SYSTEM UTILITIES
SOFTWARE SUBSYSTEM DESCRIPTION
EXTENDED OPERATING SYSTEM
3A PROCESSOR

CONTENTS PAGE

CONTENTS PAGE

CONTENTS PAGE

Figures

Table PAGE

1. GENERAL

1.01 This section provides a description of the common utility functions used with the 3A Central Control (3A CC). The utility programs are both resident and nonresident and are used for loading, storing and/or monitoring various registers and memory locations, and performing various tape operations and reload functions. Utility programs are initiated by TTY messages. Output can be either a return TTY message or a system status panel display of data.

1.02 ▶This section has been reissued to reflect changes and improvements to system utilities. Revision arrows have been used to denote the changes.◀

1.03 The following sections contain the descriptions of functions related to this section.

SECTION	TITLE
254-340-001	Extended Operating System Overview, Software Subsystem Description, 3A Processor
254-340-040	Data Administration, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-080	Maintenance Overview, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-084	Resident Maintenance, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-086	Initialization and Recovery, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-088	Processor and Memory Diagnostics, Software Subsystem Description, Extended Operating System, 3A Processor

SECTION	TITLE
254-340-090	Peripheral Diagnostics, Software Subsystem Description, Extended Operating System, 3A Processor
254-340-104	Program Listing Organization and Usage, Software Subsystem Description, Extended Operating System, 3A Processor.

1.04 The following assembly units contain the code relating to the functions described in this section.

- Common System Utility Program (CUTIL)-PR-4C622—This program contains the subroutines that are used for loading, monitoring, or dumping functions on selected memory locations or registers.
- Common Nonresident Utilities Program (CNRUTL)-PR-4C623—This program contains the subroutines that are used to perform the patching and overwrite functions.
- Common Base Level Monitor Program (CBLM)-PR-4C617—Provides monitor control for the base level routines.
- EOS Maintenance Task (MAINT)-PR-4C607—Provides control for EOS maintenance functions.
- Client to Data Administrator (EOSMSG)-PR-4C229—Interfaces client input TTY messages to the proper processing routines.
- User TTY Data Program (TTYTBL)-PR-4C401—Contains message formats and processing routine addresses.

A. Utility Functional Overview

1.05 System utilities are initiated via TTY messages which are processed by independent subroutines which set up the utility request buffer (UREQBUF) for all valid messages. Pending utility requests are recognized by MAINT during high level processing. When a valid request is present, MAINT transfers the message to the utility program CUTIL. Program unit CUTILBAS in CUTIL validates the request and establishes the conditions necessary to invoke the requested utility. When MAINT next performs high priority functions, the pending function which has been requested will be performed. When the request is for a nonresident utility, the TTY message entered at medium priority level is passed directly to the Multiscan Function Controller (MSFC) of the CBLM, which also runs at medium priority level. The request is accepted (if it has not been inhibited) by MSFC and the requested utility is identified. It will execute when all other requests for higher priority multiscan functions have completed. The EOS does not time slice functions under MSFC. The primary utility functional flow is shown in Fig. 1.

B. Program Structure

1.06 The system utilities consist of a set of independent subroutines contained in two assembly units: Common Utilities (CUTIL) and Common Nonresident Utilities (CNRUTL). These subroutines are selected in accordance with an input TTY message. CNRUTL is first invoked as a single multiscan function after which the common processing determines which function (subroutine) to initiate. The overall structure of the system utilities is shown in Fig. 2.

1.07 ⧫The entry point of each TTY function is listed in Table A.⧫

2. COMMON UTILITY PROCESSING

A. Common Utility TTY Subroutines

2.01 There is a TTY input subroutine for each utility function processed by CUTIL. All input parameters are set up by the TTY program (EOSMSG), and the subroutines contain the TTY acknowledgment codes which they return in addition to basic processing.

Set Match Message

2.02 The SET:MATCH utility message is entered at entry point SETMATCH in CUTIL. This message loads the Central Control registers which control the panel address and data match circuits. The ADRMAT and DATAMAT flags are set so that other utility input routines can determine if matchers were set, thus allowing ON CONDITION (ONC) functions. This message does not enable the matcher circuits however. The message format is:

SET:MATCH:ADR(AA,MM),DATA(DD,NN)!

