PACAF-ADAC INTERIM OPERATIONAL CAPABILITY. VOLUME III. PROGRAM DOCUMENTATION, SUPPORT AND UTILITY PROGRAM

Don Cole, et al

Informatics, Incorporated

Prepared for:

Rome Air Development Center

April 1974

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

AD778 601

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-74-75, Volume III	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): PACAF-ADAC Interim Operational Capability Volume III - Program Documentation, Support and Utility Program	5. TYPE OF REPORT & PERIOD COVERED: Interim - Dec 72 - Nov 73
	6. PERFORMING ORG. REPORT NUMBER: None
7. AUTHOR(s): Don Cole, Raymond Isawa, James Lum, et al.	8. CONTRACT OR GRANT NUMBER(s): F30602-72-C-0205
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Informatics, Inc. 6000 Executive Boulevard Rockville, MD 20857	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 31025F IDHS0501
11. CONTROLLING OFFICE NAME AND ADDRESS: Rome Air Development Center (IRDA) Griffiss AFB NY 13441	12. REPORT DATE: April 1974
	13. NUMBER OF PAGES: 280
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release. Distribution Unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: Mr. John V. Weber (IRDA)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computers
Data Processing
On-Line Systems
Interactive Systems
Display

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This report contains flow chart and narrative information for an Interim Operational Capability (IOC) developed to support air defense analysis functions of the PACAF Air Defense Analysis Center. The IOC provides on-line interactive support to the analyst for file manipulation functions, such as file creation, complex query, file sorting, file output. Several graphic display capabilities applicable to air defense analysis are also provided in the IOC.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20.

The system operates on an IBM 360/40 using formatted files with the MODS file structure. Analyst interaction is provided via a BR-90 display console.

This volume covers descriptions of IOC monitor services and functional support programs, utility programs, and program maintenance support programs.

PACAF-ADAC INTERIM OPERATIONAL CAPABILITY
Volume III - Program Documentation, Support and Utility Program

Don Cole
Raymond Isawa
James Lum
et al

Informatics, Incorporated

FOREWORD

This document reports on development work performed under Contract F30602-72-C-0205, Program Element No. IDHS0501, by Informatics, Inc., and its sub-contractors, the Bunker-Ramo Corporation and Integrated Analysis Corporation.

The Rome Air Development Center Project Engineer is Mr. John V. Weber; the Hq PACAF Contract Monitor is Mr. Leonard Frank.

Most of the contract effort was performed on-site at HQ PACAF at Hickam AFB in Honolulu. The cooperation and forbearance of the many personnel of that Headquarters who contributed to the project are gratefully acknowledged.

The project is organizationally under the Systems Development Division of the Systems and Services Company of Informatics, Inc. The Informatics, Inc. Project Manager is Mr. Don Cole. The technical staff at the end of the contract includes Mr. Raymond Isawa and Mr. James Lum. Contributions were made during the contract period by Messrs. David Del Rio, Tom Lawson, Jeff Loker, Kirk Lubbes, Clint McIntosh, Bill McKinney, Bill Miller, Barry Pang, and Stu Stern.

Mrs. Susan Anderson provided administrative and secretarial support throughout the effort.

This report has been reviewed by the Office of Information, RADC, and approved for release to the National Technical Information Service (NTIS).

This report has been reviewed and is approved.

APPROVED: John V. Weber

JOHN V. WEBER
Project Engineer
RADC (IRDA)/X-3126

APPROVED: Franz H. Dettmer

FRANZ H. DETTMER, Colonel, USAF
Chief, Intel & Recon Division

FOR THE COMMANDER: John P. Huss

JOHN P. HUSS
Acting Chief
Plans Office

CHAPTER XIII

MONITOR SERVICES AND FUNCTIONAL SUPPORT PROGRAMS

Pages iii and iv are blank.

PROGRAM ID: MONITR
PROGRAM NAME: IOC Monitor

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: The Monitor's basic function is to determine the source of any interrupt from an on-line console, determine what has to be done to process the interrupt, validate the interrupt, load the appropriate application, and pass control to it.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The IOC Monitor program is the main control program for IOC. It is core-resident until IOC is rolled out. The Monitor is basically a loop which periodically checks for BR90 and 2260 interrupts, handles any interrupts which have occurred, continues processing any pending or deferred tasks and if no IOC processing is to be done, issues a WAIT macro which releases the CPU for processing of lower priority tasks in the background partition.

The basic loop consists of a series of tests. Initially, the Monitor polls the BR90 console. If no interrupt is pending, it polls the 2260 consoles. If no interrupts have been posted from the 2260's, the PENDQ program is called. (PENDQ is intended to check for IOC programs previously checkpointed. However, since the BR90 is the only display console connected to IOC, there is no pending queue and PENDQ simply does a return to MONITR.) Since there are no pending routines, the WAIT macro is issued, releasing the CPU. The WAIT time delay is adjustable and is currently set at 1 second.

When the Monitor detects an interrupt from the BR90, the core environment is saved and the environment required to process the

interrupt is paged into core if it is not already present. The currently active IOC programs ordinarily post all anticipated interrupts in the systems common area. This enables the Monitor to determine whether or not the interrupt is valid. If it is, the console program currently residing in core, if not applicable, is paged out. The applicable console program and its environment are then paged in and control is passed to it.

If an interrupt from a 2260 console is detected, the Monitor performs the same basic functions as it does for a BR90 interrupt. Note that having the Monitor validate the interrupts eliminates unnecessary paging as well as removing the validation burden from the application programs. It also eliminates the requirement to use core space in the application program area for this function.

LOGIC NARRATIVE:

STORE - Save the registers and establish base registers. Initialize loop count to 0.

LOOP - Initialize return code (RETCOD) to 0.

BR90 - If BR90 not active or if there is no BR90 interrupt pending, go to IBM2260. Load phase 1002 and call CKBR90. Upon return, test RETCOD to see if there is a BR90 interrupt. If there is, go to LOOPEND.

IBM2260 - If no 2260 is active, go to PENDQUE. Load phase 1008 and call CK2260. Upon return, test RETCOD to see if there is a 2260 interrupt. If there is, go to LOOPEND.

PENDQUE - If the pending queue is not active, go to WAIT. Load phase 1008 and call PENDQ. Upon return, test RETCOD to see if an entry in the pending queue was processed. If yes, go to LOOP.

WAIT - If we have not cycled through this loop three times (if we have not waited long enough) go to CALLWAIT. Otherwise, reset the loop counter to 0, call SVC59 and go to LOOP.

CALLWAIT - Issue SVC24 to set time to 1 second and call WAIT. Then fall through upon return.

DONTWAIT - Go to LOOP.

LOOPEND - Reset RETCOD to 0, restore the registers and return to the calling program.

CALLING SEQUENCE:

```
                LA      R2,$MONIT
                ST      R2,PARMAD
                MVI     PARMAD,X'80'
                L       R15,RTN
                BALR    R14,R15
                .
                .
                .
        RTN     DC      V(RTN)
        PARMAD  DC      F'0'
                DC      X'80'
```

OPTIONS: None.

CALLED BY: AWAIT

SCREENS USED: None.

INPUTS: MONITR tests three switches in the systems common (COMMON) area. They are:

SWT3 BR90 console active
SWT4 IBM2260 console active
SWT5 Pending Queue active

The program also tests the BR90 bit in the communications region (core location 16 decimal). MONITR calls XPHASE to load phases 1002 and 1008.

OUTPUTS: None.

ENTRY POINTS:

MONITR

REGISTER USAGE:

R1 - Used to pass phase number to XPHASE.

R4 - Program loop counter.

R14 - Address of return from subroutine call.

R15 - Address of called subroutine.

PROGRAM CONSTANTS:

N1002 - Phase number for utility phase 1002.

N1008 - Phase number for utility phase 1008.

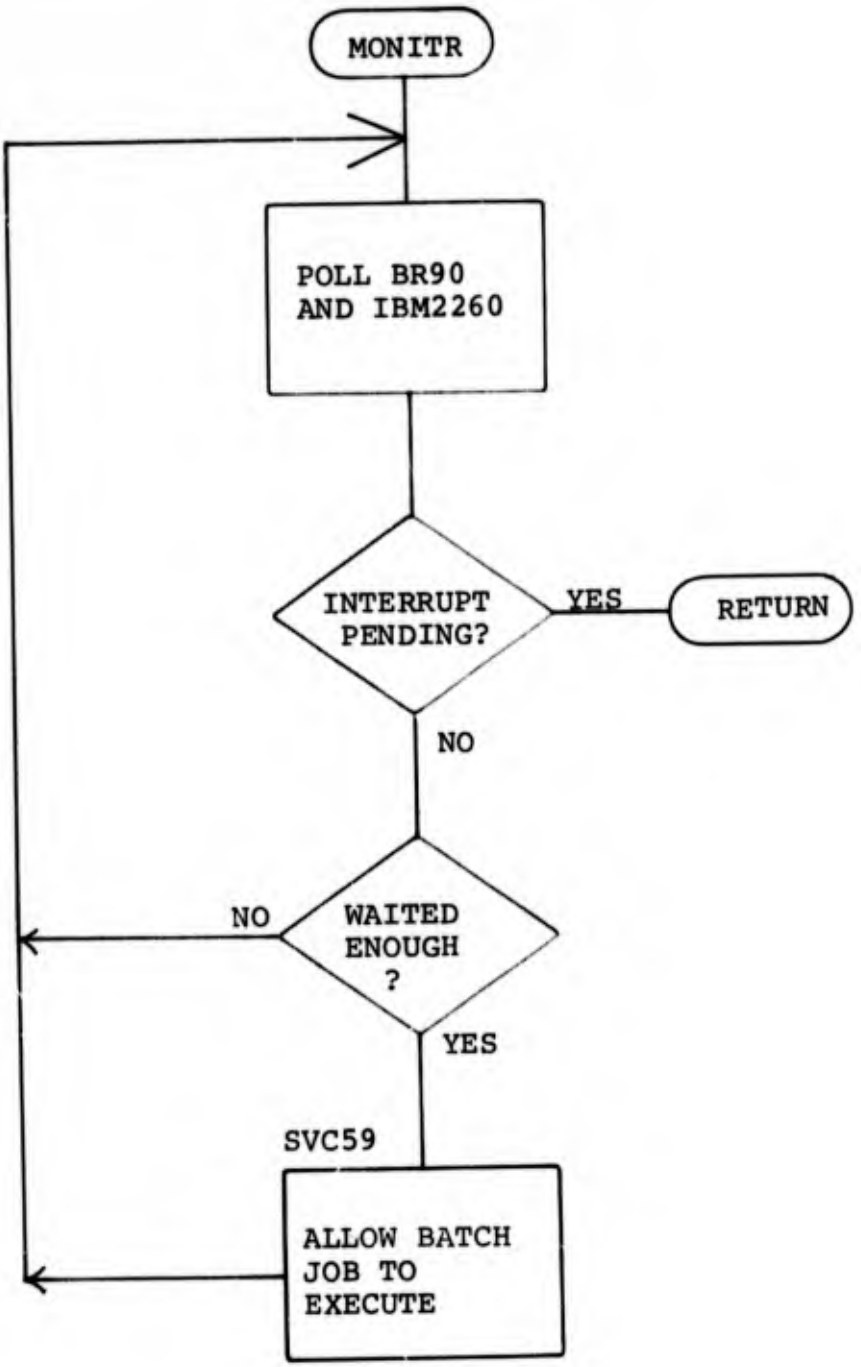
PROGRAM VARIABLES:

RETCOD - Return code set by calls to CKBR90 and CK2260.
This field exists in COMMON.
0 = No interrupt for that device.
Nonzero = interrupt exists.

ERROR MESSAGES: None.

SUBROUTINES CALLED: CKBR90, CK2260, PENDQ, SVC59, XPHASE.

MONITR: Logic Diagram



427

PROGRAM ID: RTN
PROGRAM NAME: IOC Paging Routine

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: RTN handles phase swapping which is needed when routines are called in different phases. TRACE allows a program to take a "snapshot" dump. GTDATE returns the system date.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: RTN is an executive routine used to establish and save the appropriate environments required to process IOC's dynamic overlay when an application subroutine located in one phase calls an application subroutine located in another program phase. In IOC, routines are not called directly but through RTN. When RTN receives control from the calling routine, it saves the current environment indicators such as the current routine ID, the phase number of the current routine, the numbers of the current VOL and PAGE, and copies the parameter list to be passed to the called routine.

RTN recognizes three categories of routines, graphic, utility, and applications. If the called routine is a graphic routine, RTN first checks to see if it is already in core. If it isn't, the phase that is in core is paged out and IIGRAF is called. IIGRAF loads the phase and transfers control to the called routine.

If the called routine is a utility routine, RTN first checks to see if it is already in core. If it isn't, the proper phase is paged into core and the Phase Table is searched for the transfer vector. For utility programs, the previous phase is not saved.

If the called routine is an applications routine, RTN checks to see if it is already in core. If it isn't, the previous phase is paged out and the proper phase is paged in. The Phase Table is then searched for the transfer vector.

When control is returned to RTN from the called routine, the previously saved indicators such as current routine ID, current phase number, etc., are restored. The environment has been re-established, control is returned to the calling routine.

LOGIC NARRATIVE:

RTN,IRTN, RRTN - Establish the base register. Pick up a pointer to next DSA entry from NEXTDSA and update NEXTDSA.

STORE - Save CRTN, CPHASE, CVOL and CPAGE in DSATBL. If the call has no parameters, go to RTNTYPE. Otherwise, loop through the parameter list and move the entries to the parameter list area. Update the list pointer, NEXTPARM.

RTNTYPE - Pick up the requested routine's ID and save it in REQRTN. If it is a graphics routine, go to GRAPHICS. If it is a utility routine, go to UTILITY. Otherwise, fall through to process a small or large phase.

PROGRAM - Set up initially to process a small phase. If current phase (CPHASE) in core is a small phase, go to LOOP2. Otherwise, set up to process a large phase.

LOOP2 - Loop through current phase table, looking for the CRTN in the phase table. If this is not found, go to TBLERROR. Otherwise, when the correct routine is found, go to PTR.

PTR - Check if any subroutines are called by the current routine (CRTN). If not, go to TBLERROR. Otherwise, find required routine in the phase table, then go to FOUNDRTN.

FOUNDRTN - Obtain the phase number from the phase table, then go to GETPHASE.

UTILITY - Get the correct phase number from UTILTBL. Check if MONITR is being called. If not, go to GETPHASE. Otherwise, set up a call to MONITR, then go to CALL.

GRAPHICS - Get IIGRAF root phase by calling XPHASE. Then, update the CRTN. Set up a call to IIGRAF, with CRTN and the pointer to the list of parameters as inputs. Then, go to CALL.

TBLERROR - Put up message by calling PNTMSG. Then, return to program which reloads system without a dump.

GETPHASE - Load the required phase (shown in REQPHASE) by calling XPHASE.

TVECTOR - Update the transfer vector of the routine or utility from the new phase table. Loop through the phase table to find the required routine. If not found, go to TBLERROR. Otherwise, move the address of the routine into ADDRESS. Register 1 is set to point to a list of parameters from PARMLIST.

CALL - Process trace option.

CALLRTN - Call the routine. On return, save Register 0 and floating Register 0, and clear the interrupt tables of any references to the current routine.

RECRTN - Re-establish CRTN and CPHASE from the DSATBL. If no change occurred in phases, go to CKLOAD. Otherwise, load in old phase.

CKLOAD - If the XVOL or XPAGE was not called which means VOL and PAGE were not changed, go to REPARM, since this means we don't want to restore VOL or PAGE. Otherwise, check if VOL or PAGE was changed. If not, go to REPARM. Otherwise, get XVOL and XPAGE into core by calling XPHASE.

REVOL - Check if VOL was changed. If not, go to REPAGE. Otherwise, call XVOL to get in the correct VOL.

REPAGE - Check if PAGE was altered. If not, go to REPARM. Otherwise, call XPAGE to get in old PAGE.

REPARM - Pop up the DSATBL and parameter list, then return to the calling routine.

TRACE - This is a subroutine which is used to take snapshot dumps in RTN.

GTDATE - This routine gets the system date for the caller.

CALLING SEQUENCE: CALL RTN($SUBR,J,K,...,N)
I = IRTN($SUBR,J,K,...,N)
Z = RRTN($SUBR,J,K,...,N)

where:
$SUBR is the routine ID of the routine to be called.
J,K,...,N are parameters for the routine to be called.
I and Z are the integer values returned from $SUBR in the case of function subroutines.

CALL TRACE(REFNO,FROM,TO)

where:
REFNO is a number which will be printed by PNTMSG. If REFNO contains ' BR' it will print message and dump. If REFNO contains ' AB', it will only dump. Otherwise, it will print message and dump.
FROM is the start address of the dump.
To is the end address of the dump.

CALL GTDATE(DATEFL)

where:
DATEFL is an 8-byte field to receive the system date.

OPTIONS: Any of the three routines can be called as independent routines.

CALLED BY: RTN is called by routines which call other IOC routines. TRACE is called by RTN in case of error, but can also be called by any other routine which needs a dump. GTDATE will be called by any routine which needs the current date.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. $SUBR,J,K,...,N are inputs to RTN. Also, RTN makes use of UTILTBL, DSATBL, and the phase table of the current and requested phase. UTILTBL is a table of phases in which utility routines reside. DSATBL is a push-down stack containing various data cells needed to restore environments. The information in the phase table of the current phase is used by RTN to find the phase of the required routine. The required phase's phase table is then used to find the actual entry point for the required routine.

OUTPUTS: If RTN is called as an integer or real function subroutine, a value is returned to I or Z, respectively. In addition, on return from RTN, the interrupt tables are cleared of references which are no longer needed. RTN also handles phase swapping and VOL and PAGE exchanges by calling XPHASE, XVOL and XPAGE.

ENTRY POINTS:

RTN
IRTN
RRTN
TRACE
GTDATE

PROGRAM CONSTANTS:

COMMON - Pointer to COMMON area.

MONITR - Pointer to MONITR routine.

IIGRAF2 - Pointer to IIGRAF2 routine.

XPHASE - Pointer to XPHASE routine.

XVOL - Pointer to XVOL routine.

XPAGE - Pointer to XPAGE routine.

PNTMSG - Pointer to PNTMSG routine.

LOADREG1- Pointer to LOADREG1 routine.

PARMADDR- Last parameter in call to XPHASE, a pointer to PARM.

MASK1 - Mask used to test for end of parameter list.

DELLIST - Parameter list for call to PNTMSG.

N0 - Contains 0.

N1 - Contains 1.

N1501 - Contains 1501.

N1003 - Contains 1003.

DSASIZE - Contains size of the DSA entry.

PROGRAM VARIABLES:

ADDRESS - Area for pointer to routine to be called.

CRTNPTR - Area to hold CRTN while searching for routine in phase table.

PARMAD - Area for parameter list for call to IIGRAF2.

PARM - Parameter list for call to XPHASE.

CTREG1 - Area for holding R1 for TRACE.

PARM1 - Message number in call to PNTMSG.

PARM2 - CRTN number in call to PNTMSG.

PARM3 - Phase number in call to PNTMSG.

REQPHASE- Required phase number.

REQRTN - ID number of required routine.

LASTRTN - ID number of last routine called.

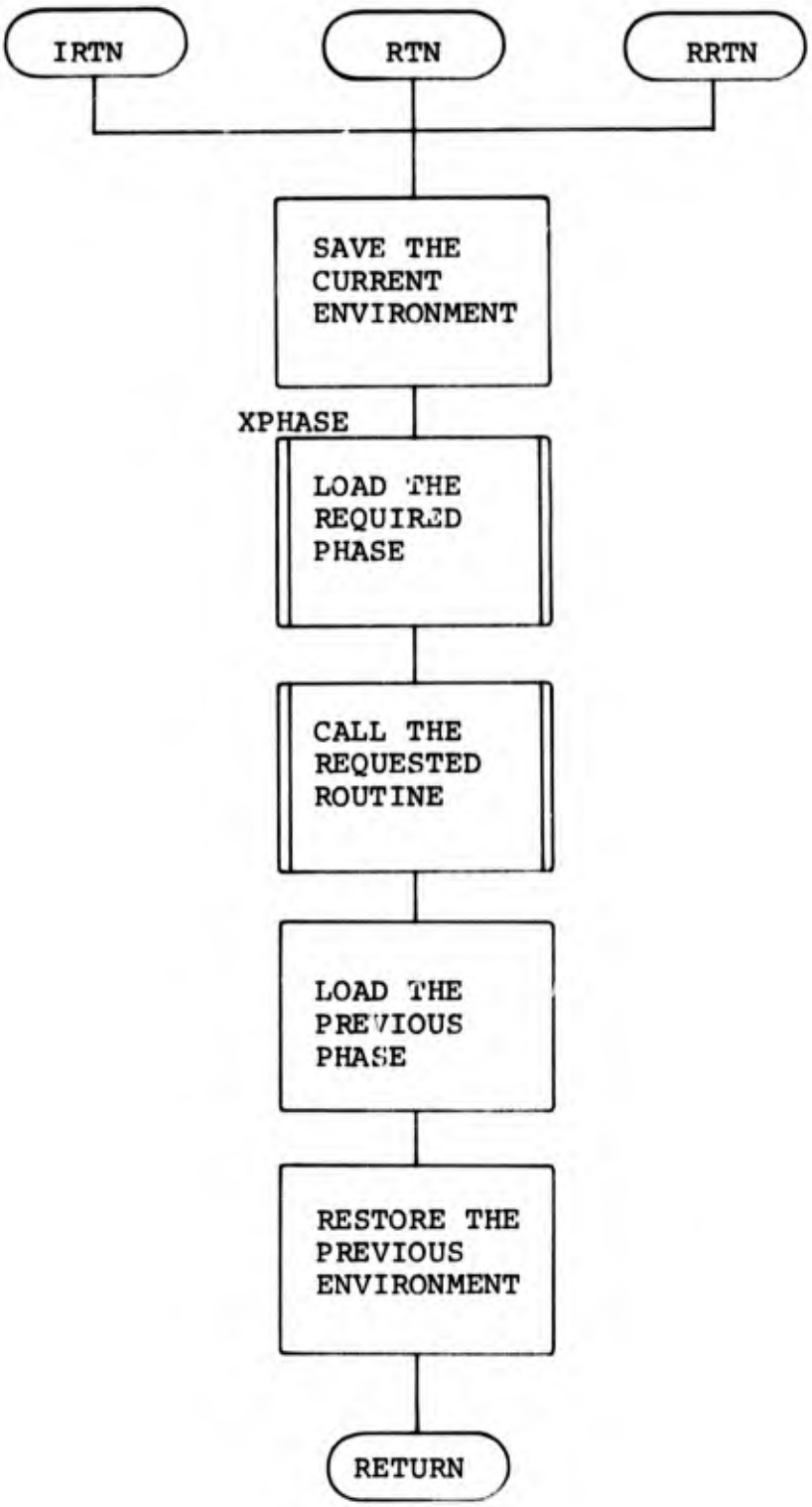
DSA - A DSECT for the DSATBL entry containing:

BRTN	-	routine ID of routine issuing call.
BPHASE	-	phase number of routine issuing call.
REGSA	-	register save area.
BVOL	-	calling routine's current VOL number.
BPAGE	-	calling routine's current PAGE number.
PARMLIST	-	address of parameter list in PARMTBL.

ERROR MESSAGES: Message 4 in PNTMSG, in addition to a dump. This occurs in various places in RTN in case of error.

SUBROUTINES CALLED: None.

RTN: Logic Diagram



PROGRAM ID: AWAIT
PROGRAM NAME: Await

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: AWAIT sets up the masks for all of the interrupts that have been enabled and calls MONITOR which actually goes into a "wait" state.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Because IOC is an interactive on-line system, many of the applications programs need to pause and wait for a user response at various points in the program. In addition to entering a pause state, the programs must enable and disable interrupts in order to guide the user through these pauses. If the interrupts were enabled and disabled whenever the applications program issued a request, it is conceivable that the lights on the BR90 would be blinking on and off at odd times and thereby distracting the user. Therefore, although the programs may issue requests to enable and disable interrupts at any point in the programs, these requests merely set indicators which are accumulated until the program is ready to enter a pause state. When the program issues a call to AWAIT, all of the interrupts are enabled and/or disabled before MONITOR is called to accomplish the wait.

LOGIC NARRATIVE:

AWAIT - Initialize the registers and save them.

LOOP1 - Set up mask for call to LVFK1 and LVFK2. Then call the routines to enable the VFK interrupts.

LOOP2 - Set up mask for call to LQURY. Then call the routine to enable the query key interrupts.

SETD - Set up call to STINT. Call the routine to turn on the status lights. Finally, call MONITOR, then return to the calling program.

CALLING SEQUENCE: CALL RTN($AWAIT)

OPTIONS: None.

CALLED BY: All IOC applications programs which need to wait for a user action.

SPECIAL NOTES: This subroutine is considered a small program. Therefore, programs that use AWAIT must include an entry in their phase table.

SCREENS USED: None.

INPUTS: None.

OUTPUTS: None.

PROGRAM CONSTANTS:

COMMON - Pointer to COMMON area.

RTN - Pointer to RTN routine.

N0 - Constant of zero.

N1 - Constant of one.

READY - Constant that defines status lights setting.

PROGRAM VARIABLES:

PARMAD - Parameter list used in subroutine calls. Points to PARM.

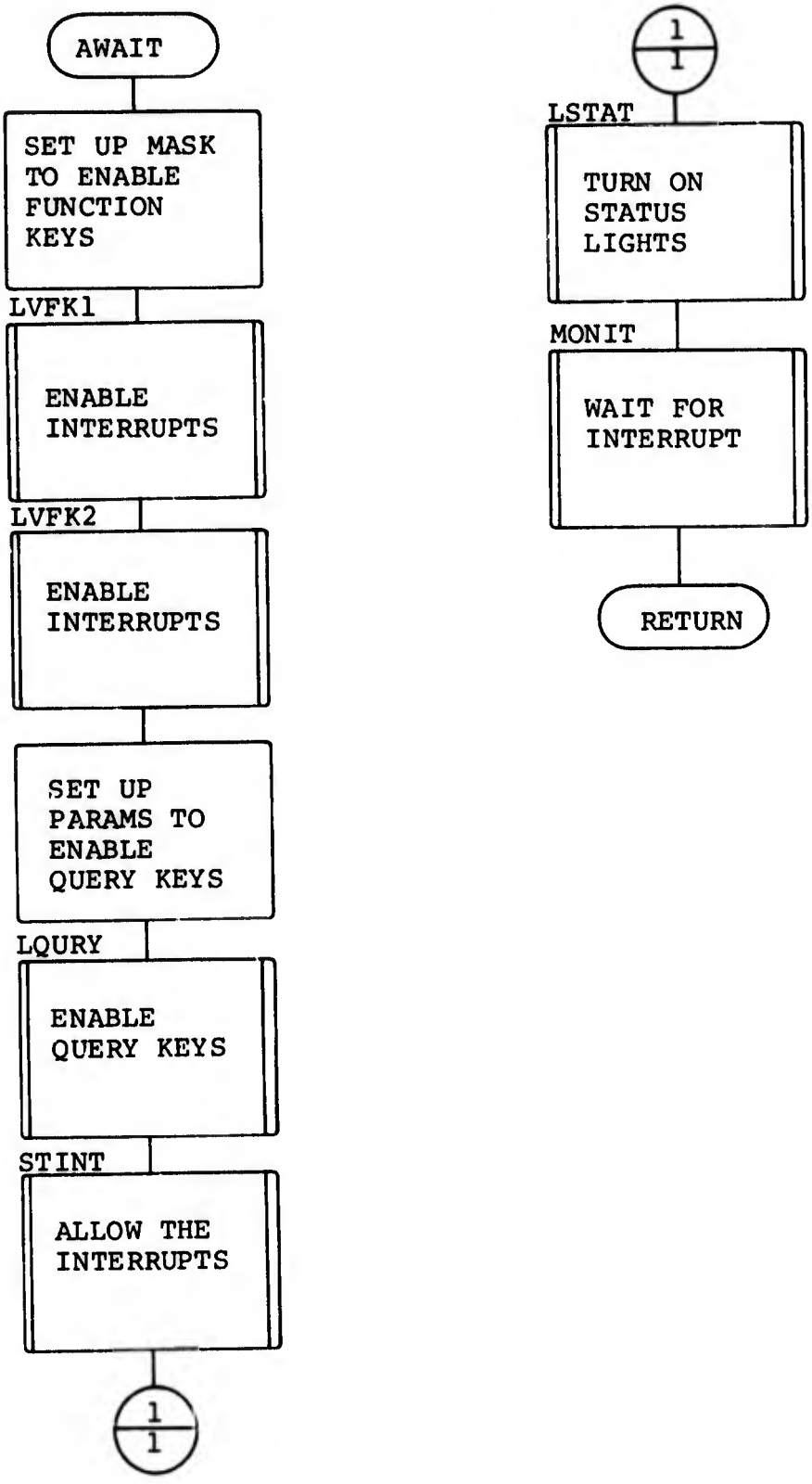
PARM - Parameter values for subroutine calls.

REGSA - Register Save Area.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: LVFK1, LVFK2, STINT, MONIT, LQURY, LSTAT

AWAIT: Logic Diagram



PROGRAM ID: INTON
PROGRAM NAME: Interrupt Enable and Disable Routine

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This routine provides the IOC programs with the capability to specify the types of interrupts that are to be enabled or disabled. INTON sets flags in an interrupt table so that certain interrupts will be recognized. INTOF does the opposite task of clearing entries in the table.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Under IOC, it was desired that interrupt handling be simplified and standardized as much as possible. Thus, in IOC, the programmer need only indicate which interrupt he wishes to receive. All enabling of interrupts, control of function keys and status lights, setting of appropriate interrupt control information, etc. for each application program is automatically handled by IOC. INTON and INTOF handle the posting and deletion of console interrupts in the system communication COMMON area.

In particular, these routines set entries in the interrupt table to either a 0 (INTOF) or the ID of the calling routine (INTON). The entries in the table specify the routines that will gain control when an interrupt occurs. These routines are called by the programmer to activate or deactivate a specific or a series of console interrupts.

LOGIC NARRATIVE:

INTON - Set routine to store the ID of the calling routine into the INTERRUPT table. Branch to common code at INTONA.

INTOF - Set routine to store a 0 into the INTERRUPT table.

INTONA - Establish base registers for program and COMMON/ COMMON/. Initialize index into parameter list to 0, then go to LOAD.

UPDATE - Update pointer to parameter list.

LOAD, TEST - Using the current parameter, branch to appropriate code to process this type of interrupt.

CKLP - Process light-pen interrupts by storing either the routine ID or 0 into the light-pen table. If the light-pen number is 45 or more, store ID or 0 into LP(45). If an LP number was less than 1 or greater than 294, go to TEST. If last parameter is detected, go to FINAL.

CKFK - Process VFK interrupts. If last parameter is detected, go to FINAL. If VFK number is less than 1 or greater than 30, go to TEST. Otherwise, store ID or 0 into VFK table.

CKPR - Set projector interrupt to either ID or 0. Then return through FINAL.

CKQK - Process Query Key interrupts. If query key number is less than 1 or greater than 15, go to TEST. Otherwise, store ID or 0 into QK table. If last parameter detected, go to FINAL.

FINAL - When hit end of parameter list, come here to return.

CALLING SEQUENCE: CALL RTN($INTON,XX,I,J,K,...)

CALL RTN($INTOF,XX,I,J,K,...)

where:

XX = LP,FK,PR, or QK for Light-Pen, Function Key, Projector, or Query Key

I,J,K,... = Specific lines, keys for items corresponding to XX.

OPTIONS: None.

CALLED BY: IOC routines that wish to activate or deactivate console interrupts.

SPECIAL NOTE: It should be noted that under this interrupt handling system, there is some danger of destroying interrupts. When a routine is called, it can set interrupts in the common tables. On return, the interrupts set by the called

routine will be reset to 0. Thus, if these are the same as those that were set by the calling routine, they will be lost and no indication of this condition is returned to the calling program. Therefore, the programmer should familiarize himself with all subroutines that are called by his program to make sure that there is no conflict in enabling and disabling interrupts.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE.

OUTPUTS: The following areas in COMMON are set:

PROJ	-	Flag for the projector interrupt
LP	-	Table for the light-pen interrupts
VFK	-	Table for the variable function key interrupts
QPANEL	-	Table for the query key interrupts

ENTRY POINTS:

INTON
INTOF

REGISTER USAGE:

R2	-	Index into parameter list.
R3	-	Pointer to current parameter.
R6	-	Base for interrupt tables.
R4	-	Offset into interrupt tables.
R10	-	Base register

PROGRAM CONSTANTS:

COMMON	-	Pointer to COMMON/COMMON/.
N0	-	Constant of 0.
N294	-	Constant of 294.

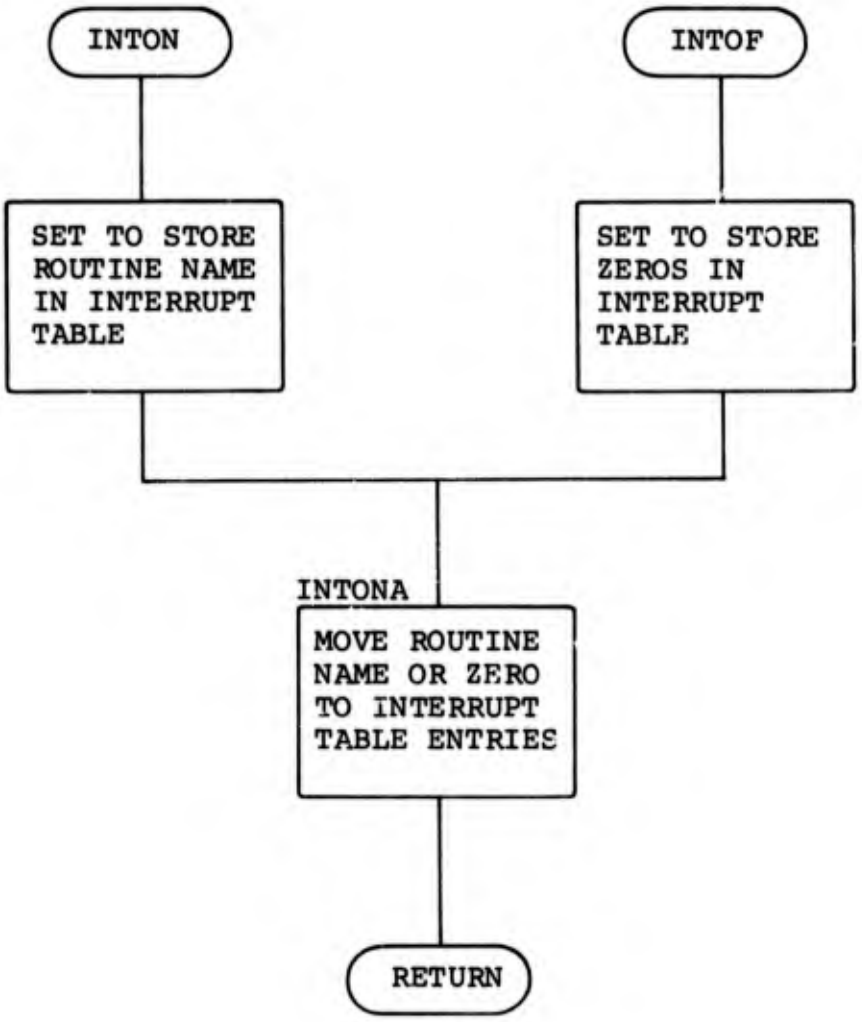
PROGRAM VARIABLES:

TEMP4	-	Temporary area for holding parameter.
REGSA	-	Save area for the registers.
CURRTN	-	Area for item to be set into table. Either the routine ID or a 0.

ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

INTON: Logic Diagram



PROGRAM ID: CKBR90
PROGRAM NAME: Primary BR90 Interface

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: CKBR90 checks for any pending BR90 interrupts and processes these interrupts.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: One of the most common functions in IOC is the check for BR90 interrupts. Rather than having this function scattered throughout the system, it has been developed as a subroutine which validates any interrupts it has detected. This results in the reduction of I/O and overlay operations.

LOGIC NARRATIVE:

CKBR90 - Go to HERE.

HERE - Save the registers and establish the base registers. Initialize the BR90 Interrupt Tables to zeros. Call DETECT through IIGRAF to check for a BR90 interrupt. Convert the return code from floating point to integer and store it in IBR90. If the number is negative, make it positive. Determine type of interrupt and take on one of the following branches:

No Interrupt	Go to RETURN
LP	Go to CKLPEN
VFK	Go to CKVFKEY
END Key	Go to CKEKEY
CANCEL Key	Go to CKCKEY

PROJECTOR	Go to CKPROJ
QUERY Key	Go to CKQKEY

Otherwise, go to PRINTERR.

CKLPEN - Compute the FV number and store in IFV. Check the Interrupt Table to see if a routine has been assigned to that FV. If no routine was assigned, go to RETURN1, otherwise go to RETURN2.

CKVFKEY - Initialize to point to the first VFK entry minus 2 bytes.

VFKA - Check the Interrupt Table to see if a routine has been assigned to that VFK. If no routine assigned, go to RETURN1; otherwise, store the routine ID in INTRTN and go to RETURN2.

CKEKEY - Check SWT6 +1 for TRACE feature and set the status light in SLIGHT.

BCLSTAT - Branch to CLSTAT to turn on status lights. Upon return, go to RETURN1.

CKCKEY - If TRACE feature is on, go to CALLDUMP; otherwise, call CPLTR through IIGRAF. Call CLSTAT to turn off status lights. Upon return, go to EOJ.

CALLDUMP - Call TRACE for dump of program area. Go to RETURN1.

CKPROJ - If no routine assigned to this interrupt, go to RETURN1. Otherwise, move the routine ID to INTRTN and go to RETURN2.

CKQKEY - If Print Key interrupt, go to QKEY7. If Plot Key interrupt, go to QKEY8. Otherwise, go to VFKA.

QKEY7 - Load routine ID of PRTCH in R3.

QKEY7A - Call CLSTAT to turn off Ready light. Call PRTCHR or PLOTER. Upon return call CLSTAT to turn Ready lights on. Go to RETURN1.

QKEY8 - Load routine ID of PLOTR in R3. Go to QKEY7A.

CLSTAT - Set up parameter table for LSTAT, then call LSTAT through IIGRAF. Return to calling code.

RETURN1 - Zero out IBR90 and RETCOD and go to RETURN.

RETURN2 - Set up parameter table for STINT, then call STINT through IIGRAF. Call LVFK1, LVFK2 and LQURY through IIGRAF. Turn off Ready light indicators and branch to CLSTAT to set the Status Lights. Upon return, fall through to RETURN.

RETURN - Restore the registers and return to the calling program.

PRINTERR - Abort the program.

CALLING SEQUENCE: CALL RTN($CBR90)

OPTIONS: None.

CALLED BY: MONITOR

SCREENS USED: None.

INPUTS: CKBR90 gets the interrupt code by calling DETECT. It uses the Interrupt Table to validate the interrupt code.

OUTPUTS: CKBR90 fills in the following COMMON items: IBR90, INTRTN, IFV. It also sets the Ready Lights and the Status Lights.

ENTRY POINT:

CKBR90

REGISTER USAGE:

R10 - Base register for CKBR90.

R8 - Base register for COMMON.

R15 - Address of called routine.

R14 - Return address for subroutine calls.

R1 - Pointer to parameter table in subroutine calls.

PROGRAM CONSTANTS:

BR90 - Contains literal of "BR".

NONREADY- Bit setting to turn off Ready Lights. "FBDEF7BF".

READY - Bit setting to turn on Ready Lights. "04210840".

CONV - Unnormalized special value used in converting floating point to integer.

N102400 - Length of F2 partition used in call to TRACE.

PROGRAM VARIABLES:

WORK - Temporary cell used in converting floating point return code to integer.

REG2 - Save area for contents of register 2.

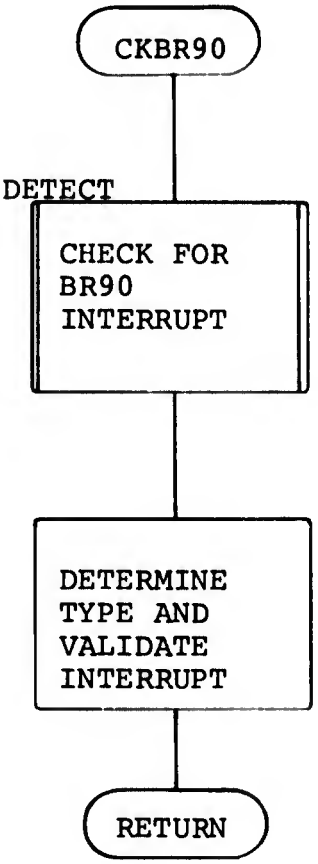
REG14 - Save area for contents of register 14.

PARMAD - Parameter table used in calls to various subroutines.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: DETECT, IIGRAF, LVFK1, LVFK2, LQURY, PRTCH, PLOTER, SETINT, LSTAT, PCLOSE.

CKBR90: Logic Diagram



PROGRAM ID: XPHASE
PROGRAM NAME: Swap a Program Phase Into Core

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: Given a phase number, XPHASE will load the phase into core, making sure that the correct phases are saved and restored.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The swapping of program phases in IOC is handled by two programs. RTN and XPHASE. When a routine is called for through RTN, it first determines whether the requested routine is a utility, graphic or application routine. Then, it checks if the routine is already in core. If it is in core, RTN passes control to it. If it is not in core, RTN calls XPHASE to swap the required phase into core. XPHASE will save the contents of the affected area or areas, and swap in the required phase.

LOGIC NARRATIVE:

XPHASE - Check if trace switch (SWT) is off. If so, go to 11001. Otherwise, take a snapshot dump by calling TRACE.

11001 - Using REQPHA (required phase number), determine the type of the desired phase. Branches are taken as follows:

Small phase	10
Large phase	501
Utility phase	1001
Graphic phase	1501

10 - Process a small phase. If the desired phase is already in area 4, go to 9000. Otherwise, process the phase. If area 4 is unoccupied, go to 8000 to fill it in. Otherwise, set PHASE flag to 4. PHASE is used to indicate what action should be taken later. If the phase in area 4 (indicated by PHASE4) is a small phase, go to 5500. Otherwise, phase in area 4 is part of a large phase. Set PHASE to 2, to indicate that IGRAF must be restored in area 3 before resuming execution of the program. Then, go to 6000.

501 - Process a large phase. If the desired phase is already in area 3 and 4, go to 9000. Otherwise, if both area 3 and 4 are inactive, go to 8500. If they are not both inactive, save one or more of these areas. Then, set PHASE to 5. If area 3 contains a large phase, go to 6000. Otherwise, if area 3 is active and area 4 is inactive, go to 5000 to save only area 3, as a graphics area (this is true since area 3 is active, but does not contain part of a large phase). If area 4 is active, and area 3 is inactive, go to 5500, to save only area 4. Otherwise, there must be a small phase in area 4 and a graphic phase in area 3. So, both must be saved. To do this, set PHASE to 1, and go to 5000.

1001 - Process a utility phase. If required phase is already in area 2, go to 9500. Otherwise, go to 7000. No utility phase is ever saved.

1501 - Process a graphic phase. If required phase is already in area 3, go to 9000. Otherwise, the phase must be swapped into core. If area 3 is inactive, go to 7500. Otherwise, set PHASE to 3, and go to 6000.

5000 - Save area 3 on disk, set PHASE3 to show area 3 is inactive, and go to 6500.

5500 - Save area 4 on disk and set PHASE4 to show that area 4 is inactive. If PHASE is set to 1, set PHASE to 5. This is done to force a branch to restore a large phase later. In either case, go to 6500.

6000 - Save areas 3 and 4 on disk, as a large phase. Set PHASE3 and PHASE4 to show that both areas are inactive.

6500 - Branch to a block of code, depending on the value of PHASE:
1, go to 5500
2, go to 8000
3, go to 7500
4, go to 8000
5, go to 8500

7000 - Get required utility phase into area 2. Set PHASE2 to indicate that area 2 is active, and go to 9500.

7500 - Get required graphic phase into area 3. Set PHASE3 to indicate that graphic routine is in area 3, and go to 9000.

8000 - Get required small phase into area 4, and set PHASE4 to show that area 4 is active. If a large phase was previously in core (PHASE=2), go to 7500 to restore graphics area. Otherwise, go to 9000.

8500 - Get required large phase into areas 3 and 4, and set PHASE3 and PHASE4 to show that both areas are active.

9000 - Set the new CPHASE from REQPHA.

9001 - Take snapshot dump, if required. Then, return to the calling routine.

9500 - Set CUTIL to REQPHA. Then, go to 9001.

CALLING SEQUENCE: CALL XPHASE(REQPHA)

where:
REQPHA - The phase number of the desired program phase.

OPTIONS: None.

CALLED BY: RTN

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. XPHASE uses REQPHA to determine what phases are affected. XPHASE also uses data sets 8 and 10 for program phases. In addition, the following items from COMMON are used: PG7288, PG3520, A2RECN, A3RECN, A4RECN, A5RECN, PHASE2, PHASE3, PHASE4.

OUTPUTS: XPHASE will swap the desired program phase into core. Also, CUTIL and CPHASE are set by XPHASE. CUTIL contains the phase number of the current utility phase in core, while CPHASE contains the current phase that is active.

ENTRY POINTS:

XPHASE

PROGRAM CONSTANTS: None.

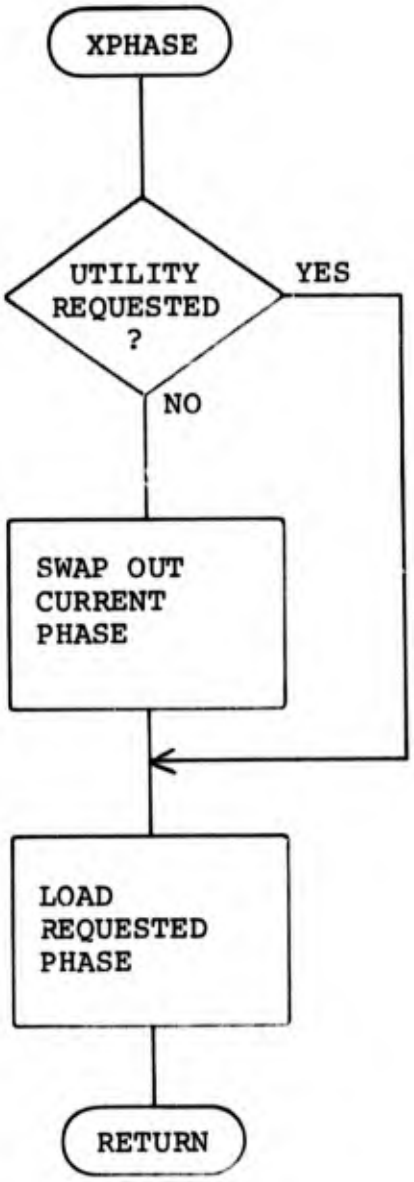
PROGRAM VARIABLES:

PHASE	-	A flag which indicates what action should be taken after phases are saved.
A2	-	Area 2 for utility phases.
A3	-	Area 3 for graphic and large phases.
A4	-	Area 4 for small and large phases.
A34	-	Area 3 for large phases.
IREC	-	Computed record number for read or write.

ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

XPHASE: Logic Diagram



PROGRAM ID: XVOL
PROGRAM NAME: VOL and PAGE Exchange Routines.

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: The various entry points in this module allow the user to interchange the contents in the VOL and PAGE areas of core with the VOL and PAGE areas on disk.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: VOL and PAGE are contiguous COMMON areas in core. They were designed to be used in conjunction with applications programs for general storage and to pass parameters from one subroutine to another. VOL and PAGE are independent areas which are pagable under programmer control. The programmer is provided with utility subroutines which retrieve or save each of the COMMON areas independently.

XVOL and XPAGE are used to exchange the current COMMON for another from disk. SVOL and SPAGE are used to save, but not replace, the current copy of COMMON. RVOL and RPAGE are used to replace, but not save, the current COMMON.

