File No. S360-25
Form Y27-7152-0

Program Logic

IBM System/360 Operating System
Graphic Programming Services for FORTRAN IV

Program Number 360S-LM-537

This publication describes the internal logic of the graphic subroutine package (GSP) for FORTRAN IV. GSP is a program that enables a FORTRAN programmer to write graphic programs for use with the IBM 2250 Display Unit in association with the IBM System/360 Operating System. It is not an extension of FORTRAN IV, but does extend graphic capabilities via subroutines and functions to the FORTRAN programmer. GSP may also be used in an assembler language program.

This program logic manual is intended for use by persons involved in program maintenance and system programmers who are altering the program design. Its primary purpose is to serve as a guide to the program listings with which it is to be used. Since program logic information is not necessary for the operation of the program, distribution of this publication is limited to those with the aforementioned requirements.

Restricted Distribution

PREFACE

This publication discusses the operation of the graphic subroutine package (GSP) routines that constitute the graphic programming services for FORTRAN IV.

The publication is organized into three sections. The first section describes the overall structure of GSP and, explains operations (such as initialization and parameter testing) that are performed similarly by many of the GSP routines. The second section discusses the program organization, including functions and logic of GSP routines. The third section contains flowcharts for the GSP routines.

Appendixes at the end of this publication provide information pertaining to the control blocks created by GSP routines, a module directory, and other reference material related to the program.

For detailed information about programming techniques used in these routines, refer to the applicable program listings.

PREREQUISITE PUBLICATIONS

Familiarity with the following publications and their prerequisites is assumed:

IBM System/360 Operating System: Graphic Programming Services for FORTRAN IV, Form C27-6932

IBM System/360 Operating System: Graphic Programming Services for IBM 2250 Display Unit, Form C27-6909.

In addition, the following publications may be convenient for reference:

IBM System/360 Operating System: Graphics Access Method, Program Logic Manual, Form Y27-7113

IBM System/360 Operating System: Supervisor and Data Management Macro Instructions, Form C28-6647.

First Edition (September 1967)

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

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

A forn for reader's comments appears at the back of this publication. Address any additional comments concerning the contents of this publication to: IBM Corporation, Programming Systems Publications, Department 637, Neighborhood Road, Kingston, New York 12401

ILLUSTRATIONS

FIGURES

TABLES

ILLUSTRATIONS

CHARTS

The graphic programming services for FORTRAN IV, called the graphic subroutine package (GSP), enable displays to be produced on one or more IBM 2250 Display Units attached to an IBM System/360 Computing System. These services are also usable with the assembler language.

CHARACTERISTICS OF GSP ROUTINES

GSP consists of subroutines and functions that can be included in a FORTRAN program, and internal routines that perform when certain subroutines and functions are executed.

Location

Three of the GSP routines (INGSP Part 1, Director Part 1, and TMGSP Part 1) reside in the FORTRAN library (SYS1.FORTLIB). All other GSP routines reside in the link library (SYS1.LINKLIB). The routines in the FORTRAN library are processed by the linkage editor at the same time the user's program is processed by the linkage editor. Therefore, these routines remain in main storage throughout execution of the user's program and are known as resident modules.

Attributes

All routines have the same two attributes: (1) problem program mode, and (2) reenterable coding. Main storage areas acquired by GSP routines are allocated from subpool zero.

Methods of Passing Control

The routines that reside in the link library are brought into main storage either by a LINK or a LOAD macro instruction. Control is passed to the routine in one of two ways:

1. LINK causes control to be passed to the routine automatically.

2. LOAD brings the routine into storage; a CALL macro instruction is then issued to pass control to it.

A copy of a routine brought into main storage via a LOAD macro instruction remains in main storage until the use of GSP is terminated. A copy of a routine brought into main storage via a LINK macro instruction is removed from main storage once its execution has been completed.

The module directory in Appendix B lists the method of entry for each GSP routine.

Level of Support

GSP offers two levels of support in regard to keying and correlating features. The term level 1 refers to the standard GSP keying and correlating features, and the term level 2 refers to a programmer-defined correlation scheme. The level under which a routine is functioning may affect the action taken at specific points in the operation. For additional information on these two levels, refer to the publication IBM System/360 Operating System: Graphic Programming Services for FORTRAN IV, Form C27-6932.

CONTROL FLOW AMONG GSP ROUTINES

All GSP subroutines are invoked by a CALL statement that is issued in the user's program. All of the subroutines (except INGSP and TMGSP) receive control through the Director. Figure 1 depicts linkage from a CALL statement to a GSP subroutine, which in turn calls another routine (called an external reference) to complete its function. The figure also shows the path taken when control is returned to the calling routine (calling program).

In Figure 1, subroutine A is called from the user's program. From the CALL statement, control is passed to Director Part 1 which computes the internal entry point for the called subroutine and passes control to Director Part 2. Director Part 2 accesses the status table to determine if the routine is already in main storage or if it must be brought into main storage, brings the routine into main storage if necessary, designates a work area to be used by the routine, and passes control to the routine (in this case, A). If A calls another routine (in this case, B), it also accesses the status table to determine the location of the routine, brings the routine into main storage if necessary, constructs a parameter list for that routine, and passes control to the routine.

Once control has passed from Director Part 2, control is always returned from the called routine to the next sequential instruction in the routine that called it. This culminates in control being returned back to Director Part 2 which in turn passes control to the next sequential instruction in the user's program. Control is never returned to Director Part 1.

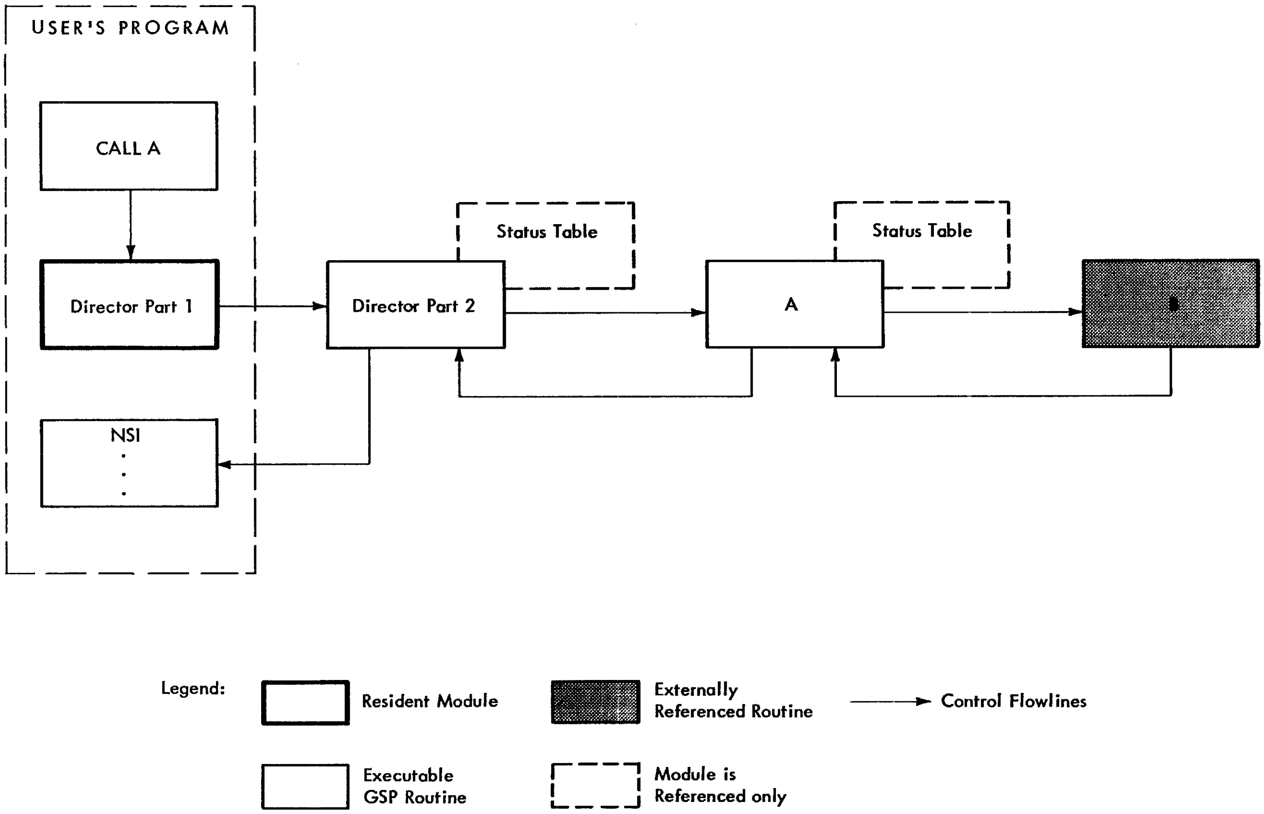


Figure 1. Linkage from a CALL to a GSP Subroutine that Uses Another GSP Routine

Table 1 shows the relationship between the routine that is called to perform a function and other GSP routines (external references) that it calls to perform the complete operation. The GSP routines called are listed alphabetically down the first column. External references are listed across the top of the table. Those routines not listed perform their complete operation without referring to GSP external references during processing.

Control flow among the initiation and termination subroutines, and among the image generation subroutines is described in the sections "Initiation and Termination of GSP" and "Image Generation Subroutines," respectively.

INPUT TO MANY GSP SUBROUTINES

Input to GSP subroutines consists of a variable length parameter list that is set up when the user issues a call to any GSP subroutine. This list consists of full-word entries containing the addresses of the arguments specified by the user.

All GSP subroutines except INGSP and TMGSP receive the address of a work/save area in register 1. The first word of this area contains the address of the parameter list described here.

CONTROL BLOCKS AND TABLES USED BY GSP

To establish communication links between the user's program, GSP, and each 2250 Display Unit, several control blocks and tables are created and used by GSP. The control blocks and tables created, initialized, and used during the processing of GSP subroutines are described briefly in the paragraphs that follow.

GSP control blocks, for which the formats and contents are described in Appendix A, are:

- Graphic subroutine package control block (GSPCB), which is constructed by INGSP for communication between the user's program and GSP subroutines; only one of these control blocks should exist.

Table 1. GSP External References and Relationship

	External References									
GSP Routine Called	Flow Ctrl	Buffer Mgt	Key Tbl Mgt	Scaling	Scis- soring	Data Gen	Data Store	Update	RESET	Other
BGSEQ			X				X			
BGSUB			X				X			
BGTRK									X	CNVRT EXEC ORGEN
CNVRT				X						
Data Store	X	X								
Data Gen							X			
ENSEQ			X				X			
ENSUB			X				X			
ENTRK									X	
GSPRD			X							
ICURS			X							RCURS
IDPOS				X	X					
INCL			X							
INDEV	X									INGDS TMGDS
ITBP										CNVRT
LKSUB			X				X			
LOCPN									X	CNVRT EXEC ORGEN
MVPOS			X	X	X	X		X	X	
OMIT			X							
ORGEN		X	X				X	X	X	
PLINE			X	X	X	X		X	X	
PLSTR			X	X	X	X		X	X	
PPNT			X	X	X	X		X	X	
PSGMT			X	X	X	X		X	X	
PTEXT			X		X		X	X	X	MVPOS
RDTRK										CNVRT
RESET		X	X							EXEC

(continued)

Table 1. GSP External References and Relationship (continued)

	External References									
GSP Routine Called	Flow Ctrl	Buffer Mgt	Key Tbl Mgt	Scaling	Scis-soring	Data Gen	Data Store	Update	RESET	Other
RQATN										CNVRT
RTBP										CNVRT
SDATM				X						
SGDSL				X						
STEOS			X				X		X	
STPOS			X	X	X	X		X	X	
TMDEV	X	X								ENATL TMGDS
TMGDS	X	X	X							
TMGSP										TMDEV

- Graphic terminal control block (GTMCB), which is constructed by INDEV for each 2250 identified by the user.

- Graphic data set control block (GDSCB), which is constructed by INGDS for each graphic data set (GDS) that is initialized for a 2250.

- Graphic attention control block (GACB), which is necessary for attention handling. Two of these are created, activated, and used for each 2250; one for the use of most GSP attention related subroutines and one for the Panic Key routine. The GACB is described in the publication IBM System/360 Operating System: Graphics Access Method, Program Logic Manual, Form Y27-7113. GSP appends eight bytes of information to the beginning of the GACB. These appendages are described in Appendix A.

- Output area control block (OACB), which is constructed to contain information regarding the graphic data output area (GDOA). Two are created per GDS, one for each GDOA.

- Attention level control block (ATLCB), which is required for attention handling and is constructed for each attention level specified by the user.

- Attention data entry queue (ADEQ), which is attached to the ATLCB and is used for attention information data. Several ADEQs follow each ATLCB.

Three tables are created by GSP routines. They are described in the section "Internal Routines." Tables used for controlling certain functions performed by GSP routines are:

- Buffer control table (BCT), which is initialized for use by Buffer Management when buffer sections are assigned or released. One BCT exists for each 2250.

- Flow control table (FCT), which is initialized for use by Flow Control Management in organizing and maintaining information about the regeneration sequence of GDSs. One FCT exists for each 2250.

- Key table, which is constructed by Key Table Management for each level 1 GDS using key and/or correlation facilities to contain keys and correlation values specified by the user.

In addition, there are several control blocks which are required for graphic programming services, and are described in detail in the publications: IBM System/360 Operating System: Graphic Programming Services for IBM 2250 Display Unit, Form C27-6909 and IBM System/360 Operating System: Graphics Access Method, Program Logic Manual, Form Y27-7113. These control blocks are:

- Data control block (DCB), which is created for each 2250 initialized.

- Data event control block (DECB), which is a parameter list used for overlapped input/output operations. That is, while the contents of one output area are being written to the buffer, another output area is being filled with data.

- Graphic data output area (GDOA), which is initialized for use in input/output operations. To facilitate input/output overlap, defined above, two of these GLOAs are acquired for each GDS.

- Communication area (COMAREA), which is related to attention handling. One COMAREA is required for each GACB to contain attention information when an attention occurs.

Finally, there is one control block for GSP which represents its own GDS. It is:

- System GDSCB, which is constructed for GSP, and is used by the light pen subroutines. Its format is the same as the GDSCB created for the user's GDS. It is always the first GDSCB created in the chain of GDSCBs.

Formats and Main Storage Arrangements

Detailed formats of all GSP control blocks except the BCT, FCT, and key table are contained in Appendix A. The BCT, FCT, and key table are described in detail in "Buffer Management," "Flow Control Management," and "Key Table Management" respectively.

Figure 2 depicts the subroutines that establish the control blocks, shows these blocks as they are arranged in main storage, and indicates the pointers contained in the various blocks.

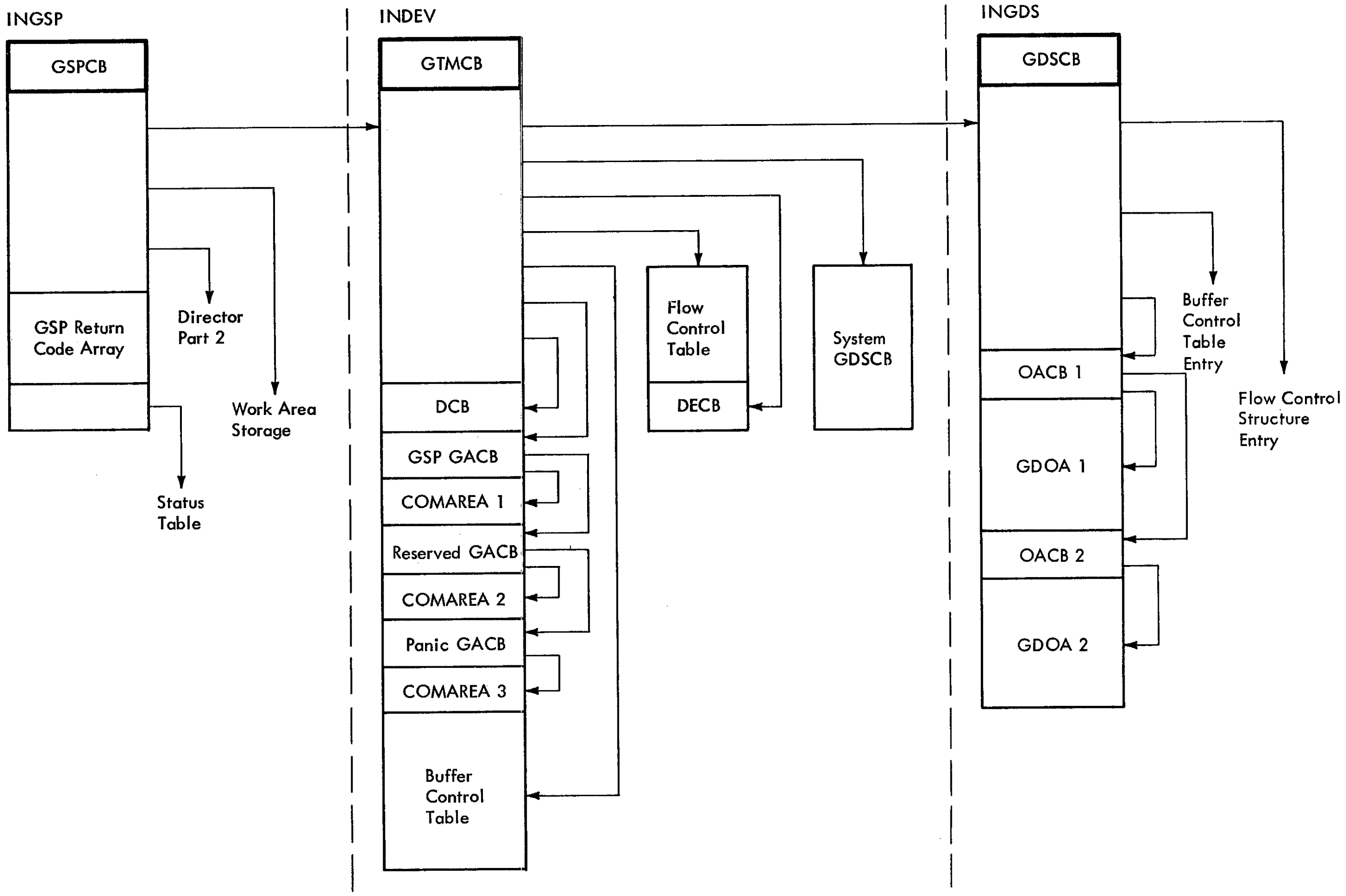


Figure 2. Control Blocks Created and Used by GSP

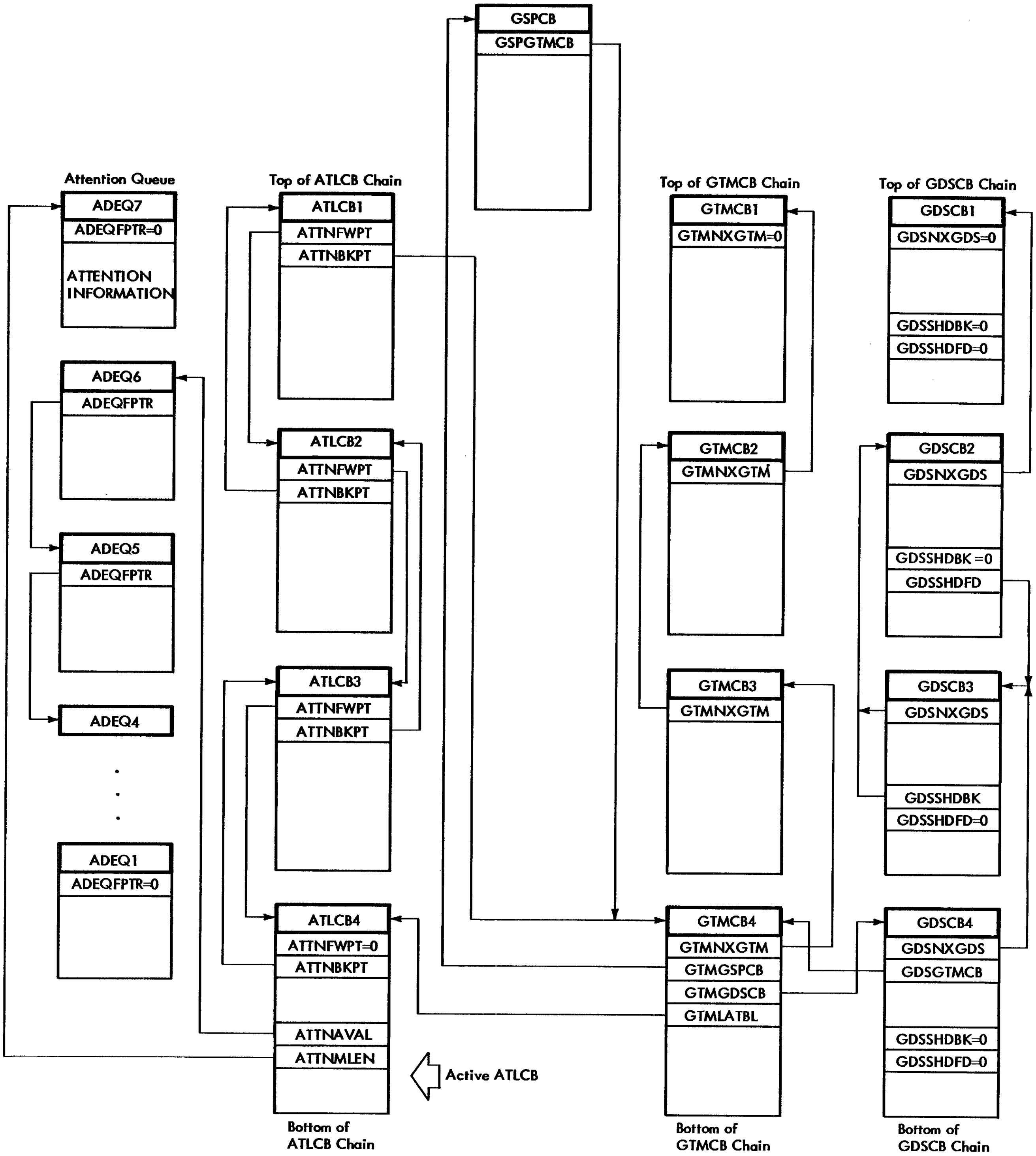


Figure 3. Relationship and Chaining of GSP Control Blocks

Chaining

Chaining of control blocks is accomplished by placing the appropriate addresses (or zero) in the fields designated as forward pointers and/or backward pointers. These fields are manipulated at the time a control block is created. An address in the field refers to the next control block of the same type, or to a related control block. A value of zero in the field indicates the end of a series of control blocks.

Figure 3 shows the relationship of the various GSP control blocks and how control blocks of the same type are chained.

The numbers 1 through 4 after GTMCB, GDSCB, and ATLCB represent four control blocks of each type in the order they were created. In all cases, the fourth control block is the most recently created one. GDSCB2 and GDSCB3 represent equivalent GDSs.

The numbers following ADEQ represent the data queues that are attached to each ATLCB. In this case, ADEQ7 already contains attention information. The remaining ADEQs are available to accept attention information.

Whenever a new control block is added to the GTMCB, GDSCB, or ATLCB chain, it is inserted at the bottom of the chain, and the pointers to the bottom of the chain are updated.

OPERATIONS PERFORMED BY ALL GSP ROUTINES

Certain operations are performed by all GSP subroutines and related internal routines. These operations are described in the paragraphs that follow, rather than being included in the description of each routine.

Initialization

All GSP routines perform certain preparatory functions before performing the main processing function. These preparatory functions, called <u>initialization</u>, consist of:

- Saving register contents upon entry to the routine.

- Establishing addressability of the routine by defining a base register.

- Establishing addressability of one or more of the four major control blocks (GSPCB, GTMCB, GDSCB, and ATLCB) by the use of DSECTs and the USING statement.

- Setting the return code array, which is used by every subroutine to pass information to the user, to zero before processing is begun by the routine. This array, called the GSPARRAY, is located in the GSPCB. This function is not performed by the internal routines.

- Chaining the register save areas together.

- Testing parameters for validity (see "Testing Parameters").

All these functions, linkages, and register usage are in accordance with standard IBM System/360 Operating System conventions.

Testing Parameters

All GSP subroutines test parameters that have been passed to them before any processing is done. Any parameter error encountered during these tests causes control to be returned to the calling program with the appropriate return code and additional information, if applicable, in the return code array. For further information about return codes, see "Passing Return Codes" in this section.

The validity of "gspname", "devicename", "gdsname", or "attnlevel" is determined by accessing the actual value of the specified variable. This value should point to one of four control blocks (GSPCB, GTMCB, GDSCB, or ATLCB). Figure 4 depicts the method of accessing control blocks. Each control block contains an identification field, which is tested to determine the validity and type of the control block.

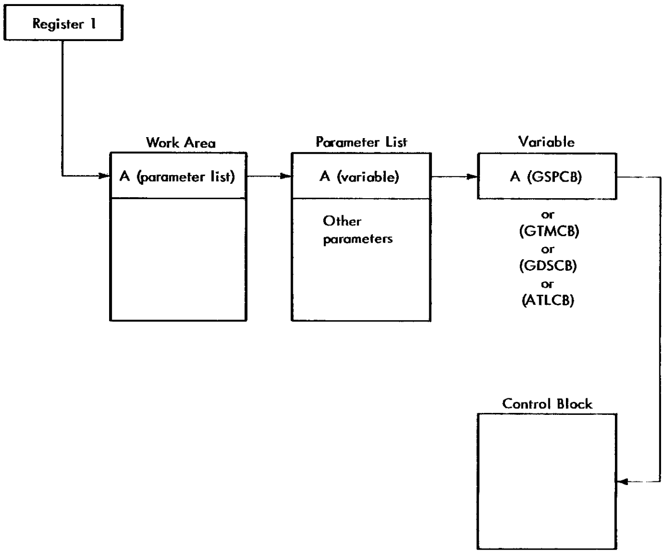


Figure 4. Accessing a Control Block

Passing Return Codes

Before returning control to the user's program, each subroutine makes a return code available to that program. These return codes and additional information are always placed in the GSPARRAY field located in the GSPCB. The GSP return code structure is described in detail in Appendix A.

Return codes and their meanings are described in the publication IBM System/360 Operating System: Graphic Programming Services for FORTRAN IV, Form C27-6932.

Internal routines in GSP and other externally referred to routines used by GSP subroutines pass a return code in register 15. A return code other than zero indicates that an error was encountered during processing.

After an internal or externally referred to routine has returned control to the GSP subroutine, any error code in register 15 is translated into a GSP return code, which is placed in the GSPARRAY. In many cases, the internal routines themselves place the appropriate return code in the GSPARRAY. If this is done, the GSP subroutine usually returns control to the calling program after any resetting function required has been accomplished. An exception occurs when image generation is to continue after the scissoring option has been applied.

MACRO INSTRUCTIONS USED BY GSP

The routines described in this publication use operating system macro instructions and Graphics Access Method (GAM) macro instructions.

Operating System Macro Instructions

Whenever the LOAD, LINK, CALL, SAVE, DELETE, GETMAIN, FREEMAIN, WAIT, DCB, OPEN, or CLOSE macro instruction is issued, control is given directly to the operating system to perform the required operation. The DCB, OPEN, and CLOSE macro instructions also cause control to be passed to the applicable GAM routines from the operating system. Whenever the RETURN macro instruction is issued, control is returned to the calling program. If an ABEND macro instruction is issued, the task is abnormally terminated. For a description of these macro instructions, refer to the publication IBM System/360 Operating System: Supervisor and Data Management Macro Instructions, Form C28-6647.

Graphics Access Method (GAM) Macro Instructions

Whenever the ASGNBFR, RLSEBFR, GREAD, GWRITE, GREADR, GCNTRL, SAEC, SPAR, ATTNINQ, or DAR macro instruction is issued, control is given directly to the GAM routines to perform the desired operation. For a description of these macro instructions and the graphic orders used by GSP, refer to the publication IBM System/360 Operating System: Graphic Programming Services for IBM 2250 Display Unit, Form C27-6909. The GAM routines are described in the publication IBM System/360 Operating System: Graphics Access Method, Program Logic Manual, Form Y27-7113.

The detailed organization of all the routines (modules) in GSP is described in this section. Each module description includes the following information:

- Module name (identification) for each routine is included in the heading for that particular routine.

- Chart identification(s), where flowcharts are included for the module. These flowcharts appear in the next section of this publication.

- Function, which states the routine's overall purpose(s).

- Entry point(s) and from which routine(s) it is entered.

- Exit, which states where the routine returns control.

- Input, which defines the information passed by the user or another routine to be used by this routine.

- Output that is passed or made available to the caller.

- Operation, which describes the manner in which this routine performs its function.

GSP subroutines and functions are divided into several areas according to the operations they perform. These areas are:

1. Initiation and termination
2. Generation and modification of a display
3. Keyboard input and buffer data analysis
4. Communication between the operator and the program
5. Testing the status of subroutines
6. Producing characters without using the 2250 character generator
7. Cirect generation of graphic orders
8. Converting coordinates

This publication is organized to describe the subroutines in this manner. In addition, the internal routines for GSP, which are not directly invoked by the user's program, are described under "Internal Routines."

INITIATION AND TERMINATION OF GSP

Initiation and termination procedures for GSP are performed by three resident modules, six subroutines, an internal routine, and a status table. The resident modules function when a GSP subroutine is called. The six subroutines serve to initialize and terminate the use of GSP, a 2250, and a GDS. The internal routine, Director Part 2, controls entry into most of the subroutines by using the status table.

Figure 5 depicts the linkage from the user's program to the initiation and termination subroutines. The figure includes the resident modules and shows the relationship of all routines or modules that function during initiation and termination procedures. It also shows the routines that are loaded at initiation time.

All subroutines that are called in the user's program except INGSP and TMGSP receive control via Director Part 1 and Director Part 2. After all processing is complete, control is returned to Director Part 2 via the same path through which it was passed. Director Part 2 then returns control to the next sequential instruction in the user's program. This is explained in the section "Control Flow Among GSP Routines."

RESIDENT MODULES

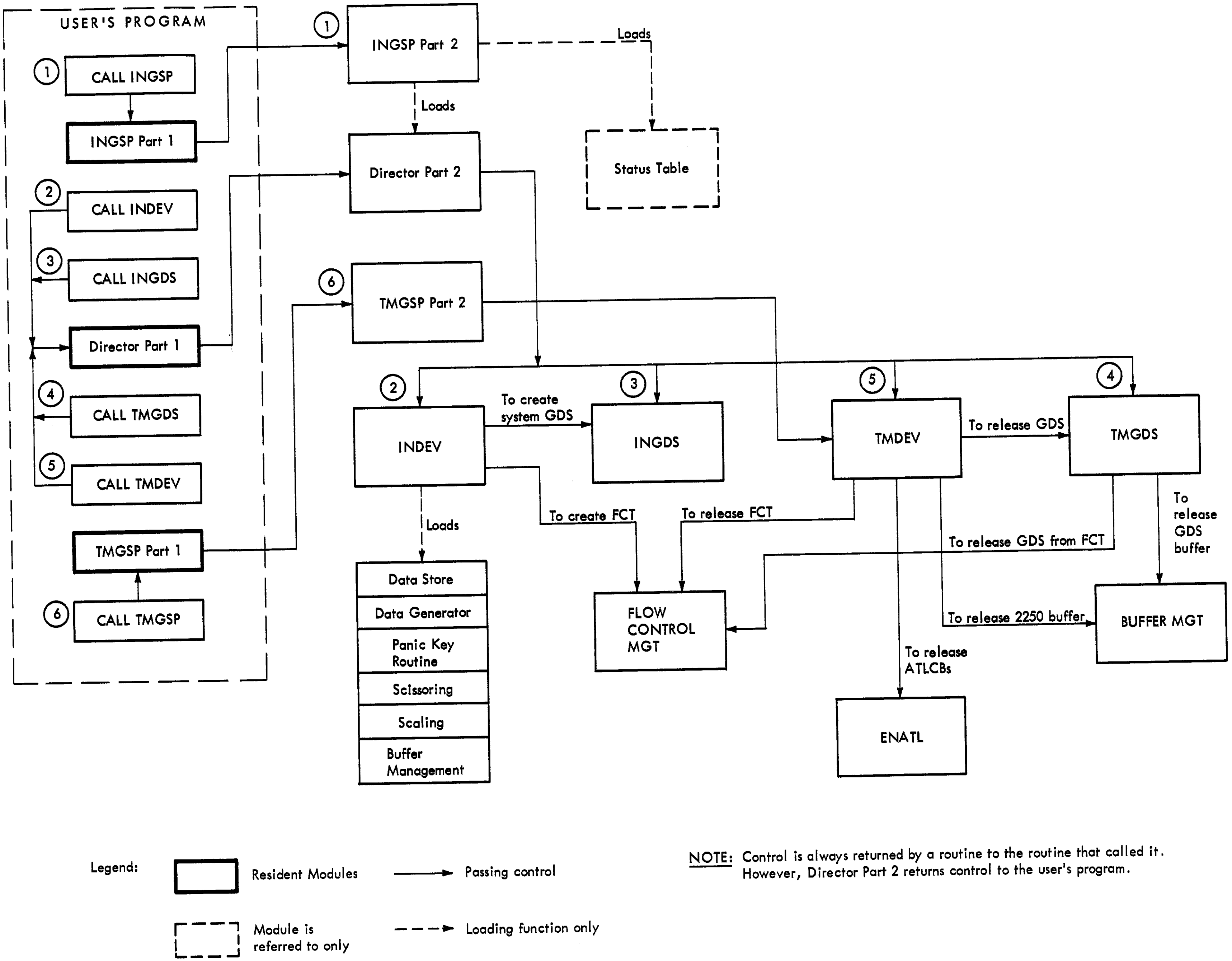
To perform its total function, each resident module invokes Part 2 of its counterpart, which is described under "Initiation and Termination Subroutines" or "Internal Director" in this section.

The resident modules consist of:

- INGSP Part 1, which initializes the use of GSP.

- Director Part 1, which passes the entry number of the called subroutine to Director Part 2, so that the location and status of the subroutine can be determined.

- TMGSP Part 1, which terminates the use of GSP.



NOTE: Control is always returned by a routine to the routine that called it. However, Director Part 2 returns control to the user's program.

Figure 5. Linkage from User's Program to Initiation and Termination Subroutines

INGSP Part 1 (Module Name IHCGSP01)

Chart: None

Function: Obtains storage for the construction cf the GSPCB and for save and work areas. Links to INGSP Part 2. Upon return of control from INGSP Part 2, frees work/save area obtained for use of INGSP. Retains other storage (GSPCB and work area) that it had obtained for use by other GSP routines.

Entry: INGSP from a call to the INGSP subroutine.

Exit: To the calling program.

Input: In register 1, the address of a parameter list specified by the user in the call to the INGSP subroutine.

Output: To INGSP Part 2, in register 1, the address of the first byte of the main storage area obtained by this routine. The first word in this area contains the address of the parameter list received as input. To the calling program, the address of the GSPCB which was placed in the user-specified variable by INGSP Part 2.

Operation: See "Function" above.

Director Part 1 (Module Name IHCGSP03)

Chart: None

Function: Locates Director Part 2, and branches to it to accomplish entry into the desired subroutine. Once Director Part 1 relinquishes control, control is not returned to it until another GSP subroutine is called.

Entry: From a call to any GSP subroutine, except INGSP or TMGSP. Entry is made into a table of entries, called the GSP directory, which contains branch instructions to the main body of executable instructions in Director Part 1.

Exit: To Director Part 2.

Input: In register 1, the address of a parameter list specified by the user in a call to a GSP subroutine (other than INGSP or TMGSP). The first word of the parameter list points to a variable that contains the address of a GSP control block (GSPCB, GTMCB, GDSCB, or ATLCB) as shown in Figure 4. This first word is the only parameter used by Director Part 1.

Output: In register 1, the address of the parameter list received as input. In register 2, the address of the GSPCB. In register 3, the entry number of the requested GSP subroutine.

Operation: The GSP directory is used to resolve the actual subroutine entry requested by this call. This directory contains the name of every GSP subroutine (no internal routines) except INGSP and TMGSP. Each name is identified as an entry, and is followed by a branch instruction that passes control to the executable instructions in Director Part 1. Effectively, the GSP directory causes the address of the location of the requested subroutine name in the table to be placed in register 15. This address is later used to determine the entry number that is passed to Director Part 2. (See "Director Part 2" and "Status Table.")

Once entry is made to the main executable instructions in Director Part 1, the first parameter in the input list is accessed, the address of the specified control block is found, and the control block is tested for validity and type (i.e., GSPCB, GTMCB, GDSCB, or ATLCB). Figure 4 shows how the control block is accessed. Depending on the type of control block located, the address of the GSPCB is obtained by accessing pointers in the appropriate control block fields. If the first parameter is not a GSP control block, the value of this parameter is set to zero, and control is returned immediately to the calling program. If the first parameter is valid, the entry number of the subroutine is computed. This entry number is used by Director Part 2 to locate the GSP subroutine entry in the status table (described in this section under "Status Table"). Control is then passed to Director Part 2.

TMGSP Part 1 (Module Name IHCGSP02)

Chart: None

Function: Links to TMGSP Part 2 to perform termination functions. Upon return of control, it deletes any stroke tables that have been loaded, the status table, and Director Part 2. It also frees main storage used for the GSPCB and the work/save area.

Entry: TMGSP from a call to the TMGSP subroutine.

Exit: To the calling program.

Input: In register 1, the address of a parameter list that contains the address of the variable that in turn contains the address of the GSPCB.

Output: To TMGSP Part 2, in register 6, the address of the GSPCB. To the calling program, there is no output.

Operation: TMGSP Part 1 tests the GSPCB for validity. If it is valid, the routine proceeds as described under "Function" above. If the GSPCB is invalid, the value of the parameter is set to zero, and control is returned immediately to the calling program.

INTERNAL DIRECTOR

The internal director, which is divided into two parts, Director Part 1 and Director Part 2, handles the function of passing control to the proper GSP subroutine whenever one is called. Director Part 1 is described in this section under "Resident Modules." Director Part 2 works in conjunction with the status table also described in this section.

Director Part 2 (Module Name IFFAHA11)

Chart: None

Function: Performs interface functions between the calling program and GSP subroutines as follows:

- Determines via information in the status table whether the requested GSP subroutine should be linked, called, or loaded and called, and passes control to the subroutine. If the subroutine is loaded by Director Part 2, it places the entry point address in the status table.
- Passes the address of a work area to the subroutine (see "Output").
- Indicates upon entry to a GSP subroutine that GSP is busy, and upon return of control from the subroutine, that GSP is not busy. The GSPBUSYS field in the GSPCB is used for this purpose.
- Upon return of control from any GSP subroutine, the GSPDUMP and GSPABEND fields in the GSPCB are appropriately set each time by Director Part 2. It also tests the return code in the GSPARRAY field. If this return code is equal to or greater than the absolute value of the null variable, the routine issues the ABEND macro instruction with appropriate completion code. If the null variable contains a value of zero, this routine treats it as a value of one. If the null variable contains a value of six or greater, no ABEND macro instruction is issued.
- Returns control to the next sequential instruction in the user's program if processing is to continue after the requested GSP subroutine completed its execution.

Entry: IFFAHA11 from Director Part 1.

Exit: Normally, to the user's program. However, under certain conditions the program is abnormally terminated.

Input: In register 1, the address of the parameter list specified by the user in the call to a GSP subroutine. In register 2, the address of the GSPCB. In register 3, the entry number of the requested GSP subroutine.

Output: In register 1, the address of a work area. This work area contains the address of the parameter list and the address of the return code array (GSPARRAY) as shown in Figure 6.

If the ABEND macro instruction is issued upon return of control from a GSP subroutine, a dump will be furnished when the null variable contains a negative value. No dump is furnished if the null variable contains a positive value. Register 2 contains the address of the status table entry last invoked. Register 3 contains the address of the GSPARRAY field in the GSPCB.

Operation: See "Function" above.

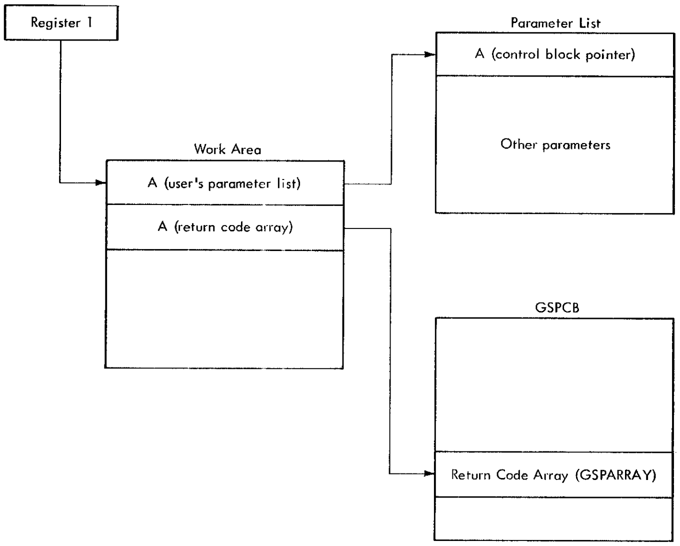

Register 1
Parameter List
A (control block pointer)
Other parameters
Work Area
A (user's parameter list)
A (return code array)
GSPCB
Return Code Array (GSPARRAY)

Figure 6. Output to GSP Subroutines from Director Part 2

Status Table (Module Name IFFAHA12)

The status table is an "only loadable" module residing in the SYS1.LINKLIB and loaded by INGSP Part 2. Whenever a GSP subroutine is called, Director Part 2 refers to it as explained earlier.

The status table consists of 12-byte entries for each GSP subroutine and internal routine. These 12 bytes contain three fields of information about each routine, as follows:

+0 Address of the entry point for the routine, which is zero initially. When the module is brought into storage by a LOAD macro instruction, the entry point address is placed in this field. If a module is linked to, this field remains zero.

+4 Module identification which consists of the last four characters of the module name. For example, "AA03" would identify INDEV (module name IFFAAA03).

+8 Flag in the first byte that indicates the proper linkage for this module, as follows:
X'01' - LINK
X'02' - LOAD and CALL

The remaining three bytes contain zeros.

Note: The linkage (method of entry) for each GSP module as contained in the status table is given in the module directory in Appendix B.

Typical status table entries are shown in Figure 7, which depicts the entries for ICURS and SDATM.

<------------- 4 bytes ----------------->

A(EP of ICURS)	
C'DA03'	
X'02'	000000
Zero (module is linked to)	
C'EA01'	
X'01'	000000

Figure 7. Status Table Entries

INITIATION AND TERMINATION SUBROUTINES

These subroutines create and delete control blocks, tables, and work areas used throughout the processing of the program for communication among the subroutines. Detailed descriptions and formats of the control blocks are contained in Appendix A.

The subroutines described herein are:

- INGSP Part 2, which completes initialization of GSP.
- INDEV, which establishes communication links between the 2250 and GSP.
- INGDS, which creates a GDS and associates it with a 2250.
- TMGDS, which terminates the use of a particular GDS.
- TMDEV, which terminates the use of a 2250 and all GDSs associated with it.
- TMGSP Part 2, which terminates the use of GSP in conjunction with TMGSP Part 1.

INGSP Part 2 (Module Name IFFAAA01)

Chart: None

Function: Initializes the GSPCB. As part of the initialization, the address of the null variable is placed in the GSPCB and the status table and Director Part 2 are loaded.

Entry: IFFAAA01 from INGSP Part 1.

Exit: To the calling program.

Input: In register 1, the address of the main storage area where the GSPCB is to be built. The first word in this area contains the address of the parameter list specified by the user in the call to the INGSP subroutine.

Output: The GSPCB is initialized and its address is placed in the user-specified variable ("gspname") which is the first parameter.

Operation: The address of the null variable is obtained from the parameter list and placed in the GSPCB. If the null variable is not specified, the value of the user's variable that points to the GDSCB is set to zero and control is returned immediately to the calling program.

If more than two parameters are specified, the additional parameters are ignored.

Next, the GSPCB address (in register 1) is placed in the user-specified variable, and the fields in that GSPCB are filled in. The status table and Director Part 2 are then loaded and their addresses are placed in the GSPCB. The address of the work area is determined and placed in the GSPCB. All GSPCB fields not initialized by this time are cleared and control is returned to INGSP Part 1.

INDEV (Module Name IFFAAA03)

Chart: AA

Function: Establishes communication links between GSP and a 2250. To do so, it:

- Creates a GTMCB.
- Creates a DCB to be used by GSP.
- Creates and activates a GSP graphic attention control block (GACB) and its associated communication area (COMAREA).
- Creates and activates a panic GACB and its associated COMAREA.
- Obtains and clears main storage for the buffer control table (BCT).
- Initiates construction of the system GDSCB.
- Initiates construction of the flow control table (FCT) and flow control structure.
- Initiates construction of the DECB used for input/output overlap.
- Loads the following routines: Panic Key (CANCEL Key), Scissoring, Scaling, Buffer Management, Data Generation, and Data Store.
- Unlocks the alphameric keyboard.

All these control blocks and tables are associated with the 2250 specified by the user as the "unit" parameter in the call to INDEV.

Entry: IFFAAA03 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The address of the GTMCB is placed in the address of the user-specified "devicename" parameter. The various control blocks and tables associated with the 2250 are established (see "Function" above).

Operation: If any function is not performed successfully throughout processing by INDEV, all functions previously performed by this execution of INDEV are negated, an appropriate return code is placed in the GSPARRAY, and control is returned to the calling program.

INDEV first tests all parameters for validity, except for the "gdoalength" parameter. If these parameters are valid, the amount of main storage necessary for the GTMCB, DCB, BCT, GSP GACB, panic GACB, and their associated COMAREAs is computed and obtained. The starting address of the main storage obtained (i.e., the address of the GTMCB) is placed in both the GTMCB and the GSPCB.

The value used to form the data definition name (ddname) of the data definition (DD) job control statement is placed in the "ddname" field of the DCB. The address of the DCB is then placed in the GTMCB and the DCB is opened.

Once the DCB is open, the bit configurations associated with the features available for the designated 2250 and the address of the 2250 are interpreted from the unit control block (UCB) and placed in the GTMCB. Next, the BCT is cleared and its address is placed in the GTMCB.

At this point, INGDS is called to initialize the system GDSCB and to place its address in the GTMCB. Flow Control Management is called to establish the flow control table and flow control structure.

Next, the GSP GACB, the panic GACB, and their associated COMAREAs are created, and the GSP GACB address is moved into the GTMCB. The length specified for GDOAs associated with this GTMCB is determined and placed in the GTMGDOAL field of the GTMCB. Then, the alphameric keyboard is unlocked.

The GTMCB is added to the chain of GTMCBs as described under "Chaining" in the "Introduction." The address of the new GTMCB is placed in the variable specified by the "devicename" parameter.

Next, the five internal routines and the Panic Key routine mentioned under "Function" are loaded by this routine, and their entry point addresses are placed in the appropriate fields in the status table. Control is then returned to the calling program.

INGDS (Module Name IFFAAA05)

Chart: AB

Function: Creates a GDS and associates it with a 2250. To do so, it:

- Establishes a GDSCB for each GDS, and each equivalent GDS.
- Establishes two OACBs for each GDSCB (one for each GDOA).

- Establishes two GDOAs for each GDSCB (for input/output overlap).

Entry: IFFAAA05 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The address of the GDSCB is placed in the corresponding variable specified as the "gdsname" parameter. The various control blocks and output areas associated with each specified GDS are established (see "Function" above).

Operation: All parameters are processed and checked for validity. If the "gdoalength" parameter is not specified, the length of the GDOA is found in the GTMGDOAL field of the GTMCB. If this value is less than 128, it is rounded to 128. If the value is greater than 128, it is rounded to the next higher multiple of 256. All information obtained from the processing of these parameters is saved for further use.

Main storage is obtained for the GDSCB, two OACBs and two GDOAs. Additional storage is obtained if equivalent GDSs are specified. If insufficient storage is available at any time, all storage previously obtained is freed, and control is returned to the calling program with an appropriate return code.

Once storage has been obtained, a skeleton GDSCB is positioned at the beginning of the main storage area just obtained, and its own address, the address of the GTMCB, the GDS level, and the GDOA length are placed in that GDSCB. Associated OACBs and GDOAs are then positioned behind the skeleton GDSCB as shown in Figure 2. Format of the OACB is shown in Appendix A.

If equivalent (shared) GDSs have been specified, the procedure described in this paragraph is repeated until all of the equivalent GDSs have been defined. Storage is obtained individually for each equivalent GDS. The address of the GDSCB for the shared GDS is placed in its associated "gdsname" variable. A flag is set in the previously created GDSCB to designate it as belonging to an equivalent GDS group. The GDSCB is duplicated at each new address in main storage for each equivalent GDS, and the GDSCBs are chained together using the GDSSHDFD and GDSSHDBK fields as shown in Figure 3. OACBs and the GDOAs associated with each GDSCB are initialized and the GDSCB is added to the chain.

When all equivalent GDSs have been processed, the address of the first GDSCB is placed in its associated "gdsname" variable. Control is returned to the calling program.

TMGDS (Module Name IFFAAA06)

Chart: AC

Function: Terminates the use of the designated GDS and all equivalent GDSs by freeing all main and buffer storage associated with the control blocks and tables used by those GDSs.

Entry: IFFAAA06 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: None

Operation: If a key table exists for this GDS, Key Table Management is called to delete the key table. Then, if a flow control entry exists, Flow Control Management is called to delete the specified GDS from the flow control structure. Buffer Management is then called to free the buffer storage assigned to this GDS.

The GDSCB is removed from the chain of GDSCBs by manipulating pointers. (See "Chaining" under "Introduction.") Its associated storage -- including OACBs and GDOAs -- is then freed. If this is an equivalent GDS, the forward and backward pointers are saved before freeing storage.

Any equivalent (shared) GDSCBs are now removed from the chain of GDSCBs in the same manner as above. The forward pointer is used to determine the next GDSCB to be deleted. When all forward shared GDSCBs are deleted, the same process is repeated with backward pointers. Then, each time an equivalent GDSCB is removed from the chain, the key table for that GDS is deleted, if present, and storage occupied by the GDSCB and its OACBs and GDOAs is freed.

When all equivalent GDSCBs have been deleted from the chain and all storage has been freed, control is returned to the calling program.

TMDEV (Module Name IFFAAA04)

Chart: AD

Function: Terminates the use of the designated 2250 by freeing all main and buffer storage associated with the control blocks and tables used by that 2250.

Entry: IFFAAA04 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: None

Operation: Flow Control Management is called to stop regeneration and to free the storage assigned to the flow control table. Buffer Management is then called to free all buffer assigned to any GDSs associated with the specified 2250.

Once all buffer has been freed, the GTMGDSCB field in the GTMCB is inspected for any open GDSs associated with that 2250. As an open GDS is encountered, TMGDS is called to terminate its use. The GTMLATBL field in the GTMCB is then inspected for any attention levels associated with that 2250. As an attention level is encountered, ENATL is called to terminate its use.

At this point, the GTMCB is removed from the chain of GTMCBs. Finally, the DCB is closed, and all main storage associated with the GTMCB, DCB, BCT, the GACBs and their COMAREAs is freed.

TMGSP Part 2 (Module Name IFFAAA02)

Chart: None

Function: Terminates the use of GSP by:

- Calling TMDEV to terminate the use of all associated 2250s.
- Deleting all GSP subroutines and internal routines that were previously loaded.

Entry: IFFAAA02 from TMGSP Part 1.

Exit: To TMGSP Part 1.

Input: In register 6, the address of the GSPCB.

Output: None

Operation: See "Function" above.

GENERATION AND MODIFICATION OF A DISPLAY

Numerous subroutines and internal routines work together to produce a display on the 2250, and to implement the update facility available in GSP. This area includes the option definition, image generation, image identification, and image control subroutines. The internal routines used by these subroutines are listed as "External References" in Table 1 and are described under "Internal Routines."

OPTION DEFINITION SUBROUTINES (MODULE NAMES IFFAEA01, IFFAEA02, IFFAEA03, IFFAEA04, IFFAEA06, AND IFFAEA07)

These subroutines define the characteristics of the input data to be supplied to image generation subroutines, the output to be generated by the Data Generator routine, and the characteristics of the display to be produced. Each one of these subroutines sets appropriate fields in the GDSCB. Locations of fields and bit settings are given with the GDSCB format description in Appendix A. Subroutine functions are as follows:

- SDATM (module name IFFAEA01) sets bits in the GDSDATMD field to indicate the type (real or integer) and form (absolute or incremental) of data that the user will provide as input to GSP subroutines that refer to the specified GDS. If the type and form of data is changed by a call to SDATM, the data limits are converted to the new data type as necessary.
- SGRAM (module name IFFAEA02) sets bits in the GDSCURMD field to define the form of output (absolute, incremental, or optimized) that will be produced by the Data Generator routine for the specified GDS.
- SCHAM (module name IFFAEA07) sets bits in the GDSCHARS field to define the size of characters to be displayed for the GDS and whether or not those characters are to be protected or unprotected.
- SDATL (module name IFFAEA03) places values in the GDSUVLLL, GDSVVLLL, GDSUVLUR, and GDSVVLUR fields that represent the user-specified x- and y-coordinates for the lower-left and upper-right limits of input data. Upon successful completion of its function, this routine sets the GDSFLGS2 field to indicate that the SDATL subroutine was called.
- SSCIS (module name IFFAEA06) sets bits in the GDSSCISS field to indicate whether or not scissoring is to be performed, where the image is to be truncated, and whether or not image generation is to be continued after scissoring.
- SGDSL (module name IFFAEA04) places integer values in the GDSXVILL, GDSYVILL, GDSXVIUR, and GDSYVIUR fields that represent the user-specified x- and y-coordinates for the lower-left and upper-right corners of the GDS. This routine also places real values in the GDSXVRLL, GDSYVRLL, GDSXVRUR, and

GDSYVRUR fields that represent the same user-specified coordinates. SGDSL tests the GDSFLGS2 field to determine if SDATL has been called. If not, this routine places the values for data limits in the appropriate fields (see SDATL above) using the same parameters passed for GDS limits. If no data has been generated for the GDS, the GDSXCURR, GDSYCURR, GDSXLAST, and GDSYLAST fields are set equal to the lower-left corner of the GDS if 0,0 is neither within nor on the boundary of the GDS.

The entry point in each routine is the same as its module name. Entry is from Director Part 2. Exit is back to Director Part 2. During the processing, appropriate return codes are placed in the GSPARRAY field of the GSPCB.

Input to each of these subroutines is described under "Input to Many GSP Subroutines."

Each routine tests the parameters for validity. If an invalid parameter is passed, a return code is set, and the fields that would normally be set according to the specified parameters remain as they were prior to this execution of the routine.

IMAGE GENERATION SUBROUTINES

These subroutines create elements necessary for displaying images on the 2250 screen. They analyze the input passed to them, and call upon various internal routines to perform the functions necessary for producing appropriate graphic orders and data, and the associated control functions for GSP.

The subroutines described in this section are:

- STPOS/MVPOS, which moves the beam in blanked mode to a specified position on the 2250 screen.
- PLINE/PPNT, which plots lines or points as specified.
- PSGMT, which plots one or several line segments.
- PTEXT, which causes characters to be displayed using the character generator feature of the 2250.
- STEOS, which generates an end-order-sequence order.

Linkage for Image Generation Subroutines

Figure 8 depicts the linkage for the image generation subroutines, and all external routines referred to by them. The figure shows the path followed when a call is issued in the user's program for each of the subroutines.

All subroutines receive control via Director Part 1 and Director Part 2. Whenever control is passed to another routine, the called routine completes its processing before control is returned to the calling routine (calling program). The use of external references may vary from one execution to another, since their use is contingent on the results of various tests performed by the image generation subroutines. When processing is complete, control is returned via the same path through which it was passed, as explained under "Control Flow Among GSP Routines."

