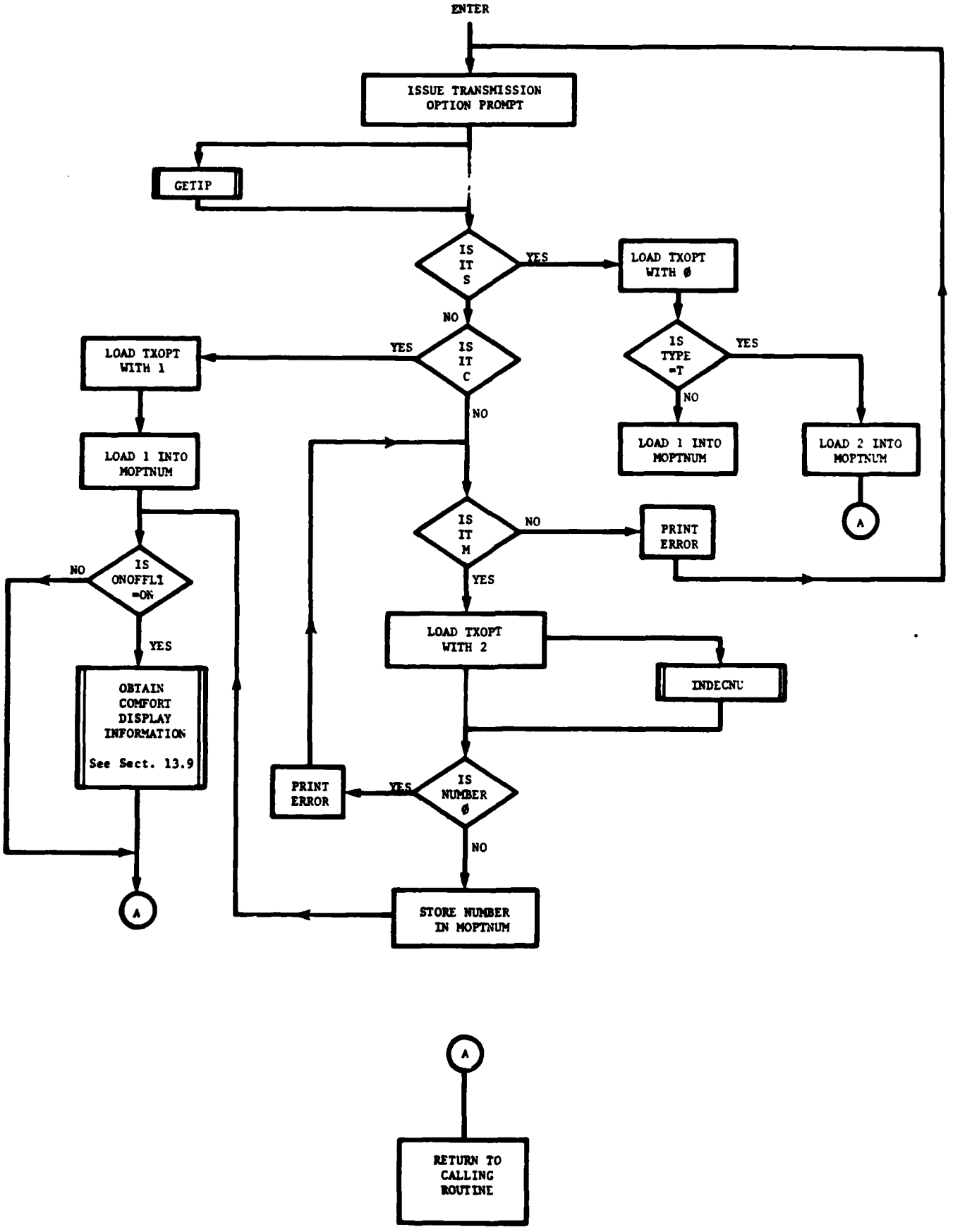


FIG 16 GET TRANSMISSION OPTION SUBROUTINE FLOW DIAGRAM

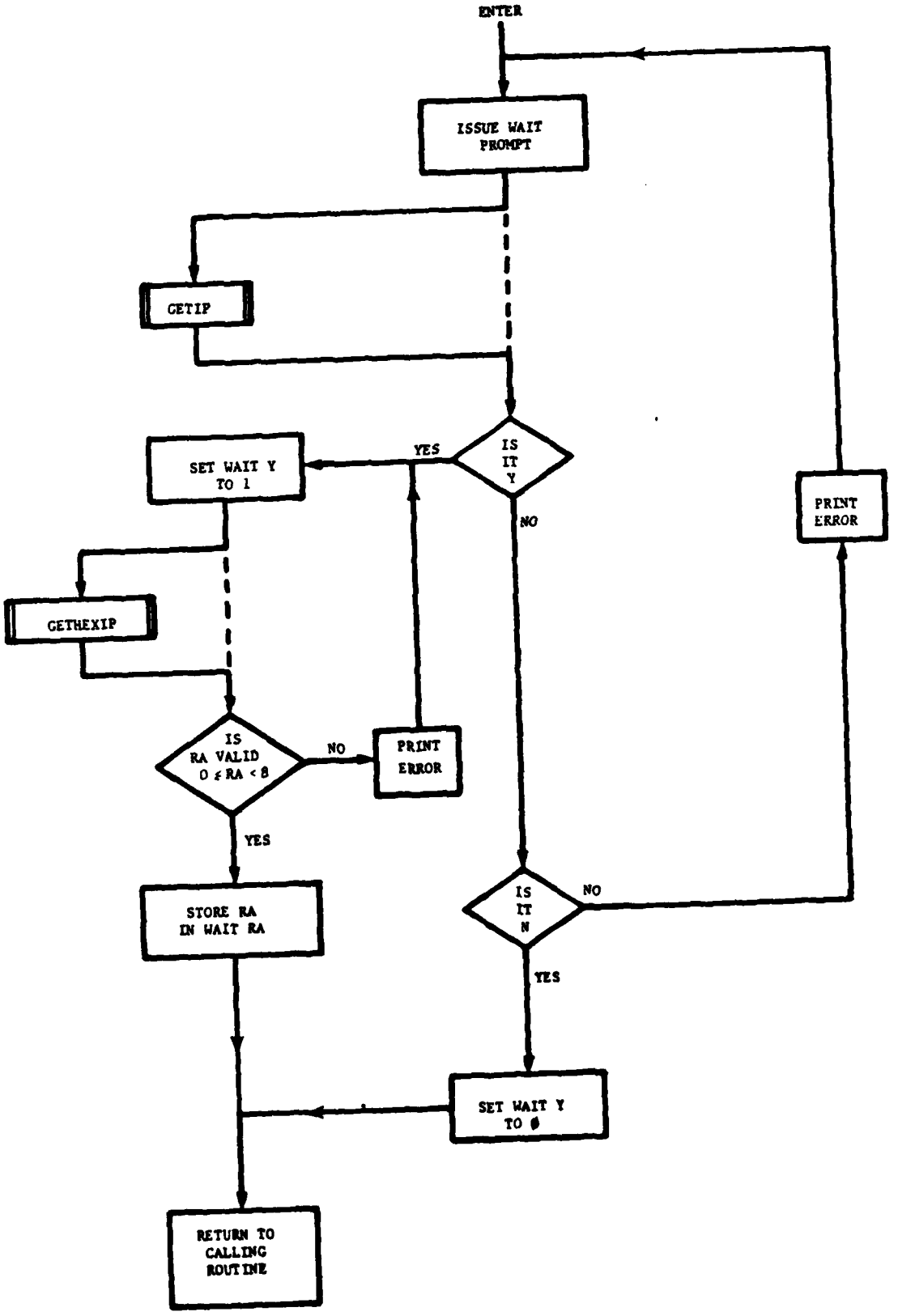