LOGIC NARRATIVE:

XVOL - If the trace switch (SWT) is off, go to 11001. Otherwise, turn on TRACE.

11001 - Check if NPAGE is valid ($0 \leq NPAGE \leq 50$). If so, go to 10.

5 - Otherwise, call ABEND.

10 - If VOL(NPAGE) is in core, go to 900. Otherwise, store CVOL into ICELL and check if CVOL is valid. If not, call ABEND. If it is valid, check if CVOL is 0. If so, go to 100. Otherwise, write the VOL in core onto VOL(CVOL) on data set 10. VOL area on data set 10 starts at record 10.

100 - If NPAGE is 0, go to 200. Otherwise, read VOL(NPAGE) from data set 10 into core.

200 - Go to 900.

SVOL - Write VOL in core out to VOL(NPAGE). Then, go to 200.

RVOL - Read VOL(NPAGE) into core, and go to 200.

XPAGE - If the trace switch (SWT) is off, go to 11010. Otherwise, turn on TRACE.

11010 - Check if NPAGE is valid. ($0 \leq NPAGE \leq 100$). If so, go to 490. Otherwise, go to 5.

490 - If PAGE(NPAGE) is in core, go to 850. Otherwise, store CPAGE into ICELL. Then check if CPAGE is valid. If not, call ABEND. If it is valid, check if CPAGE is 0. If so, go to 500. Otherwise, write PAGE in core onto PAGE(CPAGE) area on data set 8.

500 - If NPAGE is 0, go to 800. Otherwise, read PAGE(NPAGE) into core PAGE area.

800, 850 - Store NPAGE into CPAGE. Then, go to 950.

900 - Store NPAGE into CVOL.

950 - If trace switch (SWT) is off, go to 11021. Otherwise, call TRACE.

11021 - Return to the calling routine.

SPAGE - Write PAGE area in core onto data set 8, PAGE(NPAGE), then go to 800.

RPAGE - Read into PAGE area from PAGE(NPAGE), then go to 800.

CALLING SEQUENCE:
```
CALL RTN($XVOL,NPAGE)
CALL RTN($SVOL,NPAGE)
CALL RTN($RVOL,NPAGE)
CALL RTN($XPAGE,NPAGE)
```

```
CALL RTN($SPAGE,NPAGE)
CALL RTN($RPAGE,NPAGE)
```

where:
NPAGE is the number of the VOL or PAGE to be affected.

OPTIONS: The six entry points above allow six slightly different capabilities. These are:

- XVOL - replace the current VOL in core by VOL(NPAGE) on disk. VOL(CVOL) on disk gets replaced by the VOL in core.
- SVOL - the current VOL in core is saved at VOL(NPAGE).
- RVOL - the current VOL in core is replaced by the contents of VOL(NPAGE).
- XPAGE - replace the PAGE in core with PAGE(NPAGE). PAGE(CPAGE) gets replaced by PAGE in core.
- SPAGE - the current PAGE in core is saved at PAGE(NPAGE).
- RPAGE - the current PAGE is replaced by PAGE(NPAGE).

CALLED BY: Any IOC routine which will swap VOL or PAGE areas.

SPECIAL NOTE: When either XVOL or XPAGE is called with the requested page equal to zero, the result is equivalent to a call to SVOL or SPAGE. The current VOL or PAGE in core is saved on disk and it still remains intact in core.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. XVOL uses NPAGE as input to specify which areas are affected. Also, data set 8 and 10 are used to hold the PAGE and VOL images.

OUTPUTS: Both areas in core and on disk, data set 8 and 10, may be affected by XVOL modules. Note that the PAGE area begins at record 10 on data set 8, and the VOL area begins at record 10 of data set 10.

ENTRY POINTS:

XVOL
RVOL
SVOL
XPAGE
RPAGE
SPAGE

PROGRAM CONSTANTS: None.

PROGRAM VARIABLES:

- ICELL - Cell to hold the CVOL or CPAGE on entering XVOL or XPAGE module.

IREC - The actual record number of the record to be fetched or written.

SWT(6) - Trace switch.

CPAGE - Current PAGE in core.

CVOL - Current VOL in core.

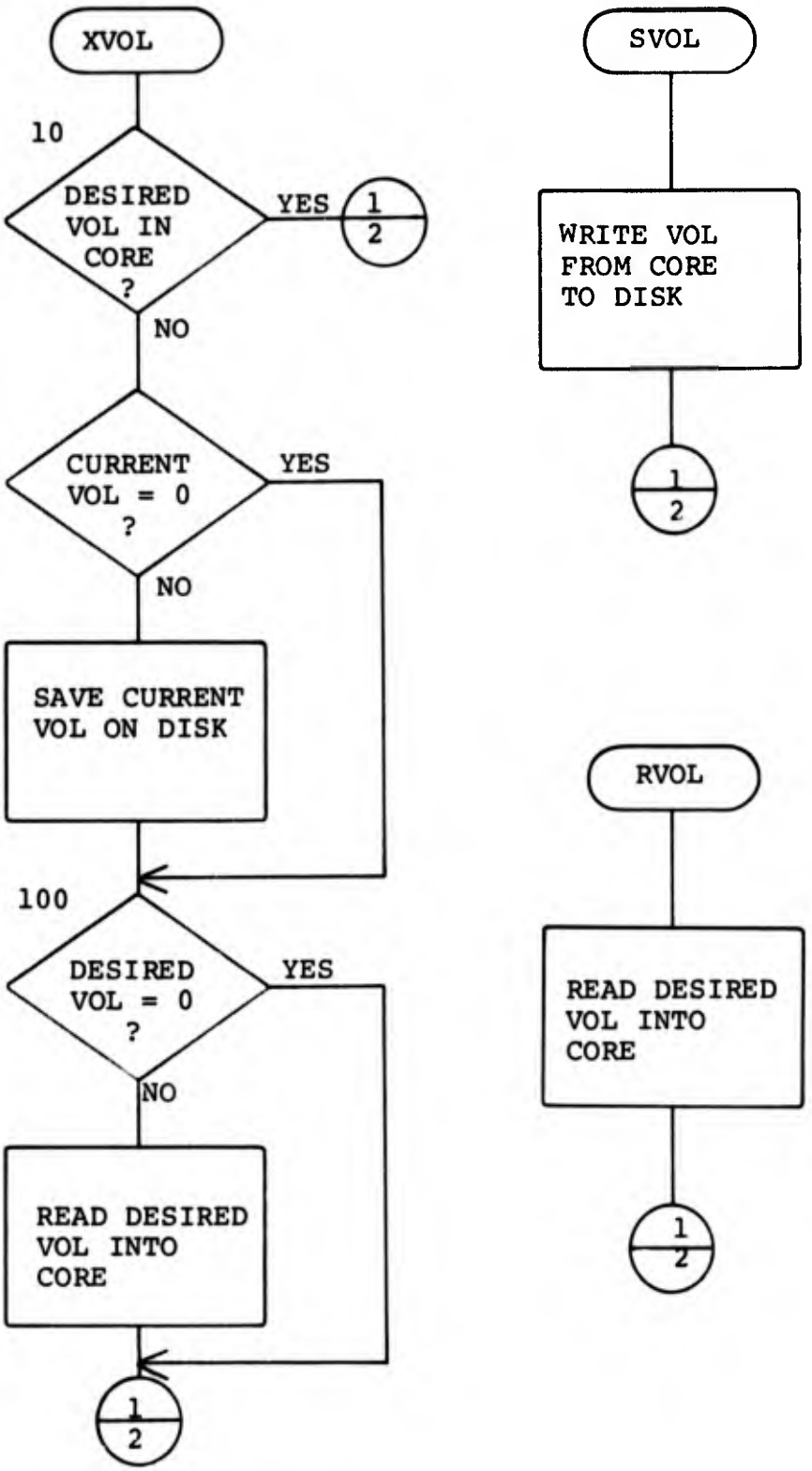
IVOL - The VOL area in core.

IPAGE - The PAGE area in core.

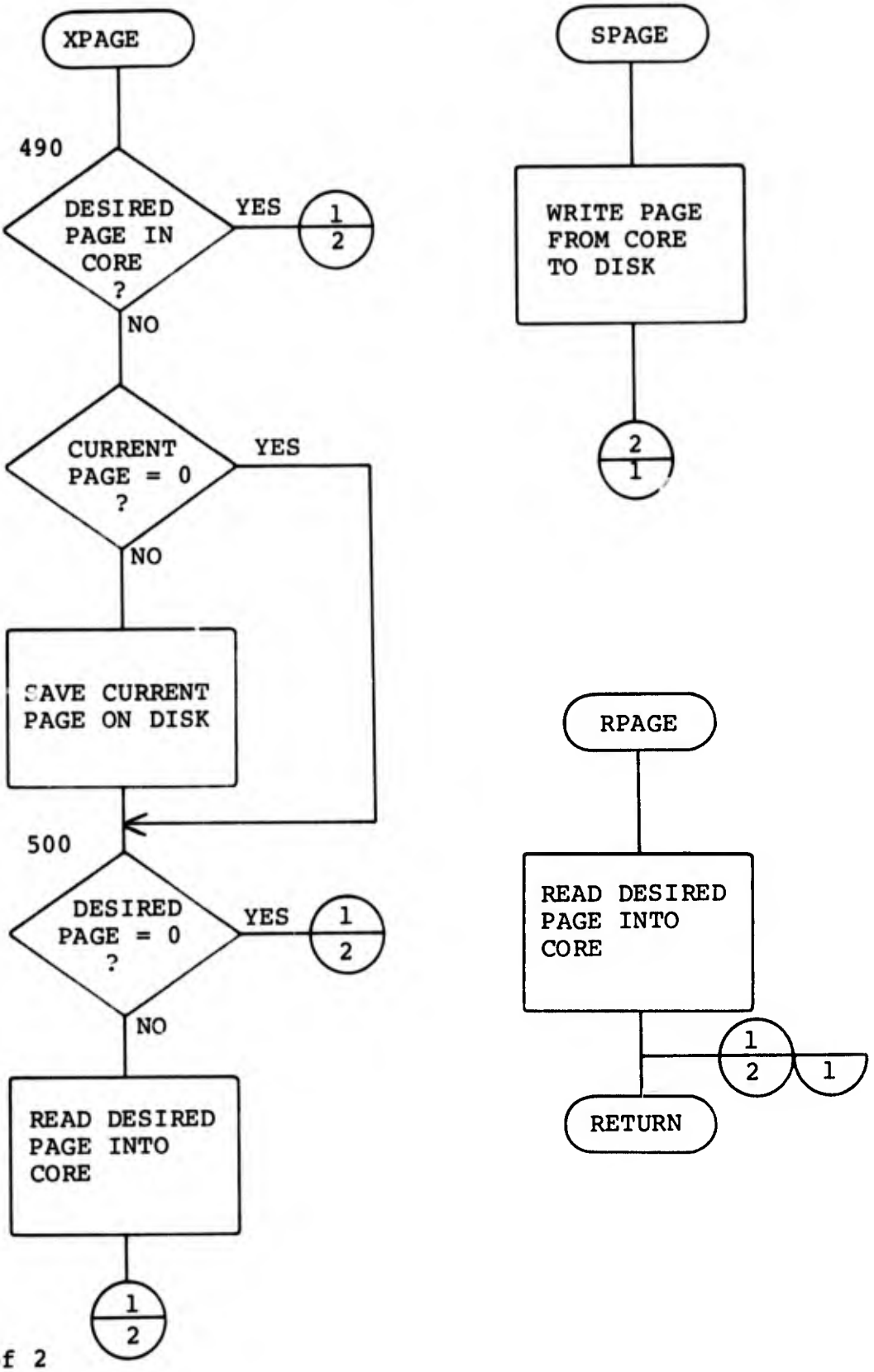
ERROR MESSAGES: None.

SUBROUTINES CALLED: TRACE

XVOL: Logic Diagram



XVOL: Logic Diagram



PROGRAM ID: PREWDSPY
PROGRAM NAME: Preprocessor for Write Display

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: PREWDSPY preprocesses the parameters for WDSPY. It moves the parameters from PARMTBL in COMMON into its own parameter table. It checks the FV number to see that it falls within the range 1-44. It calls WDSPY, then returns to RTN.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The calling sequence to WDSPY may contain up to 5 parameters. Of these parameters the first, the pointer to the display buffer, is the most commonly used. Therefore, the call to WDSPY would be greatly simplified if the other four parameters were optional. Because WDSPY (a FORTRAN program) would not function properly unless all five parameters were present, a preprocessor routine (PREWDSPY) was required in order to provide for any parameters which were missing from the calling sequence.

Any calls to WDSPY by the IOC applications programs will actually result in a call to PREWDSPY which will then call WDSPY. This is accomplished through the entries in the phase table for PREWDSPY and WDSPY.

LOGIC NARRATIVE:

PREWDSPY-STM — Save the contents of the registers. Assume that there is only 1 parameter and initialize other parameters to zeroes. Copy the first parameter into PARMLIST and test for more parameters. If only 1, go to RETURN.

COPY - Copy second and third parameters (together they define the position where the marker is to be activated) and make sure the line number (second parameter) is between 1 and 44.

RETURX - Copy any additional parameters and set the end code in the last parameter entry.

RETURN - Call WDSPY to write the display. Restore the registers and return to the calling program.

CALLING SEQUENCE: CALL RTN($WDSPY,BUFFER,NUMFV,IPOSIT,LINEA,LINEB)

BUFFER is the only parameter that is mandatory. It points to an area in the COMMON/VOL where the screen image is stored.

NUMFV, IPOSIT } Define the position in the screen where the marker is to be activated. They define the line and character, respectively. These parameters are optional.

LINEA - Line number of screen line to be blinked. This parameter is optional.

LINEB - Line number of screen line to be blinked. This parameter is optional.

OPTIONS: Options are specified by the absence or presence of parameters. If any of the optional parameters are specified, then all parameters which appear to its left in the calling sequence must also be specified even if they contain dummy values.

CALLED BY: All IOC applications programs.

SPECIAL NOTE: PREWDSPY is always executed before WDSPY. The phase table for Phase 1006 is set up so that whenever WDSPY is called, control is actually passed to PREWDSPY which then calls WDSPY.

SCREENS USED: None.

INPUTS: See discussion on CALLING SEQUENCE.

OUTPUTS: Re-formatted parameter list for WDSPY. The parameter list contains up to 5 full-word entries. They are:

PARMLIST - Address of display buffer
NUMFV - Line where marker is to be activated
IPOSIT - Column where marker is to be activated
LINEA - Screen line to be blinked
LINEB - Screen line to be blinked

ENTRY POINTS:

PREWDSPY

PROGRAM CONSTANTS:

N0 - Constains constant of 0.

N1 - Contains a constant of 1, used to check for minimum value of NUMFV.

N44 - Contains a constant of 44, used to check for maximum value of NUMFV.

PROGRAM VARIABLES:

NUMFV - Contains screen line number for marker positioning.

IPOSIT - Contains the character position within a line where the marker is to be positioned.

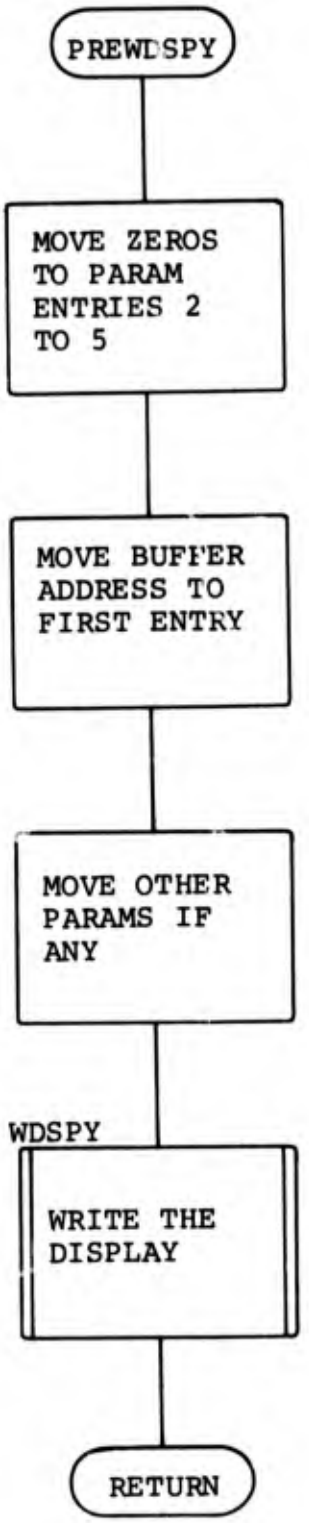
LINEA - Contains the number of the screen line which is to be blinked.

LINEB - Contains the number of the screen line which is to be blinked.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: WDSPY

PREWDSPY: Logic Diagram



PROGRAM ID: WDSPY
PROGRAM NAME: Write Display

Company: Informatics, Inc.
Programmer: Raymond T.Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: WDSPY takes the contents of the user's buffer and writes it to the BR90. It will activate the marker and blink a maximum of 2 lines on the BR90 screen if the user so requests. Entry point RDSPY provides the capability to read the contents of the BR90 screen into the user's buffer.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: With the software facilities provided by the GRAF routines, each applications program would have had to generate its own displays through a series of calls to several GRAF routines. Since most of these calls would be very similar in nature, a routine which would accept a complete display buffer as input and generate the display from that buffer would save a great deal of coding time and eliminate a lot of errors.

WDSPY serves that function. It allows the applications programs to set up an alphanumeric display in its own buffer area and have it sent to the BR90 via one call to WDSPY.

LOGIC NARRATIVE:

WDSPY

Initialize the BR90 screen boundaries. Get the system date and write the date on the top line of the screen. Initialize to scan the display buffer from line 1 through line 44.

DO 100 - Scan a line. If it is non-blank go to step 90. Otherwise, go to step 100.

90 - Write the display line to the BR90 and set an indicator if the marker is to be activated for this line.

100 - If there are anymore lines to inspect, go to DO 100. Activate the marker if it has been requested.

125 - Clear the message line (line 43) with blanks and return to the calling program.

RDSPY

Fill the user's buffer with blanks.

1010 - Read the display lines into the user's buffer area. Return to the calling program.

CALLING SEQUENCE: CALL RTN($WDSPY,BUFFER,NUMFV,IPOSIT,LINEA,LINEB)
CALL RTN($RDSPY,BUFFER)

where:

BUFFER - Address of display buffer
NUMFV - Line where marker is to be activated (Optional)
IPOSIT - Column where marker is to be activated (Optional)
LINEA - Screen line to be blinked (Optional)
LINEB - Screen line to be blinked (Optional)

OPTIONS: Depending on the input parameters to WDSPY, the program will activate the marker in a specified position in the screen and/or blink up to 2 lines on the screen.

CALLED BY: WDSPY is called through PREWDSPY by all IOC applications programs that write displays on the BR90 screen.

RDSPY is called by all IOC applications programs that read the BR90 display screen.

SPECIAL NOTE: When WDSPY is called by an applications program, PREWDSPY actually gets called to preprocess the parameters. PREWDSPY then calls WDSPY.

SCREENS USED: WDSPY will write any screen that the calling program places in the display buffer.

INPUTS: For WDSPY - See CALLING SEQUENCE.
For RDSPY - Contents of BR90 screen.

OUTPUTS: For WDSPY - Moves the contents of the user's buffer to the BR90.

For RDSPY - Moves the contents of the BR90 screen to the user's buffer.

ENTRY POINTS:

WDSPY
RDSPY

PROGRAM CONSTANTS:

BLANK4 - Literal constant of 4 blanks, in COMMON.

PROGRAM VARIABLES:

NUMSAV - Flag to indicate whether the marker is to be activated.
0 = do not activate
non-zero = activate the marker on the display line indicated by the contents of NUMSAV.

J - Subscript which points to the current FV or display line.

JJ - Subscript used in searching the user's buffer for a non-blank line.

I - Subscript used in scanning across a display line.

K - Error flag used in calls to GRAF routines.

LL - Error flag used in calls to $REACT routine.

IBLINK - Flag to indicate whether a display line is to be blinked.
0 = do not blink line
2 = blink line

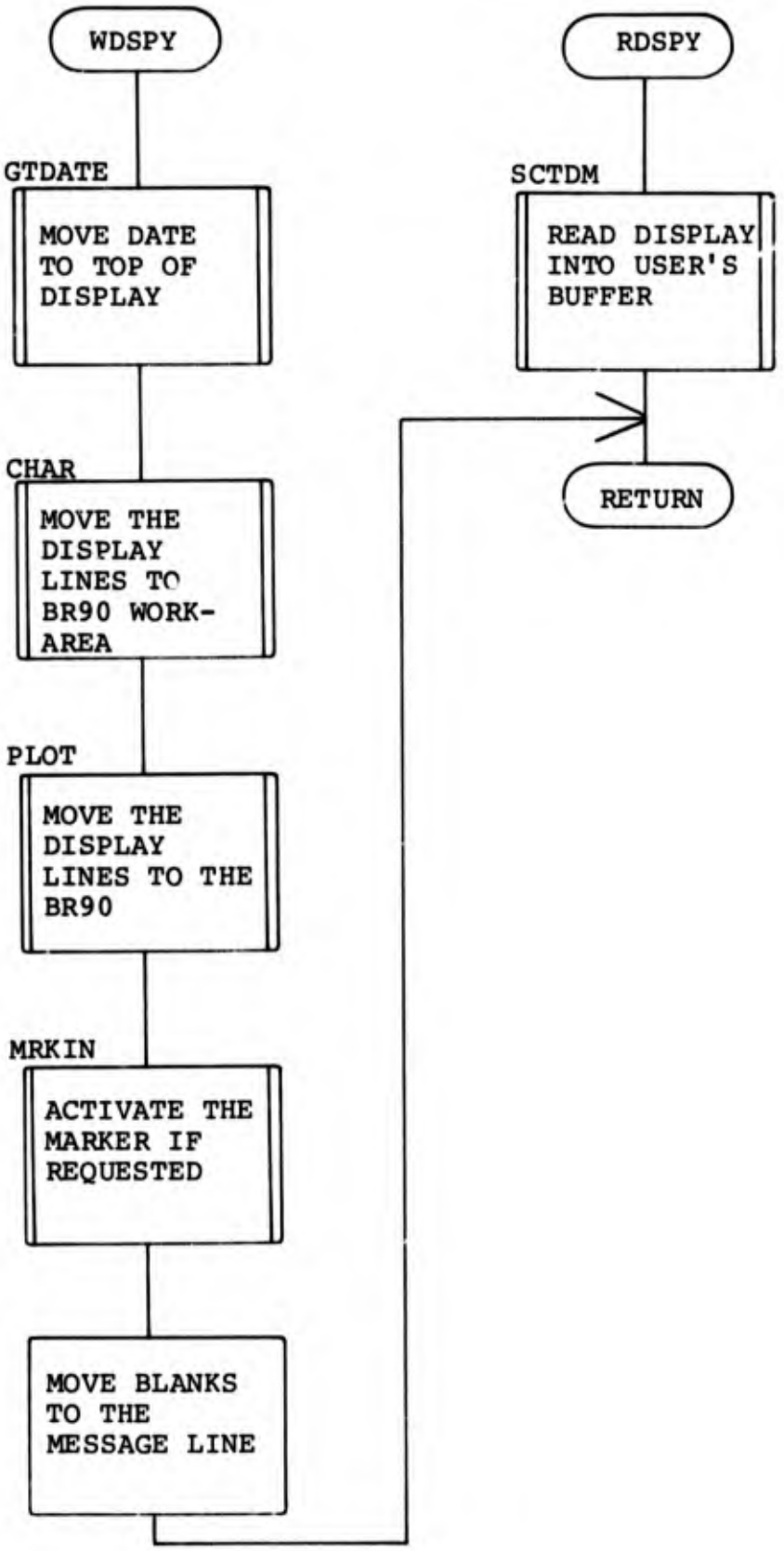
FVLINE - Each bit represents an FV (1-44) or a display line. Located in COMMON.

IMODE - Flag to indicate whether physical I/O is to be done by $SCTDM.
0 = only logical I/O
1 = physical I/O

ERROR CONDITIONS: None.

SUBROUTINES CALLED: ACT, CHAR, DCORD, DINIT, FILL, GETBIT, GTDATE, IIGRAF, IUCORD, MRKIN, PLOT, PUTBIT, REACT, RIGRAF, SCTDM

WDSPY: Logic Diagram



PROGRAM ID: IIGRAF
PROGRAM NAME: Graphic Monitor Program

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: When a graphic routine is called, IIGRAF determines all the phases required to process the call, loads the phases and transfers control to the called subroutine.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The original graphics support package for the BR90, GRAF-90, is a FORTRAN-callable package which could consume large amounts of core. Due to the IOC core constraints, GRAF-90 was revised so that it would occupy no more than 32K of core. In addition, a graphics monitor program was necessary in order to control the loading of the graphic phases. The graphics monitor program, IIGRAF, is called by RTN whenever a graphic routine is requested. IIGRAF determines the routine to be called by its identification number and extracts all of the phases required to process the graphic subroutine call. IIGRAF loads all of the required phases into core before transferring control to the called subroutine.

There are two types of calls to IIGRAF. The first type of call is designed for FORTRAN programs. It is of the form:

```
CALL IIGRAF(RTNID,ARG1,ARG2,...ARGn)
```

(The parameters are explained in the section called CALLING SEQUENCE.) When this calling sequence is used, register 1 will point to a parameter list which will be of the following format:

```
RTNID pointer
ARG1  pointer
ARG2  pointer
.
.
ARGn  pointer
```

The second type of call is designed for Assembly language programs. It is of the form:

```
LA      R1,PARMAD
L       R15,IIGRAF2
BALR    R14,R15
```

(The parameters are explained in the section called CALLING SEQUENCE.) When this calling sequence is used, the programmer is responsible for loading register 1 with a pointer to a parameter table (PARMAD) which has the following format:

```
PARMAD  DC   F'0'   RTNID pointer
        DC   F'0'   Argument list pointer
```

The argument list will be in the following format:

```
ARG1 pointer
ARG2 pointer
 .
 .
ARGn pointer
```

Note that the entry points are different in each of the two types of calls. The two entry points contain code to reconcile the differences in the input parameter formats before control is passed to a common section of code.

LOGIC NARRATIVE:

IIGRAF, RIGRAF - Set R5 to point to the list of argument pointers and set R6 to point to the routine ID. Go to COMMONCD.

IIGRAF2 - Set R5 to point to the list of argument pointers and R6 to point to the routine ID.

COMMONCD - Load R8 with address of NUMLST.

LOOPNUM - If there are any entries remaining in NUMLST go to TESTNUM. Otherwise, the end of NUMLST means that the called routine does not exist so abort the program.

TESTNUM - Compare the routine ID to the NUMLST entry. If they are equal, go to FOUND, otherwise step to next NUMLST entry and go to LOOPNUM.

FOUND - Calculate the relative location in VTABLE where the phase number of the called routine can be found. If the phase number is 0, no load is required so go to DOCALL. If the phase number is the same as the last phase that was loaded, go to DOCALL. If the phase number is between 1 and 4, go to LOADNEW. If it is equal to 5, go to CHK5. If the last phase loaded was greater than or equal to 5, go to LOADNEW. Otherwise, load phase 1505.

LOADNEW - Convert the required phase number from binary to packed decimal then load the required phase into core.

DOCALL - Set R1 to point to the list of argument pointers.

GRAF2BR - If entry point was IIGRAF2, go to NOR1MOD. If there are any input arguments, set Register 1 to point to the list of argument pointers, then go to GOTOIT.

NOR1MOD
GOTOIT - Restore registers and branch to called routine.

CHK5 - If the last phase loaded was greater than or equal to 5, go to DOCALL, otherwise go to LOADNEW.

CALLING SEQUENCE: There are two calling sequences for IIGRAF depending upon whether the call is issued by a FORTRAN program or an Assembly Language program.

FORTRAN Calls: CALL IIGRAF(RTNID,ARG1,ARG2,...ARGn)

where:
RTNID = Name of the called Graphic routine.
ARG1-n = Arguments required by called routine.

Assembly Language Calls:

```
                LA      R1,PARMAD
                L       R15,IIGRAF2
                BALR    R14,R15
                  .
                  .
IIGRAF2         DC      V(IIGRAF2)
PARMAD          DC      F'0'        Current Routine ID Pointer
```

```
        DC      F'0'        Parameter List Pointer
        DC      X'80'
```

OPTIONS: None.

CALLED BY: RTN, SBMAIN, and CKBR90

SCREENS USED: None.

INPUTS: Calling sequence must include a pointer to the routine ID of the called routine and a pointer to the argument list. IIGRAF loads Graphic phases into core from the private core image library.

OUTPUTS: IIGRAF returns data to the calling program if the called subroutine provides such data.

ENTRY POINTS:

IIGRAF - Used in calls by FORTRAN programs.
RIGRAF

IIGRAF2 - Used in calls by Assembly Language programs.

REGISTER USAGE:

R1 - Points to input parameter list.

R5 - Points to first parameter pointer.

R6 - Points to routine number pointer.

R8 - Points to NUMLST.

PROGRAM CONSTANTS:

HALF5 - Constant of 5 in a halfword, used in phase number checks.

NAME - Phase prefix and first 2 digits of phase number. Used to load Graphic phases into core.

CHAR05 - Constant of 5 which is moved to NAME when phase 5 is to be loaded.

NUMLST - List of Graphic Routine IDs. Each entry consists of a 2-byte routine ID. The last routine ID in the list must be immediately followed by a 0 entry. NUMLST is used in conjunction with VTABLE.

VTABLE - Table of VCONs for routine entry points. Each entry consists of 4 bytes. The first byte containing the phase number and the last three bytes containing the routine address. Each entry in VTABLE corresponds to an entry in NUMLST. The entries in VTABLE and NUMLST must appear in the same sequence.

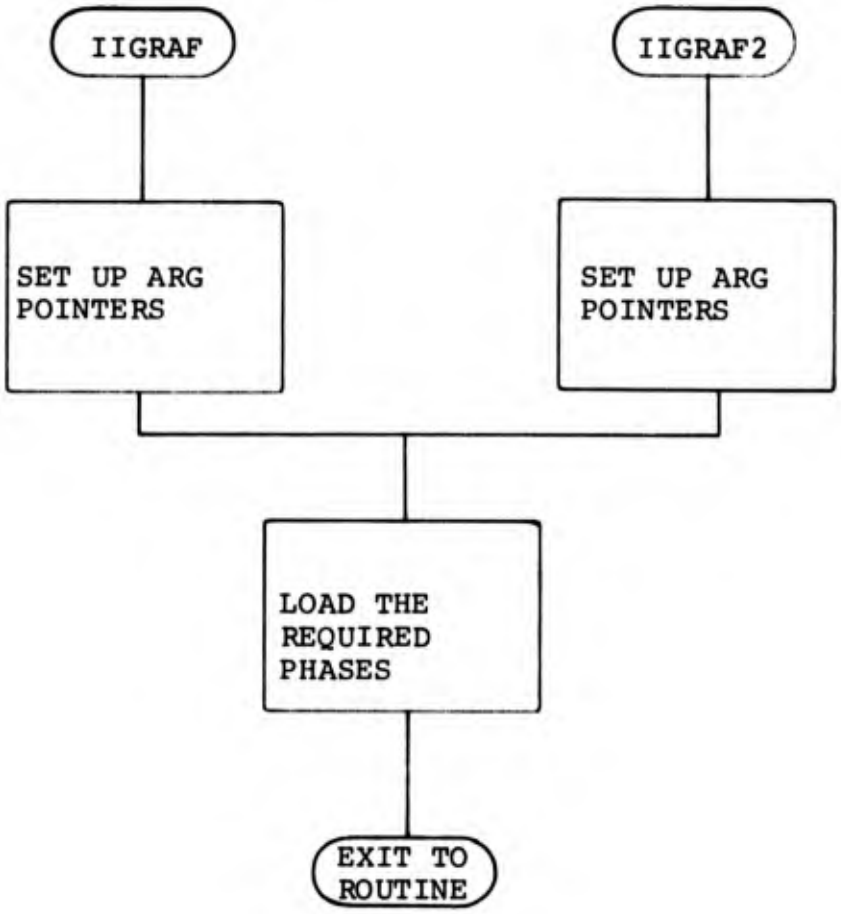
PROGRAM VARIABLES:

WORK - Work area used in converting phase numbers from binary to packed decimal.

ERROR MESSAGES: None.

SUBROUTINES CALLED: IIGRAF calls any of the requested Graphic routines.

IIGRAF: Logic Diagram



PROGRAM ID: FETCH
PROGRAM NAME: Fetch BR90 Screen

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: FETCH accesses the list of IOC screens, and moves the specified screen into the calling routine's area.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: In IOC, a generalized subsystem has been implemented to handle the screens which are used by the on-line programs. In addition to FETCH, which allows any IOC program to access screens, screen maintenance (SCREEN) and generation (IOCGENRC) routines are available.

LOGIC NARRATIVE:

FETCH - Using the input parameter, NUMBER, compute the actual record number on data set 8. Then, do a direct-access read into the output area. Finally, return to the calling routine.

CALLING SEQUENCE: CALL RTN($FETCH,NUMBER,BUFFER)

where:

NUMBER - The number of the screen to be fetched, $1 \leq NUMBER \leq 100$.

BUFFER - A 16x44-word array which will contain the screen image.

OPTIONS: None.

CALLED BY: Any IOC routine which needs one of the screens.

SPECIAL NOTE: FETCH is usually called with an area called BUFFER as an output area. However, in the routine, this array is dimensioned as a 1-dimensional array, in order to speed up the direct-access read.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. FETCH uses NUMBER as input. It also uses data set 8, which contains the 100 screen images generated by GENRC.

OUTPUTS: See CALLING SEQUENCE. FETCH generates a specified screen image in BUFFER.

PROGRAM CONSTANTS: None.

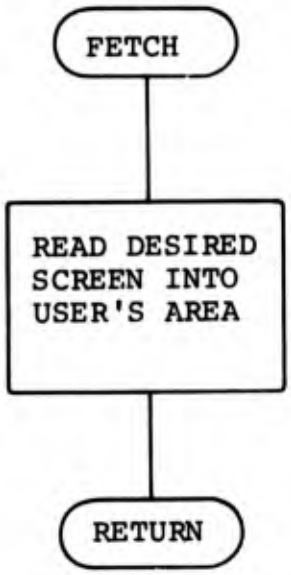
PROGRAM VARIABLES:

IREC - This contains the actual computed record number on data set 8 in which the desired screen is stored. Screens are stored in records 161 to 260 of data set 8.

ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

FETCH: Logic Diagram



PROGRAM ID: FILES
PROGRAM NAME: Generalized File Selection Routine

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This routine provides an interactive sequence which allows the user to select one of the available files for processing. The name of the selected file is returned to the calling program. In addition, a value is returned to indicate which Query key caused termination of the file selection procedure.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Most applications in IOC require some data access from files, and in most cases, the selection of which file is to be used is left to the user. Thus, this interactive procedure is a fairly common one. To facilitate and standardize this procedure, FILES was implemented. FILES allows the user to interactively select one of the workfiles, or one of the available IOC files, and passes the name of the file back to the calling routine.

LOGIC NARRATIVE:

FILESTRT- The File Selection screen is fetched, and the names of the files from which the workfiles were derived are moved to the screen buffer. Lines 1-12 and 41-44 are disabled, and lines 13-40 are enabled. Lines 13-40 will contain the names of the available files.

SHOWIT - The screen buffer is displayed by a call to WDSPY.

WAIT - AWAIT is called. On return from AWAIT, several options are available. If a Query key interrupt occurred, branch to QKEY. Otherwise, the light-penned line is determined. If the line was blank, a switch is set to indicate that a file was not selected, an error message is moved to the screen buffer, and a branch to SHOWIT takes place. Otherwise, the name of the file is moved to the return parameter, and a switch is set to show that a file was selected. Then, go to WAIT.

QKEY - Set the STATUS parameter to show either a 0 if PROCEED key was hit or the number of the Query key that was hit. Then, if it was a PROCEED key, go to GO.

RETURN - Return to the calling routine.

GO - If a file was selected, go to RETURN. Otherwise, set an error message in screen buffer, and go to SHOWIT.

CALLING SEQUENCE: CALL RTN($FILES,NAME,BUFFER,STATUS)

where:

NAME - An area where the name of the selected file is returned.

BUFFER - An area where the screen buffer will be built.

STATUS - An area where the Query key number will be returned.

OPTIONS: None.

CALLED BY: Routines or applications which require a file to be selected.

SCREENS USED: This routine uses screen 43, File Selection screen.

INPUTS: The user must light-pen one of the available files from the File Selection screen.

OUTPUTS: See CALLING SEQUENCE. The variables NAME and STATUS are returned from FILES.

ENTRY POINTS:

FILES
FILESTRT

REGISTER USAGE:

R8	-	Pointer to NAME area
R9	-	Pointer to BUFFER area
R10	-	Pointer to STATUS area
R12	-	Base register

PROGRAM CONSTANTS:

$FETCH	-	Program ID for FETCH routine
$INTON	-	Program ID for INTON routine
$INTOF	-	Program ID for INTOF routine
$WDSPY	-	Program ID for WDSPY routine
$AWAIT	-	Program ID for AWAIT routine
ASSGN	-	Constant of "DERIVED FROM"
LP	-	Constant of 'LP' for call to INTON
QK	-	Constant of 'QK' for call to INTON
K2	-	Constant of 2, used to turn QK2 on and off
K43	-	Constant of 43, used to fetch screen number 43.

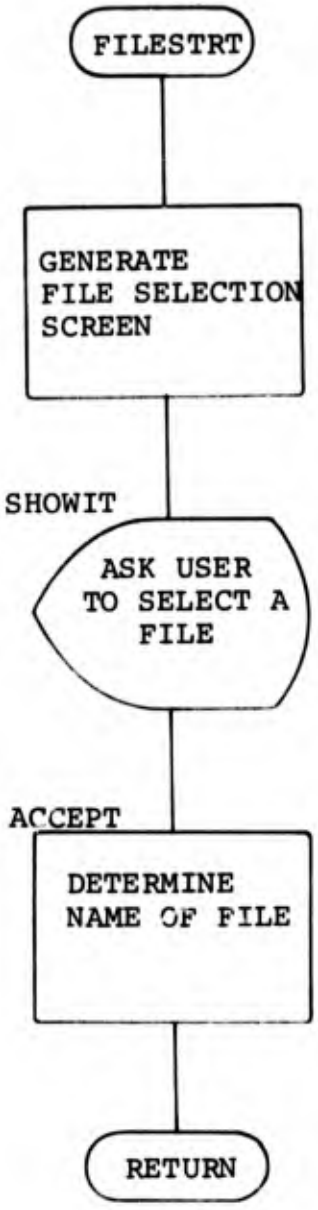
PROGRAM VARIABLES:

TERM+1	-	Switch used to cause branch to turn off only lines 41-44.
GO+1	-	Switch used to indicate if a valid file was selected.
FETCH	-	Calling sequence for RTN used to call FETCH
FETCH2	-	Last parameter in FETCH calling sequence.
LPLIST	-	Calling sequence for RTN to call INTON and INTOF
SCREEN	-	Calling sequence for RTN to call WDSPY
SCREEN2	-	Last parameter in WDSPY calling sequence
LP1,2,3,4	-	Parameters for call to INTON

ERROR CONDITIONS: 'INVALID SELECTION', a blank line was light-penned. 'PLEASE SELECT A FILE BEFORE PROCEEDING', a line was not light-penned before the PROCEED key was hit.

SUBROUTINES CALLED: AWAIT, FETCH, INTOF, INTON, RTN, WDSPY

FILES: Logic Diagram



PROGRAM ID: IIDTFCM
PROGRAM NAME: Independent DTFCM

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This program sets up a DTF for a general MODS file. This is used so that it won't be necessary for every routine which accesses the files to set up its own DTF. It modifies the DTF for a given file name and reads the FFT into a work area.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Several IOC assembly routines access MODS files. In general, to access a MODS file, a DTFCM module for each active file must be present in core. However, in IOC, it is the user who chooses which files are to be accessed, and in most cases, two files are active simultaneously. Rather than keep a DTFCM module for each possible file on disk or in core, it was decided to create the IIDTFCM module. The IIDTFCM module contains two sections: INFILE and FIXDTFCM. INFILE is a "dummy" DTFCM module, and FIXDTFCM contains code which modifies the INFILE module for a given file. Thus, in order to access a DTFCM file, an IIDTFCM module must be in core, and FIXDTFCM must be called. Under this system, only the IIDTFCM modules for those files which are active have to be in core.

LOGIC NARRATIVE:

INFILE - This is the file definition for the general DTFCM file.

FIXDTFCM- This is the entry point in this routine. Here, the file names in the DTFCM module are modified and the FFT is read into core.

TOOLG - The keylength is checked and inserted into the DTFCM module. Then control returns to the calling routine.

CALLING SEQUENCE: CALL FIXDTFCM(NAME,WORKAREA)

```
        B     ERROR
        .
        .
        .
        .
```

OPTIONS: None.

CALLED BY: All IOC programs which access the disk-resident MODS files.

SPECIAL NOTES: FIXDTFCM can only be used to access files on a retrieval basis. Only one file can be used for input at a time, if IIDTFCM is used. If more than one file is needed, the user must define his own DTF.

User must define the following as ENTRY in his program:

KEY - See INPUTS.
ENDFILE - branch location in case of end of file.

In using a file defined by using IIDTFCM, the user first sets the name of the file in the DTF called INFILE by calling FIXDTFCM. Thus, to get a record, he needs:

```
        GET     INFILE,WORKAREA
```

where INFILE is external to his program, and WORKAREA is an area in his program.

The first instruction after the call to FIXDTFCM must be a branch to an error routine. This occurs if the KEY on the file is too long.

SCREENS USED: None.

INPUTS:
NAME - 6-byte name of file to be defined.
KEY - 80-byte field used to hold the key of the record to be found. The user must define this as an ENTRY in his program.

OUTPUTS:
WORKAREA - 1280-byte area into which FFT will be read. This is one of the parameters in the calling sequence.
BYTE - 2-byte field in IIDTFCM which will contain error flags. If it is to be accessed, the user must declare it as an external reference.

ENTRY POINTS:

INFILE
IIDTFCM
FIXDTFCM
BYTE

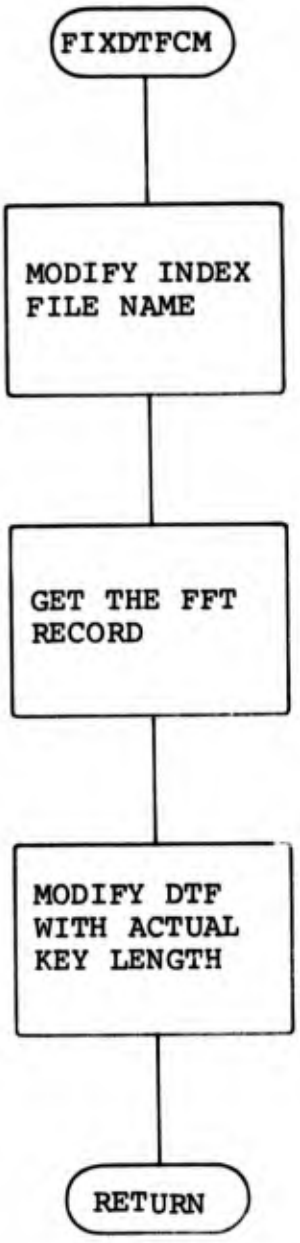
PROGRAM CONSTANTS: None.

PROGRAM VARIABLES: None.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: ENDFILE, IJIFAZZZ

IIDTFCM: Logic Diagram



PROGRAM ID: FFTINT
PROGRAM NAME: Build Screens for FFTSCR Use

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: FFTINT builds pages containing all the fields defined in a file. These pages are used by FFTSCR to display the contents of the FFT. These pages are built on IOC PAGES, with 8 BR90 screen images/page. Each screen consists of a 20-byte header line followed by up to 10-42 byte lines. In addition, FFTINT passes back data to the calling routine in the parameter list.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Some of the IOC applications programs need to display the FFT of a file on the BR90. It would have been cumbersome from both a core usage and program maintenance viewpoint to have each applications program contain the coding to access the FFT, format the data, and display the data. Therefore, two routines, FFTINT and FFTSCR, were developed to take care of these functions.

FFTINT accesses the FFT and FNT of the selected file and prepares IOC pages containing ten lines of FFT information per page. Data from the Supplemental FFT if present, are also included in the IOC pages. These pages are stored on disk and will be used by FFTSCR.

LOGIC NARRATIVE:

FFTINT - FFTINT builds screens containing field names for later use by FFTSCR. It uses IOC pages to store the screens. Each IOC page holds 8 screens. On entry, establish base registers, and save the

current VOL number. Each parameter pointer in the parameter list is stored in the table PARAM. Also, a count is kept of the number of parameters given. VOL 11 is swapped into core as a work area.

MAINLOOP- This is the outermost processing loop. Each pass through this loop processes one of the files given as input. The initial processing in this loop is concerned with first getting the correct FFT into core. Then, using the FFT, field counts for each record type are accumulated in the last byte of each entry in TABLE.

CNTSCRNS- Using the field counts from above, the total screen count is completed. These counts are returned to the user in the parameter fields.

GETFNT - After the total screen count is obtained, check if the file has a FNT. If not, set a switch to later make use of FFT names in place of FNT names.

GETSUP - Move all Supplemental FFT items concerned with current file into SFFTAREA, up to 250 items.

NEXTPAGE- Get the next available IOC page to store screens on. Increment IOC page number and set page offset to store first screen on current page.

CSM1 - Check the number of fields left in the current record type. If any are left, go to CSM2 to process them. If not, set to build screens for next record type. If next record type is the last type, go to FINISHED. Otherwise, go back to CSM1.

CSM2 - If there are more than 10 fields left in the current record type, go to CSM3. Otherwise, set current screen entry count to the actual count in TABLE, set the residual count in TABLE to 0, and go to CSM4.

CSM3 - Set residual count to 10 less, and current screen entry count to 10.

CSM4 - Set pointer to beginning of current screen in IOC page area. Then, fill in header line for screen.

TENLINES- Loop through the FFT, FNT, and Supplemental FFT (if present) area, and process the screen. When finished building the screen, set new page offset to point to the next screen in the page area. If the current page is depleted, go to NEXTPAGE. Otherwise, go to CSM1.

FINISHED- Set next page offset for next file. If not finished with all of the files, go to MAINLOOP. Otherwise, control passes to LASTPASS.

LASTPASS- The original VOL is returned to core.

TEMLOOP - The input parameter areas for each file are built, with each parameter entry having the following format:

Bytes	1-6 :	File name
	7-8 :	HOP of record type code
	9 :	Periodic flag
	10 :	Total screen count for this file
	11-12 :	First IOC page number on which screens for this file are stored.

Then, return to the calling routine.

CALLING SEQUENCE: CALL RTN($FFTIN,FILE1,...,FILE9)

where:
FILEi is a 12-byte field with bytes as follows:

Bytes	1-6 :	File name
	7-8 :	HOP of record type code
	9 :	Periodic sets allowed flag
	10 :	Total number of screen pages for this file
	11-12 :	First IOC page number for the screens.

OPTIONS: None.

CALLED BY: IIOUT, QUERYP, SORTP, STOQRY

SPECIAL NOTE: Currently, only up to 9 parameters in a single call are allowed. Also, a maximum of 250 Supplemental FFT items is allowed per file.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. Bytes 1-6 of each parameter are used as input to FFTINT. In addition, it uses the FFT, FNT, and Supplemental FFT as input.

OUTPUTS: See CALLING SEQUENCE. Bytes 7-12 of each parameter are output from FFTINT. In addition, IOC pages are generated.

ENTRY POINTS:

A1
KEY
ENDFILE
FFTINT

PROGRAM CONSTANTS:

TABLE - A table of 11 4-byte entries. Each entry consists of the following bytes:
First letter of record type FFT names
Last letter of record type FFT names
Record type
Count of number of fields for record type
This table is used by FFTINT to keep track of which fields are being processed.

VOLNUM - Contains 11. Used to call XVOL to get VOL 11 into core.