AA—Address to which address matcher is to be set
MM—Mask for address matcher
DD—Data to which data matcher is to be set
NN—Mask for data matcher.

Set Indirect Message

2.03 The SET:INDIR utility message is entered at entry point SETINDIR in CUTIL. It initializes the utility control block with data which is used by the indirect utility functions to generate an indirect address. This is accomplished by modifying the pointer address with the entered data before a load or monitor function is performed. The message format is:

- SET:INDIR MM,OO!

MM—20-bit mask to be applied to the address
OO—Decimal offset to be added to the address.

Stop Utility Message

2.04 The STOP:UTIL message is entered at entry point STPUTIL in CUTIL. It is used to terminate any active utility function. This message will disable the panel matchers, enable the automatic display, and idle the utility request buffer (UREQBUF). The update off-line main store function is marked allowable. The message format is:

STOP:UTIL!

Monitor Messages

2.05 The monitor input messages are entered at several entry points in CUTIL depending on the function requested. On a monitor store request, the entry is at MONST. For a monitor register request, the entry is at MONREG. For a monitor indirect, the entry is at MONINDIR. All of these entry points are in CUTIL. The monitor messages are used to monitor eight 16-bit data words. These words may be monitored once per loop, or only when a match occurs, as specified via the SET:MATCH message. For a MON:ST message, the default case is once per base-level loop. For a MON:REG or a MON:INDIR message, default execution is when a match occurs. Results may be directed to the TTY or display buffer with default being the TTY. When the result is directed to the TTY, the eight words will be all printed out the first time they are monitored. On subsequent cycles they will be printed only when a change is detected in one or more words. When the result goes to the display buffer, the first word of the eight is displayed each time the words are monitored.

2.06 Three forms of the monitor input message exist:

(1) The monitor store (MON:ST) message contains a store address. The eight words are the contents of that location and the seven succeeding locations. The message format is:

MON:ST AA;ONC,SGL,RDT LAMPS!

(2) The monitor register (MON:REG) message contains a general register number. The eight words are the contents of that register and the seven succeeding registers. When a succeeding register does not exist, the selection starts over from R0. The message format is:

MON:REG NN;ONC,SGL,RDT LAMPS!

(3) The monitor indirect (MON:INDIR) message contains a register number which is interpreted as the first of a pair of registers which will contain a 20-bit address when the match occurs. This address may be modified at execution time via the SET:INDIR message. When this occurs, the result is used as a store address. After this, the monitor function is identical to (1) MON:ST. The message format is:

MON:INDIR NN;ONC,SGL,RDT LAMPS!

(4) The following parameters are specified for the monitor messages:

AA—Address of first of eight store words
NN—Register number (0 through 31)
ONC—Execute ON CONDITION, ie, on a match
SGL—Execute a single time
RDT LAMPS—Results directed to the display buffer.

Load Messages

2.07 The load messages are basically the same as the monitor messages. They also enter

at various entry points depending on the function requested. For a load store request, the entry is LODST. For a load register request, the entry is at LODREG. For a load indirect, the entry is at LODINDIR. All of these entry points are in CUTIL. The three load messages exist on a one-to-one basis with the monitor messages and in each case the load function is identical to the corresponding monitor function, except that the first of the eight words is modified after the monitor is performed. Only unprotected store words and general registers may be modified.

2.08 Three forms of the load input message exist:

(1) The load store (LOD:ST) message contains a store address. The eight words are the contents of that location and the seven succeeding locations. The first word is modified by the data specified in the message. The message format is:

LOD:ST AA;ONC,SGL,RDT LAMPS:DD,MM!

(2) The load register (LOD:REG) message contains a general register number and the eight words monitored are the succeeding seven registers. If a succeeding register does not exist, the selection starts over from R0. The first register is modified by the data and mask information contained in the message after the monitoring is complete. The message format is:

LOD:REG NN;ONC,SGL,RDT LAMPS:DD,MM!