FIG 17 GET WAIT CONDITIONS SUBROUTINE FLOW DIAGRAM

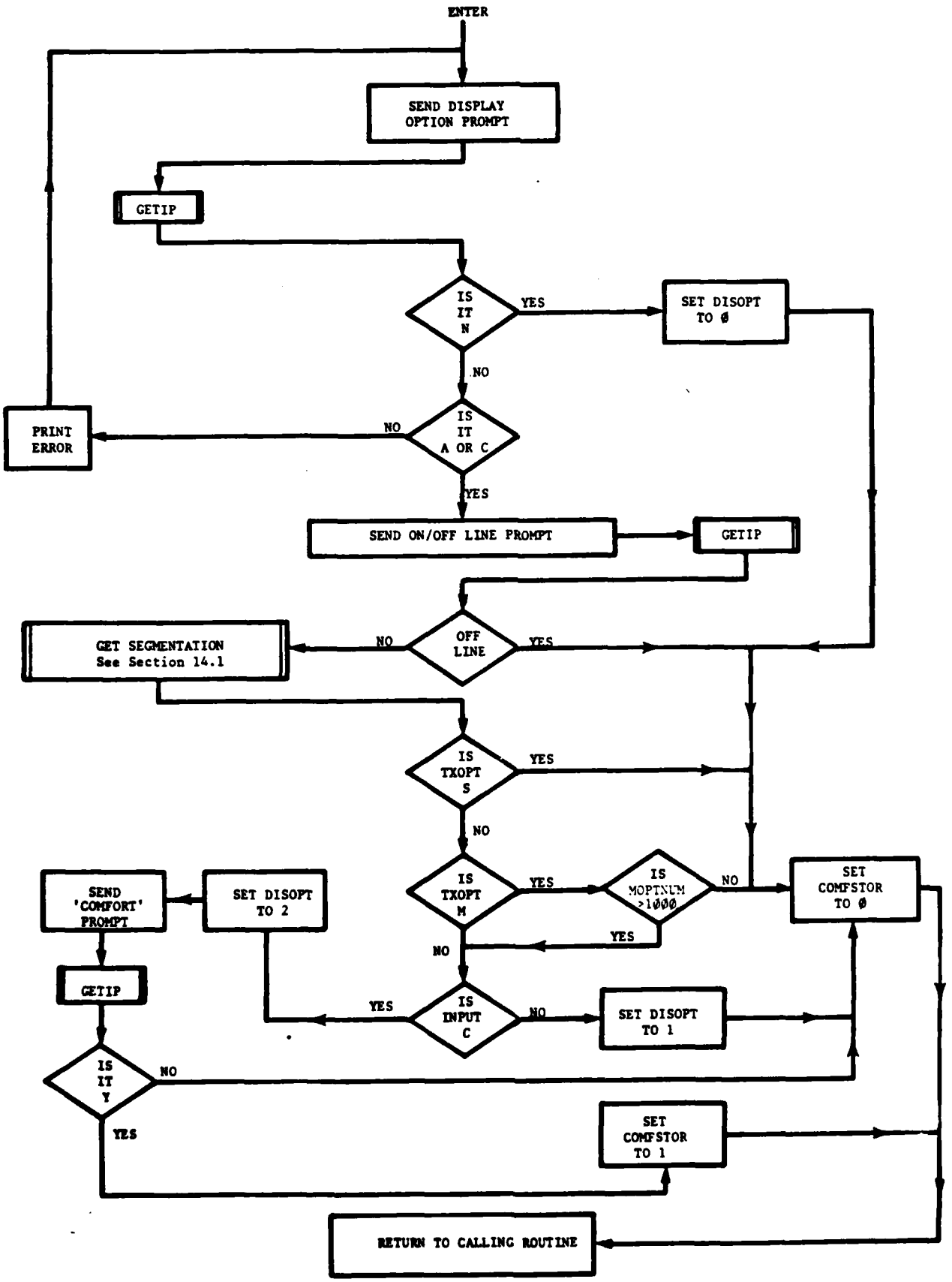
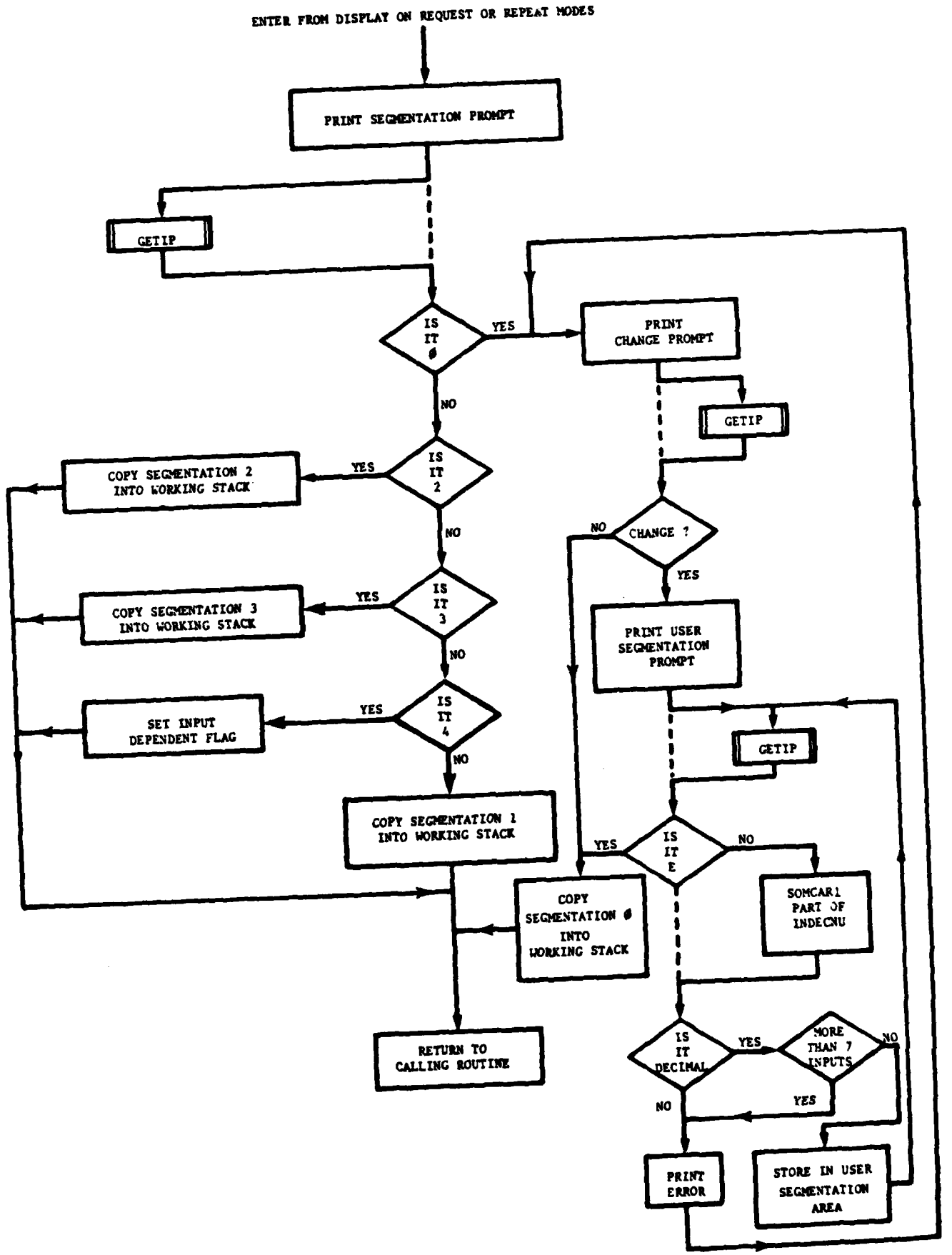