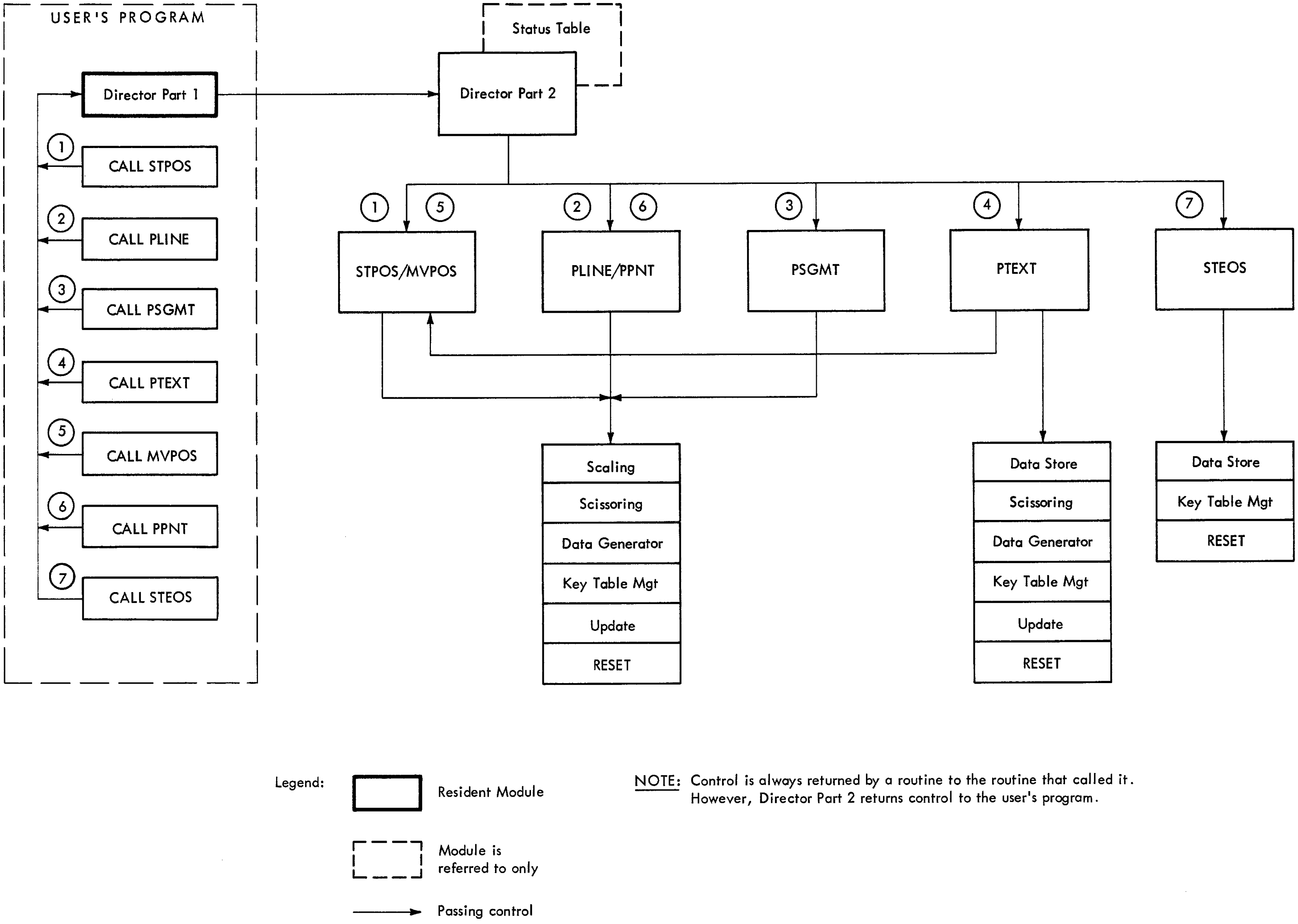
Output from Image Generation Subroutines

Output from each image generation subroutine <u>except STEOS</u> consists of calls to routines that scale, scissor, generate, and store graphic orders and data for displaying the desired images. When required, a key is built, a key and/or correlation value is placed in the key table via Key Table Management, and the key value is placed in the variable specified by the user as the "key" parameter.

General Operation of Image Generation Subroutines

Part of the operation for each of the subroutines <u>except STEOS</u> is the same. Each subroutine builds a parameter list that it will pass as input to Scaling, Scissoring, and Data Generator. Bits are set in this parameter list to indicate whether the element is a new or update element, is keyed or correlated, or is in include or omit status. Also placed in this parameter list is the correlation value (if passed) and the address of the variable into which the key is to be placed before exit from the subroutine.

Each image generation subroutine saves the current raster unit coordinate positions of the 2250 beam, and the raster unit coordinate values generated from the preceding call to an image generation subroutine. (These values are obtained from the GDSXCURR, GDSYCURR, GDSXLAST, and GDSYLAST fields in the GDSCB.) If the



NOTE: Control is always returned by a routine to the routine that called it. However, Director Part 2 returns control to the user's program.

Figure 8. Linkage from User's Program to Image Generation Subroutines

subroutine fails to perform its prescribed functions for any reason (e.g., update element is too large), these saved coordinate values are placed back in their respective fields in the GDSCB, and the GDS is reset, thus returning the program to the condition that existed prior to the call to the image generation subroutine.

Scaling and Scissoring are called as necessary to determine the raster unit coordinates for each new beam position (beginning and end points in the case of PSGMT). Scissoring updates the GDSXCURR, GDSYCURR, GDSXLAST, and GDSYLAST fields of the GDSCB appropriately. If no Scissoring is requested, the image generation subroutine updates these fields after Scaling has been performed.

If the element to be produced by an image generation subroutine is to be part of a buffer subroutine, the image generation subroutine changes the contents of the GDSDATMD field in the GDSCB to reflect incremental input. Upon exit from the image generation subroutine, this field is restored to reflect the input mode that existed prior to the time the image generation subroutine was called.

STPOS/MVPOS (Module Names IFFAFA04 and IFFAFA17)

Charts: BA, BB

Function: Causes graphic orders and data to be generated for moving the 2250 beam in blanked mode to a specified position on the screen.

Entry: IFFAFA04 for STPOS, and IFFAFA17 for MVPOS, both from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: See "Output From Image Generation Subroutines."

Operation: On entry, a switch is set designating whether the call was to MVPOS or STPOS. This switch is used in determining if the data (input) mode and graphic (output) mode set in the GDSCB are to be overridden (i.e., forced absolute input and output for STPOS). STPOS/MVPOS then builds a parameter list that it will pass as input to the Scaling, Scissoring, and Data Generator routines, and saves the current raster unit coordinate positions of the 2250 beam (see "General Operation of Image Generation Subroutines").

Scaling and Scissoring are called as necessary to determine the new beam position coordinates. (See "General Operation of Image Generation Subroutines.") Data Generator is called to produce the element for the GDOA associated with the GDS identified in the call to the STPOS/MVPOS subroutine, and to complete the key (if any) that is to be associated with that element. If the element is keyed, and it is not an update element, Key Table Management is called to insert the key in the key table.

If the element is an update element (i.e., is to be substituted for a previously created element), Key Table Management is called to determine the validity of any key or correlation value passed by the user in the call to the STPOS or MVPOS subroutine. Update is called to create a temporary GDOA. The update element is generated as described in the preceding paragraph. Once the update element has been generated, Update is recalled to substitute this new element for the previously created element (identified by the key or correlation value or both).

PLINE/PPNT (Module Names IFFAFA01 and IFFAFA16)

Charts: BA, BC

Function: Causes graphic orders and data to be generated for producing one or more lines or points at specified positions on the 2250 screen.

Entry: IFFAFA01 for PLINE, and IFFAFA16 for PPNT, both entered from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: See "Output From Image Generation Subroutines."

Operation: Is the same as described for STPOS/MVPOS, except as follows:

- An entry switch set in the work area indicates if entry was from PLINE or PPNT. If entry was from PLINE, an element for displaying lines is generated. If entry was from PPNT, an element for displaying points is generated.
- The routine plots a single line or point or multiple lines or points. For multiple input, new beam positions are obtained by indexing through the user's array of input coordinates or by adding a user-specified increment to the absolute position of the last line or point.

PSGMT (Module Name IFFAFA02)

Charts: BA, BC

Function: Causes graphic orders and data to be generated for producing line segments on the 2250 screen.

Entry: IFFAFA02 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: See "Output From Image Generation Subroutines."

Operation: Upon entry, PSGMT builds a parameter list that it will pass as input to Scaling, Scissoring, Data Generator, and Update, and saves the current raster unit coordinate positions of the 2250 beam (see "General Operation of Image Generation Subroutines").

Scaling and Scissoring are called as necessary to determine the raster unit coordinates for the starting and ending beam positions of each line segment to be produced (again, see "General Operation of Image Generation Subroutines"). Bits are set to unblank the beam for movement between the starting and end positions of each line segment, and to blank the beam for movement between the end position of one segment and the starting position of the next segment.

Scaling and Scissoring are entered twice for each line segment to be produced. For multiple line segments, depending upon the arguments specified in the call to the PSGMT subroutine, new beam positions for each segment are obtained by indexing through the user's array of input coordinates or by adding a user-specified increment to the original value that was passed.

If the input data mode is incremental, the starting positions of all line segments are determined by adding the start increments to the previous starting positions. The end position of the first line segment is determined by adding the end increment to the first starting position of the first line segment. The end positions of all other segments are determined by adding the end increments to the previous end positions.

Data Generator is called as many times as necessary to produce the orders and data for displaying each desired line segment. Data Store is called to place these orders and data in the GDOA (or buffer on GDOA overflow) associated with the GDS identified in the call to the PSGMT subroutine.

After Scaling and Scissoring, Data Generator and Data Store are entered twice for each line segment to be produced. If the element is for a level 1 GDS and is keyed or correlated, a key is structured if it is not an update element, and Key Table Management is used to insert the key in the key table.

If the element to be created is an update element, Key Table Management is called to verify the validity of any key or correlation value passed by the user. Update is called to create a temporary GDOA. The update element is generated as described above. Once the update element has been generated, Update is recalled to substitute this new element for the previously created element (identified by the key or correlation value, or both).

PTEXT (Module Name IFFAFA03)

Charts: BA, BD

Function: Causes graphic orders and data to be generated for producing the text at specified positions on the 2250 screen using the character generator.

Entry: IFFAFA03 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: See "Output From Image Generation Subroutines."

Operation: If the element is keyed or correlated, Data Store is called to set up the include/omit structure for the element. If positioning is required (i.e., the "xcoor" and "ycoor" arguments are provided), MVPOS is called to position the beam.

If scissoring is specified in the GDSCB, the length of the text to be displayed is calculated in raster units and used to determine the x- and y-coordinates for input to the Scissoring routine. Scissoring is then called to scissor the data according to the options set in the GDSCB by the SSCIS subroutine. The data returned from Scissoring is then tested. If any x- and y-coordinates have been entered in the blanked fields of the Scissoring input/output area (see "Scissoring Routine"), a positioning element is generated via Data Generator. The number of characters that will fit in the scissored area is calculated, character mode is set, graphic orders and data are generated by PTEXT, and Data Store is called as many times as necessary to store all the characters in the GDOA.

If the scissoring option set in the GDSCB indicates no scissoring is to take place, the Scissoring routine is not called; the x-coordinate is repositioned (GDSXLAST and GDSXCURR fields in the GDSCB); character mode is set; graphic orders and data are generated by PTEXT; and Data Store is called to store the characters in the GDOA.

If the element is not an update element and is a keyed or correlated (level 1 GDS) element, the key is placed in the key table via Key Table Management. If the element is an update element, Key Table Management is called in order to verify the validity of the key or correlation value passed. Update is called immediately after initialization to assign a temporary GDOA, and again after the element has been generated to write the graphic orders and data from the temporary GDOA to the buffer. Control is then returned to the calling program.

STEOS (Module Name IFFAFA15)

Chart: BE

Function: Places an end-order-sequence (GEOS) order in the specified GDS.

Entry: IFFAFA15 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The end-order-sequence order is placed in the GDOA. When required, a key is built, a key and/or correlation value is placed in the key table via Key Table Management, and the key value is placed in the variable specified by the user as the "key" parameter.

Operation: Depending on the parameters specified, the following operations are performed.

If neither "key" nor "corrval" is specified, STEOS constructs a GEOS order and calls Data Store to place it in the GDOA. Control is then returned to the calling program.

If a key for a level 2 GDS is specified, STEOS computes a new key and places its value in the variable designated as the "key" argument. It then constructs a GEOS order and calls Data Store to place the order in the GDOA. Control is then returned to the calling program.

If the user specifies either "key" or "corrval", or both, for a level 1 GDS, an appropriate key structure is established. Then, an appropriate include/omit (GNOP4/GTRU) structure is established, the GEOS order is created, and generated data is stored in the GDOA. To store the information, if necessary, Data Store may be called three times: (1) for the GNOP4/GTRU order, (2) for the GEOS order, and (3) for resolving the address associated with the GTRU order.

If an error is detected after the GNOP4/GTRU order has been stored, RESET is called to restore the GDS to the condition that existed immediately before STEOS was called, an appropriate return code is placed in the GSPARRAY field of the GSPCB, and control is returned to the calling program.

After the data is stored, Key Table Management is used to enter the key or correlation value, or both, in the key table. Control is then returned to the calling program.

IMAGE IDENTIFICATION SUBROUTINES

These subroutines create the orders which identify limits for a buffer subroutine or sequence that is to be referred to as a unit by an individual key or correlation value. They also establish linkage to a buffer subroutine by means of the key or correlation value.

The subroutines described in this section are:

- BGSEQ/BGSUB, which indicates the beginning of a sequence or a buffer subroutine.
- ENSEQ, which indicates the end of a sequence.
- ENSUB, which indicates the end of a buffer subroutine.
- LKSUB, which establishes linkage to a buffer subroutine.

BGSEQ/BGSUB (Module Names IFFAFA05 and IFFAFA18)

Chart: CA

Function: For BGSEQ, indicates that all the elements that are subsequently generated for this GDS until a call to ENSEQ is issued for the same GDS make up an entity that can be identified by a single key or correlation value. For BGSUB, designates the beginning of a buffer subroutine in a particular GDS.

Entries: IFFAFA05 for BGSEQ, IFFAFA18 for BGSUB, both entered from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: Graphic orders and data that affect the status (include or omit) of all elements within the specified sequence or buffer subroutine are placed in the GDOA via Data Store. The key value is placed in the variable specified by the user as the "key" argument. Unless it is a level 2 GDS, the key is placed in the key table via Key Table Management.

Operation: After checking the validity of the input, switches are set to indicate whether a sequence or a buffer subroutine is being established, whether the status is include or omit, and whether level 1 or level 2 key or correlation values are used. A skeleton parameter list for Data Store and the key structure (if it is level 1) are built.

If a sequence is being established, a GTRU (for omit status) or a GNOP4 (for include status) order is placed in the input parameter list for Data Store, with an indication of an unresolved address. (This address will be resolved upon an entry from ENSEQ.) Data Store is called to place the order in the GDOA. If a level 1 key is indicated, the key is completed and Key Table Management is called to place it in the key table. Control is then returned to the calling program.

If a buffer subroutine is being established, a GTRU order to the unresolved address that will be resolved at ENSUB time, a GTRU to a zero address that will be changed by buffer subroutine linkage, a GNOP4 (include) or GTRU (omit) order with an unresolved address, and a GDRD order are set up in the input parameter list for Data Store, with indication of unresolved addresses. Data Store is called to place the orders in the GDOA. The address for the include/omit order is resolved. If a level 1 key is indicated, the key is completed, placed in the key table by Key Table Management, and control is returned to the calling program.

Note: The buffer subroutine facility is provided only on an IBM 2250, Model 3. BGSUB tests the GTMFEATS field in the GTMCB for a Model 3. If any other model is indicated, control is immediately returned to the calling program along with an appropriate return code.

ENSEQ (Module Name IFFAFA06)

Chart: None

Function: To indicate the end of a sequence of elements.

Entry: IFFAFA06 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: Length of the sequence is placed in the key, and the address is resolved in the include/omit structure for the sequence via Data Store.

Operation: Input is tested for validity, and the key for the sequence is obtained from the key table via Key Table Management. The length of the sequence is computed and stored in the key. A parameter list for Data Store is set up. Data Store is called to resolve the address in the GNOP4/GTRU order in the include/omit structure of the sequence.

Control is returned to the calling program.

ENSUB (Module Name IFFAFA07)

Chart: None

Function: To indicate the end of a buffer subroutine.

Entry: IFFAFA07 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: A GTRU order which points to the beginning of the buffer subroutine is placed in the GDOA (or buffer, on GDOA overflow). If level 1 keying is used, the subroutine length is added to the key in the key table.

Operation: If level 1 keying is used, the key is obtained from the key table via Key Table Management. For both level 1 and level 2, the key address is placed in a parameter list for Data Store, a GTRU order is constructed with the computed address of the beginning of the buffer subroutine, and Data Store is called to place the order in the GDOA. The unresolved address for the first GTRU order placed in the GDOA for BGSUB is resolved at this time by Data Store. Upon the return of control from Data Store, if level 1 keying is used, the length of the buffer subroutine is placed in the key. Control is returned to the calling program.

LKSUB (Module Name IFFAFA08)

Chart: CB

Function: Creates linkage to execute a buffer subroutine and return.

Entry: IFFAFA08 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: Orders are created and stored in the GDOA (or buffer on overflow) to effect linkage to the buffer subroutine. If a level 1 key is designated for the linkage, the key is built and placed in the key table.

Operation: The key for the subroutine (either level 1 or level 2) is obtained and used for calculating addresses for the orders that LKSUB generates to establish linkage. Linkage orders are created and the input parameter list for Data Store is built. The beginning address of the subroutine for which linkage is being established is calculated and placed in the appropriate orders of the linkage with the proper displacement. Data Store is then called to resolve the return address in the orders, and to store them in the GDOA (or buffer, if the GDOA is full). If a level 1 key was specified for the linkage, the key is placed in the key table. Control is returned to the calling program.

IMAGE CONTROL SUBROUTINES

These subroutines designate when and how an image is to be displayed on the 2250 screen. Three of them implement the update facility supported by GSP.

The subroutines described in this section are:

- EXEC, which places all graphic orders and data in executable form in the buffer so that a display will appear on the 2250 screen.
- INCL/OMIT, which places a GDS, a keyed or correlated element, sequence, buffer subroutine, or buffer subroutine linkage in include or omit status.
- RESET, which resets (removes) all or part of a GDS without terminating the use of the GDS.
- IDPOS, which provides the starting points from which the x- and y-coordinates are to be computed on any call where the GDSCB may not have correct information for subsequent calls.
- FSMOD, which causes an appropriate set mode order to be generated when an image generation subroutine is subsequently called.
- ORGDS, which controls the image being displayed by ordering the regeneration sequence for GDSs associated with a particular 2250.

EXEC (Module Name IFFAFA11)

Chart: DA

Function: Writes to the buffer any data that has been generated for a GDS since the last call to EXEC or since an overflow occurred in the GDOA.

Entry: IFFAFA11 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: See "Function."

Operation: If the GDOA is empty, or if no data has been generated since the last call to EXEC, and if the GDS is not an equivalent GDS, buffer execution is started and control is returned to the calling program.

The GDSREPDT field in the GDSCB is checked to determine if any replacement data was stored from a previous overflow. This replacement data is to be used to overlay the return address to the flow control structure written out by the previous call to EXEC. If this replacement data was saved, it is written to the buffer. Then, all graphic orders and data in the GDOA located between the CRSA point and the OLP point are written to the buffer. This is followed by the GTRU order back to the flow control structure, which is contained in the GDSOVDAT field of the GDSCB.

If this is an equivalent GDS, the bit in the GDSFLAGS byte is set to indicate that this GDS is in the buffer. All similar bits in the other GDSs in the same equivalency group are turned off. If the GDS is not an equivalent GDS, a flag is set in the OACB to indicate that on overflow the first four bytes at the CRSA must be placed in the GDSREPDT field of the GDSCB. The fields in the OACB are updated to reflect the new positions in the GDOA and the buffer. Any unresolved addresses previously noted in the GDOA are noted to now be in the buffer. If there is a cursor in the GDS, EXEC inserts the cursor in the buffer.

INCL/OMIT (Module Names IFFAFA09 and IFFAFA10)

Chart: None

Function: Generates necessary include (GNOP4) or omit (GTRU) orders and sets bits to unblank or blank the 2250 beam to place elements, sequences, buffer subroutines, buffer subroutine linkages, or GDSs in include or omit status as designated by the user.

Entry: IFFAFA09 for INCL, or IFFAFA10 for OMIT, both entered from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: For multiple elements, an order immediately preceding the image generation orders and data for an element, sequence, buffer subroutine, buffer subroutine linkage, or GDS is changed either to transfer around the image generation orders specified (OMIT), or to allow them to be processed (INCL). For single elements, the beam bit of the image generation order is changed to allow blanking or unblanking. Except for sequences and STPOS/MVPOS or PTEXT elements being placed in omit status, the beam will be positioned where the element ended.

Operation: If omit status is to be established for multiple elements, a sequence, a buffer subroutine, buffer subroutine linkage, or a GDS, a GTRU is positioned in the buffer to transfer around the image generation orders for the elements referred to by the routine.

If include status is to be established for multiple elements, a GNOP4 is positioned in the buffer preceding the image generation orders for the elements. This order overlays any previously stored GTRU order which would have caused the image generation orders to be bypassed.

For a buffer subroutine, the position of the appropriate order within the structure is determined, and the include/omit status is handled the same as for multiple elements. For buffer subroutine linkage, the include/omit status is handled exactly the same as for multiple elements.

For single elements in either include or omit status, a bit specifying that the beam is to be unblanked or blanked is set in the image generation order as appropriate.

For both INCL and OMIT, a bit is always set in the key to specify the new status as include or omit.

For examples of the include/omit structure in various types of elements, see Figure 16.

RESET (Module Name IFFAFA12)

Chart: DB

Function: Resets a GDS, or a keyed or correlated element and all elements generated after the one referred to. To do so, it:

- Removes orders from the buffer as necessary.
- Frees any unused buffer.
- Sets up the GDOA and fields in the OACB to reflect the reset condition of the GDS.
- Removes appropriate keys from the key table.

Entry: IFFAFA12 from Director Part 2, BGTRK, ENTRK, LOCPN, ORGEN, or from any GSP image generation subroutine which encounters an error while generating data and must restore the GDS to the condition that existed before the call.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines." The parameter list may be passed by another GSP subroutine.

Output: The key table, GDSCB, and OACB are appropriately updated. If reset occurred in the buffer, unused buffer sections are released, and those images reset are removed from the display regeneration cycle.

Operation: Processing by RESET encompasses three major phases: (1) manipulating the key table, (2) checking for orders and data to be reset, and (3) resetting within the GDOA or the buffer. After third phase processing is completed successfully, the current mode field (GDSGRMOD) in the GDSCB is set to zero so that the next execution of an image generation subroutine will generate a set mode order. Control is then returned to the calling program. Each of the processing phases is described in detail in the paragraphs that follow.

Manipulating the key table depends on various conditions as follows:

- Only "gdsname" parameter is specified -- Key Table Management resets the entire key table for this GDS.
- The "key" and/or "corrval" parameters

are specified -- Key Table Management locates the specified key in the key table. RESET saves this key. Key Table Management then resets the key table to the specified key so that this key and all subsequent keys in the table are removed.

- Input "key" is not found in the key table and length of input "key" is zero -- RESET assumes it was called by another GSP subroutine. The key table is not reset because the key has never been entered in the key table.

- Input "key" or "corrval" is not found in the key table and length of input "key" is not zero -- RESET places appropriate return code in the GSPARRAY and returns control to the calling program.

- A level 2 GDS is specified -- key table manipulation is bypassed.

After key table manipulation, the following information is available for further use by RESET:

- If the entire GDS is to be reset, the key contains zeros.

- If it is a level 1 GDS, with a key or correlation value, the key contains assorted information about the element to which the GDS is to be reset.

- If it is a level 2 GDS, or if RESET was entered from another GSP subroutine, the key contains the logical buffer address, but does not contain information about the element to which the GDS is to be reset.

Checking for orders and data in the GDS to be reset is done next. If the GDS does not contain any graphic orders and data, the current mode field in the GDSCB is set to zero so that the next execution of an image generation subroutine generates a set mode order; control is then returned to the calling program. If the GDS contains graphic orders and data, RESET proceeds to its third processing phase, depending upon whether the GDS to be reset is in the GDOA or in the buffer.

Resetting within the GDOA includes the appropriate updating of the OACB from information in the key. The point to which the GDS is to be reset is determined and one or more of the following occurs:

- If the GDS is being reset to within a sequence or buffer subroutine, the sequence or buffer subroutine is reopened if necessary, and the key associated with it is updated accordingly.

- If the GDS is an equivalent GDS which is currently in the buffer, the reset GDS is executed by a call to EXEC.

- If the GDS is being reset to a point within a sequence or buffer subroutine that is not presently open, the key of the sequence or buffer subroutine is obtained from the key table, and the length of the key is set to zero. This reopens the sequence or buffer subroutine. The GDSKEYSV field in the GDSCB is set to the logical address of the key. The bit in the key denoting that a sequence or buffer subroutine has been started is set to one. In addition, the flow control return address is placed at the top of the sequence or buffer subroutine in the GDOA and the location of this unresolved address is saved in the OACB.

If the reset condition is none of the above, the third phase of processing is complete.

Resetting within the buffer involves computing the location within the buffer at which the resetting is to begin. The OACB is appropriately updated from information in the key. If the buffer assigned to the GDS is not a 128-byte buffer subsection, Buffer Management is used to reset any buffer freed. If any replacement data in the GDSCB is part of the reset, the appropriate GDSCB fields are set to zero and a flag in the OACB is set to save the next four bytes as replacement data. The address that transfers control back to the flow control entry is written to the buffer at the location specified in the key. If the entire GDS is being reset, this location will be at the beginning of the GDS. Thus, any image created by the graphic orders and data being reset will be removed from the display.

The point to which the GDS is to be reset is determined and one of the following occurs:

- If the GDS is being reset to the beginning of a sequence or buffer subroutine, the sequence or buffer subroutine is removed.

- If the GDS is being reset to a point within a sequence or buffer subroutine that is not presently open, the key of the sequence or buffer subroutine is obtained from the key table. Next, the length of the key is set to zero, thus reopening the sequence or buffer subroutine. The bit in the key denoting that a sequence or buffer subroutine has been started is set to one. In addition, in the case of resetting within a sequence, the physical buffer address of the sequence is computed so that the address that branches to the end of the sequence can be replaced with the address that will transfer back to the flow control entry.

If the reset condition is not one of the above, or after one of the above processes is corplete, the third phase of processing is corplete.

Note: If this is a level 2 GDS, it is impossible to determine whether or not the GDS is being reset to a point within a sequence or buffer subroutine that is not currently open. Therefore, such a sequence or buffer subroutine cannot be reopened.

IDPOS (Module Name IFFAFA13)

Chart: DC

Function: Determines absolute x- and y-coordinate beam positions for subsequent use by image generation subroutines.

Entry: IFFAFA13 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The GDSXCURR, GDSYCURR, GDSXLAST, and GDSYLAST fields of the GDSCB are updated with the new beam position coordinates.

Operation: After determining the location and validity of the GDSCB identified by the "gdsname" input parameter, IDPOS processes the values of the remaining input parameters in pairs.

Values specified as the "xlast" and "ylast" parameters are processed first. IDPOS moves these values into a data array that will be passed to Scaling, sets the GDSDATMD field of the GDSCB to indicate that absolute input data is being provided, and calls Scaling to scale the input data to actual raster unit coordinates. Upon successful completion of the scaling operation, IDPOS places the newly scaled values in the GDSXLAST and GDSYLAST fields of the GDSCB and checks to see if the "xcurr" parameter has been specified. If not, IDPOS also places the scaled values in the GDSXCURR and GDSYCURR fields of the GDSCB, and restores the GDSDATMD field of the GDSCB to the condition that existed immediately before IDPOS was entered.

If the "xcurr" and "ycurr" parameters were specified, their values are moved into a data array and Scaling is called to scale these values to raster unit coordinates. Upon successful completion of the scaling operation, IDPOS places these newly scaled values in the GDSXCURR and GDSYCURR fields of the GDSCB and restores the GDSDATMD field to the condition that existed immediately before IDPOS was entered.

If an error is encountered at any time during processing by IDPOS, contents of all fields in the GDSCB are restored to the condition that existed immediately before IDPOS was entered.

FSMOD (Module Name IFFAFA14)

FSMOD sets the GDSGRMOD field of the GDSCB to zero so that the next image generation subroutine to be called will generate a set mode order. Entry is from Director Part 2. Exit is to the calling program. Input consists of the address of a work area in register 1. The first word of this area contains the address of a variable that points to the GDSCB.

ORGDS (Module Name IFFAFA19)

Chart: None

Function: Reorders the flow control structure in the buffer and the in-use list in the FCT in main storage to reflect the new regeneration sequence of GDSs as specified by the user. The system GDS is always left as the first GDS in the flow control structure.

Entry: IFFAFA19 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: See "Function."

Operation: See "Function." For a complete discussion in regard to the flow control structure and the FCT, refer to the section "Flow Control Management."

ALLOWING KEYBOARD INPUT AND BUFFER DATA ANALYSIS

GSP provides subroutines that allow data to be entered from the alphameric keyboard, and also allow orders and data or characters to be read from the 2250 buffer into main storage to be analyzed. Two of these subroutines (ICURS and RCURS) insert or remove a cursor in the specified GDS. One subroutine (GSPRD) allows data to be read from the buffer into main storage.

ICURS (Module Name IFFADA03)

Chart: EA

Function: Inserts a cursor into a specified location within a GDS.

Entry: IFFADA03 From Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: If the cursor location within the GDS is not in the buffer, the buffer address of the cursor location is placed in the GTMCB and appropriate flags are set in the GTMCB and GDSCB. If the cursor location is in the buffer, the cursor is inserted in the buffer and appropriate flags are set in the GDSCB and GTMCB.

Operation: ICURS checks the GDSFLAGS field of the GDSCB to determine if the programmer is using a level 1 or level 2 GDS. If level 2 is being used, the logical buffer address is computed from the locations specified in the key supplied by the user.

If level 1 is being used, the logical buffer address for insertion of the cursor is computed from the key table using the key or correlation value supplied in the call to the ICURS subroutine. If both key and correlation value are supplied, the key is used to compute the logical buffer address and the correlation value is substituted in the key table for the correlation value previously assigned to the element or sequence into which the cursor is to be inserted.

ICURS then checks the GTMFLAGS field in the associated GTMCB to determine if a cursor is already present in the 2250. If it is present, RCURS is called to remove the cursor.

Once the logical buffer address has been computed, ICURS determines whether the element or sequence into which the cursor is to be inserted is in the buffer or in the main storage GDOA. If the element or sequence is in the buffer, the physical buffer address is computed, the cursor is inserted in the buffer, and flags are set in the GDSFLAGS field of the GDSCB and the GTMFLAGS field of the GTMCB to indicate the presence of a cursor. If the element or sequence is in the GDOA, the physical buffer address is computed and placed in the GTMCRLOC field of the GTMCB. Flags are then set in the GDSFLAGS field of the GDSCB and the GTMFLAGS field of the GTMCB to indicate the presence of a cursor. The cursor is then moved to the buffer the next time the contents of the GDOA are transferred to the buffer by a call to the EXEC subroutine.

RCURS (Module Name IFFADA02)

Chart: EB

Function: Removes the cursor from a GDS.

Entry: IFFADA02 from Director Part 2, or from ICURS.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: Cursor is removed from the GDS, and applicable flags are removed from the GDSFLAGS and GTMFLAGS fields of the GDSCB and GTMCB.

Operation: RCURS checks the GTMFLAGS field in the GTMCB associated with the specified GDS to determine if a cursor is present in this 2250. If there is none, the flag fields in the GTMCB and all its associated GDSCBs are reset, the GTMCRLOC field in the GTMCB is set to zero, and control is returned to the calling program.

If the cursor is present, RCURS locates any cursor that is in the buffer assigned to this 2250 which was allocated by GSP buffer management, and removes the cursor. The GTMCRLOC and the GTMFLAGS fields of the GTMCB and the GDSFLAGS fields of the associated GDSCBs are reset.

GSPRD (Module Name IFFADA01)

Chart: EC

Function: Reads all or part of a GDS from the buffer into main storage as specified in the user parameters.

Entry: IFFADA01 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: Requested data from the buffer is placed in a user-specified array in main storage. The result of the read operation is placed in the variable specified by the "termcode" parameter.

Operation: The parameter list is tested for various combinations of parameters. Based on the results of these tests, the logical length of the buffer is determined, either from the key, or the OACB (for an entire GDS). The logical buffer length and the "count" parameter are compared, and the smaller one is used to set up a read control area which determines the amount of data to be read from the buffer. The BCT is used to convert the logical buffer start address into a physical buffer start address, and the length of the buffer section is calculated.

The buffer is read into the work area. This operation is ended if (1) the cursor is detected when "count" was negative, (2) the specified amount of data has been read, or (3) the end of the GDS or the last element has been reached.

The data is then moved from the work area to the user's main storage array. If text only is specified in the parameter list, characters appearing in the work area are moved to the user-specified area. If all data is requested, all orders and data from the work area are moved to the user-specified area. The type of action that resulted in a termination of the read operation is recorded in the variable specified as the "termcode" parameter, and control is returned to the calling program.

COMMUNICATION BETWEEN 2250 OPERATOR AND GSP PROGRAM

Communication between the 2250 operator and GSP is accomplished through the use of the attention related subroutines and the light pen subroutines. These subroutines make available to the operator the light pen, the programmed function keyboard, or the alphameric keyboard. If the operator desires to do so, he can interrupt the program by depressing the CANCEL key on the alphameric keyboard. The CANCEL Key (Panic Key) routine is described in this section, although it is an internal routine which is loaded by INDEV.

ATTENTION RELATED SUBROUTINES

Attention related subroutines are used for two-way communication between the 2250 user and the GSP program. They create, manipulate, and terminate control blocks that make this communication possible. By using these control blocks, the attention related routines enable attention information to be received from the 2250 operator and passed to the GSP program. The attention related subroutines designate the attentions that are to be accepted from the 2250 operator, and those that are to be ignored. They define the type of information that is to be made available to the program, and make that information available to the program.

The attention related subroutines described in this section are:

- CRATL, which creates an attention level control block (ATLCB).
- ENATL, which terminates the use of an ATLCB.
- ENATN, which designates the sources from which attentions will be accepted.
- DSATN, which designates the sources from which attentions are no longer to be accepted.
- SLPAT, which permits light pen attentions to be accepted.
- RQATN, which makes information about an attention available to the GSP program.
- MLITS, which designates the programmed function keyboard indicator lights that are to be on and those that are to be off.
- MLPEO, which designates the type of information that is to be made available for light pen or end-order-sequence attentions.
- MPATL, which modifies the position of an ATLCB within the hierarchy of attention levels.
- CANCEL Key (also called Panic Key) routine, which allows an abnormal termination dump to be produced when the 2250 operator senses that his program is not performing properly.
- SALRM, which sounds the single stroke audible alarm on the 2250.

Figure 9 depicts how the attention related routines manipulate the contents of the control block fields as an attention level is created, the light pen attention source is enabled, a light pen attention is received, and information is requested about that light pen attention.

CRATL (Module Name IFFACA00)

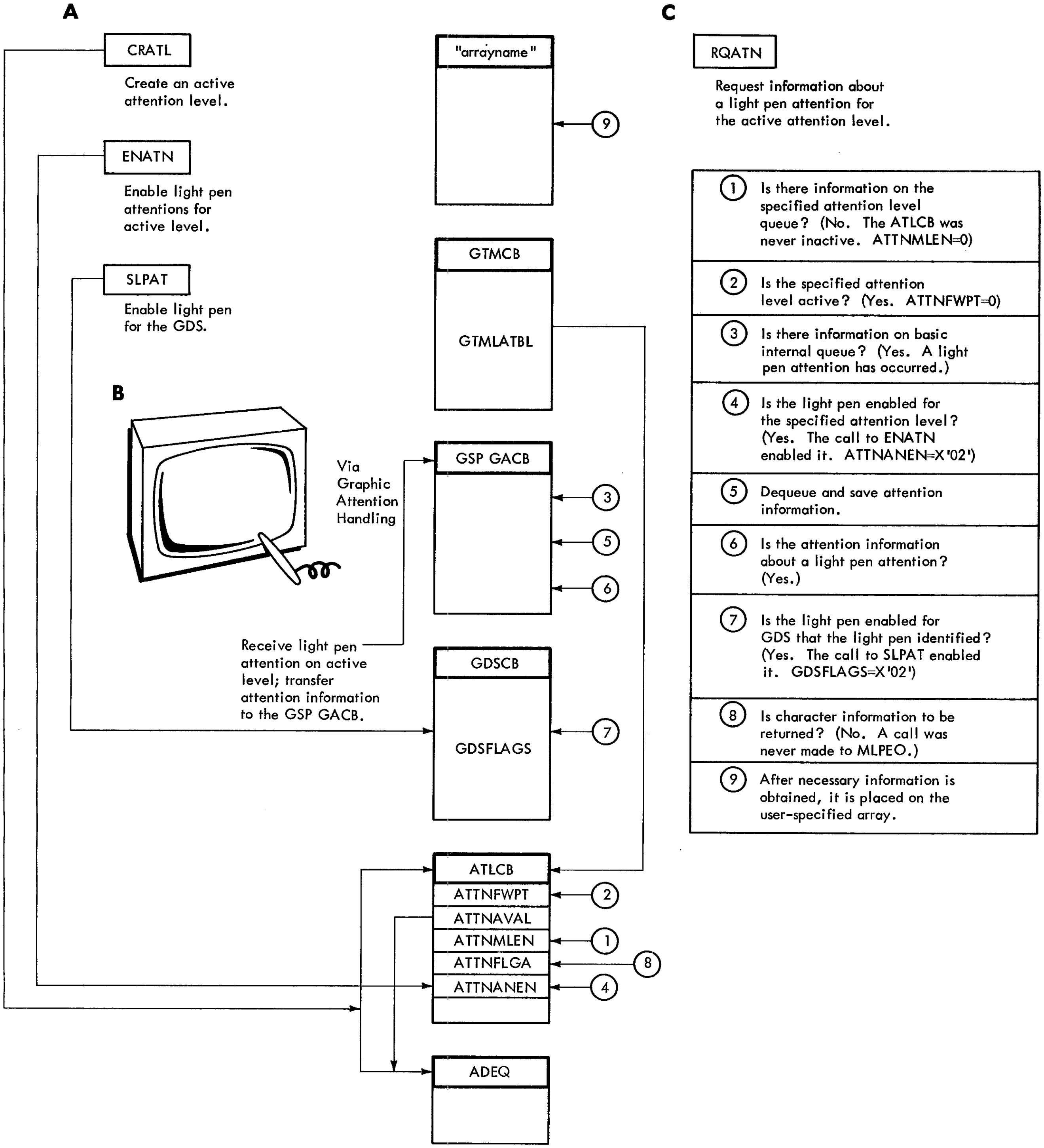
Chart: FA

Function: Establishes an active attention level by creating and initializing an attention level control block (ATLCBs) and its attention data entry queues (ADEQs). This block and its queue are described in detail in Appendix A.

Entry: IFFACA00 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."


A
CRATL
Create an active attention level.
ENATN
Enable light pen attentions for active level.
SLPAT
Enable light pen for the GDS.
B
Via Graphic Attention Handling
Receive light pen attention on active level; transfer attention information to the GSP GACB.
"arrayname"
GTMCB
GTMLATBL
GSP GACB
GDSCB
GDSFLAGS
ATLCB
ATTNFWPT
ATTNAVAL
ATTNMLEN
ATTNFLGA
ATTNANEN
ADEQ
9
3
5
6
7
2
1
8
4
C
RQATN
Request information about a light pen attention for the active attention level.

1 Is there information on the specified attention level queue? (No. The ATLCB was never inactive. ATTNMLEN=0)

2 Is the specified attention level active? (Yes. ATTNFWPT=0)

3 Is there information on basic internal queue? (Yes. A light pen attention has occurred.)

4 Is the light pen enabled for the specified attention level? (Yes. The call to ENATN enabled it. ATTNANEN=X'02')

5 Dequeue and save attention information.

6 Is the attention information about a light pen attention? (Yes.)

7 Is the light pen enabled for GDS that the light pen identified? (Yes. The call to SLPAT enabled it. GDSFLAGS=X'02')

8 Is character information to be returned? (No. A call was never made to MLPEO.)

9 After necessary information is obtained, it is placed on the user-specified array.

Figure 9. Manipulation of Control Blocks During Attention Processing

Output: An ATLCB and associated ADEQs are created. The address of the ATLCB is placed in the user-specified variable ("attnlevel"). The indicator lights on the programmed function keyboard are lit as per the GTMLIGHT field of the GTMCB.

Cperation: Main storage is obtained for the ATLCB and its ADEQs. If there is a previously active attention level, the ATTNINQ macro instruction is issued to determine if any attention information is available. If so, that information is transferred from the GSP GACB to the active ATLCB. Attention sources in the GSP GACB associated with the specified GTMCB are disabled and the light pen restart option is set for no restart. The ATLCB and its ADEQs are initialized as follows:

- The ADEQs are created immediately following the ATLCB, and are chained. The address of the first ADEQ in the chain is placed in the ATTNAVAL field.

- ATTNFLGA is set as per the "dequectl" parameter.

- ATTNLITE is set to indicate the default option in the GTMLIGHT field of the GTMCB, for lighting the programmed function indicator lights.

- The address of the ATLCB is placed in the user-specified variable.

If this is the first ATLCB created for the GTMCB, pointers are set as follows:

- ATTNFWPT is set to zero to indicate that this is the active attention level.

- ATTNBKPT is initialized to the address of the GTMCB.

- ATTNBPID is set to X'01' to indicate that ATTNBKPT points to the GTMCB.

- The address of the ATLCB is placed in the GTMCB.

If this is not the first ATLCB created for the GTMCB, pointers are set as follows:

- The address of the newly created ATLCB is placed in the ATTNFWPT field of the previously active ATLCB. Thus, the previously active attention level becomes inactive.

- The address of the previously active ATLCB is placed in the ATTNBKPT field of the newly created ATLCB.

- The ATTNFWPT field of the newly created ATLCB is set to zero to indicate it is the active attention level.

- The address of the newly created ATLCB is placed in the GTMCB, which points to the active attention level.

See "Chaining" under "Introduction" for the relationship of pointers.

Next, the programmed function indicator lights are lit as defined in the GTMLIGHT field of the GTMCB. Control is then returned to the calling program.

ENATL (Module Name IFFACA01)

Chart: FB

Function: (1) Terminates all attention levels that are lower in the attention level hierarchy for this GTMCB than the specified attention level, thus making the specified attention level active; or (2) terminates the specified attention level and all other attention level(s) lower in the hierarchy for this GTMCB, making the next higher remaining attention level active.

Entry: IFFACA01 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The appropriate ATLCBs and their ADEQs are deleted.

Operation: If the parameters are valid, the GSP GACB attention sources are disabled to prevent any attention from occurring while the attention levels are being manipulated.

If a "rangecode" of 1 has been specified, pointers are manipulated as follows:

- If the ATTNBKPT field of the specified ATLCB points to the GTMCB, the GTMLATBL field in the GTMCB is set to zero.

- Otherwise, the ATTNBKPT field of the specified ATLCB is placed in the GTMLATBL field of the GTMCB, and the ATTNFWPT of the next higher ATLCB is set to zero.

If a "rangecode" of 2 has been specified, various pointers are manipulated as follows:

- The address of the specified ATLCB is placed in the GTMCB.

- The ATTNFWPT field of the specified ATLCB is set to zero.

If an active attention level still remains after the designated attention levels have been removed from the ATLCB chain, attention sources and the light pen restart option are enabled in the GSP GACB (according to the options specified in the active ATLCB). Programmed function indicator lights are lit as specified in the active ATLCB.

Storage for the detached ATLCBs and their ADEQs is now freed, and control is returned to the calling program.

ENATN (Module Name IFFACA02)

Charts: FC, FD

Function: Performs two services, namely, (1) defines which attention sources are to receive attentions when the designated attention level is active by setting bits in the ATLCB (and in the GSP GACB if the specified level is active), and (2) removes any previous attention information for those sources that may be on the queue for the designated ATLCB.

Entry: IFFACA02 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: Appropriate bits are set in the proper control blocks (see "Function" above). Any attentions currently queued for the specified level that match attention sources just enabled are removed from the queue. Programmed function indicator lights are lit as appropriate.

Operation: While the "attnsource" parameters are being tested, two temporary attention source masks -- one for programmed function keys and one for all other attention sources -- are set up. For each "attnsource" specified, a bit is set in the appropriate mask. The meaning of the bits in these masks is the same as for the ATTNPFEN and ATTNANEN fields in the ATLCB. These masks are then added to the ATTNPFEN and ATTNANEN fields in the ATLCB.

If the specified attention level is active, all attention information in the GSP GACB that was obtained from an attention source just enabled is removed from the GSP GACB and from the ATLCB. The freed ADEQs are transferred from the ATTNMLEN field of the ATLCB to the ATTNAVAL field. The newly constructed attention source masks are now added to the attention source masks in the GSP GACB. Indicator lights for newly enabled programmed function keys are lit, and control is returned to the calling program.

If the specified attention level is inactive, the ATTNMLEN field of the ATLCB is checked and any information queued for an attention from one of the newly enabled sources is removed. The ATTNAVAL field is updated to include these free areas. Control is then returned to the calling program.

DSATN (Module Name IFFACA03)

Charts: None

Function: Causes future attentions for the designated attention sources to be disregarded by deleting the attention sources from the enabled attention source masks in the ATLCB, and in the GSP GACB if the designated attention level is active.

Entry: IFFACA03 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The enabled attention source bits are properly reset in the appropriate control block(s). (See "Function" above.) Programmed function indicator lights are relit as appropriate.

Operation: As in ENATN, while the "attnsource" parameters are being tested, two temporary attention source masks are set up. These masks are now subtracted from the corresponding ATTNPFEN and ATTNANEN fields in the specified ATLCB. If the ATLCB is inactive, control is returned to the calling program.

If the ATLCB is active, the new attention masks in the ATLCB are placed in the associated GSP GACB. Programmed function indicator lights are relit as appropriate, and control is returned to the calling program.

SLPAT (Module Name IFFACA06)

Chart: None

Function: Enables or disables light pen attentions for a specified GDS.

Entry: IFFACA06 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The appropriate bit is set in the GDSCB and the appropriate graphic order is placed in the buffer.

Operation: Depending on the value of the "detect" parameter, the following steps are taken:

- If "detect" is 1, bit 7 in the GDSFLAGS field of the GDSCB is set to one, and the Enable Switch Detect (GESD) order is written to the buffer location specified by the GDSFCBUF field in the GDSCB. Control is then returned to the calling program.

- If "detect" is 2, bit 7 in the GDSFLAGS field is set to zero, and the Disable Light Pen Detect (GDPD) order is written to the buffer location specified by GDSFCBUF. Control is then returned to the calling program.

RQATN (Module Name IFFACA08)

Charts: FD, FE, FF, FG

Function: Performs the following as designated in the call to the RQATN subroutine:

- Makes available attention information about an attention from a designated source. It dequeues that information as it is made available unless the do-not-dequeue option for an inactive ATLCB was specified in the call to the CRATL subroutine for the associated ATLCB.

- Delays execution of the main line program until an attention occurs from a designated source. Then information about that attention is made available to the calling program.

- Places additional information for light pen, end-order-sequence, and programmed function keyboard attentions in a user-defined array.

Entry: IFFACA08 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: A code indicating which designated source caused the attention is assigned as the value of the "codeloc" parameter. If the designated source was the light pen, an end-order-sequence order, or a programmed function key, additional information about the attention is placed in the area identified by the "arrayname" parameter.

Operation: While the "attnsource" parameters are being tested for validity, two temporary attention source masks are established. These masks are used later to determine (1) whether attention information is from one of the sources designated in the call to the RQATN subroutine, and (2) whether those sources are enabled for the ATLCB if that ATLCB is active. The masks are discussed in detail in the description of the ENATN routine.

Once the attention source masks are established, processing proceeds in two major phases: (1) to determine if an attention has occurred from designated attention sources, and (2) to determine which source caused the attention and then make appropriate information about that attention available to the calling program.

Determining if an attention had occurred from designated attention sources. A test is made to determine if attention information for any of the designated sources is on the ATLCB queue. If it is on the queue, that information is saved in a work area. The information is then removed from the queue unless the ATLCB is inactive and the do-not-dequeue option is specified (see CRATL subroutine).

If there is no information on the queue and the specified ATLCB is inactive, control is immediately returned to the calling program. If no immediate return is wanted, a return code designating a parameter error is placed in the GSPARRAY field of the GSPCB.

If there is no information on the queue and the specified ATLCB is active, processing continues as follows:

1. If the designated attention sources are disabled, control is immediately returned to the calling program along with an appropriate return code.

2. If the designated attention sources are enabled, RQATN determines if there is any attention information from one of the designated sources in the GSP GACB. If there is none and an immediate return is wanted, control is immediately returned to the calling program. If there is none and no immediate return is wanted, RQATN waits until an attention occurs. When the awaited attention occurs, information about it is saved in a work area and removed from the GACB.

Determining which source caused the attention and making appropriate information available. Once information either from the ATLCB queue or from the GSP GACB has been obtained, tests are made to determine which requested source caused the attention. Depending on the attention source, the following processing is done:

1. If the light pen caused the attention, the GDSCB is located via the buffer control table (BCT). If the GDSCB cannot be located, or if the light pen is not enabled for this GDSCB, regen-

eration of the display occurs (if the restart flag is set to zero) and a test is made to determine if any other attention has occurred for one of the requested sources.

If the light pen is enabled for this GDSCB, RQATN determines if character information is to be returned. If it is, the information is obtained from the buffer, and the display is regenerated (if desired).

If the "arrayname" parameter was specified, the designated array is filled in with information from the GTMCB, ECT and key table. If coordinate information is desired, CNVRT is used to provide that information. A value of 34 is placed in the "codeloc" variable and control is returned to the calling program.

2. If an end-order-sequence order caused the attention about which information has been obtained, and if coordinate information is to be returned, RQATN obtains the information from the X and Y position registers, and restarts the display (if desired).

   If the "arrayname" parameter was specified, the array is now filled in the same way that it was for a light pen attention. A value of 35 is placed in the "codeloc" variable and control is returned to the calling program.

3. If a programmed function key caused the attention about which information has been obtained, the overlay code is placed in the first word of the array if the "arrayname" parameter has been specified. The numeric value of the programmed function key is then placed in the "codeloc" variable and control is returned to the calling program.

4. If the END key caused the attention, a value of 32 is placed in the "codeloc" variable and control is returned to the calling program.

5. If the CANCEL key caused the attention, a value of 33 is placed in the "codeloc" variable before control is returned to the calling program.

Note: CANCEL key is currently unavailable for use as it is enabled for the Panic Key routine which has the highest possible priority. Thus, a CANCEL key attention is always recognized by the Panic Key routine.

MLITS (Module Name IFFACA07)

Charts: None

Function: Sets bits either in the GTMCB or the ATLCB that indicate which programmed function indicator lights are to be lit.

Entry: IFFACA07 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The designated programmed function indicator lights are lit.

Operation: Depending on the parameters that were specified, the appropriate setting of bits is accomplished. If the first parameter points to a GTMCB, bits are set in the GTMLIGHT field as shown in Table 2.

Table 2. GTMLIGHT Options

"status"	Bit Setting and Meaning
1	Error return with appropriate return code
2	X'02' Turn all indicator lights off
3	X'03' Light enabled keys
4	X'04' Use GTMCB light mask
other	Error return with appropriate return code

Note: If "status" equals four, the "lights" parameters are tested for validity. If they are invalid, control is returned to the calling program along with an appropriate return code. If they are not specified, the present light mask in the GTMPFKSS field of the GTMCB is used. Otherwise, a light mask is created. A bit is set for each "lights" parameter specified. Bit positions 0-31 in the mask correspond to programmed function keys 0-31, respectively. This light mask is placed in the GTMPFKSS field of the GTMCB.

If the first parameter points to an ATLCB, bits are set in the ATTNLITE field as shown in Table 3.

Table 3. ATTNLITE Options

"status"	Bit Setting and Meaning
1	X'01' Use GSP default option in GTMCB
2	X'02' Turn all indicator lights off
3	X'03' Light enabled keys
4	X'04' Use ATLCB light mask
other	Error return with appropriate return code

Note: If "status" equals four,

- The present light mask in the ATTNPFLM field of the ATLCB is used if "lights" is not specified.
- A light mask created by MLITS is placed in the ATTNPFLM field of the ATLCB.

Once the bits are set, the active ATLCB is located via the GTMCB and the programmed function indicator lights are lit according to the lights option byte (ATTNLITE) of the active ATLCB. Control is then returned to the calling program.

MLPEO (Module Name IFFACA05)

Charts: None

Function: Designates (1) the type of information to be queued about a light pen or end-order-sequence attention, and (2) when display regeneration is to begin after a light pen or end-order-sequence attention has occurred.

Entry: IFFACA05 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: Appropriate bits are set in ATTNFLGA field of the ATLCB. If the specified attention level is active, appropriate bits are also set in the GSP GACB.

Operation: Depending on the values of the parameters, the option bits are set in the ATTNFLGA field of the specified ATLCB as shown in Table 4.

Table 4. MLPEO Options

Parameters			ATTNFLGA Field	
"attntyp"	"info"	"restart"	Bit Setting	Meaning
1	1		Bit 0 = 0	Return no coordinate information for EOS attentions.
	2		Bit 0 = 1	Return coordinate information for EOS attentions.
2	1		Bit 2 = 0 Bit 3 = 0	Return no coordinate or character code information for light pen attentions.
	2		Bit 2 = 1	Return coordinate information for light pen attentions.
	3		Bit 3 = 1	Return character code information for light pen attentions.
	4		Bit 2 = 1 Bit 3 = 1	Return coordinate and character code information for light pen attentions.
2		1	Bit 4 = 1	Restart display immediately after attention occurs.
1		1	Bit 1 = 1	
2		2	Bit 4 = 0	Restart display after the next call to the EXEC subroutine.
1		2	Bit 1 = 0	

Note: If the "info" or "restart" parameters are not specified, a value of 1 is assumed.

If the specified ATLCB is active, the restart option byte in the GSP GACB is set to reflect the information specified by the "restart" parameter. Control is then returned to the calling program.

MPATL (Module Name IFFACA04)

Charts: FH, FJ

Function: Changes the relative position of a specified ATLCB within the hierarchy of all ATLCBs currently associated with this GTMCB.

Entry: IFFACA04 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Cutput: The hierarchy of ATLCBs is properly reordered. The light pen restart option in the GSP GACB is set as specified by the active ATLCB. Programmed function indicator lights associated with the GTMCB are lit as specified by the ATTNLITE field of the active ATLCB.

Operation: This consists of manipulating ATLCBs in the chain. For chaining information, see "Chaining" under "Introduction."

The new position in the chain for the specified ATLCB is determined as follows:

- If the "relattnlevel" parameter is not specified, the value of the "direction" parameter determines the new position.

- If the "relattnlevel" parameter is specified, a value is computed that indicates the direction (up or down) and the number of ATLCBs between the relative ATLCB and the ATLCB designated by the "attnlevel" parameter. This value is then added to the value of the "direction" parameter to produce a sum that positions the specified ATLCB. (Note: If the designated ATLCB is not associated with the specified GTMCB, control is returned to the calling program along with an appropriate return code.)

The "direction" value (or computed sum in cases when the "relattnlevel" parameter is specified) is tested for validity. This is accomplished by counting up or down the chain of ATLCBs in the desired direction until the top or bottom of the chain is reached. If this count is less than the "direction" value (or computed sum), control is returned to the calling program along with an appropriate return code.