(3) The load indirect (LOD:INDIR) message contains a general register number which is interpreted as the register address when a match occurs. The action is then the same as a load register function except that the register contained in the address will be modified. The message format is:

LOD:INDIR NN;ONC,SLG,RDT LAMPS: DD,MM!

(4) The following parameters apply to the load messages:

AA—Address of first of eight store words to be located
NN—Register number (0 through 15)
ONC—Execute ON CONDITION, ie, on a match
SGL—Execute a single time
RDT LAMPS—Results directed to the display buffer
DD—Data to be loaded
MM—Mask which modifies the data.

Dump Messages

2.09 The dump store messages enter at various entry points depending on the function requested. The dump store (DMP:ST) message enters at DMPST. The dump off-line store (DMP:OFLST) message enters at DMPOFLST. The dump on initialization (DMP:INITQ) message enters at DMPINITQ. These messages are used to print out on the TTY the contents of a block of consecutive store words. A dump can be initiated immediately or when a match occurs, if the SET:MATCH utility message was used prior to the request. The starting address and length of the block of store to dump is specified in the message. The length is always rounded off to the next multiple of eight words. If the block length is omitted from the message, the default block size is eight.

2.10 Three separate input messages are used to initiate dump utility functions:

(1) The dump store (DMP:ST) message is the normally used dump routine. It is adequate for dumping small blocks (less than 16 words) or table large blocks. When 16 or fewer words are requested by DMP:ST, the words are buffered at the time the dump is initiated, thus are consistent. When more than 16 words are requested, the dump is directly from the memory locations and the words could be changing even as the printing is being done. This could result in a printout of data which has never really existed. Only very stable (slow to change) large blocks should be dumped using this message. The message format is:

DMP:ST AA,LENGTH NN;ONC!

(2) The dump off-line store (DMP:OFLST) message is used when large blocks of guaranteed consistency must be dumped. The message takes the system out of the up-date mode thus freezing off-line store. The entire off-line store is then used as a buffer, from which the printout

routine can access large blocks of data. The message format is:

DMP:OFLST AA,LENGTH NN;ONC!

(3) The dump on initialization (DMP:INITQ) message is functionally identical to the DMP:OFLST message, except that the dump is initiated only when an initialization occurs. The message format is:

DMP:INITQ AA,LENGTH NN;ONC!

(4) The following parameters apply to all dump utility messages:

AA—Address of first word in block to be dumped
NN—Decimal length of block to be dumped
ONC—Execute ON CONDITION, ie, on a match.

B. Utility Routine Entry Processing

Message Processing

2.11 Message entry processing (Fig. 3) begins at the various entry points and consists of locating the last monitored word (MONST and LODST), verifying registers (LODREG and MONREG), initializing masks and offsets (LODINDIR and MONINDIR), and loading the proper function code for an initialization or off-line dump (DMPINITQ or DMPOFLST), or for a normal store dump (DMPST). When these functions are completed, a branch is made to one of the INMERGE entry points in DMPST. Conditions are set based on the message and its parameters. A bid is then made for a utility function. The actual request is initiated by the routine UREQ which performs a series of checks to determine that the proper conditions exist to support the requested utility function. When these checks are complete, a return is made to DMPST and appropriate action is taken based on the response, ie, no good, retry later, or in progress. The appropriate TTY response is printed out and control then passes to the base level monitor (CBLM).

Utility Request

2.12 There are two entries into the utility processing routines depending on whether or not the ONC and match option was specified (Fig. 3). When the set match has been used and the ONC option selected, entry is from a Central Control Panel Matcher interrupt. When this option was not selected, the immediate is in effect and the utility is initiated by CBLM during the high priority MAINT functions. The action in each case is identical and most of the processing is accomplished by routine UTILPROC.

Base-Level Processing

2.13 When MAINT calls CBLM (Fig. 3) to check for pending utility functions, CBLM enters CUTIL at entry point CUTILBAS. When a utility has been successfully requested and marked as active, CUTILBAS processing consists of checking to assure that the proper match conditions exist. When there is a match request pending CUTILBAS does no further processing. When the match option is not active, CUTILBAS passes control to UTILPROC. UTILPROC then initiates the requested utility function.