FIG 18 GET DISPLAY OPTION SUBROUTINE FLOW DIAGRAM



SEGMENTATION SUBROUTINE FLOW DIAGRAM

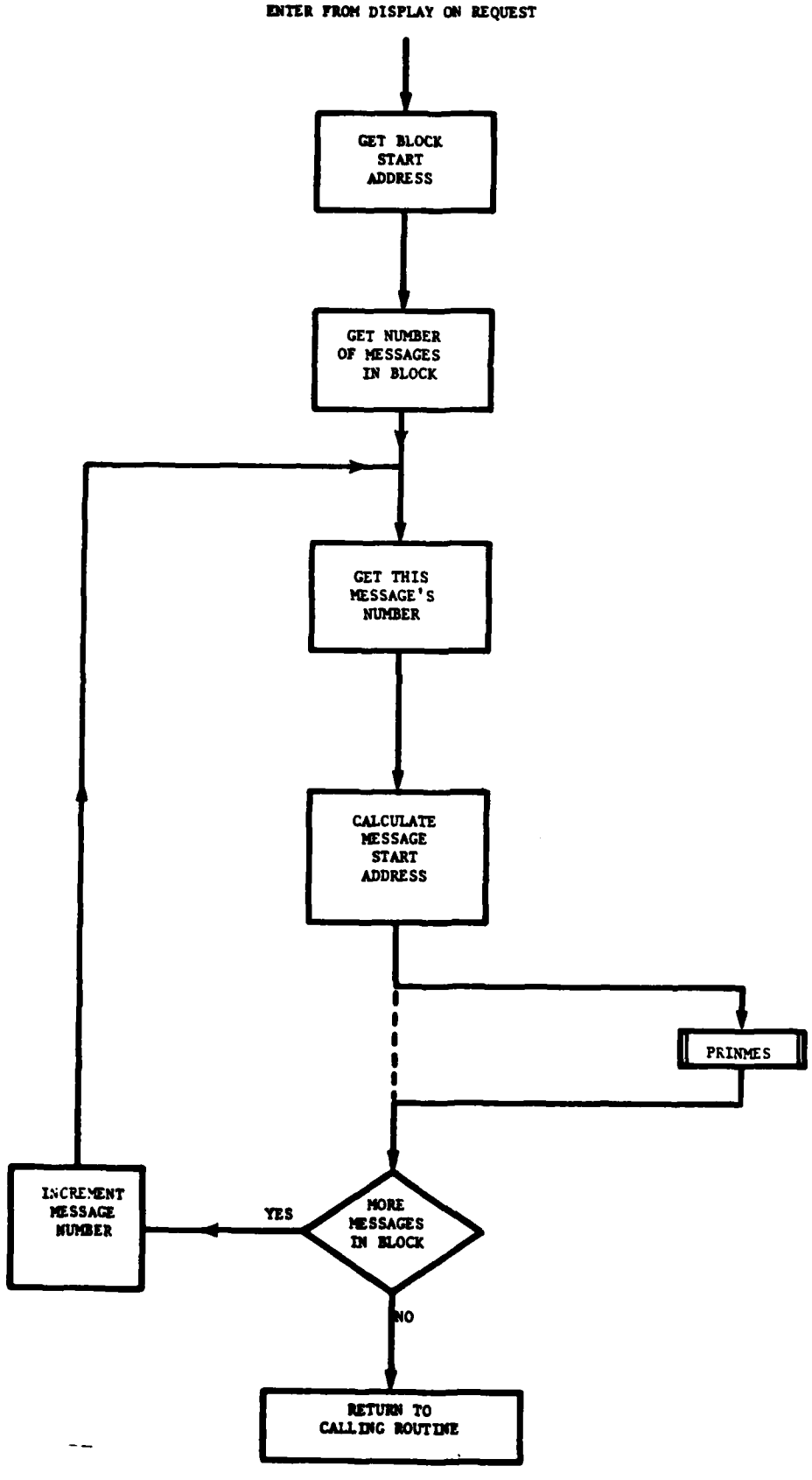
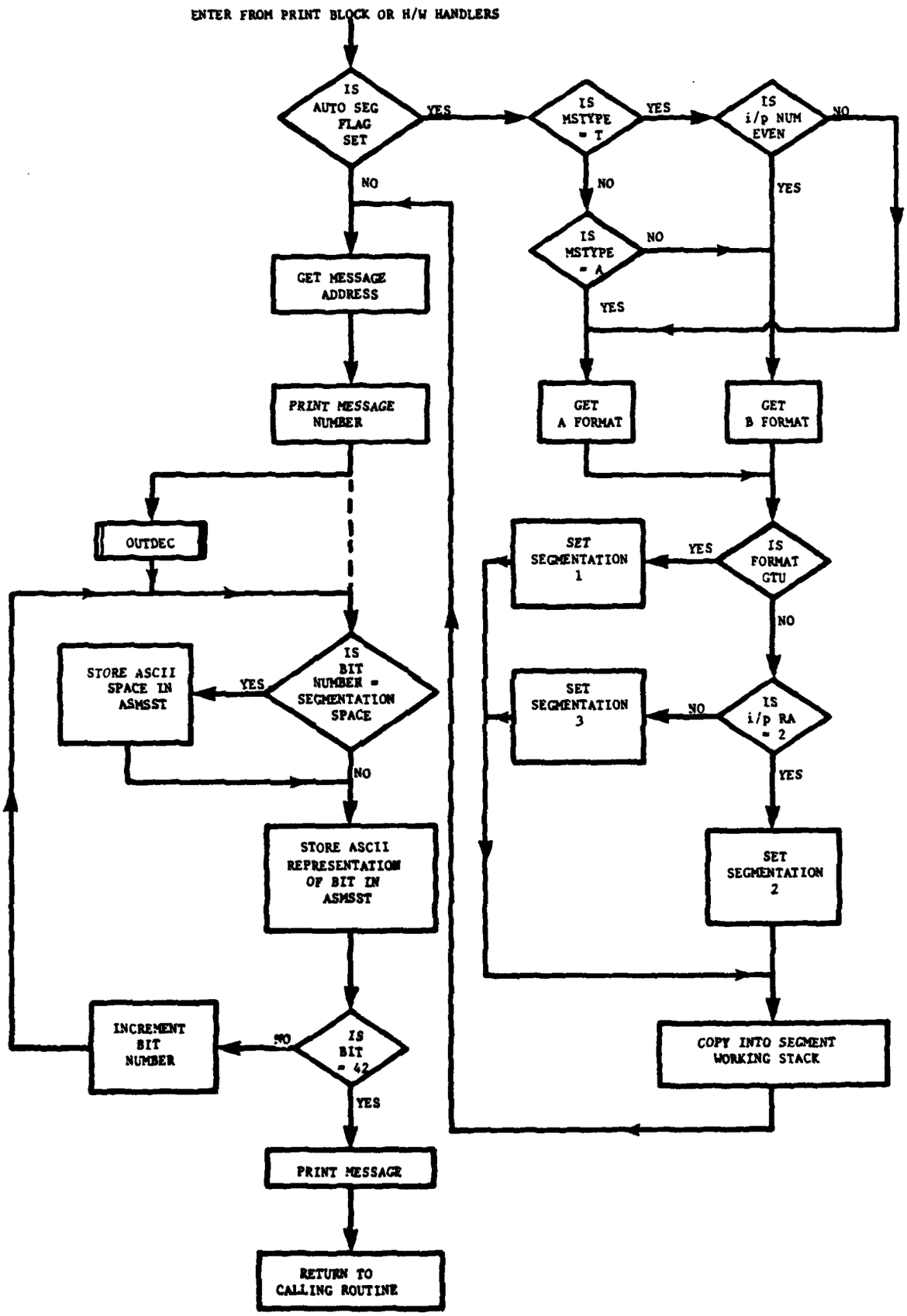