Before the specified ATLCB is moved to its new position, attention information in the GSP GACB is placed in the active ATLCB ATTNMLEN queue. All attention sources for the GSP GACB are then disabled, thus preventing attentions from occurring while the ATLCBs are being moved. The ATLCB to be moved is then deleted from the chain.

Once the ATLCB is deleted from the chain, its ATTNBPID field is set to zero and the new point of insertion is determined. This is done by using the "direction" value (or computed value) for searching up or down the chain of ATLCBs. The value is decremented by one for each ATLCB reference until the value reaches zero. The specified ATLCB is then inserted into the chain at this new location.

Attention sources for the GSP GACB are now reenabled; the light pen restart option is set in the GSP GACB as specified in the ATTNFLGA field of the active ATLCB, and programmed function indicator lights are lit as specified in the ATTNLITE field of the active ATLCB. Control is then returned to the calling program.

CANCEL Key (Panic Key) Routine (Module Name IFFAHA09)

Chart: None

Function: Presents to the user the following options:

- Abnormally terminate the program and produce a dump.

- Abnormally terminate the program without a dump.

- Resume normal processing.

Entry: IFFAHA09 from an attention caused by the depression of the CANCEL key.

Exit: Abnormal termination via the ABEND macro instruction if the END key was depressed; or a return to the calling program if the CANCEL key was depressed.

Input: In register 1, the address of the panic GACB.

Output: A dump may be produced if the END key was depressed. This depends on the null variable. If the null variable is negative, the dump is produced. If the null variable is positive, no dump is produced.

Operation: After the registers are saved, work areas are set up, and a check is made to ensure that this routine was entered by a CANCEL key attention. If it was, the END key is enabled and an attention is awaited from either the END or CANCEL key. The following action is taken when an attention is detected:

- If from the END key, an ABEND macro instruction with appropriate completion code is issued to terminate the program. Whether the null variable is a negative or positive value determines whether or not a dump is produced.
- If from the CANCEL key, the END key is disabled.

SALRM (Module Name IFFACA13)

Chart: None

Function: Sets the audible alarm on the 2250.

Entry: IFFACA13 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: A single stroke of the audible alarm is sounded.

Operation: GCNTRL macro instruction is issued to create the order for sounding the audible alarm. The DCB address used for this macro instruction is obtained from the GTMGRDCB field of the associated GTMCB.

LIGHT PEN SUBROUTINES

These subroutines facilitate two-way communication between the GSP program and the 2250 operator through use of the light pen. The subroutines involve locating a position on the screen at which the light pen is pointed (LOCPN), and using the light pen to cause the program to move a tracking symbol from one screen location to another (BGTRK, RDTRK, and ENTRK).

LOCPN (Module Name IFFAGA01)

Chart: None

Function: Locates the coordinates on the screen to which the light pen is pointed.

Entry: IFFAGA01 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The coordinates of the light pen detect are placed in the user-specified variables.

Operation: A LOCPN GACB is created in the work area to accept the light pen attention. The screen scanning orders are placed in the GDOA of the system GDS via ORGEN and then into the buffer via EXEC to start the display. Approximately 1/2 second after the start of the display, a check is made to determine if a light pen attention occurred.

If no light pen attention had occurred, regeneration of the display is started and scanning continues.

If a light pen attention had occurred, the coordinates of the location of the light pen are converted to user units via CNVRT and are placed in the variables designated by the "xpos" and "ypos" parameters. RESET is called to remove the screen scanning orders from the buffer. Before control is returned to the calling program, the LOCPN GACB is disabled.

BGTRK (Module Name IFFAGA02)

Chart: None

Function: Displays tracking symbol at designated screen position.

Entry: IFFAGA02 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The tracking symbol is displayed on the screen.

Operation: The user-specified positioning coordinates are converted to raster units. If conversion was unsuccessful or if the coordinate values are off-screen, raster unit values corresponding to the center of the screen are assumed. The return code for scissoring is placed in the GSPARRAY field of the GSPCB, and processing continues.

Once the raster unit coordinate values are obtained, they are placed in the track routine orders. The orders are entered into the GDOA of the system GDS via ORGEN and then into the buffer to start the display via EXEC.

If the track routine orders are already in the buffer, the tracking symbol is repositioned by writing the new coordinates into the buffer. The GTMREPOS field in the GTMCB contains the buffer address of the coordinates for the tracking symbol.

ENTRK (Module Name IFFAGA04)

Chart: None

Function: Removes tracking symbol from the screen.

Entry: IFFAGA04 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: None

Operation: The GTMFLAGS field of the GTMCB is checked to see if the track routine orders are in the buffer. If they are, RESET is called to remove the orders from the buffer, and the indicator in the GTMCB is reset. Otherwise, control is immediately returned to the calling program.

RDTRK (Module Name IFFAGA03)

Chart: None

Function: Reads the current position of the tracking symbol from the buffer.

Entry: IFFAGA03 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The coordinates of the tracking symbol are placed in the user-specified variables.

Operation: The GTMFLAGS field of the GTMCB is checked to see if the tracking routine orders are in the buffer. If they are not, control is immediately returned to the calling program. Otherwise, the GTMREPOS field of the GTMCB indicates where the current position of the tracking symbol can be found in the buffer. The coordinates are converted to user units via CNVRT and placed in the variables designated by the "xval" and "yval" parameters. If CNVRT fails to execute successfully, control is returned to the calling program along with an appropriate return code, and no values are placed in the user's variables.

STATUS INFORMATION FUNCTIONS (MODULE NAMES IFFAJA01, IFFAJA02, IFFAJA03, AND IFFAJA04)

These functions return status information about the execution of GSP subroutines to the user's program. The information is placed in word 6 (contents of register 0) of the save area in the user's program. Note that this area is not the save area in the routine (Director Part 2) that actually passed control to the function.

The four functions are performed by three routines as follows:

- ITRC (module name IFFAJA01) returns GSP return code information to the calling program. Depending on the value of the "code" parameter, the following information is returned:

Value of "code"	Value Placed in Word 6
1, 2, 3, 4, 5	The return code if it is higher than or equal to the value of the "code" parameter. If the return code is zero, a value of zero is placed in Word 6.
9	Contents of corresponding additional information word in the GSPARRAY, providing the value in Byte 1 of the GSPARRAY is not zero.
9	Maximum negative value, if the value in Byte 1 of the GSPARRAY is zero.
Any other value	Maximum negative value.

- ITST (module name IFFAJA04) returns information identifying current options selected for the specified GDSCB. Depending on the value of the "optionsub" parameter, the following information is returned.

Value of "optionsub"	Value Placed in Word 6
1, 2	A value of 1-4, depending on the contents of the GDSDATMD field in the GDSCB.
3	A value of 1-3, depending on the contents of the GDSCURMD field in the GDSCB.
4	A value of 1-4, depending on the contents of the GDSCHARS field in the GDSCB.

- ITBP (module name IFFAJA02) and RTBP (module name IFFAJA03) return the actual or desired beam position (actual is different from desired if scissoring occurred) for the specified GDSCB for integer and real coordinates respectively. Depending on the value of the "info" parameter, the following information is returned:

Value of "info"	Value Placed in Word 6
1	The contents of the GDSXLAST field of the GDSCB converted to user units via CNVRT subroutine.
2	The contents of the GDSXCURR field of the GDSCB converted to user units via CNVRT subroutine.
3	The contents of the GDSYLAST field of the GDSCB converted to user units via CNVRT subroutine.
4	The contents of the GDSYCURR field of the GDSCB converted to user units via CNVRT subroutine.

Note: If the information requested is not in the mode requested, the maximum negative value is returned in Word 6 of the save area.

The entry point in each routine is the same as its module name. Entry is from Director Part 2. Exit is back to Director Part 2.

Input to each of these routines consists of the address of a work area in register 1. The first word of this area contains the address of the parameter list specified by the user in requesting the status information function. In the case of ITRC, the second word of the work area contains the address of the GSPARRAY.

Each routine tests the parameters for validity. An invalid parameter causes control to be returned to the calling program along with a value of zero in Word 6 (contents of registor 0) of the save area in the user's program.

STROKE TABLE CREATION AND USE

This section contains information about the system stroke table, the creation of a stroke table, and describes the control information placed in a stroke table. The subroutine (DFSTR) that changes a stroke table, and the subroutine (PLSTR) that uses a stroke table to create orders and data for displaying symbols are also described in this section.

System Stroke Table (Module Name IFFAHA16 Alias GSP01)

The GSP system stroke table resides in the link library (SYS1.LINKLIB). Its contents and use are described in the publication IBM System/360 Operating System: Graphic Programming Services for FORTRAN IV, Form C27-6932.

Creating a Stroke Table

The GSP programmer may create his own stroke table of addresses and space for defining the strokes. The method of creating a stroke table is also described in the publication IBM System/360 Operating System: Graphic Programming Services for FORTRAN IV.

Stroke Table Control Information

The first two words of any GSP stroke table contain control information which is either inserted or referred to whenever a stroke table is requested by the user.

The first word contains the identification of the stroke table. Once a stroke table has been loaded, this word is used to locate the stroke table specified in a call to DFSTR or PLSTR. When a stroke table is loaded, the value of the "table" parameter passed by the user is placed in this word. The value "01" identifies the GSP system stroke table.

The second word contains a chain address, which points to the main storage location where the next stroke table is loaded. The chain address is used to locate all stroke tables that are loaded after the first stroke table. When a stroke table is loaded, this word is set to zero. When the next stroke table is loaded, the load address returned by the LOAD macro instruction is placed in this word. If this word is zero, it indicates the end of the chain of strokes tables.

The address of the first stroke table loaded is placed in the GSPSTRTB field of the GSPCB. This stroke table is not necessarily the GSP system stroke table.

DFSTR (Module Name IFFAGA05)

Chart: None

Function: Replaces a symbol currently in a stroke table with another symbol specified by the user.

Entry: IFFAGA05 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The strokes that create the new symbol are placed in the stroke table area that contained the strokes for creating the old symbol.

Operation: If the specified stroke table is not yet in main storage, it is loaded and added to the chain of stroke tables. The new strokes in the user-specified array are then converted to the correct form (see IBM System/360 Operating System: Graphic Programming Services for FORTRAN IV, Form C28-6932) and are placed in the stroke table area defined by the "symbol" parameter one at a time, overlaying any previous information in that area. Control is returned to the calling program when all the strokes have been transferred from the user's array.

PLSTR (Module Name IFFAGA06)

Chart: GA

Function: Causes graphic orders and data to be generated for displaying a symbol, the strokes of which are in a stroke table.

Entry: IFFAGA06 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: Orders and data are generated and stored in the GDOA to display the desired symbols.

Operation: The stroke table referred to is located or brought into main storage via the LOAD macro instruction. It is added to the chain of stroke tables, if necessary. If the "xcoor" and "ycoor" parameters are provided, their values are scaled, scissored, and graphic orders and data are generated for positioning the beam to the specified location. Otherwise, the current beam position is used as the center point of the first symbol to be displayed.

PLSTR locates the strokes necessary to produce a symbol, and converts each stroke from the stroke table format into a vector to be displayed on the screen, taking into account the height, width, and orientation provided by the user. Scissoring is called to scissor each stroke. Data Generator is called to create the orders and data for displaying the stroke, which are then placed in the GDOA before control is returned to PLSTR. This process is repeated until the orders and data for producing all desired symbols have been placed in the GDOA. After each character has been generated, the beam is positioned to the center point of the next character (determined by the "spacing" parameter or by default).

If a symbol is referred to which has no strokes defined for it, the next symbol is processed. No spacing occurs.

If the element produced is not an update element and a level 1 key or correlation value is specified, the key is established and placed in the key table via Key Table Management.

If the element is an update element, Key Table Management is called to verify the validity of the key or correlation value. Update is called to obtain a temporary GDOA before any initial beam positioning is done, and Update is recalled after the orders and data for producing all desired symbols have been placed in the GDOA to write these orders to the buffer. Control is then returned to the calling program.

DIRECT ORDER GENERATION SUBROUTINE

The direct generation of graphic orders and data is accomplished through the use of the ORGEN subroutine.

ORGEN (Module Name IFFAGA07)

Charts: HA, HB

Function: Moves user-specified graphic orders and data into a GDS from a main storage array. The logic of the moved orders is not checked.

Entry: IFFAGA07 from Director Part 2.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: User-specified graphic orders and data are placed in the designated GDS.

Operation: Processing by ORGEN depends upon the input parameters that are passed to it.

If either "key" or "corrval" parameters, or both, are specified for a level 1 GDS, ORGEN establishes appropriate key and include/omit structures. The include/omit structure is stored in the GDOA preceding the graphic orders and data that are to be moved.

If it is an update call, Key Table Management is called to verify the validity of the key or correlation value.

If a key for a level 2 GDS is specified, ORGEN computes a new key and places it in the variable designated as the "key" parameter.

In any case, Update is called to furnish one or more temporary GDOAs to be used by ORGEN to manipulate the graphic orders and data passed by the calling program. The graphic orders and data are moved to the temporary GDOAs. If a GDOA is filled before all orders and data have been transferred, the set mode order for the next GDOA is established, and ORGEN moves orders and data to the next GDOA. If more than one temporary GDOA is obtained and this is not an update call, additional buffer is assigned via Buffer Management. The transfer orders from one GDOA to another are completed with the assigned buffer addresses.

Address manipulation is accomplished within the set of graphic orders and data that are now in the temporary GDOAs. The entire set of orders and data is searched for set mode orders. As each set mode order is encountered, the following is performed:

- Non-address-type orders remain unchanged and the search continues.

- Address-type orders that refer to a location within the array of orders and data are resolved by determining which GDOA contains the address referred to and then calculating the physical buffer address. This procedure is referred to as relocation.

- Address-type orders that refer to a location represented by X'FFFF' are resolved by translating the logical address in the corresponding key into a physical buffer address. The key is not checked for validity. However, it is checked to ensure that it is associated with the specified GDS.

If the call to ORGEN is not an update call, the contents of each temporary GDOA are moved, one GDOA at a time, to the original GDOA for the GDS. After the contents of each temporary GDOA have been transferred, Data Store is called to transfer data to the buffer assigned to that particular GDOA. Data Store is not called when the contents of the last GDOA have been moved. However, the OLP, LOGCURR, and LENLEFT fields of the original OACB are updated. (The user must call the EXEC subroutine to transfer this data to the buffer.)

ORGEN then frees the storage obtained for temporary GDOAs and OACBs, completes the key, places the key in the user's variatle (if specified), and proceeds to complete the include/omit structure via Data Store (if required). The key is added to the key table via Key Table Management.

In case of an update call, ORGEN completes the include/omit structure as above, and then calls Update to write the temporary GDOAs to the buffer and to release storage used for the temporary GDOAs and OACBs. If a correlation value is specified, ORGEN places this value in the appropriate field of the key associated with the element.

The GDSGRMOD field, which designates the current mode, is set to zero to indicate that a set mode order is required for the next call before control is returned to the calling program.

If an error is detected during the operation of ORGEN, an appropriate return code is placed in the GSPARRAY field of the GSPCB. Storage obtained for the temporary GDOAs and OACBs is freed. If buffer was acquired, Buffer Management is called to release this buffer. Control is then returned to the calling program.

CONVERTING COORDINATES SUBROUTINE

The CNVRT subroutine converts coordinates from user values to raster units and vice versa.

CNVRT (Module Name IFFAGA08)

Chart: JA

Function: Accomplishes the following:

- Converts real or integer user coordinates to integer raster units.

- Converts integer raster units to real or integer user coordinates.

Entry: IFFAGA08 from Director Part 2, LOCPN, BGTRK, RDTRK, or RQATN.

Exit: To the calling program.

Input: See "Input to Many GSP Subroutines."

Output: The converted information is placed in the user-specified variables.

Operation: The following operations are performed, based on the specified value of the "convert" parameter:

- If "convert" equals 1, input is assumed to be absolute, integer raster unit coordinates. If the GDSDATMD field of the GDSCB specifies real mode, the input values are converted to real. Then, these values are converted to user units and placed in the user-specified variables. If the GDSDATMD field does not specify real mode, the integer values are converted to user units and placed in the user-specified variables.

- If "convert" equals 2, input is assumed to be absolute and either real or integer, as specified in the GDSDATMD field. An appropriate parameter list is set up for Scaling, and the GDSDATMD field is set to force absolute mode before Scaling is called to convert the values. Output from the Scaling routine is absolute and integer. CNVRT places this output in the user-specified variables.

INTERNAL ROUTINES

Internal routines in GSP perform various services for GSP subroutines. These internal routines are described in detail according to the following categories:

- Flow Control Management
- Buffer Management
- Key Table Management
- Scaling and Scissoring of Input Data
- Data Generation, Data Storing, and Updating

FLOW CONTROL MANAGEMENT

When INDEV is called to initialize the 2250, Flow Control Management is used to establish and initialize the flow control table (FCT) in main storage and the flow control structure in a buffer section.

During the processing of the user's program, Flow Control Management updates the FCT and the flow control structure as required and terminates their use when the use of the 2250 is terminated.

Together, the FCT and the flow control structure manage the regeneration sequence of GDSs as specified in the user's program. The FCT is used to record the status (in-use or free) of each possible entry in the flow control structure. The flow control structure governs the transfer of control to GDSs within the buffer. Both are described in the paragraphs that follow.

Flow Control Table

The FCT consists of ten bytes of control information followed by a series of two-byte entries that contain the buffer addresses for each flow control entry in the flow control structure, including the system GDS. A total of 24 flow control entries is permitted for the user's GDSs. Figure 10 shows the format and contents of the FCT at the time it is first initialized.

	<-------------- 4 Bytes	-------------->
+ 0	Address of Free List (FCT + 12)	
+ 4	Length of In-use List (2)	
+ 8	Buffer Address	Buffer + 6 (System GDS)
+12	Buffer + 16	Buffer + 26
+16	Buffer + 36	Buffer + 46
+20	Buffer + 56	Buffer + 66
+24	Buffer + 76	Buffer + 86
+28	Buffer + 96	Buffer + 106
+32	Buffer + 116	Buffer + 126
+36	Buffer + 136	Buffer + 146
+40	Buffer + 156	Buffer + 166
+44	Buffer + 176	Buffer + 186
+48	Buffer + 196	Buffer + 206
+52	Buffer + 216	Buffer + 226
+56	Buffer + 236	Buffer + 246

Figure 10. Flow Control Table Upon Initialization

In this figure, the address of the free list is the location in the FCT of the first buffer address that is free for entering the next flow control entry. Since the system GDS is always the first one in the order of flow control, the first available flow control entry is located at FCT + 12. Each time a flow control entry is used, the address of the free list is incremented by two bytes. Each time a flow control entry is released, this address is decremented by two bytes. All in-use entries are always in the first part of the FCT. All free entries are always in the latter part of the FCT.

The length of the in-use list at FCT + 4 is the total length (in bytes) of the entries in the FCT (beginning at FCT + 10 to include the system GDS) that are pointing to buffer locations currently being used for flow control entries. Initially, this field contains two. Each time a GDS is assigned a flow control entry, the length is incremented by two bytes. When a GDS is deleted and its flow control entry is released, the length is decremented by two bytes. This length field determines if all 24 entries have been used.

The buffer address field contains the address of the buffer section assigned when Flow Control Management requested buffer for the flow control structure. The system GDS entry at FCT + 10 is never deleted until the FCT is freed (i.e., use of the 2250 is terminated). The rest of the buffer addresses act as pointers to the flow control entries. During the processing of the user's program, these addresses may become rearranged and will not necessarily be sequential. However, the in-use addresses will always be at the beginning of the FCT, followed by the free addresses.

Rearrangement occurs when the use of a GDS is terminated and its flow control entry is deleted, or when ORGDS is called. If the address of the flow control entry is not at the end of the in-use list when it is deleted, the remaining addresses in the in-use list are moved up two bytes in the table so that all the entries in the in-use list are together. The deleted flow control entry address becomes the first free entry in the FCT, and the others in the free list remain untouched. Figure 11 depicts an example of reordering in the FCT.

This example shows the FCT in-use entries when six GDSs have been initialized and included in the flow control structure. It also shows these same entries after the user has issued a call to ORGDS.

FCT after six GDSs have been initialized:

+12	Buffer + 16 GDS1	Buffer + 26 GDS2
+16	Buffer + 36 GDS3	Buffer + 46 GDS4
+20	Buffer + 56 GDS5	Buffer + 66 GDS6

CALL ORGDS(GDS5,GDS6,GDS3,GDS1,GDS2,GDS4)

+12	Buffer + 56 GDS5	Buffer + 66 GDS6
+16	Buffer + 36 GDS3	Buffer + 16 GDS1
+20	Buffer + 26 GDS2	Buffer + 46 GDS4

Figure 11. Reordering the FCT

Flow Control Structure

The flow control structure in the buffer section consists of six bytes that contain a GSRT order and a GTRU order to the system GDS flow control entry. The remaining 250 bytes of this buffer segment contain a series of ten-byte entries required for each GDS. The system GDS is always assigned the first entry in the flow control structure (located at buffer + 6). The user's flow control entries begin at buffer + 16.

When a GDS is called into use, an entry is made in the flow control structure that governs the light pen mode setting, the transfer of control to the graphic orders and data in the GDS, and the transfer of control to the appropriate flow control entry when the execution of the GDS is complete. Each of the ten-byte flow control entries contains a light pen mode setting order (i.e., GESD, GDPD, or GNOP2) in its first two bytes, a GTRU with the address of the GDS in the next four bytes, and a GTRU in the last four bytes. This last GTRU order contains either the address of the GSRT order at the beginning of the structure or the address of the next flow control entry in the structure. Figure 12 depicts the flow control structure.

Flow Control Management Routine (Module Name IFFAHA01)

Charts: KA, KB

Function: Manages the regeneration sequence of the GDSs in the buffer. To do so, it creates and maintains a flow control table (FCT) in main storage, and a flow control structure in a buffer section. The FCT and the flow control structure are described in detail in this section under "Flow Control Table" and "Flow Control Structure."

Flow Control Management performs four different functions depending on the codes passed to it as input, as follows:

Code 1 -- Causes the FCT and the flow control structure to be initialized, and a data event control block (DECB) to be built following the FCT.

Code 2 -- Causes the specified GDS to be added to the flow control structure and the FCT to be updated accordingly.

Code 3 -- Causes the specified GDS to be deleted from the flow control structure and the FCT to be updated accordingly.

Code 4 -- Causes the flow control structure to be deleted, and frees the storage acquired for the FCT and the DECB.

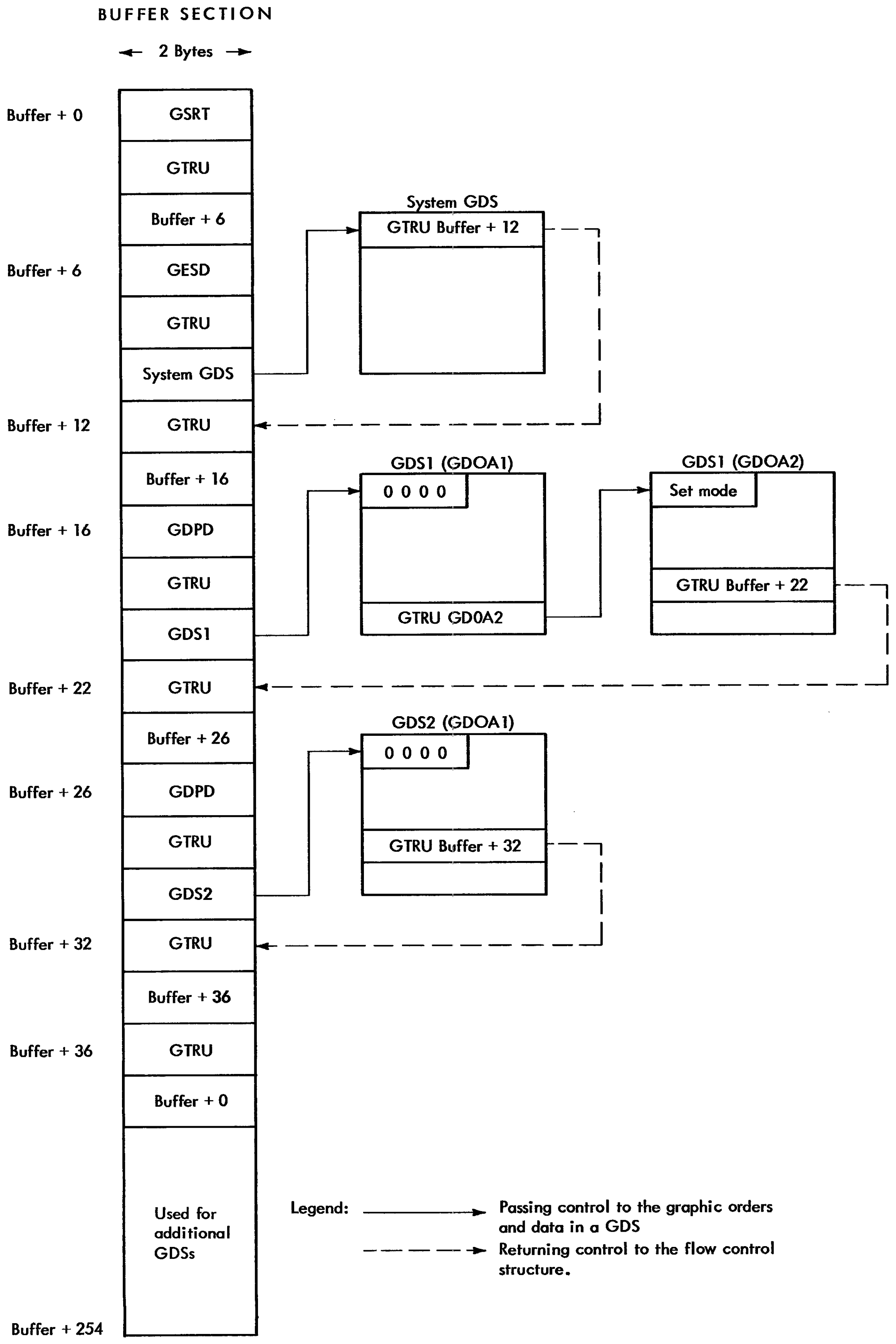


Figure 12. Flow Control Structure

Entry: IFFAHA01 from INDEV, Data Store, TMGDS, or TMDEV routines.

Exit: To the calling program.

Input: In register 1, the address of a parameter list which varies with the calling program. The first word of the parameter list for each of the four codes passed contains the actual code. The other fields in the parameter list contain information as follows:

Codes 1 and 4:
+4 Address of the GTMCB.

Code 2:
+4 Address of the GDSCB for the GDS to be added.
+8 Address of the buffer assigned to this GDS.

Code 3:
+4 Address of the GDSCB for the GDS to be deleted.

Output: Varies with the input code upon entry. See "Function" above.

Operation: Flow Control Management first tests the code passed to it as input to determine the function required. The services performed for a corresponding function are described in the paragraphs that follow.

Code 1 -- Buffer is acquired for the flow control structure. If this is completed successfully, storage for the FCT is obtained. If buffer is unavailable, control is returned immediately to the calling program with the appropriate return code in the GSPARRAY. If main storage is unavailable, the buffer is released and control is returned to the calling program with the appropriate return code in the GSPARRAY.

Next, the FCT is initialized, and a DECB (used for input/output operations) is built following the FCT. Graphic orders and data for initializing the flow control structure are written to the buffer.

Code 2 -- The address of the next free flow control entry is obtained from the FCT, and the FCT is updated. Graphic orders and data for the new flow control entry for the specified GDS are written to the buffer.

If all 24 flow control entries available to the GDSs have already been used, control is returned to the calling program with the appropriate return code.

Code 3 -- The GTMFCTBL field in the GTMCB is accessed to determine if the FCT is already freed (field is zero), in which case control is returned immediately to the calling program.

Otherwise, the GDSFCBUF field is accessed to determine if an entry was made in the flow control structure. If no entry exists, control is returned to the calling program.

When the address of the flow control entry exists, the FCT is updated and the removed entry is made available again. Then, the flow control entry is removed from the flow control structure in the buffer. This is done by writing an address to the preceding flow control entry that will branch either to the beginning of the flow control structure (if the removed entry was the last one in use), or around the flow control entry being removed.

Code 4 -- The buffer regeneration cycle is stopped, the main storage used for the FCT and the DECB is freed, and the GTMFCTBL and GTMDECB1 fields in the GTMCB are cleared.

Note: Buffer assigned to the flow control structure is not released by this routine, but is released by the Buffer Management routine which is called by TMDEV.

BUFFER MANAGEMENT

Assignment and release of buffer for the GDSs is handled by Buffer Management through the use of a buffer control table (BCT). Buffer Management consists of a buffer control table (BCT) and the Buffer Management routine, which creates, maintains, and terminates the BCT.

Buffer Control Table

The BCT consists of 128 three-byte entries, each of which is associated with a 256-byte buffer section. The INDEV subroutine obtains storage for the BCT following the GTMCB, and fills the entire table with zeros.

Buffer Management completes the three-byte BCT entries with information indicating the status of each buffer section as it is assigned or released. Each BCT entry corresponds to one buffer section (256 bytes). Entries are arranged in the table in ascending order as shown in Figure 13. The appropriate BCT entry number is found by dividing the buffer address by 256 (length of buffer). The buffer address can be obtained by multiplying the entry number by 256.

Generally, the BCT entry is divided into three one-byte fields which contain (1) the identification code that associates it with the appropriate GDSCB, (2) the pointer to the next entry in the BCT for the same GDSCB, and (3) a <u>count</u> of the number of assignments made for this GDSCB. Specifically, the three-byte BCT entry contains information as follows:

- <u>If a buffer section is divided into two 128-byte subsections</u>, the first byte of the entry contains a code that identifies the GDSCB associated with the first subsection. The second byte contains a code that identifies the GDSCB associated with the second subsection. The third byte always contains zeros. If either of the subsections is unassigned, a code of X'80' is placed in the byte associated with that subsection. The code placed in the entry to identify each GDSCB ranges from X'81' through X'FF'.

- <u>If one 256-byte buffer section is assigned</u>, the first byte of the entry contains a code that identifies the GDSCB associated with this buffer section. The code ranges from X'01' through X'7F'. The second byte contains a pointer to itself in the form of the entry number in the BCT. The third byte contains a zero indicating that it is the first section assigned.

- <u>If another buffer section is obtained for the same GDSCB</u>, the same identification code is placed in the first byte of the corresponding entry. The previous entry assigned to this GDSCB is updated to point to this entry by placing this entry number into the second byte of the previous entry. Then, this entry is completed by setting the second byte to point to itself, and then adding one to the value in the third byte of the previous entry, and placing this sum in the third byte of the current entry. This serves as a count of the number of sections assigned to this GDSCB. Additional buffer sections assigned are not necessarily contiguous.

- <u>If multiple buffer sections are assigned at one time to one GDSCB</u>, the code for the GDSCB is placed in the first byte of each entry that corresponds to the buffer sections assigned. The second byte in the first entry is a pointer to itself (until additional buffer is assigned to this GDSCB), and the third byte is zero, indicating that this is the first multiple section assigned. The second byte in the other entries contains a pointer (entry number) to the first entry, and the third byte in the other entries contains a X'FF' to indicate that they are part of a multiple assignment. Each multiple assignment consists of contiguous 256-byte buffer sections. However, additional multiple sections are not necessarily contiguous with the other multiple sections assigned to a GDS.

<u>Note:</u> The entry number multiplied by three results in the displacement of any given entry from the beginning of the BCT.

Figure 13 shows an example of entries that may be made in the buffer control table by Buffer Management. The example depicts three buffer sections divided into 128-byte subsections, the assignment of non-contiguous buffer sections to one GDSCB, the assignment of more than one multiple section to a GDSCB, and an unassigned buffer section.

<u>Buffer Management Routine (Module Name IFFAHA02)</u>

<u>Charts</u>: KC, KD

<u>Function</u>: Obtains buffer for the various GDSs, releases all or part of the buffer associated with a GDS, releases all buffer associated with a 2250, and keeps information as to which GDS is assigned which buffer sections.

<u>Entry</u>: IFFAHA02 from Data Store, TMGDS, TMDEV, ORGEN, and RESET routines.

<u>Exit</u>: To the calling program.

<u>Input</u>: In register 1, the address of a work area, the first word of which points to a parameter list. This parameter list varies depending on the service desired. The first word of the parameter list always contains the code that specifies the service desired. The services and their respective parameter lists are:

- Allocate 128-byte buffer subsection to a GDS being initialized.

  +0 ("code") 1
  +4 ("gdscb") Address of the GDSCB being initialized.

- Allocate 256-byte buffer section(s) to a GDS being initialized.

  +0 ("code") 2
  +4 ("gdscb") Address of the GDSCB being initialized.
  +8 ("gdslength") Size of buffer to be allocated.

		BCT Entries			
Buffer Address	Entry Number In BCT (Hexadecimal)	First Byte	Second Byte	Third Byte	Comments
0	X'00'	X'86'	X'92'	X'00'	Two 128-byte subsections assigned
256	X'01'	X'36'	X'05'	X'00'	First 256-byte section assigned
512	X'02'	X'42'	X'07'	X'00'	First section of a 768-byte assignment
768	X'03'	X'42'	X'02'	X'FF'	Second section of a 768-byte assignment
1024	X'04'	X'42'	X'02'	X'FF'	Third section of a 768-byte assignment
1280	X'05'	X'36'	X'0C'	X'01'	Second 256-byte section assigned to GDSCB with code '36'
1536	X'06'	X'80'	X'AA'	X'00'	First subsection unassigned, and second subsection assigned
1792	X'07'	X'42'	X'07'	X'01'	Second multiple assignment
2048	X'08'	X'42'	X'07'	X'FF'	Second 256-byte section of multiple assignment
2304	X'09'	X'42'	X'07'	X'FF'	Last 256-byte section of multiple assignment
2560	X'0A'	X'36'	X'0A'	X'03'	Fourth section assigned to GDSCB with code '36'
2816	X'0B'	X'96'	X'80'	X'00'	First subsection assigned, second subsection unassigned
3072	X'0C'	X'36'	X'0A'	X'02'	Third section assigned to GDSCB with code '36'
3328	X'0D'	X'00'	X'00'	X'00'	Unassigned buffer section

Figure 13. Example of Buffer Control Table Entries

- Allocate an additional 256 bytes of buffer, or more, to a GDS which has buffer already allocated.

  +0 ("code") 3
  +4 ("gdscb") Address of the GDSCB.

- Release a 128-byte buffer subsection allocated to a GDS to be terminated.

  +0 ("code") 4
  +4 ("gdscb") Address of the GDSCB to be terminated.

- Release all 256-byte buffer sections allocated to a GDS to be terminated.

  +0 ("code") 5
  +4 ("gdscb") Address of GDSCB to be terminated.

- Release one or more 256-byte buffer sections most recently allocated to a GDS that is to remain in use.

  +0 ("code") 6
  +4 ("gdscb") Address of GDSCB.
  +8 ("bufloc") Address of buffer after which sections are to be released. This section is not released.

- Release all buffer currently assigned to a GTMCB being terminated.

  +0 ("code") 7
  +4 ("gtmcb") Address of GTMCB to be terminated.

Output: Varies according to the "code" specified in input parameter list. See "Operation" for this routine.

Operation: Depending on the value of the "code" parameter, control is transferred to the appropriate routine within this module. (The services associated with each "code" parameter are described under "Input" above.)

The operations of the service routines are described in the following paragraphs under their respective codes.

Code 1 -- A 128-byte buffer subsection is obtained by searching the BCT. If the first byte of a BCT entry indicates an assigned subsection, the second byte of the entry is investigated.

If no unassigned subsection is found, buffer is obtained. The buffer address is saved, the BCT entry number associated with that buffer address is placed in the GDSBCTEL field of the GDSCB, and the second byte of that entry is set to X'80'.

If an unassigned subsection is found by searching the BCT, the buffer address is saved and the BCT entry number is placed in the GDSBCTEL field of the GDSCB.

The identification code, which is to be placed in the BCT entry and the GDSBCTID field of the GDSCB, is now obtained in the following manner. The value of the GTMBCTSM field in the GTMCB is compared to X'7E'. If it is less than or equal to X'7E', this field plus a X'81' forms the code. A X'01' is then added to this field so that the next code can be computed.

When the GTMBCTSM field compares higher than X'7E', all the possible identification codes have already been used once. However, only 24 GDSs can be associated with each 2250 at any one time. Thus, some of the identification codes must be available for use again since at least one associated GDS must have been terminated. When the GDS is terminated, its code is removed from the BCT entry. Every possible code is now compared to every code still assigned in the BCT until one that is not in use is obtained.

Once an identification code is determined, it is placed in the appropriate fields of the BCT and the GDSCB. The buffer address of the allocated subsection is placed in register 1, and control is returned to the calling program.

Code 2 -- A 256-byte buffer section, or contiguous multiple thereof, is obtained. If insufficient buffer is available, control is returned to the calling program along with an appropriate return code. Otherwise, the buffer address is saved and the BCT entry number is placed in the GDSCB.

The identification code, which is to be placed in the BCT entry and the GDSBCTID field of GDSCB, is now obtained in the following manner. The value of the GTMBCTLG field in the GTMCB is compared to X'7E'. If it is less than or equal to X'7E', a one is added to the field and this sum is used as the code.

When the GTMBCTLG field compares higher than X'7E', the code is determined in the same manner that it was for Code 1 when the GTMBCTSM field was higher than X'7E'.

Once an identification code is determined, it is placed in the appropriate fields of the BCT and GDSCB. The number of 256-byte buffer sections obtained is determined, the entry number of the first associated BCT entry is placed in the GDSCB, and the BCT is appropriately completed (see "Buffer Control Table"). Control is then returned to the calling program.

Code 3 -- An additional 256-byte buffer section, or contiguous multiple thereof, is obtained. If a 128-byte buffer subsection was previously obtained for this GDS, control is returned to the calling program along with an appropriate return code. Otherwise, the amount of buffer desired (from the length specified in the GDSGDOAL field of the GDSCB) is obtained.

The number of buffer sections is determined, and appropriate entries are completed in the BCT (see "Buffer Control Table"). Register 1 contains the buffer address. Control is then returned to the calling program.

Code 4 -- A 128-byte buffer subsection is released, and the BCT entry is set to X'80'. If the byte designating the other subsection contains X'80', the buffer is released and the entire BCT entry is set to zeros. The GDSBCTEL and GDSBCTID fields in the GDSCB that contain the BCT entry number and the identification code associated with the buffer just released are also set to zeros. Control is then returned to the calling program.

Code 5 -- All buffer assigned to the GDS being terminated is released. After the BCT entry number is saved, the GDSBCTEL and GDSBCTID fields in the GDSCB are set to zeros. The first buffer address is obtained by multiplying by 256, and the pointer to the next buffer section to be released is contained in the BCT entry. Each time the buffer associated with an entry is released, the BCT entry is set to zeros. When all buffer assigned to this GDSCB is released, control is returned to the calling program.

Code 6 -- All buffer allocated to a GDS

more recently than the buffer section specified by the "bufloc" parameter is released. If the buffer to be released is not the last section obtained, all sections obtained after it are released in the same manner as they were for Code 5. However, the GDSCB fields cleared for Code 5 are not cleared because some buffer is still associated with the GDSCB. When all specified buffer is released, control is returned to the calling program.

Code 7 -- All buffer currently assigned to a 2250 being terminated is released. The entire BCT is set to zeros. As each GDSCB associated with the specified 2250 is referred to, the GDSBCTEL and GDSBCTID fields are set to zeros. Control is then returned to the calling program.

KEY TABLE MANAGEMENT

When standard GSP keying and correlating features are used, known as level 1, Key Table Management is called upon to create and maintain a key table for each GDS that has keys and/or correlation values specified in the program. The key table is used for storing keys associated with key and correlation values as they are assigned within each GDS. If the programmer uses his own correlation scheme, known as level 2, Key Table Management is bypassed.

Key Table

The key table is a list of all keys and correlation values assigned for a level 1 GDS. The table occupies one or more 496-byte blocks of storage. These blocks are chained together, but may or may not be in contiguous locations. The address of the key table is entered in the GDSKEYTB field of the GDSCB.

Each block in the key table contains 12 bytes (three words) of control information followed by a series of 12-byte key entries (called keys). There may be up to 40 entries in a block. The last word of each block is not used. The key table format and contents are shown in Figure 14.

The control information is set up as follows:

Word 1 -- Contains the chaining address, which is zero in the last block acquired for the key table. If the address is not zero, it contains the address of the next block of storage used to continue the key table.

```
      <-------------- 4 Bytes ------------->
      ┌─────────────────────────────────────┐
  +0  │ Chain address (zero if last block)  │
      ├─────────────────────────────────────┤
  +4  │ Next available space for key entry  │
      ├─────────────────────────────────────┤
  +8  │ Address of last word in block       │
      ├─────────────────────────────────────┤
 +12  │ Key 1 (12 bytes of information)     │
      │                                     │
      ├─────────────────────────────────────┤
 +24  │ Key 2 (12 bytes of information)     │
      │                                     │
      ├─────────────────────────────────────┤
                        •
                        •
                        •
      ├─────────────────────────────────────┤
 +480 │ Key 40 (12 bytes of information)    │
      │                                     │
      ├─────────────────────────────────────┤
 +492 │              Not used               │
      └─────────────────────────────────────┘
```

Figure 14. The Key Table

Word 2 -- Contains the address of the next unused space in the block. If there is no space available for key entries in the block, this address points to the last word in the block. If space is available in the block, this address indicates the location where the next key can be added. It also indicates that the end of the table has been reached when a search for key or correlation value is made.

Word 3 -- Contains the address of the last word of the block (the unused word). When the address in word 3 is the same as the address in word 2, the block is full and another block of storage must be acquired for additional keys.

GSP Keys

Keys consist of 12 bytes (three words) of information about the element, sequence, buffer subroutine, or buffer subroutine linkage associated with the specified key or correlation value.

If a level 1 GDS is being used, and an update is not requested, certain GSP subroutines compose key values and set up key entries for both the keys and the correlation values that are specified by the user. A correlation value causes a standard key to be structured with the correlation value placed in the third word of the key entry. If update of an element is specified, this key is used to locate the element to be updated. Keys are entered in the key table and removed from the key table as described under "Key Table Management Routine."

Each key consists of the following three words:

Word 1 -- The first two bytes contain a value indicating the amount of buffer space (in bytes) used to store the graphic orders and data for the element, sequence, or buffer subroutine associated with this key. The next two bytes contain a value indicating the logical buffer start address, formed by determining the total amount of storage previously used for storing graphic orders and data for this GDS. This address is not the same as the physical buffer address.

Word 2 -- The first byte contains eight flag bit settings that describe the input data associated with the key. The second byte, used only for a single request, consists of data that is to be inserted in the second byte of the image generation order, to indicate blanking or unblanking of the beam. The third byte contains a count of characters if text input is associated with the key. The fourth byte contains flags required for updating.

Word 3 -- Contains the correlation value, if any, specified by the user. Otherwise, it contains zero.

Table 5 lists the information that is placed in the three words of a GSP key. It also shows the arrangement of this information in the various fields.

Table 5. GSP Key Information

Word No.	Bit No.	Contents
Word 1	0 - 15	Buffer length
	16 - 31	Logical buffer start address
Word 2	0	0 = Single entry 1 = Multiple entry
	1	0 = First order is not set mode 1 = First order is set mode
	2 - 4	Output is: 000 = Line 001 = Character 010 = Point 011 = Subroutine linkage 100 = Sequence 101 = Buffer subroutine 110 = End Order Sequence order 111 = ORGEN
	5	0 = Omit 1 = Include
	6 - 7	00 = X,Y are absolute 01 = X,Y are incremental
	8 - 15	One byte of data
	16 - 23	Count of characters
	24	0 = Not part of buffer subroutine 1 = Part of buffer subroutine
	25	0 = Not part of a sequence 1 = Part of a sequence
	26 - 31	Reserved
Word 3	0 - 31	Correlation value (if specified)

Key Table Management Routine (Module Name IFFAHA03)

Chart: KE

Function: Builds and maintains a key table, and upon request:

- Locates a key in the key table.
- Adds a key to the key table.
- Resets the key table from a specified key.
- Deletes an entire key table.

Entry: IFFAHA03 from INCL/OMIT, PLINE/PPNT, STPOS/MVPOS, PSGMT, PTEXT, STEOS, BGSEQ/BGSUB, ENSEQ, ENSUB, LKSUB, RESET, ICURS, GSPRD, ORGEN, PLSTR, and TMGDS subroutines.

Exit: To the calling program.

Input: In register 1, the address of the following parameter list:

+0 GDSCB address.
+4 Address of a code that designates the function to be performed by this execution of Key Table Management. The code is one of the following:
    1 = add a key
    2 = search for a key
    3 = reset from a key
    4 = delete the key table
    5 = search for a correlation value
+8 Address of key or correlation value (or zero).

Output: For the various codes, as follows:

Code 1 -- If no key table exists for this GDS, one is built and its address is placed in the GDSKEYTB field of the GDSCB. The key is entered in the table, and control information is updated.

Codes 2 or 5 -- In register 1, the address of the key in the key table.

Code 3 -- Starting with the specified key, the keys for a GDS are made inaccessible. Any blocks of storage beyond the one containing the specified key are freed. Control information is updated.

Code 4 -- Storage used for the key table is released, and the key table address is removed from the GDSCB.

Operation: The code in the input parameter list is tested, and, depending upon the code designated, the following operations are performed:

Code 1 -- The routine tests for a key table address in the GDSCB. If there is none, storage is obtained and the key table is initialized. If the key table block is full, additional storage is obtained. The first 12 bytes of this storage are set up with control information (see "Key Table"), and the beginning address of this storage block is placed in the first word of the preceding storage block in the key table. The key is placed in the key table, the address in Word 2 of the control portion of the block is updated to reflect the next available entry address, and control is returned to the calling program.

If a key table has already been set up and is not full, the routine simply places the key in the key table at the next available address indicated in the control portion of the block, and increments the next available entry address in Word 2.

Code 2 -- The key table address is obtained from the GDSCB. If it is zero, an error return is made immediately to the calling program. If it is not zero, a search is made for the key. The search consists of comparing the key value from the input parameter list to each key in the key table until a matching value is found. Then, the address of the key within the key table is placed in register 1, and control is returned to the calling program.

Code 3 -- The key table address is obtained from the GDSCB. If it is zero, an immediate error return is made to the calling routine. If it is not zero, a search is made for the specified key. When the key is found in the key table, its address is placed in Word 2 of the control portion making it the next available entry. Any blocks of storage that were used for the key table that follow this block are released. Control is returned to the calling program.

Code 4 -- The key table address is set to zero in the GDSKEYTB field of the GDSCB, and all blocks of storage used for the key table are released. Control is returned to the calling program.

Code 5 -- The steps performed are the same as described under Code 2, except that the correlation value is used for the search to find the key. The address of the first key located within the key table that contains the same correlation value is placed in register 1. Control is returned to the calling program.

SCALING AND SCISSORING OF INPUT DATA

Scaling and scissoring of input data is accomplished by two routines which are described in the paragraphs that follow. These routines are called during the generation of graphic orders and data by the image generation subroutines.

Scaling Routine (Module Name IFFAHA06)

Chart: KF

Function: Performs the following:

- Scales user input coordinates to raster units (integer).

- Converts any real number to integer.

Entry: IFFAHA06 for scaling. IFFAHA15 for converting real numbers to integer.

Exit: To the calling program.

Input: For entry at IFFAHA06, in register 1, the address of a work/save area. The first word of this area contains the address of the following parameter list:

+ 0 Address of the GDSCB.

+ 4 Address of the GSPARRAY.

+ 8 A data array consisting of eleven full words. The first four words of this array are ignored. The next four words contain the input coordinates as follows:

+24 SCAXABS Absolute value of x-coordinate, or zero if it is incremental.

+28	SCAYABS	Absolute value of y-coordinate, or zero if it is incremental.
+32	SCAXINC	Incremental value of x-coordinate, or zero if it is absolute.
+36	SCAYINC	Incremental value of y-coordinate, or zero if it is absolute.
+40		These two words are ignored.
+48	SCAPTCNT	Count of points scaled to date.

For entry at IFFAHA15, in register 1, the number to be converted from real to integer.

Output: Both the incremental and absolute values are computed and placed in the data array fields (SCAXABS, SCAXINC, SCAYABS, SCAYINC). If entry was at IFFAHA15, register 1 contains the results of the conversion from real to integer. For both entries, register 15 contains zero if scaling and conversion were completed successfully, or four if scaling or conversion was unsuccessful. The GSPARRAY field is set as required.

Operation: Scaling computes the scaled value of the input coordinates according to their input data mode as determined by the GDSDATMD field of the GDSCB. Based on values found in various GDSCB fields, the computations are made as indicated in Table 6. If the input coordinates are real, they are first computed with real numbers, and then converted to integer. If the conversion is unsuccessful, the GSPARRAY is appropriately set to specify a scaling error, and control is returned to the calling program.

If the scaled values for the x-coordinate are successfully obtained, the scaled values for the y-coordinate are computed. This is accomplished in the same manner as for the x-coordinate (shown in Table 6), except that the corresponding GDSCB fields dealing with the y-coordinate are used in the computation. When all scaled values have been obtained, control is returned to the calling program.

Scissoring Routine (Module Name IFFAHA07)

Chart: KG

Function: Scissors data according to the option specified in the most recent call to the SSCIS subroutine.

Entry: IFFAHA07 from PLINE/PPNT, PSGMT, STPOS/MVPOS, PTEXT, IDPOS, and PLSTR subroutines.

Exit: To the calling program.

Input: In register 1, the address of a work/save area. The first word of this area contains the address of the input/output array shown in Table 7.

Output: Positioning coordinates (absolute and incremental, blanked or unblanked mode) are placed in the input/output array if either the current or any previous data was within screen limits (see "Input"). The appropriate return code is placed in the GSPARRAY field of the GSPCB and in register 15. The GDSXCURR, GDSYCURR, GDSXLAST, and GDSYLAST fields in the GDSCB are updated as necessary.

Operation: Coordinates for data that fall within the prescribed limits are placed in the unblanked area (+24 through +36) of the input/output array (see "Input"). Coordinates for any data that falls outside the prescribed limits are placed in the blanked area (+8 through +20) of that array unless image generation is not to be continued when this occurs. If image generation is not to be continued, only the coordinates that fall within the screen limits are passed in the unblanked area of the input/output array; no coordinates are placed in the blanked area.

All fields in the GDSCB mentioned under "Output" are updated except when no data has been plotted within screen limits. In this case, the GDSXCURR and GDSYCURR fields of the GDSCB are not updated.

Figure 15 shows examples of the application of scissoring options. The boundary refers to the prescribed limits (either GDS or screen) as specified in the GDSSCISS field of the GDSCB. The values of these limits for the GDS are also specified in the GDSCB. If the GDS limits are off the screen for any boundary, then Scissoring assumes the screen limits for that particular boundary. These limits determine whether or not there is an intersection with any boundary, and how many intersections occur (0, 1, or 2).

In Figure 15, points and intersections are indicated by dots and labeled appropriately. In all cases, plotting takes place in the direction from the previous point to the current point. Broken lines indicate the portions of the lines that are not displayed. Solid lines indicate those lines that are plotted and displayed. The orders that are generated in each case are also indicated.

Table 6. Formulas Used in Scaling

Mode for X-Coordinate	Values Placed in Data Array

Integer, absolute	SCAXABS = $\frac{\text{GDSXVIUR - GDSXVILL}}{\text{GDSUVLUR - GDSUVLLL}}$ times SCAXABS (input) - GDSUVLLL + GDSXVILL
	SCAXINC = SCAXABS - GDSXCURR

Integer, incremental	SCAXINC = $\frac{\text{GDSXVIUR - GDSXVILL}}{\text{GDSUVLUR - GDSUVLLL}}$ times SCAXINC (input)
	SCAXABS = SCAXINC + GDSXLAST
	SCAXINC[1]= SCAXABS - GDSXCURR

Real, absolute	SCAXABS[2]= $\frac{\text{GDSXVRUR - GDSXVRLL}}{\text{GDSUVLUR - GDSUVLLL}}$ times SCAXABS (input) - GDSUVLLL + GDSXVRLL
	SCAXINC = SCAXABS[2] - GDSXCURR

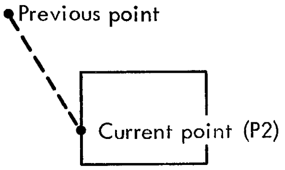
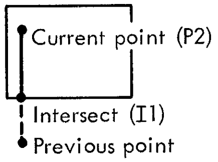
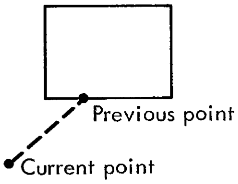
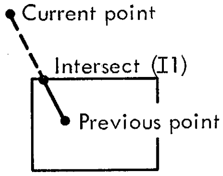
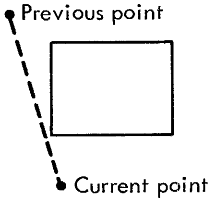
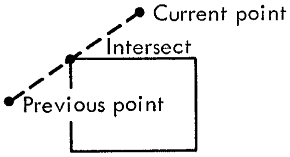
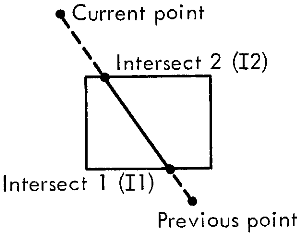
Real, incremental	SCAXINC[2]= $\frac{\text{GDSXVRUR - GDSXVRLL}}{\text{GDSUVLUR - GDSUVLLL}}$ times SCAXINC (input)
	SCAXABS = SCAXINC[2] + GDSXLAST
	SCAXINC[1]= SCAXABS - GDSXCURR

[1] Indicates this is the final output value passed to the calling program. Computations are performed in the order shown.

[2] Indicates this is an integer value. It is first computed with real numbers, and then converted to integer.

Table 7. Input/Output Data Array for Scissoring

Displacement in Bytes	Input	Output
+ 0	Address of GDSCB	Same as input
+ 4	Address of return code array	Same as input
+ 8	Zero	X-coordinate, absolute, blanked
+12	Zero	Y-coordinate, absolute, blanked
+16	Zero	X-coordinate, incremental, blanked
+20	Zero	Y-coordinate, incremental, blanked
+24	X-coordinate, absolute (to be scissored)	X-coordinate, absolute, unblanked
+28	Y-coordinate, absolute (to be scissored)	Y-coordinate, absolute, unblanked
+32	X-coordinate, incremental (to be scissored)	X-coordinate, incremental, unblanked
+36	Y-coordinate, incremental (to be scissored)	Y-coordinate, incremental, unblanked
+40	Total x-increment to date	Same as input
+44	Total y-increment to date	Same as input
+48	Number of points scissored to date	Same as input

Number of Intersections	0	1	2
If the previous point is outside the boundary, and the current point is within or on the boundary: NOTE: 0 intersections indicates that the current point is directly on boundary.			Not applicable
Orders Generated	GDV P2X, P2Y, B GDV P2X, P2Y, U	GDV I1X, I1Y, B GDV P2X, P2Y, U	Not applicable
If the previous point is within or on the boundary, and the current point is outside the boundary: NOTE: 0 intersections indicates that the previous point is directly on boundary.			Not applicable
Orders Generated	None	GDV I1X, I1Y, U	Not applicable
If the previous and the current points are both outside boundaries: NOTE: 1 intersections indicates that it intersects directly on the boundary.			 Current point Intersect 2 (I2) Intersect 1 (I1) Previous point
Orders Generated	None	None	GDV I1X, I1Y, B GDV I2X, I2Y, U

NOTE: If both the previous point and the current point are within boundaries, no scissoring occurs. The order generated would be GDV P2X, P2Y, U.