PROGRAM VARIABLES:

DOUBLE - Area used to unpack numbers to convert them to decimal.

PARAM - Area used to save parameter list. Each of the 10 16-byte entries consist of:
Bytes 1-12: Image of output parameter
13-16: Pointer to input parameter area.

PSAVE - Pointer to current entry in PARAM.

SAVE1 - Number of parameters inputted.

USERSVOL- Entering VOL number.

IOCPAGEN- Next IOC page to be used.

SAVE13 - Area used to save register 13.

CSMCOUNT- Count of number of parameters.

SUPCOUNT- Total number of Supplemental FFT items in SFFTAREA.

TOTSCRNS- Total number of pages for file.

PCKDPAGE- Packed FFTSCR page number, used for header line.

PGOFFSET- Pointer to beginning of page in IOC page buffer.

COMP - Area used to store the current file name's first three letters. Used to compare with SUPFFT ID fields. If input file is a workfile, the COMP will have the first three letters of the origin file's name.

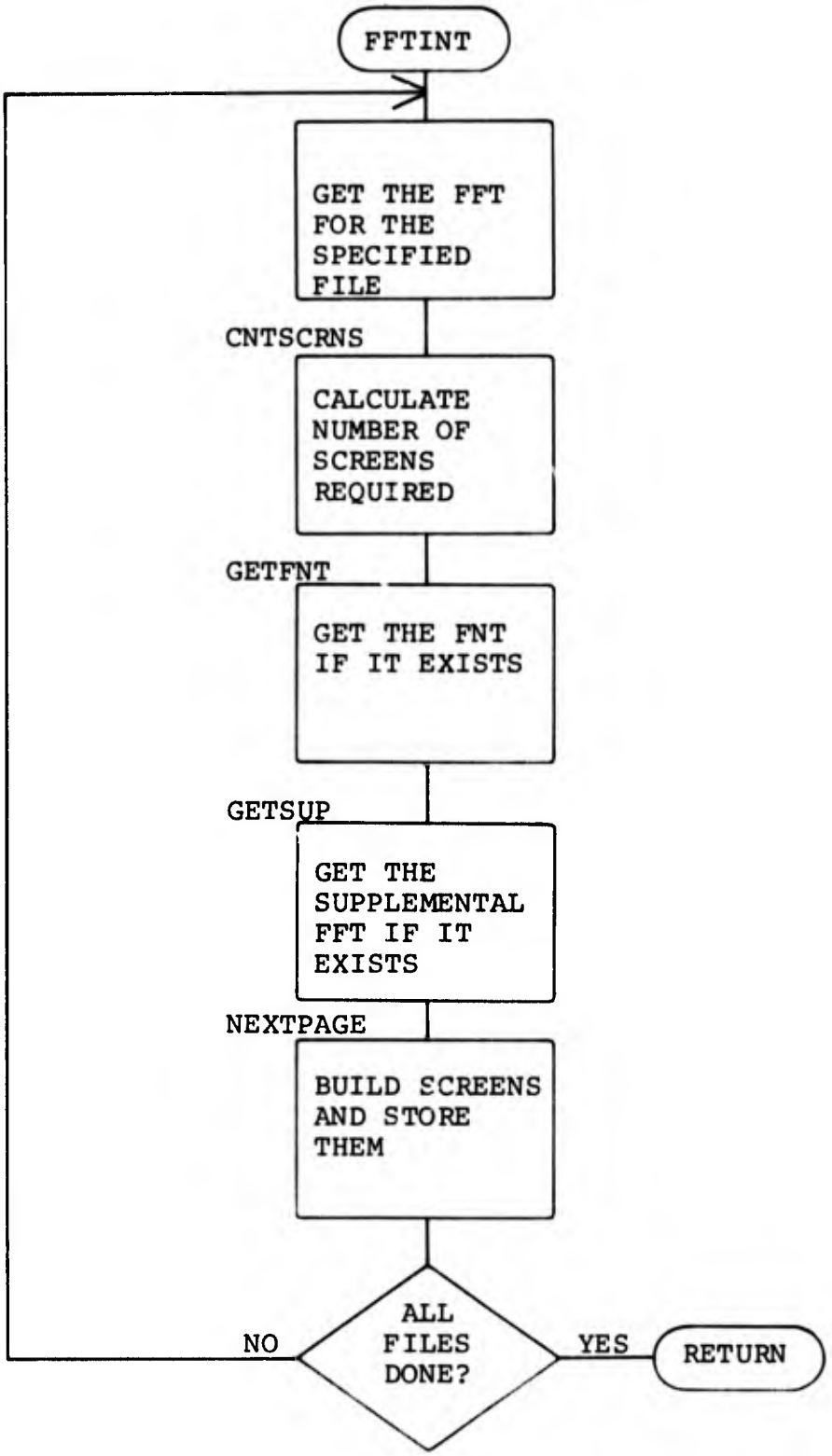
FILENAME- Current file name.

IOCDS - A DSECT for the BR90 screen page image. The following fields are in the DSECT:

PAGENO - Page number of current screen.
TOTPGS - Total number of screens for this file.
FILENM - File name.
SETID - Set identification.
SETMN - Set mnemonic.
FFTID - FFT name for field.
FNTID - FNT name for field.
SUPFFTX - Supplemental FFT item for field.
FFTSTAT - FFT status, HOP, length.
LIGHTGUN- Lightgun allowed flag.

SUBROUTINES CALLED: FIXDTFCM, XPAGE, XVOL

FFTINT: Logic Diagram



PROGRAM ID: FFTSCR
PROGRAM NAME: Create FFT Field Screens

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This program provides BR90 screen pages of 10 data lines/page containing FFT, FNT, and Supplemental FFT information. The data is retrieved from IOC pages which were generated by FFTINT.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The initial discussion on FFT and FNT processing is presented in the description of FFTINT. After that program has formatted and stored the IOC pages on disk, FFTSCR will be called to make these pages available in the calling program's buffer area. Using the number of the requested page, FFTSCR will read that page into core, add header information and move it into the calling program's buffer.

LOGIC NARRATIVE:

FFTSCR - The registers are saved and the base register established. The desired line number is used to calculate the position in the output buffer.

MODULO - The desired page is taken modulo the total number of pages, and then the correct IOC page, along with the offset, is calculated. The correct page is gotten by a call to XPAGE.

PGINCORE- Using data on the page, a header line is built in the buffer. Then, the rest of the buffer is cleared, along with the codes table. If the type

of the current field is fixed, or if periodic type fields are not inhibited from display, or if periodic fields are inhibited, and this field is of the one type which may be displayed, go to BLDSCRN. Otherwise, put out a message stating that the selected page contained invalid fields, and return to the calling program.

BLDSCRN - The buffer is built up with ten lines containing FFT, FNT, and Supplemental FFT information. Each line also contains an indicator as to whether it is a field or a group. The code table is built up of 6-byte entries which are: light-pen flag, type of field, and code information from the calling parameter.

RETURN - The registers are restored, and it then returns to the calling program.

CALLING SEQUENCE: CALL RTN($FFTSC,FILE,PAGE,LINENO,CODES)

where:

FILE - a 12-byte file information block

PAGE - a 2-byte number of the desired page

LINENO - a 2-byte line number in ALPHABUF for page

CODES - a 6 x 11-byte array for codes relating to each entry.

OPTIONS: None.

CALLED BY: IFSEL, QUERY, SORTP

SPECIAL NOTE: FFTINT must be called to build the IOC pages before FFTSCR can execute properly.

SCREENS USED: FFTSCR does not write any screens. However, it moves the IOC Field Selection screen into the user's buffer area.

INPUTS: IOC pages built by FFTINT and the items described in the calling sequence (FILE,PAGE,LINENO).

OUTPUTS: The CODES table is output. Also, the desired IOC Field Selection screen is built in the buffer area.

ENTRY POINTS:

A2
FFTSCR

PROGRAM CONSTANTS:

MASK - Header for IOC Field Selection screen

$XPAGE - Routine ID for call to XPAGE

PROGRAM VARIABLES:

CPAGE - 4-byte item in COMMON containing the current page number

SAVE13 - Area for saving the contents of Register 13

IOCPAGEN- This variable will hold the current IOC page number, and is used in the call to XPAGE.

MYSAVE - Save area for FFTSCR

FILEBLOK- A DSECT for the FILE input parameters. The fields are as follows:

F1	- 6 bytes for file name
F2	- Record type location
PSETID	- Code to show which periodic set is allowed
SCRNCNT	- Total number of screen pages for this file
IOCPAGE	- First IOC page for this file.

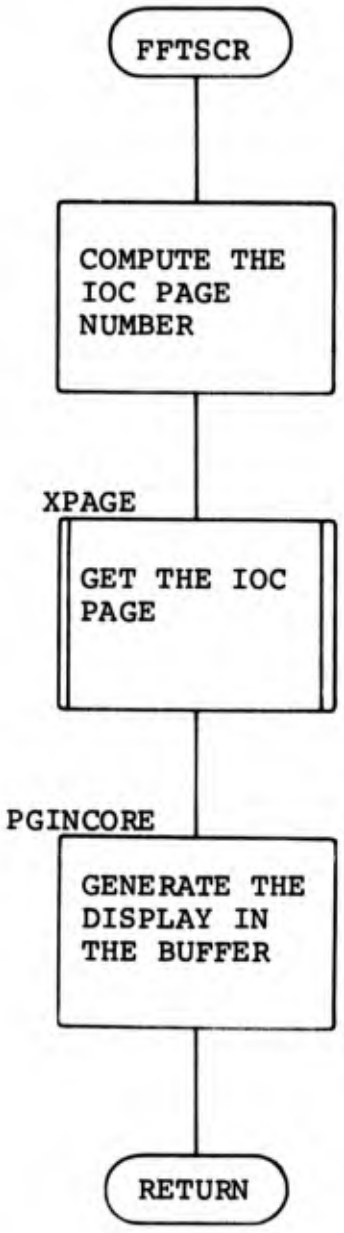
IOCDS - A DSECT for the entries on screen pages generated by FFTINT

PAGENO	- Current BR90 page number
TOTPGS	- Total number of BR90 pages for this file
FILENM	- File name
SETID	- Set identification
SETMN	- Set identification mnemonic
FFTID	- The FFT name for field
FNTID	- The FNT name for field
SUPFFT	- The Supplemental FFT information for this field
CODES	- Codes for this field
LIGHTGUN	- Light-gun allowed flag.

ERROR CONDITIONS: 'PERIODIC SET X HAS ALREADY BEEN ACTIVATED. NO SELECTIONS FROM THIS SET ARE ALLOWED AT THIS TIME.' A field from a periodic set which is not allowed is contained on the selected screen page.

SUBROUTINES CALLED: XPAGE

A2: Logic Diagram



PROGRAM ID: GETWK
PROGRAM NAME: Get Workfile Status

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: GETWK allows IOC programs to retrieve the current information on the workfile status.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The workfiles in IOC have been designed to be retained from one processing session to the next. In order to do this, the workfile status data is stored in the Workfile Status Table which is maintained on disk. During IOC initialization, this data is read into an area called IOCWRK in COMMON so that the data becomes available to all IOC routines. The Workfile Status Table in core and on disk is updated each time the status of a workfile is changed.

LOGIC NARRATIVE:

GETWK - Save the original registers. Then, open WKTABLE. When this is done, read the contents of WKTABLE into the IOCWRK area. Then, return to the calling routine.

CALLING SEQUENCE: CALL GETWK

OPTIONS: None.

CALLED BY: TSKID

SCREENS USED: None.

INPUTS: GETWK uses the disk file, WKTABLE, as input. This area contains the current workfile status information.

OUTPUTS: The contents of the WKTABLE are read into the IOCWRK area in COMMON. Each IOCWRK entry is seven bytes long. The first six bytes contain the file name from which the file was derived. The last byte contains the alphabetic suffix of the workfile.

ENTRY POINT:

GETWK

PROGRAM CONSTANTS:

CSM - CCW for the seek to do the WKTABLE read.

PROGRAM VARIABLES:

SEEK - Area for seek address.

CYLHEAD - Cylinder and head number for WKTABLE.

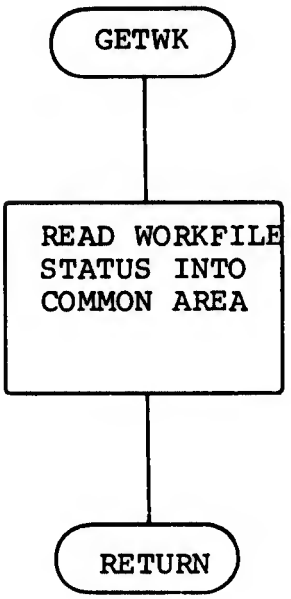
RECNUM - Record number for WKTABLE.

WKTABLE - The DTFPH module for the input file.

ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

GETWK: Logic Diagram



PROGRAM ID: SETWRK
PROGRAM NAME: Set Workfile Names in Screen

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: SETWRK checks the contents of IOCWRK in COMMON. For each non-blank entry, it moves "DERIVED FROM" and the name of the file from which the workfile was derived, into the display buffer.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The File Selection screen is defined with the names of the standard data files, the five workfiles, and the Test and Development files. The workfiles are initially assumed to be unassigned and the literal constant "UNASSIGNED" appears after each workfile name. When a workfile is created (i.e. when it is filled with data records) the entry for the selected workfile is modified with the literal constant "DERIVED FROM" and the name of the file from which it was derived. Subsequent to the creation of a workfile, each time the File Selection screen is called, the entry for that workfile (and all other workfiles which have been created) must be modified to indicate what file it was derived from. The subroutine SETWRK was developed to provide this simple, routine function.

LOGIC NARRATIVE:

SETWRK - Save contents of registers and set buffer pointer to first character of line 14. Branch to BASE.

SETWK2 - Save contents of registers and set buffer pointer to first character of line 5.

BASE - Set Base register.

IOC - Get address of IOCWRK.

F1 - If workfile is active, move the literal "DERIVED FROM" and the name of the file into the buffer.

F2 - Step to next entry and branch to F1 until all workfiles have been processed. Then return to the calling program.

CALLING SEQUENCES: CALL SETWRK
CALL SETWK2

OPTIONS: None.

CALLED BY: IIOUT

SPECIAL NOTE: SETWRK must be included in the same phase with the calling program.

SCREENS USED: This program uses screen 43, the File Selection screen.

INPUTS: SETWRK expects the contents of IOCWRK in COMMON to contain the status of the workfiles. It also expects the File Selection screen to be in the area called ALPHABUF.

OUTPUTS: The output from SETWRK is the updated File Selection screen.

ENTRY POINTS:

SETWRK - moves workfile data into line 14 of the screen.

SETWK2 - moves workfile data into line 5 of the screen.

REGISTER USAGE:

R2 - Pointer to the area in the buffer where the next line of workfile data is to be moved.

R1 - Pointer to workfile entry in IOCWRK.

PROGRAM CONSTANTS:

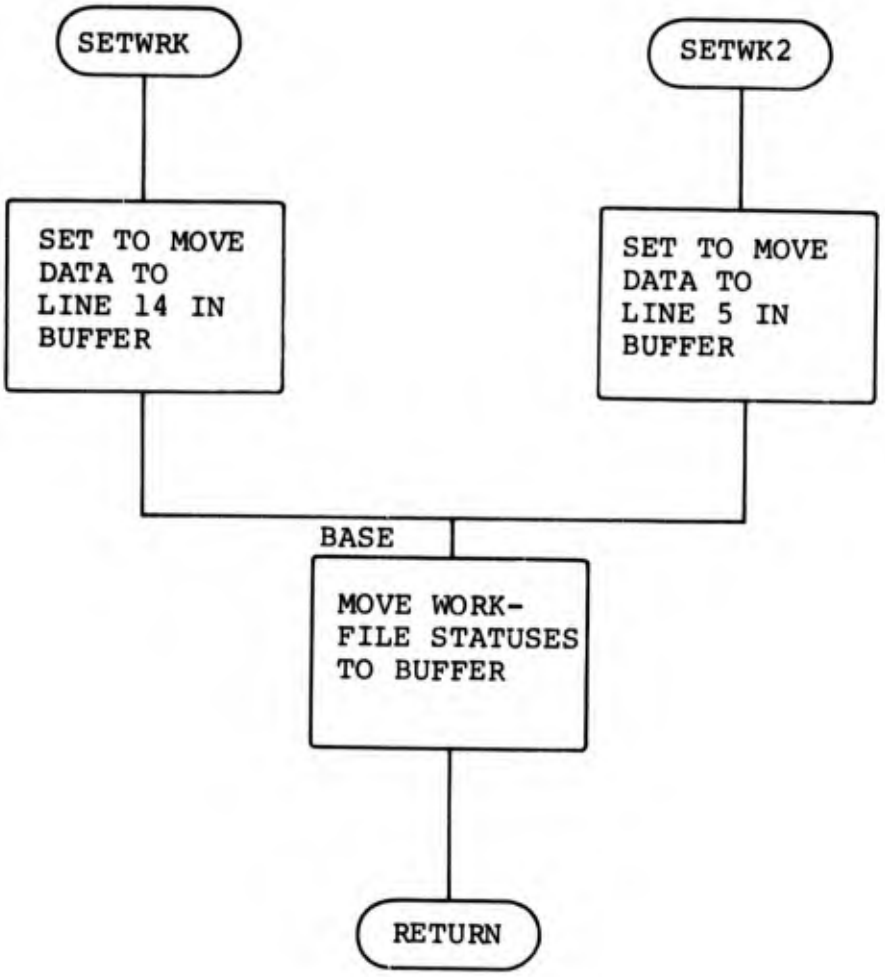
ASSGN - Literal constant of "DERIVED FROM".

PROGRAM VARIABLES: None.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

SETWRK: Logic Diagram



PROGRAM ID: SEQOPN
PROGRAM NAME: Build Pointer File

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: SEQOPN builds a pointer file for the file whose name is in FILENAME. SEQOPN sequentially accesses the records in the file and builds an entry for each fixed record, variable record, and the first subset of a periodic set. These entries are written on the ADDRESS file in 3516-byte blocks.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: After querying a file and placing the results in a workfile, a user may later wish to process all of the records in the workfile without having to execute another query. This routine allows him to go directly from the workfile to output processing by building a pointer file directly from the workfile (or from any other file in the system).

SEQOPN will sequentially access each record of the selected file and build an ADDRESS file entry for each fixed record, variable record, and the first subset of a periodic set.

LOGIC NARRATIVE:

SEQOPN - Program saves registers. File name gets moved from parameter into RTVFIL in COMMON/COMMON/. Also, a 6-byte name is moved to FILENAME. The ADDRESX file name is modified with an 'S'. FIXDTFCM is called to read the input file FFT and modify the output file DTF. The record type location is computed and stored in MOVE+5. The ADDRESS file is then opened.

RTRV - Get an input record. If it is a fixed record, up record count, and go to INCLUDE. If it is a remarks record, go to INCLUDE. If it is the same type as the last periodic set, go to RTRV. Otherwise, go to INCLUDE.

INCLUDE - Move RECTYPE, SEQ, and compressed disk address to OUTBLK. If the size of OUTBLK is not exceeded, go to RTRV. Otherwise, PUT the OUTBLK, and check if the number of blocks is greater than 18, then check if an error flag is needed.

ENDFILE - Put '************' into OUTBLK to indicate end of entries. PUT out OUTBLK, close ADDRESX, INFILE, and then return to the calling program.

CALLING SEQUENCE: CALL RTN($SEQOP,FILEBL,IFLAG)

where:

FILEBL - 12-byte field containing 6-byte file name.

IFLAG - Optional parameter. This is an error indicator which is set to 1 if the file from which the pointer file is being generated contains more than 10530 records, thus exceeding the extents for the pointer file.

OPTIONS: None.

CALLED BY: IIOUT

SPECIAL NOTES: SEQOPN makes use of ADFILE2, which consists of the DTF for ADDRESX, which is an output file. However, the DLBL for the address file is ADDRESS. Thus, in the initial code of SEQOPN, an 'S' is stored in the DTF for ADDRESX so that when ADDRESX is opened, the file named ADDRESS will be opened. The reason this was done was in order that both DTF's could be used for the same file (input,output). Another possible way to do this would be to define another DLBL for the same area. e.g. // DLBL ADDRESX,'ADDRESS FILE'.

SCREENS USED: None.

INPUTS: SEQOPN reads the file pointed to in FILEBL of the calling sequence.

OUTPUTS: SEQOPN creates an ADDRESS pointer file and sets IFLAG in the calling sequence if an error was detected during processing.

ENTRY POINTS:

SEQOPN
KEY
ENDFILE
ADBUFCON

REGISTER USAGE:

R5	-	Count of A records
R7	-	Pointer to OUTBLK position
R12	-	Base register
R3,R15, R14,R1	-	Work registers
R8	-	Points to temporary area for RTRVX macro to store compressed disk address.

PROGRAM CONSTANTS: None.

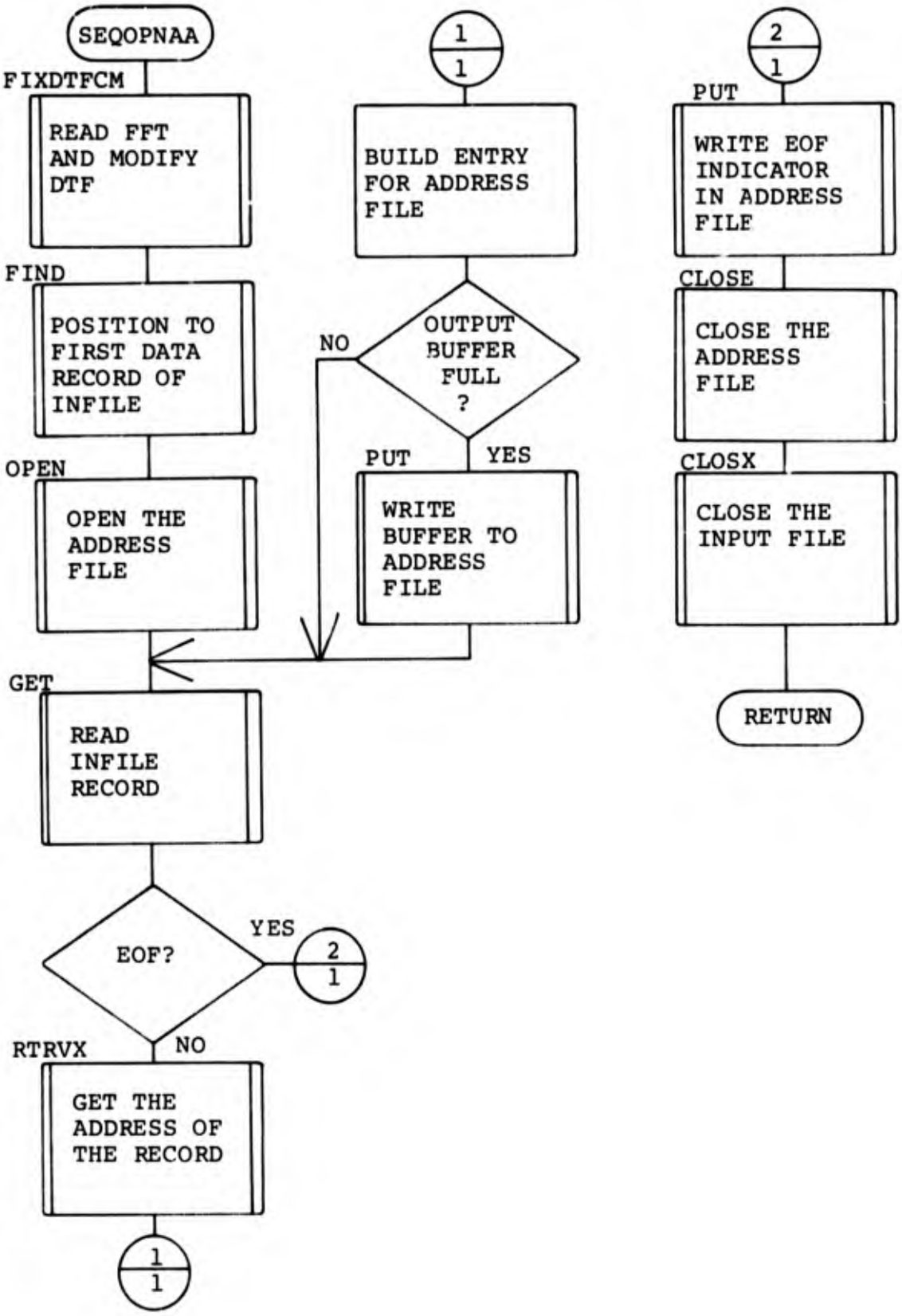
PROGRAM VARIABLES:

MOVE+5	-	Contains the record type location with respect to the beginning of MYAREA.
ID	-	Contains type code of the record.
SEQ	-	Sequence number or periodic set number.
TRACK	-	Contains the compressed address of the record.
FILENAME	-	Will contain input file name.
SAVE13	-	Contents of Register 13.
SAVEAREA	-	Register SAVE AREA.
MYAREA	-	Buffer for INFILE.
OUTBLK	-	Buffer for ADDRESX file.
ADBUFCON	-	First two words of OUTBLK, used for IOCS length count.

ERROR CONDITIONS: SEQOPN returns an error code of 1 in IFLAG if the size of the ADDRESS file is exceeded.

SUBROUTINES AND MACROS CALLED: FIXDTFCM, RTRVX, CLOSX

SEQOPN: Logic Diagram



PROGRAM ID: DRECGT
PROGRAM NAME: Retrieval Using Address File

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: The four routines included in DRECGT provide the FORTRAN programmer with the capability to access any IOC file.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Disk access methods under FORTRAN are inconvenient to use. Thus, when it is necessary to access data on IOC files, programs written in FORTRAN must interface with the files via some assembly routines. The purpose of the DRECGT module is to give the FORTRAN programmer a generalized disk access facility, which could be easily used. Thus, both input to and output from DRECGT is in a form which is fairly standard to FORTRAN programs. This serves two purposes:

1) It eases the FORTRAN programmer's task.
2) It makes the file access method standard, thus reducing the possibility of errors.

LOGIC NARRATIVE:

DRECGT - This routine will read records from a file, using pointers from the address file, and, using information from TABLE, will move selected data fields to a display buffer. Here, the registers are initialized.

LOOP:FXED-MAXILOOP - Loop through the pointer file until a fixed record is found. If no fixed record is found before the end of the address file, go to ENDADDRS. Otherwise, if the fixed record addresses are desired, return it in the user's area, and set count to look for

all 11 record types that are in TYPETBL.

TABLSCAN- Scan the input TABLE to determine which types of records are desired:
If fixed only, set count to look for only 1 type in TYPETBL, then go to LOOPTYPE.
If fixed and periodic, go to LOOPTYPE.
If periodic only, go to TRYNEXT to look for next type in TYPETBL. Also set switch at FOUNDTYP so that the program will skip the current record.

LOOPTYPE- Loop through TABLE to process each entry with a field of the same type as the current TYPETBL entry. If the end of TABLE is encountered, go to TRYNEXT. When identical entries are found, go to FOUNDTYP.

FOUNDTYP- If not first time through, go to COMPTYPE to process record. Otherwise, step to next entry in the address file.

CHK3516 - If the current record is a fixed record, the rest of the fields in the current site will have asterisks in them, since all periodic records for this site have been processed. This is done by branching to ASTRREST.

COMPTYPE- Set the switch at FOUNDTYP so that no skip will take place on the next pass. Compare the current TYPETBL entry with the address file entry type:
If TYPETBL entry is less than the address file entry type, put asterisks in all the fields of this type, since all records in the current site of this type have been processed. This is done by branching to ASTERTYP.
If TYPETBL entry is equal to address file entry type, go to CHKPEROP.
If TYPETBL entry is greater than the address file entry type, go to ASTERREST; since current pointer is pointing to data for the next site.

NEXTPTR - Step to next entry in the pointer file, and go to CHK3516.

ASTERTYP- Set up branches to put asterisks in rest of fields of current type, and go to FIELDLOP.

CHKPEROP- If the current pointer file entry record type is an 'A' or 'Z', or if PEROPTN was set for all periodics, go to PROCESS. Otherwise, check the PTRSETNM field of the pointer file entry to see if this record qualified. If not, go to NEXTPTR to

look at the next record. If it did qualify, go to PROCESS.

PROCESS - Get the record pointed to by the current entry in the pointer file. Set the switch at ASTERBR to move data.

FIELDLOP- Loop through TABLE, and find all entries which are of the same record type as the current pointer file entry. Whenever one is found, go to DOMOVE. When the end of TABLE is hit, go to TRYNEXT to look at the next record type in TYPETBL.

TRYNEXT - Reset the ASTERBR switch to move data, in case it was previously set to put asterisks in the rest of the fields. Set TYPETBL to look at the next record type.

TRYNEXT2- If required to put asterisks in the rest of the fields, go to ASTERAL2. This will set ASTERBR back to move asterisks.

LOOPEND2- This is the end of the loop for each type in TYPETBL. If all types have not been processed, go to LOOPTYPE. Otherwise, check if the maximum number of sites has been processed. If not, go to MAXILOOP.

STORE - Return the number of sites processed, then go to EXIT.

DOMOVE - The actual move is done from the record to the screen buffer. First, the name of the field is moved, if desired.

ASTERBR - A branch to ASTERFLD takes place if the field is to be filled with asterisks. Otherwise, the field is processed for LAT/LONG, 9's complement, or date conversion, and then it is moved to the screen buffer.

ENDMOVE - Step up to next entry in TABLE, then go to FIELDLOP to process it.

ASTRREST- The ASTERBR switch is set to cause asterisks to be put in all fields of the current type, and the TRYNEXT2 switch is set to cause a branch to ASTERAL2 to reset the ASTERBR switch that way for the rest of the types in TYPETBL. Then, go to FIELDLOP to process the remaining fields.

ASTERAL2- The ASTERBR switch is set to move asterisks, then go to LOOPEND2.

ASTERFLD- The actual moving of the asterisks into the screen buffer takes place, then a branch is done to ENDMOVE.

DRECOP - The DRECGT and files are initialized. Various switches and parameters are set in the DRECGT module. The four parameters FILENAME, NAMEFLAG, PEROPTN, and ADDROPTN are saved for use by DRECGT. Line spacing for later display, and maximum sites/call is determined. If this is the initial call to DRECOP, FIXDTFCM is called to open the input file, and the address file is opened. Then, go to EXIT. If the files are already open, go to EXIT.

DRECCL - If the input and address files have not been opened, go to EXIT. Otherwise, close the files, then go to EXIT.

DRECRN - This routine will get one fixed record whose address is passed to it in the last parameter, and will process this record. DRECRN will process only fixed records.

EXIT - Return to the calling routine.

ENDADDRS- When an end-of-file is encountered on the address file, control passes to ENDADDRS. Here, a branch to STORE takes place.

GETNEWPT- The address file buffer is filled with the next set of pointers from the file. Then, it returns to the calling area.

CALLING SEQUENCE: CALL RTN($DRCOP,FILENM,NAMEXX,IPEROP,IRECOP)

where:

FILENM is an 8-character entry with the 6-byte file name and the last two bytes blank.

NAMEXX = 1, if FNT names are not wanted
0, if otherwise

IPEROP = 1, if first qualifying record of type is wanted
0, if first record of type is wanted

IRECOP = 1, if fixed record address is not wanted
0, if fixed record address is wanted.

```
CALL RTN($DRCGT,AREA,IFLAG,TABADD,RECADD)
```

where:

AREA is the beginning position of area where data from the input file will go.
IFLAG is the count of the number of sites returned.
TABADD is the TABLE of field information.
RECADD is the area where fixed record addresses will be returned, if requested.

```
CALL RTN($DRCCL)

CALL RTN($DRCRN,AREA,IFLAG,TABADD,RECADD)
```

where:

AREA is the beginning position of area where data from the input file will go.
IFLAG is the count of the number of sites returned.
TABADD is the TABLE of field information.
RECADD contains disk address of fixed record to be processed.

OPTIONS: These routines allow the programmer to access IOC files sequentially by using DRECGT, or to access fixed records randomly by using DRECRN.

CALLED BY: APLC1, IILIST, IIPLOT, IIRANG, IISUM

SCREENS USED: None.

INPUTS:

DRECOP - See CALLING SEQUENCE. FILENM, NAMEXX, IPEROP, and IRECOP are inputs.

DRECGT - See CALLING SEQUENCE. TABADD is input. Also, DRECGT uses records from the input and address files as input.

DRECCL - No input.

DRECRN - See CALLING SEQUENCE. TABADD and RECADD are inputs. Also, DRECRN uses the input file as input.

OUTPUTS:

DRECOP - No outputs.

DRECGT - See CALLING SEQUENCE. DRECGT generates the desired fields in AREA, a table of fixed record addresses in RECADD, and a count of sites retrieved in IFLAG.

DRECCL - No outputs.

DRECRN - See CALLING SEQUENCE. DRECRN generates the desired fields from the specified fixed record in AREA, and a count in IFLAG.

ENTRY POINTS:

DRECOP
DRECGT
DRECRN
DRECCL
ENDFILE
KEY

REGISTER USAGE:

DRECOP:

R2	-	FILENM parameter pointer.
R3	-	NAMEXX parameter pointer.
R4	-	IPEROP parameter pointer.
R5	-	IRECOP parameter pointer.
R12	-	Used to reference data in DRECGT module.
R15	-	Base register.

DRECCL:

R12	-	Used to reference data in the DRECGT module.
R15	-	Base register.

DRECRN:

R2	-	AREA parameter pointer (Set by DRECGT).
R5	-	RECADD parameter pointer (Set by DRECGT).
R12	-	Used to reference data in the DRECGT module.
R15	-	Base register.

DRECGT:

R2	-	Pointer to current entry in TABLE.
R4	-	Pointer to current type in TYPETBL.
R6	-	Pointer to current entry in address file.
R10	-	Count of number of record types to be processed.
R11	-	Count of maximum number of sites to be fetched.
R12	-	Base register.

PROGRAM CONSTANTS:

TYPETBL - Contains a string of all of the record types. 'AJKLMNOPQRZ'. This is used to keep track of which record type is being processed.

NINE99 - Area containing 999, used for trace code.

EIGHT88 - Area containing 888, used for trace code.

ENDBUFAD- Pointer to the end of the address file buffer.

PROGRAM VARIABLES:

SAVE1 - Area used to save the contents of R1 during execution of GETNEWPT.

PARAMS - Area for storage of 4 parameters to DRECGT.

DATALOC - Area for storage of 1st parameter to DRECGT.

IFLAG - Area for storage of 2nd parameter to DRECGT.

TABLEADD- Area for storage of 3rd parameter to DRECGT.

ADDRPTR - Area for storage of 4th parameter to DRECGT.

ANYFIXED- Switch indicating whether there were any fixed fields in TABLE.

ANYNONFX- Switch indicating whether there were any periodic fields in TABLE.

PEROPTON- Switch indicating whether 1st periodic or 1st qualifying periodic is wanted.

ADDROPTN- Switch indicating whether fixed record addresses are wanted or not.

NAMEFLAG- Switch indicating whether field names are wanted or not.

SPACING - Contains spacing for output groups.

MAXTOGET- Contains maximum number of sites/fetch.

RECTYPLC- Contains pointer to record type indicator in WORKAREA.

SAVEAREA- Register save area for DREC.

COORDS - Work area for LAT/LONG conversion.

RESULT - Area for LAT/LONG conversion results.

POINTER - Pointer to area used in call to COMPNINI.

LENGTH - Pointer to area used in call to COMPNINI.

WORKCELL- Used to compute data item position in WORKAREA.

POINTBX - Pointer to next entry in address file.

FILENAME- Area to hold file name.

RECADD - Area for 4-byte compressed disk address.

KEY - Area for MODS key.

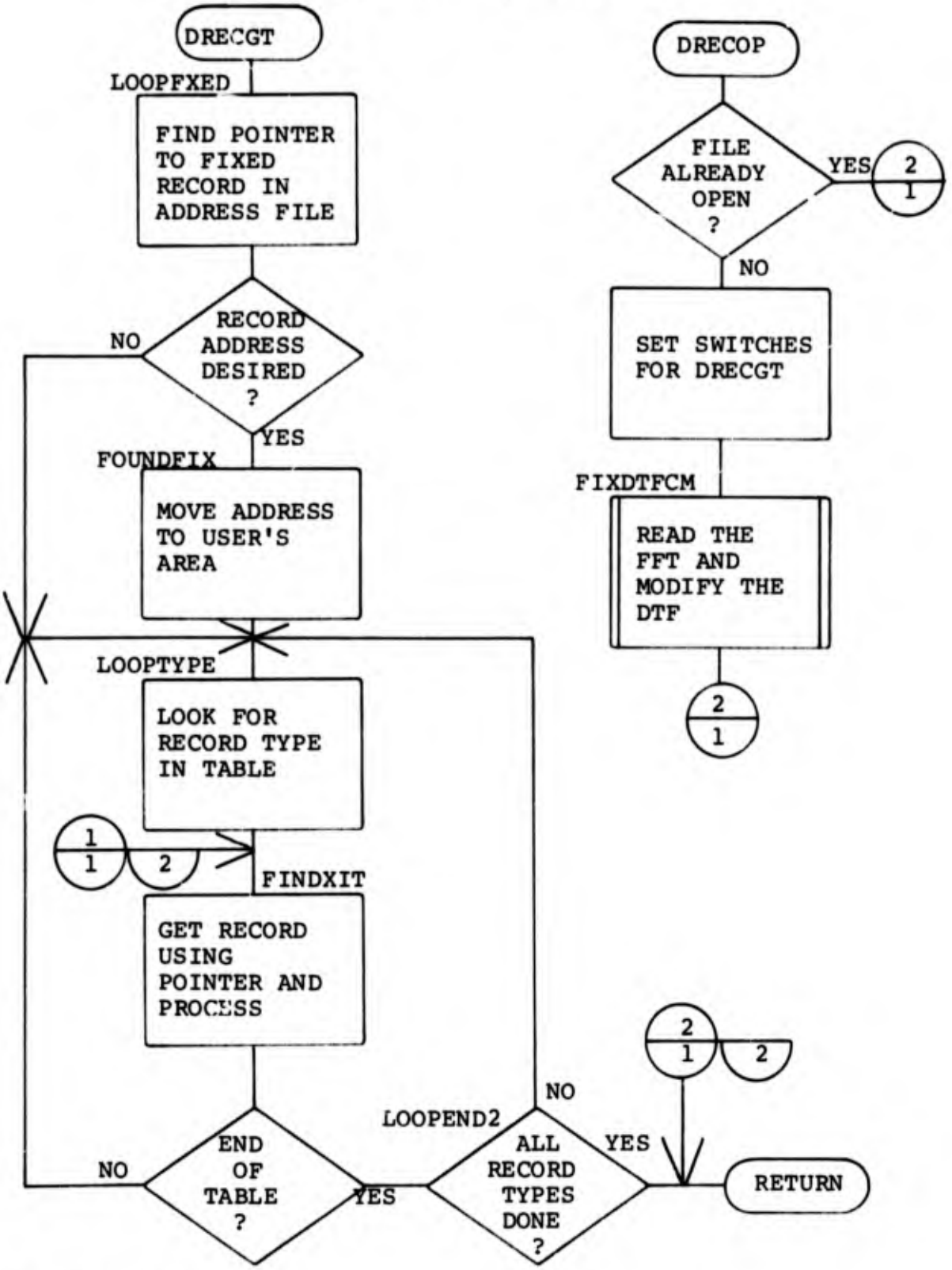
WORKAREA- Area for input file read.

BUFFER - Area for address file read.

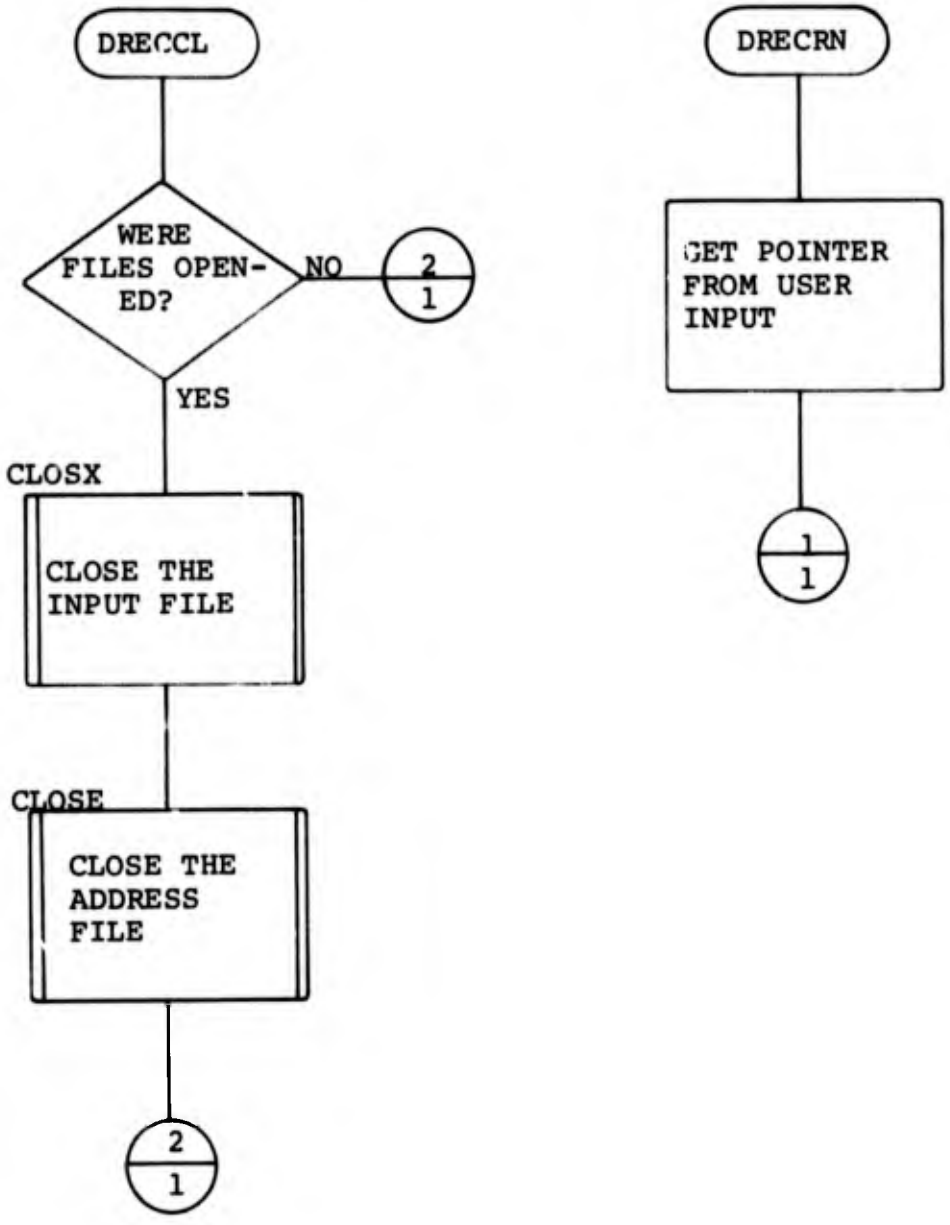
ERROR CONDITIONS: None.

SUBROUTINES CALLED: COMPNINI, FIXDTFCM, LATCONV2, LONCONV2.

DRECGT: Logic Diagram



Page 1 of 2


DRECCL
WERE FILES OPEN-ED?
NO
2
1
YES
CLOSX
CLOSE THE INPUT FILE
CLOSE
CLOSE THE ADDRESS FILE
2
1
DRECRN
GET POINTER FROM USER INPUT
1
1

CHAPTER XIV

IOC UTILITY PROGRAMS

PROGRAM ID: MAT
PROGRAM NAME: Byte Matrix Reference

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: MAT is a FORTRAN function subroutine that allows the programmer to reference the first 9216 bytes of VOL as if it were a 96 x 96-byte matrix.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: When the routine AlCON was originally written, it used a 19x20 matrix made up of halfwords to represent the grid of intensity values. However, when the on-line version of the AlCON application was being implemented, it was decided that a 96x96 matrix was needed. If the original method of using one halfword/entry in the matrix was to be carried over, an 18K area of VOL would be needed. Thus, it was decided to use one byte per entry in the matrix. Since AlCON was written in FORTRAN and already made references to a matrix called MAT, the easiest way to implement this change was to write a function subroutine called MAT. Thus, the coding in AlCON would remain unchanged but the routine would actually be referencing a 96x96-byte matrix instead of a 19x20 halfword matrix.

LOGIC NARRATIVE:

MAT - Taking the input parameters, call AlRET to get the byte. The results will be returned in M. Return to the calling routine with the result of the call in the usual function subroutine convention.

CALLING SEQUENCE: I = MAT(J,K)

where:

I = The contents of the (J*96+K)th byte in VOL.

J = A fullword variable, from 0-95.

K = A fullword variable, from 0-95.

OPTIONS: None.

CALLED BY: AlCON

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. J and K are inputs. In addition, MAT references the first 9216 bytes of VOL.

OUTPUTS: See CALLING SEQUENCE. When MAT is called as above, the result will be in the rightmost byte of I.

ENTRY POINT:

MAT

PROGRAM CONSTANTS: None.

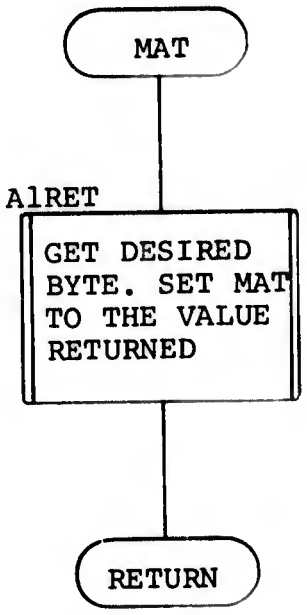
PROGRAM VARIABLES:

M - Temporary variable used to hold the result of the call to AlRET to fetch the byte.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: AlRET

MAT: Logic Diagram



PROGRAM ID: MSG
PROGRAM NAME: Message Routine

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: This subroutine's function is to move a specified message into column 1 of line 3 of the displayed screen. (Line 1 being the bottom of the screen.)

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The purpose of MSG is to conserve core space in the IOC applications programs by having the on-line messages defined in one subroutine. Then whenever a FORTRAN program wishes to use one of these messages, it merely has to call MSG with a number identifying the message.

LOGIC NARRATIVE:

This routine moves the specified message into column 1 of line 3 of the display screen on the BR90, (line 1 starting at the bottom of the screen). It then returns to the calling routine.

CALLING SEQUENCE: CALL RTN($MSG,NUM,ARRAY)

where:
NUM is the number of the message desired.

ARRAY is the first location of the buffer into which the 64 character message is to be moved.

OPTIONS: None.

CALLED BY: IOC programs that wish to use any of the messages defined in this routine.

SCREENS USED: None.

INPUTS: See NUM in CALLING SEQUENCE.

OUTPUTS: MSG moves the specified message into the display buffer.

PROGRAM CONSTANTS:

MSG1 - (All blank)

MSG2 - THIS OPTION NOT AVAILABLE AT THIS TIME.

MSG4 - DATA SELECTION EXCEEDS 24 ITEMS. LIGHTPEN ANOTHER DISPLAY OPTION.

MSG5 - INPUT VALUES CANNOT EXCEED TWENTY ITEMS.

MSG6 - INPUT AREA FILLED. LP ANOTHER DISPLAY OPTION OR RESTART QUERY.

MSG7 - INSERT MISSING STOP CODE. LP DISPLAY OPTION AGAIN.

MSG8 - PLEASE MAKE SELECTION, THEN REQUEST DISPLAY OPTION.

MSG9 - SELECT A FILE. THEN SELECT A DISPLAY OPTION.

MSG10 - MRTCODE EQUALS - CHECK FFT FOR ERROR.

MSG11 - SELECT ANOTHER DISPLAY OPTION. END OF FFT ITEMS.

MSG12 - SELECTION ACCEPTED.

MSG13 - PROCESSING YOUR REQUEST.

MSG14 - ERROR CODE 01 - INCORRECT DATA FILE NAME (LEADING BLANK).

MSG15 - ERROR CODE 20 - QQT REFERENCES A NON-EXISTENT QPT STATEMENT.