FIG 20 PRINT BLOCK SUBROUTINE FLOW DIAGRAM


ENTER FROM PRINT BLOCK OR H/W HANDLERS
IS AUTO SEG FLAG SET
YES
IS MSTYPE = T
YES
IS i/p NUM EVEN
NO
YES
NO
IS MSTYPE = A
NO
YES
NO
GET MESSAGE ADDRESS
PRINT MESSAGE NUMBER
GET A FORMAT
GET B FORMAT
OUTDEC
SET SEGMENTATION 1
YES
IS FORMAT GTU
NO
STORE ASCII SPACE IN ASMSST
YES
IS BIT NUMBER = SEGMENTATION SPACE
SET SEGMENTATION 3
NO
IS i/p RA = 2
YES
NO
STORE ASCII REPRESENTATION OF BIT IN ASMSST
SET SEGMENTATION 2
INCREMENT BIT NUMBER
NO
IS BIT = 42
YES
COPY INTO SEGMENT WORKING STACK
PRINT MESSAGE
RETURN TO CALLING ROUTINE

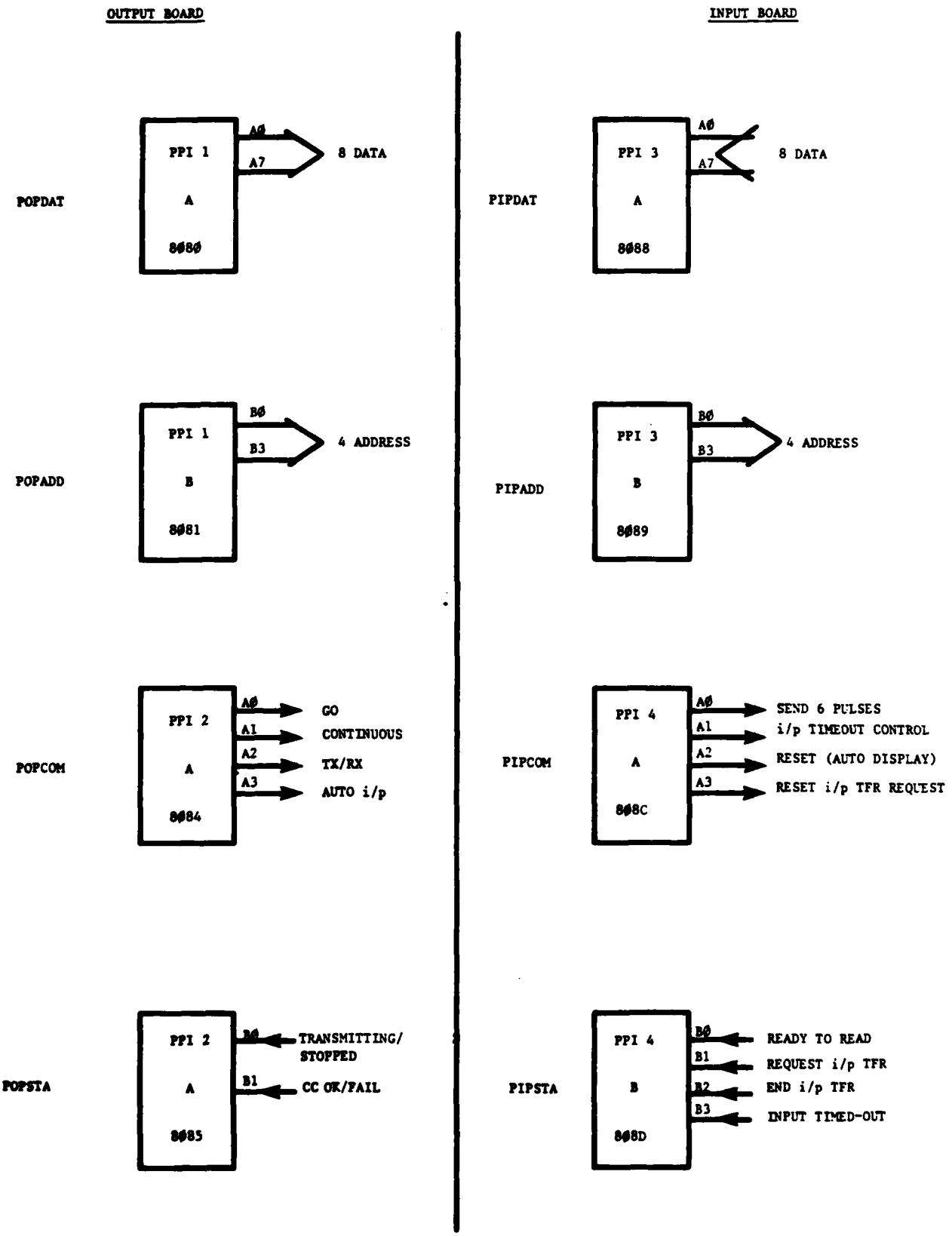


FIG 22 PPI PORT AND BIT ALLOCATIONS

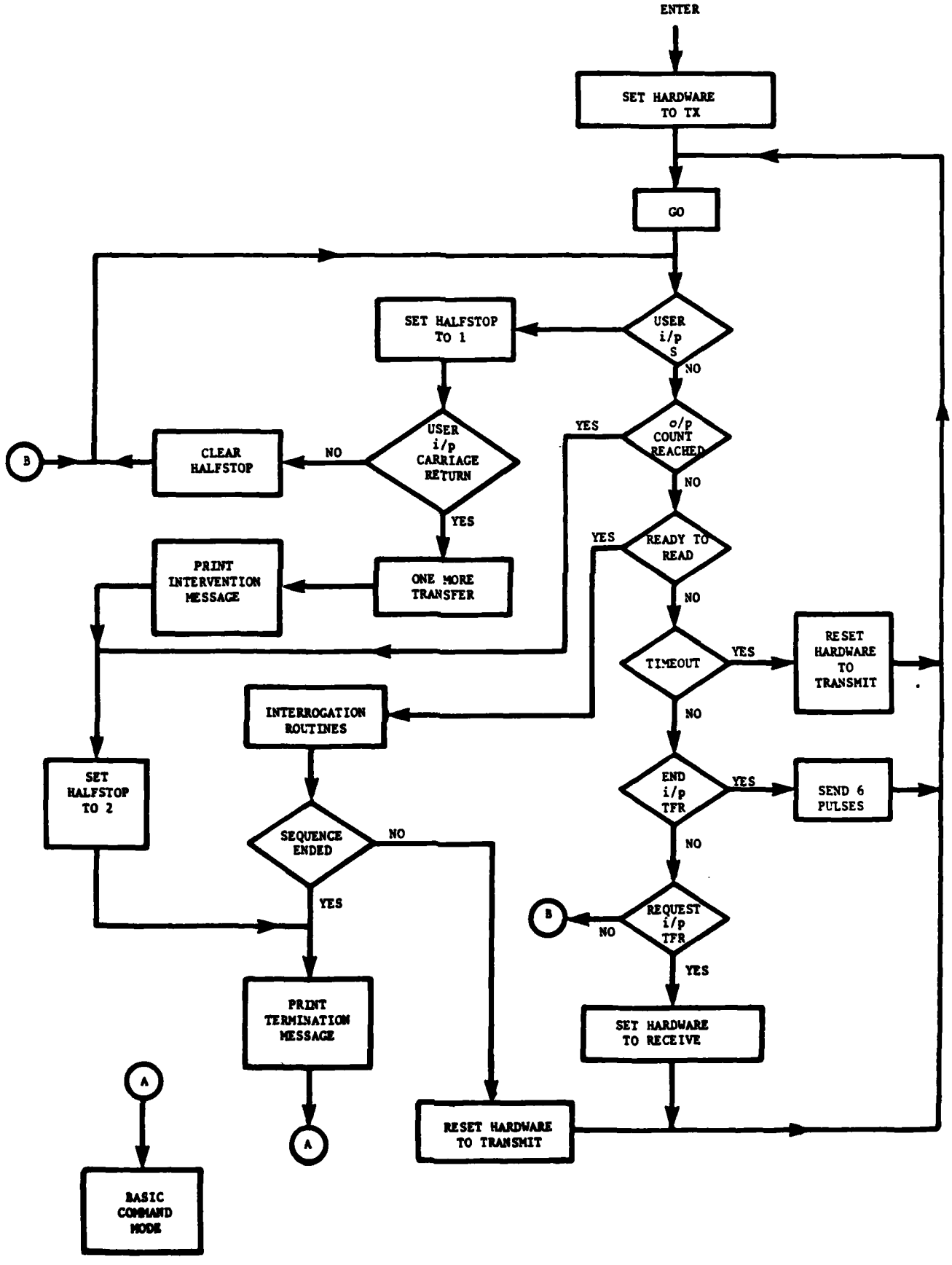


FIG 23 POLLING ROUTINES FLOW DIAGRAM

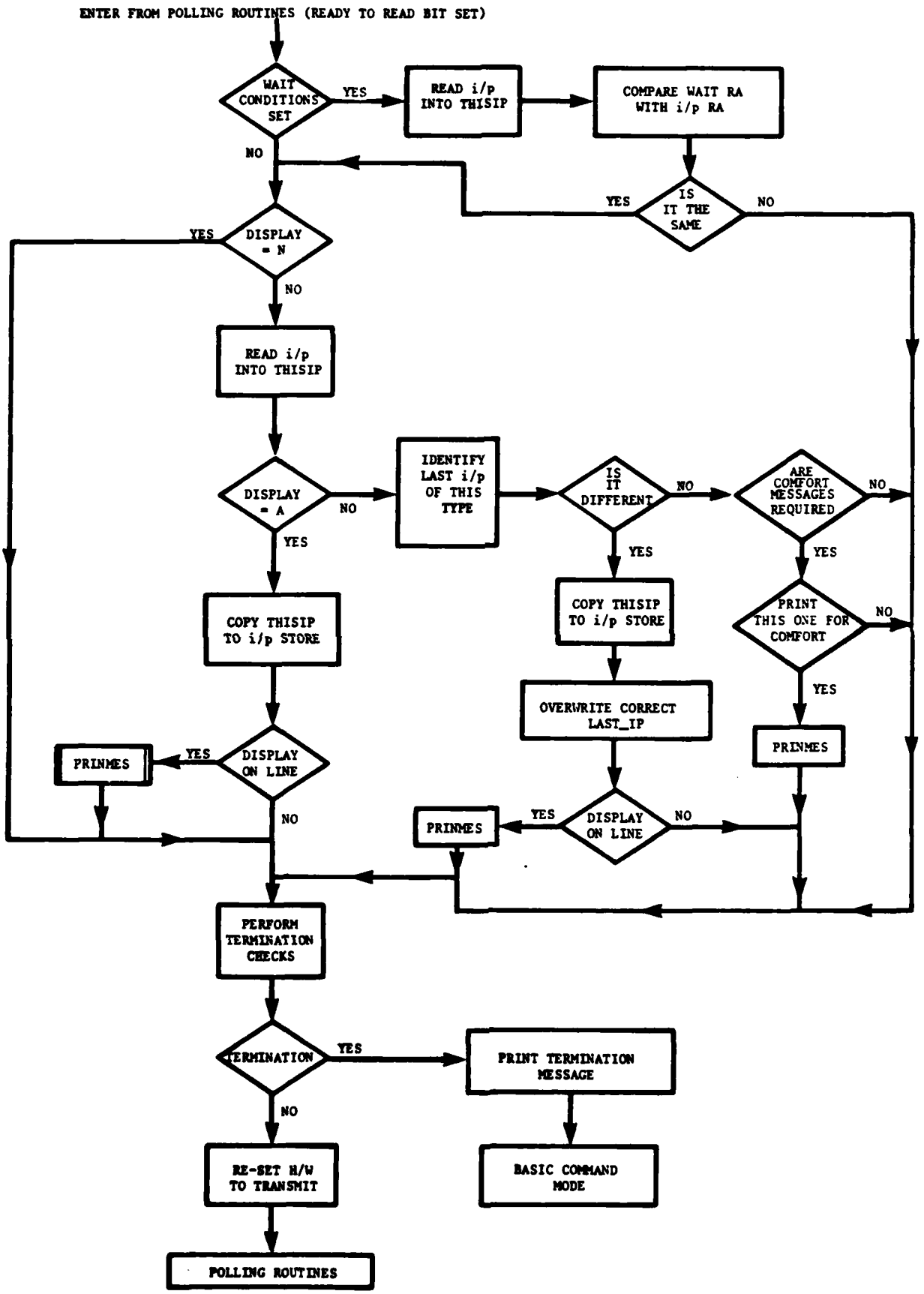
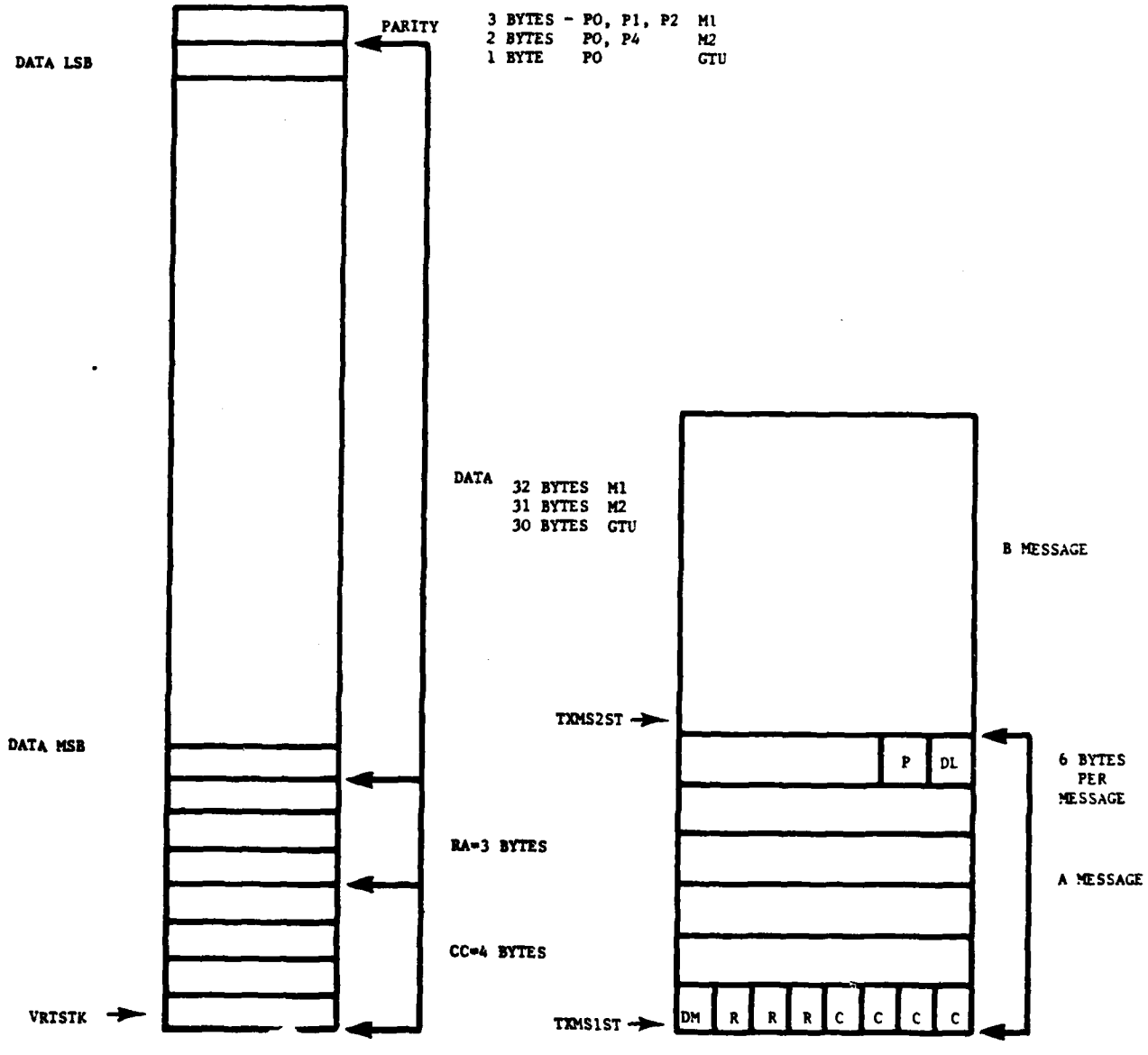