Legend:

X	= x-coordinate of specified point or intersection.
Y	= y-coordinate of specified point or intersection.
B	= blanked beam.
U	= unblanked beam.
---	Indicates portion of line that is not displayed.
——	Indicates that line is plotted and displayed.

Figure 15. Examples of Scissoring

DATA GENERATION, DATA STORING, AND UPDATING

The Data Generator routine uses information passed to it by image generation subroutines and produces the graphic orders and data for displaying appropriate images. The Data Store routine places the orders and data produced by Data Generation in the GDOA so that they can be written to the buffer and executed by the 2250. The Update routine handles functions made necessary when the update facility is applied. These three routines are described in the paragraphs that follow.

To assist in understanding the structures of the various elements, sequences, and buffer subroutines that are created during image generation, there are four figures included in this section. These figures are:

- Figure 16, which depicts the include and omit structures for single keyed elements, and for multiple keyed and non-keyed elements.
- Figure 17, which depicts the buffer subroutine structure and linkage, both keyed and non-keyed.
- Figure 18, which depicts the structure of keyed sequences in both include and omit status.
- Figure 19, which depicts the GDOAs and their logical and physical buffer addresses.

Data Generator Routine (Module Name IFFAHA04)

Chart: KH

Function: Generates the graphic orders and data necessary for performing the function requested by a call to an image generation subroutine. Calls Data Store to place orders and data in the GDOA for the GDS associated with the element to be produced.

Entry: IFFAHA04 from PLINE/PPNT, STPOS/MVPOS, PSGMT, PTEXT, or PLSTR subroutines.

Exit: To the calling program.

Input: In register 1, the address of the following parameter list:

Displacement (In Bytes)	Contents
+0	A value indicating the length of the available buffer. This value is zero if the element is not an update element.
+2	A value indicating the logical buffer start address of the entire element.
+4	Information about the data to be generated. This word is called the data information word. Its contents are depicted in Table 8.
+8	The correlation value of the element or sequence associated with this cycle of Data Generator. This value is zero if the element is not correlated.
+12	The address of the user's variable where the key is to be placed. This field is zero if the element is not keyed.
+16	The address of the associated GDSCB.
+20	The address of the return code array in the GSPCB.
+24	A value indicating the x-coordinate, in absolute form, for positioning the beam.
+28	A value indicating the y-coordinate, in absolute form, for positioning the beam.
+32	A value indicating the x-coordinate, in incremental form, for positioning the beam.
+36	A value indicating the y-coordinate, in incremental form, for positioning the beam.
+40	A value indicating the absolute x-coordinate requested.
+44	A value indicating the absolute y-coordinate requested.
+48	A value indicating the incremental x-coordinate requested.
+52	A value indicating the incremental y-coordinate requested.
+56	A value indicating the total

of all x-coordinate increments requested. This will be zero upon entering the first cycle.

+60 A value indicating the total of all y-coordinate increments requested. This will be zero upon entering the first cycle.

Table 8. Contents of Data Information Word Used by Data Generator

Bit Number	Interpretation of Setting
0	0 = Single request 1 = Multiple requests
1	Reserved
2-4	Output will be: 000 = Line 001 = Character 010 = Point 011 = Subroutine linkage 100 = Sequence 101 = Buffer subroutine 110 = End-order-sequence 111 = ORGEN
5	0 = Omit status 1 = Include status
6-7	00 = X,Y are absolute 01 = X,Y are incremental 11 = X,Y are optimized
8-15	Reserved
16-17	00 = First cycle(request) 01 = Not first or last 11 = Last cycle(request)
18	0 = Data not keyed, level 1 1 = Data keyed, level 1
19	0 = Data not correlated 1 = Data correlated
20	0 = Beam blanked (off) 1 = Beam unblanked (on)
21	0 = Repositioning element required 1 = No repositioning element required
22-23	Reserved
24	0 = Not part of subroutine 1 = Part of subroutine
25	0 = Not part of sequence 1 = Part of sequence
26-31	Reserved

Output: Graphic orders and data for the associated GDS in a form that can be executed by the 2250.

Operation: Data Generator tests the data information word (Table 8) to determine the type of output to be produced. Operation of Data Generator varies for keyed or non-keyed elements according to the type of output desired and the type of cycle designated by the calling routine. A cycle is defined as one execution of Data Generator resulting from a request by another routine.

Single or multiple requests for the execution of Data Generator may be issued by the calling program, as determined by the "count" parameter, or if the increment is such that it requires more than one incremental set of coordinates, the entry is treated as a multiple request. Each time a cycle of Data Generator is requested, bits are set by the calling routine to indicate whether this is the first, the last, or an intermediate cycle.

During any cycle of Data Generator, if the code returned in register 15 from Data Store indicates that the attempt to store orders was unsuccessful, Data Generator immediately returns control to its calling program, passing that same code.

The following paragraphs describe the operation for keyed and non-keyed elements. If an element has a correlation value, this routine considers it to be keyed.

Non-Keyed Element

1. Preliminary processing for first cycle of the routine:

The graphic orders to set the mode and to position the beam are generated from data in the input parameter list, and then stored via Data Store before the basic cycle operation is begun.

2. Basic operation for each cycle of the routine:

For absolute or incrememtal output, graphic orders and data for displaying the desired image are generated and stored. The type of image to be displayed and whether the beam is to be blanked or unblanked is determined from the data information word. The x- and y-coordinates are determined from data in the input parameter list.

For optimized output, the length of the increment is checked, and the shortest output is generated. Determination of the optimal mode depends upon the mode previously established. If this previously established mode was absolute, the incremental line or point must be two bytes to cause the mode to be changed to incremental; if it is more than two bytes, the output mode remains absolute. If the previously established mode was incremental, the incremental line or point must be greater than six bytes to cause the mode to be changed to absolute; otherwise, output mode remains incremental. After the mode has been determined for this cycle, lines or points are determined from data in the input parameter list and graphic orders and data are generated and stored in the GDOA.

Keyed Element
Single request processing:

Output mode is checked. If it is absolute, the graphic orders and data to display a line or point and any mode setting required are generated from information contained in the input parameter list.

If the output mode is incremental, the length of the incremental line or point is computed and the graphic orders and data are generated and stored as described for absolute mode.

If the output mode is optimized, the length of the increment is checked and the shortest output is generated. If the incremental length is greater than two bytes, orders are created to generate a line or point in absolute mode. If the incremental length is two bytes, orders are created to generate a line or point in incremental mode.

Multiple request processing:

1. Preliminary processing for the first cycle of the routine:

For the first element of multiple requests, the structure required to place the element in include or omit status is built, and stored in the GDOA via Data Store. This structure consists of a GTRU order (for omit status) or a GNOP4 order (for include status) to an unresolved address, and bits are set in the input parameter list for Data Store to indicate that there is an address to be resolved during the last cycle of Data Generator. See Table 9, Code X'01' and Figure 16 in this section.

2. Basic operation for each cycle of the routine:

This is the same as described for non-keyed elements, basic operation.

3. Post-processing for last cycle of the routine:

A GTRU to transfer around the orders for repositioning the beam is set up. Current mode settings are turned off in the GDSGRMOD field. Graphic orders for repositioning the beam are generated for all calls except STPOS/MVPOS. The unresolved address mentioned under first cycle processing is resolved to the address of the repositioning order, if applicable.

Data Store Routine (Module Name IFFAHA05)

Charts: KJ, KK

Function: For various routines, performs the following functions:

- Stores data in the GDOA and writes the GDOA to the buffer on an overflow.
- Resolves any unresolved addresses that have been flagged by the calling routine (see Table 9).
- Causes buffer to be assigned to this GDS if the STATUS field in the OACB indicates that none has been assigned previously.
- Causes an entry to be made in the flow control structure if buffer is assigned to the GDS by this execution of Data Store.

Entry: IFFAHA05 from BGSEQ/BGSUB, ENSEQ, ENSUB, LKSUB, STEOS, PTEXT, Data Generator, or ORGEN routines.

Exit: To the calling program.

Input: In register 1, the address of a work/save area, containing information as follows:

+0 Address of the GDSCB.

+4 Address of the GSPARRAY field in the GSPCB where the return code is placed.

+8 Flags which indicate to the routine that some action must be taken in resolving addresses. The codes used to set these flags are explained in Table 9.

+10 Length (in bytes) of the data to be stored.

+12 Graphic orders and data to be stored. (Length is nine full words.)

+48 Register save area.

Table 9. Data Store Input Flag Codes

Code	Orders Passed to Data Store
X'01'	GTRU/GNOP4 unresolved address (1)
X'02'	No order; (1) will be resolved Address(3) will be resolved at ENSUB time
X'03'	GTRU unresolved address (2) Address (1) will be resolved Address (2) is resolved when the repositioning order is stored
X'04'	GTRU unresolved address (3) GTRU zero address GDRD
X'08'	GMVA subroutine address + 6, unresolved address (5) GTRU subroutine address + 8 GPDI Address of (5) will be resolved
X'14'	GTRU unresolved address (3) GTRU zero address GTRU/GNOP4 unresolved address (4) GDRD Address of (4) will be resolved
X'18'	GTRU/GNOP4 unresolved address (6) GMVA subroutine address + 6, unresolved address (7) GTRU subroutine address + 8 GPDI Addresses (6) and (7) will be resolved
X'20'	No order; perform overflow function for ORGEN

Note: The numbers within parentheses are referenced under "Operation" where the resolving of addresses is explained.

Output: Graphic orders and data are moved to the GDOA. If Data Store is unsuccessful in any of its functions, a return code of four is placed in register 15.

Operation: Data Store tests the STATUS field in the OACB to determine if buffer has yet been assigned to the specified GDS. If no buffer has been assigned, Buffer Management is called to assign buffer. Then, Flow Control Management is called to create a flow control structure entry for the specified GDS. These two functions are bypassed if the specified GDS has had buffer assigned to it before this execution of Data Store. Should Data Store be unsuccessful during any of its functions, control is returned immediately to the calling program with the appropriate code in register 15.

Data Store resolves any unresolved addresses as explained later, and moves the graphic orders and data to the GDOA.

If the GDOA is full or if the remaining space is not large enough for all the graphic orders and data to be moved (the last four bytes of the GDOA are reserved), the GDOA is filled with as much of the data as it can contain and a GTRU order to the buffer section for the next GDOA is placed in its last four bytes. Buffer is obtained for the next GDOA by calling Buffer Management. The buffer address that is returned is placed in the GTRU order. Any unresolved addresses in the OACB fields (+32 through +47) are then noted as being in the buffer. A test is made to determine if four bytes of data from the CRSA point in the GDOA must be saved in the GDSCB. The graphic orders and data are written to the buffer from the CRSA point on; if four bytes of data were saved, it is written from CRSA + 4. This is known as the overflow function.

Finally, the next OACB is accessed and initialized for the next GDOA, and the appropriate set mode order is determined and placed in the first two bytes of the next GDOA. Any remaining graphic orders and data are moved to the next GDOA, and control is returned to the calling program.

Figure 19 depicts the GDOAs with logical and physical buffer addresses.

Unresolved addresses may be resolved either in the GDOA or in the buffer. If the address to be resolved is for an order in the buffer, field BUF1 or BUF2 in the OACB will contain the buffer location of the order containing the unresolved address. If the address to be resolved is for an order in the GDOA, field CPU1 or CPU2 of the OACB will contain the GDOA location of the order containing the unresolved address. These fields in the OACB are set up at the time the order is placed either in the GDOA or in the buffer with an unresolved address.

Resolving of addresses is accomplished, based on codes in the flag field of the input work area shown in Table 9. Numbers within parentheses refer to Table 9.

X'01' -- Indicates entry from BGSEQ, PTEXT, ORGEN, or Data Generator (first call for multiple requests) where a key is used. No addresses are resolved. See Figures 17 and 19 for the structure of multiple requests.

X'02' -- Indicates entry from ENSEQ, ENSUB, PTEXT, ORGEN, or Data Generator (last call for multiple requests) where a key is used. Address (1) shown in Table 9 is resolved to the current location. Address (3) is resolved at ENSUB time to the current location. Address (2) is resolved to the current location when a repositioning order is stored.

X'03' -- Indicates entry from Data Generator for a keyed element, where a GTRU has been set up preceding the positioning element (multiple request). Address (1) shown in Table 9 is resolved to the current location.

X'04' -- Indicates entry from BGSUB for a level 2 buffer subroutine. Address (3) is resolved at ENSUB time with a code of X'02'. This GTRU transfers control around the subroutine so that it is not executed in line.

X'08' -- Indicates entry from LKSUB for non-keyed linkage to a buffer subroutine. See Figure 17 for the structure of buffer subroutine linkage. Address (5) is resolved to the current location + 10, which is the return address from the buffer subroutine.

X'14' -- Indicates entry from BGSUB for a keyed buffer subroutine. Address (4) shown in Table 9 is resolved to the current location minus four, which points to the return transfer. See Figure 17.

X'18' -- Indicates entry from LKSUB for keyed linkage to a buffer subroutine. Addresses (6) and (7) shown in Table 9 are resolved to the current location + 14, which is the return address from the buffer subroutine.

X'20' -- No addresses are resolved, but it indicates entry from ORGEN, and only part of the overflow function is performed.

Update Routine (Module Names IFFAHA13 and IFFAHA14)

Chart: None

Function: Obtains storage for a temporary GDOA and OACB for storing graphic orders and data created for an update, or writes these orders and data from the temporary GDOA to the 2250 buffer.

Entry: IFFAHA13 or IFFAHA14 from STPOS/MVPOS, PSGMT, PLINE/PPNT, PTEXT, PLSTR, or ORGEN subroutines.

Exit: To the calling program.

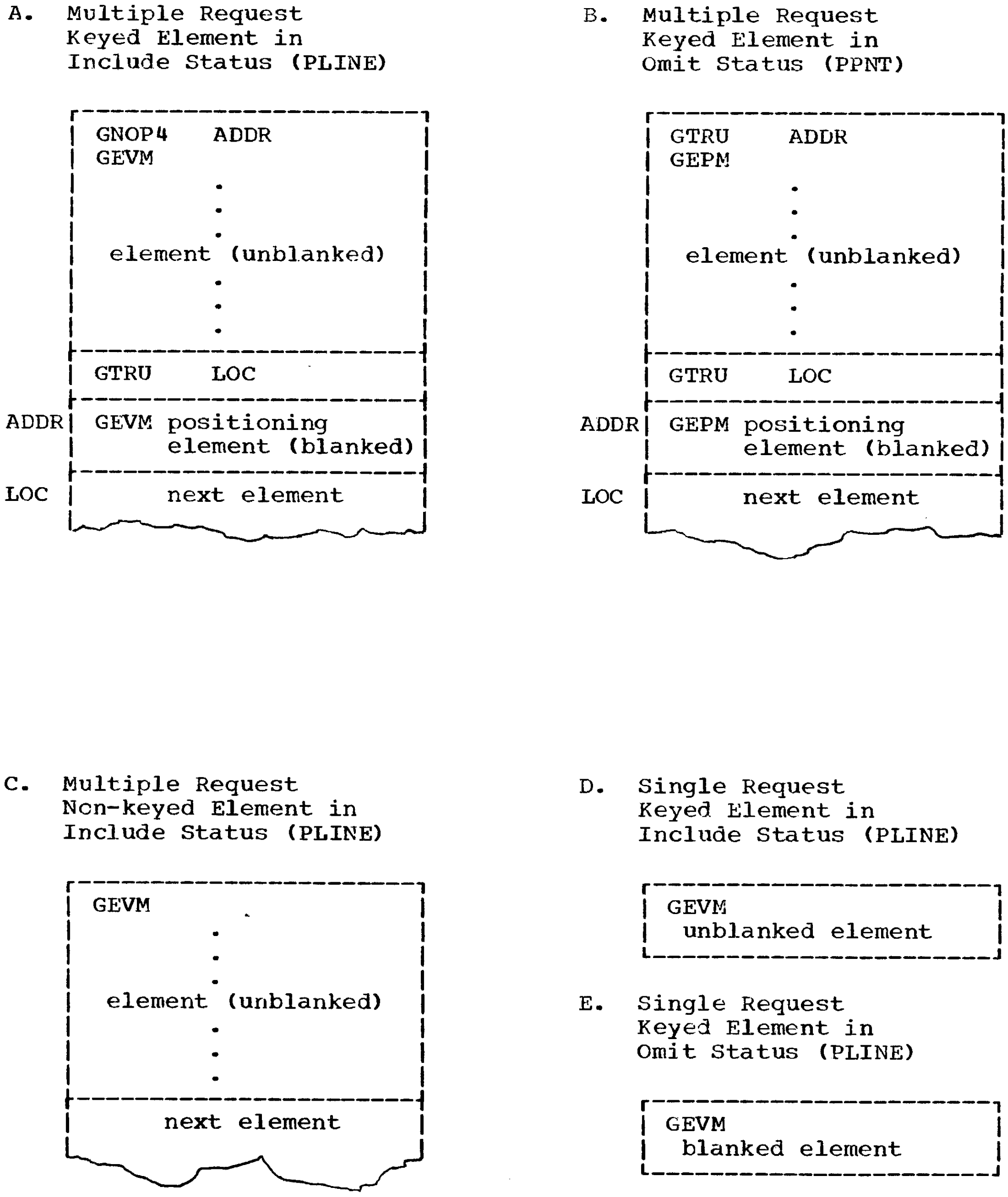
Input: In register 1, the address of a work/save area. The first word of this area contains the address of the following input parameter list:

+0	Address of the key in the key table.
+4	Three full words reserved.
+16	Address of the GDSCB.
+20	Address of the GSPARRAY field in the GSPCB.
+24	Eleven full words reserved.

Output: For entry IFFAHA13, a temporary GDOA and its OACB are constructed and the address of the temporary OACB replaces the address of the regular OACB in the GDSCB. For entry IFFAHA14, the contents of the temporary GDOA are written to the buffer, the storage occupied by the temporary GDOA and the OACB is freed, and the GDSAOACB field in the GDSCB is restored to its previous condition.

Operation: At IFFAHA13, storage needed for temporary GDOAs and OACBs is obtained. Fields in the OACB are set to point to this temporary GDOA. The original OACB address in the GDSCB is saved when the replacement OACB address is inserted. Control is returned to the calling program.

At IFFAHA14, if the length of the update orders and data is less than the length of the element(s) being replaced, a GNOP2 order (if only two bytes are left) or a GTRU order (if four bytes or more are left) to the next element is placed in the temporary GDOA. Contents of the temporary GDOA are then written to the buffer, and the storage used for the temporary GDOAs and OACBs is freed. The GDSAOACB field in the GDSCB is restored, and control is returned to the calling program.



A. Multiple Request
Keyed Element in
Include Status (PLINE)

B. Multiple Request
Keyed Element in
Omit Status (PPNT)

C. Multiple Request
Ncn-keyed Element in
Include Status (PLINE)

D. Single Request
Keyed Element in
Include Status (PLINE)

E. Single Request
Keyed Element in
Omit Status (PLINE)

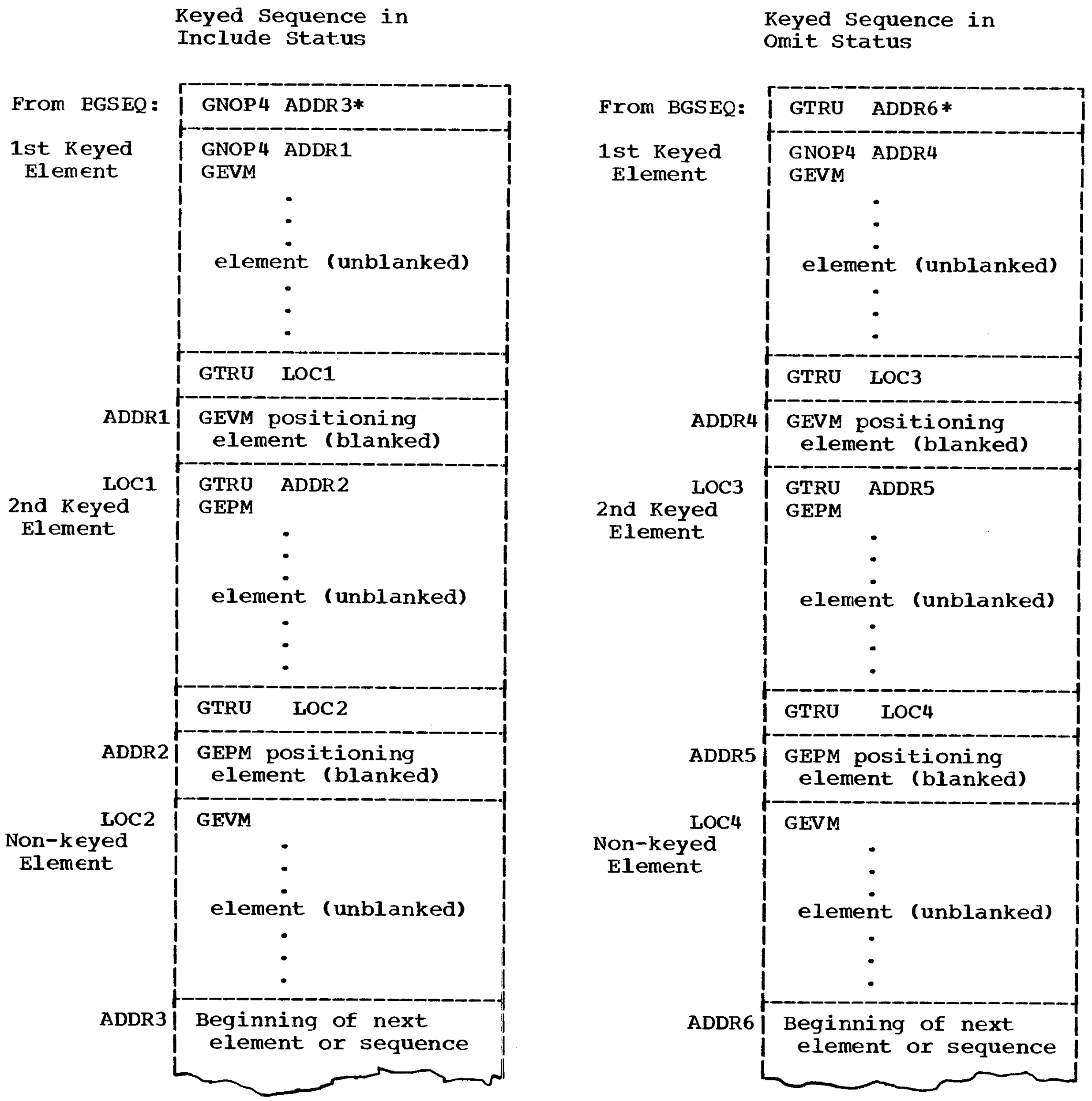
Figure 16. Include/Omit Structure by Type of Element

```
                A.  Structure of Level 1 Keyed
                    Buffer Subroutine

                   ┌──────────────────────────┐
From EGSUB: SUBR│ GTRU  B                    │ This address is resolved by a call to ENSUB.
            A   │ GTRU  RET                  │ This address is resolved by a call to LKSUB.
                │ GTRU  A                    │ (Omit status)
                │     or                     │
                │ GNOP4 A                    │ (Include status)
                │ GDRD                       │
                ├────────────────────────────┤
                │            .               │ Note:  The include/omit structure is omitted
                │            .               │ for a non-keyed or level 2 buffer
                │            .               │ subroutine.
                │         elements           │
                │            .               │
                │            .               │
                │            .               │
                ├────────────────────────────┤
From ENSUB:     │ GTRU  SUBR+4               │
                ├────────────────────────────┤
            B   │ Next element beyond        │
                │ buffer subroutine.         │
                └────────────────────────────┘
```

```
B.  Linkage for                     C.  Linkage for
    Level 1 Keyed                       Level 1 Non-keyed or
    Buffer Subroutine                   Level 2
                                        Buffer Subroutine
    ┌──────────────────────┐
    │GTRU    RET           │ (Omit)      ┌──────────────────────┐
    │    or                │             │GMVA   SUBR+6,RET     │ No include/omit
    │GNOP4   RET           │ (Include)   │GTRU   SUBR+8         │ structure is set up.
    │GMVA    SUBR+6,RET    │             ├──────────────────────┤
    │GTRU    SUBR+8        │         RET │GPDI                  │
    ├──────────────────────┤             └──────────────────────┘
RET │GPDI                  │
    └──────────────────────┘
```

Figure 17. Buffer Subroutine Structure and Linkage



*These addresses are resolved by a call to ENSEQ.

Figure 18. Examples of Structure of Keyed Sequences

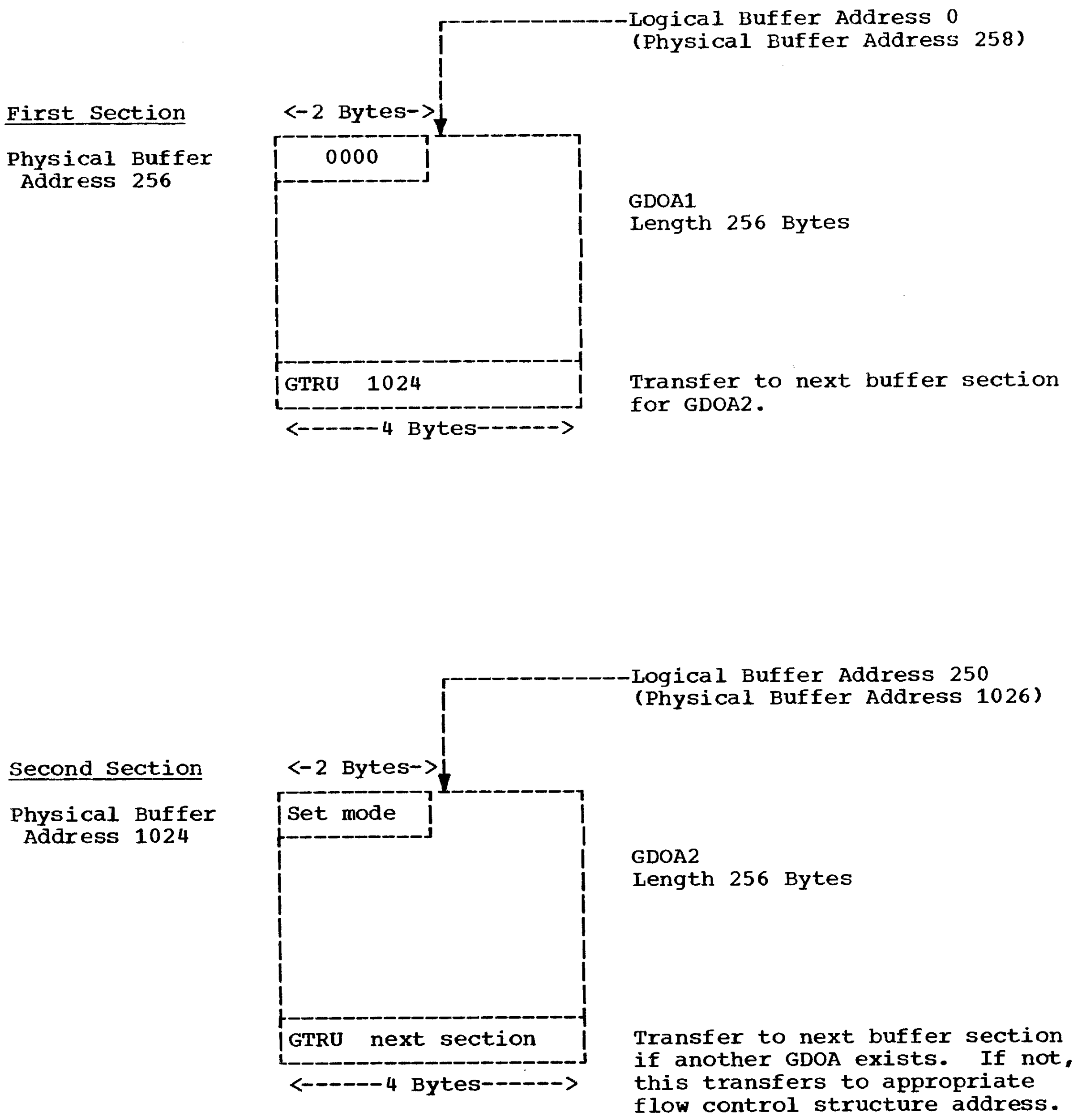

Logical Buffer Address 0
(Physical Buffer Address 258)
First Section
Physical Buffer Address 256
<-2 Bytes->
0000
GDOA1
Length 256 Bytes
GTRU 1024
Transfer to next buffer section for GDOA2.
<------4 Bytes------>
Logical Buffer Address 250
(Physical Buffer Address 1026)
Second Section
Physical Buffer Address 1024
<-2 Bytes->
Set mode
GDOA2
Length 256 Bytes
GTRU next section
Transfer to next buffer section if another GDOA exists. If not, this transfers to appropriate flow control structure address.
<------4 Bytes------>

Figure 19. GDOAs with Logical and Physical Buffer

CHARTS

This section contains autocharts for many of the GSP routines described in the preceding section. Charts are not included for all GSP routines, but only for the routines where it is believed the charts are an aid toward better understanding. The charts should be used in conjunction with the detailed descriptions of corresponding routines in the preceding section. The charts are ordered alphabetically (according to identification number) in the sequence in which the routines are described. Refer to Appendix D for an explanation of the symbols used on the autocharts.

Chart AA. INDEV Subroutine

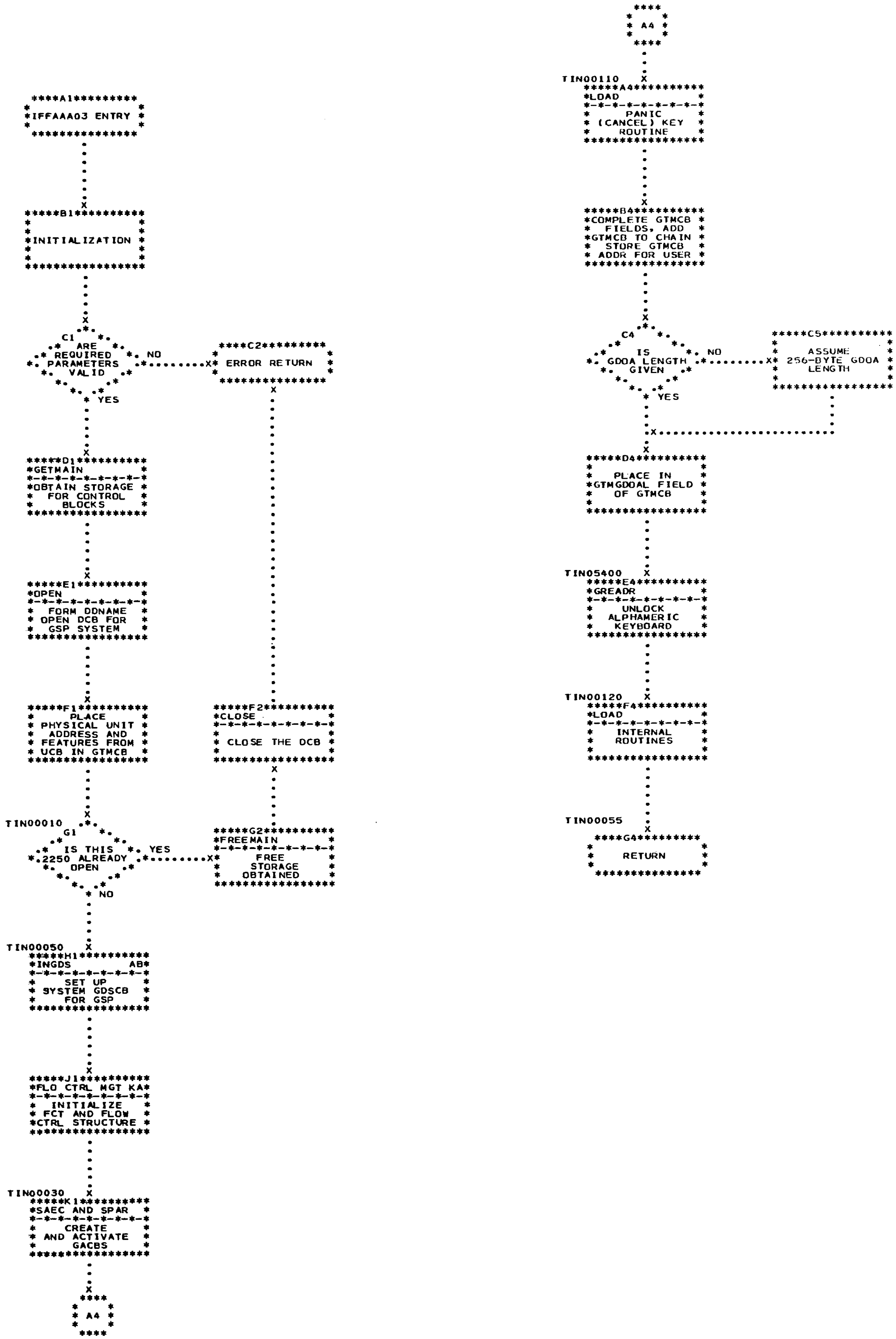

A1
IFFAAA03 ENTRY
B1
INITIALIZATION
C1
ARE REQUIRED PARAMETERS VALID
NO
YES
C2
ERROR RETURN
D1
GETMAIN
OBTAIN STORAGE FOR CONTROL BLOCKS
E1
OPEN
FORM DDNAME OPEN DCB FOR GSP SYSTEM
F1
PLACE PHYSICAL UNIT ADDRESS AND FEATURES FROM UCB IN GTMCB
F2
CLOSE
CLOSE THE DCB
TIN00010
G1
IS THIS 2250 ALREADY OPEN
YES
NO
G2
FREEMAIN
FREE STORAGE OBTAINED
TIN00050
H1
INGDS AB
SET UP SYSTEM GDSCB FOR GSP
J1
FLO CTRL MGT KA
INITIALIZE FCT AND FLOW CTRL STRUCTURE
TIN00030
K1
SAEC AND SPAR
CREATE AND ACTIVATE GACBS
A4
A4
TIN00110
A4
LOAD
PANIC (CANCEL) KEY ROUTINE
B4
COMPLETE GTMCB FIELDS. ADD GTMCB TO CHAIN STORE GTMCB ADDR FOR USER
C4
IS GDOA LENGTH GIVEN
NO
YES
C5
ASSUME 256-BYTE GDOA LENGTH
D4
PLACE IN GTMGDOAL FIELD OF GTMCB
TIN05400
E4
GREADR
UNLOCK ALPHAMERIC KEYBOARD
TIN00120
F4
LOAD
INTERNAL ROUTINES
TIN00055
G4
RETURN

Chart AB. INGDS Subroutine

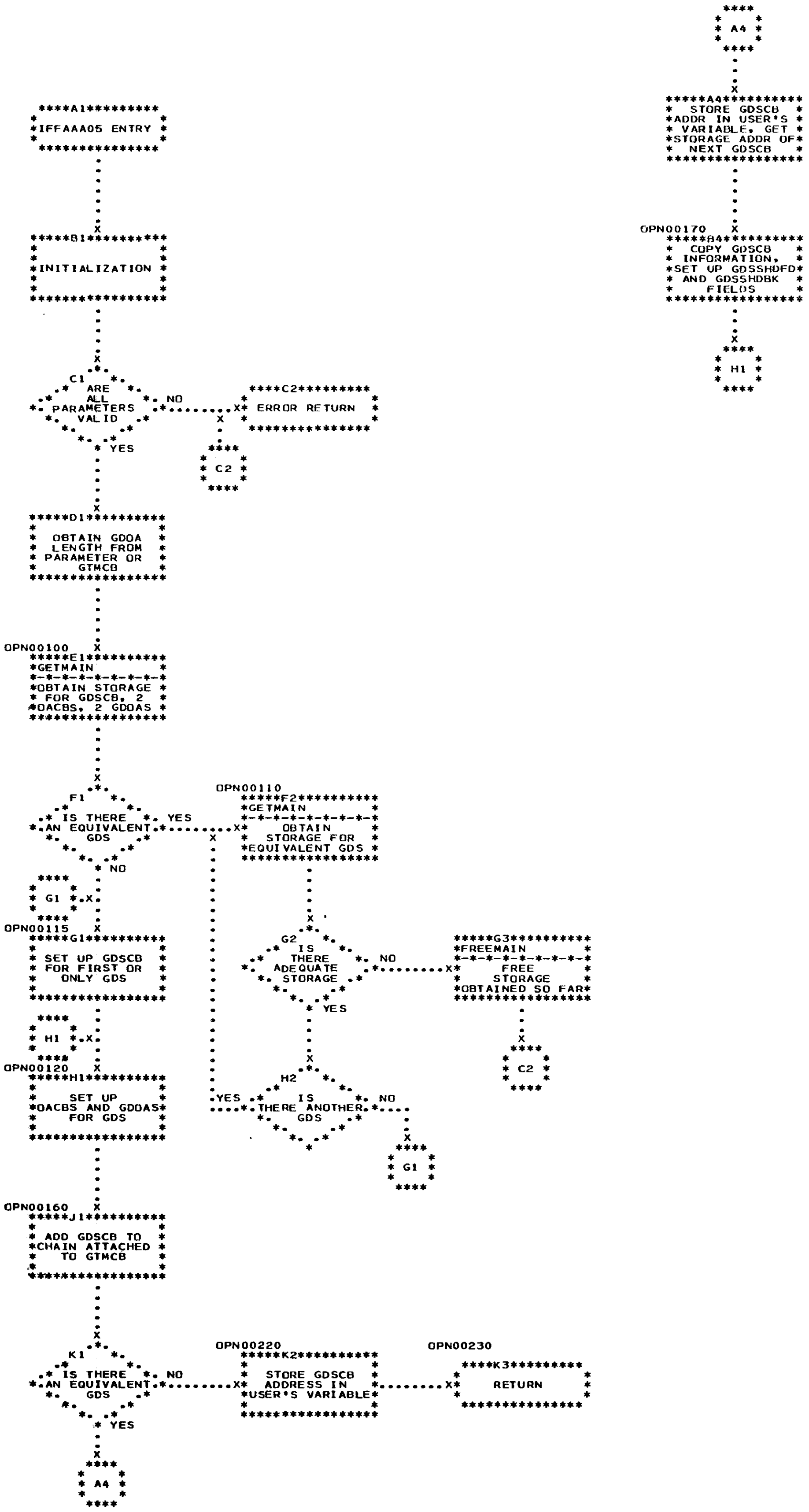

A4
A4
STORE GDSCB ADDR IN USER'S VARIABLE. GET STORAGE ADDR OF NEXT GDSCB
OPN00170
B4
COPY GDSCB INFORMATION. SET UP GDSSHDFD AND GDSSHDBK FIELDS
H1
A1
IFFAAA05 ENTRY
B1
INITIALIZATION
C1
ARE ALL PARAMETERS VALID
NO
YES
C2
ERROR RETURN
C2
D1
OBTAIN GDDA LENGTH FROM PARAMETER OR GTMCB
OPN00100
E1
GETMAIN
OBTAIN STORAGE FOR GDSCB, 2 OACBS, 2 GDOAS
F1
IS THERE AN EQUIVALENT GDS
YES
NO
OPN00110
F2
GETMAIN
OBTAIN STORAGE FOR EQUIVALENT GDS
G1
OPN00115
G1
SET UP GDSCB FOR FIRST OR ONLY GDS
G2
IS THERE ADEQUATE STORAGE
NO
YES
G3
FREEMAIN
FREE STORAGE OBTAINED SO FAR
C2
H1
OPN00120
H1
SET UP OACBS AND GDOAS FOR GDS
H2
IS THERE ANOTHER GDS
YES
NO
G1
OPN00160
J1
ADD GDSCB TO CHAIN ATTACHED TO GTMCB
K1
IS THERE AN EQUIVALENT GDS
NO
YES
OPN00220
K2
STORE GDSCB ADDRESS IN USER'S VARIABLE
OPN00230
K3
RETURN
A4

Chart AC. TMGDS Subroutine

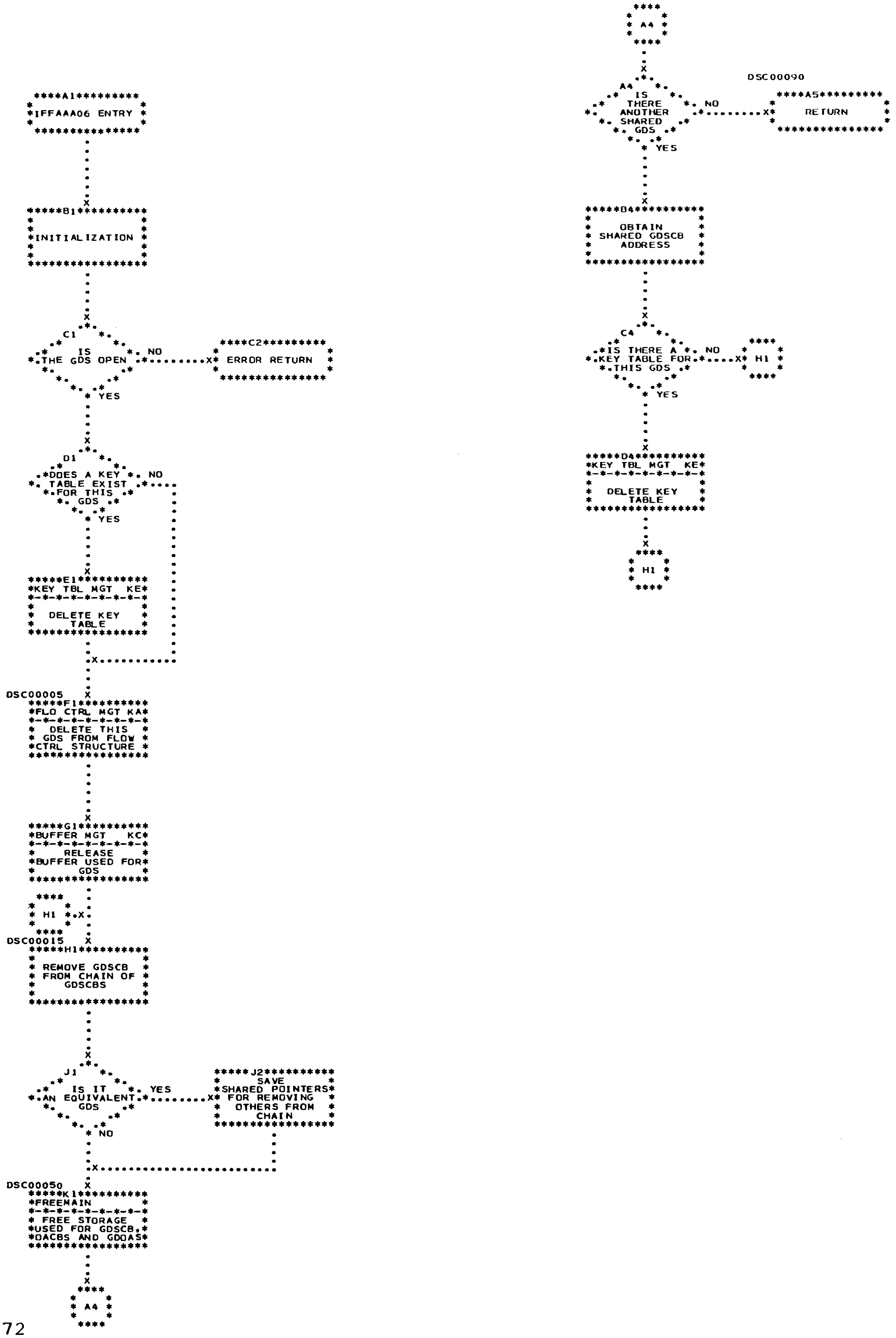

A1
IFFAAA06 ENTRY
B1
INITIALIZATION
C1
IS THE GDS OPEN
NO
C2
ERROR RETURN
YES
D1
DOES A KEY TABLE EXIST FOR THIS GDS
NO
YES
E1
KEY TBL MGT KE
DELETE KEY TABLE
DSC00005
F1
FLO CTRL MGT KA
DELETE THIS GDS FROM FLOW CTRL STRUCTURE
G1
BUFFER MGT KC
RELEASE BUFFER USED FOR GDS
H1
DSC00015
H1
REMOVE GDSCB FROM CHAIN OF GDSCBS
J1
IS IT AN EQUIVALENT GDS
YES
J2
SAVE SHARED POINTERS FOR REMOVING OTHERS FROM CHAIN
NO
DSC00050
K1
FREEMAIN
FREE STORAGE USED FOR GDSCB, OACBS AND GDOAS
A4
A4
A4
IS THERE ANOTHER SHARED GDS
NO
DSC00090
A5
RETURN
YES
B4
OBTAIN SHARED GDSCB ADDRESS
C4
IS THERE A KEY TABLE FOR THIS GDS
NO
H1
YES
D4
KEY TBL MGT KE
DELETE KEY TABLE
H1

Chart AD. TMDEV Subroutine

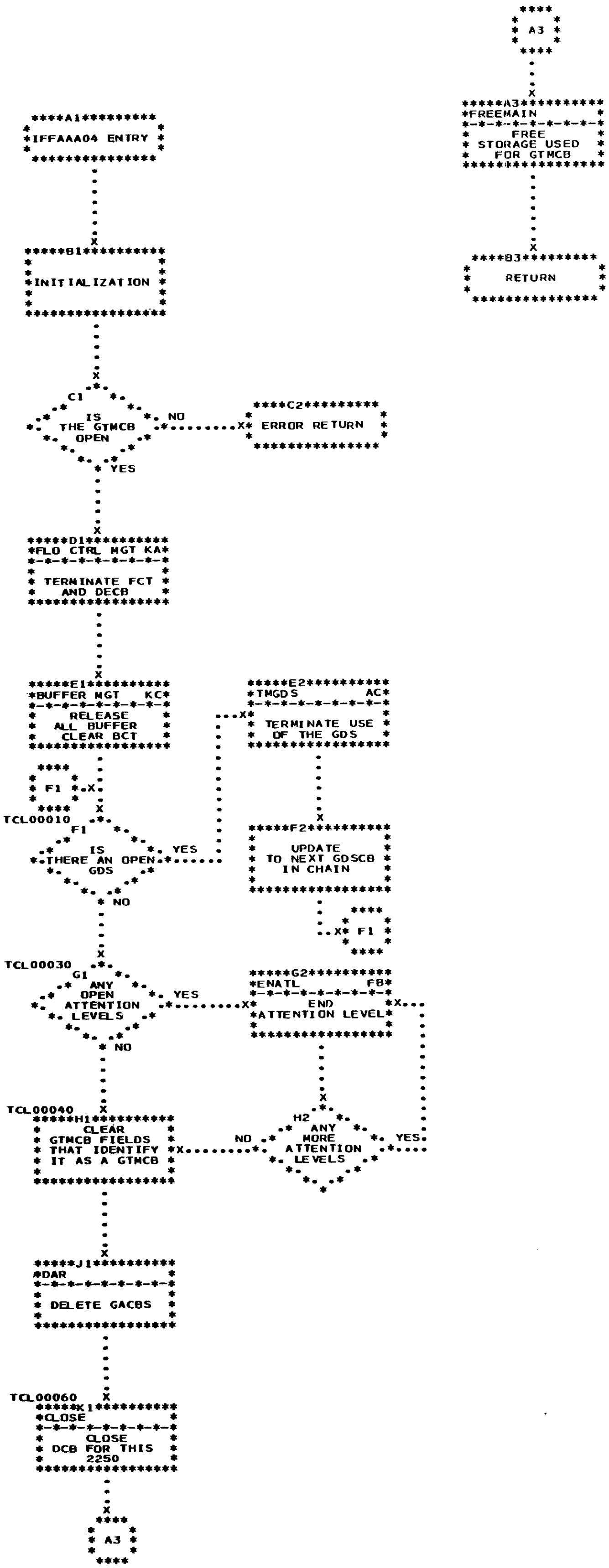


Chart BA. Image Generation Subroutines

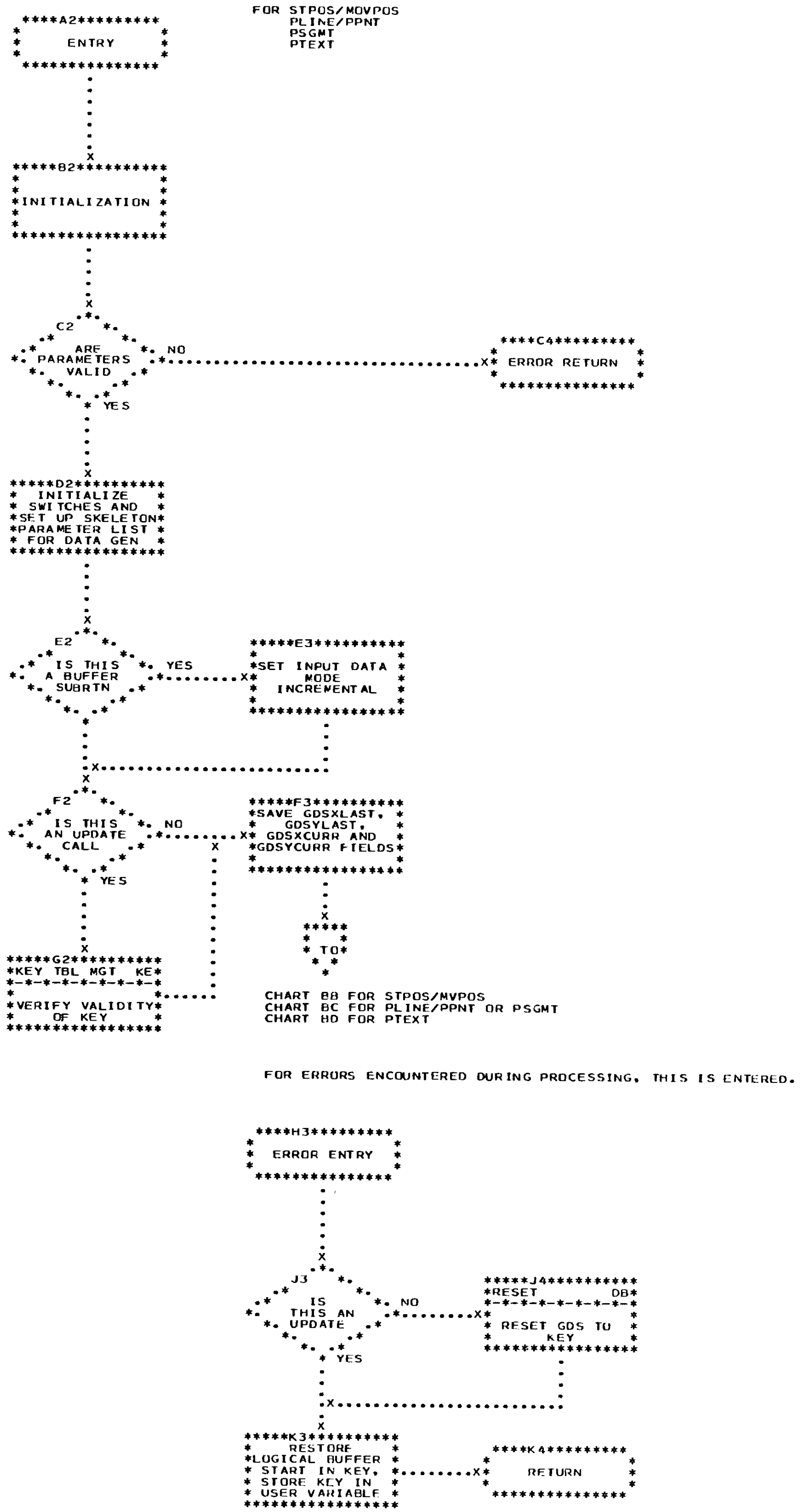


Chart BB. STPOS/MVPOS Subroutines

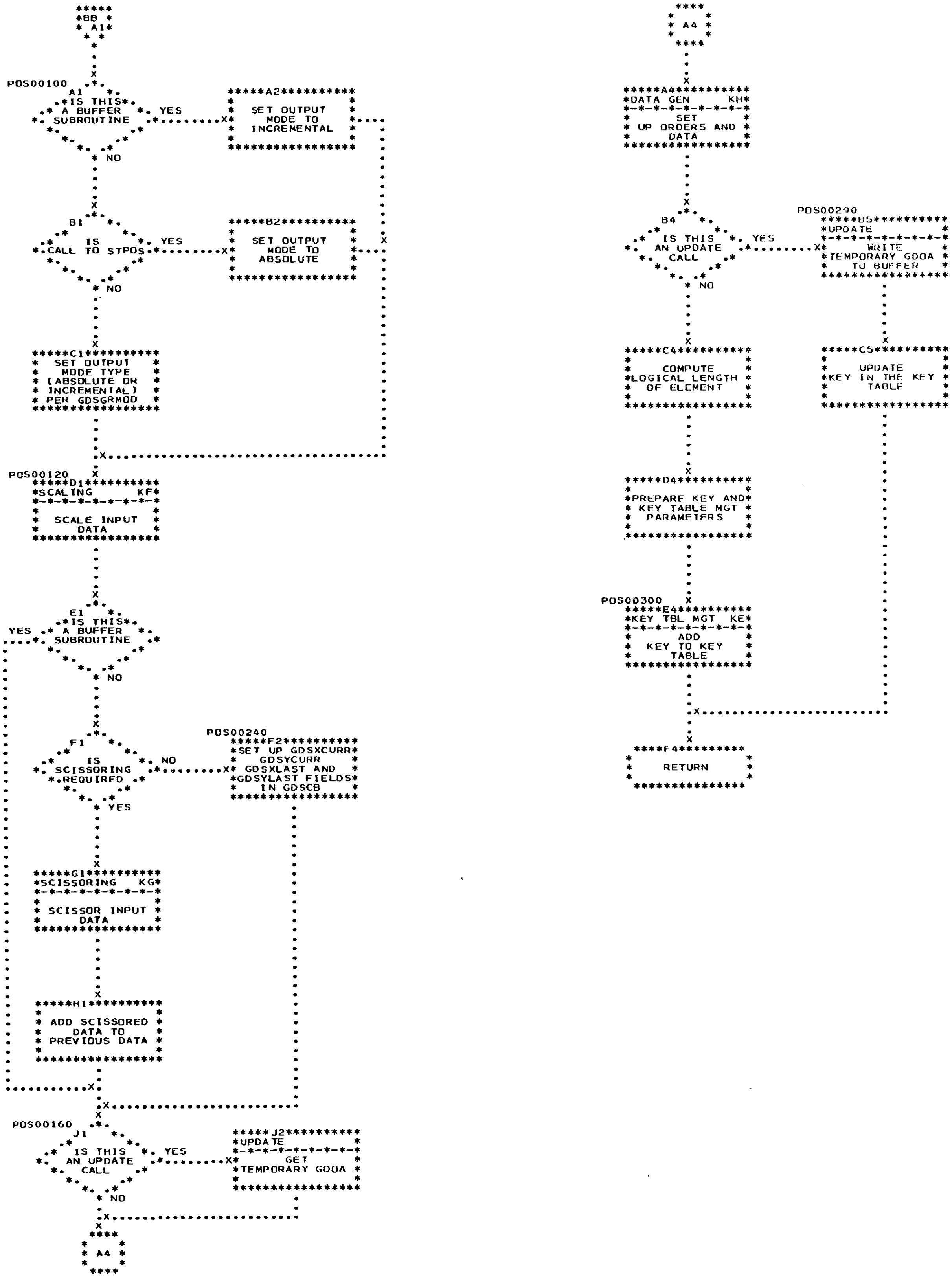

BB A1
POS00100
A1 IS THIS A BUFFER SUBROUTINE
YES
NO
A2 SET OUTPUT MODE TO INCREMENTAL
B1 IS CALL TO STPOS
YES
NO
B2 SET OUTPUT MODE TO ABSOLUTE
C1 SET OUTPUT MODE TYPE (ABSOLUTE OR INCREMENTAL) PER GDSGRMOD
POS00120
D1 SCALING KF
SCALE INPUT DATA
E1 IS THIS A BUFFER SUBROUTINE
YES
NO
F1 IS SCISSORING REQUIRED
NO
YES
POS00240
F2 SET UP GDSXCURR GDSYCURR GDSXLAST AND GDSYLAST FIELDS IN GDSCB
G1 SCISSORING KG
SCISSOR INPUT DATA
H1 ADD SCISSORED DATA TO PREVIOUS DATA
POS00160
J1 IS THIS AN UPDATE CALL
YES
NO
J2 UPDATE
GET TEMPORARY GDOA
A4
A4
A4 DATA GEN KH
SET UP ORDERS AND DATA
B4 IS THIS AN UPDATE CALL
YES
NO
POS00290
B5 UPDATE
WRITE TEMPORARY GDOA TO BUFFER
C4 COMPUTE LOGICAL LENGTH OF ELEMENT
C5 UPDATE KEY IN THE KEY TABLE
D4 PREPARE KEY AND KEY TABLE MGT PARAMETERS
POS00300
E4 KEY TBL MGT KE
ADD KEY TO KEY TABLE
F4 RETURN