Match Interrupt Processing

2.14 When the match option has been selected and is active, an interrupt entry to MATCHINT when the panel matchers detect an address or data match. MATCHINT performs necessary interrupt processing and determines that the number of interrupts has not exceeded the allowed limit (three interrupts since last base-level entry). When the limit is exceeded the function is terminated, otherwise a branch is made to UTILPROC to initiate the function.

C. Utility Routine Processing

2.15 UTILPROC selects the correct utility processing routine based on the function code present in the utility request buffer UREQBUF. UTILPROC is divided into three major divisions:

(a) Monitor and Load Functions: MONMERGE and MONREGR

(b) Dump Functions: DMPLSTR and DMPSTR

(c) Application Utility Functions: AUTI1

Initial processing relocates the contents of UREQBUF to specified general registers and positions the utility address for use by the major divisions.

Control is then passed to the appropriate major division.

2.16 The monitor store (Fig. 4) is the basic monitor/load function, all others are made to resemble it and then merge with it at entry point MONMERGE. MONMERGE retrieves registers stored by the interrupt and establishes a scratch area. The data in the scratch area is used to form the store address of the monitor words. The routine checks to assure that the address is to a valid store location and that the highest address to be monitored actually contains data. The first word to be monitored, or loaded, is then fetched and positioned for output. At this point, the processing depends on the function requested (eg, monitor, load, or dump) and what kind of location (eg, store or register). In each case, determination is made between the display buffer (SSP lamps) or TTY.

Monitor Function

2.17 On all monitor functions, a check is performed to determine that this is the first time the words have been monitored. When it is the first monitor, the print conditions are set to output all eight words; when it is not, the words are output only when a change is detected. The lamps option is also checked and when it has been selected the routine LOADDB is called to load the panel data buffer with the first word. Once the words have been printed out or displayed, the single monitor option is checked and when selected the STPUTIL routine will terminate the function. When single was not selected, a return is made to CBLM and the monitor function will continue until terminated by the STP:UTIL message.

Load Function

2.18 On a load request the data to be changed is loaded into the first store word or register location, thus the first monitor will be on the data just fetched from that location and the changed word will not be monitored until the next cycle. Except for this function, the load functions are identical to the monitor functions described in paragraph 2.17.

Dump Function

2.19 The UTILPROC (Fig. 3) routine initiates the proper function which is specified by the function code in the utility request buffer. At this point, the request may be for either a DMP or DMPL.

2.20 Processing of dump functions (Fig. 5) is divided into two parts:

(1) A header is printed out giving the address of the location being dumped.

(2) Four lines of data (32 words).

These five lines are then repeated as often as necessary to complete the requested output.

2.21 The primary difference between a dump store and a dump off-line store is that for an off-line store dump the entire off-line memory is frozen and used as a buffer for the dump instead of the DMPBUF area of on-line store.

2.22 The output of the header is controlled by the DMP function code which causes entry to be at entry point DMPSTR. Once the header has been printed, the function code is changed to DMPL. Changing the function code causes the entry point to be shifted to DMPLSTR. DMPLSTR processes the four lines of data, and when either the last of the four lines has been printed or when all required lines have been printed it resets the function code from DMPL to DMP and the cycle is repeated until all lines have been dumped.

3. COMMON NONRESIDENT UTILITIES

A. General

3.01 The common nonresident utilities provide routines which allow overwrite procedures, used to modify the generic program, and an off-line register dump function which will display register contents of the off-line central control. These functions are stored off-line and called in only as required. The routines are all independent except they are performed as a single multiscan function called OFLUCTL. Initiation is by TTY message. ▶Each routine responds with an appropriate TTY output message:

(a) NG—This message is illegal at this time.

(b) RL—A prior overwrite request is being executed.

(c) PF—The execution of the request has begun. A TTY printout will occur when it completes.

(d) OK—Request successfully performed.