MSG16 - ERROR CODE 21 - LATITUDE COORDINATE FIELD IN ERROR.

MSG17 - ERROR CODE 22- LONGITUDE COORDINATE FIELD IN ERROR.

MSG18 - ERROR CODE 23 - LESS THAN THREE POLYGON POINTS WERE SPECIFIED.

MSG19 - ERROR CODE 24 - MORE THAN TWENTY POLYGON POINTS SPECIFIED.

MSG20 - ERROR CODE 25 - ONLY ONE PERIODIC SET MAY BE QUERIED.

MSG21 - ERROR CODE 258 - DATA FILE IN ERROR. (ERGO NO FIXED SET)

MSG22 - ERROR CODE 259 - I/O ERROR IN PROCESSING DATA FILE OR ADDR FILE.

MSG23 - INVALID FILE IDENTITY.

MSG24 - FIRST TIME FLAG SET OFF OPTION CODE MUST BE A, B, OR C.

MSG25 - INVALID OPTION CODE REQUESTED.

MSG26 - FIELD ID IN FFT DOES NOT HAVE A VALID FIRST CHARACTER. (A-Z)

MSG27 - FIELD ID REQUESTED NOT FOUND IN FFT.

MSG28 - FIELD MNEMONIC REQUESTED NOT FOUND IN FFT.

MSG29 - MQPT ITEM SELECTED IS NOT A VALID FIELD MNEMONIC FROM FNT.

MSG30 - RECORD TYPE REQUESTED FROM FFT INVALID. MUST BE A J-R, Z.

MSG31 - REQUESTED FILE ID HAS NO FFT. FILE MAY BE STRUCTURED INCORRECT.

MSG32 - FIELD ID IN FFT HAS INVALID FIRST CHARACTER. MUST BE A-Z.

MSG33 - INVALID FILE ID SELECTED. TRANSLATE TABLE OPTION.

MSG34 - FIELD MNEMONIC (FOR FILE SELECTED) NOT FOUND. TRANSLATE TABLE OPTION.

MSG35 - ITEM REQUESTED FROM TRANSTAB NOT AVAILABLE. UPDATE TABLES.

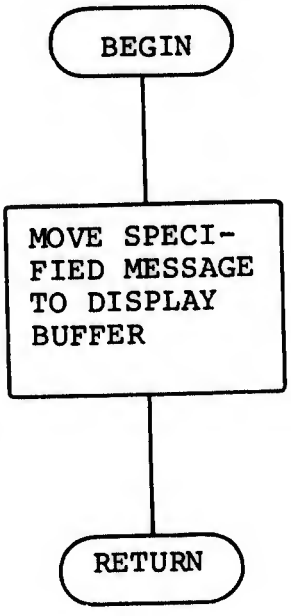
MSG36 - COORDINATE CONVERSION ERROR - RE-REGISTER THE SLIDE.

PROGRAM VARIABLES: None.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: FILL

MSG: Logic Diagram



PROGRAM ID: PNTMSG
PROGRAM NAME: Print Message Routine

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: This subroutine prints out messages for IOC Traces, and abnormal terminations.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Printer output can be handled conveniently through FORTRAN since the I/O functions are all provided. Therefore, PNTMSG was implemented to enable Assembly language programs to take advantage of these existing I/O facilities.

LOGIC NARRATIVE:

Go to the process specified by the message number.

1000 - Print the "IOC Trace from BR90" message and go to 9000.

2000 - Print the normal "IOC Trace" message and go to 9000.

3000 - Print the "IOC ABEND" message and go to 9000.

4000 - Print the "Phase Table error" message and call ABEND.

9000 - Return to the calling program.

CALLING SEQUENCE: CALL PNTMSG(MSGNUM,CODE,LOC)

where:

MSGNUM - Identifies message to be printed (See Error Messages)

CODE - For messages 1 and 2, contains pointer to current routine number. For message 3, it contains a pointer to the termination code. For message 4, it contains a pointer to the requested phase number.

LOC - For message 1, it contains a pointer to user number. For message 2, not used. For message 3, it contains a pointer to the PSW. For message 4, it contains a pointer to the phase number.

OPTIONS: The calling program specifies the message desired by identifying it in MSGNUM.

CALLED BY: STXIT and RTN.

SPECIAL NOTE: None.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE.

OUTPUTS: Messages on the printed output.

PROGRAM CONSTANTS: See ERROR MESSAGES.

PROGRAM VARIABLES: None.

ERROR MESSAGES:

Message 1: "*** IOC TRACE *** FROM BR90, NOW ENTERING ROUTINE nnnn"

Message 2: "*** IOC TRACE *** FROM ROUTINE nnnn, REF. NO. xxxx"

Message 3: "0*** IOC ABEND *** ABEND CODE yyyyyy, ABEND PSW AND REGS AT LOC zzzzzzzz"

Message 4: "1*** IOC ERROR *** ROUTINE nnnn IS NOT IN PHASE TABLE Pmmmm"

where:

mmmm - represents the phase number.

nnnn - represents the routine ID.

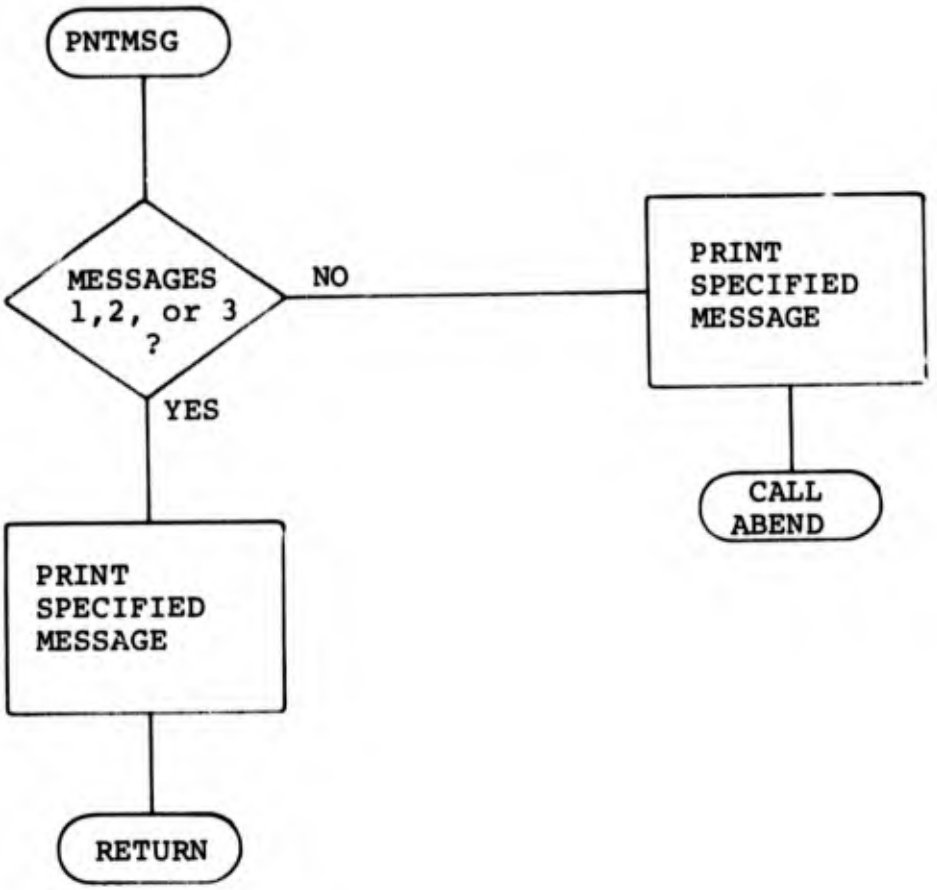
xxxx - represents the location in the calling routine where the call was issued.

yyyyyyyy- represents the abnormal termination code.

zzzzzzzz- represents the location of the PSW in the core dump.

SUBROUTINES CALLED: ABEND

PNTMSG: Logic Diagram



PROGRAM ID: IIUTIL
PROGRAM NAME: Utility Routines for FORTRAN Programs

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: IIUTIL provides the following byte-level processing for FORTRAN programs:

- Move a specified number of data bytes
- Compare a specified number of data bytes
- Set a specified number of data bytes to a desired value.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: FORTRAN provides access to data down to the half-word level. In order to provide IOC FORTRAN programs with the capability to process at the byte-level, several utility routines were developed. These routines provide the following capabilities:

- Move a specified number of data bytes
- Compare a specified number of data bytes
- Set a specified number of data bytes to a desired value.

Due to the similarities in the routines, they were combined into a single program. The different capabilities are provided through calls to different entry points in the program.

LOGIC NARRATIVE:

IICOMM - Save the registers. Set R9 for branch to compare code. Load R8 with pointer to result field, then go to IIMOV2.

IICOMF - Save the registers. Set R9 for branch to compare code. Load R8 with pointer to result field, then go to IIFIL2.

IIMOVE - Save the registers. Set R9 to fall through to move processing.

IIMOV2 - Get "TO" and "FROM" locations and number of bytes to move, then go to SAME.

IIFILL - Save the registers. Set R9 to fall through to move processing.

IIFIL2 - Compute "TO" and "FROM" locations using byte offsets. Get number of bytes to move.

SAME - Check number of bytes to be processed. If none, go to EXIT. Otherwise, process the bytes in groups of 256. If less than 256, go to LASTCHK.

BIGCHK - If Compare processing, go to COMPCD. Otherwise, fall through for move processing.

BIGMOVE - Move 256 bytes from the sending to the receiving field.

INCRMNT - Increment pointers to next group of 256 bytes and go to BIGCHK. If less than 256 bytes, go to LASTCHK.

LASTCHK - If compare processing, go to LASTCM. Otherwise, move the remaining bytes and go to EXIT.

COMPCD - Compare 256 bytes from the two fields. If they are equal, continue processing by going to INCRMNT.

AHILOW - Set high or low indicator and go to COMEXT.

LASTCM - Compare remaining bytes. If not equal, go to AHILOW. Otherwise, set equal indicator.

COMEXT - Store the indicator in the result cell and go to EXIT.

IISETM - Save the registers. Get the parameter values and go to SAME2.

IISETF - Save the registers. Get the parameter values. Compute the address of the receiving field using the byte offset.

SAME2 - Check the number of bytes to be processed. If none, go to EXIT. Fill the specified area of the receiving field with the input value.

EXIT - Restore the registers. Return to the calling program.

CALLING SEQUENCE:

```
CALL RTN($ICOMM,AFIELD,BFIELD,NUMBYT,IRSULT)
CALL RTN($ICOMF,AFIELD,IBYTA,BFIELD,IBYTB,NUMBYT,IRSULT)
```

where:

AFIELD	=	Starting location of first field.
IBYTA	=	Byte offset, relative to 1, for AFIELD.
BFIELD	=	Starting location of second field.
IBYTB	=	Byte offset, relative to 1, for BFIELD.
NUMBYT	=	Number of bytes to compare
IRSULT	=	Contains result of the compare:

```
 0 = AFIELD EQ BFIELD
 1 = AFIELD GT BFIELD
-1 = AFIELD LT BFIELD
```

```
CALL RTN($IMOVE,TO,FROM,NUMBYT)
CALL RTN($IFILL,TO,IBYTT,FROM,IBYTF,NUMBYT)
```

where:

TO	=	Address of receiving field.
IBYTT	=	Byte offset, relative to 1, for receiving field.
FROM	=	Address of sending field.
IBYTF	=	Byte offset, relative to 1, for sending field.
NUMBYT	=	Number of bytes to move.

```
CALL RTN($ISETM,FIELD,NUMBYT,IVALUE)
CALL RTN($ISETF,FIELD,IBYTE,NUMBYT,IVALUE)
```

where:

FIELD	=	Address of field to be modified.
IBYTE	=	Byte offset, relative to 1, for field to be modified
NUMBYT	=	Number of bytes to be modified.
IVALUE	=	Value with which the field is to be modified.

OPTIONS: None.

SPECIAL NOTE: Subroutines IICOMM, IICOMF, IIMOVE and IIFILL all branch to the same code after some individual preprocessing on the input parameters. Subroutines IISETM and IISETF branch to a set of common code after preprocessing their own set of input parameters.

SCREENS USED: None.

INPUTS: See discussion on CALLING SEQUENCES.

OUTPUTS: Subroutines IICOMM and IICOMF return an indicator for the result of the compare. See IRSULT in CALLING SEQUENCE.

ENTRY POINTS:

IICOMM - This subroutine compares a specified number of bytes in one field against the same number of bytes in another field.

IICOMF - This subroutine compares a specified number of bytes in one field (with a specified byte offset) against the same number of bytes in another field (with a specified byte offset).

IIMOVE - This subroutine moves a specified number of bytes from one field to another.

IIFILL - This subroutine moves a specified number of bytes from one field (with a specified byte offset) to another field (with a specified byte offset).

IISETM - This subroutine sets a specified number of bytes in a field to a value provided by the calling program.

IISETF - This subroutine sets a specified number of bytes in a field (with a specified byte offset) to a value provided by the calling program.

REGISTER USAGE:

R10 - Constant of 1, used to decrement byte count. Also used to compute the result code.

R6 - Contains number of bytes for last move or compare.

R9 - Flag used in switching to move or compare processing.
00 = Fall through to move processing.
F0 = Branch to compare processing.

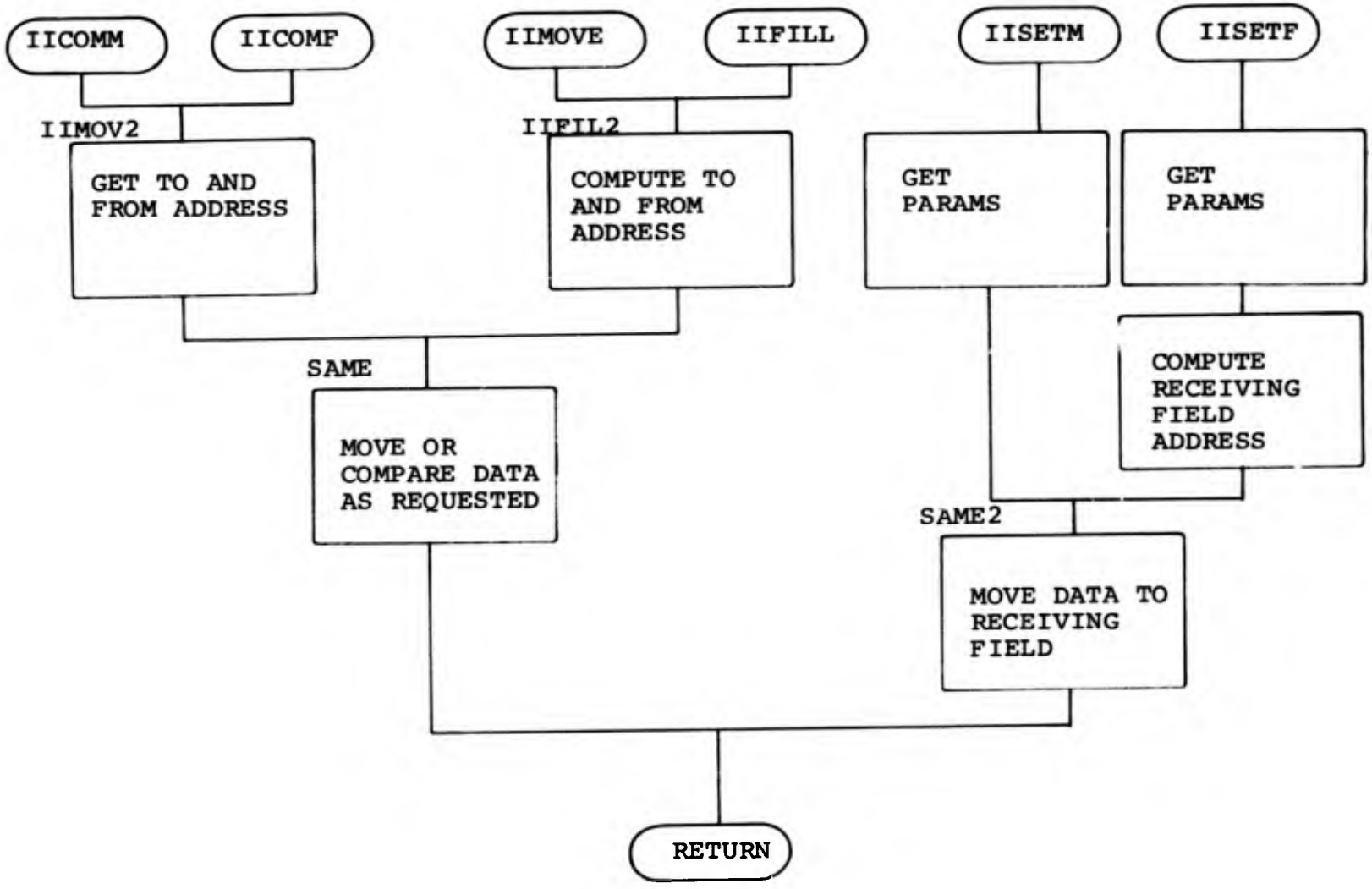
PROGRAM CONSTANTS: None.

PROGRAM VARIABLES: None.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

IIUTIL: Logic Diagram



PROGRAM ID: IITBL2
PROGRAM NAME: Table Search and Scan Routines

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: IITBL2 is a generalized table search subroutine. It searches the specified table for an entry that is equal to the input argument.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: IITBL2 is a generalized table search routine. It has three entry points which provide the user with the capability to specify the following:

- Location of the table to be searched,
- Location of first argument within the table,
- Size of each entry,
- Number of entries to be searched,
- Size of each argument, and
- Location of the argument within each entry.

The user can also specify byte-offsets for any of the locations he specifies. This provides FORTRAN programs with all of the flexibility they need for table searches.

LOGIC NARRATIVE:

IITBL2 - Save the registers. Load argument and table addresses and offsets. Compute the initial address of Argument and Table. Load the remaining parameters and go to COM2.

IITBLS - Save the registers. Load the parameter registers.

COM2 - Load R5 with table entry size. Decrement field size and field position by one and load R4 with first compare location in TABLE. Then go to COMMON.

IISCAN - Save the registers. Load the parameter registers. Compute the initial address of Argument and Table. Load R5 with table entry size. Decrement field size by one and fall through to COMMON.

COMMON - Load R6 with number of entries and R0 with number of entries plus one.

LOOP - Compare argument against table entries until a match is found or until table is exhausted. If a match is found, move matching entry number to R0. Otherwise, set R0 to zero.

EXIT - Restore the registers and return to the calling routine.

CALLING SEQUENCES:

```
IRSULT = IITBLS(ARG,TABLE,IENTSZ,ITBCNT,IFLDSZ,IFLDPS)
```

where:

IRSULT - Return code provided by IITBLS: 0 = Argument not in table. n = Entry number of the argument in the table.

ARG - Location (high-order) of search argument.

TABLE - Location (high-order) of table to be searched.

IENTSZ - Size of each entry in the table.

ITBCNT - Number of entries in the table.

ILFDSZ - Size of the argument and the search field in the table.

IFLDPS - Relative byte position of the search field within the table entry (relative to 1).

```
IRSULT = IITBL2(ARG,BYTEA,TABLE,BYTEB,IENTSZ,ITBCNT,IFLDSZ,
                IFLDPS)
```

where:

IRSULT - Return code. (See above.)

ARG - Location (high-order) of search argument.

BYTEA - Byte offset (relative to 1) of search argument.

TABLE - Location (high-order) of table to be searched.

BYTEB - Byte offset (relative to 1) of table.

IENTSZ - Size of each entry in the table.

ITBCNT - Number of entries in the table.

IFLDSZ - Size of the argument and the search field in the table.

IFLDPS - Relative byte position of the search field within the table entry (relative to 1).

```
IRSULT = IISCAN(ARG,NBYTE,TABLE,MBYTE,ITBCNT,IFLDSZ)
```

where:

IRSULT - Return code. (See above)

ARGO - Location (high-order) of search argument.

NBYTE - Byte offset (relative to 1) of search argument.

TABLE - Location (high-order) of table to be searched.

MBYTE - Byte offset (relative to 1) of table.

ITBCNT - Number of entries in the table.

IFLDSZ - Size of the search argument and the entries in the table.

OPTIONS: None.

CALLED BY: IOC FORTRAN programs that require table searches at the byte-level.

SCREENS USED: None.

INPUTS: See discussion on CALLING SEQUENCES.

OUTPUTS: Pointer to the matching entry in IRSULT.

ENTRY POINTS:

IITBLS
IITBL2
IISCAN

REGISTER USAGE:

R3 - Address of search argument.

R4 - Address of table to be searched.

R5 - Size of table entries.

R6 - Number of entries in the table.

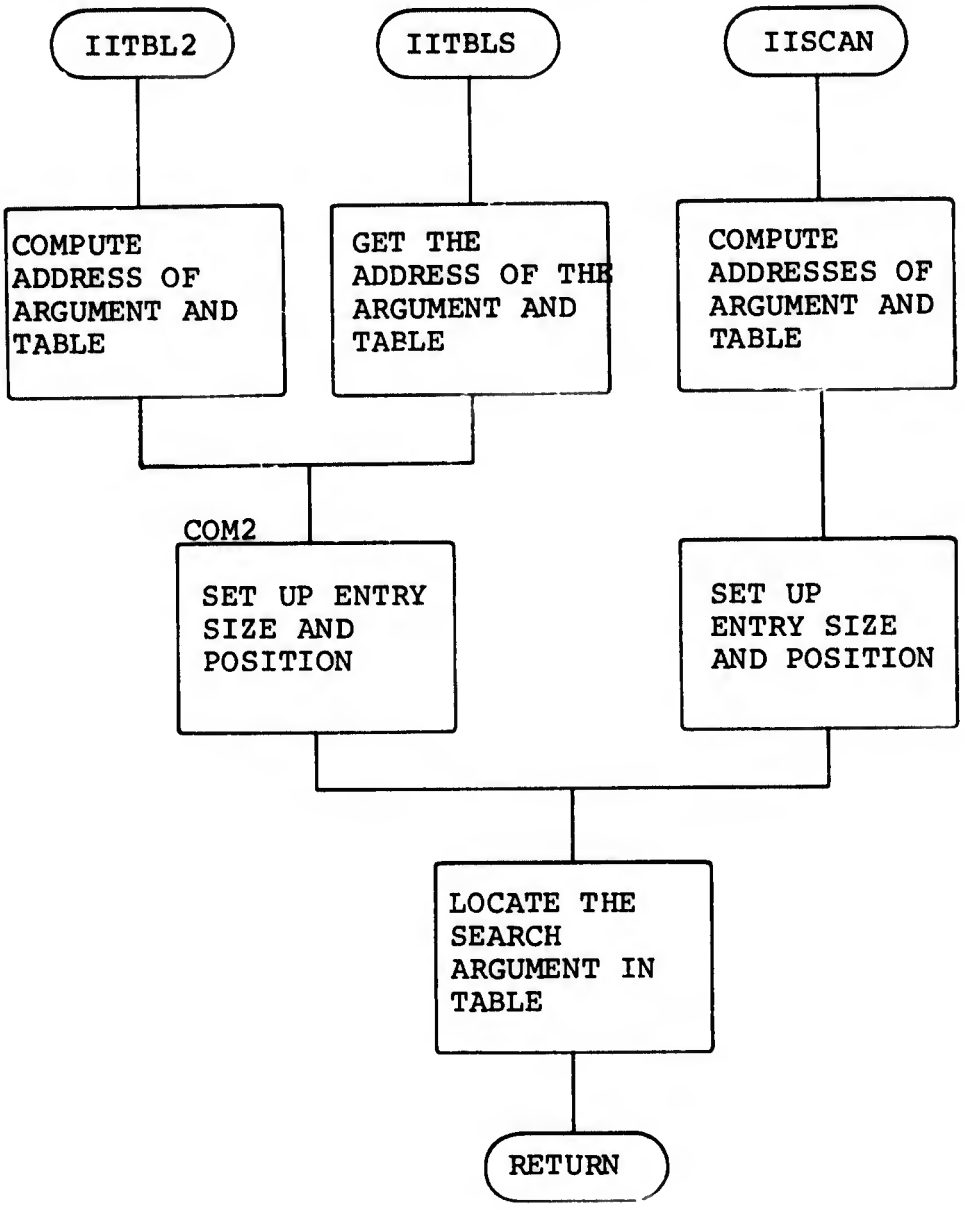
R7 - Size of search argument.

R8 - Byte offset within an entry.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

IITBL2: Logic Diagram



PROGRAM ID: IISRTB
PROGRAM NAME: General-Purpose Internal Sort Routine

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This routine sorts the entries in a table into ascending order.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: This routine sorts a table with ITBCNT entries of IENTSZ bytes on a IFLDSZ-byte field within the entry. In order to do this without limiting the entry size, it is required that the table have a dummy entry of IENTSZ bytes, to be used as a work area. Initially, the table looks like:

DUMMY,ENTRY1,...,ENTRYn

and after sort:

ENTRYA,....,ENTRYX,GARBAGE

The algorithm used is as follows: Given a table with N entries (including the dummy), we will have N-1 passes. On the i^{th} pass, find the smallest entry in the last N-i entries. Move this entry to the i^{th} place. Then replace the smallest entry by the i^{th} lst entry. Increment i by 1 and start a new pass.

LOGIC NARRATIVE:

IISRTB - Initialization takes place. Also, pointers to the first entry and the field within the entry are set. A check is made to see if there is at least 1 entry in the table. If not, go to EXIT.

OUTERLOP- Execute the sort algorithm as explained in the APPROACH section.

EXIT - Return to the calling program.

CALLING SEQUENCE: CALL IISRTB(TABLE,IENTSZ,ITBCNT,IFLDSZ,IFLDPS)

where:

TABLE	=	beginning of table to be sorted
IENTSZ	=	size of each entry in the table
ITBCNT	=	number of actual entries in the table
IFLDSZ	=	size of sort key in entry
IFLDPS	=	position of sort key in entry, starting with 1.

OPTIONS: None.

CALLED BY: This routine is called by IOC and batch programs as required.

SPECIAL NOTES: The entry size and key size must be less than 256 bytes. The reason this limit is needed is that IISRTB uses MVC and CLC instructions, and the length limit on these is 256 bytes.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. All parameters are used as input.

OUTPUTS: See CALLING SEQUENCE. Output from IISRTB is a sorted table.

ENTRY POINTS: IISRTB

REGISTER USAGE:

R15	-	base register
R3	-	entry size
R11	-	pointer to current smallest entry in the table
R12	-	current next compare argument
R9	-	counter for outerloop of sort

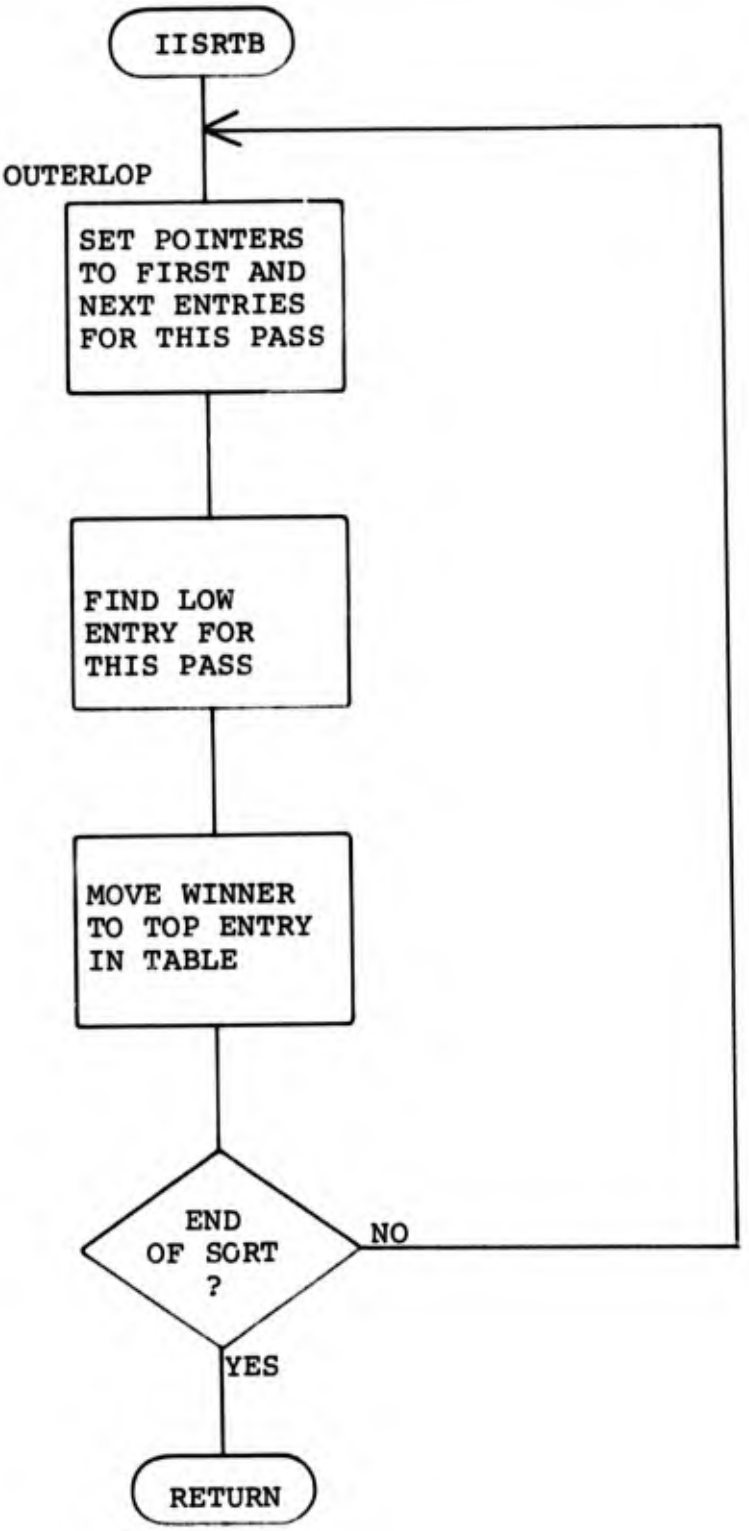
PROGRAM CONSTANTS: None.

PROGRAM VARIABLES: None.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

IISRTB: Logic Diagram



PROGRAM ID: IISRTA
PROGRAM NAME: Core Sort Routine A

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: IISRTA is a core sort routine. It sorts table entries based on a key field within the entry. It is a "logical" sort.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: IOC applications programs have internal tables or arrays of odd sizes. Since these tables have to be sorted, a generalized table sort routine was implemented.

The sort is accomplished in the following manner. During the first pass through the table, the first entry is assumed to be the lowest value and it is compared against the other entries until a lower value is found or the end of table is reached. If a lower value is found, that value is exchanged with the previous low value. At the end of the pass, the lowest value is moved into the first entry of the table and the pointers are adjusted to show that the second entry is the next available entry and n-1 entries remain to be sorted. The compare and replace procedure is repeated until the whole table is sorted.

LOGIC NARRATIVE:

IISRTA - Initialize by saving the contents of registers, getting the input parameters and adjusting their values for processing.

OUTERLOP - Adjust the pointers to the first and second sort keys for this scan. Adjust the number of scans that remain.

COMPDATA - Look for lowest entry for this pass. Register 2 points to current low entry and Register 12 points to next entry in the table. At the end of the pass, move the low entry to the first entry for this pass. Branch to OUTERLOP if any more passes required for the table. Otherwise, fall through to EXIT.

EXIT - Restore the contents of the registers and return to the calling program.

CALLING SEQUENCE: CALL IISRTA(TABLE,IENTSZ,ITBCNT,IFLDSZ,IFLDPS)

where:

TABLE - First location (high order) of table to be sorted.

IENTSZ - Size of the table entries.

ITBCNT - Number of entries in the table.

IFLDSZ - Size of key (sort) field within each entry.

IFLDPS - Relative position of the key (sort) field within each table entry. (First position is 1, not 0.)

OPTIONS: None.

CALLED BY: IISRTA is called by IOC programs as required.

SPECIAL NOTE: IISRTA can easily be modified to recognize an end-of-table flag instead of using a counter.

SCREENS USED: None.

INPUTS: Table which is to be sorted. In addition, see discussion on CALLING SEQUENCE.

OUTPUTS: Output from IISRTA is the sorted table.

ENTRY POINTS:

IISRTA

REGISTER USAGE:

R9 - Contains the number of entries in the table which is also the number of compare cycles that must be executed.

R2 - Initially, contains address of table to be sorted. Thereafter, points to entry where next lowest value is to be stored.

R3 - Contains the size of the table entries.

R4 - Initially contains number of entries in table. The number is decremented as the table is sorted.

R5 - Contains the size of the key field. It is decremented by one for use in the compare instruction.

R6 - Contains the relative position of the key field within each table entry. It is decremented by one before being used in address modification.

R7 - Contains the size of the table entries. It is decremented by one for use in the Execute instruction.

R11 - Pointer to current low entry for a pass.

R12 - Pointer to next entry in the table.

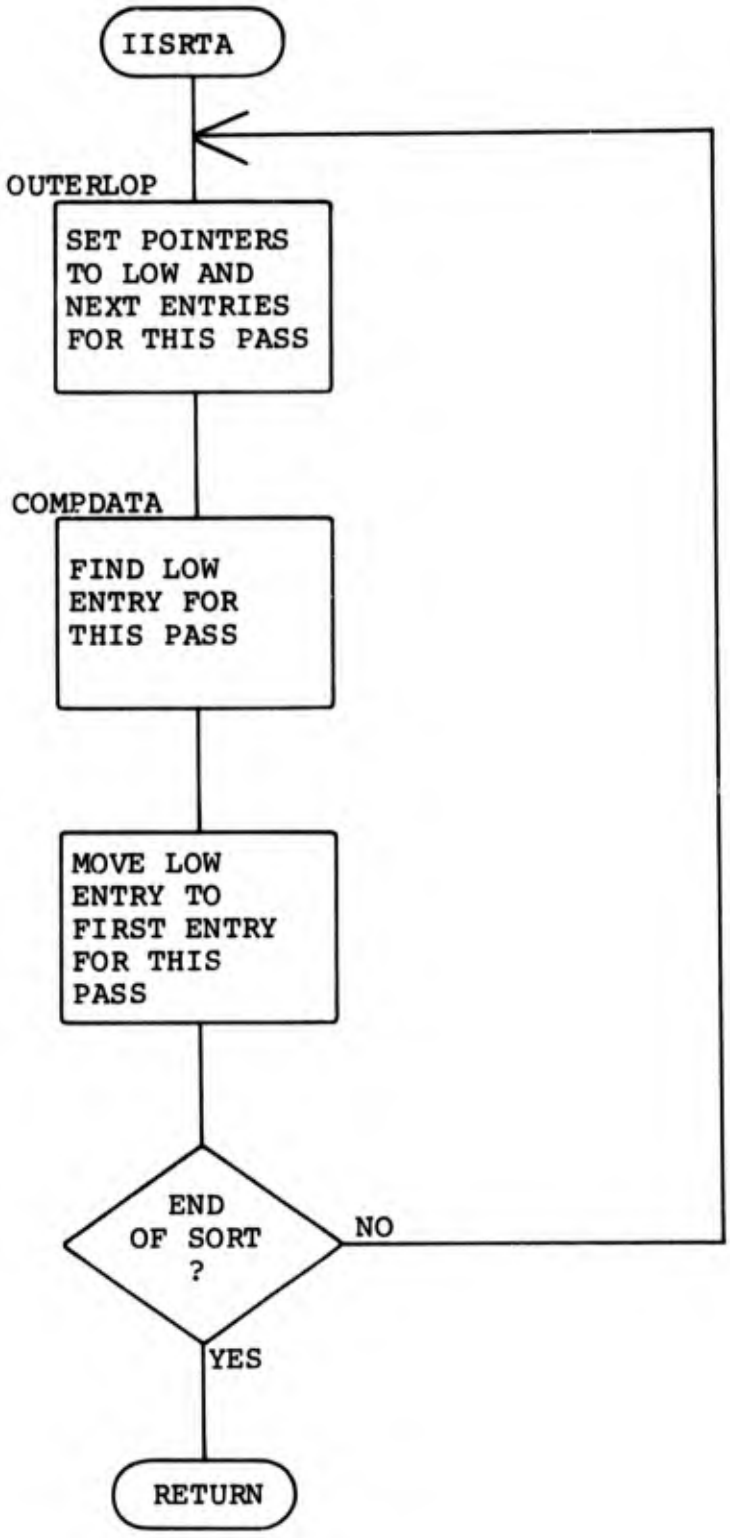
PROGRAM CONSTANTS: None.

PROGRAM VARIABLES: None.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

IISRTA: Logic Diagram



PROGRAM ID: IISHFT
PROGRAM NAME: Shift Routine

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: IISHFT is used to shift a field a specified number of bytes to the right or left. In the case where the field width is less than the shift count, this is equivalent to a move. IISHFV is used to shift a field a specified number of bytes to the right or left. In addition, all vacated bytes are filled with the specified value.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Occasionally, in FORTRAN routines, shifting fields of data several bytes to the left or right is necessary. FORTRAN does not allow this capability. Thus, in IOC, two routines, IISHFT and IISHFV were implemented. They are both general-purpose shifting routines. IISHFV also shifts a desired byte value into vacated byte positions. In order to attain full generality, this shifting operation is accomplished by a series of data moves.

LOGIC NARRATIVE:

IISHFV - Get ready for value fill-in.

IISHFT - Check field pointers for validity. Compute field width, pick up shift count, and determine whether left or right shift. If left shift, no possibility of overwriting, so go to EASY1. Otherwise, if field size less than shift count, the shift will go beyond the field, so go to EASY2. In the last case, do moves in groups of length NUMBYT, starting with the right-most group. Then go to VALCHK.

EASY1 - Set up for possible value fill-in.

EASY2 - Move the field in groups of maximum 256 bytes, until field is exhausted. Then check if value fill-in is needed. If not, RETURN.

VALCHK - Fill in vacated bytes. Then RETURN.

CALLING SEQUENCE: CALL IISHFT(LEFT,RIGHT,NUMBYT)
CALL IISHFV(LEFT,RIGHT,NUMBYT,IVALUE)

where:

LEFT = Leftmost position of the field to be shifted
RIGHT = Rightmost position of the field to be shifted
NUMBYT = Number of bytes to be shifted. If it is positive, a right shift takes place. Otherwise, a left shift takes place.
IVALUE = Value to be shifted is in the rightmost byte of IVALUE.

OPTIONS: None.

CALLED BY: FORTRAN routines that need to shift data on the byte level.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE.

OUTPUTS: Shifted field with value fill as specified in CALLING SEQUENCE.

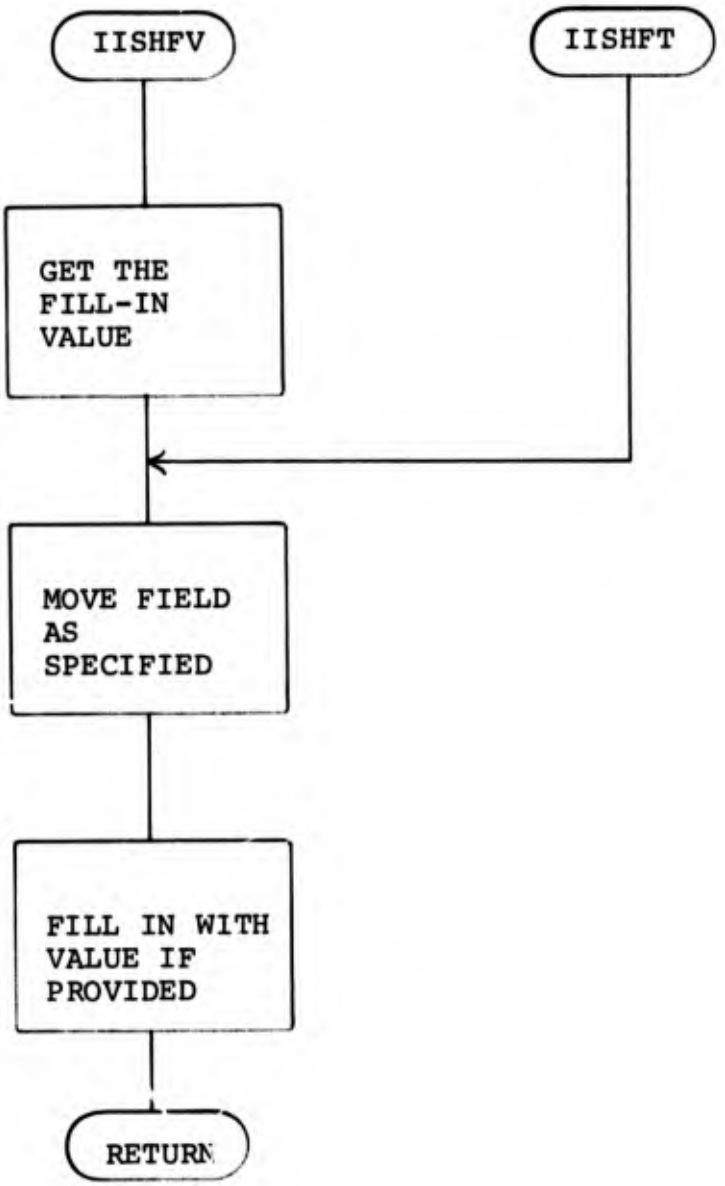
REGISTER USAGE:

R7 - Used as a switch if value fill-in is needed
R6 - Contains IVALUE pointer
R15 - Base register
R2 - Pointer to LEFT
R3 - Pointer to RIGHT
R4 - Pointer to NUMBYT
R10 - Leftmost pointer (used to fill-in)
R5 - Count
R8 - Field width
R1,R0, R9,R11, R12 - Work registers

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

IISHFT: Logic Diagram



PROGRAM ID: IIBOOL
PROGRAM NAME: Boolean Operators and Address Function

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This program consists of three routines. The first two, IIANDF and IIORF are boolean operator functions, and return the AND and OR of their operands, respectively. The third, IIADRF, returns the address of a variable.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Under FORTRAN, operations to the half-word level are allowed. However, operations such as bit manipulation are not possible. One way of providing this ability is to allow the user to use some sort of boolean operations, in order to mask bits out, etc. In IOC, these capabilities are implemented as function subroutines, IIORF and IIANDF. These two are general purpose OR and AND functions.

LOGIC NARRATIVE:

IIANDF - This is the entry point for the AND operator. Here, an indicator is set to point to the AND instruction for later use. Then, control passes to the common code at SAMECD.

IIORF - An indicator is set to point to the OR instruction.

SAMECD - This is the common code section. Here, the routine goes sequentially through the list of parameters and does the indicated operation. On completion, the routine returns to the caller.

IIADRF - The routine simply takes the contents of the address given in R1 and puts it into R0, then returns.

CALLING SEQUENCE: I1 = IIANDF(F1,...,FN)

where:
F1 is a full-word variable
I1 is a full-word variable

The result is returned in Register 0.

I1 = IIORF(F1,...,FN)

where:
Fi is a full-word variable
I1 is a full-word variable

The result is returned in Register 0.

I1 = IIADRF(F1)

where:
F1 is a full-word variable
I1 is a full-word variable

The result is returned in Register 0.

OPTIONS: None.

CALLED BY: IOC FORTRAN applications programs as required.

SPECIAL NOTES: None.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE.

OUTPUTS: All routines return results in Register 0.

ENTRY POINTS:

IIADRF
IIANDF
IIBOOL
IIORF

REGISTER USAGE: IIANDF

R6 - Points to ANDF label to control EX instruction.

R15 - Base register for common code

R0 - Result of operation

R5 - Used to point to 1st parameter

R2 - Points to current parameter pointer

R4 - Points to current parameter

R14 - Return address

IIORF

R6 - Points to ORF label to control EX instruction

R15 - Base register for common code

R0 - Result of operation

R5 - Used to point to 1st parameter

R2 - Points to current parameter pointer

R4 - Points to current parameter

R14 - Return address

IIADRF

R1 - Initially, contains pointer to pointer list.

R0 - Result of finding address of parameter.

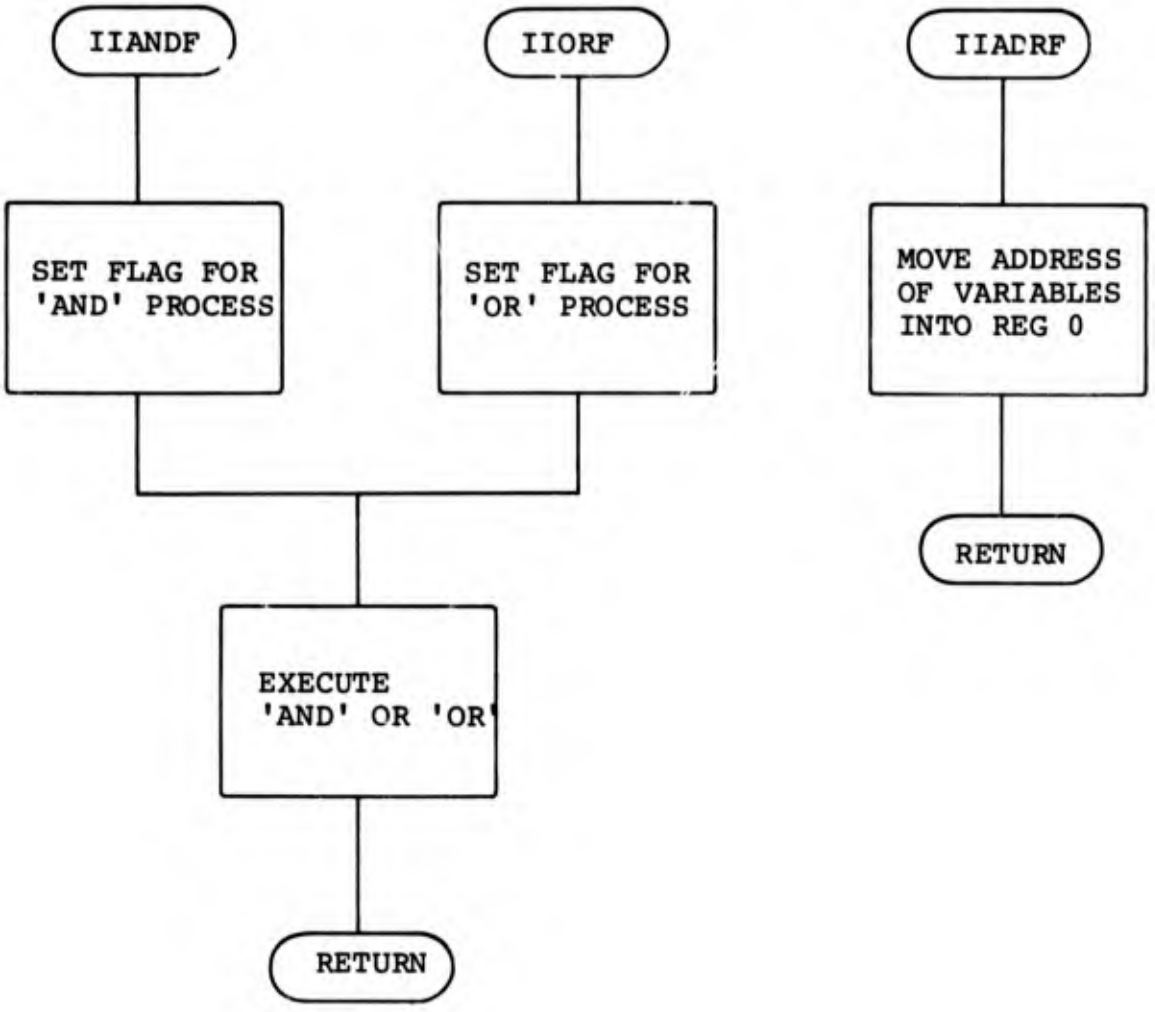
PROGRAM CONSTANTS: None.