Chart BC. PLINE/PPNT and PSGMT Subroutines

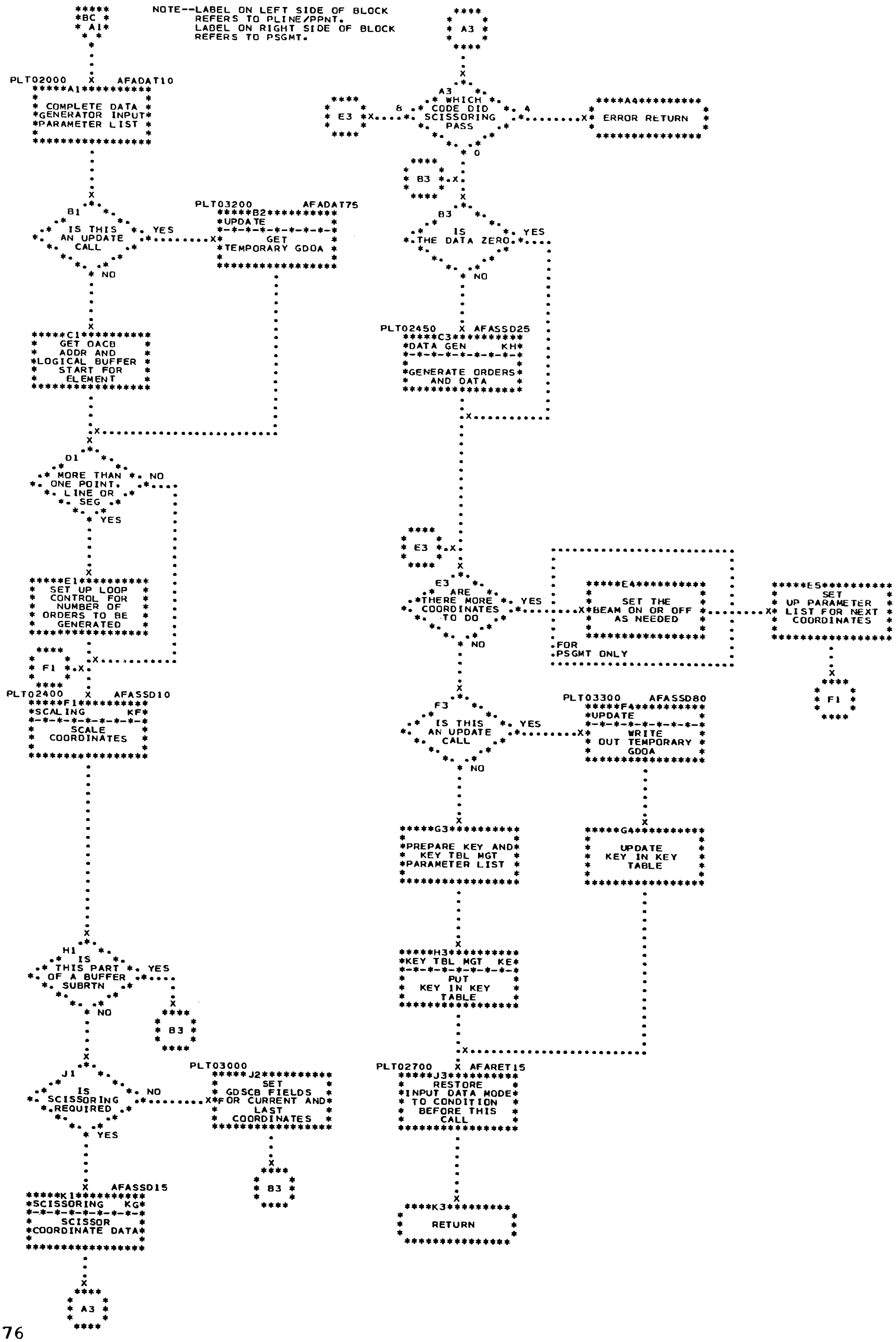

BC A1
NOTE--LABEL ON LEFT SIDE OF BLOCK REFERS TO PLINE/PPNT. LABEL ON RIGHT SIDE OF BLOCK REFERS TO PSGMT.
PLT02000 AFADAT10
A1 COMPLETE DATA GENERATOR INPUT PARAMETER LIST
B1 IS THIS AN UPDATE CALL
YES
NO
PLT03200 AFADAT75
B2 UPDATE
GET TEMPORARY GDOA
C1 GET OACB ADDR AND LOGICAL BUFFER START FOR ELEMENT
D1 MORE THAN ONE POINT, LINE OR SEG
NO
YES
E1 SET UP LOOP CONTROL FOR NUMBER OF ORDERS TO BE GENERATED
F1
PLT02400 AFASSD10
F1 SCALING KF
SCALE COORDINATES
H1 IS THIS PART OF A BUFFER SUBRTN
YES
B3
NO
J1 IS SCISSORING REQUIRED
NO
PLT03000
J2 SET GDSCB FIELDS FOR CURRENT AND LAST COORDINATES
B3
YES
AFASSD15
K1 SCISSORING KG
SCISSOR COORDINATE DATA
A3
A3
A3 WHICH CODE DID SCISSORING PASS
E3
8
4
A4 ERROR RETURN
0
B3
B3 IS THE DATA ZERO
YES
NO
PLT02450 AFASSD25
C3 DATA GEN KH
GENERATE ORDERS AND DATA
E3
E3 ARE THERE MORE COORDINATES TO DO
YES
E4 SET THE BEAM ON OR OFF AS NEEDED
FOR PSGMT ONLY
E5 SET UP PARAMETER LIST FOR NEXT COORDINATES
F1
NO
F3 IS THIS AN UPDATE CALL
YES
PLT03300 AFASSD80
F4 UPDATE
WRITE OUT TEMPORARY GDOA
NO
G3 PREPARE KEY AND KEY TBL MGT PARAMETER LIST
G4 UPDATE KEY IN KEY TABLE
H3 KEY TBL MGT KE
PUT KEY IN KEY TABLE
PLT02700 AFARET15
J3 RESTORE INPUT DATA MODE TO CONDITION BEFORE THIS CALL
K3 RETURN

Chart BD. PTEXT Subroutine

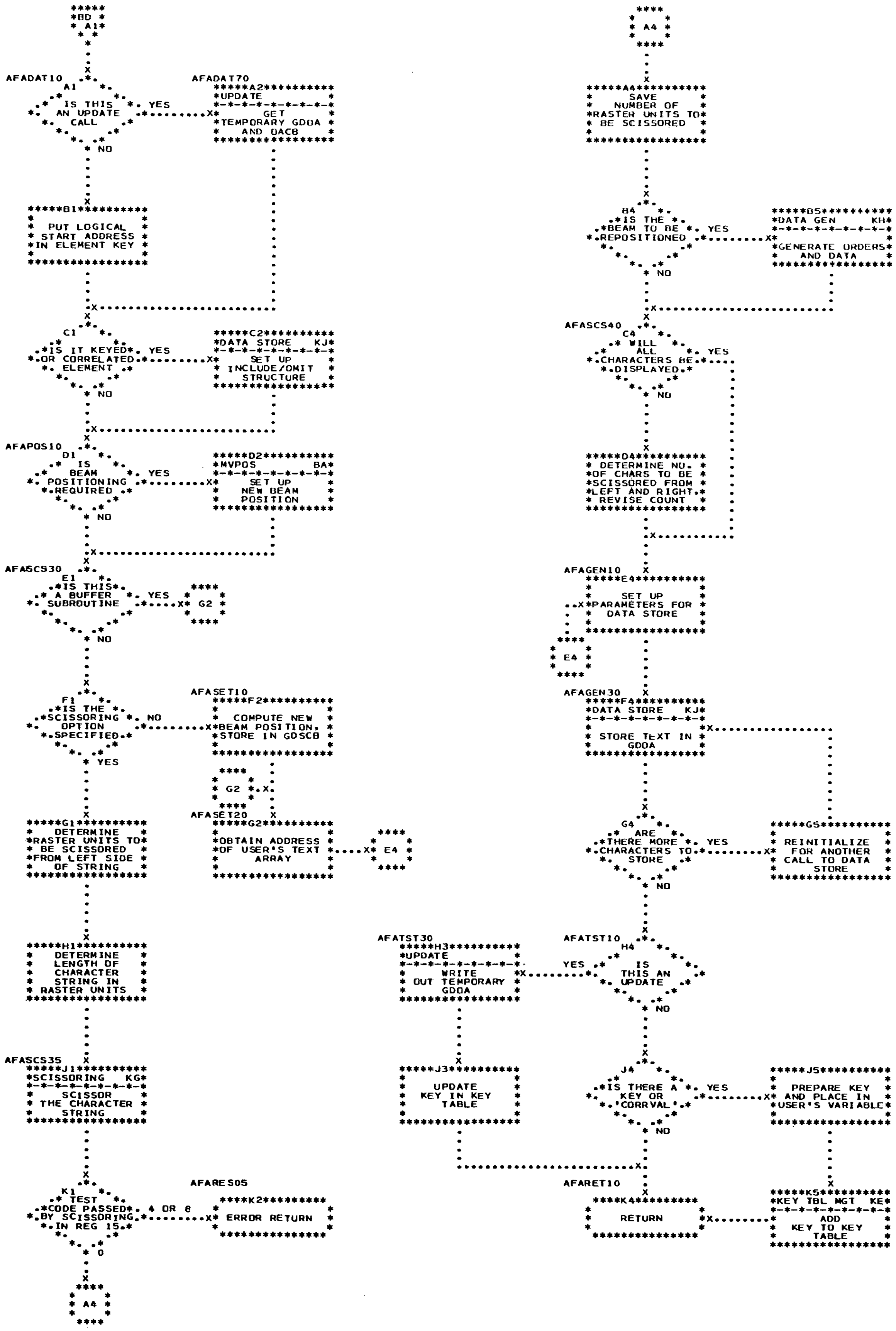
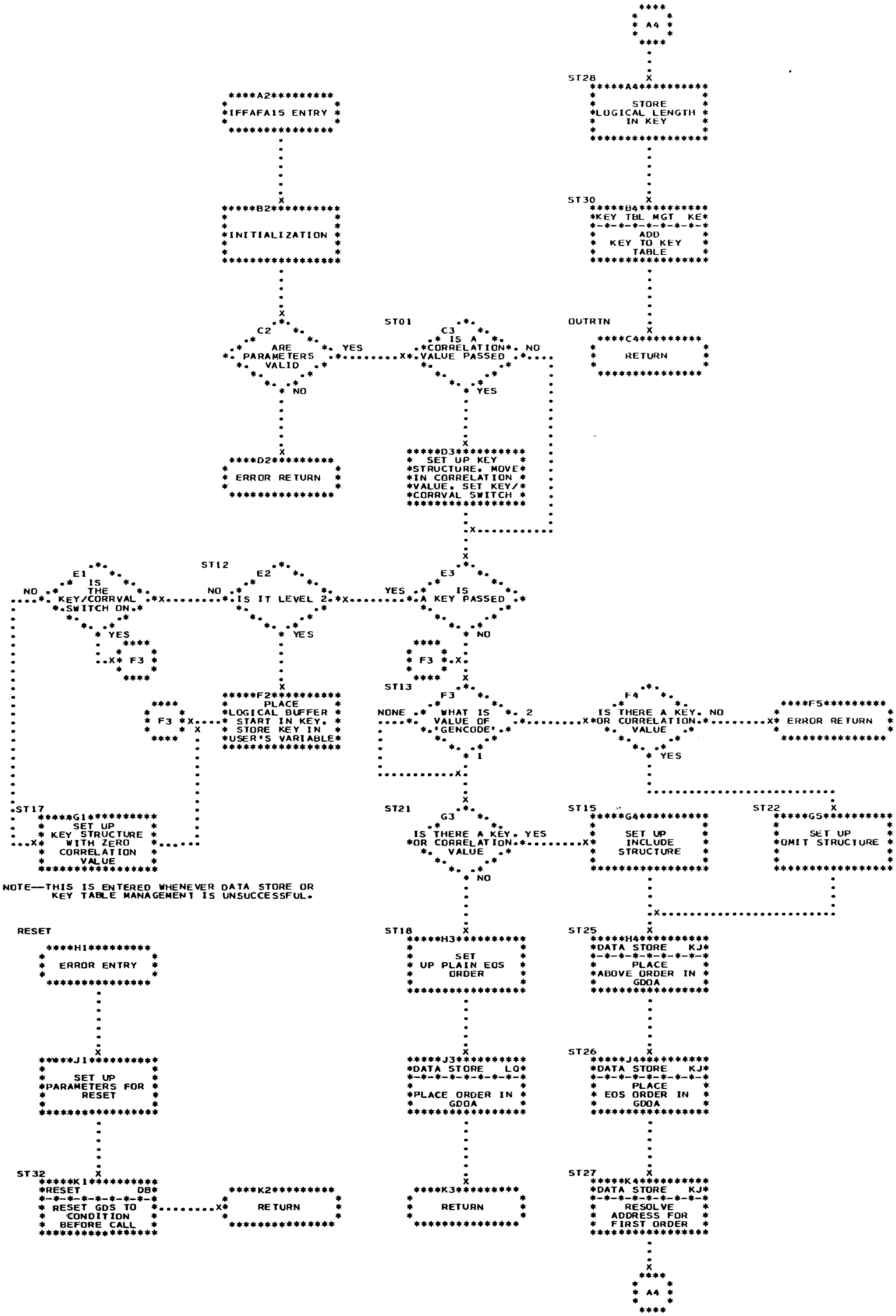

BD A1
AFADAT10
A1
IS THIS AN UPDATE CALL
YES
NO
AFADAT70
A2
UPDATE
GET TEMPORARY GDOA AND DACB
B1
PUT LOGICAL START ADDRESS IN ELEMENT KEY
C1
IS IT KEYED OR CORRELATED ELEMENT
YES
NO
C2
DATA STORE KJ
SET UP INCLUDE/OMIT STRUCTURE
AFAPOS10
D1
IS BEAM POSITIONING REQUIRED
YES
NO
D2
MVPOS BA
SET UP NEW BEAM POSITION
AFASCS30
E1
IS THIS A BUFFER SUBROUTINE
YES
G2
NO
F1
IS THE SCISSORING OPTION SPECIFIED
NO
YES
AFASET10
F2
COMPUTE NEW BEAM POSITION, STORE IN GDSCB
G2
AFASET20
G2
OBTAIN ADDRESS OF USER'S TEXT ARRAY
E4
G1
DETERMINE RASTER UNITS TO BE SCISSORED FROM LEFT SIDE OF STRING
H1
DETERMINE LENGTH OF CHARACTER STRING IN RASTER UNITS
AFASCS35
J1
SCISSORING KG
SCISSOR THE CHARACTER STRING
K1
TEST CODE PASSED BY SCISSORING IN REG 15
4 OR 8
0
AFARES05
K2
ERROR RETURN
A4
A4
A4
SAVE NUMBER OF RASTER UNITS TO BE SCISSORED
B4
IS THE BEAM TO BE REPOSITIONED
YES
NO
B5
DATA GEN KH
GENERATE ORDERS AND DATA
AFASCS40
C4
WILL ALL CHARACTERS BE DISPLAYED
YES
NO
D4
DETERMINE NO. OF CHARS TO BE SCISSORED FROM LEFT AND RIGHT, REVISE COUNT
AFAGEN10
E4
SET UP PARAMETERS FOR DATA STORE
E4
AFAGEN30
F4
DATA STORE KJ
STORE TEXT IN GDOA
G4
ARE THERE MORE CHARACTERS TO STORE
YES
NO
G5
REINITIALIZE FOR ANOTHER CALL TO DATA STORE
AFATST30
H3
UPDATE
WRITE OUT TEMPORARY GDOA
AFATST10
H4
IS THIS AN UPDATE
YES
NO
J3
UPDATE KEY IN KEY TABLE
J4
IS THERE A KEY OR 'CORRVAL'
YES
NO
J5
PREPARE KEY AND PLACE IN USER'S VARIABLE
AFARET10
K4
RETURN
K5
KEY TBL MGT KE
ADD KEY TO KEY TABLE

Chart BE. STEOS Subroutine



NOTE—THIS IS ENTERED WHENEVER DATA STORE OR KEY TABLE MANAGEMENT IS UNSUCCESSFUL.

Chart CA. BGSEQ/BGSUB Subroutines

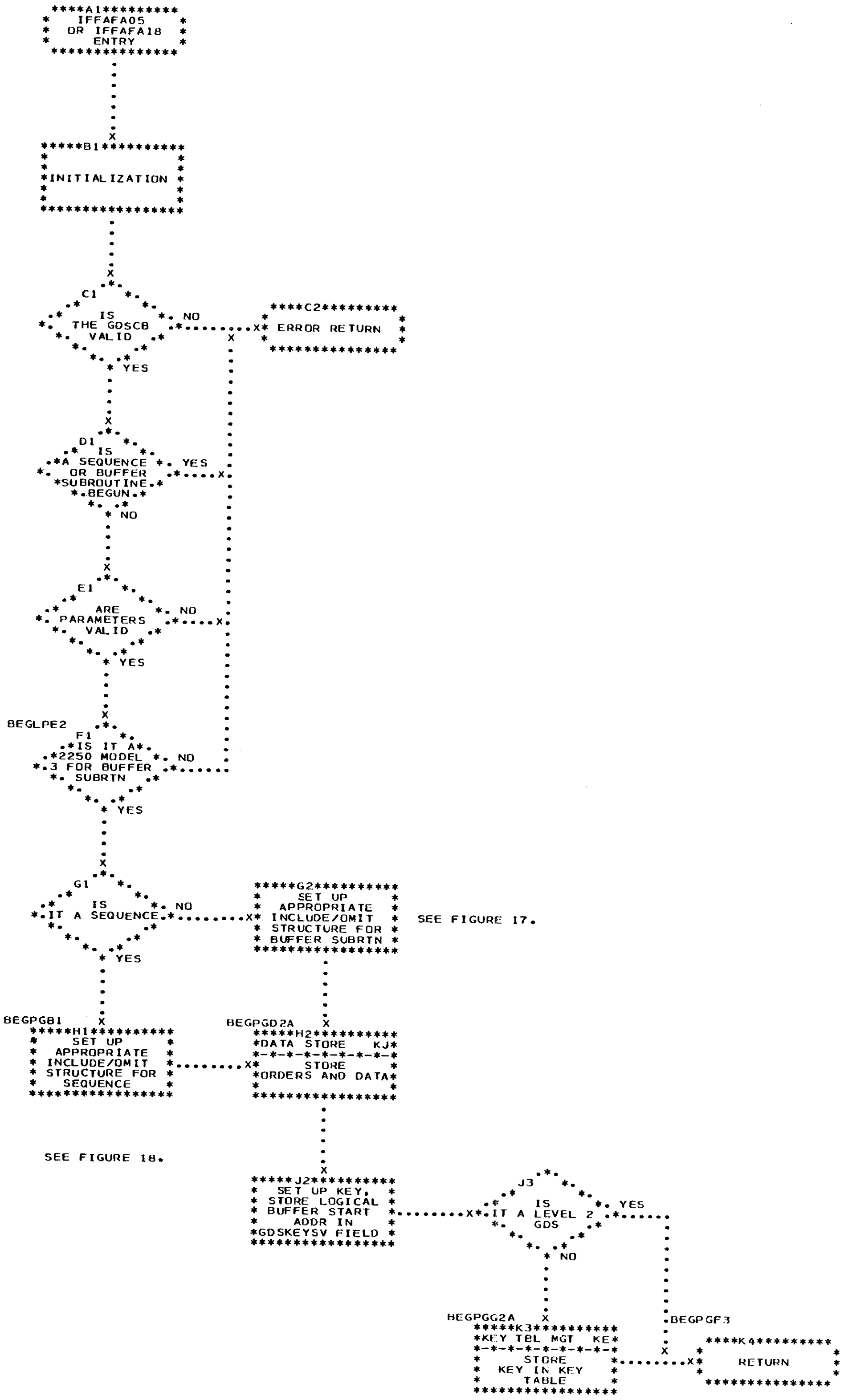


Chart CB. LKSUB Subroutine

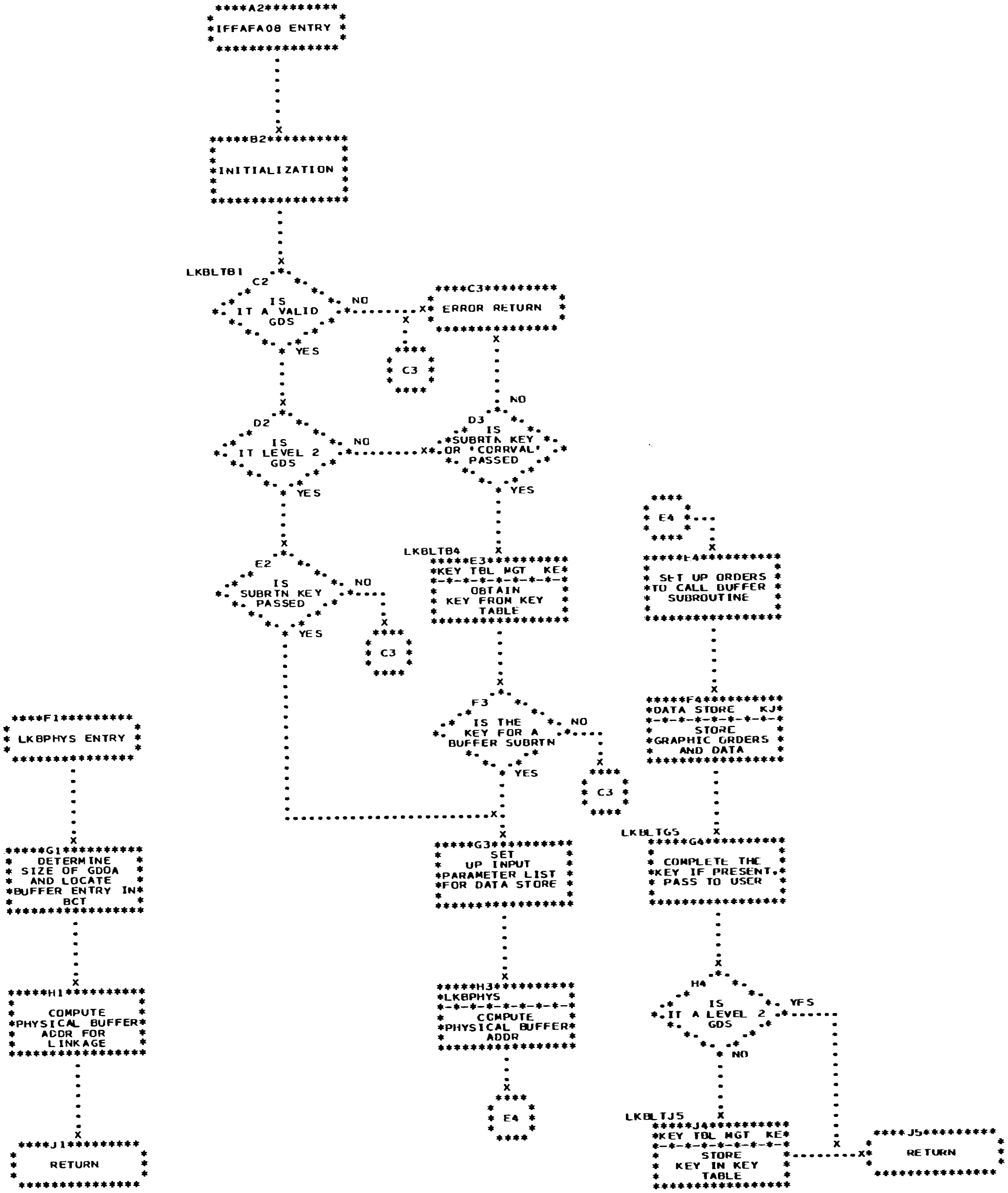

A2
IFFAFA08 ENTRY
B2
INITIALIZATION
LKBLTB1
C2
IS IT A VALID GDS
NO
YES
C3
ERROR RETURN
C3
D2
IS IT LEVEL 2 GDS
NO
YES
D3
IS SUBRTN KEY OR 'CORRVAL' PASSED
NO
YES
E2
IS SUBRTN KEY PASSED
NO
YES
C3
LKBLTB4
E3
KEY TBL MGT KE
OBTAIN KEY FROM KEY TABLE
E4
E4
SET UP ORDERS TO CALL BUFFER SUBROUTINE
F1
LKBPHYS ENTRY
F3
IS THE KEY FOR A BUFFER SUBRTN
NO
YES
C3
F4
DATA STORE KJ
STORE GRAPHIC ORDERS AND DATA
G1
DETERMINE SIZE OF GDOA AND LOCATE BUFFER ENTRY IN BCT
G3
SET UP INPUT PARAMETER LIST FOR DATA STORE
LKBLTG5
G4
COMPLETE THE KEY IF PRESENT. PASS TO USER
H1
COMPUTE PHYSICAL BUFFER ADDR FOR LINKAGE
H3
LKBPHYS
COMPUTE PHYSICAL BUFFER ADDR
E4
H4
IS IT A LEVEL 2 GDS
YES
NO
J1
RETURN
LKBLTJ5
J4
KEY TBL MGT KE
STORE KEY IN KEY TABLE
J5
RETURN

Chart DA. EXEC Subroutine

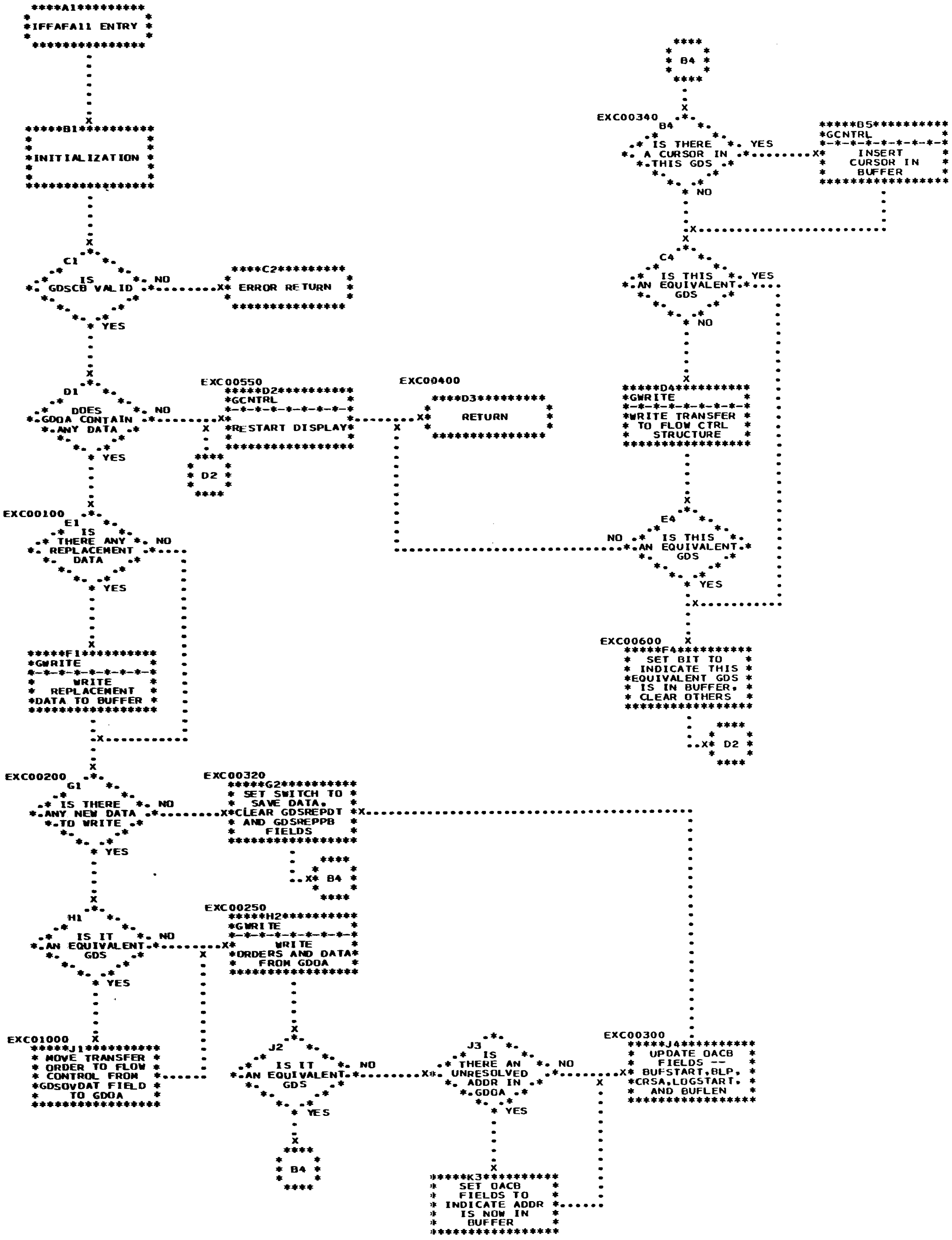

A1
IFFAFA11 ENTRY
B1
INITIALIZATION
C1
IS GDSCB VALID
NO
YES
C2
ERROR RETURN
D1
DOES GDOA CONTAIN ANY DATA
NO
YES
EXC00550
D2
GCNTRL
RESTART DISPLAY
D2
EXC00400
D3
RETURN
EXC00100
E1
IS THERE ANY REPLACEMENT DATA
NO
YES
F1
GWRITE
WRITE REPLACEMENT DATA TO BUFFER
EXC00200
G1
IS THERE ANY NEW DATA TO WRITE
NO
YES
EXC00320
G2
SET SWITCH TO SAVE DATA. CLEAR GDSREPDT AND GDSREPPB FIELDS
B4
H1
IS IT AN EQUIVALENT GDS
NO
YES
EXC00250
H2
GWRITE
WRITE ORDERS AND DATA FROM GDOA
EXC01000
J1
MOVE TRANSFER ORDER TO FLOW CONTROL FROM GDSOVDAT FIELD TO GDOA
J2
IS IT AN EQUIVALENT GDS
NO
YES
B4
J3
IS THERE AN UNRESOLVED ADDR IN GDOA
NO
YES
K3
SET OACB FIELDS TO INDICATE ADDR IS NOW IN BUFFER
EXC00300
J4
UPDATE OACB FIELDS -- BUFSTART,BLP, CRSA,LOGSTART, AND BUFLEN
B4
EXC00340
B4
IS THERE A CURSOR IN THIS GDS
YES
NO
B5
GCNTRL
INSERT CURSOR IN BUFFER
C4
IS THIS AN EQUIVALENT GDS
YES
NO
D4
GWRITE
WRITE TRANSFER TO FLOW CTRL STRUCTURE
E4
NO
IS THIS AN EQUIVALENT GDS
YES
EXC00600
F4
SET BIT TO INDICATE THIS EQUIVALENT GDS IS IN BUFFER. CLEAR OTHERS
D2

Chart DB. RESET Subroutine

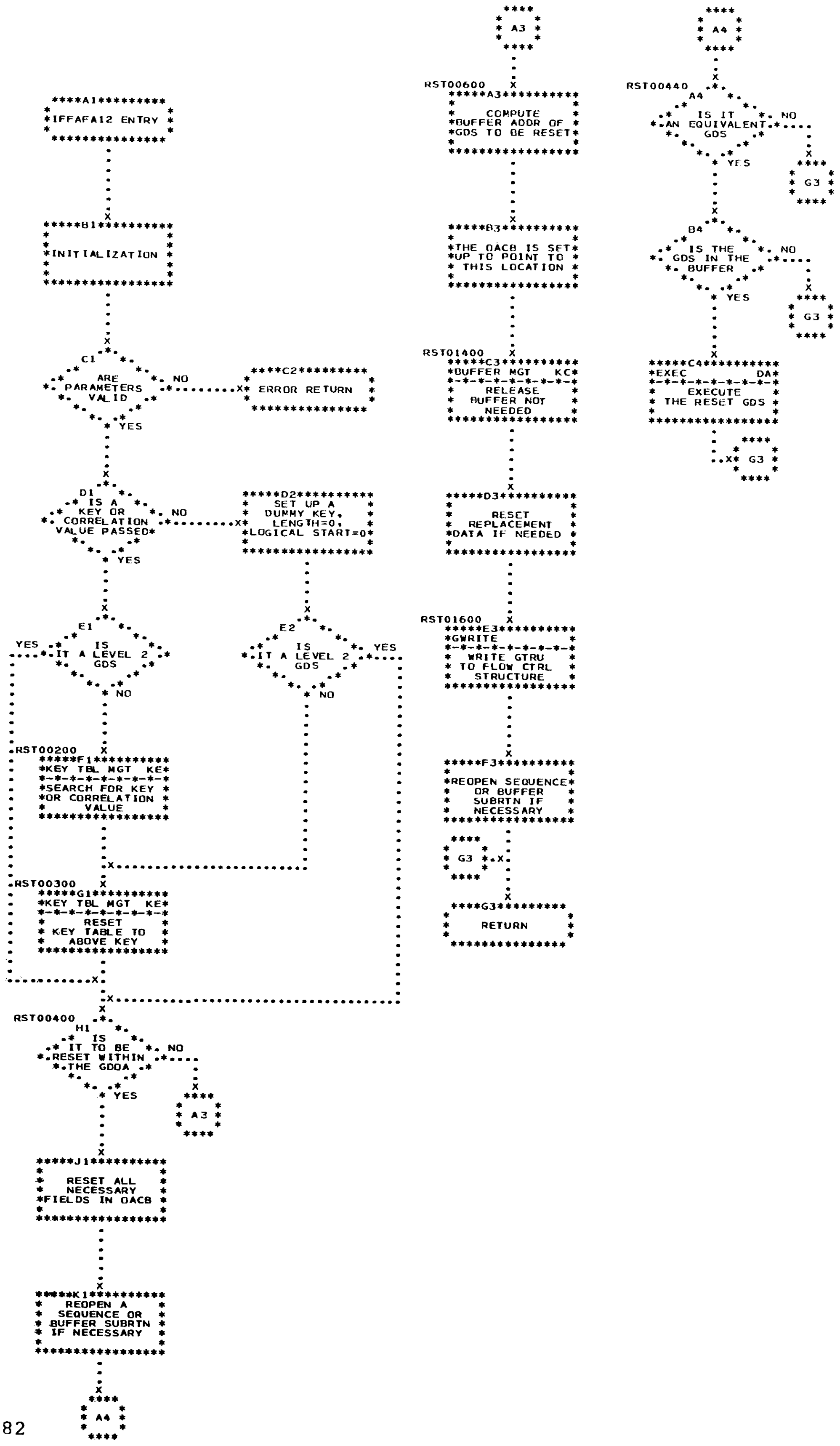

A1
IFFAFA12 ENTRY
B1
INITIALIZATION
C1
ARE PARAMETERS VALID
NO
YES
C2
ERROR RETURN
D1
IS A KEY OR CORRELATION VALUE PASSED
NO
YES
D2
SET UP A DUMMY KEY. LENGTH=0. LOGICAL START=0
E1
IS IT A LEVEL 2 GDS
YES
NO
E2
IS IT A LEVEL 2 GDS
YES
NO
RST00200
F1
KEY TBL MGT KE
SEARCH FOR KEY OR CORRELATION VALUE
RST00300
G1
KEY TBL MGT KE
RESET KEY TABLE TO ABOVE KEY
RST00400
H1
IS IT TO BE RESET WITHIN THE GDOA
NO
YES
A3
J1
RESET ALL NECESSARY FIELDS IN DACB
K1
REOPEN A SEQUENCE OR BUFFER SUBRTN IF NECESSARY
A4
A3
RST00600
A3
COMPUTE BUFFER ADDR OF GDS TO BE RESET
B3
THE DACB IS SET UP TO POINT TO THIS LOCATION
RST01400
C3
BUFFER MGT KC
RELEASE BUFFER NOT NEEDED
D3
RESET REPLACEMENT DATA IF NEEDED
RST01600
E3
GWRITE
WRITE GTRU TO FLOW CTRL STRUCTURE
F3
REOPEN SEQUENCE OR BUFFER SUBRTN IF NECESSARY
G3
G3
RETURN
A4
RST00440
A4
IS IT AN EQUIVALENT GDS
NO
G3
YES
B4
IS THE GDS IN THE BUFFER
NO
G3
YES
C4
EXEC DA
EXECUTE THE RESET GDS
G3

Chart DC. IDPOS Subroutine

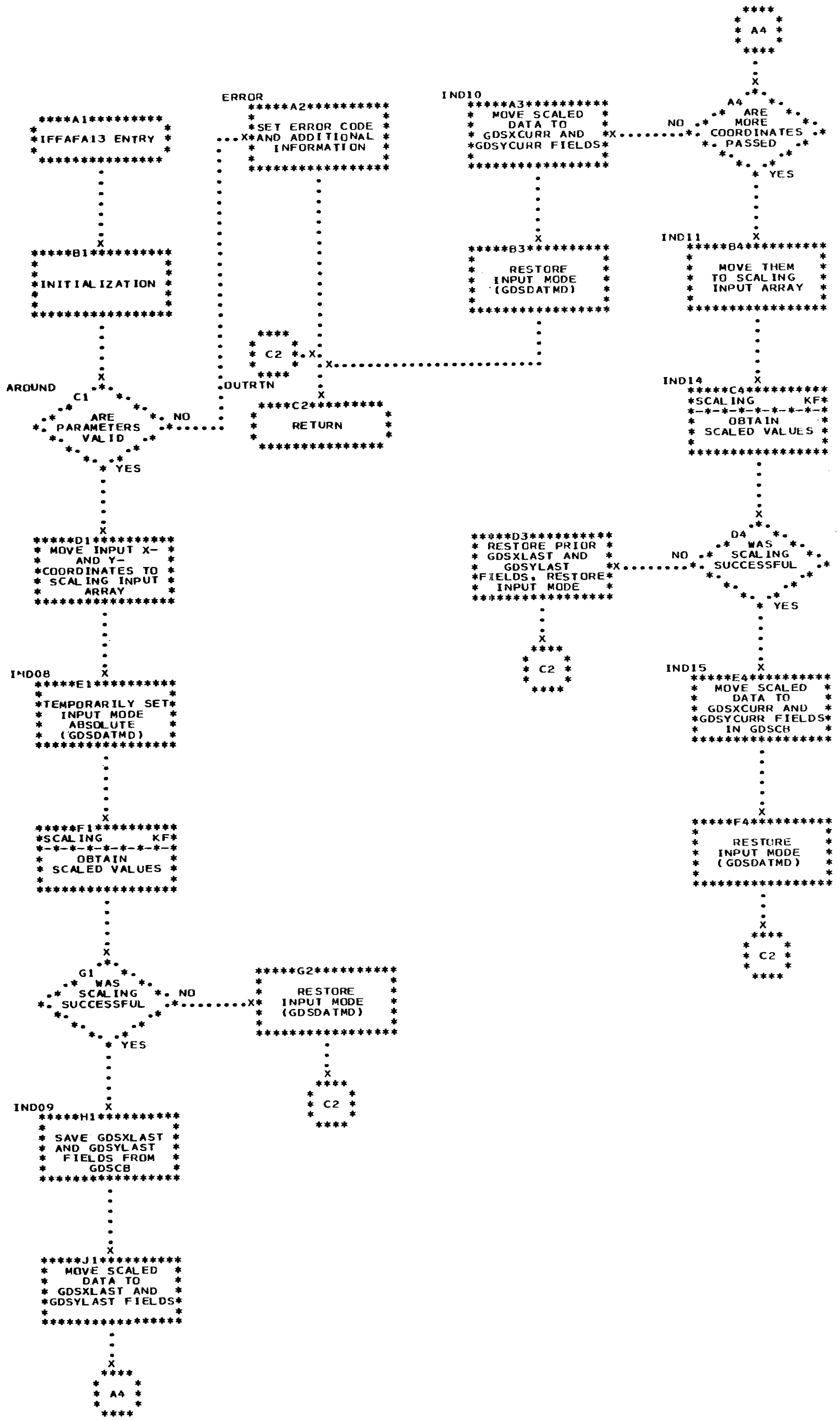

A4
A1
IFFAFA13 ENTRY
ERROR
A2
SET ERROR CODE AND ADDITIONAL INFORMATION
IND10
A3
MOVE SCALED DATA TO GDSXCURR AND GDSYCURR FIELDS
A4
ARE MORE COORDINATES PASSED
NO
YES
B1
INITIALIZATION
B3
RESTORE INPUT MODE (GDSDATMD)
IND11
B4
MOVE THEM TO SCALING INPUT ARRAY
C2
OUTRTN
C2
RETURN
AROUND
C1
ARE PARAMETERS VALID
NO
YES
IND14
C4
SCALING KF
OBTAIN SCALED VALUES
D1
MOVE INPUT X- AND Y- COORDINATES TO SCALING INPUT ARRAY
D3
RESTORE PRIOR GDSXLAST AND GDSYLAST FIELDS. RESTORE INPUT MODE
C2
D4
WAS SCALING SUCCESSFUL
NO
YES
IND08
E1
TEMPORARILY SET INPUT MODE ABSOLUTE (GDSDATMD)
IND15
E4
MOVE SCALED DATA TO GDSXCURR AND GDSYCURR FIELDS IN GDSCB
F1
SCALING KF
OBTAIN SCALED VALUES
F4
RESTORE INPUT MODE (GDSDATMD)
C2
G1
WAS SCALING SUCCESSFUL
NO
YES
G2
RESTORE INPUT MODE (GDSDATMD)
C2
IND09
H1
SAVE GDSXLAST AND GDSYLAST FIELDS FROM GDSCB
J1
MOVE SCALED DATA TO GDSXLAST AND GDSYLAST FIELDS
A4

Chart EA. ICURS Subroutine

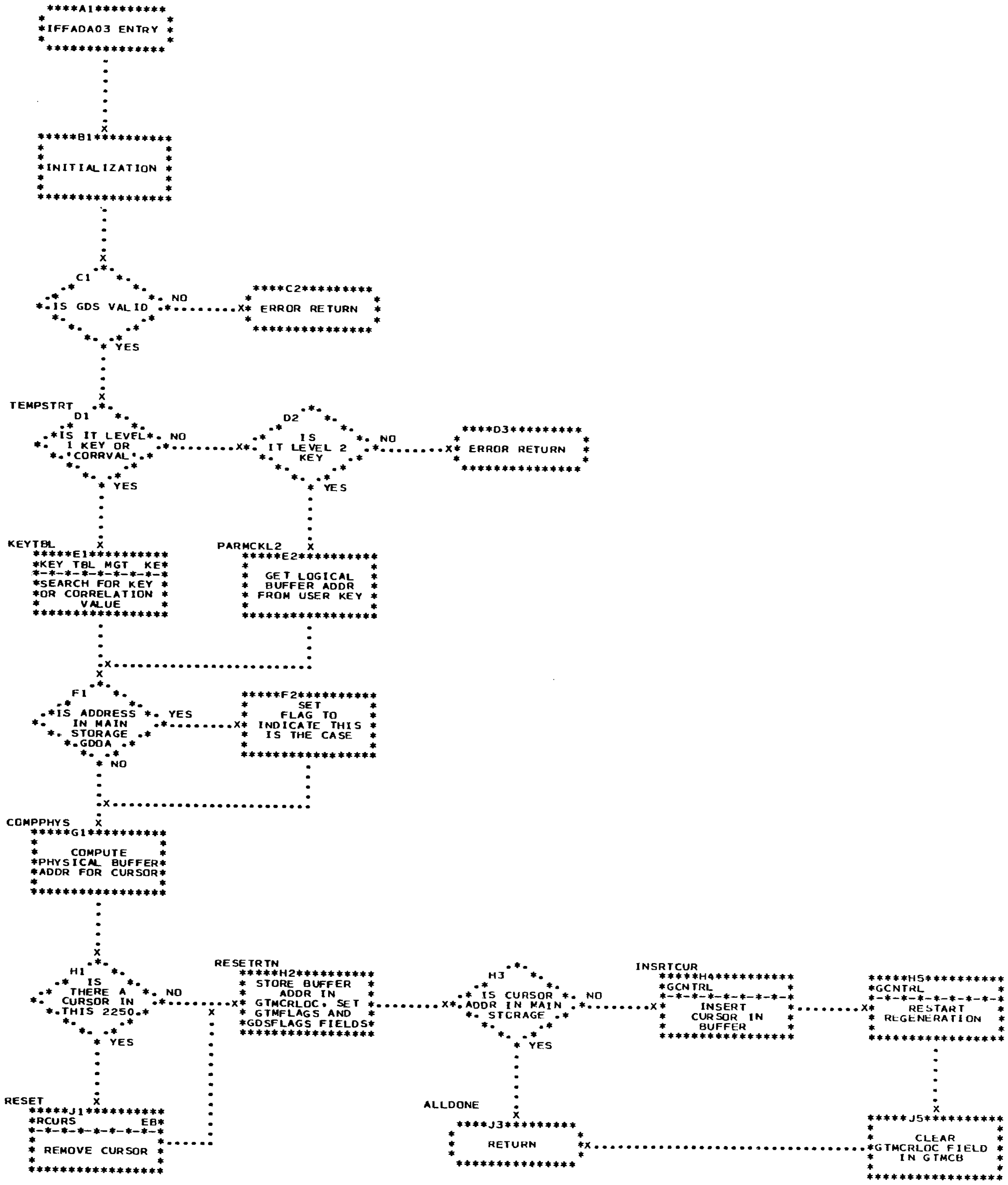

A1
IFFADA03 ENTRY
B1
INITIALIZATION
C1
IS GDS VALID
NO
C2
ERROR RETURN
YES
TEMPSTRT
D1
IS IT LEVEL 1 KEY OR CORRVAL
NO
D2
IS IT LEVEL 2 KEY
NO
D3
ERROR RETURN
YES
YES
KEYTBL
E1
KEY TBL MGT KE
SEARCH FOR KEY OR CORRELATION VALUE
PARMCKL2
E2
GET LOGICAL BUFFER ADDR FROM USER KEY
F1
IS ADDRESS IN MAIN STORAGE GDDA
YES
F2
SET FLAG TO INDICATE THIS IS THE CASE
NO
COMPPHYS
G1
COMPUTE PHYSICAL BUFFER ADDR FOR CURSOR
H1
IS THERE A CURSOR IN THIS 2250
NO
RESETRTN
H2
STORE BUFFER ADDR IN GTMCRLOC. SET GTMFLAGS AND GDSFLAGS FIELDS
H3
IS CURSOR ADDR IN MAIN STORAGE
NO
INSRTCUR
H4
GCNTRL
INSERT CURSOR IN BUFFER
H5
GCNTRL
RESTART REGENERATION
YES
YES
RESET
J1
RCURS EB
REMOVE CURSOR
ALLDONE
J3
RETURN
J5
CLEAR GTMCRLOC FIELD IN GTMCB

Chart EB. RCURS Subroutine

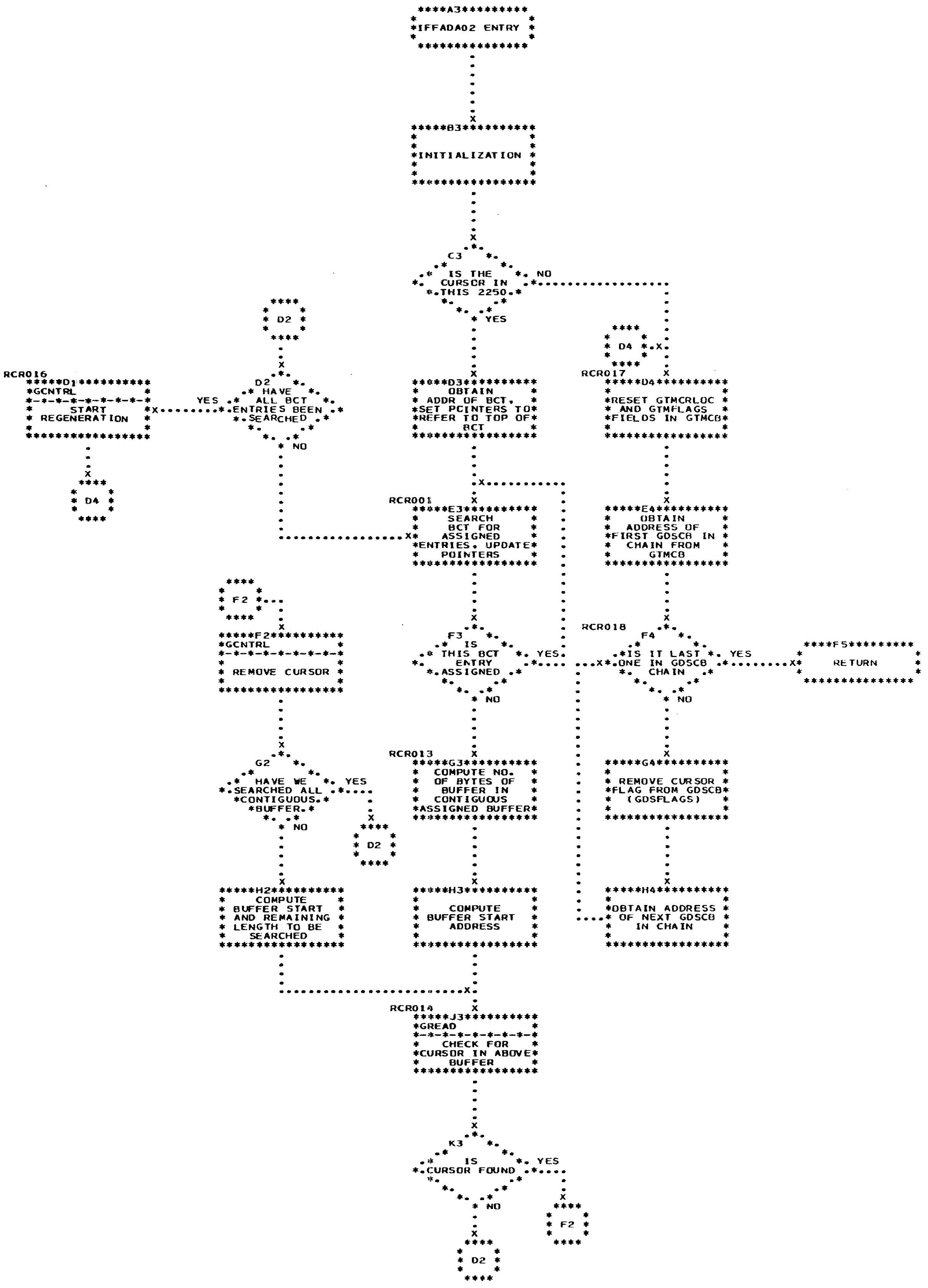

A3
IFFADA02 ENTRY
B3
INITIALIZATION
C3
IS THE CURSOR IN THIS 2250
NO
YES
D2
D4
RCR016
D1
GCNTRL
START REGENERATION
D2
HAVE ALL BCT ENTRIES BEEN SEARCHED
YES
NO
D3
OBTAIN ADDR OF BCT, SET PCINTERS TO REFER TO TOP OF BCT
RCR017
D4
RESET GTMCRLOC AND GTMFLAGS FIELDS IN GTMCB
D4
RCR001
E3
SEARCH BCT FOR ASSIGNED ENTRIES, UPDATE POINTERS
E4
OBTAIN ADDRESS OF FIRST GDSCB IN CHAIN FROM GTMCB
F2
F2
GCNTRL
REMOVE CURSOR
F3
IS THIS BCT ENTRY ASSIGNED
YES
NO
RCR018
F4
IS IT LAST ONE IN GDSCB CHAIN
YES
NO
F5
RETURN
G2
HAVE WE SEARCHED ALL CONTIGUOUS BUFFER
YES
NO
D2
RCR013
G3
COMPUTE NO. OF BYTES OF BUFFER IN CONTIGUOUS ASSIGNED BUFFER
G4
REMOVE CURSOR FLAG FROM GDSCB (GDSFLAGS)
H2
COMPUTE BUFFER START AND REMAINING LENGTH TO BE SEARCHED
H3
COMPUTE BUFFER START ADDRESS
H4
OBTAIN ADDRESS OF NEXT GDSCB IN CHAIN
RCR014
J3
GREAD
CHECK FOR CURSOR IN ABOVE BUFFER
K3
IS CURSOR FOUND
YES
NO
F2
D2