FIG 24 INTERROGATION ROUTINES FLOW DIAGRAM

ON LINE/OFF LINE	ONOFFLI
DISPLAY OPTION	DISOPT
WAIT REGISTER ADDRESS	WAITRA
WAIT Y OR D	WAITY
NUMBER	OPTXNUM1
NUMBER	OPTXNUM
NUMBER	MOPTNUM1
NUMBER	MOPTNUM
TRANSMISSION OPTION	TXOPT
'B' MESSAGE FORMAT/STRUCTURE	B FORMAT
'B' PARITY	BPARTRU
'B' DATA	D25MS
'B' DATA	D24MS
'B' DATA	D23MS
'B' DATA	D22MS
'B' DATA	D21MS
'B' REGISTER ADDRESS	RA2MS
'B' CHECK CODE	CC2MS
'A' MESSAGE FORMAT/STRUCTURE	A FORMAT
'A' PARITY	APARTRU
'A' DATA	D15MS
'A' DATA	D14MS
'A' DATA	D13MS
'A' DATA	D12MS
'A' DATA	D11MS
'A' REGISTER ADDRESS	RA1MS
'A' CHECK CODE	CC1MS
MESSAGE TYPE	MSTYPE

FIGURE 25 INTERNAL MESSAGE STRUCTURE



KEY
M1 = MATRIX CONNECTION
M2 = MATRIX COMMAND

FIGURE 26A FORMED FOR PARITY

FIGURE 26B READY FOR TRANSMISSION

FIG 26 TX MESSAGE STRUCTURE

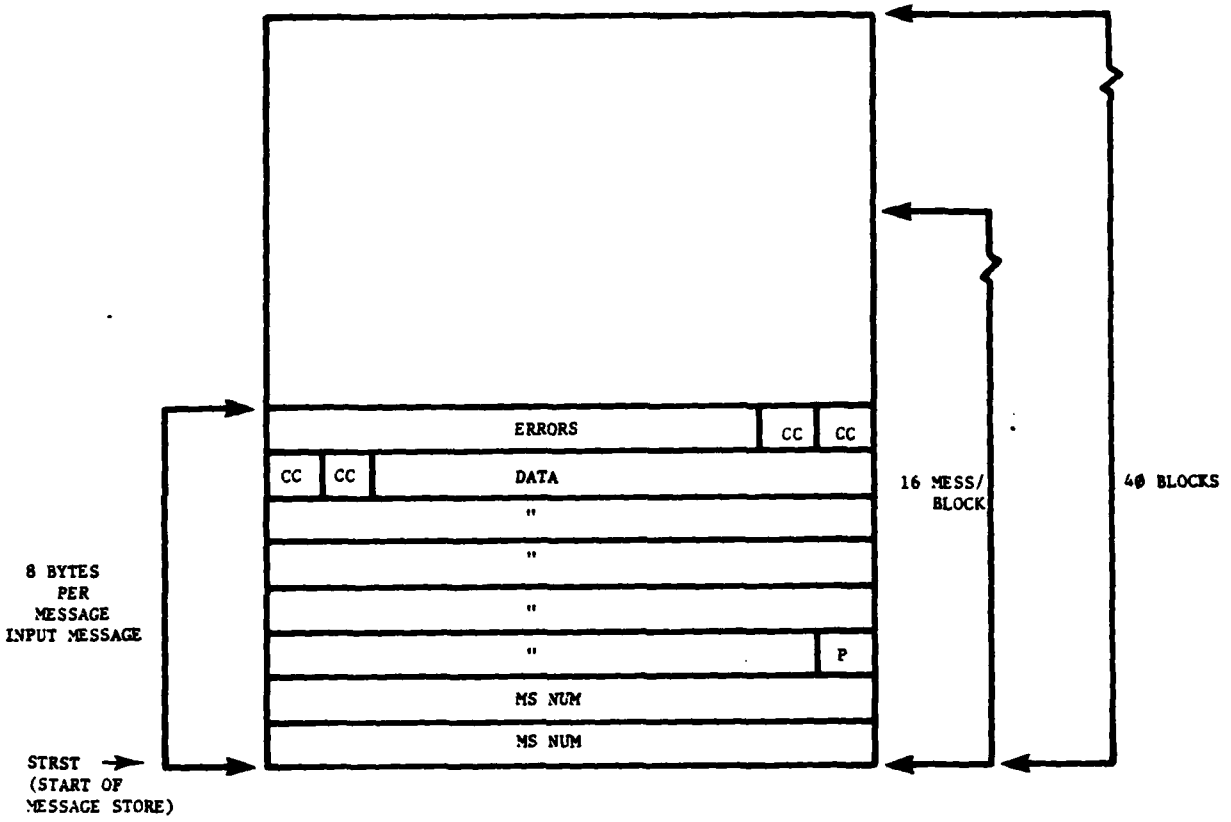


FIG 27A INPUT MESSAGE STORE FORMAT

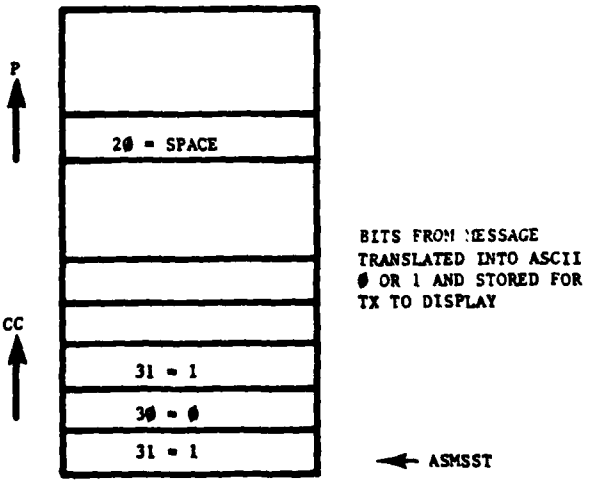
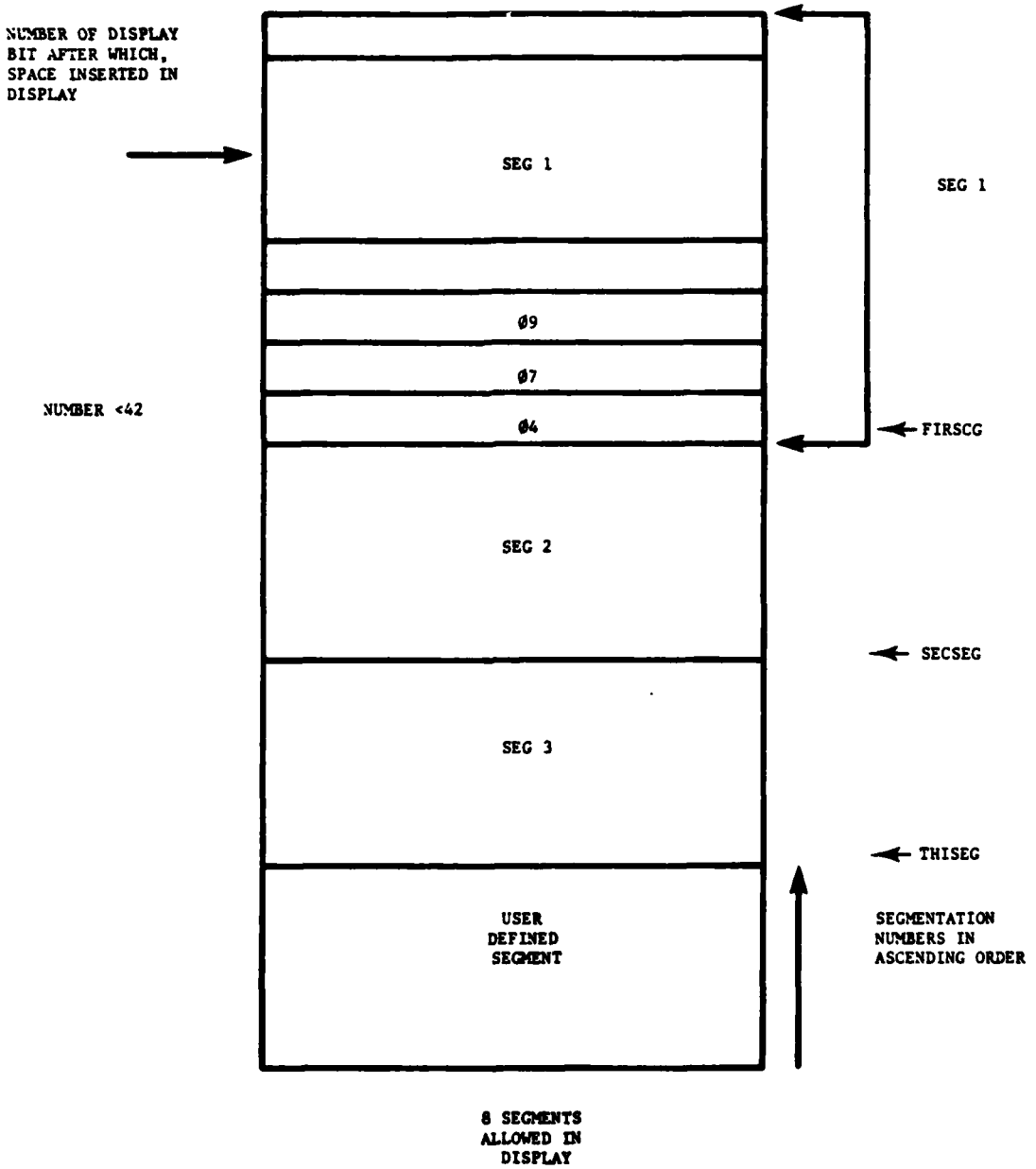


FIG 27B READY TO DISPLAY


NUMBER OF DISPLAY BIT AFTER WHICH, SPACE INSERTED IN DISPLAY
SEG 1
SEG 1
Ø9
Ø7
NUMBER <42
Ø4
FIRSCG
SEG 2
SECSEG
SEG 3
THISEG
USER DEFINED SEGMENT
SEGMENTATION NUMBERS IN ASCENDING ORDER
8 SEGMENTS ALLOWED IN DISPLAY