PROGRAM VARIABLES: None.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

IIBOOL: Logic Diagram



PROGRAM ID: USED
PROGRAM NAME: Test or Set Used Bit Table

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This routine will either test a specified bit in a 96 x 96 bit array, or set that bit to a specified value.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: USED is called by the subroutine AlCON to access a 96 x 96 bit array in VOL. This bit array is used by AlCON to indicate the status of nodes in a contour map, which is represented by a 96 x 96 array. In order to minimize core requirements for the status table, it was decided that a bit array should be used. However, FORTRAN facilities only allow data access to the half-word level. Thus, the routine USED was implemented.

USED simply uses the first index in the (IX,IY) pair to indicate the row in the bit array and the second index to calculate the byte and bit in the row. Using these, the actual position of the bit is found and set or tested.

LOGIC NARRATIVE:

USED - The actual position in the bit array is determined from the input (I,J) type index.

SPLIT - Here, a branch is made to TEST if the program is to test the bit. Otherwise, a branch is made to SET.

TEST - The bit is tested, and the IR parameter is set accordingly. It then returns to the calling program.

SET - The bit is set from the (IR) parameter. It then returns to the calling program.

CALLING SEQUENCE: CALL USED(IY,IX,ACTION,IR)

where:

(IX,IY) = position in bit table from 1-96
ACTION = 0, for test value into IR
1, for set value from IR
IR = either test result or set value

OPTIONS: The routine will either test or set a bit in a bit table, depending on the value of ACTION. (See CALLING SEQUENCE)

CALLED BY: AlCON

SPECIAL NOTES: IX and IY must be 1-96. Any smaller or larger number will result in error.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE.

OUTPUTS: See CALLING SEQUENCE.

ENTRY POINTS: USED

REGISTER USAGE:

R3 - holds byte address in which affected bit will be found

R2 - bit mask for bit in byte

R15 - base register for program

PROGRAM CONSTANTS: None.

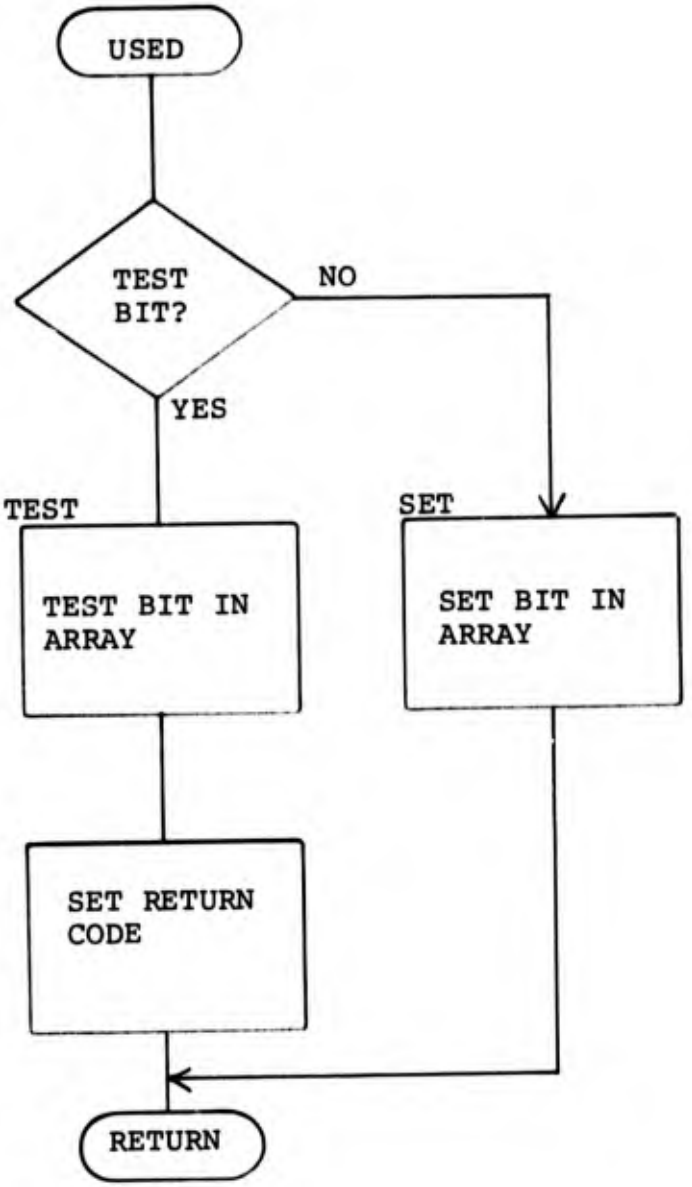
PROGRAM VARIABLES:

ARRAY - a 1152 byte table used as the bit table.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

USED: Logic Diagram



PROGRAM ID: GETBIT
PROGRAM NAME: Get or Put a Bit in a Bit Table

Company: Informatics. Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This program will get or put a bit in a specified position in a bit table of at most 256 bits.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The routines GETBIT and PUTBIT are intended to enable the programmer to get or put bits in a bit table. The primary use of such facilities in a system like IOC is to create and make use of table entries which would otherwise require as much as a half-word per flag position. In order to allow the FORTRAN programmer some alternative to this, and thus to allow him to reduce core requirements, he is given this ability to access data on a bit level.

LOGIC NARRATIVE:

GETBIT - This is the entry point to get a bit setting from the table. It sets up a switch in the common code to get a bit from the table, then branches to COMMONCD.

PUTBIT - Set a switch so that the routine will put a bit into the bit table.

COMMONCD- The initial code in the common section sets up a pointer to the appropriate byte in the table. Then, depending on how the switch is set, control passes to either GETIT or STORE.

STORE - Here the actual putting is done, then it returns to the calling routine.

GETIT - The get portion is done here. If the bit is 1, the last parameter is set to 1. Otherwise, the last parameter is set to 0.

CALLING SEQUENCE: CALL RTN($GETBT,TABLE,IBITNM,IRSULT)
CALL RTN($PUTBT,TABLE,IBITNM,IVALUE)

where:
TABLE is the lst byte of a table with a maximum size of 32 bytes
IBITNM is the bit within the table to be referenced. The range is from 1 - 256
IRSULT will be a full-word 1 if the bit is on, or a 0 if the bit is off
IVALUE contains the bit to be stored in its rightmost position and is a full-word integer.

OPTIONS: None.

CALLED BY: IOC routines that need to access tables at the bit level.

SPECIAL NOTE: The program checks to see if IBITNM is greater than 256. If so, it returns immediately. However, this does not take into account the actual size of the bit table. Thus, the user must make sure that IBITNM is always less than the size he allocates for the table.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE.

OUTPUTS: See CALLING SEQUENCE.

ENTRY POINTS:

GETBIT
PUTBIT

REGISTER USAGE:

R2,3,4 - Parameter registers initially

R12 - Base register

R9,R11 - Registers used to compute the location of the byte which contains the specified bit.

R6 - Contains the byte from the table

R8 - Bit offset within the byte.

R0 - Work register

R14 - Return register

PROGRAM CONSTANTS:

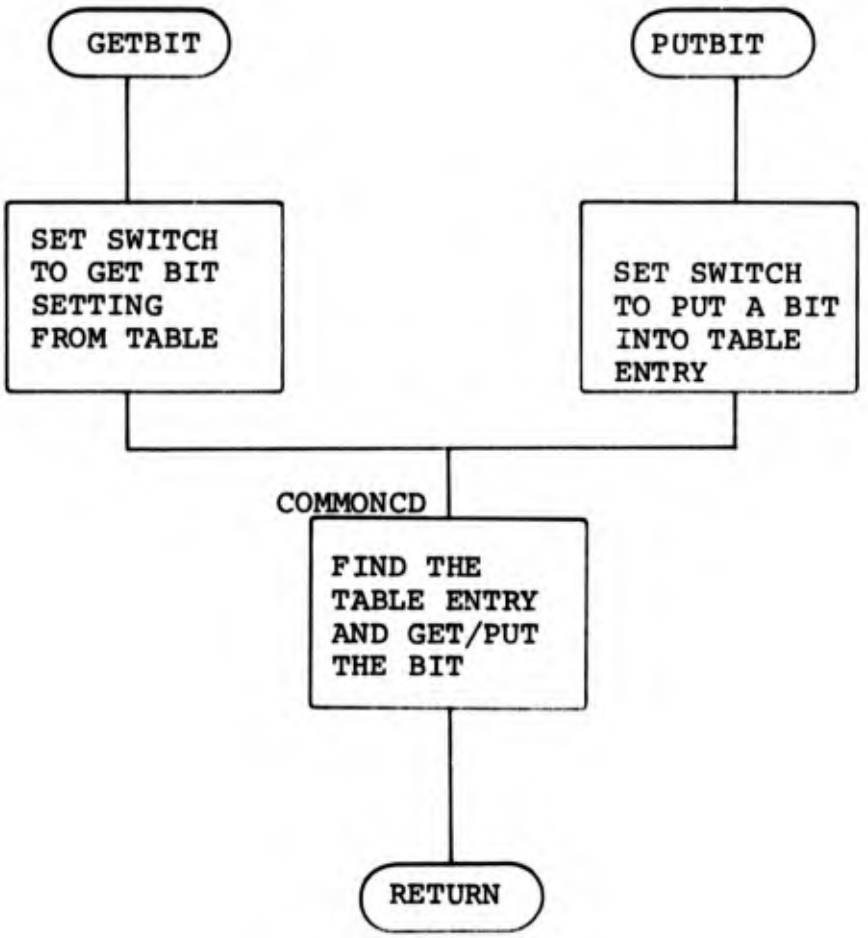
BIT - A table consisting of 8 bytes, each with one bit on in sequential relative position. Used as a mask in getting or putting the bit.

PROGRAM VARIABLES: None.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

GETBIT: Logic Diagram



PROGRAM ID: JBCON
PROGRAM NAME: Convert 3-Bit Octal to 4-Bit Octal

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This routine was written in order to put numbers which are taken from the BR90 internal cursor into the same form as on the BR90 screen. In other words, it converts raster units into octal representations.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: JBCON was written specifically to fulfill a need which occurred in the slide update process. It was desired that a punched card be produced for each slide which was registered on the BR90. On this card, among other data items, was the octal coordinates which appear on the BR90 screen. Since the internal coordinates passed to the 360 did not correspond to these coordinates, JBCON was written to do the conversion.

LOGIC NARRATIVE:

JBCON - This routine converts a 12-bit binary number into the octal representation of the number divided by 4. Originally, this was used by the slide update routine. One of the variables passed from the BR90 was the Cursor coordinates. This variable was in rasters (1_{10} - 4096_{10}). However, what appeared on the BR90 screen was an octal number (1_8 - 1777_8). In order to punch a card with these octal numbers on it, the update routine called JBCON. Each octal digit of the result occupies 4 bits. Initially, the input string is right-shifted 2 bits to divide it by 4. Then, 3 bits

at a time are masked out and placed in a 4-bit position in the output string.

CALLING SEQUENCE: ICURS=JBCON(CURSR)

where:
CURSR is a full-word variable containing a number less than 4096_{10}.

OPTIONS: None.

CALLED BY: SLIDER

SPECIAL NOTE: JBCON was designed to handle numbers less than 4096_{10}. However, if CURSR is less than 2^{16}, it will be converted correctly. After that, the numbers will be taken modulo 2^{16}.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE.

OUTPUTS: JBCON leaves the converted result in Register 0.

ENTRY POINTS:

JBCON

REGISTER USAGE:

R0 - Result register

R1 - Input string

R2 - Holds 3 bits of input string which are masked out

R3 - 3-bit mask. Mask moves to get 3 bits from input string

PROGRAM CONSTANTS:

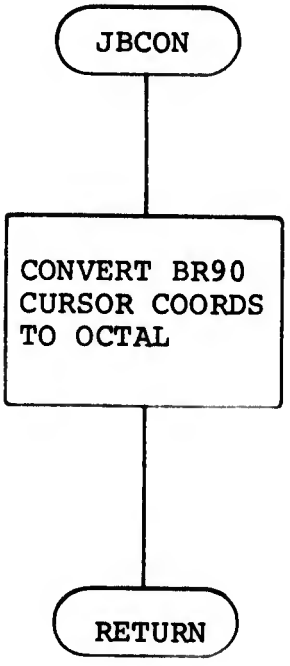
MASK - Full-word with following contents: $0....0111_2$

PROGRAM VARIABLES: None.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

JBCON: Logic Diagram



PROGRAM ID: IICBTD
PROGRAM NAME: Convert Binary to Decimal or Decimal to Binary

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: IICBTD converts a 4-byte FORTRAN integer into an 8-byte character string. The resulting field is: FOFNFNFNFNFNFN.

IICDTB converts a string consisting of from 1 to 8 bytes into a FORTRAN integer.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: As with many systems which handle a mixture of character and numeric information, there is a problem of converting from one format to the other. Unfortunately, FORTRAN does not provide a facility for handling this conversion. Thus, the routines IICBTD and IICDTB were implemented. These two routines allow conversion from binary to its numeric character string equivalent, and vice-versa.

LOGIC NARRATIVE:

IICBTD - This routine consists of a simple CVD and UNPK sequence of instructions. In addition, it puts an 'F0' in the first byte of the decimal field. Then, it returns to the calling program.

IICDTB - This routine converts a number for the decimal format to its binary equivalent.

CHKLOOP - Here any blank or non-numeric bytes are converted to zeroes. Then, it does a PACK and CVB to convert the number into binary. Finally, it returns to the calling program.

CALLING SEQUENCES: CALL IICBTD(IDEC(1),IINTGR)

where:

IDEC(1) is an 8-byte field

IINTGR is a 4-byte integer to be converted.

CALL IICDTB(IINTGR,IDEC,NUMBYT)

where:

IINTGR is a 4-byte field

IDEC is a field of 1 to 8 bytes containing a string to be converted.

NUMBYT is the number of bytes in IDEC to be converted.

OPTIONS: None.

CALLED BY: Any program which needs to do decimal-binary conversion, or vice-versa.

SPECIAL NOTES: In IICBTD, it will convert up to 7 decimal digits, and the rest will be ignored.

In IICDTB, up to 8 bytes can be converted.

SCREENS USED: None.

INPUTS: None.

OUTPUTS: None.

ENTRY POINTS:

IICBTD
IICDTB

REGISTER USAGE: IICBTD

R15 - Base register

R2,R3 - Parameter registers (IDEC and IINTGR)

R7 - Work register

IICDTB

R15 - Base register

R2 - IINTGR pointer

R3 - IDEC pointer

R4 - NUMBYT pointer

R7 - Pointer to current byte, used in character check

R6 - Count of number of bytes, used in character check

R5 - Work register

PROGRAM CONSTANTS: None.

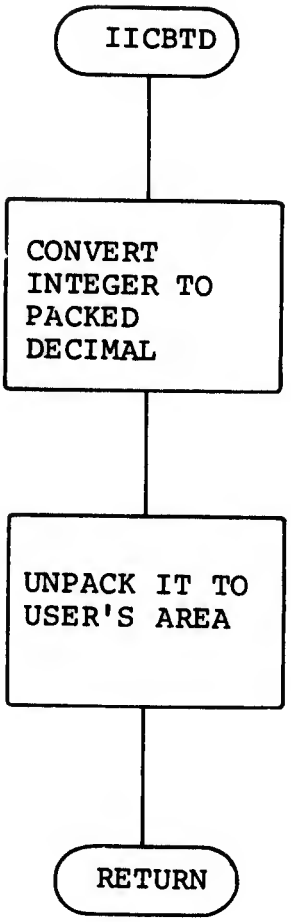
PROGRAM VARIABLES:

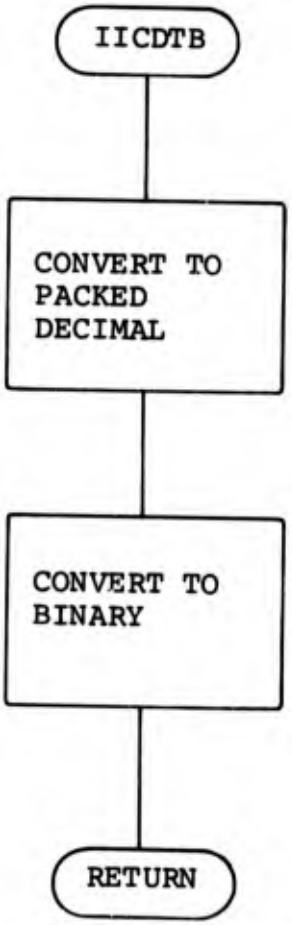
WORK - 8-byte field used in packing, unpacking, and converting.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

IICBTD: Logic Diagram





PROGRAM ID: LATCONV2
PROGRAM NAME: Convert Latitude and Longitude from Minutes to Geographic Coordinates

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This routine will convert a coordinate in minutes into the appropriate geographic coordinate.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Many IOC application routines deal with geographic coordinates. In order to handle them more easily, especially in situations which require some arithmetic operations, they can be converted to minutes. The reverse process is also necessary, since many times the coordinates must be displayed. This is the purpose of the two routines LATCONV2 and LONCONV2.

LOGIC NARRATIVE:

LATCONV2- Initially, the error flag is set to 1. Then, the minutes input is checked to see if it is less than 180 * 6000. If not, it branches to RETURN. Otherwise, compensate for the hemisphere, if needed.

CONVERT - Convert the minutes to degrees, minutes and seconds. Then, interpret the slash code, insert slashes as required, and set the error flag to 0.

RETURN - Return to the calling program.

LONCONV2- The error flag is preset to 1. Then a check is made to see if minutes is less than 360 * 6000. If not, go to RETURN. Otherwise, the hemisphere is determined, and then go to CONVERT to do the actual conversion.

CALLING SEQUENCE: CALL LATCONV2(ICOORD)
CALL LONCONV2(ICOORD)

where:
ICOORD is a 15-byte area with the following contents:

Bytes 1-8	8-byte result field with geographic coordinate
9	1-byte offset to conform to SS/FFS format
10-13	The coordinate in minutes
14	Slash code, indicates number of slashes to be generated
15	Error flag: 0 = no error in conversion 1 = error in conversion

OPTIONS: None.

CALLED BY: IIMNLL

SPECIAL NOTE: For LATCONV2, minutes must be less than 180*6000. For LONCONV2, minutes must be less than 360*6000.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. Bytes 9-14 are input to these routines.

OUTPUTS: See CALLING SEQUENCE. Bytes 1-8 and 15 are outputs from the routines.

ENTRY POINTS:

LATCONV2
LONCONV2

PROGRAM CONSTANTS:

MASKA - An 11-byte field used as a pattern in editing the input.

MASKB - Byte table for translation instructions.

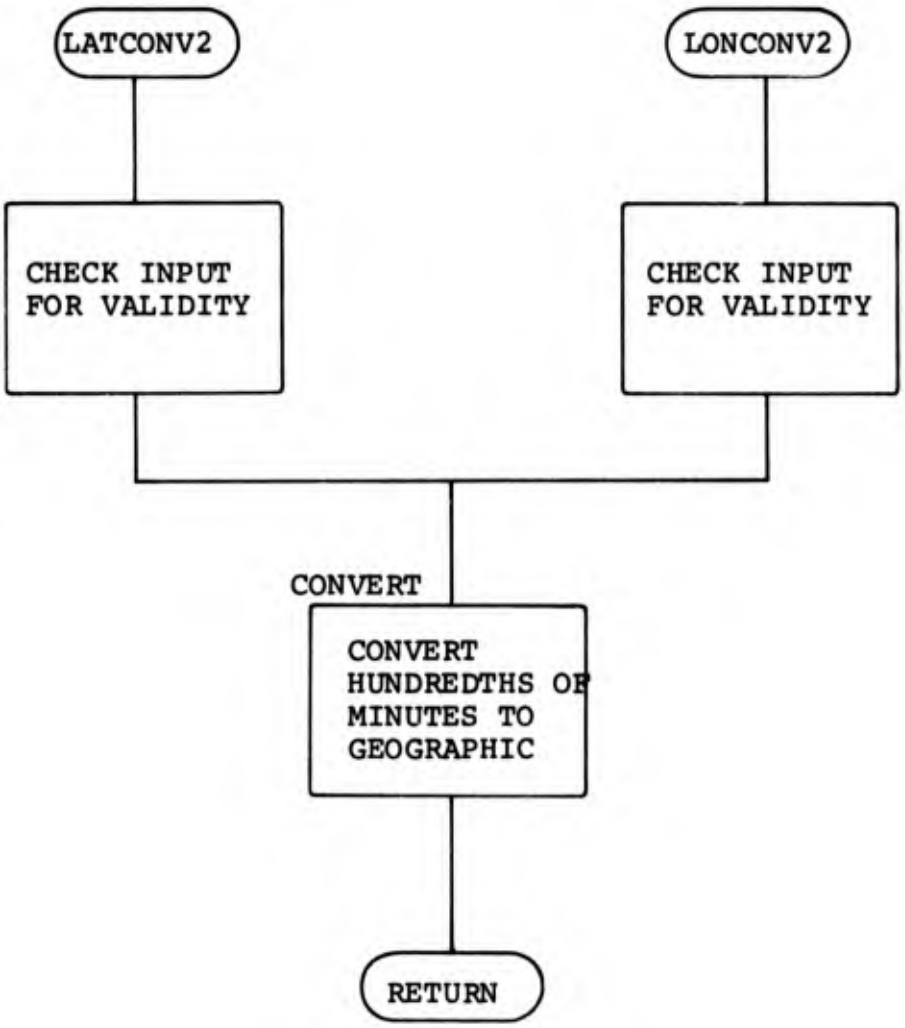
PROGRAM VARIABLES:

DWORD - An 8-byte field used as work area in conversion.

MASK1 - Area used to hold MASKA in edit process.

SUBROUTINES CALLED: None.

LATCONV2: Logic Diagram



PROGRAM ID: IIMNLL
PROGRAM NAME: Set up to Convert Minutes to Latitude and Longitude

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: IIMNLL sets up an area for calls to LATCONV2 and LONCONV2 in order to convert a pair of coordinates in minutes to the appropriate geographic coordinates.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: In order to make use of the two minutes to geographic coordinate conversion routines, the IOC routines must adhere to the conventions in these routines, which call for the minutes input to be placed in an area offset from a full-word. The set-up routine, IIMNLL, was written in order to make LATCONV2 and LONCONV2 accessible to all IOC routines, whether they be FORTRAN or Assembly program.

LOGIC NARRATIVE:

IIMNLL - Registers are saved, the input parameters are accessed and code is set to make sure no slashes are generated by conversion routines. The minutes to be converted are moved to ICOORD+9 and LATCONV2 is called to do the conversion. If errors occurred, go to ERRRET. Otherwise, save the latitude result, and do the same processing for longitude conversion.

RETURN - Return to the calling program.

ERRRET - Set error code to indicate error, then go to RETURN.

CALLING SEQUENCE: CALL IIMNLL(LATLON,ILAT,ILONG,IRSULT)

where:

LATLON will be a string of 16 bytes as follows:
DDMMSSH - 7 bytes for latitude
ø - 1 blank
DDDMMSSH - 8 bytes for longitude

ILAT is minutes and hundredths of latitude

ILONG is minutes and hundredths of longitude

IRSULT is a return code as follows:
0 = all ok
1 = error during conversion attempt.

OPTIONS: None.

CALLED BY: IOC applications programs as required.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. The inputs are ILAT, and ILONG.

OUTPUTS: See CALLING SEQUENCE. The outputs are LATLON and IRSULT.

ENTRY POINTS:

IIMNLL

PROGRAM CONSTANTS: None.

PROGRAM VARIABLES:

ICOORD - This field of 8 bytes will contain the result of the call to LATCONV2 and LONCONV2.

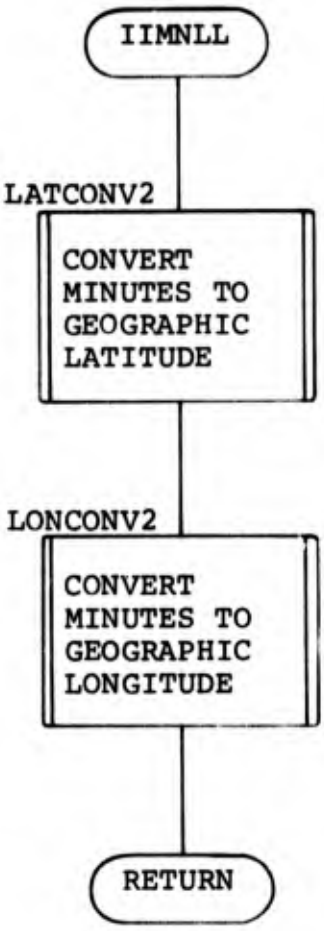
RESULTS - This 6 byte field will hold the following contents, and is used as input to LATCONV2 and LONCONV2.
Offset - 1 byte set to F0
Minutes - Minutes to be converted
Slash Code - Set to F0 by IIMNLL to indicate that no slashes are desired.

ERRFLG - Flag value from LATCONV2 and LONCONV2 for error:
0 = no error
1 = there is an error

ERROR CONDITIONS: None.

SUBROUTINES CALLED: LATCONV2, LONCONV2

IIMNLL: Logic Diagram



PROGRAM ID: IILLMN
PROGRAM NAME: Set up to Convert Latitude/Longitude to Minutes

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This routine provides the interface between FORTRAN routines and the conversion routines CONVLAT2 and CONVLON2.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: In order to convert coordinates from their geographic representation to the appropriate number of minutes, IOC routines make use of the two Assembly routines, CONVLAT2 and CONVLON2. However, these two routines return results in an area which is offset one byte from a full-word boundary. In order to make these two functions easily available to IOC programs, the interface routine IILLMN was written. It allows input to CONVLAT2 and CONVLON2 to be entered in a convenient form, and returns the results of the conversion in a standard manner.

LOGIC NARRATIVE:

IILLMN - This routine will make use of CONVLAT2 and CONVLON2 to convert latitude and longitude in geographic coordinates to minutes. It first moves the 7-byte latitude field to ICOORD, right-justified in the 8-byte field. Then CONVLAT2 is called. On return, if an error is indicated by the byte at ICOORD+14, the routine goes to ERRRET to set the error code. Otherwise, the result is moved from ICOORD+9 and a similar procedure is followed for the longitude field conversion, with the exception that the longitude field is an 8-byte field and CONVLON2 is called.

RETURN - The error code is stored and control returns to the calling program.

ERRRET - The error code is set to indicate an error. Then control passes to RETURN.

CALLING SEQUENCE: CALL IILLMN(LATLON,ILAT,ILONG,IRSULT)

where:
LATLON is a string of 16 bytes as follows:
DDMMSSH - 7 bytes for latitude
Ø - 1 blank
DDDMMSSH - 8 bytes for longitude
ILAT is the minutes and hundredths returned for latitude
ILONG is the minutes and hundredths returned for longitude
IRSULT is the return code: 0 = no error during conversion
1 = error occurred during conversion

OPTIONS: None.

CALLED BY: IOC routines which require a geographic coordinate to minutes conversion.

INPUTS: See discussion under CALLING SEQUENCE. LATLON is the input.

OUTPUTS: See discussion under CALLING SEQUENCE. ILAT, ILONG, and IRSULT are the outputs.

ENTRY POINTS:

IILLMN

PROGRAM CONSTANTS: None.

PROGRAM VARIABLES:

ICOORD - This field of 8 bytes will be the argument to CONVLAT2 and CONVLON2, and will contain the geographic coordinates.

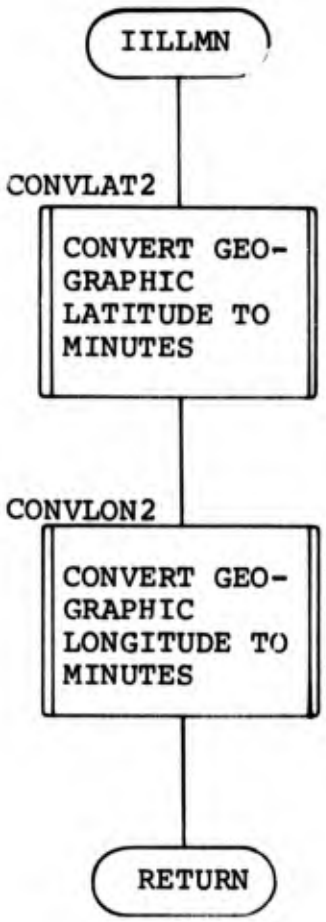
RESULTS - This 6 byte field will hold the following results from CONVLAT2 and CONVLON2:
offset - a 1-byte field containing F0
result - a 4-byte field with minutes
slash code - a 1-byte field indicating as follows: F0 = no slash present
F1 = 2nd slash present
F2 = 1st slash present

ERRFLG - Flag value from CONVLAT2 and CONVLON2 as follows:
1 = error in input
0 = no error

ERROR CONDITIONS: None.

SUBROUTINES CALLED: CONVLAT2, CONVLON2

IILLMN: Logic Diagram



PROGRAM ID: CONVLAT2
PROGRAM NAME: Convert Latitude and Longitude to Hundredths of Minutes

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: This routine takes an 8-byte gecgraphic latitude or longitude and converts it to hundredths of minutes. It edits and checks the input to ensure that each digit is in the correct range. It handles slashes in the seconds field.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Many IOC applications programs deal with geographic coordinates. Under most circumstances, operations with geographic coordinates in their usual format are unwieldy. In order to alleviate this problem, and in order to have the coordinates in the same format as in certain MODS applications, the routine CONVLAT2 was implemented. It simply converts the alphanumeric geographic coordinates into the corresponding number in hundredths of minutes.

LOGIC NARRATIVE:

CONVLAT2- Registers are saved and initialized. An error flag is set to 1. Then, a check is made to see if degrees are greater than 90. If so, return by branching to RETURN. Then check if hemisphere is N or S. If not, RETURN.

COMMON - This is the code common to both CONVLAT2 and CONVLON2, and is the actual processing of the geographical data. Check if all digits, other than possible /'s in the seconds fields, are within the range 0-9. Then check if minutes and

seconds are less than or equal to 59. If any of these checks fail, branch to RETURN.

PASSED2 - The processing reaches this point if both edit checks have been passed. The degrees and minutes are converted to hundredths of minutes and added together. A slash indicator is set, in the caller's area, as follows:
F2 if 1st slash is present
F1 if 2nd slash is present
F0 if no slash is present
Then the seconds are converted to hundredths of minutes and added. Also, compensation for the hemisphere is done. The result and error code is passed back to the calling routine in the caller's area, then it branches to RETURN.

RETURN - Return to the calling program.

CONVLON2- The registers are initialized. Then, the degrees are checked to see if they are greater than 180. If so, it branches to RETURN. Otherwise, a check is made to see if the hemisphere is E or W. If not, it branches to RETURN. Otherwise, go to COMMON.

CALLING SEQUENCE: CALL CONVLAT2(ICOORD)
CALL CONVLON2(ICOORD)

where:
ICOORD is a 15-byte area which contains the following:

Byte 1-8 - Geographic coordinate to be converted. For CONVLAT2, it is of the form bDDMMSSH and H is either N or S. For CONVLON2, it is of the form DDDMMSSH and H is either E or W.

9 - Field with F0 in it.

10-13 - Result of conversion.

14 - Slash indicator. This can be:
F0 = no slash present
F1 = 2nd slash present
F2 = 1st slash present

15 - Error flag. 1 = error
0 = no error

OPTIONS: None.

CALLED BY: IILLMN

SPECIAL NOTES: Latitude 0 ≤ degrees ≤ 90
0 ≤ minutes,sec ≤ 59
Longitude 0 ≤ degrees ≤ 180
0 ≤ minutes,sec ≤ 59

Also, slashes can occur only in the seconds field. If one slash occurs, the second digit is multiplied by 10 to get the proper seconds. If two slashes occur, the seconds field is ignored.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. Bytes 1-8 are input to the routine.

OUTPUTS: See CALLING SEQUENCE. Bytes 9-15 are output by the routines.

ENTRY POINTS:

CONVLAT2
CONVLON2

REGISTER USAGE:

R2 - Points to the beginning of ICOORD

R3 - Points to current position in ICOORD, when editting. Later used in conversion.

R15 - Base register.

PROGRAM CONSTANTS: None.

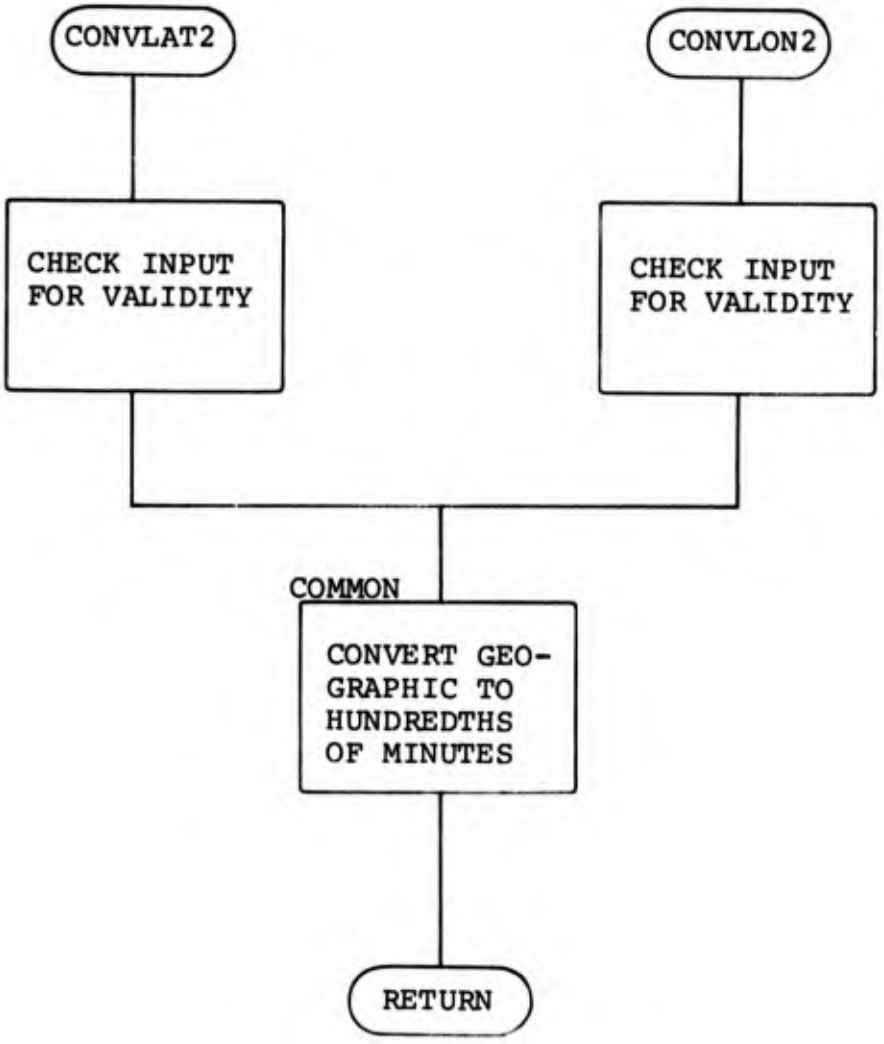
PROGRAM VARIABLES:

DWORD - An 8-byte field used as a work area in the conversion.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

CONVLAT2: Logic Diagram



PROGRAM ID: GENRC
PROGRAM NAME: Generate Direct Access BR90 Records

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: This program generates BR90 screen images and stores them in direct access records so they may be used by the IOC routines. It generates screens from punched cards.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: IOC applications programs which interact with the user frequently use fixed screens, especially for showing the user procedures. Rather than have each routine generate and maintain its own screen images, IOC maintains a library of screens. GENRC is the program which generates this library.

LOGIC NARRATIVE:

5 - Clear the screen buffer so that any lines with no information will be left blank. Read control card. Check if columns 77-80 are blanks. If not, go to 9999. Otherwise, set record number where screen will go, and get card count. If card count or record number is beyond limits, go to 9999. Otherwise, read cards with line images, and, if the line number is within the limit, fill in the screen buffer. If it is not within the limit, go to 8500. Otherwise, write out the buffer, and go to 5.

8500 - Print line error message, with line number.

9999 - Loop through all input cards, then end the routine.

CALLING SEQUENCE: None. (Batch Program)

OPTIONS: None.

CALLED BY: None. GENRC is a main program.

SPECIAL NOTE: A maximum of 100 screens may be created for IOC. This limit should be changed for future applications.

Card columns 77-80 must be blank on the control cards.

The valid number of cards is 44 per set.

SCREENS USED: None.

INPUTS: GENRC uses punched cards as input. Input consists of sets of cards, each set corresponding to one screen image. The format of the set is:

CARD 1:	Column	7-10	Record number
		11-20	Card count
		77-80	Blanks
CARD 2,etc:		1-64	Screen line image
		79-80	Screen line, bottom of screen is line 1.

OUTPUTS: GENRC outputs the screen images on data set 8, starting at record 161. It also prints each screen on the line printer.

PROGRAM CONSTANTS:

IBLANK - Contains '$\not\beta\not\beta\not\beta\not\beta$', used to compare for all blank field.

PROGRAM VARIABLES:

LNUM - Line number on the screen. Punched in columns 79-80 of the card.

NCARDS - Number of cards in the set, binary.

CRDCNT - Number of cards in the set, decimal.

IREC - Record number of the screen.

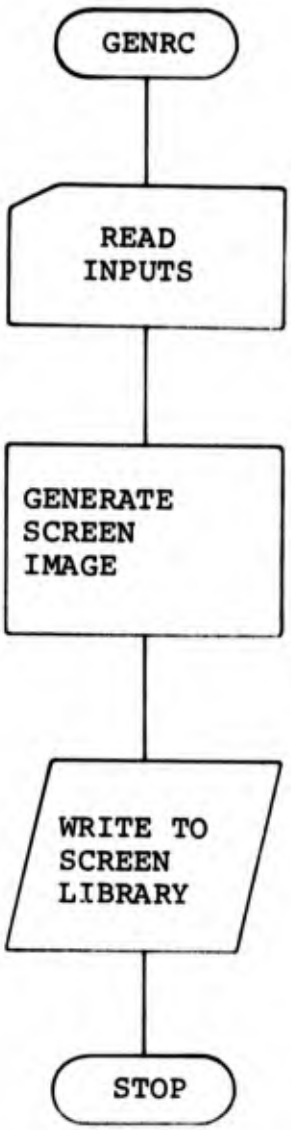
NUM - Screen line number, binary.

IBUF, IARRAY - Screen buffer. IARRAY is one-dimensional, while IBUF is 2-dimensional.

SUBROUTINES CALLED: ICDTB

*See SCREEN for on-line screen maintenance.

GENRC: Logic Diagram



PROGRAM ID: GENSLIDE
PROGRAM NAME: Generate Slide Coordinate List and Slide Screens

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: This program creates the slide list and screens from cards for use by IOC. In conjunction with SLIDER, this program allows full maintenance of the slide data areas.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Under the original concept, IOC would have a facility for maintaining and utilizing slides. In order to utilize the slides in the system, some method of correlating slides with screen positions had to be developed. Thus, the slide coordinate list was included in the system. In order to generate the list, GENSLIDE was implemented. Originally, it was decided to allow input of coordinates for any point on the lower left corner and upper right corner of the slide area. However, later it was found that it would be more desirable to have the origin points at the corners in the slide lists. Thus, although input from cards can be for any two corner points, GENSLIDE will extrapolate these to the farthest corners. This is done by first finding a latitude factor and longitude factor (minutes/rasters). Using these factors, the geographic coordinates of the points of (0,592) and (4096,5456) are easily found.

LOGIC NARRATIVE:

GENSLIDE - Initialize output record pointers, and initialize screen image BUFFER to blanks. Add header lines and procedure lines to BUFFER.

1500 - Prepare to begin slide list printout.

10 - Clear out area for slide coordinate list.

20 - Read slide input card. If there are blanks in the lower left latitude field, go to 7372. Otherwise, convert lower left and upper right coordinates to minutes and hundredths. Convert the cursor coordinates to the appropriate raster units. Using these values, extrapolate the geographic coordinates to the farthest corners of the slide area.

7474 - Print next line of the slide list. Add lower left and upper right geographic coordinates to the slide screen. If the current slide screen is full, go to 50. Otherwise, go to 5.

5 - Check if the current block of the slide coordinate list is full. If not, go to 20. Otherwise, write out the block and go to 10.

77 - When an end-of-file is encountered on the input deck, control passes to this point. Here, if there are no entries in the current block of the slide list, go to 85. If there are, write out the last block. If there aren't any lines in the current slide list, go to 80. Otherwise, write out the slide screen.

80 - Read the first block of the slide list, and add the final count of the number of blocks (JRECNO) and the number of slides in the last block (JNDX) to the block. Write the block out. Print the count of the number of slides processed. Halt execution.

85 - Set the count of the number of entries in the current block to 99, and go to 80.

7372 - Increment counts, go to 7474.

7373 - Set error flag, go to 7474.

50 - Write out slide screen, clear out screen image, and go to 5.

CALLING SEQUENCE: None. GENSLIDE is a batch program.

OPTIONS: None.

CALLED BY: None. This is a batch program.

SCREENS USED: None.

INPUTS: As mentioned earlier, GENSLIDE uses card inputs. Each card has the following format:

Columns	
1-10	Slide identification
11-14	Lower left x-cursor coordinate in octal
15	∅
16-19	Lower left y-cursor coordinate in octal
20	∅
21-28	Lower left latitude (DDMMSSH∅)
29-30	∅∅
31-38	Lower left longitude (DDDMMSSH)
39-40	∅∅
41-44	Upper right x-cursor coordinate in octal
45	∅
46-49	Upper right y-cursor coordinate in octal
50	∅
51-58	Upper right latitude (DDMMSSH∅)
59-60	∅∅
61-68	Upper right longitude (DDDMMSSH)
69-70	∅∅
71-80	Slide number

OUTPUTS: GENSLIDE produces output in three different forms:

1) Slide screen images on data set 8 starting at record 161. The third line contains the heading:
'IOC SLIDE LIST'
Then, 33 lines of slide data follow:

Columns	
1-12	Slide identification
13-24	Slide number
25-32	Lower left latitude (DDMMSSH∅)
33-40	Lower left longitude (DDDMMSSH)
41-44	∅∅∅∅
45-52	Upper right latitude (DDMMSSH∅)
53-60	Upper right longitude (DDDMMSSH)

Note that in the slide screen, the latitudes and longitudes are not the extrapolated values. The last three lines of the screen have procedures on them.

2) A print-out of the slide list. The first print line contains:
'LIST OF COORDINATE CARDS'
The rest of the listing consists of lines in the following format:

Columns

1-10	Slide identification
11-14	Lower left x-cursor coordinate in octal
15	∅
16-19	Lower left y-cursor coordinate in octal
20	∅
21-28	Lower left latitude (DDMMSSH∅)
29-30	∅∅
31-38	Lower left longitude (DDDMMSSH)
39-40	∅∅
41-44	Upper right x-cursor coordinate in octal
45	∅
46-49	Upper right y-cursor coordinate in octal
50	∅
51-58	Upper right latitude (DDMMSSH∅)
59-60	∅∅
61-68	Upper right longitude (DDDMMSSH)
69-70	∅∅
71-80	Slide number
81-86	∅∅∅∅
87-90	Error flag or ∅∅∅∅

The last line contains a count of the cards processed.

3) The slide coordinate list is generated on data set 8 starting at record 111. Each record consists of two tables. The first table contains 99 entries in the following format:
(lower left longitude, lower left latitude, upper right longitude, upper right latitude)
All coordinates in this table are in minutes and hundredths, and are extrapolated values.

The second table contains 99 entries of the following format:
(lower left x-cursor coordinate, lower left y-cursor coordinate, upper right x-cursor coordinate, upper right y-cursor coordinate)
All cursor coordinates are in rasters, and are extrapolated values. The first block of this data set contains JRECNO and JNDX as the last two words.

PROGRAM CONSTANTS:

I111 - Contains 111, the beginning record number for the slide coordinate list.

IERRMS - Contains '*ERR', the error flag on the slide list print-out.

IHEAD - Contains 'IOC SLIDE LIST', the heading for slide screens.

LINE39 - Contains 'LIGHT PEN SLIDE, THEN PRESS -PROCEED-, OR'

LINE40 - Contains '&---& TYPE IN NUMBER AND PRESS -PROCEED- &-&'

LINE42 - Contains 'NOTE - USE -PAGE FORWARD-/-PAGE BACKWARD- KEYS TO SCAN SLIDE LIST'

IBLANK - Contains 'ØØØØ'

PROGRAM VARIABLES:

INDX - Number of entries in current block of slide coordinate list.

NCOUNT - Count of number of lines on current slide screen.

ITOTAL - Current total of cards processed.

IRECNO - Current record number of slide coordinate list.

NIREC - Current record of slide screen.

IXYTBL - Current block of slide coordinate list for x,y coordinates.

ITABLE - Current block of slide coordinate list for geographic coordinates.

IT - Array used in extrapolating ITABLE values.

LATF - Calculated factor used to extrapolate latitude.

LONGF - Calculated factor used to extrapolate longitude.

ISAV1 - Array for slide identification.

ISAV2 - Array for slide number.

LLVL - Contains 8 digits of lower left x and y octal values.

IUVL - Contains 8 digits of upper right x and y octal values.

LLLAT - Lower left latitude.

LLLONG - Lower left longitude.

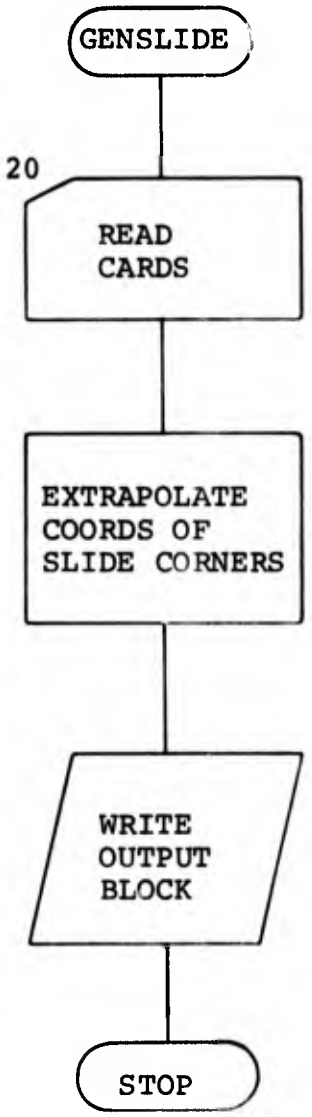
IURLAT - Upper right latitude.

IURLNG - Upper right longitude.

ERROR MESSAGES: Any slides which had an error in geographic coordinates will have an '*ERR' printed in the rightmost column.

SUBROUTINES CALLED: ILLMN, FILL

GENSLIDE: Logic Diagram



PROGRAM ID: SUPPMNTB
PROGRAM NAME: Supplemental FFT Batch Maintenance

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: COBOL
COMPUTER: IBM 360, Model 40
FUNCTION: SUPPMNTB reads the input transaction cards and executes the specified update functions of loading a new Supplementary FFT file or updating the existing one.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: SUPPMNTB updates the Supplemental FFT of IOC-supported files in a batch mode. The user may create a new Supplemental FFT file or update an existing Supplemental FFT file. The print option enables the user to request a print-out of the Supplemental FFT file.

The program accomplishes the update by copying the old Supplemental FFT (SUPP FFT1) to create a new one (SUPP FFT2). Changes and Deletions are processed during the copying process. Adds are processed after SUPP FFT1 has been completely processed. Adds are simply added to the end of SUPP FFT2. When all of the transactions have been processed, SUPP FFT2 is copied back to SUPP FFT1.

Additional information on the Supplemental FFT file can be found in the PHILOSOPHY AND APPROACH for program SUPPMNTN.

LOGIC NARRATIVE:

001-HOUSEKEEPING
Open the card input file and read a card. If the transaction is a Load, open SUPP FFT2 and go to 002-LOAD-FILE. If the transaction is an Update, open SUPP FFT1 (input) and SUPP FFT2 (output) and go to

005-UPDATE-FILE. If the transaction is Print, open SUPP FFT1 and the print file, and go to 040-PRINT-ONLY.

002-LOAD-FILE
Read the next card. If it is a Print request, open the print file, otherwise go to 002B-CONTINUE.

002A-CONTINUE
Read the next card.

002B-CONTINUE
If this is an End card, go to 096-TEST-STATUS. Otherwise add the transaction to SUPP FFT2 and go to 002A-CONTINUE.