Chart EC. GSPRD Subroutine

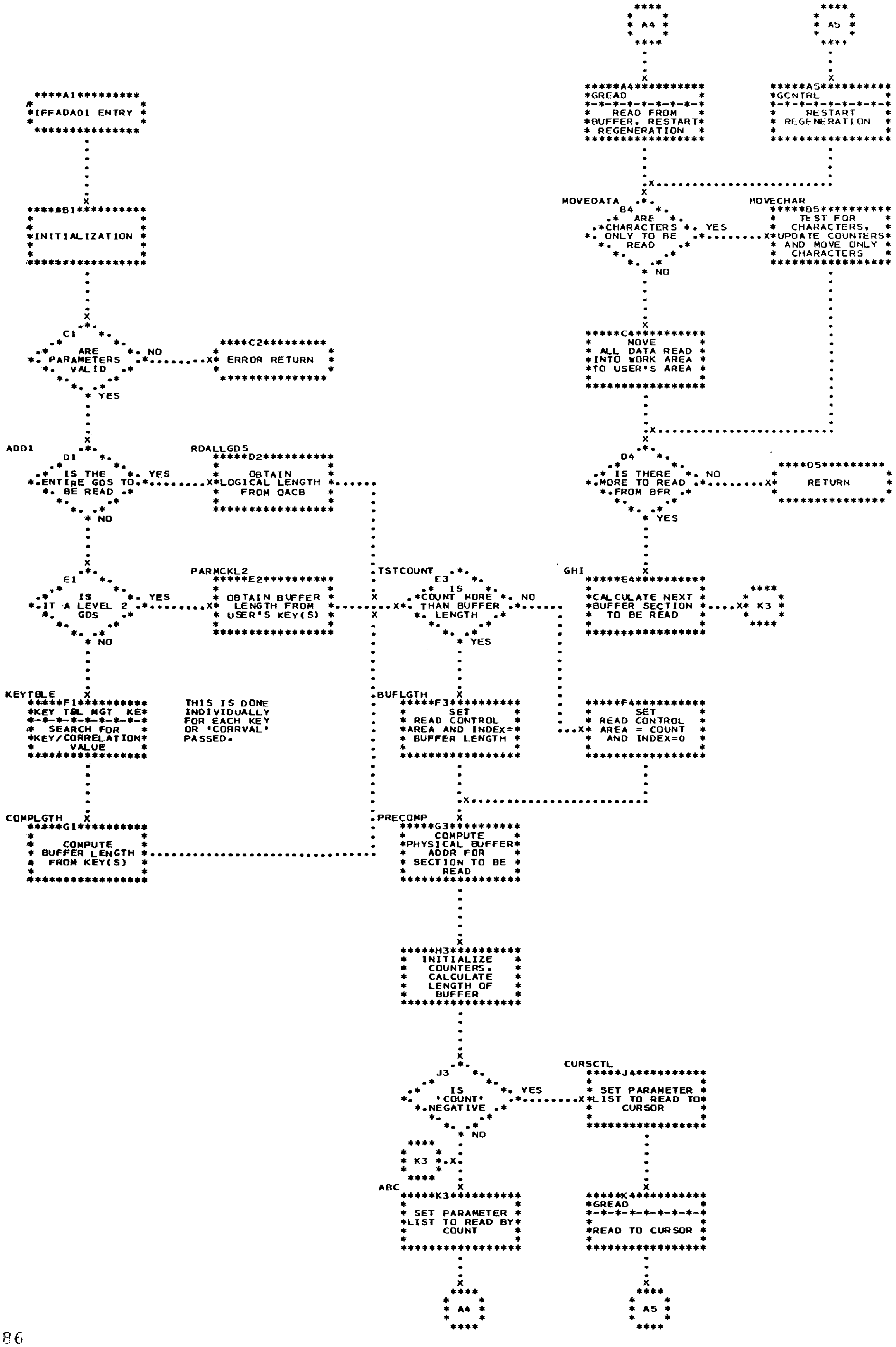

A4
A5
A1
IFFADA01 ENTRY
A4
GREAD
READ FROM BUFFER, RESTART REGENERATION
A5
GCNTRL
RESTART REGENERATION
B1
INITIALIZATION
MOVEDATA
B4
ARE CHARACTERS ONLY TO BE READ
YES
NO
MOVECHAR
B5
TEST FOR CHARACTERS, UPDATE COUNTERS AND MOVE ONLY CHARACTERS
C1
ARE PARAMETERS VALID
NO
YES
C2
ERROR RETURN
C4
MOVE ALL DATA READ INTO WORK AREA TO USER'S AREA
ADD1
D1
IS THE ENTIRE GDS TO BE READ
YES
NO
RDALLGDS
D2
OBTAIN LOGICAL LENGTH FROM DACB
D4
IS THERE MORE TO READ FROM BFR
NO
YES
D5
RETURN
E1
IS IT A LEVEL 2 GDS
YES
NO
PARMCKL2
E2
OBTAIN BUFFER LENGTH FROM USER'S KEY(S)
TSTCOUNT
E3
IS COUNT MORE THAN BUFFER LENGTH
NO
YES
GHI
E4
CALCULATE NEXT BUFFER SECTION TO BE READ
K3
KEYTBLE
F1
KEY TBL MGT KE
SEARCH FOR KEY/CORRELATION VALUE
THIS IS DONE INDIVIDUALLY FOR EACH KEY OR 'CORRVAL' PASSED.
BUFLGTH
F3
SET READ CONTROL AREA AND INDEX= BUFFER LENGTH
F4
SET READ CONTROL AREA = COUNT AND INDEX=0
COMPLGTH
G1
COMPUTE BUFFER LENGTH FROM KEY(S)
PRECOMP
G3
COMPUTE PHYSICAL BUFFER ADDR FOR SECTION TO BE READ
H3
INITIALIZE COUNTERS. CALCULATE LENGTH OF BUFFER
J3
IS 'COUNT' NEGATIVE
YES
NO
CURSCTL
J4
SET PARAMETER LIST TO READ TO CURSOR
K3
ABC
K3
SET PARAMETER LIST TO READ BY COUNT
K4
GREAD
READ TO CURSOR
A4
A5

Chart FA. CRATL Subroutine

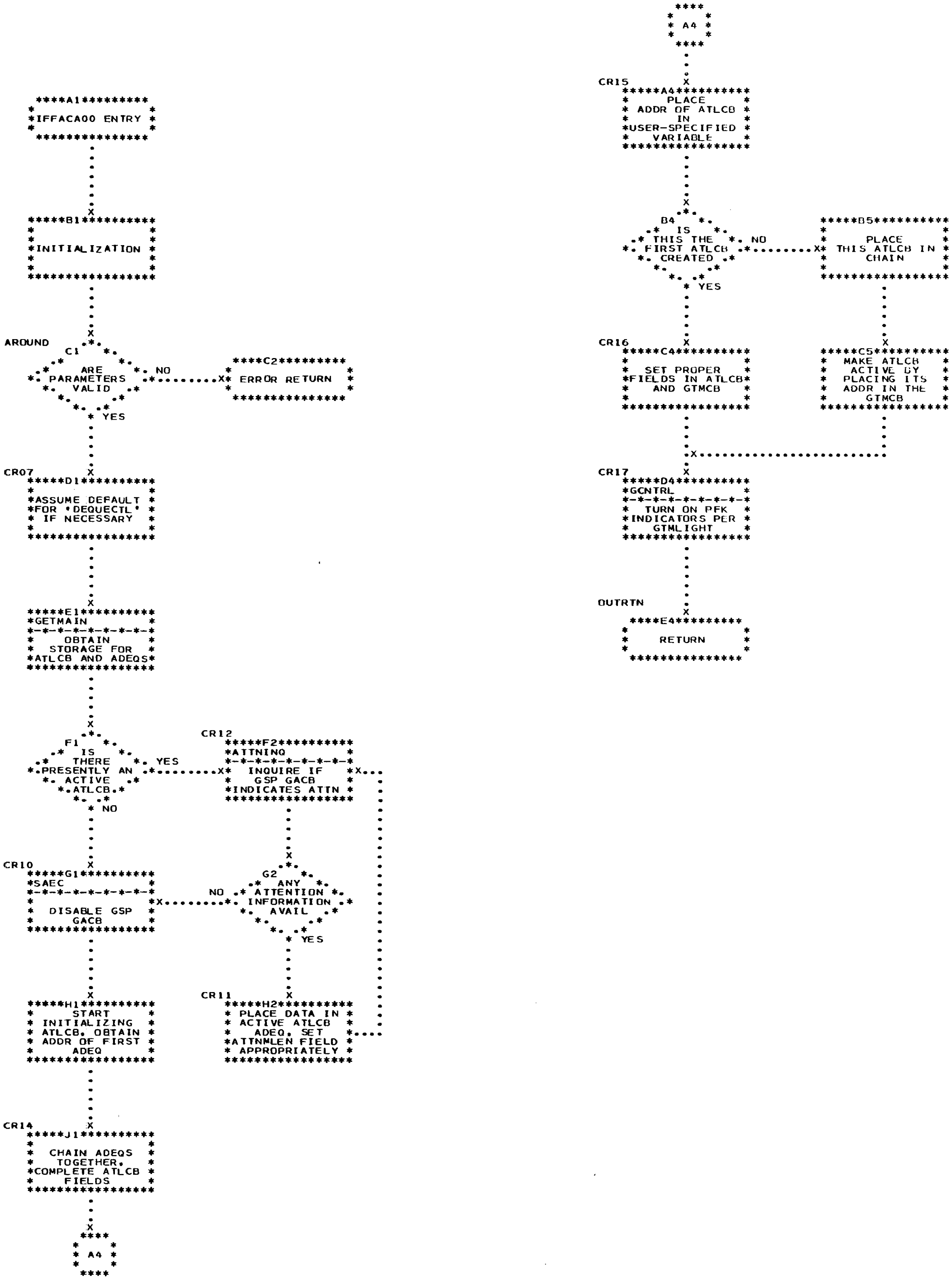

A1
IFFACA00 ENTRY
B1
INITIALIZATION
AROUND
C1
ARE PARAMETERS VALID
NO
C2
ERROR RETURN
YES
CR07
D1
ASSUME DEFAULT FOR 'DEQUECTL' IF NECESSARY
E1
GETMAIN
OBTAIN STORAGE FOR ATLCB AND ADEQS
F1
IS THERE PRESENTLY AN ACTIVE ATLCB
YES
NO
CR12
F2
ATTNINQ
INQUIRE IF GSP GACB INDICATES ATTN
G2
ANY ATTENTION INFORMATION AVAIL
NO
YES
CR10
G1
SAEC
DISABLE GSP GACB
H1
START INITIALIZING ATLCB. OBTAIN ADDR OF FIRST ADEQ
CR11
H2
PLACE DATA IN ACTIVE ATLCB ADEQ. SET ATTNMLEN FIELD APPROPRIATELY
CR14
J1
CHAIN ADEQS TOGETHER. COMPLETE ATLCB FIELDS
A4
A4
CR15
A4
PLACE ADDR OF ATLCB IN USER-SPECIFIED VARIABLE
B4
IS THIS THE FIRST ATLCB CREATED
NO
B5
PLACE THIS ATLCB IN CHAIN
YES
CR16
C4
SET PROPER FIELDS IN ATLCB AND GTMCB
C5
MAKE ATLCB ACTIVE BY PLACING ITS ADDR IN THE GTMCB
CR17
D4
GCNTRL
TURN ON PFK INDICATORS PER GTMLIGHT
OUTRTN
E4
RETURN

Chart FB. ENATL Subroutine

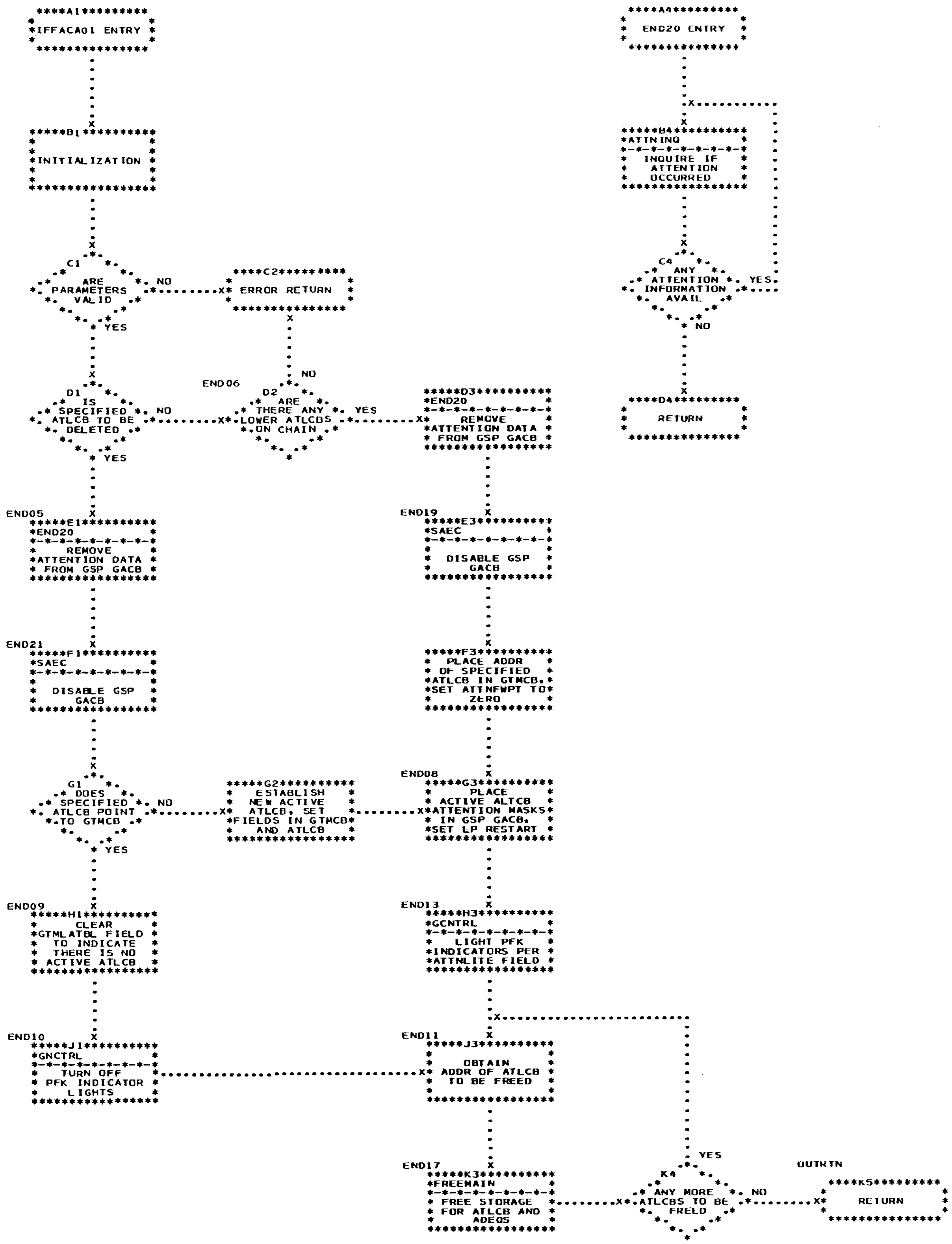

A1
IFFACA01 ENTRY
B1
INITIALIZATION
C1
ARE PARAMETERS VALID
NO
YES
C2
ERROR RETURN
D1
IS SPECIFIED ATLCB TO BE DELETED
NO
YES
END06
D2
ARE THERE ANY LOWER ATLCBS ON CHAIN
NO
YES
D3
END20
REMOVE ATTENTION DATA FROM GSP GACB
END05
E1
END20
REMOVE ATTENTION DATA FROM GSP GACB
END19
E3
SAEC
DISABLE GSP GACB
END21
F1
SAEC
DISABLE GSP GACB
F3
PLACE ADDR OF SPECIFIED ATLCB IN GTMCB. SET ATTNFWPT TO ZERO
G1
DOES SPECIFIED ATLCB POINT TO GTMCB
NO
YES
G2
ESTABLISH NEW ACTIVE ATLCB, SET FIELDS IN GTMCB AND ATLCB
END08
G3
PLACE ACTIVE ALTCB ATTENTION MASKS IN GSP GACB. SET LP RESTART
END09
H1
CLEAR GTMLATBL FIELD TO INDICATE THERE IS NO ACTIVE ATLCB
END13
H3
GCNTRL
LIGHT PFK INDICATORS PER ATTNLITE FIELD
END10
J1
GNCTRL
TURN OFF PFK INDICATOR LIGHTS
END11
J3
OBTAIN ADDR OF ATLCB TO BE FREED
END17
K3
FREEMAIN
FREE STORAGE FOR ATLCB AND ADEQS
K4
ANY MORE ATLCBS TO BE FREED
YES
NO
OUTRTN
K5
RETURN
A4
END20 ENTRY
B4
ATTNINQ
INQUIRE IF ATTENTION OCCURRED
C4
ANY ATTENTION INFORMATION AVAIL
YES
NO
D4
RETURN

Chart FC. ENATN Subroutine

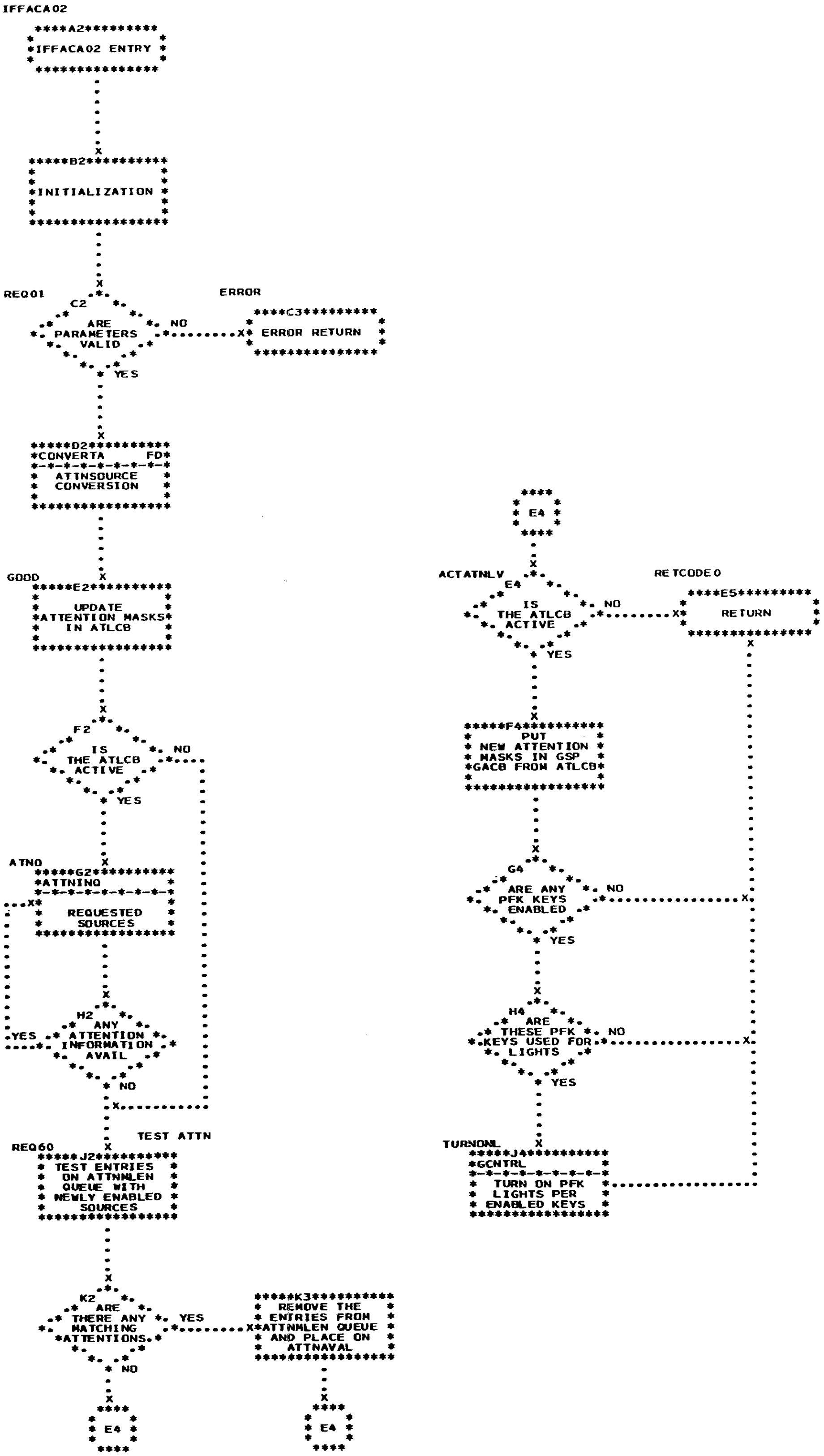
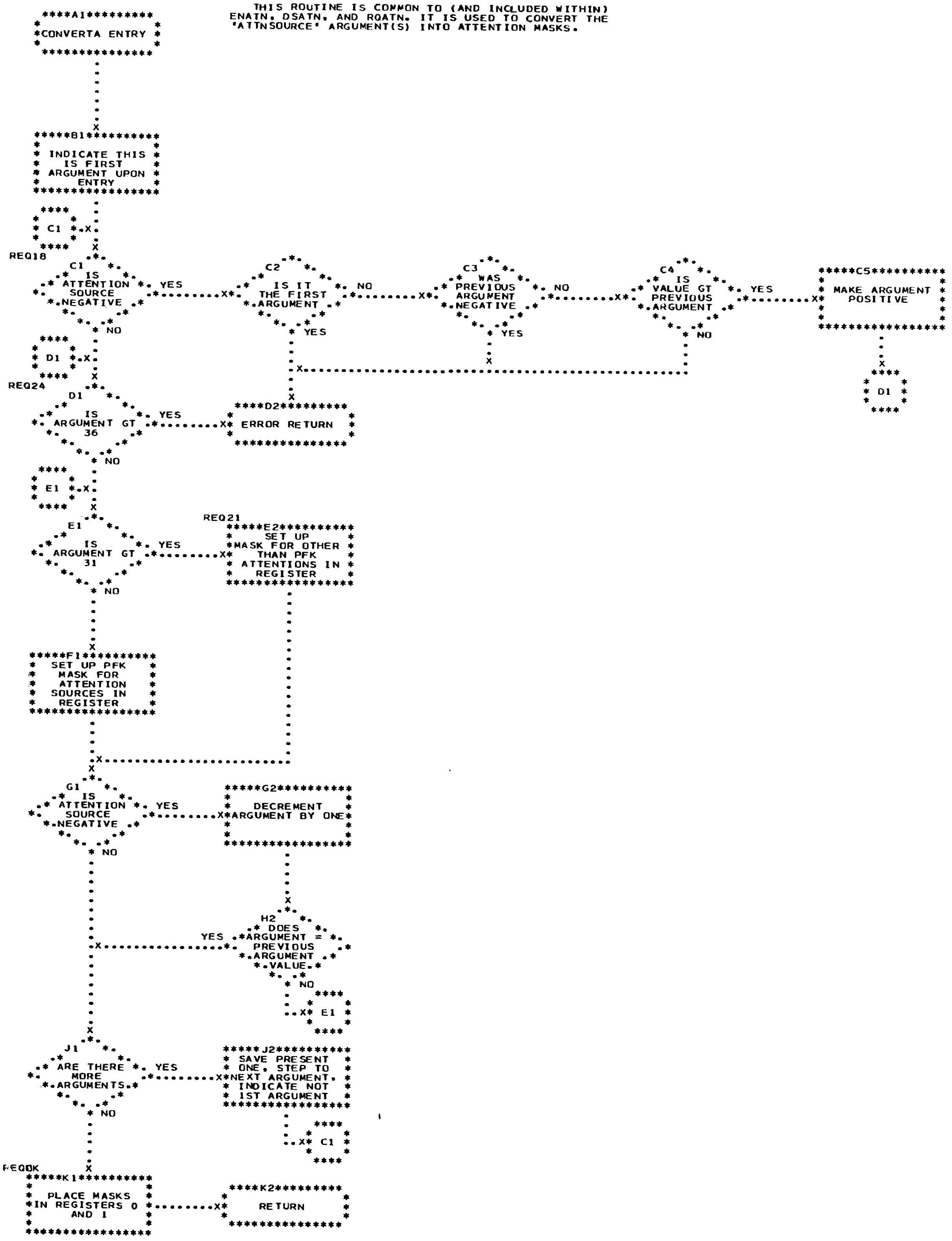


Chart FD. CONVERTA Routine

THIS ROUTINE IS COMMON TO (AND INCLUDED WITHIN) ENATN, DSATN, AND RQATN. IT IS USED TO CONVERT THE 'ATTNSOURCE' ARGUMENT(S) INTO ATTENTION MASKS.



A1
CONVERTA ENTRY
B1
INDICATE THIS IS FIRST ARGUMENT UPON ENTRY
C1
REQ18
C1
IS ATTENTION SOURCE NEGATIVE
YES
NO
C2
IS IT THE FIRST ARGUMENT
NO
YES
C3
WAS PREVIOUS ARGUMENT NEGATIVE
NO
YES
C4
IS VALUE GT PREVIOUS ARGUMENT
YES
NO
C5
MAKE ARGUMENT POSITIVE
D1
D1
REQ24
D1
IS ARGUMENT GT 36
YES
NO
D2
ERROR RETURN
E1
E1
IS ARGUMENT GT 31
YES
NO
REQ21
E2
SET UP MASK FOR OTHER THAN PFK ATTENTIONS IN REGISTER
F1
SET UP PFK MASK FOR ATTENTION SOURCES IN REGISTER
G1
IS ATTENTION SOURCE NEGATIVE
YES
NO
G2
DECREMENT ARGUMENT BY ONE
H2
DOES ARGUMENT = PREVIOUS ARGUMENT VALUE
YES
NO
E1
J1
ARE THERE MORE ARGUMENTS
YES
NO
J2
SAVE PRESENT ONE, STEP TO NEXT ARGUMENT, INDICATE NOT 1ST ARGUMENT
C1
REQOK
K1
PLACE MASKS IN REGISTERS 0 AND 1
K2
RETURN

Chart FE. RQATN Subroutine

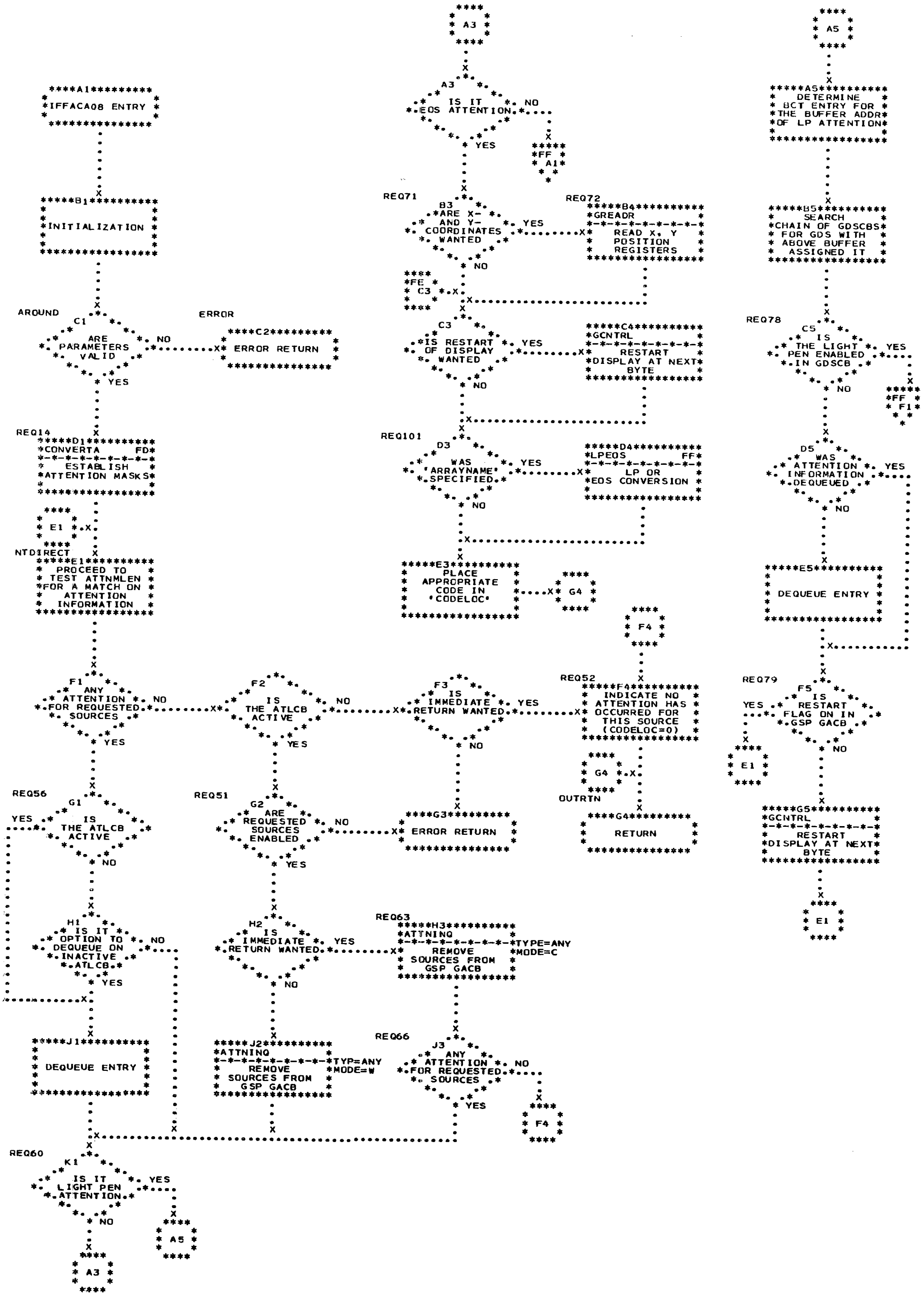

A1
IFFACA08 ENTRY
B1
INITIALIZATION
AROUND
C1
ARE PARAMETERS VALID
NO
YES
ERROR
C2
ERROR RETURN
REQ14
D1
CONVERTA FD
ESTABLISH ATTENTION MASKS
E1
NTDIRECT
E1
PROCEED TO TEST ATTNMLEN FOR A MATCH ON ATTENTION INFORMATION
F1
ANY ATTENTION FOR REQUESTED SOURCES
NO
YES
REQ56
G1
IS THE ATLCB ACTIVE
YES
NO
H1
IS IT OPTION TO DEQUEUE ON INACTIVE ATLCB
NO
YES
J1
DEQUEUE ENTRY
REQ60
K1
IS IT LIGHT PEN ATTENTION
YES
NO
A5
A3
F2
IS THE ATLCB ACTIVE
NO
YES
REQ51
G2
ARE REQUESTED SOURCES ENABLED
NO
YES
H2
IS IMMEDIATE RETURN WANTED
YES
NO
J2
ATTNINQ
REMOVE SOURCES FROM GSP GACB
TYP=ANY
MODE=W
A3
A3
IS IT EOS ATTENTION
NO
YES
FF
A1
REQ71
B3
ARE X- AND Y- COORDINATES WANTED
YES
NO
REQ72
B4
GREADR
READ X, Y POSITION REGISTERS
FE
C3
C3
IS RESTART OF DISPLAY WANTED
YES
NO
C4
GCNTRL
RESTART DISPLAY AT NEXT BYTE
REQ101
D3
WAS 'ARRAYNAME' SPECIFIED
YES
NO
D4
LPEOS FF
LP OR EOS CONVERSION
E3
PLACE APPROPRIATE CODE IN 'CODELOC'
G4
F3
IS IMMEDIATE RETURN WANTED
YES
NO
G3
ERROR RETURN
REQ63
H3
ATTNINQ
REMOVE SOURCES FROM GSP GACB
TYPE=ANY
MODE=C
REQ66
J3
ANY ATTENTION FOR REQUESTED SOURCES
NO
YES
F4
F4
REQ52
F4
INDICATE NO ATTENTION HAS OCCURRED FOR THIS SOURCE (CODELOC=0)
G4
OUTRTN
G4
RETURN
A5
A5
DETERMINE BCT ENTRY FOR THE BUFFER ADDR OF LP ATTENTION
B5
SEARCH CHAIN OF GDSCBS FOR GDS WITH ABOVE BUFFER ASSIGNED IT
REQ78
C5
IS THE LIGHT PEN ENABLED IN GDSCB
YES
NO
FF
F1
D5
WAS ATTENTION INFORMATION DEQUEUED
YES
NO
E5
DEQUEUE ENTRY
REQ79
F5
IS RESTART FLAG ON IN GSP GACB
YES
NO
E1
G5
GCNTRL
RESTART DISPLAY AT NEXT BYTE
E1

Chart FF. RQATN Subroutine (continued)

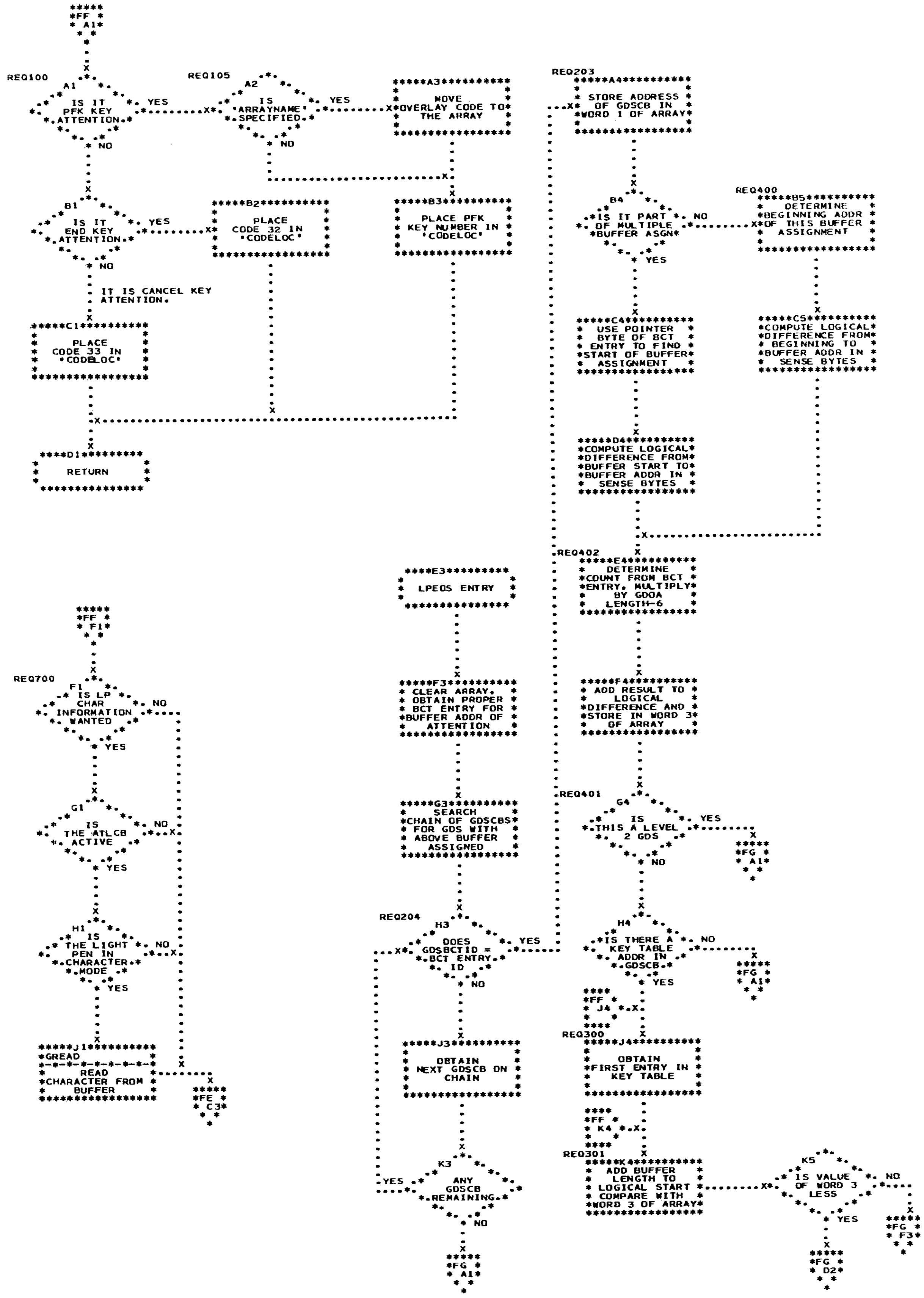

FF A1
REQ100
A1
IS IT PFK KEY ATTENTION
YES
NO
REQ105
A2
IS 'ARRAYNAME' SPECIFIED
YES
NO
A3
MOVE OVERLAY CODE TO THE ARRAY
REQ203
A4
STORE ADDRESS OF GDSCB IN WORD 1 OF ARRAY
B1
IS IT END KEY ATTENTION
YES
NO
B2
PLACE CODE 32 IN 'CODELOC'
B3
PLACE PFK KEY NUMBER IN 'CODELOC'
B4
IS IT PART OF MULTIPLE BUFFER ASGN
NO
YES
REQ400
B5
DETERMINE BEGINNING ADDR OF THIS BUFFER ASSIGNMENT
IT IS CANCEL KEY ATTENTION.
C1
PLACE CODE 33 IN 'CODELOC'
C4
USE POINTER BYTE OF BCT ENTRY TO FIND START OF BUFFER ASSIGNMENT
C5
COMPUTE LOGICAL DIFFERENCE FROM BEGINNING TO BUFFER ADDR IN SENSE BYTES
D1
RETURN
D4
COMPUTE LOGICAL DIFFERENCE FROM BUFFER START TO BUFFER ADDR IN SENSE BYTES
REQ402
E4
DETERMINE COUNT FROM BCT ENTRY, MULTIPLY BY GDOA LENGTH-6
E3
LPEOS ENTRY
FF F1
REQ700
F1
IS LP CHAR INFORMATION WANTED
NO
YES
F3
CLEAR ARRAY. OBTAIN PROPER BCT ENTRY FOR BUFFER ADDR OF ATTENTION
F4
ADD RESULT TO LOGICAL DIFFERENCE AND STORE IN WORD 3 OF ARRAY
G1
IS THE ATLCB ACTIVE
NO
YES
G3
SEARCH CHAIN OF GDSCBS FOR GDS WITH ABOVE BUFFER ASSIGNED
REQ401
G4
IS THIS A LEVEL 2 GDS
YES
NO
FG A1
REQ204
H3
DOES GDSBCTID = BCT ENTRY ID
YES
NO
H1
IS THE LIGHT PEN IN CHARACTER MODE
NO
YES
H4
IS THERE A KEY TABLE ADDR IN GDSCB
NO
YES
FG A1
FF J4
REQ300
J4
OBTAIN FIRST ENTRY IN KEY TABLE
J1
GREAD
READ CHARACTER FROM BUFFER
FE C3
J3
OBTAIN NEXT GDSCB ON CHAIN
FF K4
REQ301
K4
ADD BUFFER LENGTH TO LOGICAL START COMPARE WITH WORD 3 OF ARRAY
K5
IS VALUE OF WORD 3 LESS
NO
YES
FG F3
FG D2
K3
ANY GDSCB REMAINING
YES
NO
FG A1

Chart FG. RQATN Subroutine (continued)

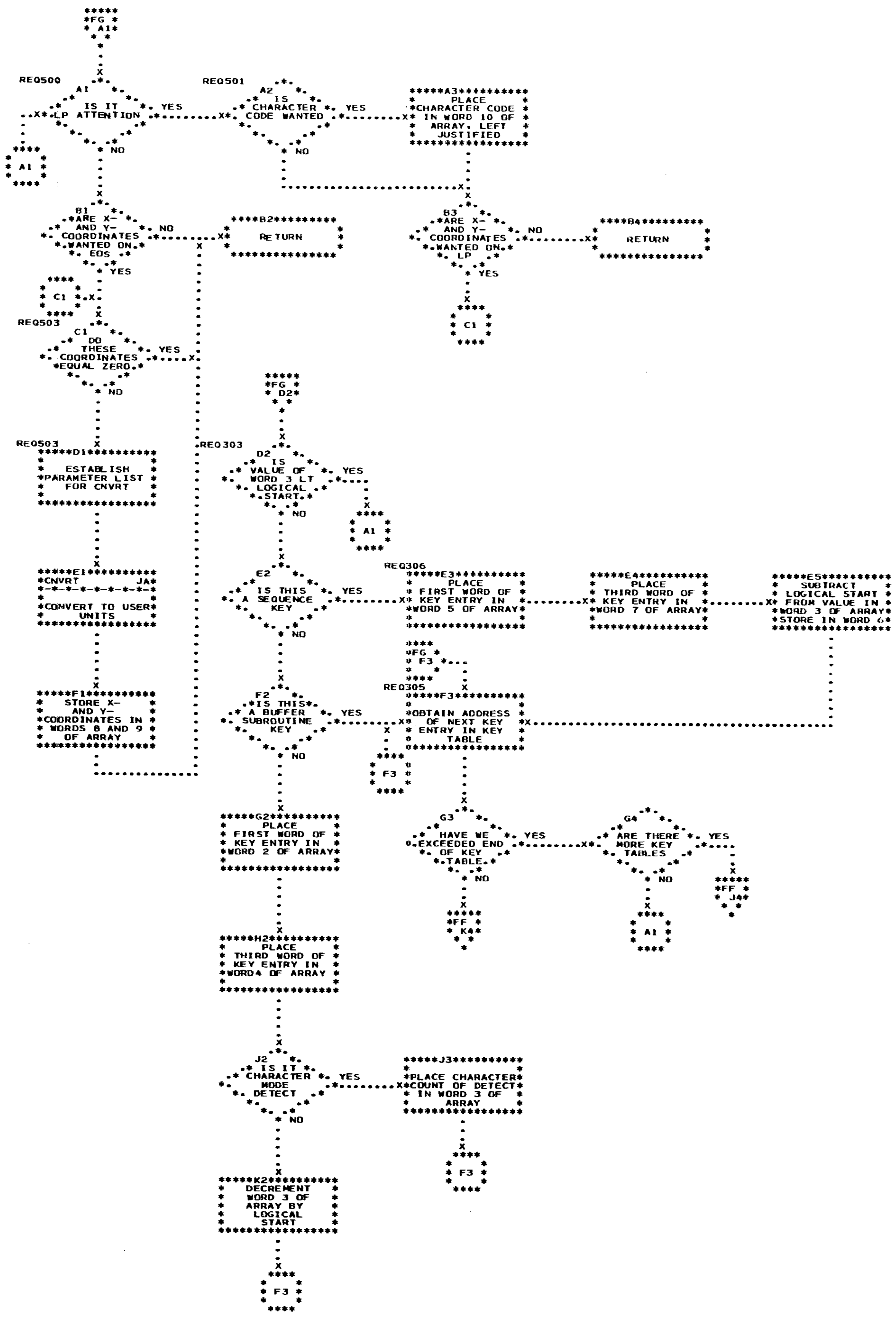
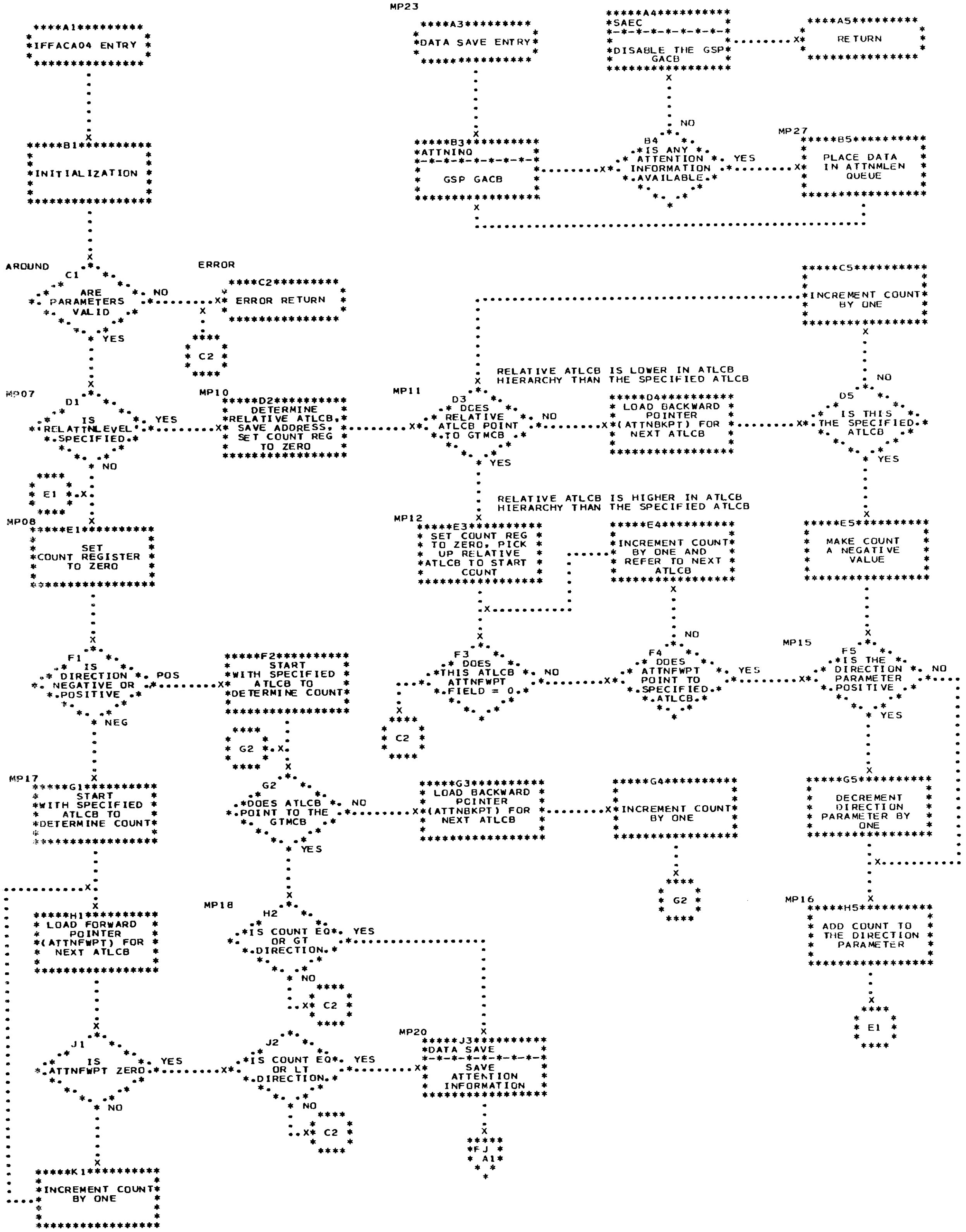

FG A1
REQ500
A1
IS IT LP ATTENTION
YES
NO
A1
REQ501
A2
IS CHARACTER CODE WANTED
YES
NO
A3
PLACE CHARACTER CODE IN WORD 10 OF ARRAY, LEFT JUSTIFIED
B1
ARE X- AND Y- COORDINATES WANTED ON EOS
NO
YES
B2
RETURN
B3
ARE X- AND Y- COORDINATES WANTED ON LP
NO
YES
B4
RETURN
C1
C1
REQ503
C1
DO THESE COORDINATES EQUAL ZERO
YES
NO
REQ503
D1
ESTABLISH PARAMETER LIST FOR CNVRT
E1
CNVRT JA
CONVERT TO USER UNITS
F1
STORE X- AND Y- COORDINATES IN WORDS 8 AND 9 OF ARRAY
FG D2
REQ303
D2
IS VALUE OF WORD 3 LT LOGICAL START
YES
NO
A1
E2
IS THIS A SEQUENCE KEY
YES
NO
REQ306
E3
PLACE FIRST WORD OF KEY ENTRY IN WORD 5 OF ARRAY
E4
PLACE THIRD WORD OF KEY ENTRY IN WORD 7 OF ARRAY
E5
SUBTRACT LOGICAL START FROM VALUE IN WORD 3 OF ARRAY STORE IN WORD 6
FG F3
REQ305
F3
OBTAIN ADDRESS OF NEXT KEY ENTRY IN KEY TABLE
F2
IS THIS A BUFFER SUBROUTINE KEY
YES
NO
F3
G2
PLACE FIRST WORD OF KEY ENTRY IN WORD 2 OF ARRAY
G3
HAVE WE EXCEEDED END OF KEY TABLE
YES
NO
FF K4
G4
ARE THERE MORE KEY TABLES
YES
NO
A1
FF J4
H2
PLACE THIRD WORD OF KEY ENTRY IN WORD4 OF ARRAY
J2
IS IT CHARACTER MODE DETECT
YES
NO
J3
PLACE CHARACTER COUNT OF DETECT IN WORD 3 OF ARRAY
F3
K2
DECREMENT WORD 3 OF ARRAY BY LOGICAL START
F3

Chart FH. MPATL Subroutine



A1 IFFACA04 ENTRY
B1 INITIALIZATION
AROUND
C1 ARE PARAMETERS VALID
NO
YES
ERROR
C2 ERROR RETURN
C2
MP07
D1 IS RELATTNLEVEL SPECIFIED
YES
NO
E1
MP10
D2 DETERMINE RELATIVE ATLCB. SAVE ADDRESS. SET COUNT REG TO ZERO
MP08
E1 SET COUNT REGISTER TO ZERO
F1 IS DIRECTION NEGATIVE OR POSITIVE
POS
NEG
F2 START WITH SPECIFIED ATLCB TO DETERMINE COUNT
G2
MP17
G1 START WITH SPECIFIED ATLCB TO DETERMINE COUNT
H1 LOAD FORWARD POINTER (ATTNFWPT) FOR NEXT ATLCB
J1 IS ATTNFWPT ZERO
YES
NO
K1 INCREMENT COUNT BY ONE
G2 DOES ATLCB POINT TO THE GTMCB
NO
YES
MP18
H2 IS COUNT EQ OR GT DIRECTION
YES
NO
C2
J2 IS COUNT EQ OR LT DIRECTION
YES
NO
C2
MP23
A3 DATA SAVE ENTRY
B3 ATTNINQ
GSP GACB
A4 SAEC
DISABLE THE GSP GACB
A5 RETURN
B4 IS ANY ATTENTION INFORMATION AVAILABLE
NO
YES
MP27
B5 PLACE DATA IN ATTNMLEN QUEUE
C5 INCREMENT COUNT BY ONE
RELATIVE ATLCB IS LOWER IN ATLCB HIERARCHY THAN THE SPECIFIED ATLCB
MP11
D3 DOES RELATIVE ATLCB POINT TO GTMCB
NO
YES
D4 LOAD BACKWARD POINTER (ATTNBKPT) FOR NEXT ATLCB
D5 IS THIS THE SPECIFIED ATLCB
NO
YES
RELATIVE ATLCB IS HIGHER IN ATLCB HIERARCHY THAN THE SPECIFIED ATLCB
MP12
E3 SET COUNT REG TO ZERO. PICK UP RELATIVE ATLCB TO START CCUNT
E4 INCREMENT COUNT BY ONE AND REFER TO NEXT ATLCB
E5 MAKE COUNT A NEGATIVE VALUE
F3 DOES THIS ATLCB ATTNFWPT FIELD = 0
NO
C2
F4 DOES ATTNFWPT POINT TO SPECIFIED ATLCB
NO
YES
MP15
F5 IS THE DIRECTION PARAMETER POSITIVE
NO
YES
G3 LOAD BACKWARD POINTER (ATTNBKPT) FOR NEXT ATLCB
G4 INCREMENT COUNT BY ONE
G2
G5 DECREMENT DIRECTION PARAMETER BY ONE
MP16
H5 ADD COUNT TO THE DIRECTION PARAMETER
E1
MP20
J3 DATA SAVE
SAVE ATTENTION INFORMATION
FJ A1

Chart FJ. MPATL Subroutine (continued)

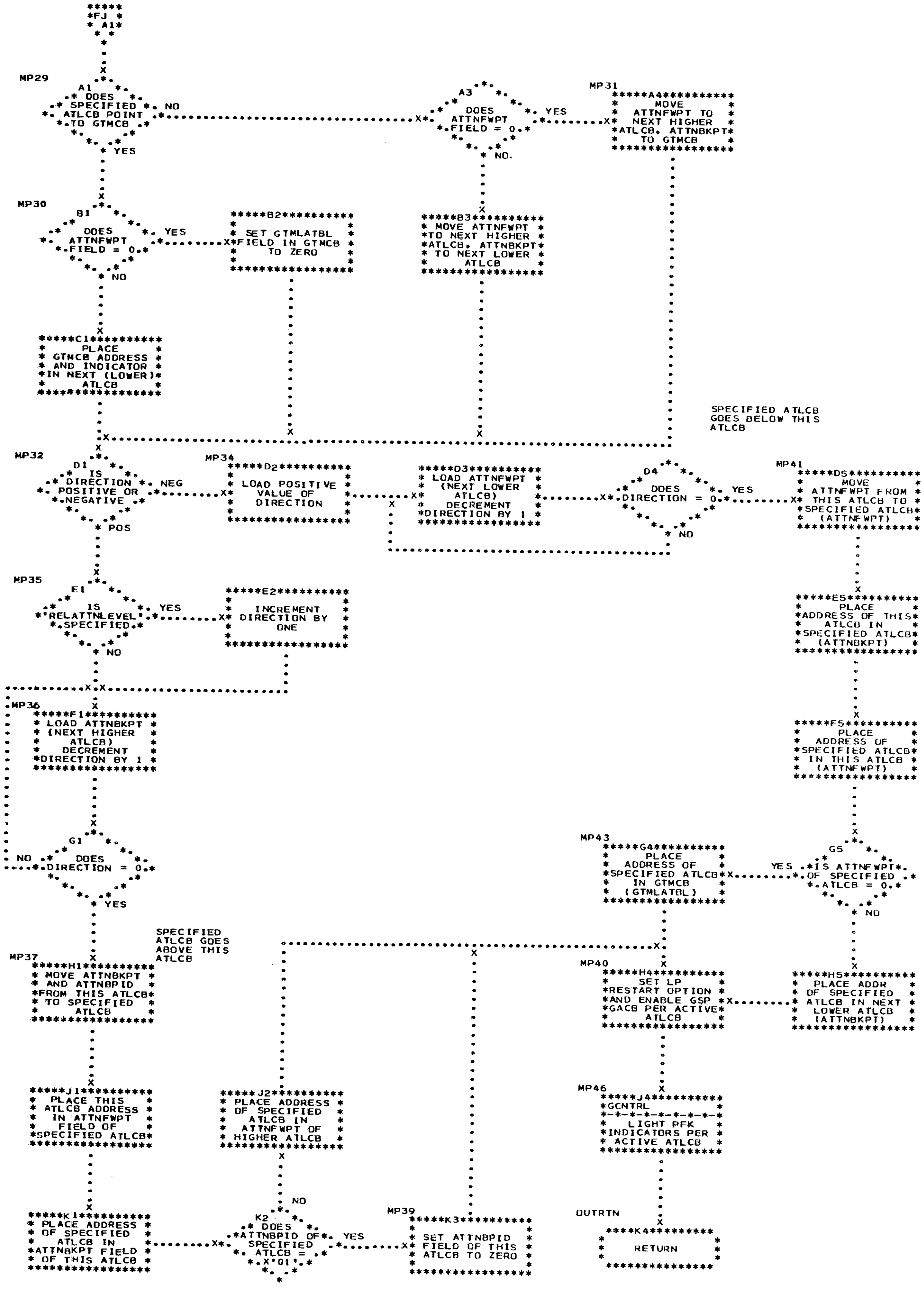

FJ A1
MP29
A1
DOES SPECIFIED ATLCB POINT TO GTMCB
NO
YES
A3
DOES ATTNFWPT FIELD = 0
YES
NO.
MP31
A4
MOVE ATTNFWPT TO NEXT HIGHER ATLCB. ATTNBKPT TO GTMCB
MP30
B1
DOES ATTNFWPT FIELD = 0
YES
NO
B2
SET GTMLATBL FIELD IN GTMCB TO ZERO
B3
MOVE ATTNFWPT TO NEXT HIGHER ATLCB. ATTNBKPT TO NEXT LOWER ATLCB
C1
PLACE GTMCB ADDRESS AND INDICATOR IN NEXT (LOWER) ATLCB
SPECIFIED ATLCB GOES BELOW THIS ATLCB
MP32
D1
IS DIRECTION POSITIVE OR NEGATIVE
NEG
POS
MP34
D2
LOAD POSITIVE VALUE OF DIRECTION
D3
LOAD ATTNFWPT (NEXT LOWER ATLCB) DECREMENT DIRECTION BY 1
D4
DOES DIRECTION = 0
YES
NO
MP41
D5
MOVE ATTNFWPT FROM THIS ATLCB TO SPECIFIED ATLCB (ATTNFWPT)
MP35
E1
IS RELATTNLEVEL SPECIFIED
YES
NO
E2
INCREMENT DIRECTION BY ONE
E5
PLACE ADDRESS OF THIS ATLCB IN SPECIFIED ATLCB (ATTNBKPT)
MP36
F1
LOAD ATTNBKPT (NEXT HIGHER ATLCB) DECREMENT DIRECTION BY 1
F5
PLACE ADDRESS OF SPECIFIED ATLCB IN THIS ATLCB (ATTNFWPT)
G1
DOES DIRECTION = 0
NO
YES
MP43
G4
PLACE ADDRESS OF SPECIFIED ATLCB IN GTMCB (GTMLATBL)
G5
IS ATTNFWPT OF SPECIFIED ATLCB = 0
YES
NO
SPECIFIED ATLCB GOES ABOVE THIS ATLCB
MP37
H1
MOVE ATTNBKPT AND ATTNBPID FROM THIS ATLCB TO SPECIFIED ATLCB
MP40
H4
SET LP RESTART OPTION AND ENABLE GSP GACB PER ACTIVE ATLCB
H5
PLACE ADDR OF SPECIFIED ATLCB IN NEXT LOWER ATLCB (ATTNBKPT)
J1
PLACE THIS ATLCB ADDRESS IN ATTNFWPT FIELD OF SPECIFIED ATLCB
J2
PLACE ADDRESS OF SPECIFIED ATLCB IN ATTNFWPT OF HIGHER ATLCB
MP46
J4
GCNTRL
LIGHT PFK INDICATORS PER ACTIVE ATLCB
K1
PLACE ADDRESS OF SPECIFIED ATLCB IN ATTNBKPT FIELD OF THIS ATLCB
K2
DOES ATTNBPID OF SPECIFIED ATLCB = X'01'
NO
YES
MP39
K3
SET ATTNBPID FIELD OF THIS ATLCB TO ZERO
OUTRTN
K4
RETURN