B. Overwrite Function

Message Processing

3.02 The overwrite function (Fig. 6) is initiated by the ALW:OW input message which serves to initialize an overwrite control block for use by the overwrite TTY messages. (ALW:OW must be entered before any other overwrite TTY messages are allowed to be entered.) Entry is at a small resident routine NRPRES which identifies the requested function and causes the appropriate monitors, TTY catalogs, etc, to be loaded from magnetic tape into the resident paging buffer.

3.03. When the paging buffer has been loaded, a bid is made to MSFREQ in CBLM for a multiscan function. When the multiscan function is allowed, return is made to entry point NRUBGN. If not allowed, the request is canceled.

3.04 NRUBGN identifies the requested function as the overwrite request and passes control to the overwrite multiscan function OWMSF which monitors all overwrite functions. When NRUBGN does not locate an active utility, it returns control directly to UTILRTN which cancels the request.

Overwrite Multiscan Function Controller

3.05 The overwrite multiscan function controller OWMSF (Fig. 7) determines whether this request is an abort, a first time entry, or a normal return entry. When an abort is received, the ALW OW INH message is printed and the routine is terminated. For a first entry, the proper control elements are established and the ALW OW COMPL message is printed. On a normal return entry, control is passed immediately to an execute progress mark routine which keeps track of the progress of the utility on each scan and will either return control directly back to CBLM or branch to the location retained by the progress mark to continue processing.

3.06 Once the ALW OW COMPL message has been received, the overwrite function is active and will accept additional overwrite function input messages. These messages are expected in a specific order and are used to control the complete overwrite procedure. The messages will be accepted and processed as long as the overwrite multiscan function controller is active.

C. Input Overwrite Function

Message Processing

3.07 The input overwrite function (Fig. 8) is initiated by either the IN:OW N;TTY, or the IN:OW N;TAPE input message, where "N" is an overwrite number. The TTY option starts a new overwrite entry which allows data to be input via the IN:OWDATA entry. The tape option causes the overwrite number "N" to be transferred from the overwrite file on tape into the overwrite buffer (OWBUF). These operations assume that the generic and issue identifiers have been previously loaded into the generic issue buffer (GENISS) via the IN:GENID and IN:ISSID TTY input messages.

3.08 The input messages are processed by INOW which checks the status of the function, performs necessary conversions and outputs the appropriate TTY messages. When all checks are complete, control is returned to the multiscan function controller (MSFC).

Input Overwrite Processing

3.09 The next entry to the function is started by OWMSF as described in paragraphs 3.05 and 3.06. The controlling progress mark identifies starting location INOW. INOW first calls OPENTCB to open the patch file and start tape unit operations. INOW then determines whether the request is for the TTY or TAPE option. When it is TTY, the generic and issue identification is verified and control is passed to SEARCH. When the option is TAPE, control is passed directly to SEARCH.

3.10 The SEARCH function scans the patch file on tape for overwrite number "N". The overwrite number should be preset when the TAPE option is active and not present for the TTY option. When TAPE is specified and "N" is found INOWFAIL will stop the function and cause the appropriate TTY message, INOW DATA INH, to be printed.

3.11 When correct conditions are established, control passes to RDOWRCRD for tape operations or FMT OWBUFFER for TTY. On a

tape operation, RDOWRCRD copies the tape record containing the specified overwrite number into the overwrite buffer OWBUFFER. On a TTY operation, FMT OWBUFFER formats the OWBUFFER header. The header consists of the header length, the overwrite number, and the current time of day. The tape and TTY operations then merge to INOWPASS which closes the patch file. REQPASS will then cause a completion message to be output and returns control to CBLM. When the TTY option was selected to establish a new overwrite, the new data is entered via the input overwrite data function.

D. Input Overwrite Data Function

Message Processing

3.12 Overwrite data (Fig. 9) is entered after the ALW:OW is successfully completed. The input overwrite data function is initiated by input message IN:OWDATA with parameters which specify the segment number, address, old data, and new data. An overwrite may consist of any number of these messages.

3.13 Entry is at INOWDATA where the OWBUFFER is set up and validity checks are performed. The data is organized in OWBUFFER in blocks according to ascending segment numbers and addresses. The segment number and address thus become the first element in each block.