005-UPDATE-FILE
Read the next card. If it is a Print request, open the print file. Otherwise, go to 005B-CONTINUE.

005A-CONTINUE
Read the next card.

005B-CONTINUE
If this is an End card, go to 096-TEST-STATUS. Otherwise, go to one of the following:

ADD	-	go to 006-FIRST-ADD-TEST
DELETE	-	go to 020-DELETE-ITEM
CHANGE	-	go to 030-CHANGE-ITEM.

006-FIRST-ADD-TEST
If this is not the first Add transaction, go to 010-NEW-ADD. Otherwise, copy SUPP FFT1 to SUPP FFT2, then go to 095-TEST-STATUS.

010-NEW-ADD
Add the transaction to SUPP FFT2 and go to 005A-CONTINUE.

020-DELETE-ITEM
Search SUPP FFT1 for the record to be Deleted by reading a record, comparing record ID's and copying the records to SUPP FFT2 until a match is found and go to 005A-CONTINUE.

030-CHANGE-ITEM
Search SUPP FFT1 for the record to be Changed by reading a record, comparing record ID's and copying the records to SUPP FFT2 until a match is found. Then change the record, write it to SUPP FFT2 and go to 005A-CONTINUE.

040-PRINT-ONLY
Read SUPP FFT1 and print the records. At EOF, go to 095-TEST-STATUS.

050-SET-RETURN
Return to the calling program.

091-EOJ
If the option processed was a Load or an Update, write an EOF record to SUPP FFT2. Copy SUPP FFT2 to SUPP FFT1, close both files, reinitialize flags and counters and return to the calling program.

095-TEST-STATUS
If end of transactions, go to 091-EOJ, otherwise indicate EOF on SUPP FFT1 and go to 010-NEW-ADD.

096-TEST-STATUS
If the option was Load, go to 091-EOJ. If the option wasn't Update, go to 091-EOJ. If EOJ on SUPP FFT1, go to 091-EOJ. Otherwise, set end of transactions flag and go to 006-FIRST-ADD-TEST.

CALLING SEQUENCE: None. This is a batch program which is executed when the following control card is encountered:
// EXEC IISUPLMN

OPTIONS: The options are specified in the input cards. (See INPUTS.)

CALLED BY: Batch program.

SCREENS USED: None.

INPUTS: The program accepts transactions in the following card format:

Card Column	Contents
1	Transaction Type
2 - 4	File Identity
5 - 9	Field Mnemonic
10 - 11	Field Identity
12 - 31	Descriptive Name
32 - 47	Selection Criteria (Ignored by IOC)
48	Translation Table Flag
49 - 80	Not Used

The following are the valid transaction type codes:
L = Create a new Supplemental FFT file.
U = Update the existing Supplemental FFT file.
P = Print only (if this is in the first transaction card) or update and print (if this is the second transaction card).
A = Add this item to the Supplemental FFT file.
D = Delete this item from the Supplemental FFT file.

C = Change this item on the Supplemental FFT file.
E = End of transactions.

OUTPUTS: The outputs are the updated Supplemental FFT file on disk and the print-outs if they have been requested.

ENTRY POINT:

SUPPMNTB

PROGRAM CONSTANTS:

PAGE-RESET	- Number of print lines per page. Value is 48.
INIT-PAGE	- Page number for first pass through print routine. Value is 1.
PRT-HEADER	- Supplemental FFT header for each print page.
DATA-HEADER	- Field names of Supplemental FFT items used in print output.
MSG-ADD	- "*****ADDED*****" for print output.
MSG-DEL	- "*****DELETED***" for print output.
MSG-CHG	- "*****CHANGED***" for print output.

PROGRAM VARIABLES:

PRINT-IMAGE	- Buffer for print lines.
IN-REC	- Input area for records from SUPP FFT1.
OUT-REC	- Output buffer for records for SUPP FFT2.
PRT-SWITCH	- Switch to indicate whether records are to be printed. P means print the records. Ø means no print requested.
FIRSTADDSW	- Switch to indicate first Add Transaction. N means not the first Add. Y means first Add so SUPP FFT1 must be copied to SUPP FFT2 before processing the transactions.
OPTIONHOLD	- Contains one of the following option codes: L - Load new Supplemental FFT, destroying the previous contents. U - Update the Supplemental FFT. P - Print existing file (no update).

DISKINEOF - EOF indicator for SUPP FFT1. Values are Y and N.

CARDSINEOF - End-of-transactions Indicator. Values are Y and N.

PAGE-LINES - Number of print lines remaining on a page.

CARD-IN-WORK- Work area for transaction card.

DISK-IN-WORK- Work area for SUPP FFT1 record.

DISK-OUT-WORK-Work area for SUPP FFT2 record.

DATA-PRINT-AREA- Work area for print line.

ERROR MESSAGES:

ERRMSG1 - "INVALID PROCESSING OPTION SELECTED". Option was other than L, U, or P.

ERRMSG2 - "TRANS CODE NOT A, D, OR C--IGNORED". Invalid transaction code specified. It was ignored.

ERRMSG3 - "ADDITION ATTEMPTED BEFORE EOF OLDFILE". SUPP FFT1 must be copied to SUPP FFT2 before any Add transactions can be processed.

ERRMSG4 - "DELETION ATTEMPTED AFTER EOF OLDFILE". Deletions cannot be processed after an EOF is encountered on SUPP FFT1.

ERRMSG5 - "DELETION TRANS NOT ON OLD FILE--TERMINATE". The record to be deleted cannot be found on SUPP FFT1.

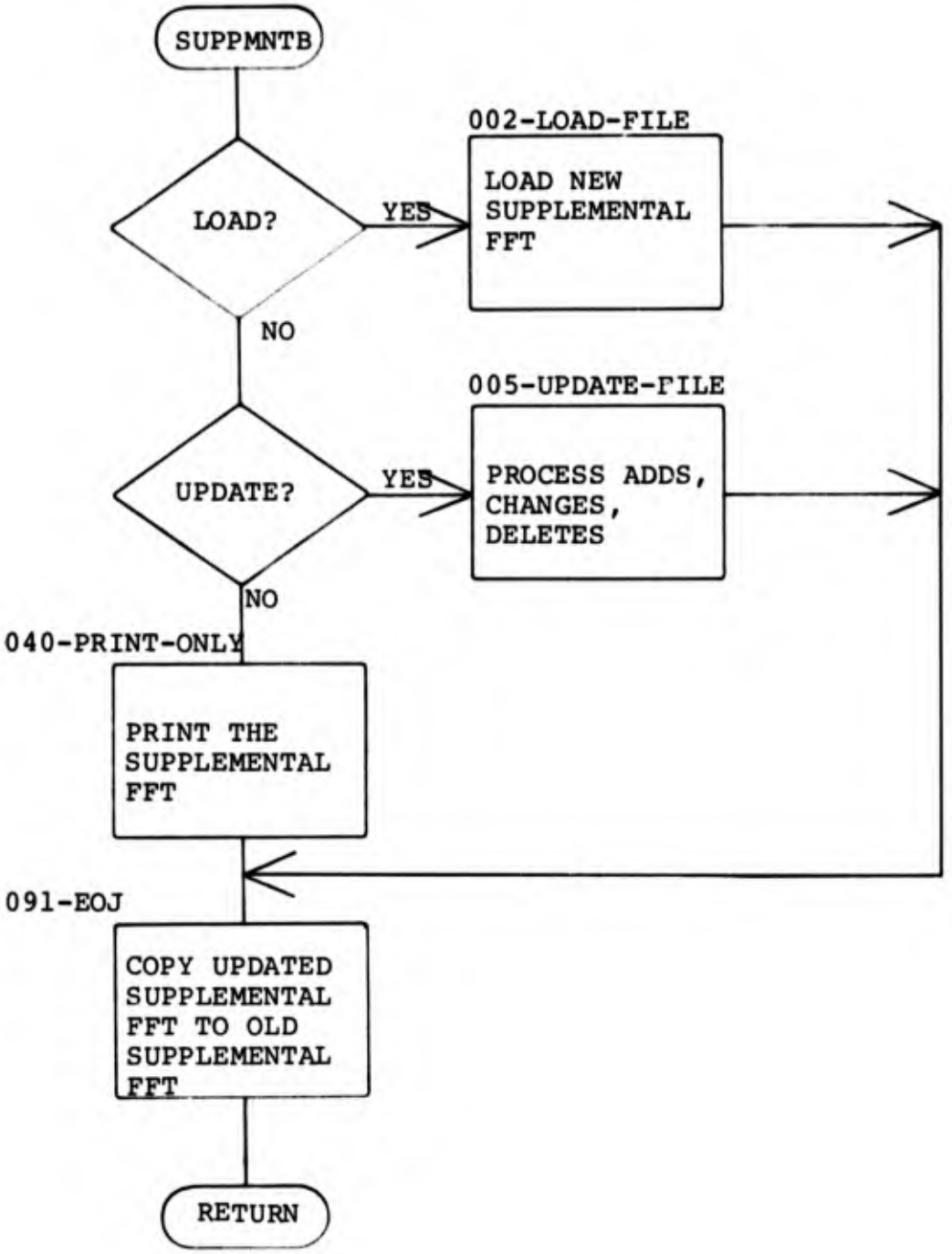
ERRMSG6 - "CHANGE ATTEMPTED AFTER EOF OLDFILE". Changes cannot be processed after an EOF is encountered on SUPP FFT1.

ERRMSG7 - "CHANGE TRANS NOT ON OLD FILE--TERMINATE". The record to be changed cannot be found on SUPP FFT1.

SPECIAL NOTE: The Load option should be used only for the initial generation of the Supplemental FFT because the program assumes that no old Supplemental FFT exists when this option is selected. If the Supplemental FFT for a new file is to be added, it should be done via the Update option.

SUBROUTINES CALLED: None.

SUPPMNTB: Logic Diagram



PROGRAM ID: TRTBMNTN
PROGRAM NAME: Translation Table Maintenance

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: COBOL
COMPUTER: IBM 360, Model 40
FUNCTION: This batch program creates the Translation Table data sets for transactions that have been punched on cards.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Translation Tables are provided in IOC in order to make the system more readily understandable to the untrained user. The Translation Tables, which are associated with each IOC-supported MODS file, provide for translating the actual codes in the file records to clear English equivalents. These tables are maintained by a batch program called TRTBMNTN.

TRTBMNTN processes the Translation Table data sets (one data set for each IOC-supported MODS file) and the associated Translation Table directory. The program assumes that all translation items for a given file are collected under one data set and that a 2000 character block is allocated to each field mnemonic requiring translation items. No more than 25 translation items may be established for a single field mnemonic in a given file. The maintenance program has options to create a new Translation Table data set from card input, and print out the Translation Tables as they are being created.

The translation tables to complement data files used for IOC development will be maintained as separate data sets - one such data set for each file referenced. In addition, a separate data set known as the Translation Table Directory, will be maintained. It will serve as an index to the actual translation table data sets.

The directory items (see format below) will each be 13 characters in length, with 153 items contained in each 2000 character block.

3 bytes	5 bytes	3 bytes	2 bytes
FILE IDENTITY	FIELD MNEMONIC	BLOCK NO. WITHIN DATA SET	ITEMS IN BLOCK USED

FILE IDENTITY = Three character file identity (IOC data base) used for calling access.

FIELD MNEMONIC = Five character FNT name, from the FFS Field Name Table.

BLOCK NO. WITHIN DATA SET = Three character pointer (a numeric value) used to indicate which block (within a file identity group of blocks) contains the items for the field mnemonic referenced.

ITEMS IN BLOCK USED = Two character numeric value indicating how many 80 character slots have been used, out of a maximum of 25. Each 2000 character block in the actual data base translation tables is devoted to a single field mnemonic. Unused portions of the block (occurs when less than 25 translation items are used for a given field mnemonic) will be set to blanks.

The translation items (see format below) will each be 80 characters in length, with 25 items contained in each 2000 character block.

4 bytes	5 bytes	2 bytes	1 byte	64 bytes	1 byte	3 bytes
FILE IDENTITY	FIELD MNEMONIC	SUBSET CONTROL	∅	BR90 SCREEN ENTRY	PLOT	∅∅∅

FILE IDENTITY = Three character file identity used for calling access, plus one character filler (FORTRAN compatibility).

FIELD MNEMONIC = Five character FNT name, from the FFS Field Name Table.

SUBSET CONTROL = Two character identifier to distinguish multiple item entries for the same data element. Value must be $\geq 01 \leq 25$.

BR90 SCREEN ENTRY = Sixty-four characters of textual information describing the data element to the user.

PLOT = One character plotting descriptor, related to text which precedes it.

NOTE: Characters indicated as ∅ are unused.

LOGIC NARRATIVE:

001-HOUSEKEEPING
Open the input card file and read a card.

001A-CONTINUE
If Load option go to 002-LOAD-FILES. If Update or Print options, no code provided so eventually end up at 096-TEST-STATUS.

002-LOAD-FILES
Read a card. If Print requested, open the print file, and read the next card.

002A-CONTINUE
If Translation Table transaction go to the specific file process (in this case go to 010-TRNAAA-PROCESS which is typical for all files) otherwise, go to 005-TRNDRC-PROCESS.

005-TRNDRC-PROCESS
Open the directory file, process all of the directory transaction cards and create the directory. When an End card is read, write an END record and close the directory file. Read the next card. If it is an End card, go to 096-TEST-STATUS, otherwise go to 001A-CONTINUE.

010-TRNAAA-PROCESS
Open the Translation Table data set for the data file, process all of the transaction cards for the file and create a Translation Table. Each field mnemonic will occupy one block of 2000 characters. If the block is not filled with data, the unused portion is filled with asterisk records. When the End card is read, write an END record and close the file. Read the next card. If it is an End card, go to 096-TEST-STATUS, otherwise go to 001A-CONTINUE. This process is basically the same for all other files.

091-EOJ
Close the card file. If there were any print-outs close the print file. Then STOP RUN.

096-TEST-STATUS
Set the EOF indicator and go to 091-EOJ.

CALLING SEQUENCE: None. This is a batch program which is executed when the following control card is encountered:
// EXEC IITRTBMN.

OPTIONS: The options are specified in the input cards. (See INPUTS).

CALLED BY: Batch program.

SCREENS USED: None.

INPUTS: The maintenance program accepts the following two types of transaction formats. The transaction items are explained in detail in the PHILOSOPHY AND APPROACH.

Translation Table Transaction Format

Card Column	Content
1	Transaction Type P = Print the file L = Create new file; no old file E = End of transactions
2-4	File Identity Acceptable values are AAA, SAM, EOB, AOB, ERA, and AAC
5-9	Field Mnemonic
10-11	Subset Control
12-75	BR-90 Screen Entry
76	Plot character
77-80	Not Used

Directory Transaction Format

Card Column	Content
1	Transaction Type L = Create new file (only option available)
2-4	"DRC"
5-9	Field Mnemonic
10-12	Block within data set
13-14	Items in Block used
15-17	File Identity of data file item
18-80	Not Used

OUTPUTS: The outputs are the Translation Table directory and data sets, and hard copy output if requested.

ENTRY POINTS:

TRTBMNTN

PROGRAM CONSTANTS:

BLOCKTALLYRSTR	-	Number of records per block. Value is 25.
PAGE-RESET	-	Number of lines per page. Value is 48.
PRT-HEADER	-	Header line for print output.
DATA-HEADER	-	Header line for Translation Table field names.
PRT-DRC-HRD	-	Header line for Directory field names.
SENTL-RECORD	-	Last record indicator for 25-record block. Value is 9 asterisks.
END-RECORD	-	Last record indicator used at end of each file.
DRC-END-REC	-	Last record indicator for directory.

PROGRAM VARIABLES:

CARD-IMAGE	-	Card input buffer.
PRINT-IMAGE	-	Print output buffer.
OUT-REC-1	-	Directory buffer area.
OUT-REC-2	-	AAA buffer area.
OUT-REC-3	-	SAM buffer area.
OUT-REC-4	-	EOB buffer area.
OUT-REC-5	-	AOB buffer area.
OUT-REC-6	-	ERA buffer area.
OUT-REC-7	-	AAC buffer area.
OPTIONHOLD	-	Transaction Type.
FILEIDHOLD	-	Current file name.
FLDMNHOLD	-	Current field mnemonic.
BLOCKTALLY	-	Records remaining in block.

PRT-SWITCH - Contents are:
P = Print the records for the file
Ø = No print-outs

PAGE-LINES - Print lines remaining on page.

DISK-WORK-DIRC - Work area for directory record.

DIRC-TRANS - Work area for directory transactions.

TEMP-CARD-1 - Work area for transaction card.

DISK-OUT-WORK - Work area for Translation Table record.

DATA-PRINT-AREA - Print line for Translation Table elements.

DRC-PRINT-AREA - Print line for Directory data.

ERROR MESSAGES:

ERRMSG1 - 'INVALID PROCESSING OPTION SELECTED'. The processing option was neither Load, Update, or Print. Run terminated.

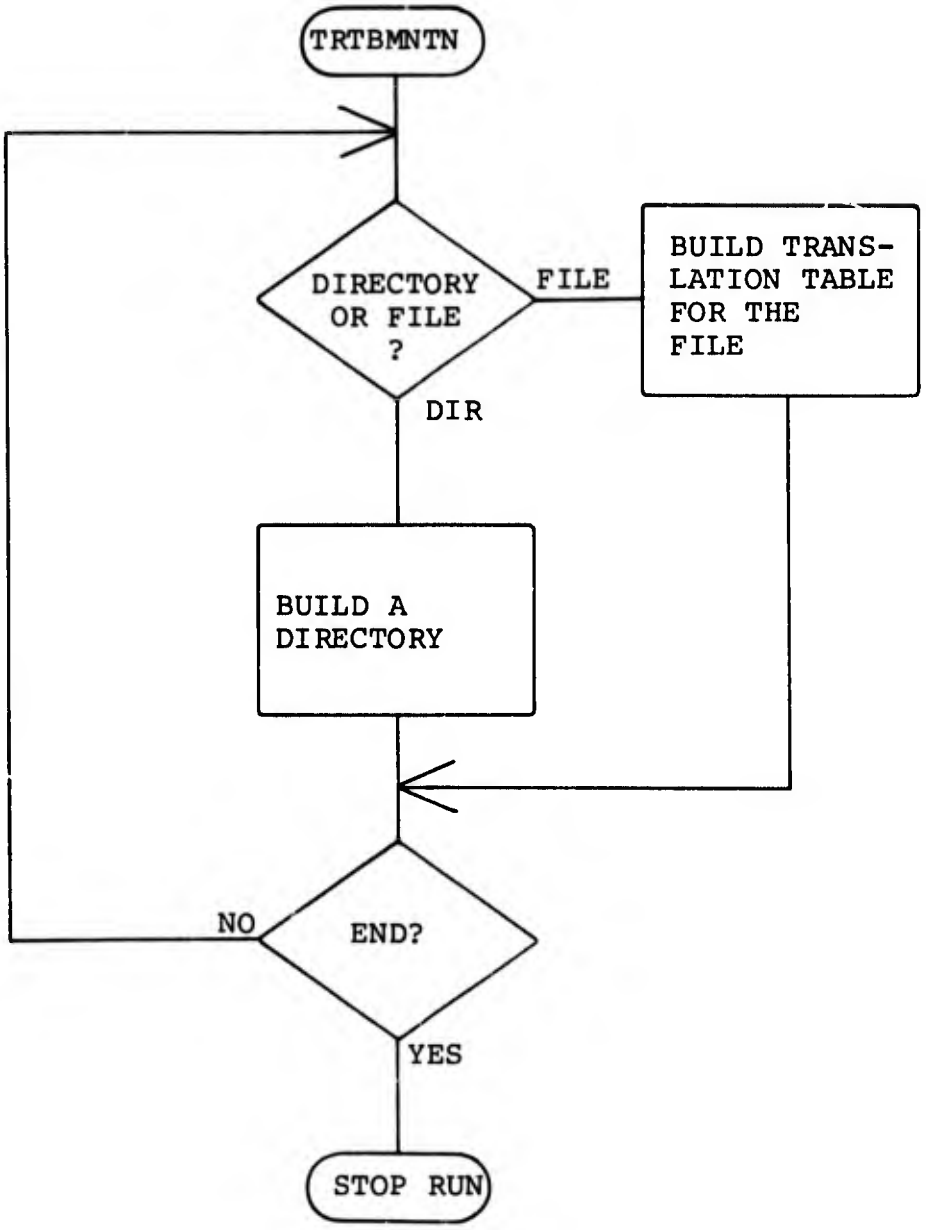
ERRMSG2 - 'INVALID FILE SELECTED'. File name on a transaction card is not an IOC-supported MODS file. Run terminated.

ERRMSG3 - 'BLOCK COUNT EXCEEDS 25 ITEMS, FILE--'. Field contains more than 25 translation items. Run terminated.

ERRMSG4 - 'FILE IDENTITY CHANGED, NO END CARD FOUND. CURRENT-ID--xxx FORMER-ID--xxx'. No End card found between transactions for two files. Run terminated.

SUBROUTINES CALLED: None.

TRTBMNTN: Logic Diagram



CHAPTER XV

IOC PROGRAM MAINTENANCE SUPPORT PROGRAMS

PROGRAM ID: COMREG
PROGRAM NAME: Communications Region Access Routine

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: COMREG returns the system date and the switch settings to the calling program.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: COMREG is an Assembly language subroutine that accesses the communications region to obtain the system date and the switch settings. It returns this data to the calling program, thus enabling FORTRAN programs to gain access to data from the communications region.

LOGIC NARRATIVE:

COMREG - Save registers and get the parameter addresses. Set List flag to 1. Issue the COMRG macro to get the pointer to the communications region. Move the system date to the caller's area. If the first bit of byte one of the switch settings is equal to zero, go to STORE. Otherwise, set List flag to 0.

STORE - Move the List flag value to the caller's area, restore the registers and return to the calling program.

CALLING SEQUENCE: CALL COMREG(DATE,LISTFL)

Where:
DATE is an 8-byte field to receive the system date.
LISTFL is the List flag. 0 = UPSI was equal to 1. 1 = UPSI

was equal to 0 or the UPSI card was not present.

OPTIONS: None.

CALLED BY: XREFIT, XREFLW

SCREENS USED: None.

INPUTS: COMREG issues a COMRG macro to get the location of the communications region. It extracts the system date and the switch settings from the communications region. (Also see CALLING SEQUENCE).

OUTPUTS: COMREG returns the system date and the switch settings to the calling program.

ENTRY POINT:

COMREG

REGISTER USAGE:

R2 - Contains List flag value.

R15 - Base register.

R3 - Pointer to receiving field for system date.

R4 - Pointer to receiving field for switch setting.

R14 - Return address.

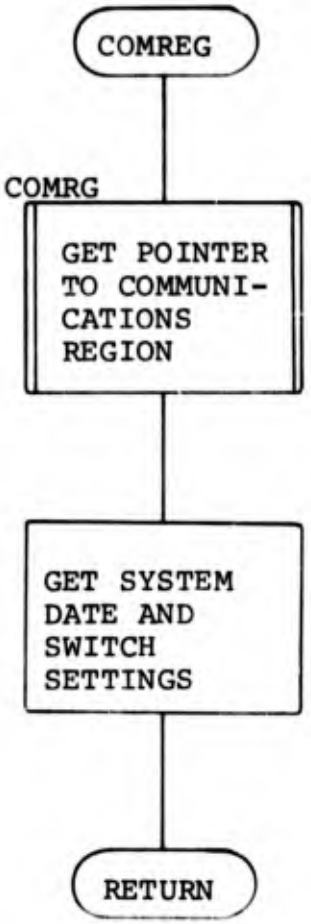
PROGRAM CONSTANTS: None.

PROGRAM VARIABLES: None.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: None.

COMREG: Logic Diagram



PROGRAM ID: DSPY (FCRFLOW)
PROGRAM NAME: Display the Final Program and Flow Lines Using the VLE's

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: DSPY is the final output phase of the flow-charting process. It outputs the card image along with the flow lines.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The form and function of the VLE's was explained earlier in the PHILOSOPHY AND APPROACH sections of MAINPGM and VLECRT. The routine DSPY uses the VLE's generated previously to display the required flow lines. As explained in VLECRT, each VLE entry consists of a VID and a VTYPE. The position of the VLE in the line image determines whether it is an exit or entry VLE. Each line is allowed one exit and five entry VLE's. DSPY sequentially processes each line image. It goes sequentially through all of the entry VLE's in the line, generating symbols in the vertical line buffer (LINEBF), by calling SETBF. The symbol entered is determined by the VTYPE of the VLE, while the print position which is modified is determined by mapping the VID into a line nesting level. This nesting level is then mapped into the actual position through a table created by MAINPGM. When the entry VLE's for one line are done, the exit VLE for the line is processed, and the appropriate number of +'s is moved into the print-time buffer (PTBF). Then, the PTBF is completed by filling it in from LINEBF. As can be seen, LINEBF maintains the current image of vertical lines, while PTBF generally contains the current horizontal line image. At actual print-time, PTBF is modified by LINEBF to get the output line.

As in the case of VLECRT, the details of DSPY (ALCFLOW) and DSPY (FORFLOW) are quite different, while the philosophies are quite similar. Most of what was said above is also true of DSPY (ALCFLOW). Several differences occur because the input to the FORFLOW process comes from cards; whereas, the input to the ALCFLOW process is from the assembler. The differences are:

- In DSPY (ALCFLOW), the print-line count is gotten from the assembler output; in DSPY (FORFLOW), the count is kept by the program.

- In DSPY (ALCFLOW), lines which never are outputted are indicated by a pair of dashes in the code area of the line image; in DSPY (FORFLOW), this situation does not occur at all.

- When DSPY (ALCFLOW) has finished printing the flow chart for the program, the cross-reference is also printed from data set 9, where it was put by the assembler; in DSPY (FORFLOW), the cross-reference is generated by another job step.

- The size and contents of the data set 9 records is slightly different. In ALCFLOW, the record format is:

bytes	1 - 72	card image
	73 - 76	entry VLE1
	77 - 80	entry VLE2
	81 - 84	entry VLE3
	85 - 88	entry VLE4
	89 - 92	entry VLE5
	93 - 96	exit VLE
	97 - 100	code, length
	101 - 103	branch address, gotten from assembler output
	104 - 107	print-line number
	108 - 111	op code, as gotten from assembler output
	112 - 115	macro indicator.

  In FORFLOW, the record format is the same as above, except only the first 100-bytes are used.

LOGIC NARRATIVE:

DSPY - Set card count (I) and print line number (LINCT) to 0.

2 - Increment I. Read the next card image from data set 9. Set a pointer to the first available print position (IBLK) in the card image. Blank out PTBF. Set count of intersections (ICOUNT) to 0.

Also, set rightmost vertical line pointer (MAX) to 0. If the current line is an END card, go to 1. If it is a continue or comment card, go to 8192. If there are any VLE's on this line, print a blank line for spacing. Increment LINCT, since this is a countable line. Loop through all the entry VLE's. When a blank VLE is encountered, go to 5. Otherwise, find the vertical nesting level for this VLE (by searching the VLETAB/BEGPT tables as set in MAINPGM), and the actual line position (by using the mapping array MAP, as set by line spacing algorithm in MAINPGM). Save the print position in the intersection table, INTRSC. Set in the appropriate symbol by calling SETBF, then loop for next VLE, if any. If all VLE's are processed, fall through to 5. Move the appropriate number of dashes into PTBF and the card image (to create the horizontal line for the entry VLE's). If there are no exit VLE's, go to 8. If there is a 0 in the exit VLE, go to 8194. Otherwise, process the exit VLE, as was done for the entry VLE's.

8 - Fill in the rest of the PTBF from LINEBF. If there were no intersections of vertical or horizontal lines, go to 19. Otherwise, loop through the INTRSC table and move an intersection symbol into PTBF for each intersection.

19 - Print the line number (LINCT), card image, and PTBF. If there was an exit or entry VLE on the line, print a blank line. Then, go to 2.

1 - Increment LINCT, print LINCT and the card image. Then, return to the calling routine.

8192 - Print the card image and LINEBF, then go to 2.

8194 - Put the exit message into the card image, then go to 8.

CALLING SEQUENCE: CALL DSPY(DUMMY)

where:
DUMMY is a variable. Currently, it is not used as input.

OPTIONS: None.

CALLED BY: MAINPGM (FORFLOW)

SCREENS USED: None.

57

INPUTS: Several common areas are used as input. The common area MAPS contains an array, MAP, such that:
MAP(i) = actual line position of nesting level i.

The common area LINTAB contains the following areas of interest:

- VLETAB - table of VID's for vertical lines.
- ENDPT - table of end points for lines in VLETAB.
- BEGPT - table of levels for lines in VLETAB. Note that this table is altered by MAINPGM during the line spacing algorithm.
- VLECT - count of number of entries in VLETAB.

In addition, DSPY requires the card images and VLE's on data set 9, as described in VLECRT.

OUTPUTS: The original program listing, along with flow lines, is outputted onto the line printer.

PROGRAM CONSTANTS:

- EXITR - Contains '++++++++++++++++++(EXIT)' which is an exit message.
- COM - Contains 'CM', the comment code.
- COND - Contains 'CN', the continuation code.
- BLANK - '$\phi\phi\phi\phi$'
- PLUSS - Contains 72 +'s.
- MINUSS - Contains 72 -'s.
- IBTSYM - Contains 'O$\phi\phi\phi$', the intersection symbol.
- ONE - Contains '1$\phi\phi\phi$'.
- BLAN1 - Contains '$\phi\phi$'.
- ENDR - Contains 'EN', the end code.

PROGRAM VARIABLES:

- LINCT - The print line count. This is different from I, since not all lines are counted in the print-out.
- I - Card count of card to be read.
- IBLK - Pointer to position in the card image where horizontal line should begin.

MAX - The rightmost print position in PTBF which will contain a part of the current horizontal line.

ICOUNT - Count of number of intersections in the current line.

IVERT - The current VTYPE.

ERROR MESSAGES: None.

SUBROUTINES CALLED: IICBTD, IIFILL, IITBLS, SETBF.

PROGRAM ID: GTPT
PROGRAM NAME: GET, PUT and CLOSE Routines for XREFIT

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: GTPT serves three functions. They are:

- Close the input and output files
- Read cards from the input deck
- Print the output lines.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The programs that create the FORTRAN cross-reference listings were written in such a way as to use as little core as possible. To accomplish this, XREFIT was written so that it did not do any I/O operations. All I/O operations were provided by Assembly Language subroutines. This eliminated the requirement for IOCS which is normally loaded by FORTRAN. Then by making XREFIT a subroutine which is called by an Assembly Language program, it was possible to avoid the loading of IOCS by FORTRAN.

The subroutines that provide XREFIT with the I/O capabilities have been assembled in one CSECT called GTPT. These subroutines are CLOSER, GETCRD, and PUTLIN.

LOGIC NARRATIVE:

CLOSER - Save the registers, close the card input and print output files, reset file flag to closed, restore registers and return to the calling program.

GETCRD - Save the registers and get the input parameter pointers (BUFFER and IFLAG).

FIRSTBR - If the card input and print output files are already open, go to OPENDONE. Otherwise, open the files.

OPENDONE - Get a card; on EOF go to EOFHIT. Load R0 with return code of 0 to indicate a good read.

RETURNIT - Store R0 in IFLAG, restore registers and return to the calling program.

EOFHIT - Load R0 with return code of 1 to indicate EOF and go to RETURNIT.

PUTLIN - Save the registers and get the output buffer address. Write a line. If the call was for a page eject rather than a write, go to DOPRNT.

BLANK - Blank out the output buffer, restore the registers and return to the calling program.

DOPRNT - Write the page header, space 2 lines and go to BLANK.

CALLING SEQUENCE: CALL CLOSER

CALL GETCRD(BUFFER,IFLAG)

where:
BUFFER is an 80-character input buffer.
IFLAG is an EOF indicator:
0 = No EOF
1 = EOF on input file.

CALL PUTLIN(PRINTBUF)

where:
PRINTBUF is the print line buffer.

OPTIONS: None.

CALLED BY: XREFIT

SPECIAL NOTE: These subroutines are not general-purpose. They process 2 files called RECIN and RECLST.

SCREENS USED: None.

INPUTS:

CLOSER - None
GETCRD - This subroutine reads the file called RECIN which contains the FORTRAN source deck. It expects the calling program to send it the address of the input buffer.

PUTLIN - This subroutine expects the print line to be sent to it in a buffer area which is pointed to by PRINTBUF.

OUTPUTS:

CLOSER - This subroutine sets a switch in GETCRD when it closes the input and output files.

GETCRD - IFLAG is set after each read. (See CALLING SEQUENCE)

PUTLIN - This subroutine sends print lines to the printer.

ENTRY POINTS:

CLOSER
GETCRD
PUTLIN

REGISTER USAGE:

R2 - Base Register.

R14 - Return address in calling program.

R3 - Buffer address, input and output.

PROGRAM CONSTANTS: None.

PROGRAM VARIABLES:

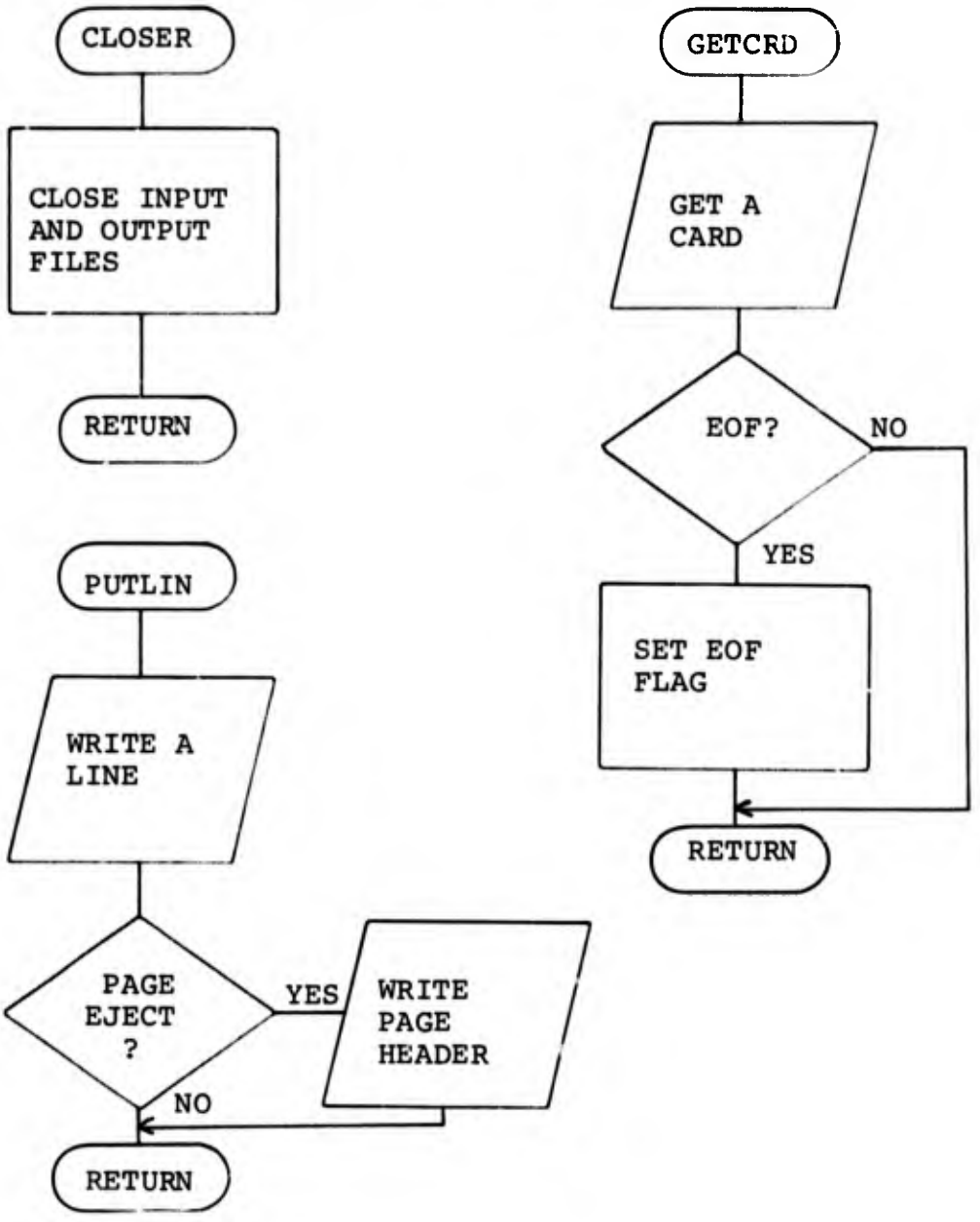
IN1, IN2 } Input buffers.

PRTLINE1, PRTLINE2 } Output buffers.

ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

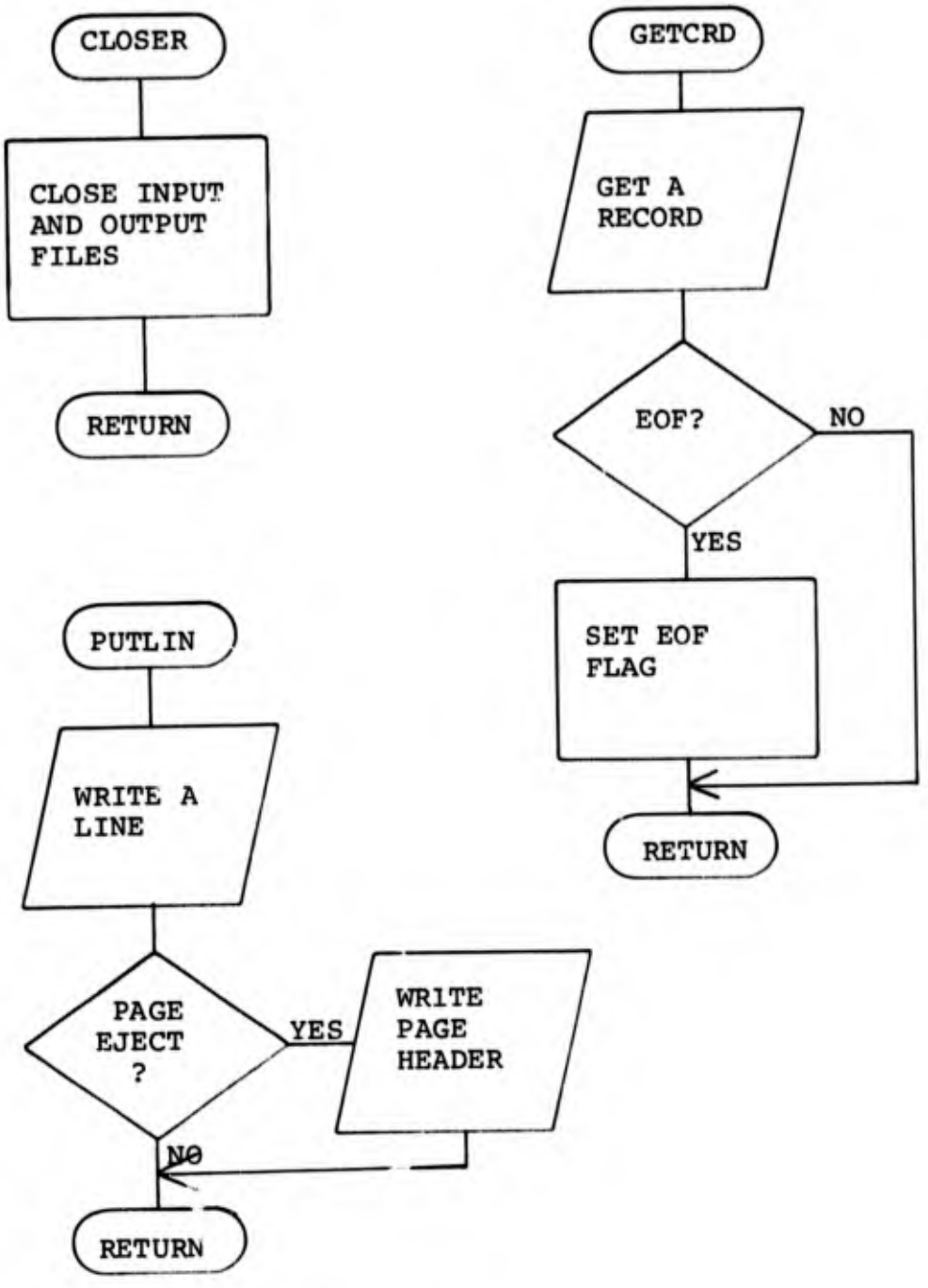
GTPT: Logic Diagram



PROGRAM ID: GTPX
PROGRAM NAME: GET, PUT and CLOSE Routines for XREFLW

PHILOSOPHY AND APPROACH: GTPX differs from GTPT in that the subroutine GETCRD has been modified to read the source deck from a disk data set. Each card from the source deck occupies a 100-byte disk record. Except for this difference, GTPX and GTPT are identical.

GTPX: Logic Diagram



PROGRAM ID: JICNT
PROGRAM NAME: Count Blanks in the Current Line

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: JICNT determines the position of the rightmost non-blank character in the card image. This determines where the horizontal line can begin in the final flowchart.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: In flowcharting on the line printer, it was necessary to find the rightmost non-blank character in each line, in order to draw horizontal lines. JICNT fulfills this need, by leaving this count in the actual line image on disk.

LOGIC NARRATIVE:

JICNT - Load registers with pointers to beginning of line image and end of line image.

LOOP - Starting from the rightmost character position in the line image, search for a non-blank character. When found, go to NONBLNK.

NONBLNK - Compute the position of the rightmost non-blank character, relative to the beginning of the line image. Store this into bytes 97 and 98 of the line image. Then, return to the calling routine.

CALLING SEQUENCE: CALL JICNT(LINE)

where:
LINE - A 100-byte area having the following format:

bytes	1 - 72	card image
	73 - 76	entry VLE 1
	77 - 80	entry VLE 2
	81 - 84	entry VLE 3
	85 - 88	entry VLE 4
	89 - 92	entry VLE 5
	93 - 96	exit VLE
	97 - 98	non-blank pointer
	99 - 100	card code.

OPTIONS: None.

CALLED BY: MAINPGM (ALCFLOW)
MAINPGM (FORFLOW)

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. JICNT uses the card image portion of LINE as input.

OUTPUTS: See CALLING SEQUENCE. The count obtained in JICNT is stored in bytes 97-98 of LINE.

ENTRY POINTS:

JICNT

PROGRAM CONSTANTS: None.

PROGRAM VARIABLES: None.

REGISTER USAGE:

R3 - Pointer to current position in LINE.

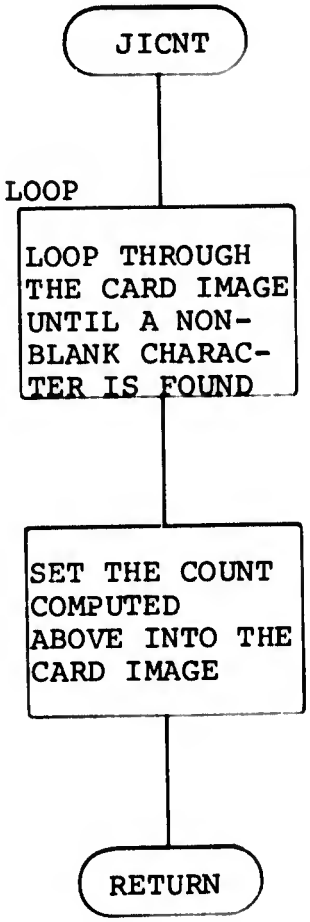
R15 - Base register.

R2 - Pointer to the beginning of the line.

ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

JICNT: Logic Diagram



PROGRAM ID: LKFORT
PROGRAM NAME: FORTRAN Statement Scan Routine

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: LKFORT provides FORTRAN programs with the capability to execute a byte-by-byte scan of an input character string. The scans are terminated when the user-specified character type is encountered in the character string or when the end of the string is encountered.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: LKFORT provides FORTRAN programs with the capability to scan a FORTRAN statement or any character string, byte-by-byte, for the following character types:

- Non-blank character
- Non-numeric character
- Non-alphanumeric character

Using two calls to LKFORT, the FORTRAN programmer can locate and determine the size of a variable name or statement number. For example, to process a FORTRAN statement, the first call would be to scan for a non-blank character. The statement is scanned till a non-blank is found and the data type code is returned to the calling program. If the type found is either alphabetic or numeric, a second call is made specifying a scan for a non-alphanumeric character. The routine scans the statement for a non-alphanumeric and returns the size of the field that is being processed. Similar calls are made until the whole statement has been scanned.

LKFORT uses the Translate and Test (TRT) instruction to carry out its scanning operations. Three tables are defined, one for each of the three valid types of scans. The NONAN (non-

alphanumeric) table (Figure XV-2) defines the characters which can terminate a non-alphanumeric scan. The NONBLNK (non-blank) table (Figure XV-3) defines the characters which can terminate a non-blank scan and the NONNUM (non-numeric) table (Figure XV4)defines the characters which can terminate a non-numeric scan.

LOGIC NARRATIVE:

LKFORT - Save registers. Compute register settings for first byte to scan and number to scan. Get the address of the proper TRT table based on the option indicator.
0 = NONBLNK
1 = NONNUM
2 = NONAN

DOALPH - Calculate the number of multiples of 256 in the characters to be scanned. If total character count is less than 256, go to LASTGRP.

LOOP - Process 256 bytes at a time using the TRT instruction. If the search character is found go to GETSIZ; otherwise, stay in this loop until all groups of 256 characters are done. Then fall through.

LASTGRP - Process the last group of characters and go to GETSIZ if the search character is found. Otherwise, fall through.

EXIT - Restore registers and return to the calling program.

GETSIZ - Compute the number of characters that were skipped over before finding a hit. Move that number and the data type of the character that terminated the scan to the calling program's area.

CALLING SEQUENCE: IRSULT = LKFORT(DATA,BYTE,NUMSCN,IOPT,DATYP)

where:
DATA and BYTE give the location of the starting byte for the left-to-right scan.
NUMSCN tells the maximum number of characters to scan.
IOPT gives the scan option.
0 = Scan for non-blank; 1 = Scan for non-numeric; 2 = Scan for non-alphanumeric.
DATYP identifies the type of character that terminated the scan.

1 = Alphabetic including dollar sign ($).
2 = Numeric (0-9).
3 = Punctuation excluding ()*=
4 = Blank
5 = (Opening parenthesis.
6 =) Closing parenthesis.
7 = * Asterisk.
8 = = Equal sign.

IRSULT will contain the relative byte number of the found item which is equivalent to the size of the current field. Values are: 0 = Condition not found
n = Relative byte number of found item.

OPTIONS: See IOPT under CALLING SEQUENCE.

CALLED BY: XREFIT, XREFLW

SCREENS USED: None.

INPUTS: See DATA, BYTE, NUMSCN, and IOPT under CALLING SEQUENCE.

OUTPUTS: See DATYP and IRSULT under CALLING SEQUENCE.

ENTRY POINT:

LKFORT

REGISTER USAGE:

R8 - First byte of data to be scanned.

R10 - Maximum number of bytes to scan.

R11 - Type of scan.

R12 - Data type.

R1 - Address of hit byte.

R2 - Value of hit byte.

R4 - Address of translation table.

R14 - Return address.

PROGRAM CONSTANTS:

NONAN - Translation table which is used in a scan for non-alphanumeric characters. (See Figure XV-2).

NONBLNK - Translation table which is used in a scan for non-blank characters. (See Figure XV-3).

NONNUM - Translation table which is used in a scan for non-numeric characters. (See Figure XV-4).

PROGRAM VARIABLES: None.