Chart GA. PLSTR Subroutine

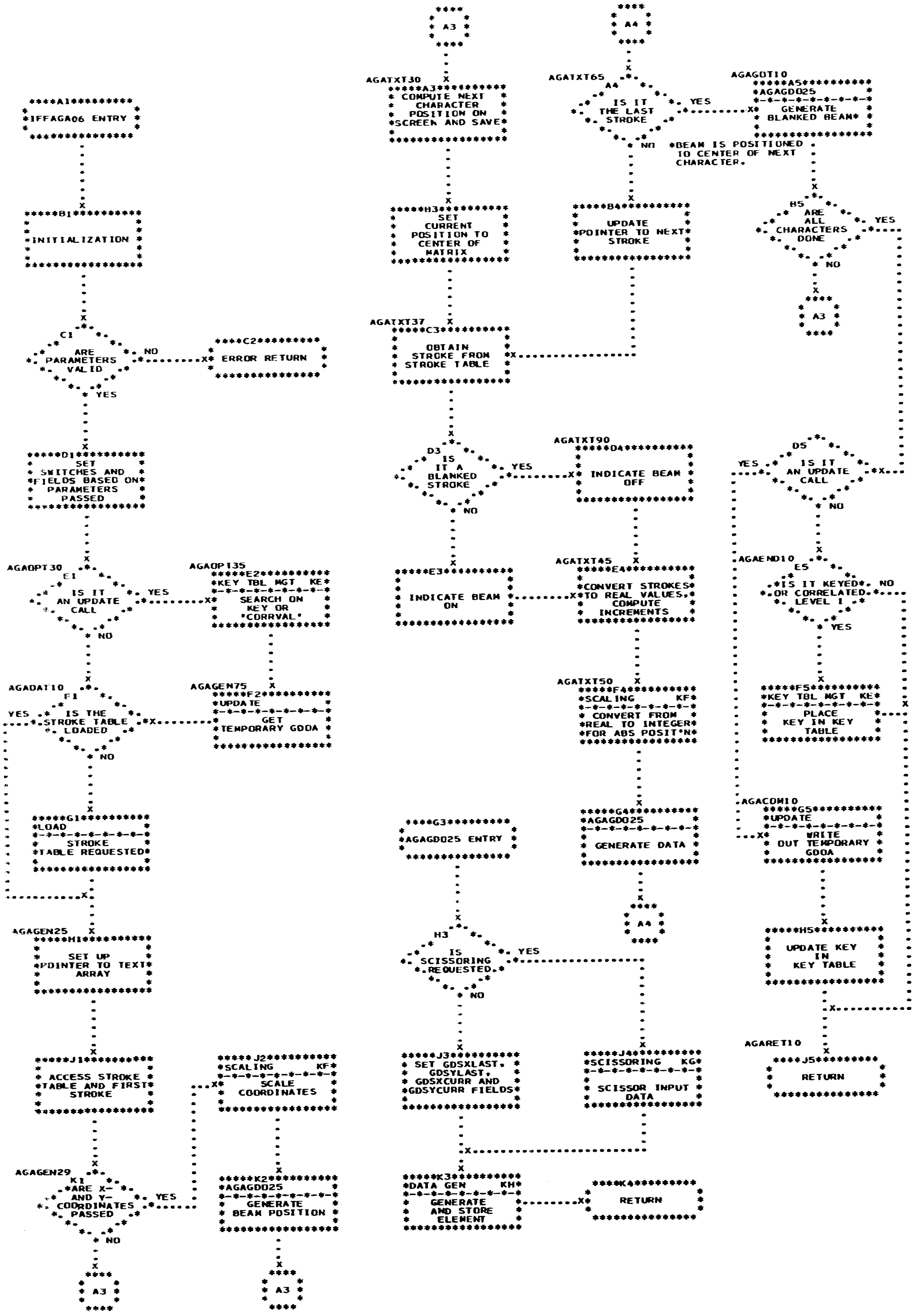


Chart HA. ORGEN Subroutine

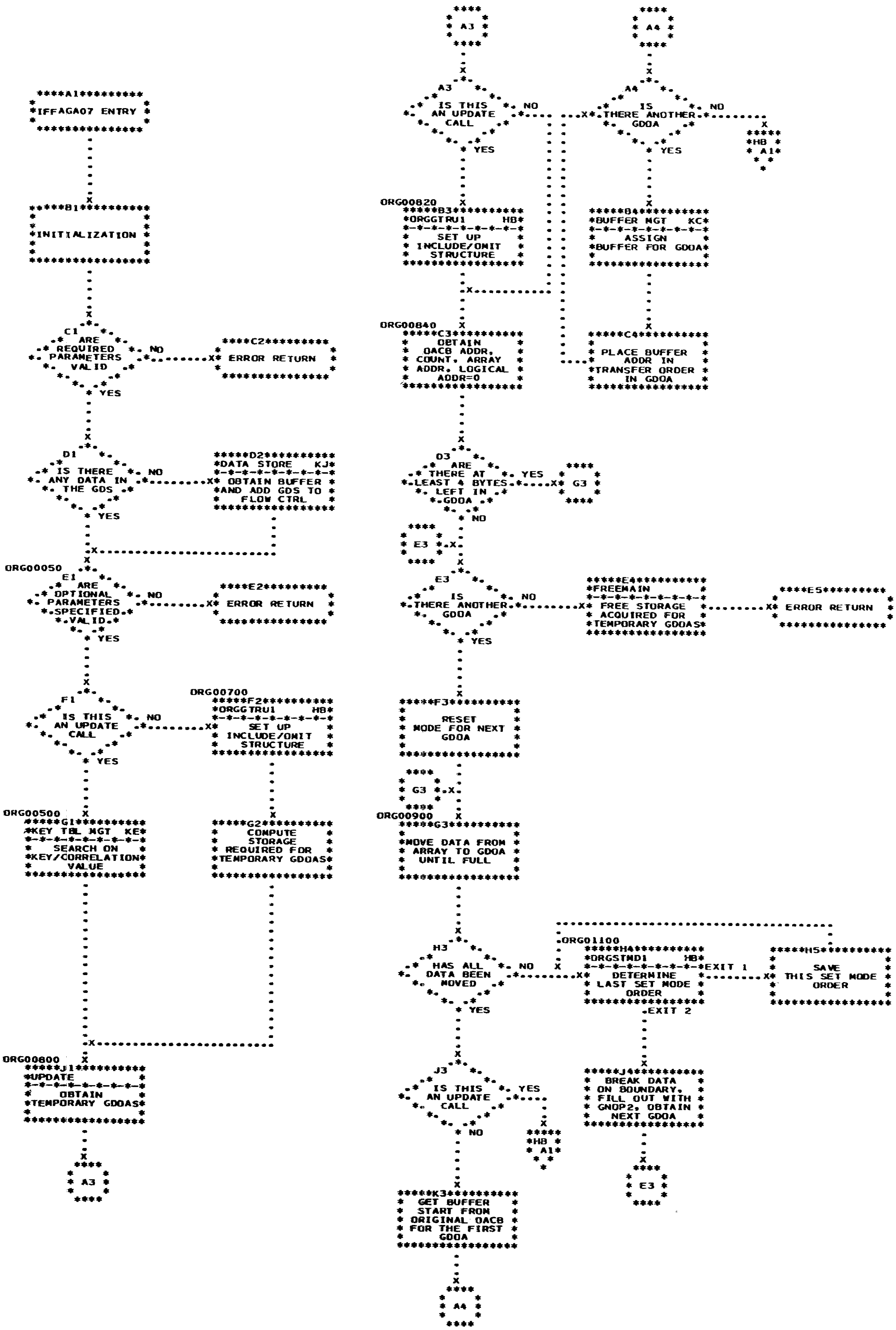

A1 IFFAGA07 ENTRY
B1 INITIALIZATION
C1 ARE REQUIRED PARAMETERS VALID
NO
YES
C2 ERROR RETURN
D1 IS THERE ANY DATA IN THE GDS
NO
YES
D2 DATA STORE KJ OBTAIN BUFFER AND ADD GDS TO FLOW CTRL
ORG00050
E1 ARE OPTIONAL PARAMETERS SPECIFIED VALID
NO
YES
E2 ERROR RETURN
F1 IS THIS AN UPDATE CALL
NO
YES
ORG00700
F2 ORGGTRU1 HB SET UP INCLUDE/OMIT STRUCTURE
ORG00500
G1 KEY TBL MGT KE SEARCH ON KEY/CORRELATION VALUE
G2 COMPUTE STORAGE REQUIRED FOR TEMPORARY GDOAS
ORG00800
J1 UPDATE OBTAIN TEMPORARY GDOAS
A3
A3 IS THIS AN UPDATE CALL
NO
YES
ORG00820
B3 ORGGTRU1 HB SET UP INCLUDE/OMIT STRUCTURE
ORG00840
C3 OBTAIN OACB ADDR, COUNT, ARRAY ADDR, LOGICAL ADDR=0
D3 ARE THERE AT LEAST 4 BYTES LEFT IN GDOA
YES
NO
G3
E3
E3 IS THERE ANOTHER GDOA
NO
YES
E4 FREEMAIN FREE STORAGE ACQUIRED FOR TEMPORARY GDOAS
E5 ERROR RETURN
F3 RESET MODE FOR NEXT GDOA
G3
ORG00900
G3 MOVE DATA FROM ARRAY TO GDOA UNTIL FULL
H3 HAS ALL DATA BEEN MOVED
NO
YES
ORG01100
H4 ORGSTMD1 HB DETERMINE LAST SET MODE ORDER
EXIT 1
EXIT 2
H5 SAVE THIS SET MODE ORDER
J3 IS THIS AN UPDATE CALL
YES
NO
HB A1
J4 BREAK DATA ON BOUNDARY, FILL OUT WITH GNOP2, OBTAIN NEXT GDOA
E3
K3 GET BUFFER START FROM ORIGINAL OACB FOR THE FIRST GDOA
A4
A4
A4 IS THERE ANOTHER GDOA
NO
YES
HB A1
B4 BUFFER MGT KC ASSIGN BUFFER FOR GDOA
C4 PLACE BUFFER ADDR IN TRANSFER ORDER IN GDOA

Chart HB. ORGEN Subroutine (continued)

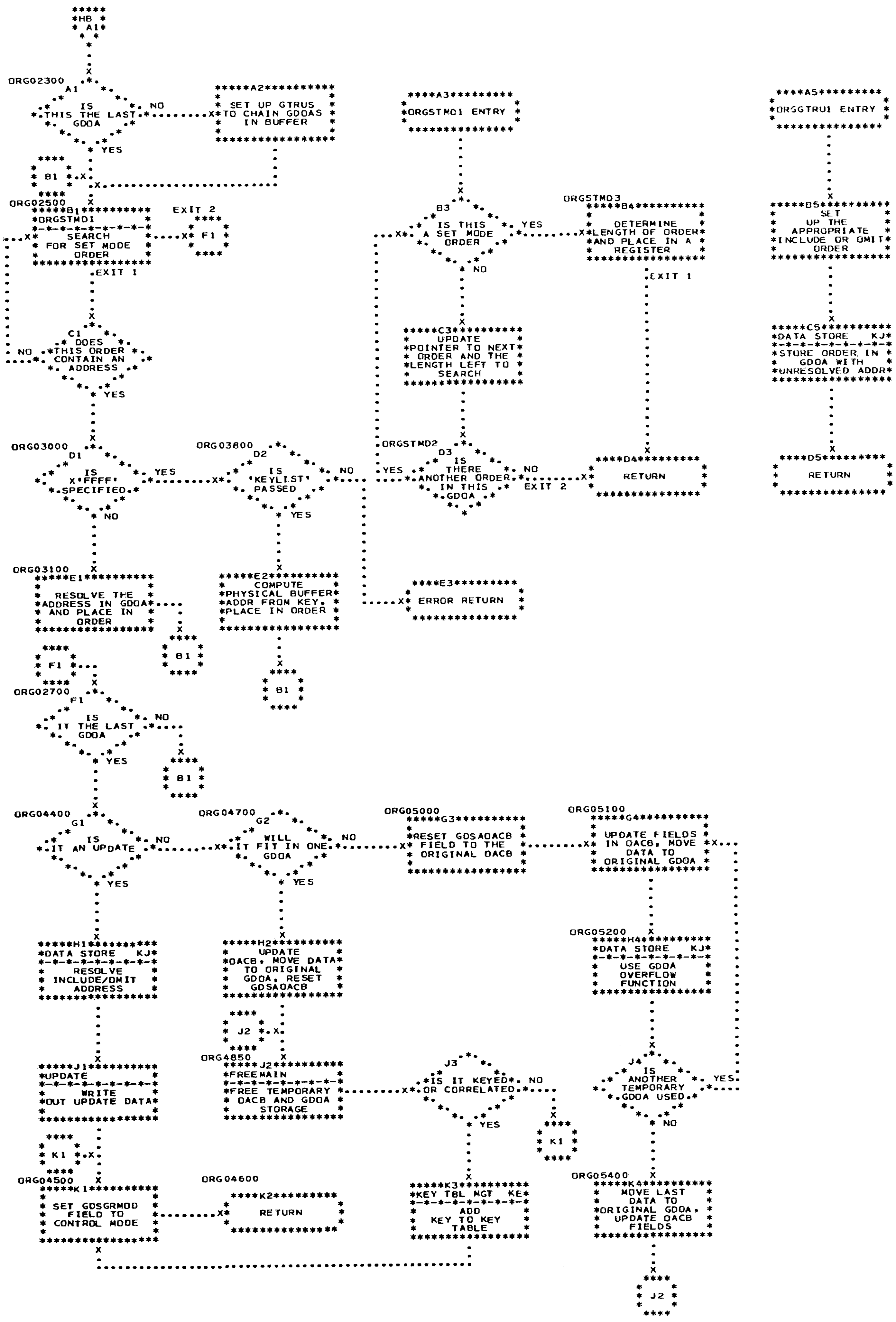

HB A1
ORG02300
A1 IS THIS THE LAST GDOA
NO
YES
A2 SET UP GTRUS TO CHAIN GDOAS IN BUFFER
A3 ORGSTMD1 ENTRY
A5 ORGGTRU1 ENTRY
B1
ORG02500
B1 ORGSTMD1 SEARCH FOR SET MODE ORDER
EXIT 2
F1
EXIT 1
B3 IS THIS A SET MODE ORDER
YES
NO
ORGSTMO3
B4 DETERMINE LENGTH OF ORDER AND PLACE IN A REGISTER
EXIT 1
B5 SET UP THE APPROPRIATE INCLUDE OR OMIT ORDER
C1 DOES THIS ORDER CONTAIN AN ADDRESS
NO
YES
C3 UPDATE POINTER TO NEXT ORDER AND THE LENGTH LEFT TO SEARCH
C5 DATA STORE KJ STORE ORDER IN GDOA WITH UNRESOLVED ADDR
ORG03000
D1 IS X'FFFF' SPECIFIED
YES
NO
ORG03800
D2 IS 'KEYLIST' PASSED
NO
YES
ORGSTMD2
D3 IS THERE ANOTHER ORDER IN THIS GDOA
YES
NO
EXIT 2
D4 RETURN
D5 RETURN
ORG03100
E1 RESOLVE THE ADDRESS IN GDOA AND PLACE IN ORDER
B1
E2 COMPUTE PHYSICAL BUFFER ADDR FROM KEY, PLACE IN ORDER
B1
E3 ERROR RETURN
F1
ORG02700
F1 IS IT THE LAST GDOA
NO
B1
YES
ORG04400
G1 IS IT AN UPDATE
NO
YES
ORG04700
G2 WILL IT FIT IN ONE GDOA
NO
YES
ORG05000
G3 RESET GDSAOACB FIELD TO THE ORIGINAL OACB
ORG05100
G4 UPDATE FIELDS IN OACB, MOVE DATA TO ORIGINAL GDOA
H1 DATA STORE KJ RESOLVE INCLUDE/OMIT ADDRESS
H2 UPDATE OACB, MOVE DATA TO ORIGINAL GDOA, RESET GDSAOACB
ORG05200
H4 DATA STORE KJ USE GDOA OVERFLOW FUNCTION
J2
ORG4850
J1 UPDATE WRITE OUT UPDATE DATA
J2 FREEMAIN FREE TEMPORARY OACB AND GDOA STORAGE
J3 IS IT KEYED OR CORRELATED
NO
YES
K1
J4 IS ANOTHER TEMPORARY GDOA USED
YES
NO
K1
ORG04500
K1 SET GDSGRMOD FIELD TO CONTROL MODE
ORG04600
K2 RETURN
K3 KEY TBL MGT KE ADD KEY TO KEY TABLE
ORG05400
K4 MOVE LAST DATA TO ORIGINAL GDOA, UPDATE OACB FIELDS
J2

Chart JA. CNVRT Subroutine

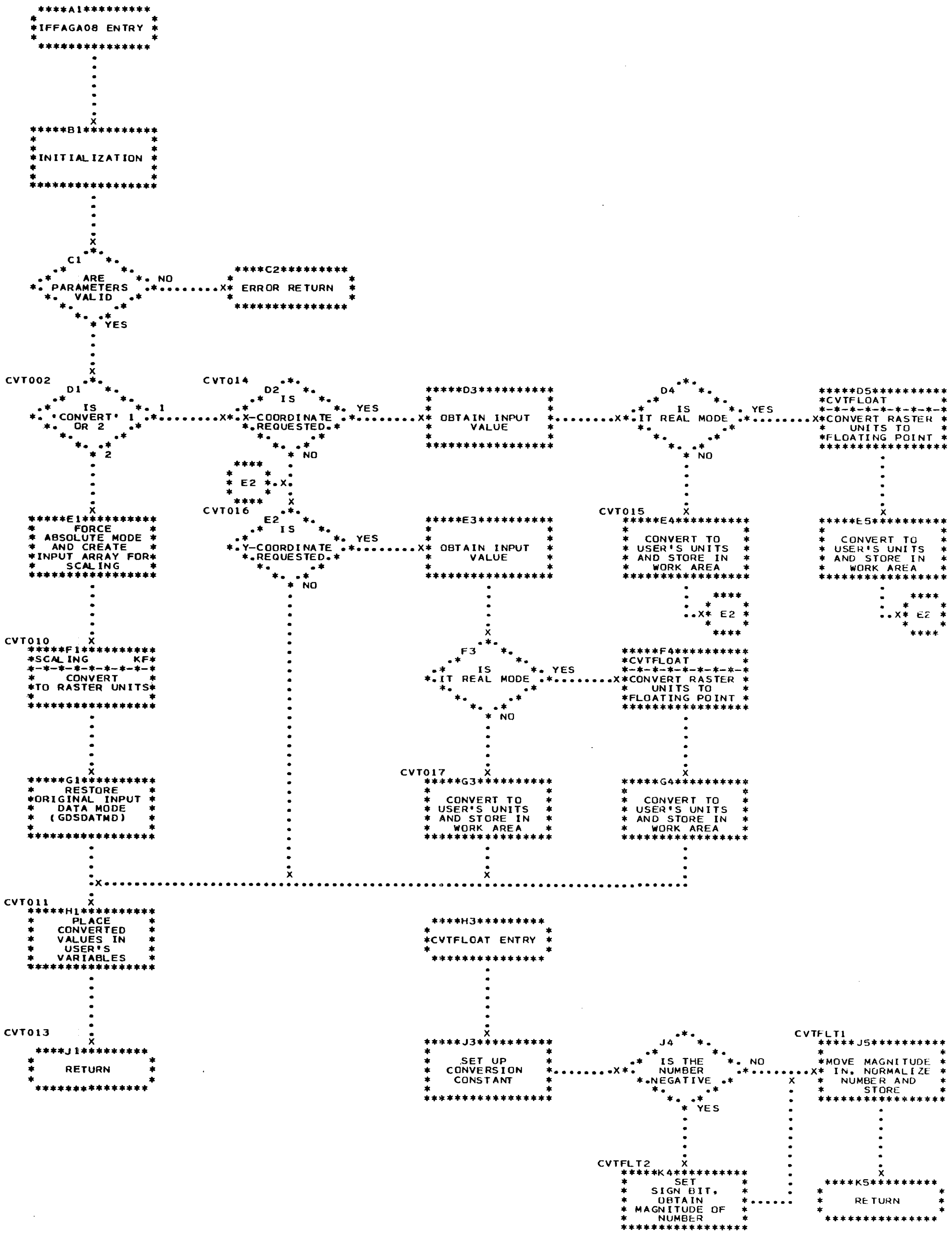
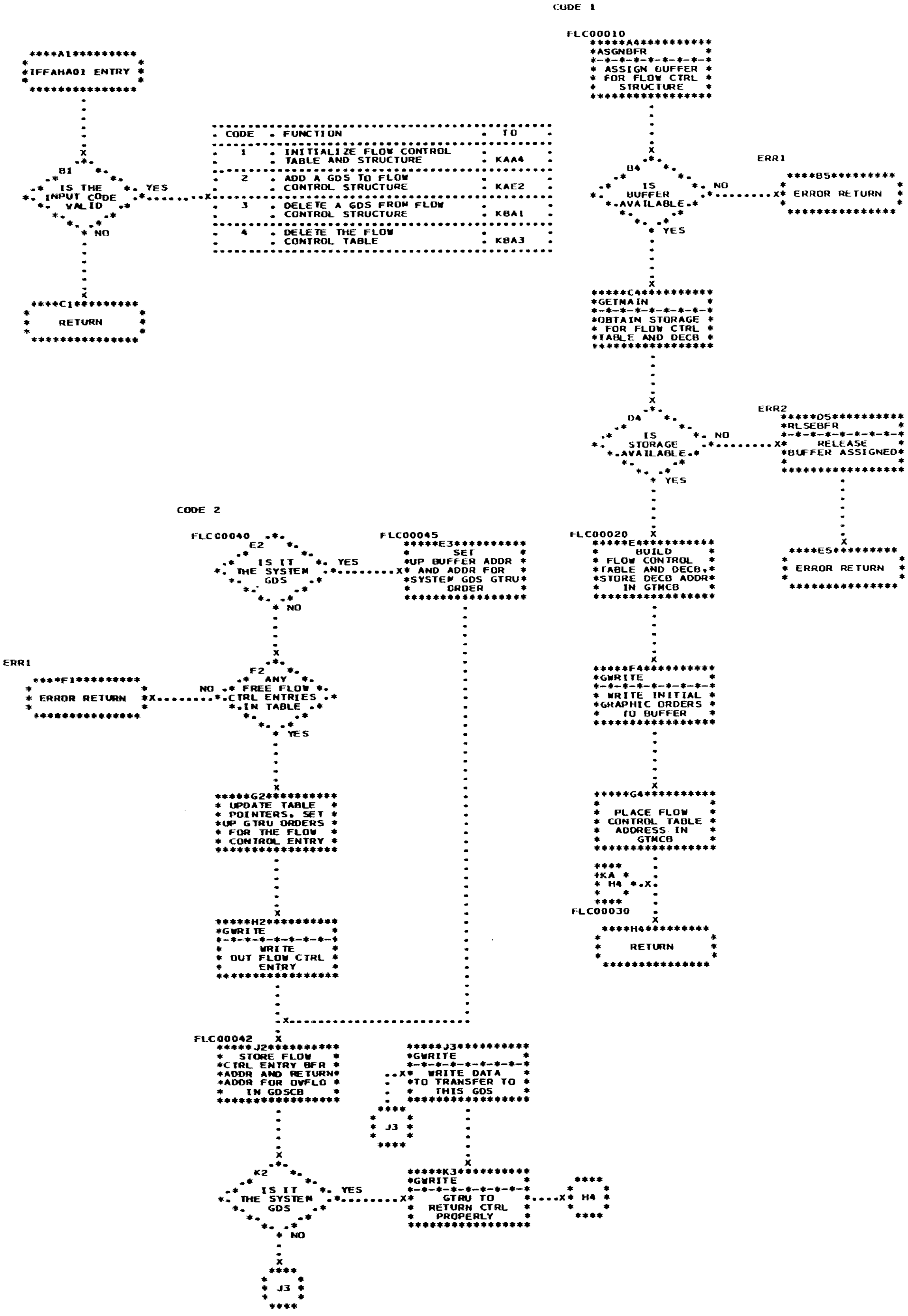

A1
IFFAGA08 ENTRY
B1
INITIALIZATION
C1
ARE PARAMETERS VALID
NO
C2
ERROR RETURN
YES
CVT002
D1
IS 'CONVERT' 1 OR 2
1
2
CVT014
D2
IS X-COORDINATE REQUESTED
YES
NO
D3
OBTAIN INPUT VALUE
D4
IS IT REAL MODE
YES
NO
D5
CVTFLOAT
CONVERT RASTER UNITS TO FLOATING POINT
E2
CVT016
E2
IS Y-COORDINATE REQUESTED
YES
NO
E1
FORCE ABSOLUTE MODE AND CREATE INPUT ARRAY FOR SCALING
E3
OBTAIN INPUT VALUE
CVT015
E4
CONVERT TO USER'S UNITS AND STORE IN WORK AREA
E2
E5
CONVERT TO USER'S UNITS AND STORE IN WORK AREA
E2
CVT010
F1
SCALING KF
CONVERT TO RASTER UNITS
F3
IS IT REAL MODE
YES
NO
F4
CVTFLOAT
CONVERT RASTER UNITS TO FLOATING POINT
CVT017
G1
RESTORE ORIGINAL INPUT DATA MODE (GDSDATMD)
G3
CONVERT TO USER'S UNITS AND STORE IN WORK AREA
G4
CONVERT TO USER'S UNITS AND STORE IN WORK AREA
CVT011
H1
PLACE CONVERTED VALUES IN USER'S VARIABLES
H3
CVTFLOAT ENTRY
CVT013
J1
RETURN
J3
SET UP CONVERSION CONSTANT
J4
IS THE NUMBER NEGATIVE
NO
YES
CVTFLT1
J5
MOVE MAGNITUDE IN. NORMALIZE NUMBER AND STORE
CVTFLT2
K4
SET SIGN BIT. OBTAIN MAGNITUDE OF NUMBER
K5
RETURN

Chart KA. Flow Control Management Routine



CODE	FUNCTION	TO
1	INITIALIZE FLOW CONTROL TABLE AND STRUCTURE	KAA4
2	ADD A GDS TO FLOW CONTROL STRUCTURE	KAE2
3	DELETE A GDS FROM FLOW CONTROL STRUCTURE	KBA1
4	DELETE THE FLOW CONTROL TABLE	KBA3

A1
IFFAHA01 ENTRY
B1
IS THE INPUT CODE VALID
YES
NO
C1
RETURN
CODE 1
FLC00010
A4
ASGNBFR
ASSIGN BUFFER FOR FLOW CTRL STRUCTURE
B4
IS BUFFER AVAILABLE
NO
YES
ERR1
B5
ERROR RETURN
C4
GETMAIN
OBTAIN STORAGE FOR FLOW CTRL TABLE AND DECB
D4
IS STORAGE AVAILABLE
NO
YES
ERR2
D5
RLSEBFR
RELEASE BUFFER ASSIGNED
E5
ERROR RETURN
FLC00020
E4
BUILD FLOW CONTROL TABLE AND DECB. STORE DECB ADDR IN GTMCB
F4
GWRITE
WRITE INITIAL GRAPHIC ORDERS TO BUFFER
G4
PLACE FLOW CONTROL TABLE ADDRESS IN GTMCB
KA H4
FLC00030
H4
RETURN
CODE 2
FLCC0040
E2
IS IT THE SYSTEM GDS
YES
NO
FLC00045
E3
SET UP BUFFER ADDR AND ADDR FOR SYSTEM GDS GTRU ORDER
ERR1
F1
ERROR RETURN
F2
ANY FREE FLOW CTRL ENTRIES IN TABLE
NO
YES
G2
UPDATE TABLE POINTERS. SET UP GTRU ORDERS FOR THE FLOW CONTROL ENTRY
H2
GWRITE
WRITE OUT FLOW CTRL ENTRY
FLC00042
J2
STORE FLOW CTRL ENTRY BFR ADDR AND RETURN ADDR FOR OVFLO IN GDSCB
J3
GWRITE
WRITE DATA TO TRANSFER TO THIS GDS
J3
K2
IS IT THE SYSTEM GDS
YES
NO
K3
GWRITE
GTRU TO RETURN CTRL PROPERLY
H4
J3

Chart KB. Flow Control Management Routine (continued)

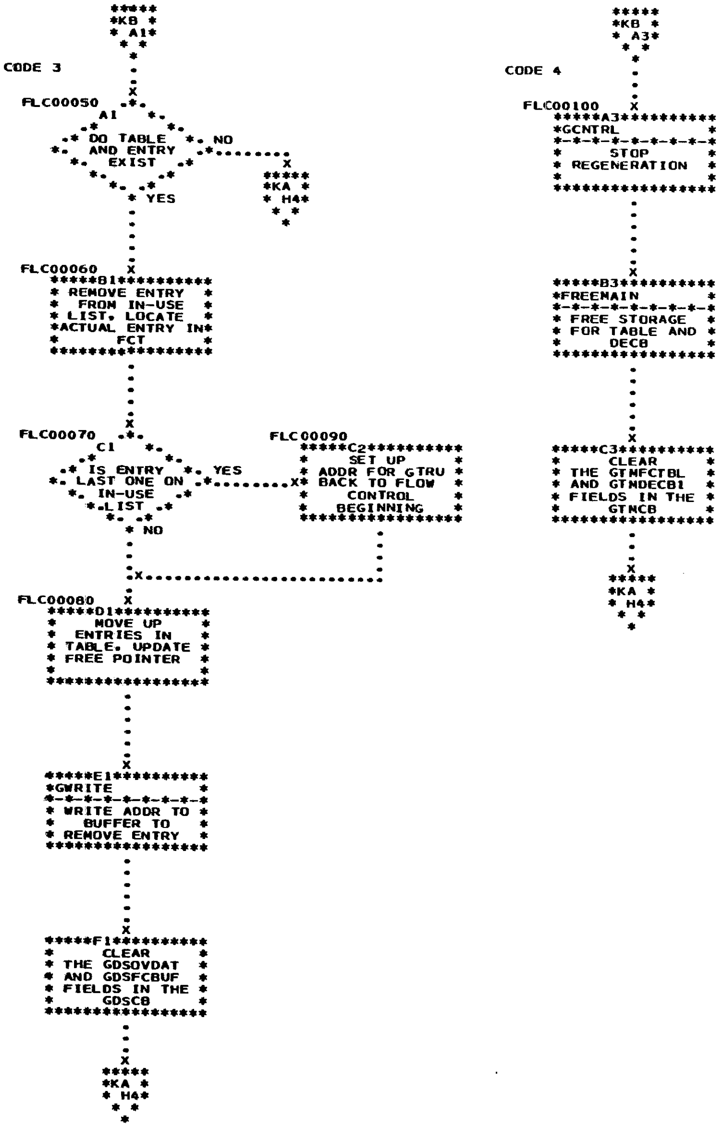
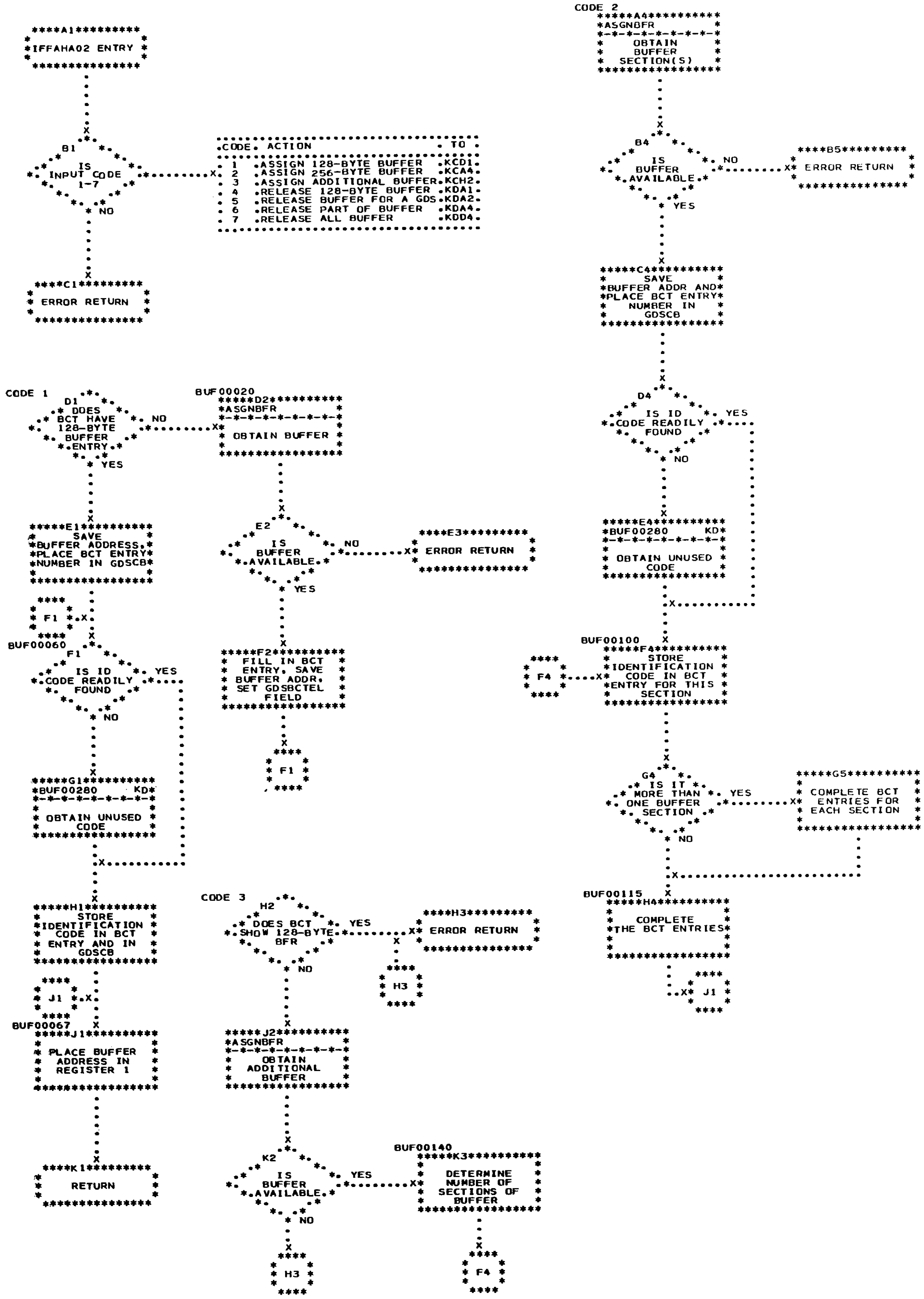

KB A1
CODE 3
FLC00050
A1
DO TABLE AND ENTRY EXIST
NO
KA H4
YES
FLC00060
B1
REMOVE ENTRY FROM IN-USE LIST. LOCATE ACTUAL ENTRY IN FCT
FLC00070
C1
IS ENTRY LAST ONE ON IN-USE LIST
YES
FLC00090
C2
SET UP ADDR FOR GTRU BACK TO FLOW CONTROL BEGINNING
NO
FLC00080
D1
MOVE UP ENTRIES IN TABLE. UPDATE FREE POINTER
E1
GWRITE
WRITE ADDR TO BUFFER TO REMOVE ENTRY
F1
CLEAR THE GDSOVDAT AND GDSFCBUF FIELDS IN THE GDSCB
KA H4
KB A3
CODE 4
FLC00100
A3
GCNTRL
STOP REGENERATION
B3
FREEMAIN
FREE STORAGE FOR TABLE AND DECB
C3
CLEAR THE GTNFCTBL AND GTNDECBI FIELDS IN THE GTNCB
KA H4

Chart KC. Buffer Management Routine



CODE	ACTION	TO
1	ASSIGN 128-BYTE BUFFER	KCD1
2	ASSIGN 256-BYTE BUFFER	KCA4
3	ASSIGN ADDITIONAL BUFFER	KCH2
4	RELEASE 128-BYTE BUFFER	KDA1
5	RELEASE BUFFER FOR A GDS	KDA2
6	RELEASE PART OF BUFFER	KDA4
7	RELEASE ALL BUFFER	KDD4

Chart KD. Buffer Management Routine (continued)

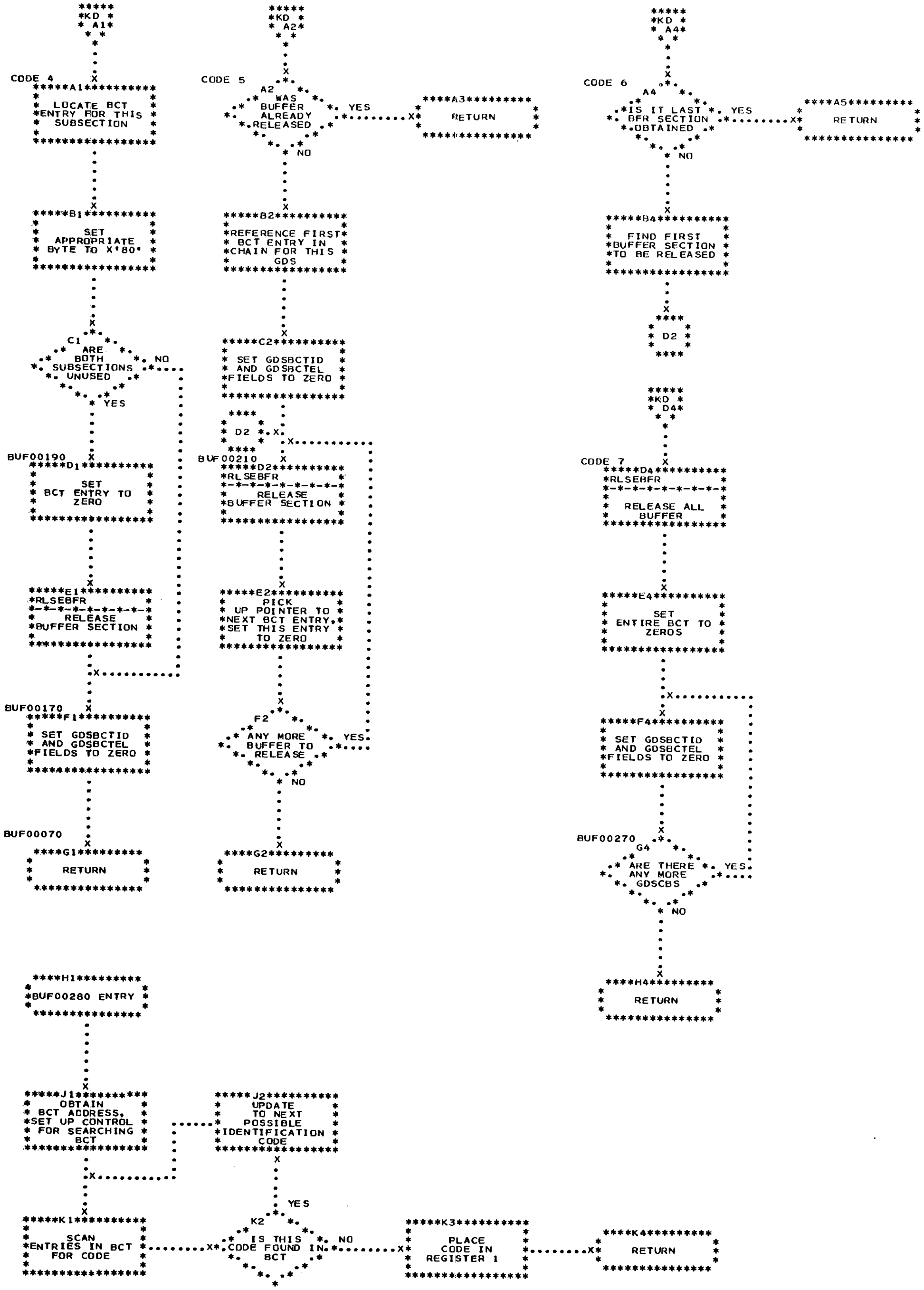
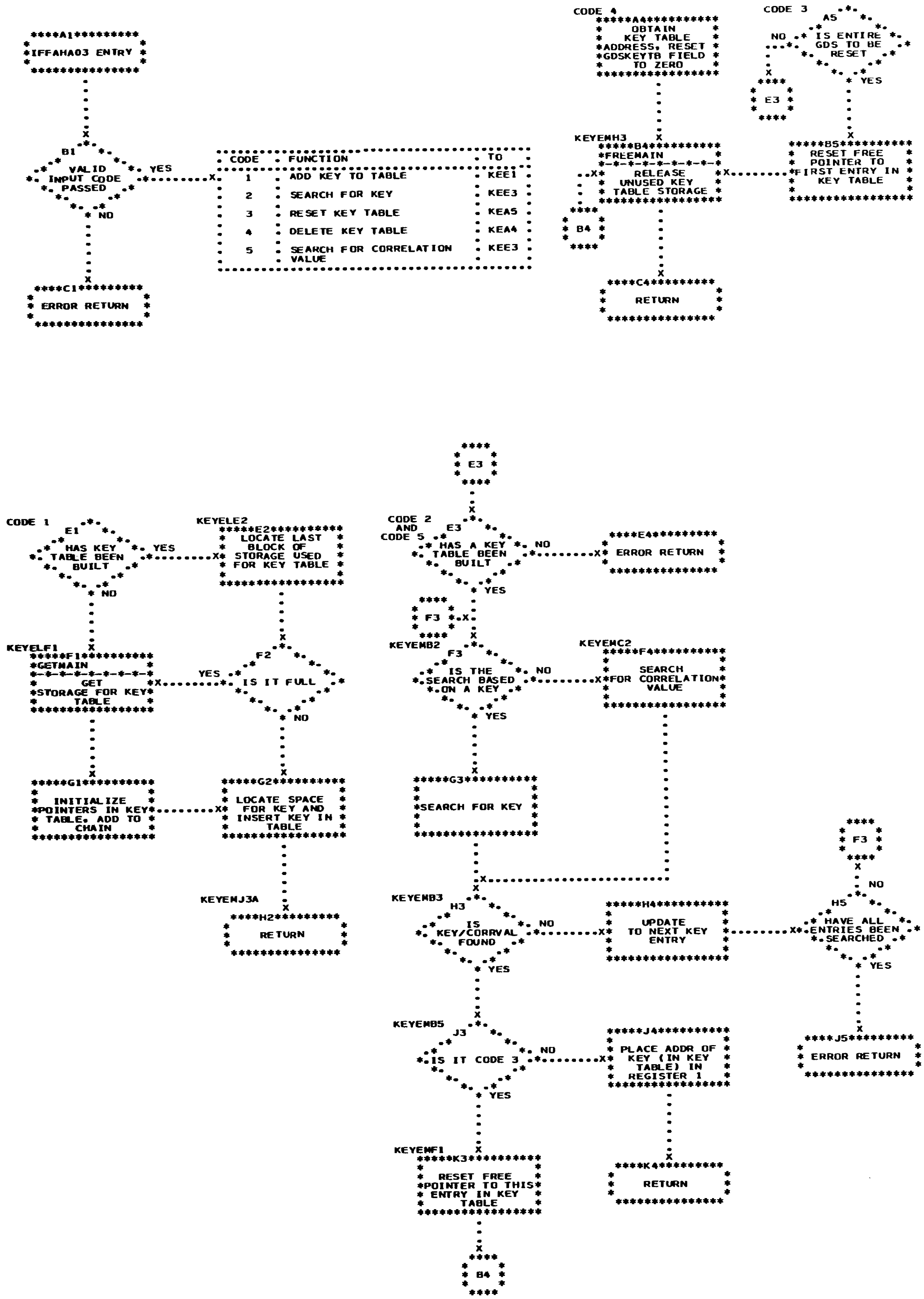

KD A1
CODE 4
A1
LOCATE BCT ENTRY FOR THIS SUBSECTION
B1
SET APPROPRIATE BYTE TO X'80'
C1
ARE BOTH SUBSECTIONS UNUSED
NO
YES
BUF00190
D1
SET BCT ENTRY TO ZERO
E1
RLSEBFR
RELEASE BUFFER SECTION
BUF00170
F1
SET GDSBCTID AND GDSBCTEL FIELDS TO ZERO
BUF00070
G1
RETURN
KD A2
CODE 5
A2
WAS BUFFER ALREADY RELEASED
YES
NO
A3
RETURN
B2
REFERENCE FIRST BCT ENTRY IN CHAIN FOR THIS GDS
C2
SET GDSBCTID AND GDSBCTEL FIELDS TO ZERO
D2
BUF00210
D2
RLSEBFR
RELEASE BUFFER SECTION
E2
PICK UP POINTER TO NEXT BCT ENTRY, SET THIS ENTRY TO ZERO
F2
ANY MORE BUFFER TO RELEASE
YES
NO
G2
RETURN
KD A4
CODE 6
A4
IS IT LAST BFR SECTION OBTAINED
YES
NO
A5
RETURN
B4
FIND FIRST BUFFER SECTION TO BE RELEASED
D2
KD D4
CODE 7
D4
RLSEBFR
RELEASE ALL BUFFER
E4
SET ENTIRE BCT TO ZEROS
F4
SET GDSBCTID AND GDSBCTEL FIELDS TO ZERO
BUF00270
G4
ARE THERE ANY MORE GDSCBS
YES
NO
H4
RETURN
H1
BUF00280 ENTRY
J1
OBTAIN BCT ADDRESS, SET UP CONTROL FOR SEARCHING BCT
J2
UPDATE TO NEXT POSSIBLE IDENTIFICATION CODE
YES
K1
SCAN ENTRIES IN BCT FOR CODE
K2
IS THIS CODE FOUND IN BCT
NO
K3
PLACE CODE IN REGISTER 1
K4
RETURN

Chart KE. Key Table Management Routine


A1
IFFAHA03 ENTRY
B1
VALID INPUT CODE PASSED
YES
NO
C1
ERROR RETURN
CODE 4
A4
OBTAIN KEY TABLE ADDRESS, RESET GDSKEYTB FIELD TO ZERO
CODE 3
A5
IS ENTIRE GDS TO BE RESET
NO
YES
E3
KEYEMH3
B4
FREEMAIN
RELEASE UNUSED KEY TABLE STORAGE
B4
B5
RESET FREE POINTER TO FIRST ENTRY IN KEY TABLE
C4
RETURN
CODE 1
E1
HAS KEY TABLE BEEN BUILT
YES
NO
KEYELE2
E2
LOCATE LAST BLOCK OF STORAGE USED FOR KEY TABLE
KEYELF1
F1
GETMAIN
GET STORAGE FOR KEY TABLE
F2
IS IT FULL
YES
NO
G1
INITIALIZE POINTERS IN KEY TABLE, ADD TO CHAIN
G2
LOCATE SPACE FOR KEY AND INSERT KEY IN TABLE
KEYEMJ3A
H2
RETURN
E3
CODE 2 AND CODE 5
E3
HAS A KEY TABLE BEEN BUILT
NO
YES
E4
ERROR RETURN
F3
KEYEMB2
F3
IS THE SEARCH BASED ON A KEY
NO
YES
KEYEMC2
F4
SEARCH FOR CORRELATION VALUE
G3
SEARCH FOR KEY
KEYEMB3
H3
IS KEY/CORRVAL FOUND
NO
YES
H4
UPDATE TO NEXT KEY ENTRY
F3
NO
H5
HAVE ALL ENTRIES BEEN SEARCHED
YES
J5
ERROR RETURN
KEYEMB5
J3
IS IT CODE 3
NO
YES
J4
PLACE ADDR OF KEY (IN KEY TABLE) IN REGISTER 1
K4
RETURN
KEYEMF1
K3
RESET FREE POINTER TO THIS ENTRY IN KEY TABLE
B4

CODE	FUNCTION	TO
1	ADD KEY TO TABLE	KEE1
2	SEARCH FOR KEY	KEE3
3	RESET KEY TABLE	KEA5
4	DELETE KEY TABLE	KEA4
5	SEARCH FOR CORRELATION VALUE	KEE3

Chart KF. Scaling Routine

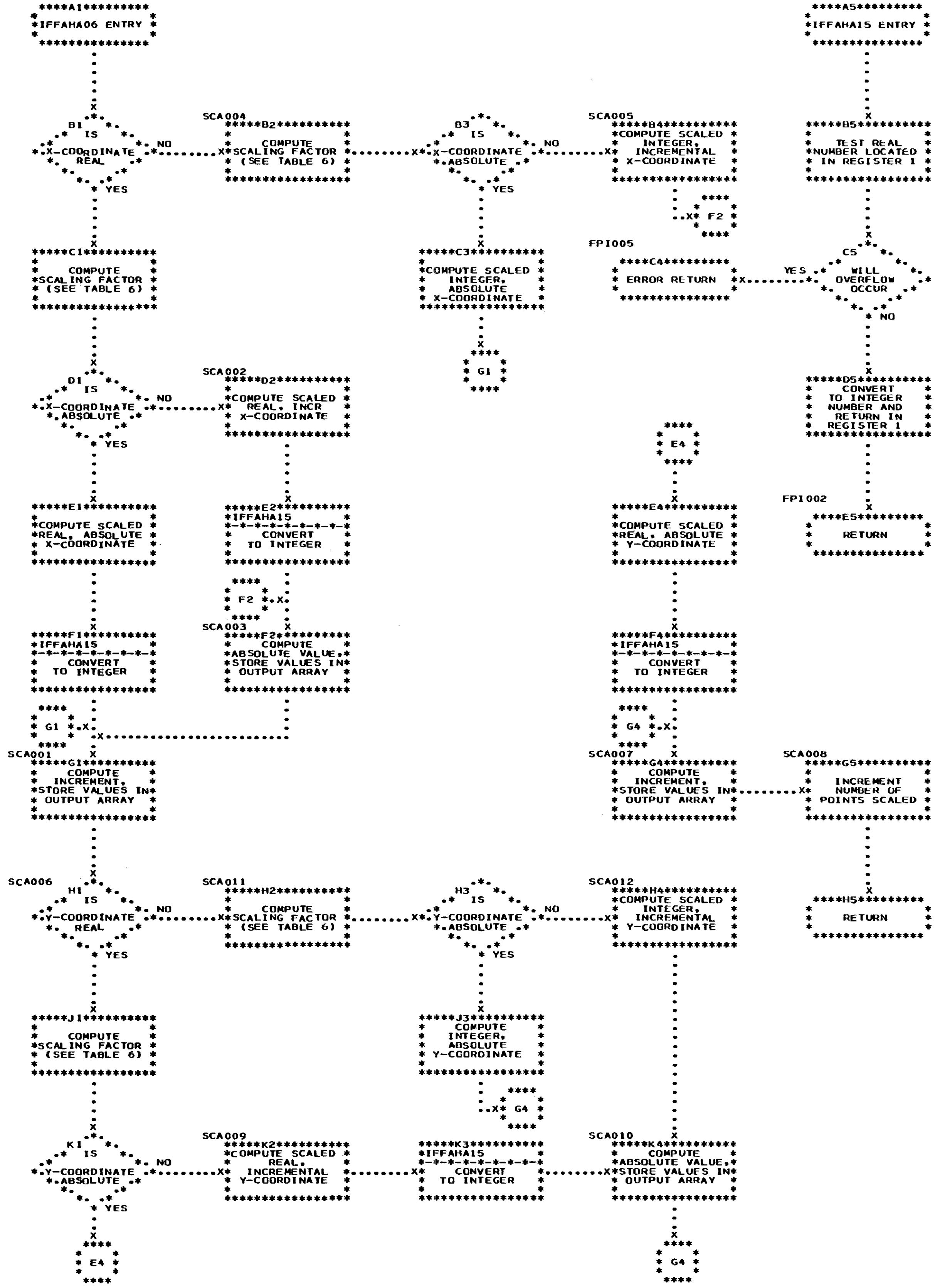

A1
IFFAHA06 ENTRY
B1
IS X-COORDINATE REAL
NO
YES
SCA004
B2
COMPUTE SCALING FACTOR (SEE TABLE 6)
B3
IS X-COORDINATE ABSOLUTE
NO
YES
SCA005
B4
COMPUTE SCALED INTEGER, INCREMENTAL X-COORDINATE
F2
A5
IFFAHA15 ENTRY
B5
TEST REAL NUMBER LOCATED IN REGISTER 1
C1
COMPUTE SCALING FACTOR (SEE TABLE 6)
C3
COMPUTE SCALED INTEGER, ABSOLUTE X-COORDINATE
G1
FPI005
C4
ERROR RETURN
C5
WILL OVERFLOW OCCUR
YES
NO
D1
IS X-COORDINATE ABSOLUTE
NO
YES
SCA002
D2
COMPUTE SCALED REAL, INCR X-COORDINATE
D5
CONVERT TO INTEGER NUMBER AND RETURN IN REGISTER 1
E4
E1
COMPUTE SCALED REAL, ABSOLUTE X-COORDINATE
E2
IFFAHA15
CONVERT TO INTEGER
E4
COMPUTE SCALED REAL, ABSOLUTE Y-COORDINATE
FPI002
E5
RETURN
F2
SCA003
F2
COMPUTE ABSOLUTE VALUE, STORE VALUES IN OUTPUT ARRAY
F1
IFFAHA15
CONVERT TO INTEGER
F4
IFFAHA15
CONVERT TO INTEGER
G1
G4
SCA001
G1
COMPUTE INCREMENT, STORE VALUES IN OUTPUT ARRAY
SCA007
G4
COMPUTE INCREMENT, STORE VALUES IN OUTPUT ARRAY
SCA008
G5
INCREMENT NUMBER OF POINTS SCALED
SCA006
H1
IS Y-COORDINATE REAL
NO
YES
SCA011
H2
COMPUTE SCALING FACTOR (SEE TABLE 6)
H3
IS Y-COORDINATE ABSOLUTE
NO
YES
SCA012
H4
COMPUTE SCALED INTEGER, INCREMENTAL Y-COORDINATE
H5
RETURN
J1
COMPUTE SCALING FACTOR (SEE TABLE 6)
J3
COMPUTE INTEGER, ABSOLUTE Y-COORDINATE
G4
K1
IS Y-COORDINATE ABSOLUTE
NO
YES
SCA009
K2
COMPUTE SCALED REAL, INCREMENTAL Y-COORDINATE
K3
IFFAHA15
CONVERT TO INTEGER
SCA010
K4
COMPUTE ABSOLUTE VALUE, STORE VALUES IN OUTPUT ARRAY
E4
G4

Chart KG. Scissoring Routine

NOTE -- THE RETURN CODE THAT SCISSORING PLACES IN REGISTER 15 INDICATES THE FOLLOWING TO THE CALLING PROGRAM --

0 = ALL OR PART OF THE INPUT DATA WILL BE DISPLAYED, AND IMAGE GENERATION IS TO BE CONTINUED.

4 = ALL OR PART OF THE INPUT DATA IS OUTSIDE SCISSORING LIMITS, AND IMAGE GENERATION IS TO BE DISCONTINUED.

8 = ALL INPUT DATA IS OUTSIDE SCISSORING LIMITS, BUT IMAGE GENERATION IS TO BE CONTINUED.