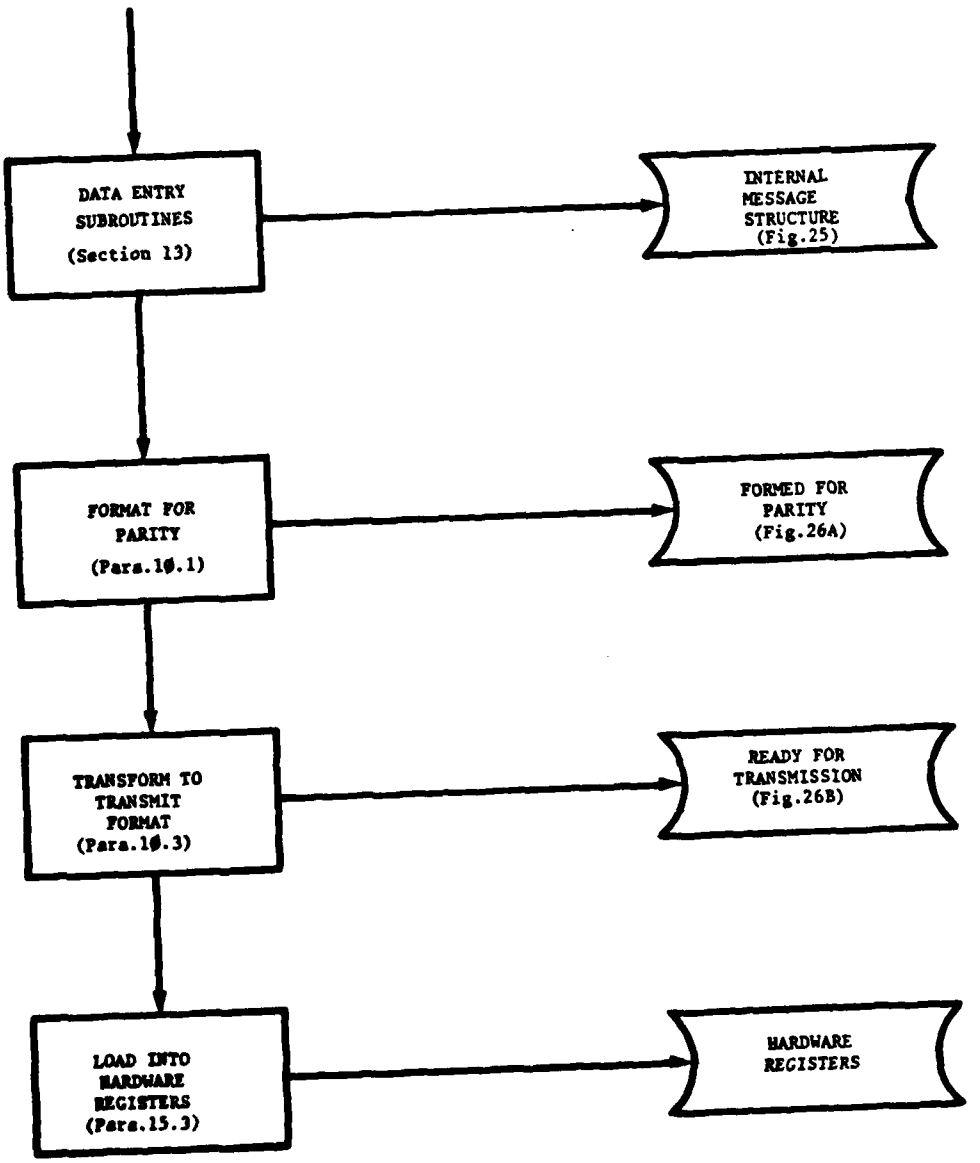


FIG.28 TRANSMIT DATA FLOW

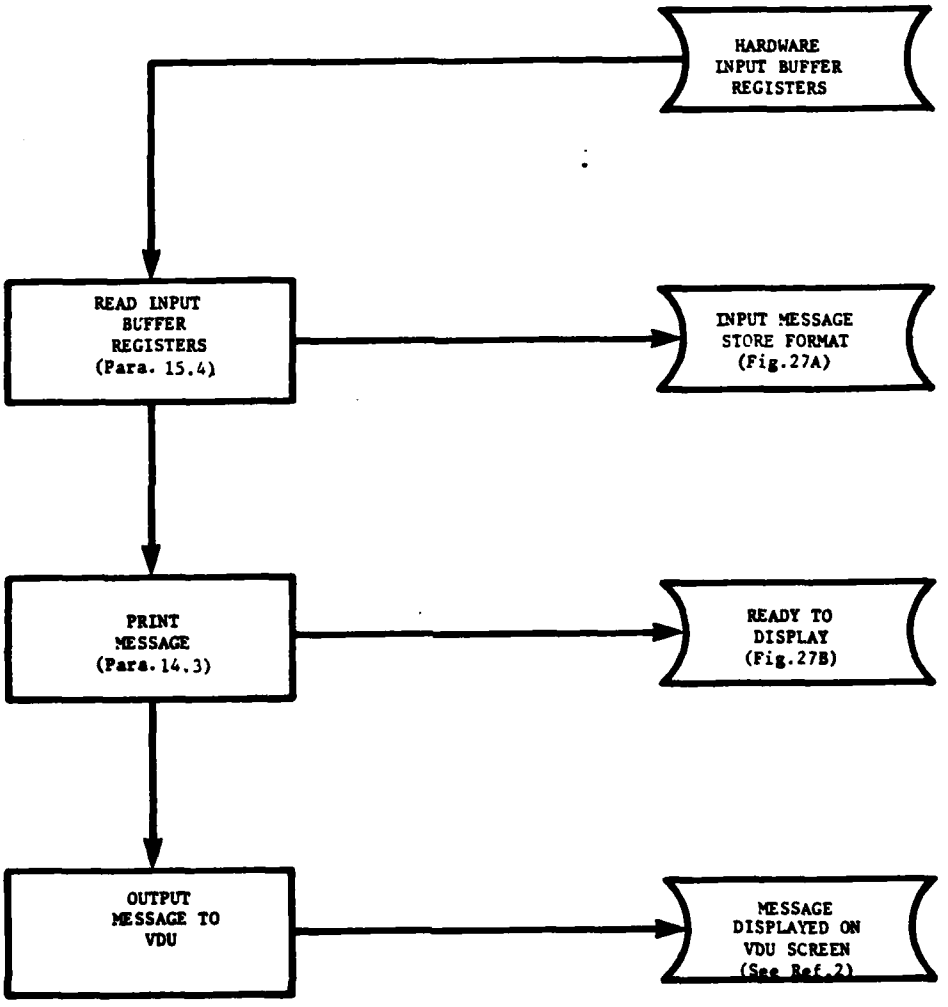


FIG 30 RECEIVE DATA FLOW

DOCUMENT CONTROL SHEET

Overall security classification of sheet ..... UNCLASSIFIED .....

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference	3. Agency Reference	4. Report Security Classification
	MEMORANDUM 3585		Unclassified

5. Originator's Code (if known)	6. Originator (Corporate Author) Name and Location
	Royal Signals and Radar Establishment

5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location

7. Title

The Axis Test Box Software Report

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials	9(a) Author 2	9(b) Authors 3,4...	10. Date	pp. ref.
Simcock, A L				

11. Contract Number	12. Period	13. Project	14. Other Reference

15. Distribution statement

Unlimited

Descriptors (or keywords)

continue on separate piece of paper

Abstract

This memorandum describes in detail the software program which the M6809 microprocessor obeys to perform the functions of the AXIS Test Box.

Extensive use is made of Flow Diagram where these either enhance or replace wordy descriptions.

FIG 18 GET DISPLAY OPTION SUBROUTINE FLOW DIAGRAM