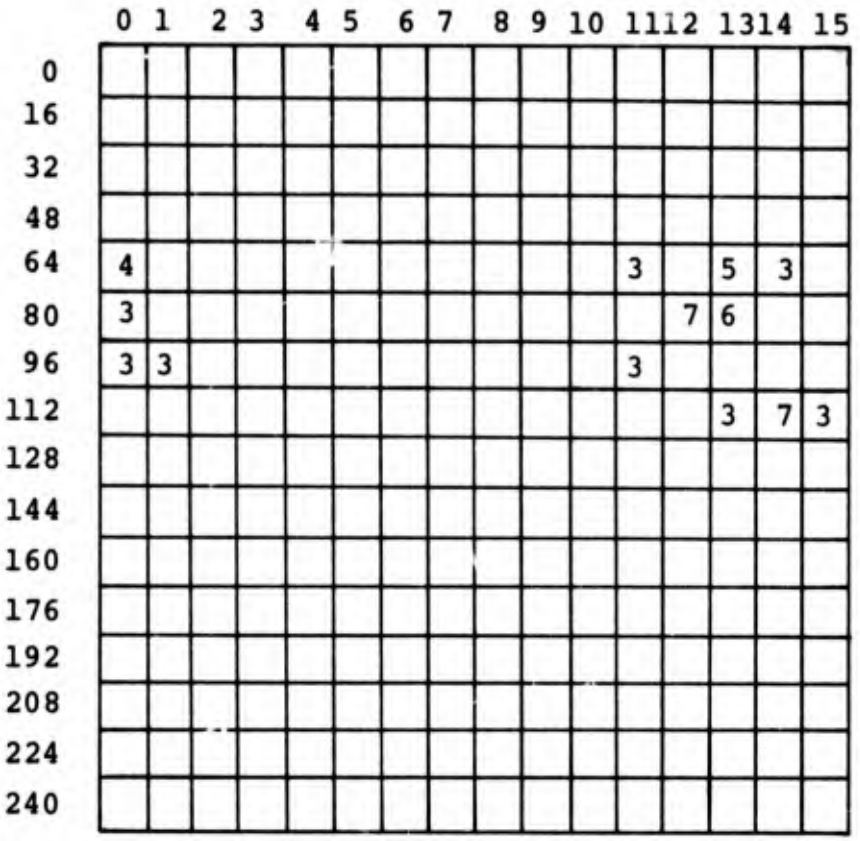
ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
16																
32																
48																
64	¢											.		(	+	
80	&											$	*	)		
96	-	/										,				
112														,	=	"
128																
144																
160																
176																
192		A	B	C	D	E	F	G	H	I						
208		J	K	L	M	N	O	P	Q	R						
224			S	T	U	V	W	X	Y	Z						
240	0	1	2	3	4	5	6	7	8	9						

This matrix defines the character set that is used by LKFORT.

Figure XV-1: EBCDIC CHARACTER MATRIX



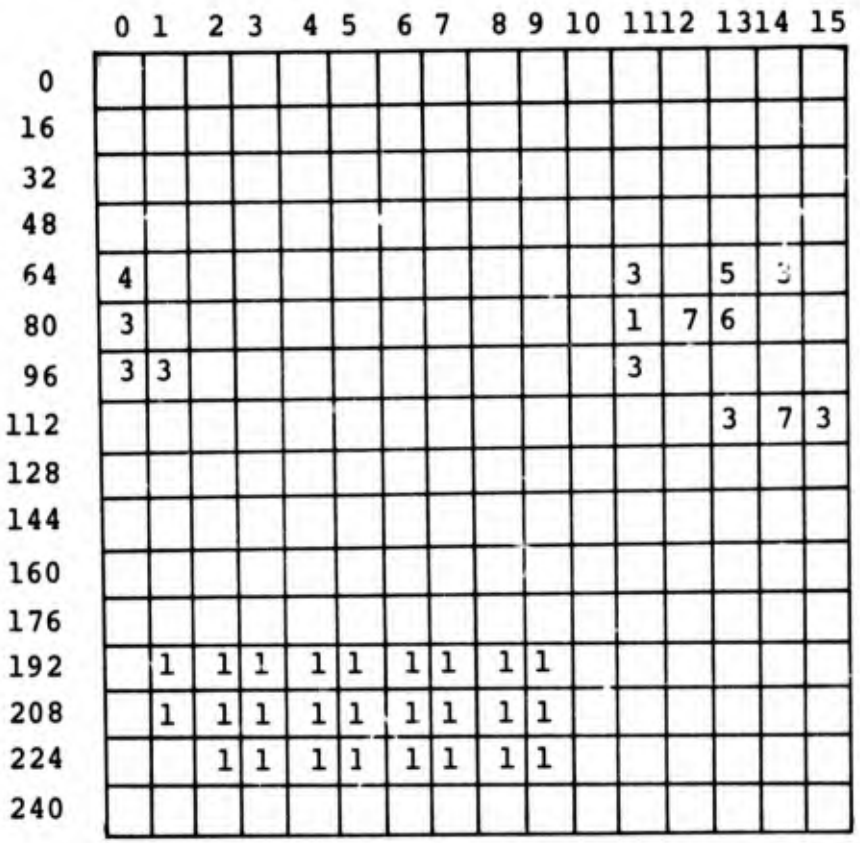
The table entries all contain zeros except for those that are assigned values as shown above.

Figure XV-2: NONAN Translation Table

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
16																
32																
48																
64												3		5	3	
80	3											1	7	6		
96	3	3										3				
112														3	7	3
128																
144																
160																
176																
192		1	1	1	1	1	1	1	1	1						
208		1	1	1	1	1	1	1	1	1						
224			1	1	1	1	1	1	1	1						
240	2	2	2	2	2	2	2	2	2	2						

The table entries all contain zeros except for those that are assigned values as shown above.

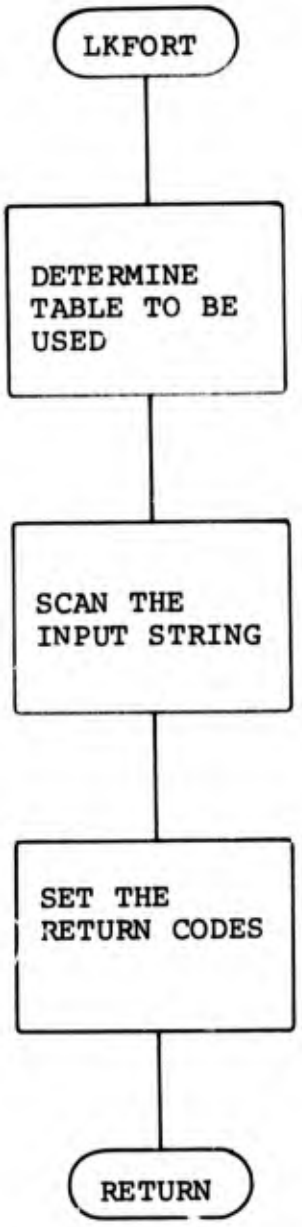
Figure XV-3: NONBLNK Translation Table



The table entries all contain zeros except for those that are assigned values as shown above.

Figure XV-4 : NONNUM Translation Table

LKFORT: Logic Diagram



PROGRAM ID: REFM
PROGRAM NAME: FORTRAN Cross Reference Driver Program

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: REFM calls routine XREFIT.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: Cross-reference listings are valuable tools to a programmer during program check-out, debugging, and maintenance. They enable the programmer to quickly check all references to statement numbers and data items.

The FORTRAN compiler that is provided for the 360/40 DOS does not produce cross-reference listings.

Therefore, a cross-reference program was developed as part of the IOC effort. The program accepts a FORTRAN source deck as input and generates a listing which consists of a source deck print-out and a cross-reference listing. The source deck print-out looks exactly like the source listing produced by the FORTRAN compiler. It may be suppressed via an indicator in one of the control cards.

Program REFM is the driver for the creation of FORTRAN cross-reference listings. Its primary purpose is to eliminate the loading of IOCS by FORTRAN since it is not needed by XREFIT.

LOGIC NARRATIVE:

BEGIN - Load the base register and call XREFIT to create the FORTRAN cross-reference listing. Upon return go to EOJ.

CALLING SEQUENCE: None. REFM is a batch program which is executed when the control card "// EXEC IIDELY" is encountered.

OPTIONS: None.

CALLED BY: None. See CALLING SEQUENCE above.

SCREENS USED: None.

INPUTS: None.

OUTPUTS: None.

ENTRY POINTS:

REFM

PROGRAM CONSTANTS:

XREFIT - Address constant which points to XREFIT.

PROGRAM VARIABLES: None.

REGISTER USAGE:

R2 - Base register.

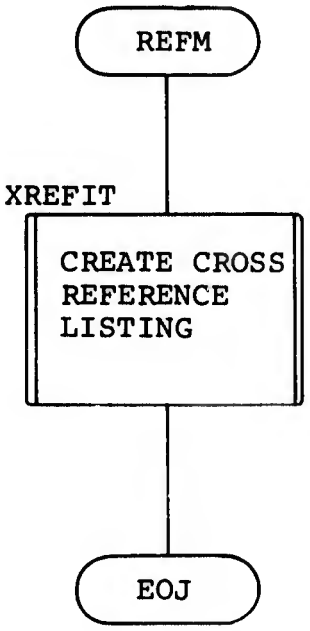
R14 - Return address in subroutine call.

R15 - Address of called subroutine.

ERROR MESSAGES: None.

SUBROUTINES CALLED: XREFIT

REFM: Logic Diagram



PROGRAM ID: REFX
PROGRAM NAME: FORTRAN Cross-Reference Driver for Flowcharts

PHILOSOPHY AND APPROACH: REFX is exactly the same as REFM except that REFX calls XREFLW and REFM calls XREFIT. By substituting "XREFLW" for "XREFIT", the documentation for XREFIT can be used for both programs. One additional difference is that REFX is executed when the control card "// EXEC IXREFLW" is encountered.

REFX: Logic Diagram

PROGRAM ID: SETBF
PROGRAM NAME: Set Print Symbol Into Print-Time Line Buffer

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: SETBF is used by DSPY to manipulate the line buffer symbols at print time. Using IVERT, it places a selected symbol in LINEBF.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer ; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: See PHILOSOPHY AND APPROACH for ALCFLOW or FORFLOW.

LOGIC NARRATIVE:

SETBF - Depending on the vertical line type (IVERT), branch to the following code:
type 1 : go to 101
type 2 : go to 102
type 3 : go to 103
type 4 : go to 9

101 - Set a "+" symbol into the line buffer (LINEBF) at position IRS. Then, go to 9.

102 - Set a "1" symbol into the line buffer at position IRS. Then, go to 9.

103 - Set a "$\not{b}$" symbol into the line buffer at position IRS.

9 - Return to the calling routine.

CALLING SEQUENCE: CALL SETBF(IRS,IVERT,LINEBF)

where:

IRS	-	Position in LINEBF to be affected.
IVERT	-	Vertical line type. This determines which symbol is to be placed in LINEBF.
LINEBF	-	Area containing print-time line buffer.

OPTIONS: None.

CALLED BY: DSPY (ALCFLOW)
DSPY (FORFLOW)

SPECIAL NOTE: The area LINEBF is the print-time line buffer. This area contains an image of the vertical line or flow line section of the next print line at all times during printing.

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. IRS, IVERT, and LINEBF are inputs to SETBF.

OUTPUTS: See CALLING SEQUENCE. SETBF outputs a symbol in LINEBF.

PROGRAM CONSTANTS:

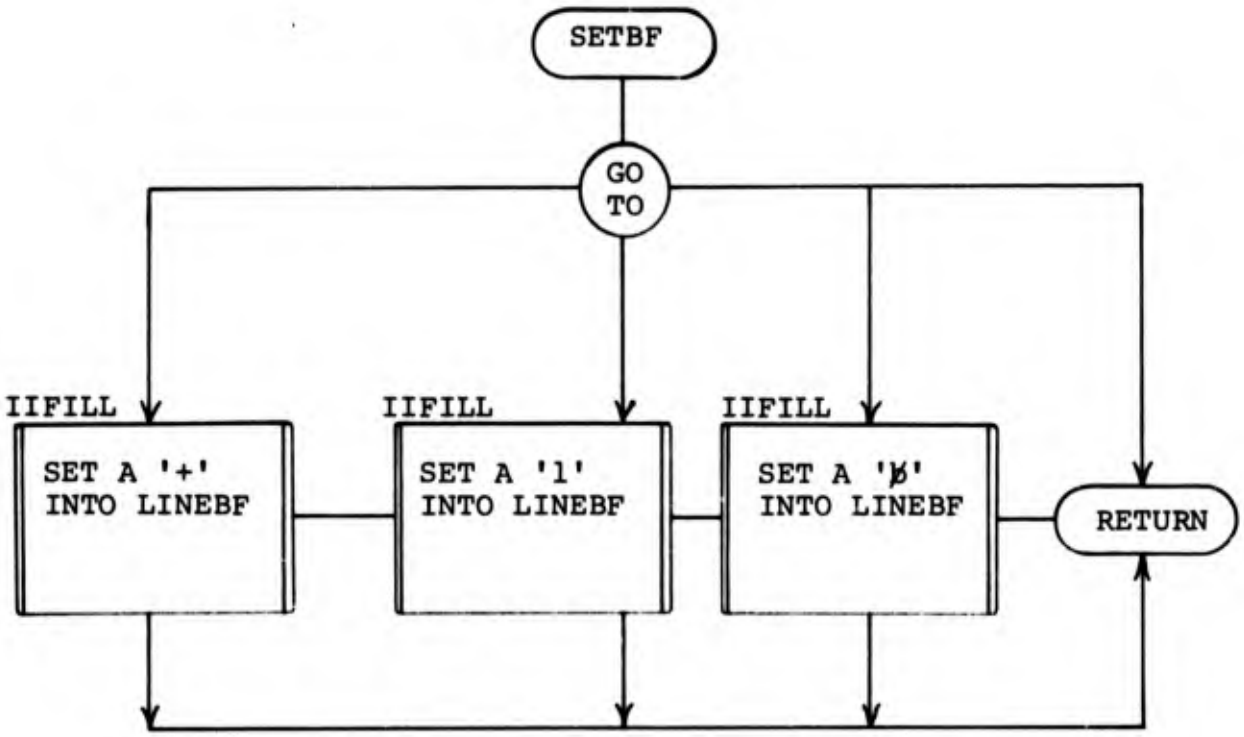
PLUSS	-	Contains '++'
ONE	-	Contains '11'
BLANK	-	Contains 'ØØ'

PROGRAM VARIABLES: None.

ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

SETBF: Logic Diagram



PROGRAM ID: TABFIL
PROGRAM NAME: Partial Ordering Bit-Table Manipulation Routine

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: The three routines in this CSECT perform operations needed to utilize the partial-ordering tables, as described in the PHILOSOPHY AND APPROACH for ALCFLOW or FORFLOW.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: See PHILOSOPHY AND APPROACH for ALCFLOW or FORFLOW.

LOGIC NARRATIVE:

TABFIL - First, compute the row in which the affected bit will be found. Then, compute the byte within the row by dividing the second index by 8. Finally, compute the actual byte location needed, and OR a "1"-bit into TABLE. Then, return to the calling routine.

TEST - Compute the needed byte position as described above. Then, test the bit and return either a "1" or "0" in Register 0.

ZERO - Using the given row number, zero out the row. Then, return to the calling routine.

CALLING SEQUENCE: CALL TABFIL(I,J)
K = TEST(L,M)
CALL ZERO(N)

where:
(I,J) is the index of the bit to be set to 1.
(L,M) is the index of the bit to be tested.
K is the value of the bit tested.
N is the row to be zeroed.

OPTIONS: TABFIL incorporates three distinct functions. These are: 1) Test a bit; 2) Set a bit to 1; 3) Zero out a row.

CALLED BY: MAINPGM (ALCFLOW)
MAINPGM (FORFLOW)

SPECIAL NOTE: In all calls to routines in TABFIL, the first index refers to the row in the matrix. For example, the (1,20)th bit is the 20th bit in the 1st row, or the 20th bit relative to the beginning of TABLE. The main reason for this implementation is that with this, ZERO can be implemented as a simple move instruction, which greatly increases speed when operating on the partial ordering table (see PHILOSOPHY AND APPROACH for ALCFLOW or FORFLOW for more details).

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. For TABFIL, I and J are inputs. Also, the common area, TABLE, is used. For TEST, L and M are inputs, as well as TABLE. For ZERO, N is input, as well as TABLE.

OUTPUTS: See CALLING SEQUENCE. For TABFIL, output consists of a bit in TABLE. For TEST, output is K. For ZERO, output is a row of zeroes in TABLE.

ENTRY POINTS:

TABFIL
TEST
ZERO

REGISTER USAGE:

TABFIL: R15 - Base register.
R1 - First index working area.
R2 - Second index.
R4 - First index.
R3 - Bit mask.

TEST: R15 - Base register.
R1 - First index working area.
R2 - Second index.
R4 - First index.
R3 - Bit mask.

ZERO: R15 - Base register.
R1 - Row index.
R3 - Area pointer.

RESTRICTIONS: Routines in TABFIL operate on a TABLE of 7200 full words. To the user, this table consists of a 480 x 480 bit table.

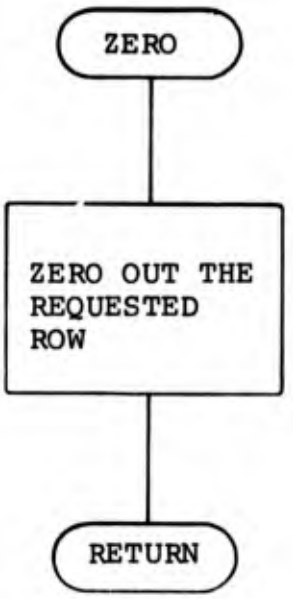
PROGRAM CONSTANTS: None.

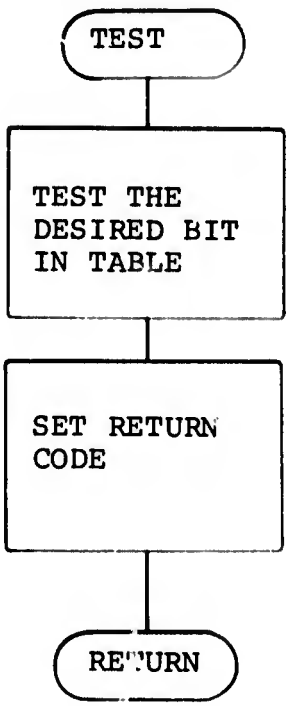
PROGRAM VARIABLES: None.

ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

TABFIL: Logic Diagram





PROGRAM ID: VLECRT (ALCFLOW)
PROGRAM NAME: Create VLE's for ALCFLOW

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: Given an exit line number and an entry line, VLECRT will create the appropriate exit and entry VLE's. VLECRT also has several auxiliary functions:

- building and sorting (VLETAB, BEGPT, ENDPT) tables for later use in line nesting and allocation algorithm.
- combining vertical lines for single-entry statements when possible.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: See PHILOSOPHY AND APPROACH for VLECRT(FORFLOW). One exception should be noted. The records used in ALCFLOW contain 120 bytes. The format of these records is:

bytes	1 - 72	card image
	73 - 76	entry VLE1
	77 - 80	entry VLE2
	81 - 84	entry VLE3
	85 - 88	entry VLE4
	89 - 92	entry VLE5
	93 - 96	exit VLE
	97 - 100	code, length
	101 - 104	branch address of entry statement
	105 - 108	print line number as generated by assembler
	111 - 114	op code of current line
	115 - 118	macro indicator

The extra four fields are generated by the assembler, and it was decided to include them in the record in order to eliminate most of the syntax analysis phase later.

LOGIC NARRATIVE:

VLECRT - Find line number of entry statement by searching address table (IADDTB).

5 - Create the VLE's for the entry and exit statement. Here, if a statement has been referenced earlier, that line can be combined with the current one. Thus, a common VLE is created for all common vertical lines. In order to keep track of these references, the (REFTAB, EXITS, PREREF) tables are used. When done, go to 1999 to modify and sort the (VLETAB, BEGPT, ENDPT) tables.

1999 - Check if the entry statement was ever referenced before. If not, go to 2000. Otherwise, modify the (VLETAB, BEGPT, ENDPT) entry. Then, go to 2127.

2000 - Add a new entry to (VLETAB, BEGPT, ENDPT) tables.

2127 - Sort the (VLETAB, BEGPT, ENDPT) tables in ascending order according to BEGPT. When done, return to the calling routine.

CALLING SEQUENCE: CALL VLECRT(LABEL,EXTLIN,LABLCT)

where:

LABEL - An entry label in binary.
EXTLIN - The line number of the exit statement to be processed.
LABLCT - Not currently used.

OPTIONS: None.

CALLED BY: MAINPGM (ALCFLOW)

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. LABEL and EXTLIN are inputs to VLECRT. In addition, a common area, LABLS, is used as input. The format for this area is:

INLINE - Count of number of entries in IADDTB.
IADDTB - Table of loading addresses for each line on data set 9. Lines which are not legitimate source lines have a 0 entry in IADDTB.

OUTPUTS: VLECRT creates VLE's for entry and exit lines as needed. Also, several tables are built in common LINTAB for later use. The format of LINTAB is:

- VLETAB - table of identifying VLE numbers for vertical lines.
- BEGPT - table of beginning lines for VLE's in VLETAB.
- ENDPT - table of ending lines for VLE's in VLETAB.
- VLECT - count of number of entries in VLETAB.

PROGRAM CONSTANTS:

- HPLUS - Contains 16, type number for horizontal VLE.
- HMINUS - Contains 32, type number for horizontal VLE.
- VPLUS - Contains 1, type number for VLE type.
- VONE - Contains 2, type number for VLE type.
- VHALT - Contains 3, type number for VLE type.
- VCONT - Contains 4, type number for VLE type.

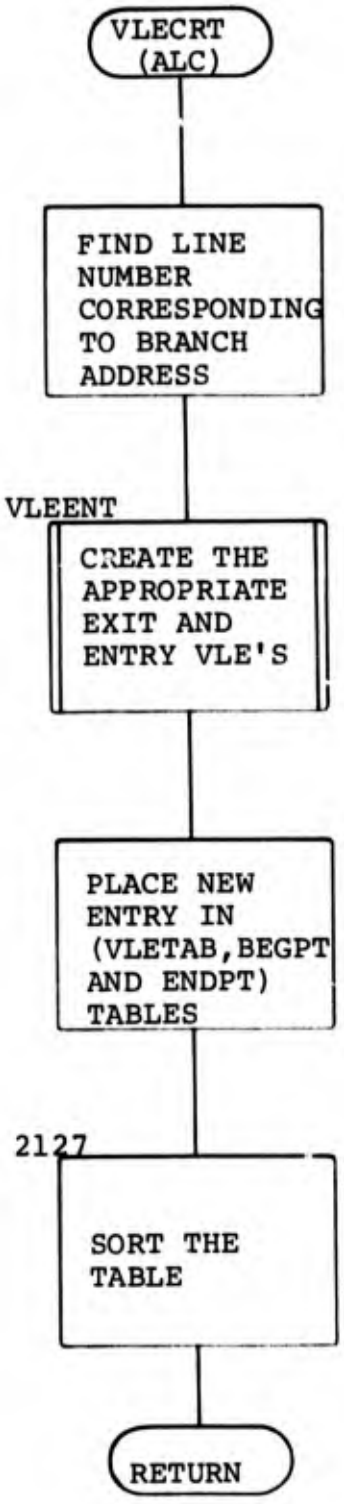
PROGRAM VARIABLES:

- VLES - Common area containing VLE to be entered into line image.
- BEGIN - Temporary current beginning point.
- ENDR - Temporary current ending point.
- ITEMP - Temporary area for sort process.
- REFTAB - Table of labels which are currently referenced.
- EXITS - Table of the current exit statements associated with labels in REFTAB.
- PREREF - Tables of the last reference to labels in REFTAB.
- REFCT - Count of number of entries in REFTAB.

ERROR MESSAGES: 'LABEL n IS MISSING' - the label n could not be found in the symbol table LABTAB.

SUBROUTINES CALLED: IITBLS, VLEENT

VLECRT(ALCFLOW): Logic Diagram



PROGRAM ID: VLECRT (FORFLOW)
PROGRAM NAME: Create VLE's for an Exit Statement in FORFLOW

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: Given an exit line number, and a table of one or more entry lines, VLECRT will create the appropriate exit and entry VLE's. VLECRT also has several auxiliary functions:

- building and sorting (VLETAB, BEGPT, ENDPT) tables for later use in line nesting and allocation algorithm.
- combining vertical lines for single-entry statements when possible.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The following is a discussion of VLECRT: Although the functional description and logic narrative for VLECRT (ALCFLOW) and VLECRT (FORFLOW) are quite different, the philosophy for both versions is essentially the same. The main differences are:

1. VLECRT (ALCFLOW) handles only single entry statements; VLECRT (FORFLOW) handles both multiple and single entry statements.

2. VLECRT (ALCFLOW) searches for the entry number by searching an address table, and then the index indicates the line; VLECRT (FORFLOW) searches a label table, and uses the index to get a line number from another table.

Other than the two points above, both versions of VLECRT are exactly alike. Thus, only one philosophy will be written for both.

The basic structure in the flowcharting process consists of generating instructions for drawing the flow lines in one pass, and drawing the flow lines later. In order to facilitate this, the idea of the vertical line event (VLE) is used. Each VLE consists of a vertical line identification (VID) and a vertical line type (VTYPE). Each VID identifies which vertical line the VLE is associated with, while the VTYPE will act as instructions to the display routine (DSPY).

In the initial phase of flowcharting, 100-byte records are written onto disk. These consist of:

bytes	1 - 72	:	line image
	73 - 76	:	entry VLE1
	77 - 80	:	entry VLE2
	81 - 84	:	entry VLE3
	85 - 88	:	entry VLE4
	89 - 92	:	entry VLE5
	93 - 96	:	exit VLE
	97 - 100	:	card code, blank pointer

Thus, each line of code can have, at most, 5 entries, and 1 exit.

Because of the limit on the number of entries into each statement, and in order to decrease the number of lines in each flowchart, an algorithm for combining vertical lines was implemented in VLECRT. This line combining process is used only for single entry statements, since line combining multiple entry statements will result in several ambiguous situations.

Basically, this line combining process was accomplished by keeping track of only the first reference to each label, and creating VLE's only for these references. Three tables are used for this process: REFTAB, EXITS, and PREREF.

An algorithm for nesting vertical lines is also incorporated in the flowcharting process. This is done partially in VLECRT (ALCFLOW and FORFLOW), and partially in MAINPGM (ALCFLOW and FORFLOW). Only the VLECRT portion of this process is described here. In VLECRT, three tables are built. These are:

VLETAB	-	table of VID's for vertical lines.
BEGPT	-	table of beginning line numbers of vertical lines.
ENDPT	-	table of ending line numbers of vertical lines.

The entries in these tables are sorted in ascending order, according to BEGPT. Thus, in VLETAB, the vertical lines are sorted according to their beginning points. Later, in MAINPGM, these tables are used to define a partial ordering relationship and generate line nesting levels.

LOGIC NARRATIVE:

VLECRT - Convert the entry labels to actual line numbers by using the label tables. If there is only one entry label, go to 5. Otherwise, must process a multi-entry statement. First, sort the entry line numbers in ascending order. Check if the exit statement occurs after all the entry lines. If so, go to 2. Otherwise, get ready to enter a VLE in the exit line which is a VMINUS-type (start a downward-vertical line). Then, go to 4.

2 - Get ready to enter a VLE in the exit line which is a VHALT-type (end a downward-vertical line).

4 - Enter the appropriate VLE in the exit line. Then, create VLE's for all entry lines. When done, go to 2001, to enter the new VLE in (VLETAB, BEGPT, ENDPT) tables and sort them.

5 - Create VLE's for entry and exit statements. Here, if a statement has been referenced earlier, that line can be combined with the current one. Thus, a common VLE is created for all common vertical lines. In order to keep track of these references, the (REFTAB, EXITS, PREREF) tables are used. When done, go to 1999 to modify and sort the (VLETAB, BEGPT, ENDPT) tables.

1999 - Check if the entry statement was ever referenced before. If not, go to 2000. Otherwise, modify the (VLETAB, BEGPT, ENDPT) entries, then go to 2127.

2000 - Add a new entry to (VLETAB, BEGPT, ENDPT) table.

2127 - Sort the (VLETAB, BEGPT, ENDPT) tables, in ascending order according to BEGPT. When done, return to the calling routine.

2001 - Determine lowest and highest (BEGIN and END) lines of current VLE, then go to 2000 to enter and sort entry in (VLETAB, BEGPT, ENDPT) tables.

CALLING SEQUENCE: CALL VLECRT(LABEL,EXTLIN,LABLCT)

where:
LABEL is a table of entry labels in binary.
EXTLIN is the line number of the exit statement to be processed.
LABLCT is the count of the number of entries in LABEL table.

OPTIONS: None.

CALLED BY: MAINPGM (FORFLOW)

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. LABEL, EXTLIN, and LABLCT are inputs to VLECRT. In addition, a common area, LABLS, is used as input. The format for this area is:

TABSZE - Count of number of entries in LABTAB.
LABTAB - Table of labels, in binary.
LABVAL - Table of line numbers for labels in LABTAB.

OUTPUTS: VLECRT creates VLE's for entry and exit lines as needed. Also, several tables are built in common LINTAB for later use. The format of LINTAB is:

VLETAB - Table of identifying VLE numbers for vertical lines.
BEGPT - Table of beginning lines for VLE's in VLETAB.
ENDPT - Table of ending lines for VLE's in VLETAB.
VLECT - Count of number of entries in VLETAB.

PROGRAM CONSTANTS:

HPLUS - Contains 16, type number for horizontal VLE.

HMINUS - Contains 32, type number for horizontal VLE.

VPLUS - Contains 1, type number for VLE type.

VONE - Contains 2, type number for VLE type.

VHALT - Contains 3, type number for VLE type.

VCONT - Contains 4, type number for VLE type.

PROGRAM VARIABLES:

VLES - Common area containing VLE to be entered into line image.

BEGIN - Temporary current beginning point.

ENDR - Temporary current ending point.

ITEMP - Temporary area for sort process.

REFTAB - Table of labels which are currently referenced.

EXITS - Table of the current exit statements associated with labels in REFTAB.

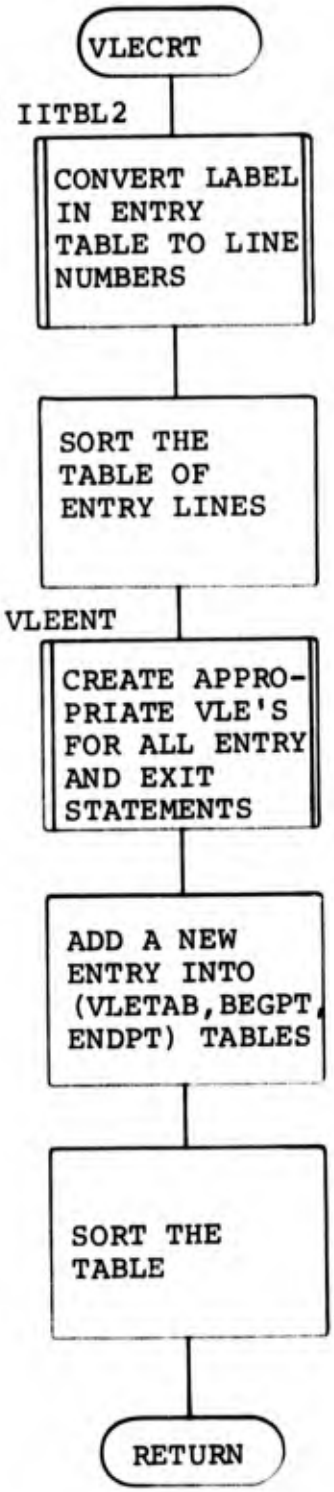
PREREF - Table of the last reference to labels in REFTAB.

REFCT - Count of number of entries in REFTAB.

ERROR MESSAGES: 'LABEL n IS MISSING' - the label n couldn't be found in the symbol table LABTAB.

SUBROUTINES CALLED: IITBLS, IISRTA, VLEENT.

VLECRT(FORFLOW): Logic Diagram



PROGRAM ID: VLEENT
PROGRAM NAME: Enter a VLE Into a Line

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: VLEENT is called by both ALCFLOW and FORFLOW routines in order to create a VLE entry in a line image on disk. These VLE's are utilized later by DSPY in order to draw flow lines.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: See PHILOSOPHY AND APPROACH for ALCFLOW or FORFLOW.

LOGIC NARRATIVE:

VLEENT - Read the line which is to be altered into core. If the VLE to be entered is an entry VLE, go to 5. Otherwise, place the VLE in the exit VLE area of the line, and write the line back out to disk. Then, return to the calling routine.

5 - Search for a blank entry VLE area. If found, store VLE in this area, and return to the calling routine. Otherwise, print error message, and return to the calling routine.

CALLING SEQUENCE: CALL VLEENT(LINE,NPLUS)

where:

LINE - The line number of the line in which the VLE in the NVLE area is to be stored.

NPLUS - 0, if the VLE is an entry VLE.
1, if the VLE is an exit VLE.

OPTIONS: None.

CALLED BY: VLECRT (FORFLOW)
VLECRT (ALCFLOW)
MAINPGM (ALCFLOW)

SPECIAL NOTE: Each line image on disk consists of records (100 byte/record in FORFLOW; 120 bytes/record in ALCFLOW). Each line image has the following format:

bytes 1 - 72	Card image
73 - 76	Entry VLE 1
77 - 80	Entry VLE 2
81 - 84	Entry VLE 3
85 - 88	Entry VLE 4
89 - 92	Entry VLE 5
93 - 96	Exit VLE
97 - 98	Blank column pointer
99 - 100	Card type code.
101 - 120	See DSPY(FORFLOW), PHILOSOPHY AND APPROACH

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. LINE and NPLUS are inputs to VLEENT. Also, the first 4 bytes in the common area VLES contains the VLE to be entered. In addition, the data set containing the lines is needed as input.

OUTPUTS: Output from VLEENT consists of the altered line on disk.

PROGRAM CONSTANTS: None.

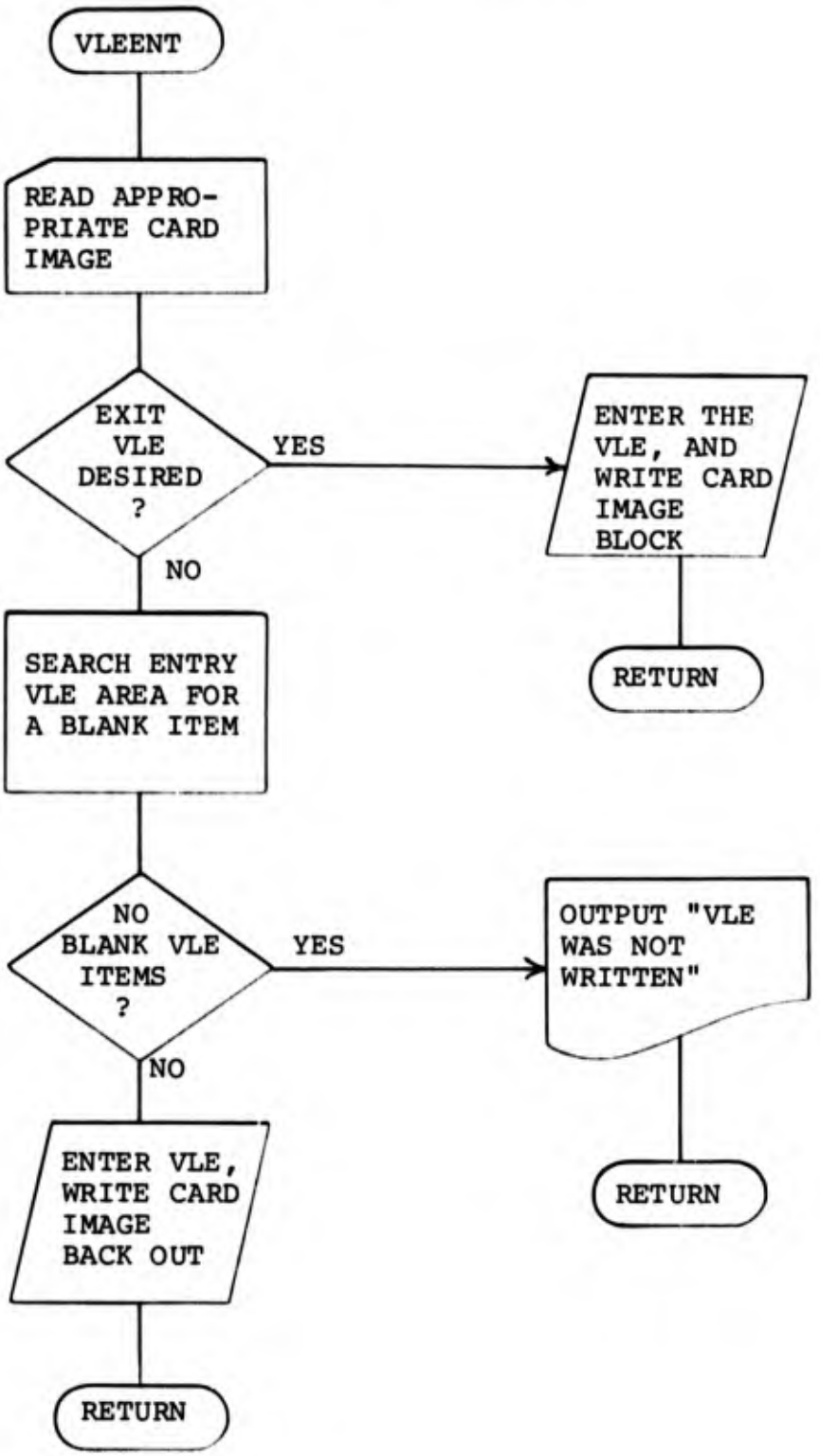
PROGRAM VARIABLES:

I - Position in line which contains a blank VLE area.

ERROR MESSAGES: 'VLE NOT WRITTEN='n - where n is the line which was not altered, due to no free area for the new VLE.

SUBROUTINES CALLED: None.

VLEENT: Logic Diagram



PROGRAM ID: XREFIT
PROGRAM NAME: FORTRAN Cross-Reference Program

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: XREFIT is a batch program that produces a source listing and a cross-reference listing on a FORTRAN source deck or a series of FORTRAN source decks.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2 as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: XREFIT is actually a subroutine which is called by a program called REFM. REFM is a small Assembly language program whose only function is to call XREFIT. The purpose of this set up is to avoid the loading of IOCS by FORTRAN since it is not needed. XREFIT calls several subroutines to provide specialized functions. They are:

- GETCRD reads the input card deck, one card per call
- PUTLIN prints the output lines, one line per call
- CLOSER closes the files
- LKFORT scans the FORTRAN statement until it encounters the type of characters specified by the calling program
- IISRTB sorts the cross-reference table entries
- COMREG returns the system data and the flag which indicates whether the source deck listing is to be printed.

XREFIT processes two types of input cards: comments ("C" in column 1) and FORTRAN statements. Comments are assigned no line numbers and are not processed. They are merely printed if a source listing has been requested.

FORTRAN statements are processed in the following manner. All continue cards for a statement are read and the contents are concatenated to form a single, complete statement in the work area. The statements are scanned from left to right starting in column 1. Generally speaking, numerics are treated as statement numbers and alphanumerics are treated as names of variables. However, in FORTRAN statements, numerics can represent constants as well as statement numbers. XREFIT is only interested in statement numbers. Therefore, the following characteristics of numerics are applied:

- Numerics appearing in columns 1-5 are statement numbers and more specifically, statement definitions (i.e., that is where the statement number is defined).

- Numerics appearing within a set of parentheses will be ignored except if this occurs after the words "GO TO". In this case, they are treated as statement numbers.

- Numerics appearing after the last closing parenthesis (i.e. the rightmost closing parenthesis in a group of nested parentheses) will be treated as statement numbers.

- Numerics appearing after an asterisk (*) or an equal sign (=) are ignored.

XREFIT scans the statements for certain reserved words which will not appear on the cross-reference listing. They are TO, AND, OR, NOT, EQ, GE, LE, GT, LT, NE, GO, IF, REAL, CALL, DATA and DO. In addition, the following three types of statements are not scanned for cross-references.

- FORMAT cards are assigned line numbers, then are treated as comment cards.

- DATA cards are assigned line numbers, then are treated as comment cards.

- END cards signal the end of a source deck. Because XREFIT accepts several source decks stacked together in the input stream, the END cards tell the program that the accumulated entries of the cross-reference table should be sorted and output before going on to the next input card.

LOGIC NARRATIVE:

Call COMREG to get system data and LISTFL which indicates whether source deck listing has been requested.

20 - Call GETCRD to get first card. If this is EOF, go to 555; otherwise, move the card to work area.

26 - If EOF on the input has been found, go to 550. If not a Comment card, go to 700; otherwise, go to 1303.

700 - Assign a line number to the card.

1303 - Set next available space pointer to end of first card in work area.

100 - If EOF on the input has been found, go to 550. Call GETCRD to get the next card; if EOF, go to 556. If Comment card, go to 200. If not continuation card, go to 200. Otherwise, attach this card to end of previous card in work area and go to 100.

556 - Set indicator to show that EOF has been encountered.

200 - If current card is a comment card, go to 275.

260 - Now process the card. Alphanumeric strings longer than 6 characters are ignored because they are not variable names. If an END card is found, go to 550. If a FORMAT card is found, go to 26. If it is a reserved word ignore it; otherwise, build an entry in the cross-reference table and go to 260.

Numeric strings are processed only if they are statement numbers. The logic used to determine whether a numeric string is a statement number, is discussed in the PHILOSOPHY AND APPROACH. Go to 260.

550 - Sort the entries in the cross-reference table. Print the cross-reference listing suppressing the printing of duplicate entries and duplicate references on the same line. An asterisk (*) in position 6 of the entry indicates a numeric string which is a statement definition. The asterisk is used to force that entry to sort first within all entries that reference that same statement number. The asterisk is blanked out before printing. When the listing is complete, if an EOF was found in the input, go to 555 else go to 20.

555 - Close the files and return to the calling program.

CALLING SEQUENCE:
```
        L     R15,XREFIT
        BALR  R14,R15
          .
          .
XREFIT  DC    V(XREFIT)
```

OPTIONS: The source listing can be suppressed through the use of the UPSI control card. A "1" in the control card indicates that no source listing is wanted; "0" indicates that the source listing is to be printed. If the UPSI card is omitted, the default is to provide both source and cross-reference listings.

CALLED BY: REFM

SCREENS USED: None.

INPUTS: XREFIT calls COMREG to get the system date and the setting from the UPSI card. XREFIT uses FORTRAN source decks (which in IOC normally start with the "SUBROUTINE" card and end with the "END" card), either one at a time or several source decks in series, to produce the source listings and cross-reference listings.

OUTPUTS: The outputs are the source listings and the cross-reference listings.

ENTRY POINTS:

XREFIT

PROGRAM CONSTANTS:

IBLANK - 8 blank bytes.

PAGE - "PAGE". Literal for page numbering.

HEADNG - Page headers for deck listing and cross-reference listing.

MAXSIZ - Maximum number of cross-reference table entries. Value is 5500.

LETC - "C⌀⌀⌀". Literal used to check for comment card.

ENDCD - "END⌀". Literal used to check for END card.

FORMT - "FORMAT⌀⌀". Literal used to check for FORMAT statement.

COMMN - "COMMON⌀⌀". Literal used to check for COMMON statement.

RESWRD - List of reserved words which are ignored in the cross-reference listing.

STAR - "*ØØØ". Literal used to indicate that a numeric string is a statement definition.

LINEMX - Maximum lines per page. Value is 52.

PROGRAM VARIABLES:

INPUT - Work area where a complete statement is put together by concatenating the contents of the first card and up to 19 continuation cards.

CARD - Buffer area into which a card is read.

PRTLIN - 132-character print line buffer.

LINECH - Receiving field for the decimal equivalent of a binary number.

WORK - Contains a binary line number which is to be converted to decimal.

CURNM - The name for which the current cross-reference line is being generated.

CURNM2 - The name from the current entry of the cross-reference array.

IFIRST - First word of statement indicator.
0 = First word not processed yet.
1 = Comment card.

ICARD - Counter for continuation card. Maximum valid value is 19.

IEOF - End of File indicator.
0 = No EOF found.
1 = EOF found; process last card.

DATE - 8-character system date.

LINECT - Counter for lines per page.

LISTFL - Source listing indicator.
0 = No source listing requested.
1 = Print source listing.

ILIST - Indicator for type of page header to use.
0 = "DECK LISTING"
1 = "CROSS REFERENCE LISTING"

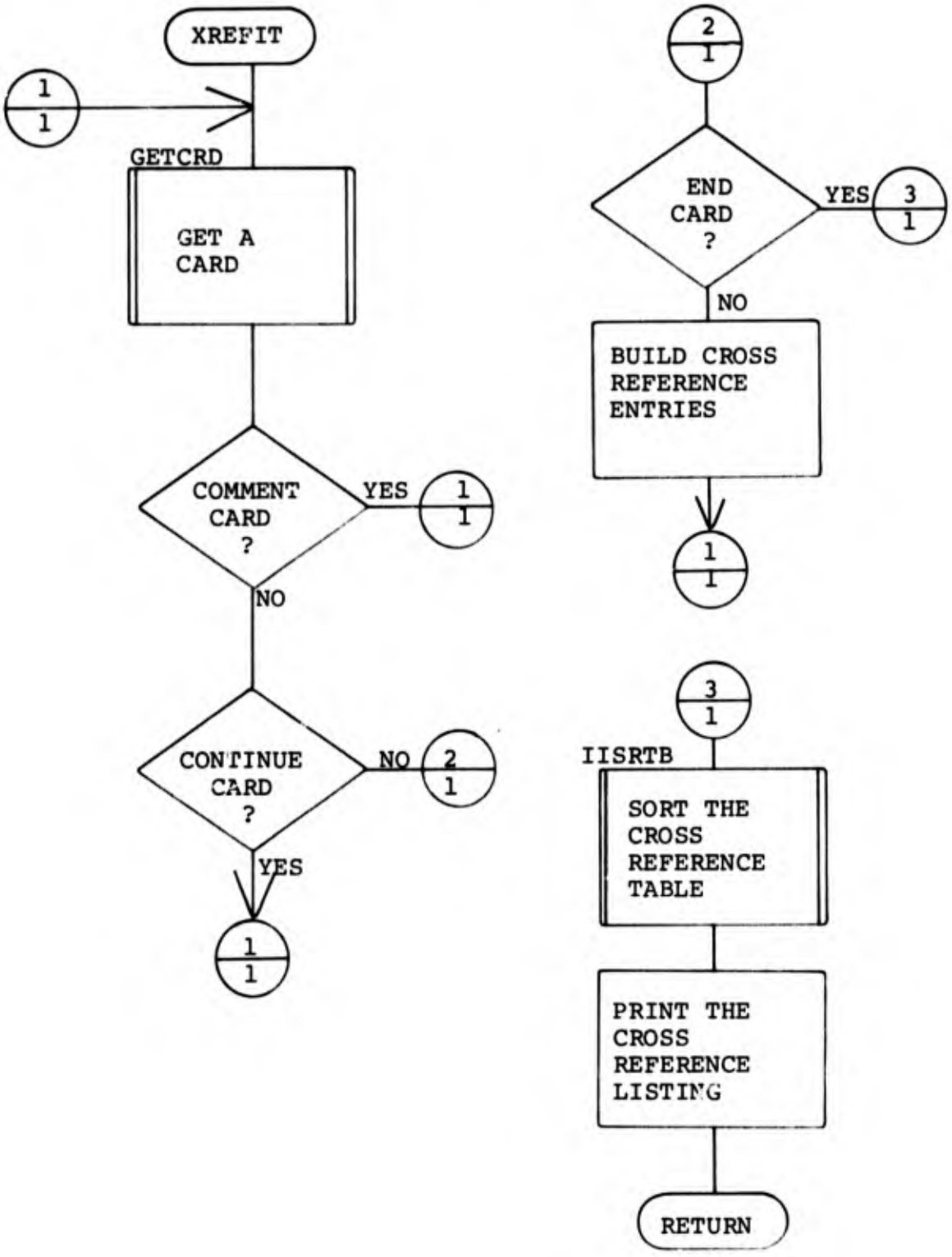
ARRAY - Cross reference table.

XARRAY - Offset 2 entries from ARRAY. The sorted entries from ARRAY will end up in XARRAY.