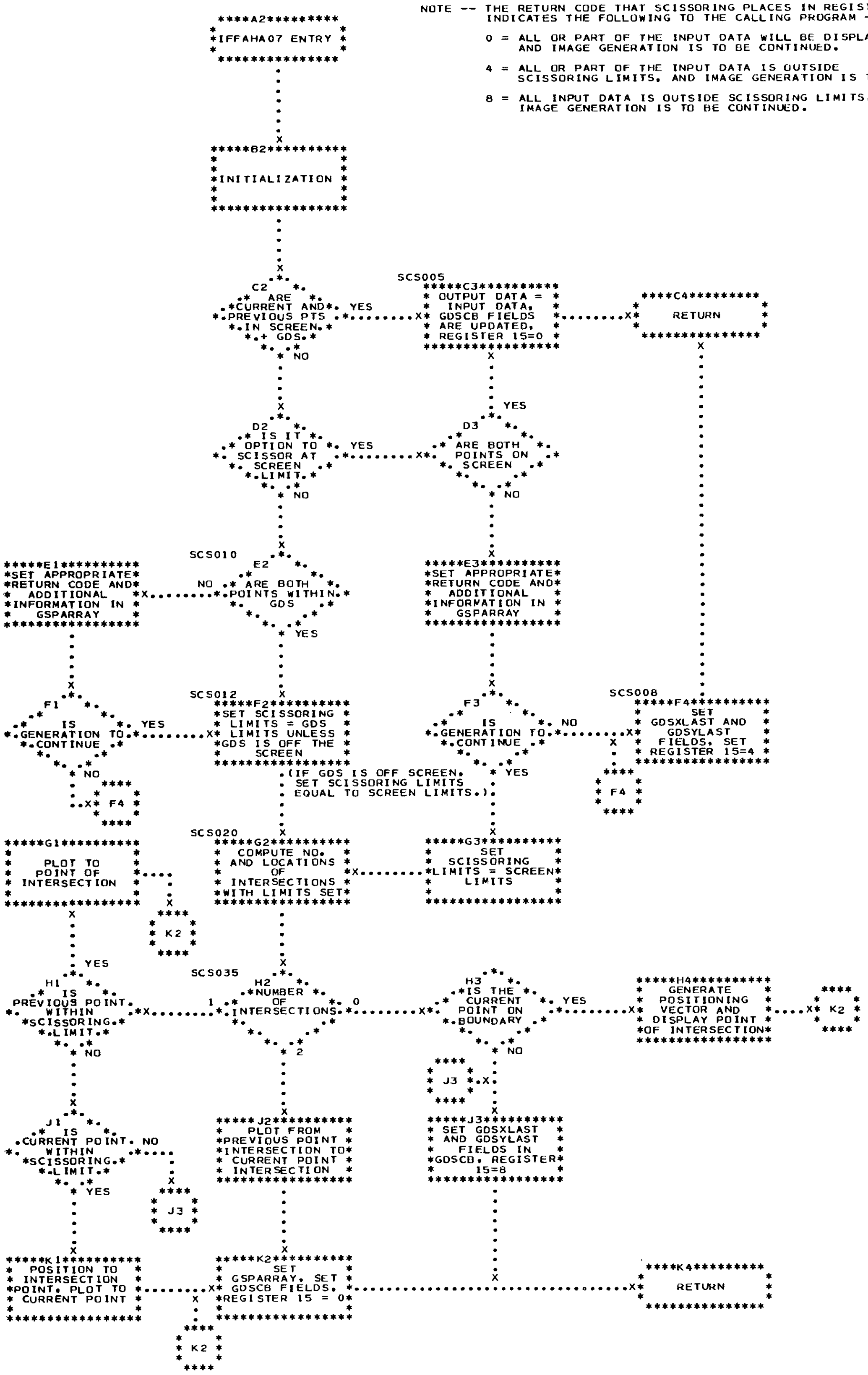
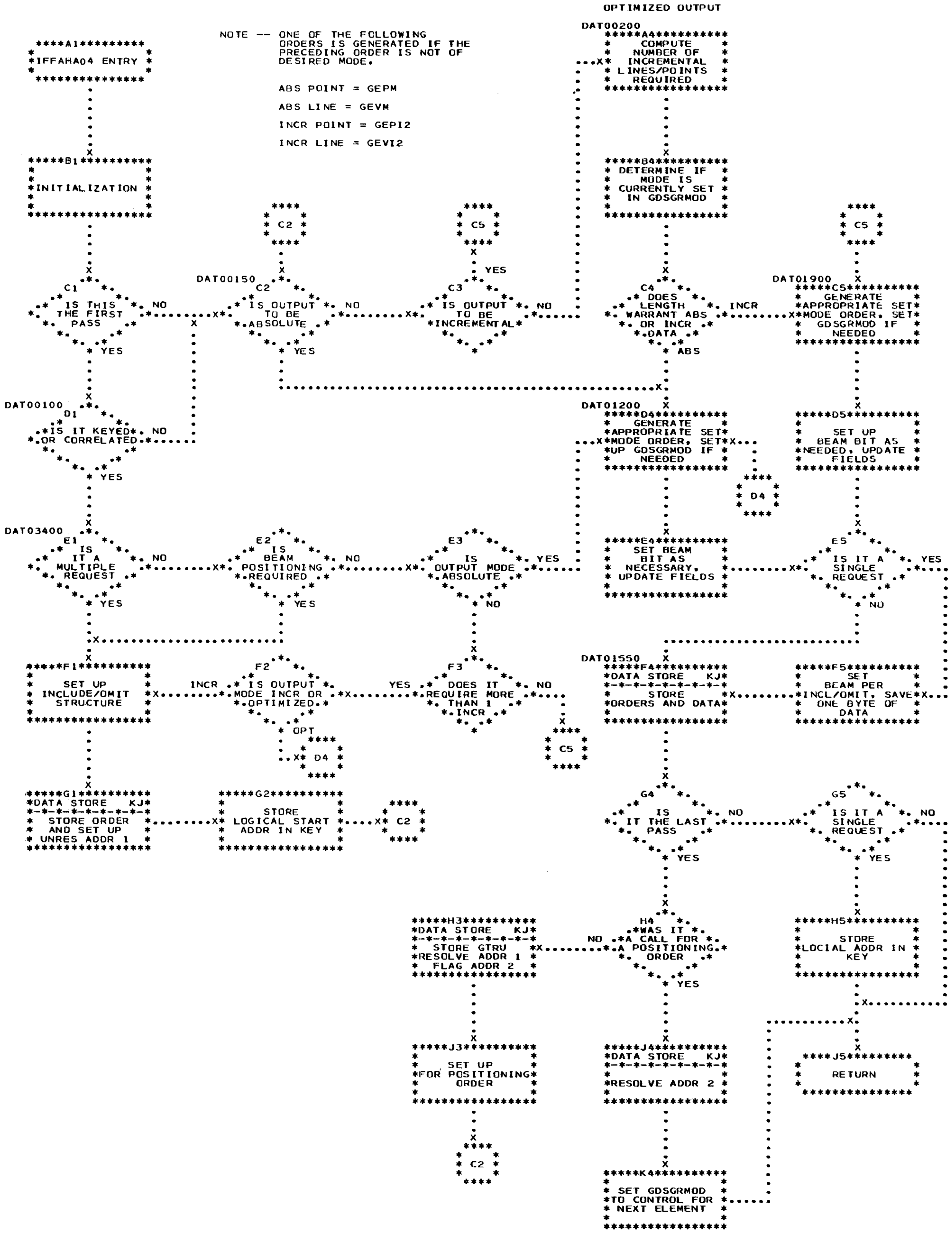

A2
IFFAHA07 ENTRY
B2
INITIALIZATION
C2
ARE CURRENT AND PREVIOUS PTS IN SCREEN + GDS
YES
NO
SCS005
C3
OUTPUT DATA = INPUT DATA, GDSCB FIELDS ARE UPDATED, REGISTER 15=0
C4
RETURN
D2
IS IT OPTION TO SCISSOR AT SCREEN LIMIT
YES
NO
D3
ARE BOTH POINTS ON SCREEN
YES
NO
SCS010
E2
ARE BOTH POINTS WITHIN GDS
NO
YES
E1
SET APPROPRIATE RETURN CODE AND ADDITIONAL INFORMATION IN GSPARRAY
E3
SET APPROPRIATE RETURN CODE AND ADDITIONAL INFORMATION IN GSPARRAY
F1
IS GENERATION TO CONTINUE
YES
NO
F4
SCS012
F2
SET SCISSORING LIMITS = GDS LIMITS UNLESS GDS IS OFF THE SCREEN
(IF GDS IS OFF SCREEN, SET SCISSORING LIMITS EQUAL TO SCREEN LIMITS.)
F3
IS GENERATION TO CONTINUE
NO
YES
SCS008
F4
SET GDSXLAST AND GDSYLAST FIELDS, SET REGISTER 15=4
F4
G1
PLOT TO POINT OF INTERSECTION
K2
SCS020
G2
COMPUTE NO. AND LOCATIONS OF INTERSECTIONS WITH LIMITS SET
G3
SET SCISSORING LIMITS = SCREEN LIMITS
YES
H1
IS PREVIOUS POINT WITHIN SCISSORING LIMIT
NO
SCS035
H2
NUMBER OF INTERSECTIONS
1
0
2
H3
IS THE CURRENT POINT ON BOUNDARY
YES
NO
J3
H4
GENERATE POSITIONING VECTOR AND DISPLAY POINT OF INTERSECTION
K2
J1
IS CURRENT POINT WITHIN SCISSORING LIMIT
NO
YES
J3
J2
PLOT FROM PREVIOUS POINT INTERSECTION TO CURRENT POINT INTERSECTION
J3
SET GDSXLAST AND GDSYLAST FIELDS IN GDSCB, REGISTER 15=8
K1
POSITION TO INTERSECTION POINT, PLOT TO CURRENT POINT
K2
K2
SET GSPARRAY, SET GDSCB FIELDS, REGISTER 15 = 0
K4
RETURN

Chart KH. Data Generator Routine



NOTE -- ONE OF THE FOLLOWING ORDERS IS GENERATED IF THE PRECEDING ORDER IS NOT OF DESIRED MODE.

ABS POINT = GEPM

ABS LINE = GEVM

INCR POINT = GEPI2

INCR LINE = GEVI2

OPTIMIZED OUTPUT
A1
IFFAHA04 ENTRY
B1
INITIALIZATION
C1
IS THIS THE FIRST PASS
NO
YES
DAT00100
D1
IS IT KEYED OR CORRELATED
NO
YES
DAT03400
E1
IS IT A MULTIPLE REQUEST
NO
YES
F1
SET UP INCLUDE/OMIT STRUCTURE
G1
DATA STORE KJ
STORE ORDER AND SET UP UNRES ADDR 1
C2
DAT00150
C2
IS OUTPUT TO BE ABSOLUTE
NO
YES
E2
IS BEAM POSITIONING REQUIRED
NO
YES
F2
IS OUTPUT MODE INCR OR OPTIMIZED
INCR
OPT
D4
G2
STORE LOGICAL START ADDR IN KEY
C2
C5
YES
C3
IS OUTPUT TO BE INCREMENTAL
NO
E3
IS OUTPUT MODE ABSOLUTE
YES
NO
F3
DOES IT REQUIRE MORE THAN 1 INCR
YES
NO
C5
H3
DATA STORE KJ
STORE GTRU RESOLVE ADDR 1 FLAG ADDR 2
J3
SET UP FOR POSITIONING ORDER
C2
DAT00200
A4
COMPUTE NUMBER OF INCREMENTAL LINES/POINTS REQUIRED
B4
DETERMINE IF MODE IS CURRENTLY SET IN GDSGRMOD
C4
DOES LENGTH WARRANT ABS OR INCR DATA
INCR
ABS
DAT01200
D4
GENERATE APPROPRIATE SET MODE ORDER, SET UP GDSGRMOD IF NEEDED
D4
E4
SET BEAM BIT AS NECESSARY, UPDATE FIELDS
DAT01550
F4
DATA STORE KJ
STORE ORDERS AND DATA
G4
IS IT THE LAST PASS
NO
YES
H4
WAS IT A CALL FOR A POSITIONING ORDER
NO
YES
J4
DATA STORE KJ
RESOLVE ADDR 2
K4
SET GDSGRMOD TO CONTROL FOR NEXT ELEMENT
C5
DAT01900
C5
GENERATE APPROPRIATE SET MODE ORDER, SET GDSGRMOD IF NEEDED
D5
SET UP BEAM BIT AS NEEDED, UPDATE FIELDS
E5
IS IT A SINGLE REQUEST
YES
NO
F5
SET BEAM PER INCL/OMIT, SAVE ONE BYTE OF DATA
G5
IS IT A SINGLE REQUEST
NO
YES
H5
STORE LOCIAL ADDR IN KEY
J5
RETURN

Chart KJ. Data Store Routine

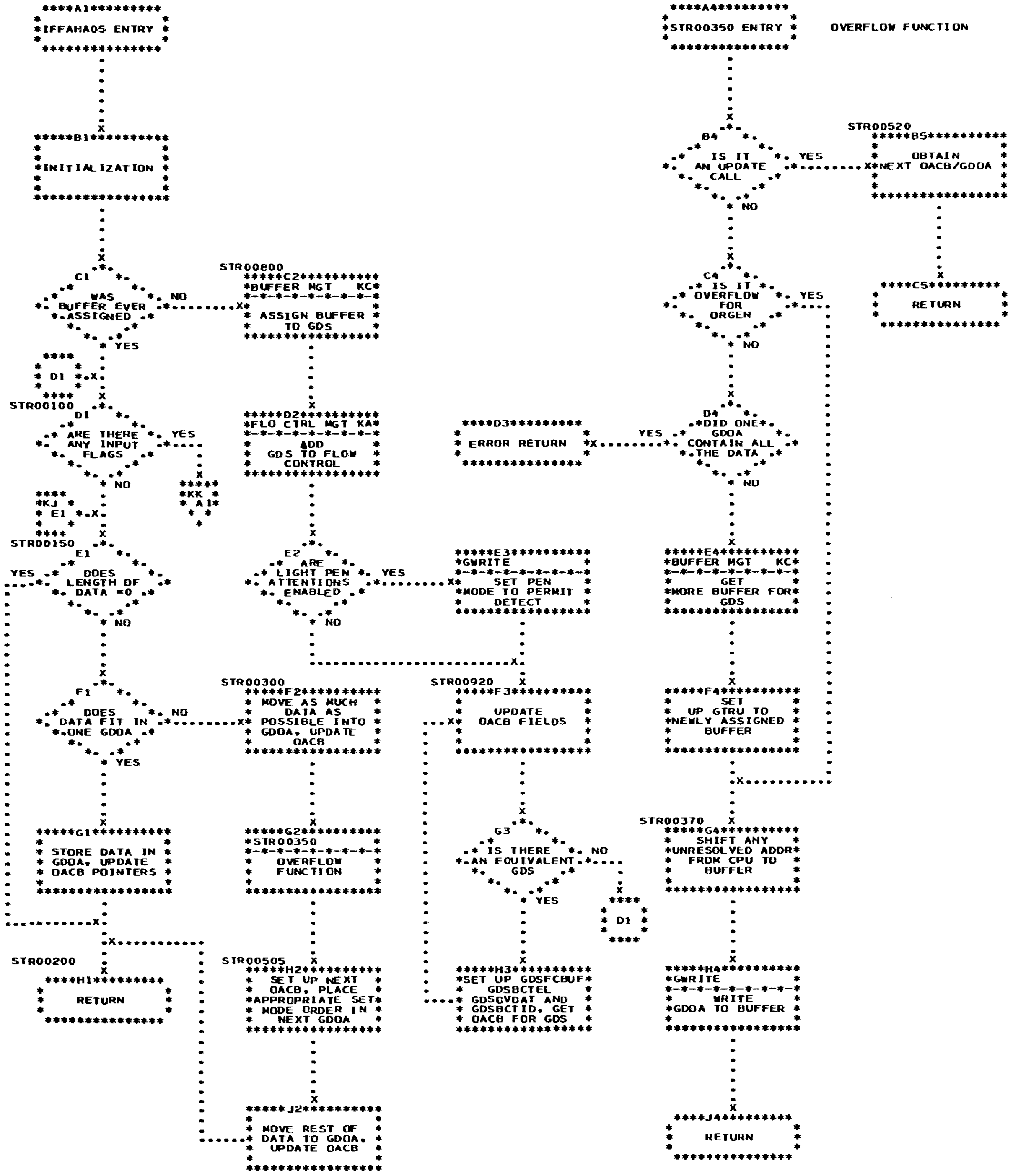
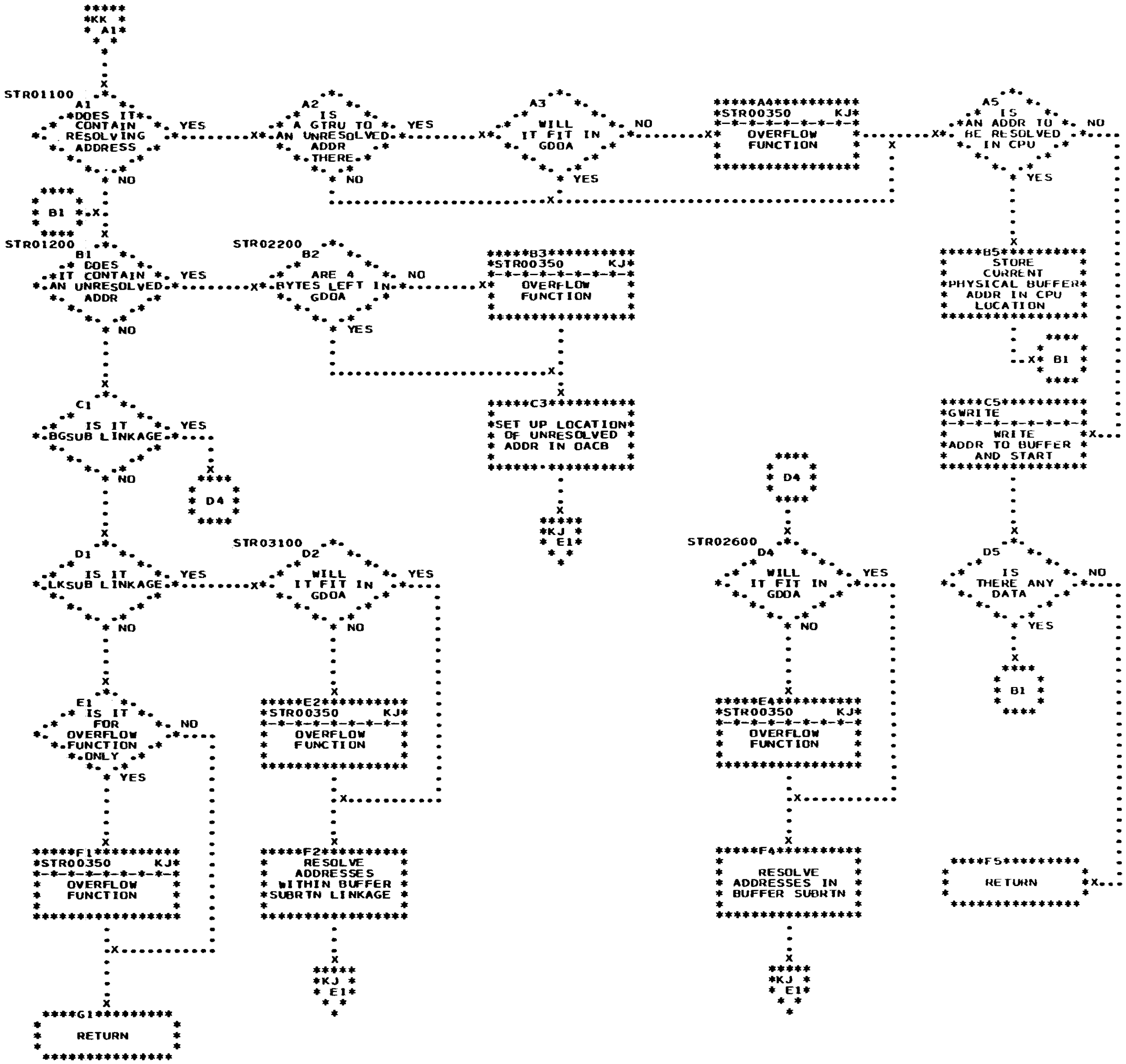


Chart KK. Data Store Routine (continued)



This appendix contains detailed formats of the control blocks that are created by GSP routines. Refer to "Control Blocks and Tables Used by GSP" for general information describing their use.

Table 10 contains a listing of each control block and table created by GSP and gives the name of the routines that create and terminate the control block or table.

Table 10. Creation and Termination of GSP Control Blocks

Control Block or Table	Created By	Terminated By
GSPCB	INGSP	TMGSP
GTMCB	INDEV	TMDEV
GDSCB	INGDS	TMGDS
GACB	INDEV, LOCPN	TMDEV, LOCPN
COMAREA	INDEV, LOCPN	INDEV, LOCPN
DCB	INDEV	TMDEV
DECB*	Flow Control Management	Flow Control Management
GDOA	INGDS, Update	TMGDS, Update
OACB	INGDS, Update	TMGDS, Update
ATLCB	CRATL	ENATL
ADEQ	CRATL	ENATL
BCT	Buffer Mgt	Buffer Mgt
FCT	Flow Control Management	Flow Control Management
Key Table	Key Tbl Mgt	Key Tbl Mgt
System GDSCB	INDEV	TMDEV

*Numerous subroutines create their own DECB for input/output operations. Such a DECB is constructed in the work/save area used by all subroutines.

The control blocks described in this appendix in their order of description are:

- Graphic subroutine package control block (GSPCB)
- Graphic terminal control block (GTMCB)
- Graphic data set control block (GDSCB)
- Graphic attention control block (GACB) appendage only
- Output area control block (OACB)
- Attention level control block (ATLCB)
- Attention data entry queue (ADEQ)

These blocks are illustrated showing the layout of the fields within them. Detailed descriptions of the contents of these fields follow each block. Fields that contain addresses which point to other control blocks, tables, buffer locations, routines, and main storage areas are indicated by an upright arrow (↑) in the field.

The buffer control table (BCT), flow control table (FCT), and key table are described under their appropriate headings in the section "Program Organization". The communication area (COMAREA), data control block (DCB), data event control block (DECB), and graphic data output area (GDOA) are described in the publication IBM System/360 Operating System: Graphic Programming Services for IBM 2250 Display Unit, Form C27-6909.

Graphic Subroutine Package Control Block (GSPCB)

<---------------4 Bytes--------------->

+ 0	GSPGTMCB ↑			
+ 4	GSPDCTR2 ↑			
+ 8	GSPSYSID	GSPCBID	GSPBUSYS	Reserved
+12	Reserved	GSPDUMP	GSPABEND	Reserved
+16	GSPAWORK ↑			
+20	GSPSTRTB ↑			
+24	GSPARRAY			
+28	Second Word of GSPARRAY			
+32	Third Word of GSPARRAY			
+36	Fourth Word of GSPARRAY			
+40	Fifth Word of GSPARRAY			
+44	GSPNULLV ↑			
+48	GSPASTAT ↑			

GSPGTMCB

Address of the most recently created GTMCB in the chain of GTMCBs attached to the GSPCB.

GSPDCTR2

Address of Director Part 2.

GSPSYSID

One-byte field containing X'F2' that identifies this as a GSP control block.

GSPCBID

One-byte field in which X'01' identifies this as the GSPCB.

GSPBUSYS

One-byte field that indicates status of GSP routines:

X'00' - not busy, or
X'FF' - busy.

GSPDUMP

One-byte field that indicates whether the null variable is negative, meaning a dump is desired if the ABEND macro instruction is issued; or positive, meaning there should be no dump. The bit setting is:

X'00' - no dump, or
X'FF' - dump.

GSPABEND

One-byte field that contains the absolute value of the null variable which determines whether or not the program is to be abnormally terminated when the return code is tested.

GSPAWORK

Address of a work/save area that follows the GSPCB.

GSPSTRTB

Address of the first stroke table that was loaded.

GSPARRAY

The first word of a five-word field which is the return code array. In this first word, Byte 0 contains the return code that is being passed by a GSP subroutine. Byte 1 contains the last return code tested by ITRC, and it is set each time a return code is tested by the ITRC subroutine. Bytes 2 and 3 are reserved. Bit settings for Bytes 0 and 1 are as follows:

Return Codes and Meanings	Bit Settings 0 1 2 3 4 5 6 7
0 - Satisfactory	0 0 0 0 0 0 0 0
1 - Scissoring used	0 1 0 0 0 0 0 0
2 - Scaling error	0 0 1 0 0 0 0 0
3 - Storage exceeded	0 0 0 1 0 0 0 0
4 - Parameter error	0 0 0 0 1 0 0 0
5 - Input/output error	0 0 0 0 0 1 0 0

Second Word

When return code 1 is issued, this word contains additional information consisting of the value of the count field associated with the data that was last scissored. A value of zero indicates no data was generated. A value of 1 is the count for the STPOS or the MVPOS subroutine.

Third Word

When return code 2 is issued, this word contains additional information consisting of the value of the count field associated with the data that was last scaled. A value of zero indicates no data was generated. A value of 1 is the count for the STPOS or the MVPOS subroutine.

Fourth Word

When return code 3 is issued, this word contains additional information which is one of the following values:

1 - update attempt failed.
2 - 128-byte GDS exceeded, or equivalent GDS exceeded.
3 - buffer for data not available.
4 - main storage not available.

Fifth Word

When return code 4 is issued, this word contains additional information which is one of the following values:

0 - cannot determine that any one parameter is in error.

1-n - parameter number, counting from left to right, that was in error. In this case, "n" represents the maximum number of parameters specified in the call to the subroutine.

GSPNULLV

Address of the null variable which is recognized in the absence of a parameter. The null variable itself is accessed to determine whether or not the program should be abnormally terminated, and to determine whether or not a dump is requested.

GSPASTAT
Address of the status table.

Graphic Terminal Control Block (GTMCB)

```
      <------------- 4 Bytes--------------->
      ┌─────────────────────────────────────┐
+ 0   │              GTMNXGTM             ↑ │
      ├─────────────────────────────────────┤
+ 4   │              GTMGSPCB             ↑ │
      ├─────────┬─────────┬─────────────────┤
+ 8   │GTMVALID │GTMTERID │    Reserved     │
      ├─────────┴─────────┴─────────────────┤
+12   │              GTMGTMCB             ↑ │
      ├─────────────────────────────────────┤
+16   │              GTMGRDCB             ↑ │
      ├─────────────────────────────────────┤
+20   │              GTMGDSCB             ↑ │
      ├─────────────────────────────────────┤
+24   │              GTMLATBL             ↑ │
      ├─────────────────────────────────────┤
+28   │              GTMGACBA             ↑ │
      ├─────────────────────────────────────┤
+32   │              GTMSYGDS             ↑ │
      ├─────────────────────────────────────┤
+36   │              GTMBCTBL             ↑ │
      ├─────────────────────────────────────┤
+40   │              GTMFCTBL             ↑ │
      ├─────────────────────────────────────┤
+44   │              GTMPFKSS               │
      ├───────────────────┬─────────────────┤
+48   │     GTMFEATS      │    GTMREPOS     │
      ├───────────────────┴─────────────────┤
+52   │              GTMDECB1             ↑ │
      ├───────────────────┬─────────────────┤
+56   │     GTMGDOAL      │    GTMDVADR     │
      ├───────────────────┼─────────────────┤
+60   │     Reserved      │    GTMCRLOC   ↑ │
      ├─────────┬─────────┼────────┬────────┤
+64   │GTMBCTSM │GTMBCTLG │GTMFLAGS│GTMLIGHT│
      └─────────┴─────────┴────────┴────────┘
```

GTMNXGTM
Address of the next GTMCB in the chain.

GTMGSPCB
Address of the GSPCB.

GTMVALID
One-byte field containing X'F2' that identifies this as a GSP control block.

GTMTERID
One-byte field in which X'02' identifies this as a GTMCB.

GTMGTMCB
Address of this GTMCB used for determining that it is the control block specified by the user.

GTMGRDCB
Address of the DCB created for this 2250.

GTMGDSCB
Address of the most recently created GDSCB in the chain of GDSCBs that is associated with this 2250.

GTMLATBL
Address of the active attention level, which is represented by the most recently created ATLCB.

GTMGACBA
Address of the GSP GACB.

GTMSYGDS
Address of the system GDSCB. This GDSCB represents the GDS used for the graphic orders and data generated by the light pen subroutines for screen scanning or displaying the tracking symbol.

GTMBCTBL
Address of the buffer control table for this 2250.

GTMFCTBL
Address of the flow control table for this 2250.

GTMPFKSS
This field contains the light mask which designates the programmed function indicator lights that are to be lit. Bits 0 through 31 correspond to the programmed function keys 0 through 31. This field is used in conjunction with the GTMLIGHT field.

GTMFEATS
This half-word field is divided into one-byte fields which define the characteristics of this 2250, as follows:

Setting	Meaning
Byte 0	
X'01'	IBM 2250, Model 1
X'03'	IBM 2250, Model 3
Byte 1	
X'04'	Graphic design feature
X'08'	Character generator
X'10'	Programmed function keyboard
X'20'	Light pen
X'40'	Alphameric keyboard
X'80'	Absolute vectors

GTMREPOS
A half-word field that contains the buffer address of the coordinates used for repositioning the 2250 beam for light pen tracking.

GTMDECB1
Address of the data event control block (DECB) used by GSP for

input/output overlap. This DECB is created by Flow Control Management.

GTMGDOAL
A half-word field that contains the length of each GDOA for the GDSs attached to this 2250 as specified by a call to the INDEV subroutine.

GTMDVADR
The physical address of this 2250 as obtained from the UCB.

GTMCRLOC
Address of the physical buffer location that will contain the cursor.

GTMBCTSM
Index used to obtain an identification number for Buffer Management when a 128-byte buffer subsection is assigned.

GTMBCTLG
Index used to obtain an identification number for Buffer Management when a 256-byte buffer section, or multiple thereof, is assigned.

GTMFLAGS
One-byte field containing miscellaneous flags, as follows:
  X'01' - cursor is in the 2250.
  X'08' - pen tracking subroutine is in the buffer.

GTMLIGHT
One-byte field that specifies the programmed function keyboard lighting option selected:
  X'02' - light no programmed function indicators (default option).
  X'03' - light enabled keys.
  X'04' - light keys as specified in the light mask (GTMPFKSS).

<u>Graphic Data Set Control Block (GDSCB)</u>

	<------------4 Bytes--------------->			
+ 0	GDSNXGDS ↑			
+ 4	GDSGTMCB ↑			
+ 8	GDSVALID	GDSGDSID	GDSFLAGS	GDSFLGS2
+ 12	GDSAOACB ↑			
+ 16	GDSGDSCB ↑			
+ 20	GDSKEYTB ↑			
+ 24	GDSXLAST			
+ 28	GDSYLAST			
+ 32	GDSXCURR		GDSYCURR	
+ 36	GDSUVLLL			
+ 40	GDSVVLLL			
+ 44	GDSUVLUR			
+ 48	GDSVVLUR			
+ 52	GDSXVILL			
+ 56	GDSYVILL			
+ 60	GDSXVIUR			
+ 64	GDSYVIUR			
+ 68	GDSXVRLL			
+ 72	GDSYVRLL			
+ 76	GDSXVRUR			
+ 80	GDSYVRUR			
+ 84	GDSCHARS	GDSSCISS	GDSCURMD	GDSDATMD
+ 88	GDSOVDAT			
+ 92	GDSFCBUF ↑		GDSBCTEL ↑	
+ 96	GDSBCTID	GDSGRMOD	GDSGDOAL	
+100	GDSSHDBK ↑			
+104	GDSSHDFD ↑			
+108	GDSREPPB ↑		GDSKEYSV	
+112	GDSREPDT			

GDSNXGDS
Address of the next GDSCB in the chain of GDSCBs attached to the GTMCB.

GDSGTMCB
Address of the GTMCB associated with this GDS.

GDSVALID
One-byte field containing X'F2' that identifies this as a GSP control block.

GDSGDSID
One-byte field in which X'03' identifies this as a GDSCB.

GDSFLAGS
One-byte field that contains assorted flags, as follows:

Bit	0 or Off	1 or On
0	None	Equivalent GDS is in the buffer
1	Correlation value passed (BGSEQ/BGSUB)	No correlation value passed
2	Cursor absent	Cursor present
3	Level 1 GSP	Level 2 GSP
4	Not equivalent	Equivalent GDS
5	No sequence	Sequence begun
6	No subroutine	Subroutine begun
7	Disable light pen attentions	Permit light pen attentions

GDSFLGS2
One-byte field that contains flags, as follows:
X'01' - SDATL subroutine has been called.

GDSAOACB
Address of the active output area control block.

GDSGDSCB
Address of this GDSCB used for determining that it is the control block specified by the user.

GDSKEYTB
Address of the key table for this GDS, if present, or zero.

GDSXLAST
Value in raster units of the x-coordinate for the last call to an image generation subroutine, before scissoring.

GDSYLAST
Value in raster units of the y-coordinate for the last call to an image generation subroutine, before scissoring.

GDSXCURR
Current x-coordinate position of the 2250 beam in raster units, after scissoring.

GDSYCURR
Current y-coordinate position of the 2250 beam in raster units, after scissoring.

GDSUVLLL
Value in user's units of lower-left x-coordinate of the GDS. This value may be real or integer.

GDSVVLLL
Value in user's units of lower-left y-coordinate of the GDS. This value may be real or integer.

GDSUVLUR
Value in user's units of upper-right x-coordinate of the GDS. This value may be real or integer.

GDSVVLUR
Value in user's units of upper-right y-coordinate of the GDS. This value may be real or integer.

GDSXVILL
Integer value in raster units of lower-left x-coordinate of the GDS.

GDSYVLLL
Integer value in raster units of lower-left y-coordinate of the GDS.

GDSXVIUR
Integer value in raster units of upper-right x-coordinate of the GDS.

GDSYVIUR
Integer value in raster units of the upper-right y-coordinate of the GDS.

GDSXVRLL
Real value in raster units of the lower-left x-coordinate of the GDS.

GDSYVRLL
Real value in raster units of the lower-left y-coordinate of the GDS.

GDSXVRUR
Real value in raster units of the upper-right x-coordinate of the GDS.

GDSYVRUR
Real value in raster units of the upper-right y-coordinate of the GDS.

GDSCHARS
One-byte field that contains the character option flags, as follows:

X'08' - basic, protected
X'04' - large, protected
X'02' - basic, unprotected
X'01' - large, unprotected

GDSSCISS
One-byte field that contains the scissoring option flags, as follows:
X'10' - no scissoring
X'08' - scissor at screen boundaries
X'04' - scissor at GDS boundaries
X'02' - continue generation
X'01' - discontinue generation

GDSCURMD
One-byte field that contains the output data mode option, as follows:
X'04' - optimized
X'02' - absolute
X'01' - incremental

GDSDATMD
One-byte field that contains the input data mode option, as follows:

Code	Coordinate	Real	Int	Abs	Incr
X'80'	X	Yes		Yes	
X'40'	X	Yes			Yes
X'20'	X		Yes	Yes	
X'10'	X		Yes		Yes
X'08'	Y	Yes		Yes	
X'04'	Y	Yes			Yes
X'02'	Y		Yes	Yes	
X'01'	Y		Yes		Yes

The default option is set by using X'88' which indicates real, absolute input data will be supplied for both x- and y-coordinates. This applies in cases where the SDATM subroutine is not called.

GDSOVLAT
Contains a GTRU order that transfers control to the flow control structure in the buffer when the GDS is executed.

GDSFCBUF
A half-word field containing the buffer address of the flow control entry for this GDS.

GDSBCTEL
A half-word field containing the BCT entry number for the first buffer section assigned to this GDS.

GDSBCTID
One-byte field containing the identification code of this GDS that is entered in the buffer control table entry.

GDSGRMOD
One-byte field that indicates the current mode of the 2250 associated with this GDS, as follows:
X'80' - character basic, protected
X'40' - character large, protected
X'20' - character basic, unprotected
X'10' - character large, unprotected
X'08' - line absolute
X'04' - line incremental
X'02' - point absolute
X'01' - point incremental
X'00' - control

GDSGDOAL
A half-word field that contains the length of every GDOA associated with this GDS as specified by a call to the INGDS subroutine, or by the GTMGDOAL field in the GTMCB.

GDSSHDBK
Address of the equivalent GDS that was created just prior to this one.

GDSSHDFD
Address of the equivalent GDS that was created immediately after this one.

GDSREPPB
A half-word field containing the physical buffer address of the replacement data stored in the GDSREPDT field.

GDSKEYSV
A half-word field containing the logical buffer address which is in the key created by the call to BGSEQ or BGSUB. It is used to locate the key in the key table when ENSEQ or ENSUB is called.

GDSREPDT
Four bytes of orders and data from the GDOA associated with this GDS that will be used to replace the GTRU order to the flow control structure when the next call to the EXEC subroutine is made.

Graphic Attention Control Block (GACB)

Three GACBs, one of which is reserved, are created by INDEV. The format of each of these GACBs is the same as created by the Graphics Access Method (GAM), except that each GACB has eight extra bytes appended to the beginning of it. The format of the GAM GACB is depicted in the publication IBM System/360 Operating System: Graphics Access Method, Program Logic Manual, Form Y27-7113. The format of the eight extra bytes in each GACB for GSP and the priority of each GACB for GSP are shown below.

GSP GACB: Priority = 0

<----------------4 Bytes---------------->
Address of reserved GACB ↑
Address of associated GTMCB ↑

RESERVED GACB: Priority = 1

<---------------- Bytes---------------->
Address of Panic GACB ↑
Address of associated GTMCB ↑

PANIC GACB: Priority = 127

<----------------4 Bytes---------------->
Zero
Address of associated GTMCB ↑

<u>Output Area Control Block (OACB)</u>

	<------------- Bytes--------------->
+ 0	OACB2PT ↑
+ 4	GDOA1PT ↑
+ 8	STATUS
+12	CRSA ↑
+16	OLP ↑
+20	BLP ↑
+24	BUFSTART ↑
+28	BUFLEN
+32	CPU1 ↑
+36	BUF1 ↑
+40	CPU2 ↑
+44	BUF2 ↑
+48	LOGSTART
+52	LOGCURR
+56	LENLEFT

OACB2PT
    Address of the next OACB for this GDS.

GDOA1PT
    Address of the GDOA associated with this OACB.

STATUS
    Flag field used to show the status of the OACB. Flags are as follows:
    X'80' - This GDS has not been added to the flow control structure, and no buffer has been assigned yet.
    X'40' - Four bytes of data located at CRSA must be saved in the GDSCB, along with the physical buffer address.

CRSA
    (Current Routine Start Address) is the address of the location in the GDOA that is the start of the set of graphic orders to be stored by the next input/output operation.

OLP
    (Order Load Point) is the address of the next available location in the GDOA where the next graphic order of the current routine will be stored.

BLP
    (Buffer Load Point) is the address of the buffer location where the first byte of data from the output area will be written by the next input/output operation.

BUFSTART
    Is the buffer start address which is the address of the buffer location where the first byte of data from the GDOA is written.

BUFLEN
    Length of the buffer that is currently available for writing out graphic orders and data. Initially this is set to the GDOA length. It is adjusted whenever part of the GDOA is written to the buffer.

CPU1
    Location in the GDOA of an unresolved address in connection with a sequence or buffer subroutine. This unresolved address is resolved by Data Store.

BUF1
    Location in the buffer, because of an overflow, of an unresolved address in connection with a sequence or buffer subroutine. Either CPU1 <u>or</u> BUF1 is used by Data Store for resolving addresses, but both of these fields are never used at a given time.

CPU2
    Location in the GDOA of a second unresolved address that is resolved by Data Store.

BUF2
    Location in the buffer, because of an overflow, of a second unresolved address. Either CPU2 <u>or</u> BUF2 is used

by Data Store for resolving addresses, but both of these fields are never used at a given time.

LOGSTART
Logical buffer address of the buffer load point (BLP).

LOGCURR
Logical buffer address of the order load point (OLP).

LENLEFT
Unused length (in bytes) remaining in the GDOA after graphic orders have been stored, which does not include the last four bytes (reserved for a GTRU to the next buffer section).

Attention Level Control Block (ATLCB)

	<-------------4 Bytes------------->			
+0	ATTNFWPT ↑			
+4	ATTNBKPT ↑			
+8	ATTNLVID			
+12	Reserved			
+16	ATTNAVAL ↑			
+20	Reserved			
+24	ATTNMLEN ↑			
+28	ATTNFLGA	Reserved	ATTNBPID	ATTNLITE
+36	ATTNPFLM			
+40	ATTNPFEN			
+44	ATTNANEN			
+48	Reserved			
+52	Reserved			
+56	Reserved			
+60	Reserved			
+64	Reserved			
+68	Reserved			
+72	Reserved			
+76	Reserved			

ATTNFWPT
Address of next lower ATLCB in the chain. If there is no lower ATLCB, ATTNFWPT = 0.

ATTNBKPT
Address of next higher ATLCB in the chain. If there is no higher ATLCB, ATTNBKPT contains the address of the GTMCB.

ATTNLVID
A value of X'F2040000' to identify this control block as a valid ATLCB.

ATTNAVAL
Address of the first available ADEQ where the information from the next attention will be placed.

ATTNMLEN
Address of the ADEQ where information from the first attention that occurred has been placed.

ATTNFLGA
One-byte field which contains the following options for the designated attention sources:

Source and Option	Bit Settings 0 1 2 3 4 5 6 7
EOS 1. Return x-, y-coordinates	1 0 0 0 0 0 0 0
EOS 2. Restart display immediately	0 1 0 0 0 0 0 0
LP 1. Same as EOS 1	0 0 1 0 0 0 0 0
LP 2. Same as EOS 2	0 0 0 0 1 0 0 0
LP 3. Return character information	0 0 0 1 0 0 0 0
All: Do not dequeue data on inactive level	0 0 0 0 0 1 0 0
Note: If data is always to be dequeued after a call to RQATN, bit 5 remains zero.	

ATTNBPID
One-byte field which indicates whether the ATTNBKPT field points to another ATLCB (ATTNBPID = X'00') or the GTMCB (ATTNBPID = X'01').

ATTNLITE
One-byte field which contains programmed function indicator lights options for this attention level as follows:

Bit Setting	Meaning
X'01'	Use default option in GTMCB
X'02'	Turn off all indicator lights
X'03'	Light enabled keys
X'04'	Use ATTNPFLM light mask

ATTNPFLM
Determines which programmed function indicator lights are to be lit if ATTNLITE = X'04'. Bit positions 0-31 of this field correspond to programmed function keys 0-31, respectively. When the bit is on, its corresponding indicator light is lit.

ATTNPFEN
Specifies which programmed function keys are enabled for this ATLCB. Bit positions 0-31 correspond to programmed function keys 0-31, respectively. When the bit is on, its corresponding key is enabled.

ATTNANEN
Specifies which attention sources other than programmed function keys are enabled for this ATLCB. The following bits are set in byte 0 to enable the specified attention source:

Attention Source	Bit Settings 0 1 2 3 4 5 6 7
END key	0 0 0 0 0 0 0 1
CANCEL key	0 0 0 0 1 0 0 0
Light pen	0 0 0 0 0 0 1 0
End-order-sequence	0 0 0 0 0 1 0 0

Attention Data Entry Queue (ADEQ)

	<-------------4 Bytes--------------->
+0	ADEQFPTR ↑
+4	Reserved
+8	ADEQWRD1
+12	ADEQWRD2
+16	ADEQWRD3
+20	Reserved

ADEQFPTR
Address of next ADEQ in the chain.

ADEQWRD1, ADEQWRD2, ADEQWRD3
Contain information from the first three words of the COMAREA associated with the GSP GACB.

This appendix contains the names of all the modules in GSP, listed in alphameric order, together with their associated routine name, type, and method of entry, which is indicated in the status table. These module names also appear in the table of contents.

Module Name	Associated Routine	Type of Routine	Method of Entry
IFFAAA01	INGSP Part 2	Initiation	LINK
IFFAAA02	TMGSP Part 2	Termination	LINK
IFFAAA03	INDEV	Initiation	LINK
IFFAAA04	TMDEV	Termination	LINK
IFFAAA05	INGDS	Initiation	LINK
IFFAAA06	TMGDS	Termination	LINK
IFFACA00	CRATL	Attention Related	LINK
IFFACA01	ENATL	Attention Related	LINK
IFFACA02	ENATN	Attention Related	LINK
IFFACA03	DSATN	Attention Related	LINK
IFFACA04	MPATL	Attention Related	LINK
IFFACA05	MLPEO	Attention Related	LINK
IFFACA06	SLPAT	Attention Related	LINK
IFFACA07	MLITS	Attention Related	LINK
IFFACA08	RQATN	Attention Related	LOAD and CALL
IFFACA13	SALRM	Attention Related	LINK
IFFADA01	GSPRD	Keyboard Input and Buffer Data Analysis	LOAD and CALL
IFFADA02	RCURS	Keyboard Input and Buffer Data Analysis	LOAD and CALL
IFFADA03	ICURS	Keyboard Input and Buffer Data Analysis	LOAD and CALL
IFFAEA01	SDATM	Option Definition	LINK
IFFAEA02	SGRAM	Option Definition	LINK
IFFAEA03	SDATL	Option Definition	LINK
IFFAEA04	SGDSL	Option Definition	LINK
IFFAEA06	SSCIS	Option Definition	LINK
IFFAEA07	SCHAM	Option Definition	LINK
IFFAFA01	PLINE	Image Generation	LOAD and CALL

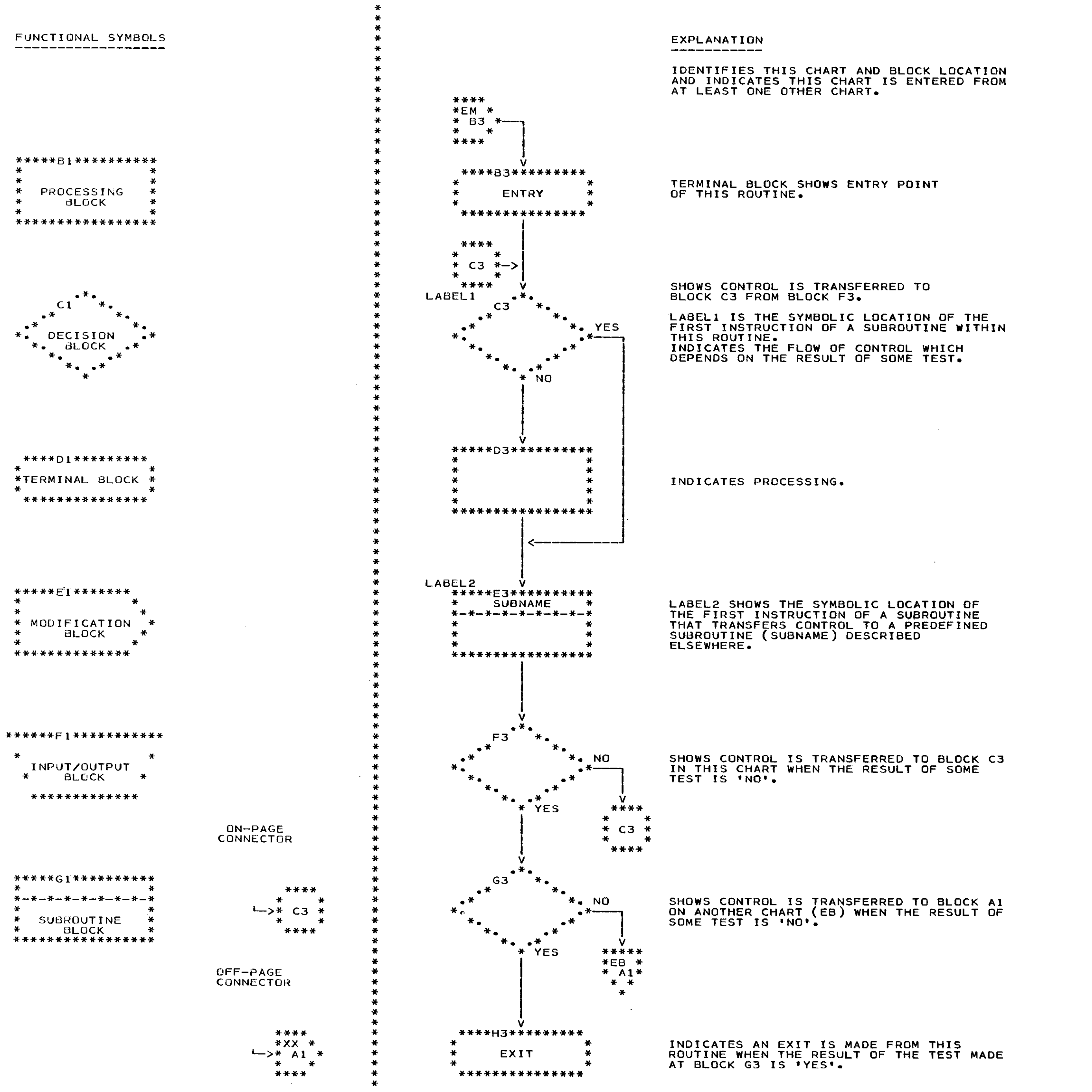
Module Name	Associated Routine	Type of Routine	Method of Entry
IFFAFA02	PSGMT	Image Generation	LOAD and CALL
IFFAFA03	PTEXT	Image Generation	LOAD and CALL
IFFAFA04	STPOS	Image Identification	LINK
IFFAFA05	BGSEQ	Image Identification	LINK
IFFAFA06	ENSEQ	Image Identification	LINK
IFFAFA07	ENSUB	Image Identification	LINK
IFFAFA08	LKSUB	Image Identification	LINK
IFFAFA09	INCL	Image Control	LINK
IFFAFA10	OMIT	Image Control	LINK
IFFAFA11	EXEC	Image Control	LOAD and CALL
IFFAFA12	RESET	Image Control	LINK
IFFAFA13	IDPOS	Image Control	LINK
IFFAFA14	FSMOD	Image Control	LINK
IFFAFA15	STEOS	Image Generation	LINK
IFFAFA16	PPNT	Image Generation	LOAD and CALL
IFFAFA17	MVPOS	Image Generation	LINK
IFFAFA18	BGSUB	Image Identification	LINK
IFFAFA19	ORGDS	Image Control	LINK
IFFAGA01	LOCPN	Light Pen	LINK
IFFAGA02	BGTRK	Light Pen	LINK
IFFAGA03	RDTRK	Light Pen	LINK
IFFAGA04	ENTRK	Light Pen	LINK
IFFAGA05	DFSTR	Stroke Generator	LINK
IFFAGA06	PLSTR	Stroke Generator	LINK
IFFAGA07	ORGEN	Miscellaneous	LINK
IFFAGA08	CNVRT	Miscellaneous	LINK
IFFAHA01	Flow Control Management	Internal	LOAD and CALL
IFFAHA02	Buffer Management	Internal	LOAD and CALL
IFFAHA03	Key Table Management	Internal	LOAD and CALL
IFFAHA04	Data Generator	Internal	LOAD and CALL

Module Name	Associated Routine	Type of Routine	Method of Entry
IFFAHA05	Data Store	Internal	LOAD and CALL
IFFAHA06	Scaling	Internal	LOAD and CALL
IFFAHA07	Scissoring	Internal	LOAD and CALL
IFFAHA09	CANCEL Key (Panic Key)	Attention Related	LOAD and CALL
IFFAHA11	Director Part 2	Internal	LOAD and CALL
IFFAHA12	Status Table	Only Loadable	LOAD
IFFAHA13	Update	Internal	LOAD and CALL
IFFAHA14	Update	Internal	LOAD and CALL
IFFAHA15	Scaling	Internal	LOAD and CALL
IFFAHA16	Stroke Table	Only Loadable	LOAD
IFFAJA01	ITRC	GSP Function	LOAD and CALL
IFFAJA02	ITBP	GSP Function	LOAD and CALL
IFFAJA03	RTBP	GSP Function	LOAD and CALL
IFFAJA04	ITST	GSP Function	LOAD and CALL
IHCGSP01	INGSP Part 1	Initiation	User's Program
IHCGSP02	TMGSP Part 1	Termination	User's Program
IHCGSP03	Director Part 1	Internal	User's Program

APPENDIX C. ACRONYMS AND ABBREVIATIONS

abs	absolute
addr	address
ADEQ	attention data entry queue
asgn	assignment
ATLCB	attention level control block
attn	attention
avail	available
BCT	buffer control table
bfr	buffer
chars	characters
COMAREA	communication area
CPU	central processing unit
ctrl	control
DCB	data control block
DECB	data event control block
EOS	end-order-sequence
EP	entry point
EQ	equal
FCT	flow control table
flo	flow
GACB	graphic attention control block
GDOA	graphic data output area
GDS	graphic data set
GDSCB	graphic data set control block
gen	generator
GSPCB	graphic subroutine package control block
GT	greater than
GTMCB	graphic terminal control block
ID	identification
incr	incremental
LP	light pen
LT	less than
mgt	management
no.	number
OACB	ouput area control block
opt	optimized
ovflo	overflow
PFK	programmed function keyboard
pts	pointers
reg	register
seg	segment
subrtn	subroutine
tbl	table
UCB	unit control block
unres	unresolved

Figure 20 shows the functional symbols used by Autochart. These symbols and the explanation of the sample flowchart in the figure describe the way they are used in the flowcharts for this manual.



IDENTIFIES THIS CHART AND BLOCK LOCATION AND INDICATES THIS CHART IS ENTERED FROM AT LEAST ONE OTHER CHART.

TERMINAL BLOCK SHOWS ENTRY POINT OF THIS ROUTINE.

SHOWS CONTROL IS TRANSFERRED TO BLOCK C3 FROM BLOCK F3.

LABEL1 IS THE SYMBOLIC LOCATION OF THE FIRST INSTRUCTION OF A SUBROUTINE WITHIN THIS ROUTINE.
INDICATES THE FLOW OF CONTROL WHICH DEPENDS ON THE RESULT OF SOME TEST.

INDICATES PROCESSING.

LABEL2 SHOWS THE SYMBOLIC LOCATION OF THE FIRST INSTRUCTION OF A SUBROUTINE THAT TRANSFERS CONTROL TO A PREDEFINED SUBROUTINE (SUBNAME) DESCRIBED ELSEWHERE.

SHOWS CONTROL IS TRANSFERRED TO BLOCK C3 IN THIS CHART WHEN THE RESULT OF SOME TEST IS 'NO'.

SHOWS CONTROL IS TRANSFERRED TO BLOCK A1 ON ANOTHER CHART (EB) WHEN THE RESULT OF SOME TEST IS 'NO'.

INDICATES AN EXIT IS MADE FROM THIS ROUTINE WHEN THE RESULT OF THE TEST MADE AT BLOCK G3 IS 'YES'.

Figure 20. Autochart Functional Symbols and Sample Flowchart

INDEX

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

READER'S COMMENTS

IBM System/360 Operating System: Graphic Programming Services for FORTRAN IV; Program Number 360S-LM-537

Y27-7152-0

Your comments will help us produce better publications for your use. Please check or fill in the items below, adding explanations and other comments in the space provided. All comments and suggestions become the property of IBM.

Which of the following terms best describes your job?

□ Programmer
□ Manager
□ Operator
□ Instructor
□ Systems Analyst
□ Engineer
□ Mathematician
□ Student/Trainee
□ Customer Engineer
□ Systems Engineer
□ Sales Representative
□ Other (explain) ____________

Does your installation subscribe to the SRL Revision Service? □ Yes □ No

How did you use this publication?

□ As an introduction
□ As a reference manual
□ As a text (student)
□ As a text (instructor)
□ For another purpose (explain) ______________________________________

Did you find the material easy to read and understand? □ Yes □ No (explain below)

Did you find the material organized for convenient use? □ Yes □ No (explain below)

Specific criticisms (explain below)

Clarifications on pages ______________________________
Additions on pages ______________________________
Deletions on pages ______________________________
Errors on pages ______________________________

Explanations and other comments:

No postage necessary if mailed in U.S.A.

FOLD FOLD

FIRST CLASS
PERMIT NO. 116
KINGSTON, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
NEIGHBORHOOD ROAD
KINGSTON, N. Y. 12401

ATTN: PROGRAMMING PUBLICATIONS
DEPARTMENT 637

FOLD FOLD

IBM®

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

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

Printed in U.S.A. Y27-7152-0

IBM

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

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