ERROR MESSAGES: None.

SUBROUTINES CALLED: COMREG, CLOSER, GETCRD, IICBTD, IICOMM, IICOMF, IIFILL, IIMOVE, IISRTB, IITBL2, LKFORT, PUTLIN.

XREFIT: Logic Diagram



PROGRAM ID: XREFLW
PROGRAM NAME: Cross-Reference for FORTRAN Flowcharts

Company: Informatics, Inc.
Programmer: Raymond T. Isawa/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: XREFLW is a batch program that produces a cross-reference listing from a FORTRAN source deck that has been previously loaded on disk.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: XREFLW is basically the same program as XREFIT. The differences are:

- XREFLW will process only one FORTRAN source deck per run
- XREFLW is called by REFX instead of REFM
- XREFLW produces only a cross-reference listing and no source listing
- XREFLW calls a modified version of GETCRD which reads the source deck from a disk data set.

LOGIC NARRATIVE: Same as XREFIT except that after an END card is encountered, XREFLW does not look for any more input source decks. It produces the cross-reference listing, closes the files and returns to the calling program.

CALLING SEQUENCE:
```
          L     R15,XREFLW
          BALR  R14,R15
          .
          .
XREFLW    DC    V(XREFLW)
```

OPTIONS: None.

CALLED BY: REFX

SCREENS USED: None.

INPUTS: Same as XREFIT except that the FORTRAN source decks have been previously loaded on disk, each input card occupying a 100-byte record.

OUTPUTS: XREFLW produces a cross-reference listing.

ENTRY POINT:

XREFLW

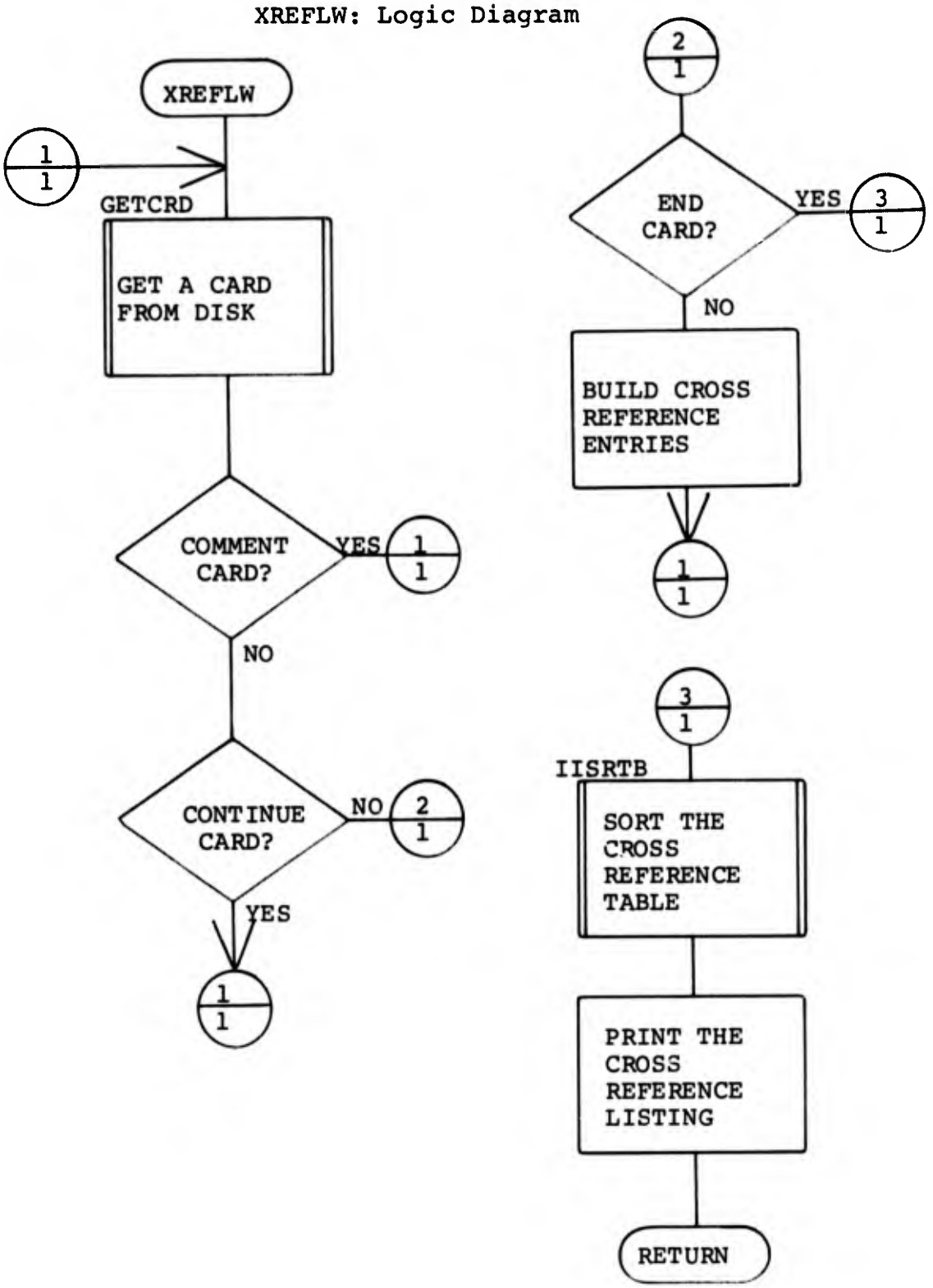
PROGRAM CONSTANTS: See XREFIT.

PROGRAM VARIABLES: See XREFIT.

ERROR MESSAGES: None.

SUBROUTINES CALLED: See XREFIT. Although the subroutine names are identical, the XREFLW version of GETCRD has been modified to read the source deck from disk.

XREFLW: Logic Diagram



PROGRAM ID: MAINPGM(FORFLOW)
PROGRAM NAME: Main Program for Flowcharting FORTRAN Routines

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: This program directs the flowcharting process. It builds the label tables, directs the building of the VLE's, performs the line nesting algorithm, performs the line spacing algorithm, and then calls DSPY to output the flowcharts.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The network consisting of lines connecting all related nodes in a FORTRAN program can be printed directly on the source listing, provided that the number of vertical lines at any one point on the listing does not exceed the number of print positions available. Nodes in a FORTRAN program consist of conditional and unconditional branches, DO statements, and all labelled statements which are referenced at least once. By identifying all labelled statements and branch statements, eventually all nodes in a program can be processed.

In the FORTRAN flowcharting program, FORFLOW, several conventions are observed:

1) Since the limited character set of the line printer rules out printing realistic arrow-heads, directed line segments to the right and upwards are represented by rows and columns of +'s. Directed line segments to the left and downwards are represented by rows and columns of -'s and 1's, respectively. To avoid confusion when lines cross each other, but do not connect, a separate symbol (0) is used to designate an actual connection.

2) All arithmetic IF, GO TO, computed GO TO, and DO statements are defined as "exit" statements or nodes. Each of these generate one, and only one, vertical line on a flowchart. This vertical line may be connected via negatively-directed horizontal line segments to more than one labelled statement, as in the case of a computed GO TO statement. Also, such a vertical line may be connected via positively-directed horizontal line segments to more than one "exit" statement, in the case when several vertical lines are combined.

3) The cards in the source deck are numbered consecutively. In subsequent processing, each card is stored in a record, and the card number of the first card in an "exit" statement is used as the identifier of the corresponding vertical line on a flowchart. This number is called the vertical line ID (VID).

The following is a general discussion of the strategy for identifying and processing the nodes and lines in a flowchart. Several passes through the source code, some random, some sequential, are necessary.

In the first pass, each source card is read and written onto disk, creating sequentially-numbered records. Each record consists of 100 bytes.

In this pass, some preprocessing is also done. COMMENT cards, continuation cards, and the END card are flagged by setting a code in the CODE field of each record. Also a count which points to the rightmost non-blank character position +1 is stored in the record. Certain tables are also built in core as this preprocessing is accomplished. The general strategy is to build a list of record numbers of all labelled statements and to build a table of the corresponding label values for these statements. The former table is called the LABVAL table and the latter is called the LABTAB table. In theory, this pass through the deck, and building the symbol tables could be skipped. However, since the cards must be copied to disk for subsequent random access, these tasks were done here.

In the second pass, each record is read, and it is determined if it is an "exit" statement. This is done by the routine VLEPRC. If it is not an "exit" statement, the next statement is processed. If it is, the labels referenced are stored in a table, and the routine VLECRT is called to create the appropriate vertical line event indicators (VLE's).

During this pass, several tables are built for later use in the line nesting algorithm and line spacing algorithm. These include the beginning point table (BEGPT), end point table (ENDPT) and corresponding vertical event table (VLETAB). Entries in these tables are sorted according to the BEGPT entry. Considering all three of these tables, each entry serves to

describe the "external" features of a vertical line. The VLETAB entry gives the VID of the line, and the BEGPT and ENDPT entries tell where the line begins and ends.

The next step in drawing the flowcharts is not really a pass through the source code. Rather, data accumulated in earlier passes is used to prepare for the final display of lines. Here, using the BEGPT, ENDPT, VLETAB tables, an algorithm for nesting of vertical lines is used to create a table of relative levels for these vertical lines. This table replaces the BEGPT entries. Then, using this table of levels, an algorithm is used to space the vertical lines out in a more readable format. This algorithm generates a level/line map called MAP.

The final pass through the source is performed by the routine DSPY. This routine uses two print-time line buffers to maintain the vertical lines and the final print line. DSPY is described in detail in the program documentation for DSPY.

In this section, only those topics with which the MAINPGM is concerned are discussed. Subsidiary functions such as the VLECRT function are described in the program documentation for those programs.

As noted earlier, the records created during preprocessing consist of 100 byte records. Each record has the following format:

bytes	1-72	card image
	73-76	entry VLE1
	77-80	entry VLE2
	81-84	entry VLE3
	85-88	entry VLE4
	89-92	entry VLE5
	93-96	exit VLE
	97-98	card code
	99-100	non-blank pointer

During preprocessing, the first 72 bytes of each record are scanned from the rightmost byte to determine the first non-blank character encountered. This task is performed by JICNT. Later, in DSPY, this pointer is used to define the starting point of a horizontal line on the flowchart. As described above, each record provides for up to five entry vertical line events (VLE), and one exit VLE. A VLE consists of the VID and an event type. There are four possible event types:

1) start a positive vertical line
2) start a negative vertical line
3) end a line
4) continue a line

The routine VLEPRC examines a given statement, and if it is an exit statement, it extracts the labels and builds a table. It then calls VLECRT. VLE's are created by the routine VLECRT during the second pass through the source code. Note that each statement can have only one positively-directed horizontal line, while it can have up to 5 vertical lines which connect to its negatively-directed horizontal line.

In order to make the final flowchart more readable, an algorithm for nesting lines was developed and implemented. This algorithm makes use of a 496 x 496 bit table called TABLE, and the (VLETAB, BEGPT, ENDPT) tables generated in earlier passes. Bits in TABLE have the following definition:

TABLE(I,J)=1 if vertical line I is contained by J, but not equal in length, or if I=J.

=0 if otherwise.

As can be seen, this table defines a partial-ordering over the set of vertical lines, since if we let S = the set of vertical lines, and

$x,y \in S,\ x \leq y \prec = \supset$ TABLE(x,y) = 1

Then,

1) $x \leq y$

2) $\forall x,y,z \in S,\ x \leq y,\ y \leq z = \supset\ x \leq z$

3) $\forall x,y \in S,\ x \leq y = \supset\ y \not\leq x$

Now, using TABLE, which is a relational matrix for a partially-ordered set, there are standard methods for finding nesting levels of elements. (In this case, finding levels in a flow diagram.) By adding the additional condition that the lines are sorted according to their beginning points, we can determine which vertical lines belong to which levels. The algorithm is as follows:

1) Pick a line which is not $\leq$ any other line except itself. When it is found, add it to the current level.

2) Delete this line from TABLE.

3) If that was the last line in the TABLE, start at the beginning of TABLE, but all lines go in the next level. Go to 1.

The result of performing the above algorithm is a table with the levels for corresponding vertical lines (identified by the VID's in VLETAB). The line nesting algorithm would be sufficient,

except for the fact that all lines will still be crowded into one side of the flow line space. So, a line spacing algorithm was implemented, the idea here was to create a line map (MAP) which mapped levels to line position. Note that using a line map permits postponing of the assignment of line positions until actual print-time. The line spacing algorithm chosen has the effect of grouping levels into small, more easily followed groups, which are spaced apart. The algorithm is given here:

Let N = maximum number of vertical lines at any point in the flowchart:

[I(a) is defined as the integer part of a.]

1) If $N \leq 13$, the vertical lines are spaced with $I\left(\frac{52-N}{N}\right)$ blanks between the vertical lines.

2) If $14 \leq N \leq 44$, the lines are grouped in groups of five, with spacing S within the groups, with the groups separated by B blanks. There are G "gaps" each consisting of B blanks.

If $14 \leq N \leq 20$ Set S=2, R=52-2N

If $21 \leq N \leq 44$ Set S=1, R=52-N

$$G = I\left(\frac{N-1}{5}\right)$$

$$B = I\left(\frac{R}{G}\right)$$

3) If $44 < N < 52$ Set S=1, R=52-N

$$G = R$$

$$B = 1$$

The number of lines per group is $I\left(\frac{N}{G+1}\right) + 1$

4) If N = 52 - each level occupies one line.

5) If $N > 52$ - there is no possible space solution.

LOGIC NARRATIVE:

MAINPGM - Output the FORFLOW heading.

2 - This is the main loop in the symbol table building phase. Each card is read, and the label table (LABTAB) and the label line table (LABVAL) are built. Also, comment cards, continuation cards, and the end card are marked.

14 - This is the loop in which each statement is processed. Continuation lines are put together into one line, and VLEPRC is called to process the line. VLEPRC analyzes the statement, and, if necessary, calls VLECRT to create VLE's.

40 - This is the end of the loop which clears out the bit table TABLE. The loop following this fills in the bit table as described in the PHILOSOPHY AND APPROACH section.

1000 - Using the algorithm outlined in the PHILOSOPHY AND APPROACH section, nesting levels are generated for each VLE.

142 - This is the end of the nesting levels generation loop. If not done with all levels, go to 2000. Otherwise, fall through to perform the line spacing algorithm. If there are exactly 52 levels, go to 144. If there are more than 13 levels, go to 1724. Otherwise, set space for lines evenly, with (52/total levels) spaces between them.

144 - Call DSPY to display the line. Then, end execution.

1724 - Perform the line spacing algorithm and set MAP as identified by the algorithm. Then, go to 144.

CALLING SEQUENCE: None.

OPTIONS: None.

CALLED BY: None. This is a main program.

SCREENS USED: None.

INPUTS: Card deck consisting of a FORTRAN subroutine or program.

OUTPUTS: The output consists of flowcharts, which are generated by DSPY.

PROGRAM CONSTANTS:

CEND - Contains 'END$\not\emptyset$'

LETC - Contains 'C$\not\emptyset\not\emptyset\not\emptyset$'

BLANKS - Contains '$\not\emptyset\not\emptyset\not\emptyset\not\emptyset\not\emptyset\not\emptyset\not\emptyset\not\emptyset\not\emptyset$'

COM - Contains 'CM'

COND - Contains 'CN'

ENDR - Contains 'EN'

IAST - Contains 72 *'s

LINE1 - Contains 'FORFLOW CONVENTIONS'

LINE2 - Contains second line of heading

LINE3 - Contains third line of heading

LINE4 - Contains fourth line of heading

PROGRAM VARIABLES:

ICARD - A 100-byte area for the entire data record.

CARD - A 72-byte area for the card image. This is in the first 72 bytes of ICARD.

TOTCRD - Area for building image of an entire statement, including continue cards.

MAP - Contains entries such that:
MAP(i) = actual line position of nesting level i, as determined in the line nesting algorithm.

The following variables are used in the line nesting algorithm:

SELLIN - Table of indexes into the (VLETAB/BEGPT) tables. This table is used to accumulate all of the VLE's in one level in order to generate the nesting levels.

TOT - Count of the total number of VLE's already processed in the nesting level algorithm.

SELCT - Count of number of VLE's in the current level.

ITOT - Sum of number of '1' bits in a row of the bit table.

LVLCT - Current level being processed.

The following variables are used in the line spacing algorithm:

IFACTR - The spacing between lines, when an even spacing is required.

IGROUP - Number of lines per group.

IGAP - Number of gaps between groups.

ISPACE - Spacing between lines within a group of lines.

IREM - Number of remaining lines other than in fixed groups.

LINGP - Number of lines per group, including spacing.

IBLAN - Number of lines in a gap.

NUMGRP - Number of the group in which a level appears.

In addition to the above, several areas of common are used in MAINPGM(FORFLOW). The common area LABLS contains:

TABSZE - Count of entries in label tables.

LABTAB - Table of labels, in binary.

LABVAL - Table of label line numbers.

This area is used as a symbol table to find the location of the statement corresponding to the label.

The area TABLE contains:

TABLE - Area for 480 x 480 bit table, which is used as the partial ordering table in the line nesting algorithm.

The common area LINTAB is shared between MAINPGM and VLECRT. This area is used to hold the following:

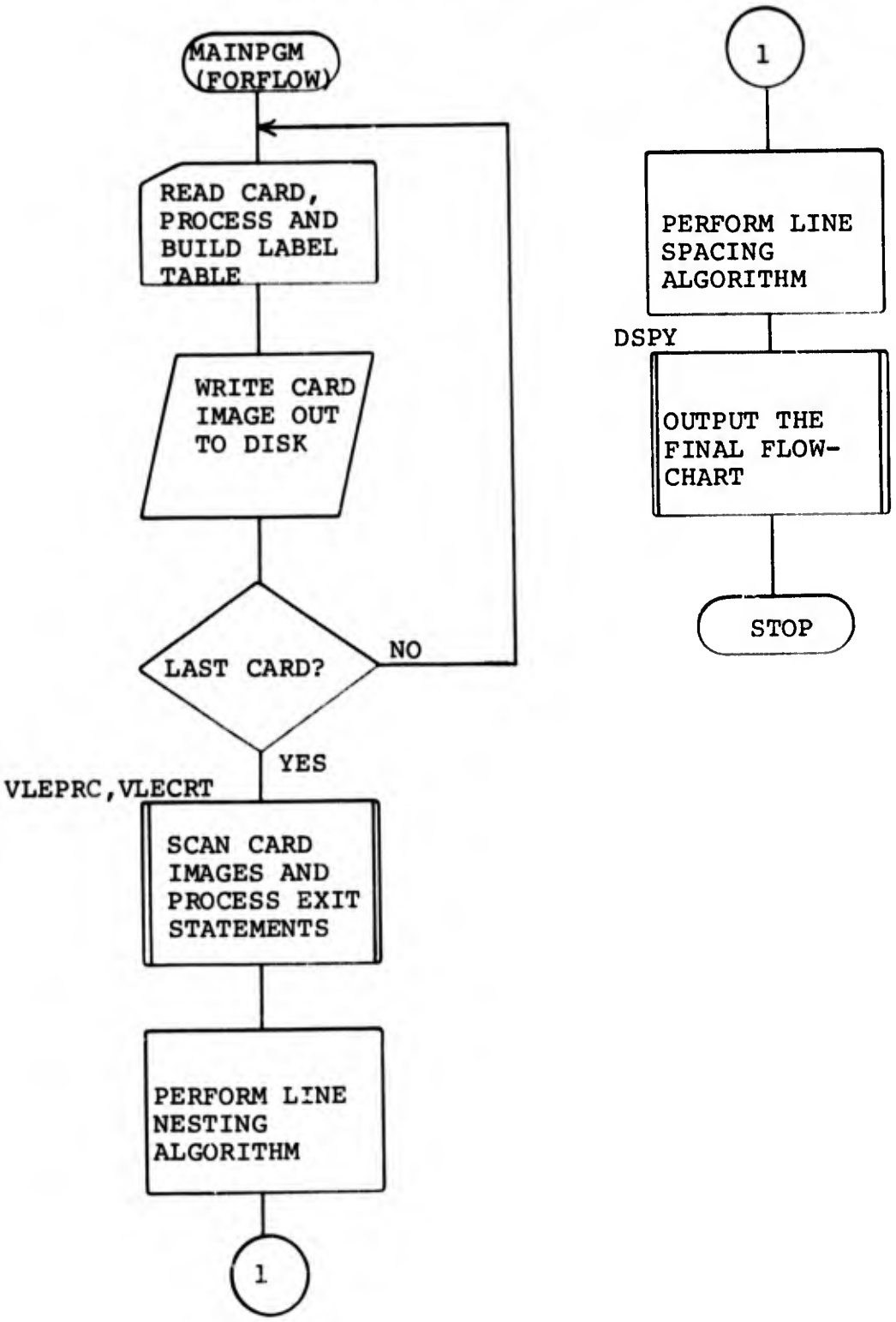
VLETAB	-	Table of VID's for vertical lines.
ENDPT	-	Table of end points for vertical lines in VLETAB.
BEGPT	-	Table of beginning points for vertical lines in VLETAB.
VLECT	-	Count of number of entries in VLETAB.

VLECRT builds the tables above each time it is called. Then, MAINPGM uses them to create the bit table, TABLE. Finally, the TABLE is used to generate nesting levels.

ERROR MESSAGES: None.

SUBROUTINES CALLED: DSPY, IICDTB, IICOMF, IICOMM, IIFILL, IIMOVE, JICNT, TABFIL, TEST, VLEPRC, ZERO.

MAINPGM(FORFLOW): Logic Diagram



PROGRAM ID: MAINPGM(ALCFLOW)
PROGRAM NAME: Main Program for Flowcharting Assembly Programs

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: This is the base phase for the flowcharting process. For further details, see the PHILOSOPHY AND APPROACH section.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: The general structure which was used to generate flowcharts for FORTRAN programs was used to generate Assembly Language flowcharts. Because of the structure of the Assembly Language, several modifications in the process were made. These are described here.

In order to generalize the handling of Assembly programs, the ALCFLOW process was designed to make use of the listing generated by the assembler as input. This was done by assigning SYSLST to a disk area which is later assigned to an input file. This has several advantages:

1. Since the assembly listing includes all entry addresses related to exit statements, a syntax analysis step is not needed. Thus, VLEPRC is not used in the ALCFLOW.

2. Since the assembly listing exit addresses relate to addresses which are also on the listing, a symbol table is not needed. Instead, a table of addresses is used to relate addresses to line numbers.

3. The assembler generates a cross-reference listing.

Because the above method was utilized several edit steps had to be done in order to eliminate lines generated in the SYSLST area, such as page headings, etc. This is done in the first phase. Lines which are to be ignored in the flowcharting process are marked, and in a later phase, DSPY checks for these marks.

Also, because the output of the assembler is used as input, the size of the records used was altered. Instead of being 100 bytes/card, the records for ALCFLOW are 120 bytes. During the first phase of the ALCFLOW process, the format of these records is changed.

Initially, the output record of the assembler consists of the following format:

bytes	1	carriage control
	2-7	relative location
	8	'0'
	9-10	operation code
	11-29	various parts of assembler output
	30-34	"branch to" address
	35-36	'00'
	37-40	line number
	41	macro expansion indicator
	42-113	card image

After the address table build and edit step, the format of each record is:

bytes	1- 72	-	card image
	73- 76	-	entry VLE1
	77- 80	-	entry VLE2
	81- 84	-	entry VLE3
	85- 88	-	entry VLE4
	89- 92	-	entry VLE5
	93- 96	-	exit VLE
	97- 98	-	non-blank character pointer
	99-100	-	card-type code
	101-104	-	"branch to" address
	105-108	-	line number
	109-112	-	operation code
	113-116	-	macro expansion indicator

It should be noted that the first 100 bytes of the record are in exactly the same format as used in FORFLOW. The extra 5 words are used in order to save data from the assembly listing.

As in FORFLOW, the second phase of the process consists of processing each statement, and generating the appropriate VLE's. This is done by using the contents of the 5 words above to determine if VLECRT is needed. If so, it is called.

The next phase, that of generating the partial ordering table, and performing the line nesting and line spacing algorithm, is the same as used in FORFLOW.

The final processing step, that of generating the output listing, is done by DSPY. Some modifications of DSPY (FORFLOW) were needed. The main changes were that DSPY(ALCFLOW) had to recognize and skip processing of lines that were to be ignored. This is true since the record format was left unchanged. Another change was that DSPY had to recognize the end of the assembly listing, and print the cross-reference which followed.

Since much of DSPY has remained unaltered between the FORFLOW version and the ALCFLOW version, a complete write-up for it was not done. Instead, documentation for it references points in DSPY(FORFLOW).

LOGIC NARRATIVE:

MAINPGM - Print flowchart heading. Set initial values for processing.

2 - This is the first phase of the process. The IADDTB is built, and all lines to be ignored are marked.

4 - The VLE create phase begins here. Depending on the operation code generated by the assembler, the following branches take place:

BAL	-	go to 1505
BC	-	go to 1507
BCT	-	create the VLE for this line by calling VLECRT, then go to 150
Otherwise	-	go to 150

1505 - Check if this is an addressed instruction. If so, go to 1506. If it contains an exit statement in the displacement, base register notation, generate a dummy VLE, and go to 150.

1507 - Check if this is a BC instruction. If so, go to 1506. If it is in the D(B) notation, generate a dummy VLE, then go to 150.

1506 - Call VLECRT to process the statement.

150 - This is the end of the DO loop which begins at location 4.

40 - The line nesting algorithm is performed, using data generated in the previous phases. When this is done, go to 1724.

144 - A call to DSPY takes place, to generate the output, then STOP.

1724 - The line spacing algorithm takes place. When it is done, go to 144.

CALLING SEQUENCE: None.

OPTIONS: None.

CALLED BY: None.

SCREENS USED: None.

INPUTS: Input to the ALCFLOW process consists of the SYSLST file generated by the assembler.

OUTPUTS: The output of the ALCFLOW process consists of three things:

1) Source listing
2) Flow lines
3) Cross-reference listing.

PROGRAM CONSTANTS:

LINE1 - Line 1 of the flowchart heading.

LINE2 - Line 2 of the flowchart heading.

LINE3 - Line 3 of the flowchart heading.

LINE4 - Line 4 of the flowchart heading.

LINE5 - Line 5 of the flowchart heading.

LINE6 - Line 6 of the flowchart heading.

IAST - Contains 42 *'s.

COM - Contains 'ØØCM'.

COND - Contains 'ØØCN'.

IPAGE - Contains 'PAGE', used by program to locate page headings.

ILOC - Contains 'LOCØ', used to locate listing heading.

PROGRAM VARIABLES:

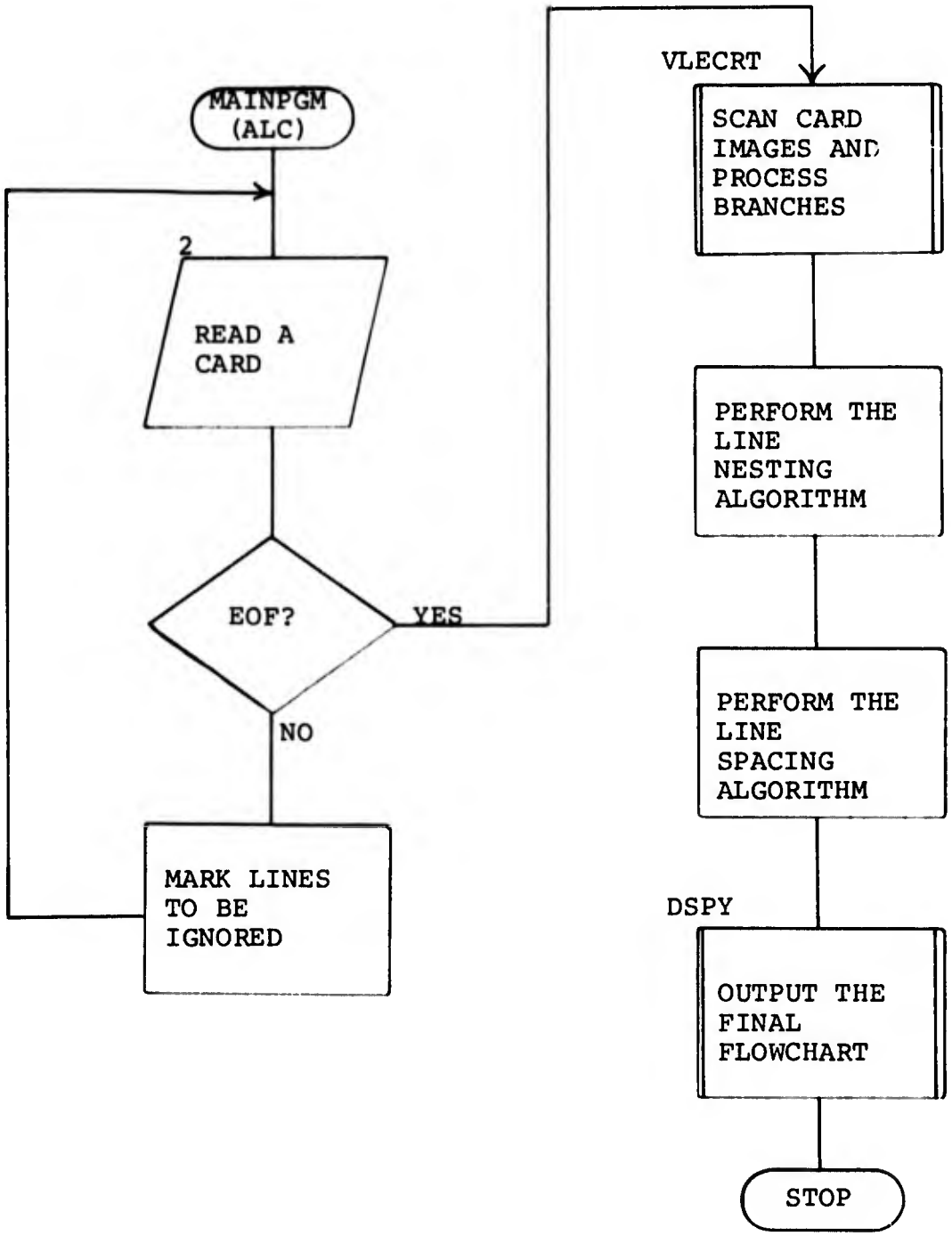
Variables in MAINPGM(ALCFLOW) are identical to those in MAINPGM(FORFLOW) except for the following:

ICROSS	-	Pointer to the beginning line of the cross-reference listing.
INLINE	-	ICROSS-1.
IADDTB	-	Table containing addresses related to lines.

ERROR CONDITIONS: None.

SUBROUTINES CALLED: BAL, BC, IICOMF, IIFILL, JICNT, TABFIL, TEST, VLECRT, VLEENT, ZERO.

MAINPGM(ALCLFOW; : Logic Diagram



PROGRAM ID: DSPY(ALCFLOW)
PROGRAM NAME: Display Flowchart for Assembly Program

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: FORTRAN
COMPUTER: IBM 360, Model 40
FUNCTION: This program processes the records generated in the ALCFLOW process, and outputs the appropriate flow lines and cross-reference.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: For details on DSPY(ALCFLOW), see notes in the PHILOSOPHY AND APPROACH section of MAINPGM(ALCFLOW) and DSPY(FORFLOW).

LOGIC NARRATIVE: As noted, the logic for DSPY(ALCFLOW) is essentially the same as in DSPY(FORFLOW).

CALLING SEQUENCE: See DSPY(FORFLOW).

OPTIONS: None.

CALLED BY: MAINPGM(ALCFLOW)

SCREENS USED: None.

INPUTS: Records on disk, as explained in the PHILOSOPHY AND APPROACH above. Also, several variables and tables in common areas are used. These are the same as used in DSPY(FORFLOW).

OUTPUTS: Output consists of flow lines drawn along side the listing of the source program, along with a cross-reference.

PROGRAM CONSTANTS:

IMINUS	-	Contains '--' used to check for ignored line.
COM	-	Contains 'CM'.
COND	-	Contains 'CN'.
BLANK	-	Contains 'ØØØØ'.
PLUSS	-	Contains 72 +'s.
MINUSS	-	Contains 72 -'s.
IBTSYM	-	Contains '0ØØØ'.
NOST	-	Contains 'NOØS'.
EXITR	-	Contains '+++++ BRANCH D(B)'.

PROGRAM VARIABLES:

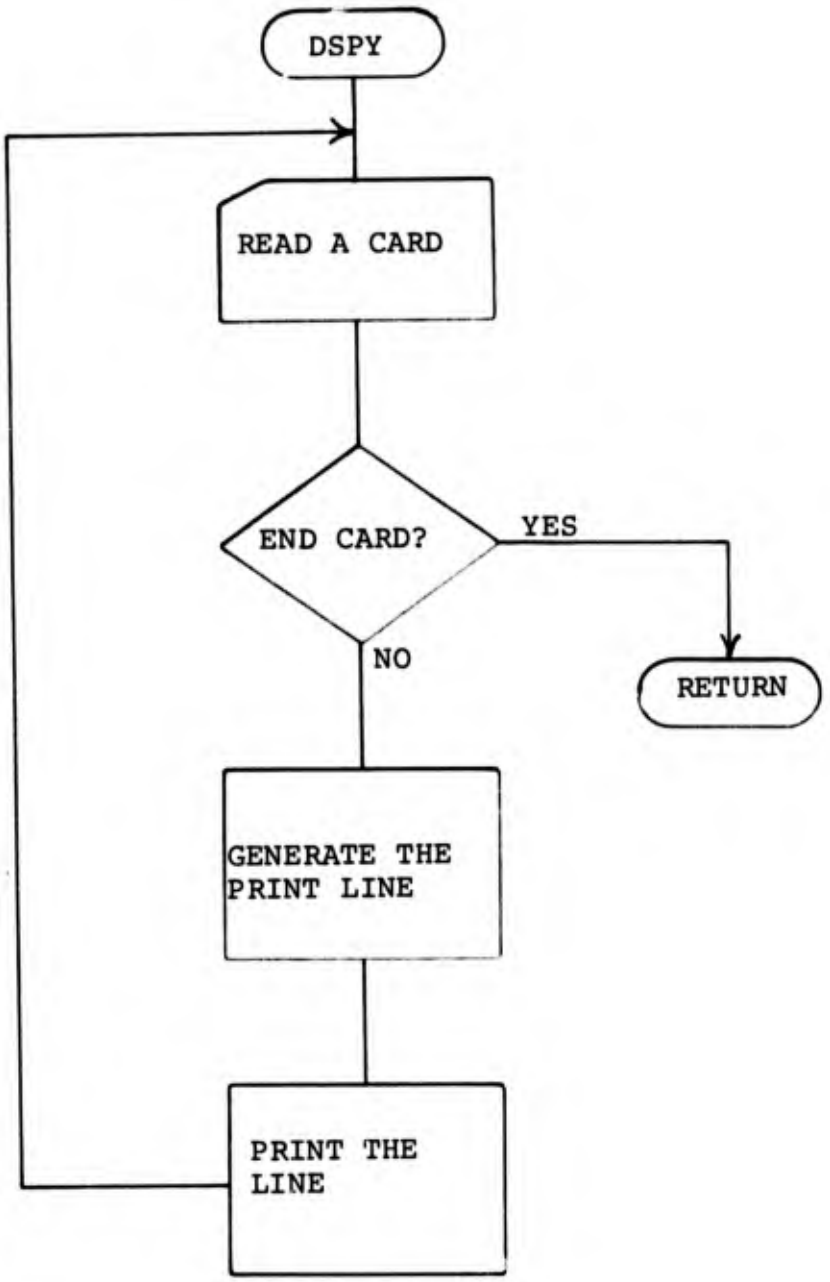
Note: Only variables which differ from those used in DSPY(FORFLOW) are defined here.

INLINE	-	Pointer to the line on which the cross-reference listing begins.
INLIN1	-	Points to first header line of cross-reference.
INLIN2	-	Points to second header line of cross-reference.

ERROR MESSAGES: None.

SUBROUTINES CALLED: See DSPY(FORFLOW).

DSPY: Logic Diagram (ALCFLOW)



PROGRAM ID: BAL
PROGRAM NAME: Check for Displacement/Base Register Notation

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: Given an assembly statement, BAL determines if the branch is in displacement/base register notation.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: When the flowchart system was designed, it was desired to keep the system as modular as possible. By moving most of the syntax analysis routines out of the MAINPGM, this end was achieved. BAL determines whether or not a BAL instruction is in base/displacement notation. If it is, a branch line cannot be generated. Thus, a VLE with a 0-VID is generated. Later, in DSPY, when a 0-VID is found, a special horizontal line is generated.

LOGIC NARRATIVE:

BAL	-	Set register 3 to 0, register 1 pointing to LINE, and register 2 pointing to I.
COMMA	-	Find the first ',' in the LINE.
FIELD	-	If the first character is not a number, go to NOTEXIT. Otherwise, set register 3 to 1.
NOTEXIT	-	Store register 3 into 1, and return to the calling routine.

CALLING SEQUENCE: CALL BAL(LINE,I)

where:

LINE	-	Assembly language statement.
I	-	Indicates whether statement is in displacement/base register notation. 1 = It is in that notation. 0 = It is not in that notation.

OPTIONS: None.

CALLED BY: MAINPGM(ALCFLOW)

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. LINE is the input to BAL.

OUTPUTS: See CALLING SEQUENCE. I is output from BAL.

REGISTER USAGE:

R15	-	Base register.
R1	-	Pointer to character in LINE being analyzed.
R2	-	Pointer to I.

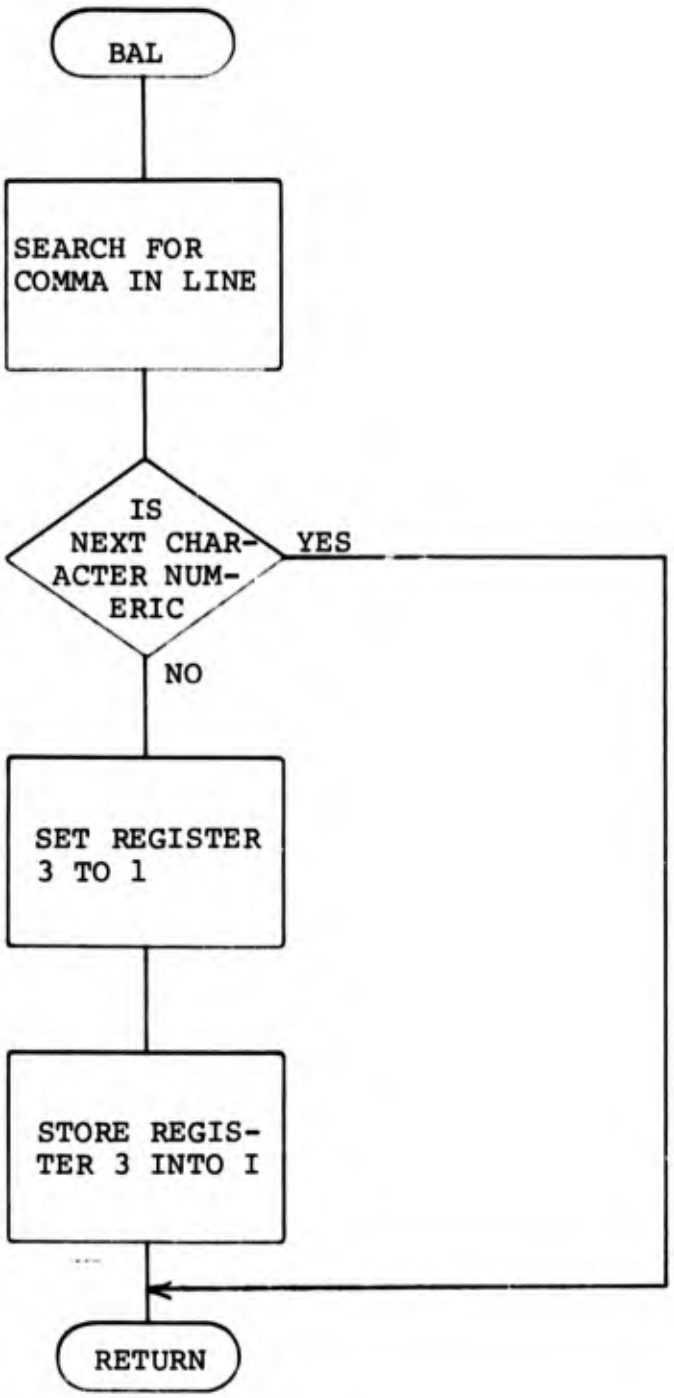
PROGRAM CONSTANTS: None.

PROGRAM VARIABLES: None.

ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

BAL: Logic Diagram



PROGRAM ID: BC
PROGRAM NAME: Analyze a BC Instruction

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: BC serves to analyze an instruction with a X'47' operation code.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: In the flowcharting process, it was desired to have the syntax analysis functions segregated from the main logic. Thus, the functions were developed into routines such as BAL and BC.

LOGIC NARRATIVE:

BC	-	Set registers to initial values.
LABEL	-	Step R1 past any labels to the blank field.
NOLABEL	-	Step R1 to the first op code field. If the code is an extended mnemonic, go to EXTEND. Otherwise, go to NONEXT.
EXTEND	-	Step R1 past the op code field.
BLANK	-	Step R1 past the blanks, then go to FIELD1.
NONEXT	-	Step R1 to the first ','. When found, go to FIELD.
FIELD	-	Increase R1 by 1.

FIELD1 - Test the next character. If it is not a numeric, go to NOTEXIT. Otherwise, set R3 to 1.

NOTEXIT - Store R3 into I. Return to the calling routine.

CALLING SEQUENCE: CALL BC(LINE,I)

where:

LINE - Card image.

I - Indicates whether a BC instruction is in base/displacement notation. 1 = It is in that notation; 0 = It is not.

OPTIONS: None.

CALLED BY: MAINPGM(ALCFLOW)

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE. LINE is the input to BC.

OUTPUTS: See CALLING SEQUENCE. I is output.

REGISTER USAGE:

R1 - Points to character in LINE.

R2 - Points to I.

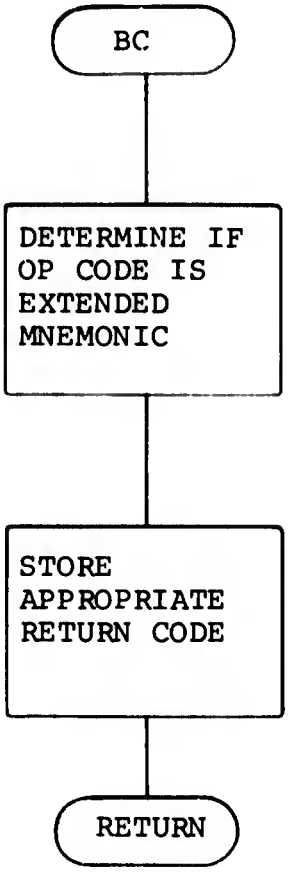
PROGRAM CONSTANTS: None.

PROGRAM VARIABLES: None.

ERROR MESSAGES: None.

SUBROUTINES CALLED: None.

BC: Logic Diagram



PROGRAM ID: VLEPRC
PROGRAM NAME: Process Statements and VLE's

Company: Informatics, Inc.
Programmer: James C.P. Lum/(301)770-3000
Contract: AF30602-72-C-0205
Sponsored By: Rome Air Development Center
Project Engineer: Jerald M. Plante

ABSTRACT:

LANGUAGE: 360 Assembly
COMPUTER: IBM 360, Model 40
FUNCTION: Given a FORTRAN statement, VLEPRC will process this statement, and create the appropriate VLE's.

COMPUTER DEFINITION: The hardware elements used are: Model 2314 disk units; Model 2400-series tape units (7-track); Model 1403 printer; GERBER 1000-series flat-bed plotter; and a Bunker-Ramo BR90 CRT console.

SYSTEM DESCRIPTION: DOS Release 26, modified to provide Roll-In/Roll-Out capability in F2, as installed in PACAF IDHS computer.

PROGRAM DESCRIPTION:

PHILOSOPHY AND APPROACH: In designing the flowcharting system for FORTRAN programs, it was decided to make the entire system of routines as modular as possible. Under this concept, the VLEPRC routine was the syntax analyzer portion of the process. It should be noted that, in many ways, the flowcharting process is a subset of a compilation process. This is most noticeable in the function of VLEPRC.

LOGIC NARRATIVE:

VLEPRC - Set up to process statement. Save value of I.

BEGIN, TRYAGIN - Check current string. If it is a DO statement, go to DOIT. If it is a GO TO statement, go to GOIT. If it is an IF statement, go to IFIT. If it is a RETURN statement, go to EXITRS. If it is a STOP statement, go to EXITRS. If a non-blank character, go to NOTEXIT. Otherwise, up the string pointer, and go to TRYAGIN.

DOIT - Process the DO statement, and call VLECRT as needed.

NOTEXIT - Return to the calling routine.

GOIT - Process the GO TO statement. On completion, return to the calling routine via NOTEXIT.

IFIT - Process the IF statements. On completion, return to the calling routing via NOTEXIT.

EXTRACT - Extract the label character string from the statement, and convert it to binary. This routine is used throughout VLEPRC.

CALLING SEQUENCE: CALL VLEPRC(TOTCRD,I)

where:

TOTCRD - 500 word area which holds the card image of the FORTRAN source statement.

I - Line number of the FORTRAN source statement.

OPTIONS: None.

CALLED BY: MAINPGM(FORFLOW)

SCREENS USED: None.

INPUTS: See CALLING SEQUENCE.

OUTPUTS: If the statement being processed is an exit statement, VLEPRC creates VLE's in the exit and all related entry statements.

PROGRAM CONSTANTS:

ONE - Contains 1.

DO - Contains 'DOGOIF'.

TO - Contains 'TO'.

RETURN - Contains 'RETURN'.

STOP - Contains 'STOPbbbb'.

ZERO - Contains 0.

PROGRAM VARIABLES:

LABELS - A table of entry statement labels, which is built up by VLEPRC prior to calling VLECRT.

LABLCT - Count of entries in LABELS.

LABELS1 - Table of up to 3 entry labels, used for processing arithmetic IF's.

THREE - Count of entries in LABELS1.

LENGTH - Length of label in LABEL.

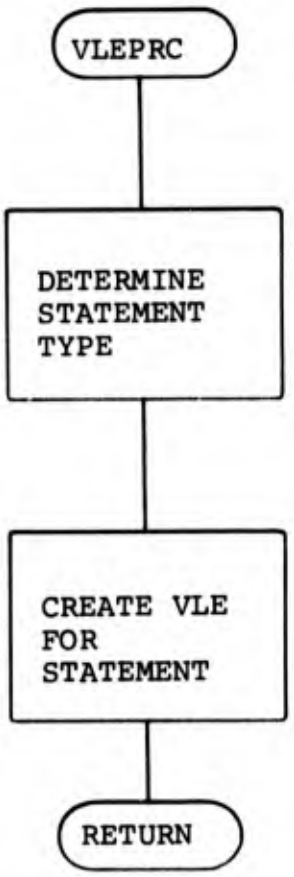
LABEL - Contains alphanumeric label to be converted to its numeric value.

I - Contains line number of line being processed.

ERROR MESSAGES: None.

SUBROUTINES CALLED: IICDTB, VLECRT, VLEENT.

VLEPRC: Logic Diagram



CHAPTER XVI

INDEX

Page number references in this index are to the first page on which a multi-page discussion appears. Should other material intervene, page reference is made to the first page of the next appearance of that topic. Thus, if a topic appears on pages 3-4, 3-5, 4-1 through 4-6, 4-10, and 6-1 through 6-4, it would be indexed as 3-4, 4-1, 4-10, 6-1.

MISSION
of
Rome Air Development Center

RADC is the principal AFSC organization charged with planning and executing the USAF exploratory and advanced development programs for electromagnetic intelligence techniques, reliability and compatibility techniques for electronic systems, electromagnetic transmission and reception, ground based surveillance, ground communications, information displays and information processing. This Center provides technical or management assistance in support of studies, analyses, development planning activities, acquisition, test, evaluation, modification, and operation of aerospace systems and related equipment.