Input Data Processing

3.14 A location must be established for each block to determine if it is to be added to the end of the blocks already in the buffer or inserted between existing blocks. These functions are performed by TESTEOB. When the block is to be added, control passes to INSERTDATA. INSERTDATA determines overflow conditions and if the new data will cause an overflow it causes the INOWDATA INH message to be output and stops the function. When the data is to be inserted, the function CONTSRCH will find the proper location and pass the information to INSERTDATA. INSERTDATA tests for overflow as described previously. In both cases, INSERTDATA calls MONST to perform the proper word adjustments in OWBUFFER. After this is accomplished, INSERTDATA compares and stores each block into the proper location in OWBUFFER. When all blocks have been stored, the overwrite data is ready for loading into the proper store locations and verified. The actual loading is accomplished by the load overwrite function.

E. Load Overwrite Function

Message Processing

3.15 The purpose of this function (Fig. 10) is to load or verify overwrite data. The function is initiated by either LOD:OW or VFY:OW input message. Either old or new data can be selected as an option. Processing is almost identical in each case. Each location specified in OWBUFFER is accessed. When the location contains resident data, the on-line store location is accessed. For nonresident data locations, the nonresident program file on tape is accessed. During a verify operation, the data in accessed locations is compared to the data for that location recorded in OWBUFFER. OWBUFFER contains both old and new data words and the message option determines which data words will be compared. A load request causes the appropriate words to be written from the OWBUFFER into the specified store locations.

Load or Verify Resident Data

3.16 The function is entered at entry point LODOW. The processing is accomplished by two loop operations. The first or outer loop cycles through and determines if the data words contained in the ORGBLKs are resident or nonresident and tests for the end of OWBUFFER. When modified data is found, it is written on to the appropriate tape block overwriting the old data after which a completion message OW COMPL is output. When the data is resident, the ORGBLKs are processed in sequence by the second loop (load or verify) starting at entry point VFYRESLP. Each ORGBLK is checked to determine load or verify status and the load or verify is accomplished as required. During a verify operation, the first noncompare will cause the output message ALW OW INH to be printed out and the function will be terminated. When all the words in an ORGBLK have been processed, a return is made to the outerloop to check the next ORGBLK which will be processed in the same manner until the end of OWBUFFER is reached.

Load or Verify Nonresident Data

3.17 When the data is nonresident, the tape must be searched. The tape is formatted into physical blocks which do not correspond to the ORGBLKs. Processing is started by the common tape handler program (CTAPH) which opens the tape operations. By furnishing the segment number specified in each ORGBLK, the appropriate tape record can be located and read into the input-output buffer (I_OBUF). The processing is then similar to that performed on resident data. The outer loop loops through I_OBUF and checks each address specified in the ORGBLK against those in I_OBUF. When an affected address is found, the inner loop performs either a load or compare as specified. The first noncompare will return the ALW OW INH message as before. The loop continues until all words in and ORGBLK are completed or until the end of I_OBUF is reached. The outer loop is then entered and either sets up the next ORGBLK or reads in the next tape block. During this processing any modified data is overwritten into the proper tape block. The loops are repeated until the end of OWBUFFER is reached at which time the completion message is outputted and the function terminated.

F. Initialize, Output, or Update Overwrite File

Message Processing

3.18 The purpose of this function (Fig. 11) is to initialize, update, or output the patch file. The function is initiated by any of the following messages: OP:OWFILE, INIT:OWFILE, or UPD:OWFILE. Processing is similar in each case.

3.19 Initiate overwrite function must be entered before any other overwrite actions can be taken. On initiation the tape functions of CTAPH locate the patch file. A header is then formatted and written onto the tape, and any overwrite already in the file wiped out. When this is complete, OPOWFILEPASS closes the patch file and generates a completion message. Any failure will be processed at OPOWFILEFAIL where the patch file is closed and an error message is generated. The function is then terminated. The updata overwrite function is identical except that existing overwrites in the file are not wiped out.

3.20 The output function OPOWFIL starts with the tape actions by CTAPH, which locates the patch file. A census is then taken of the file. This consists of a header line, two lines for the generic and issue identifiers and a line for each overwrite in the patch file. Successful completion causes a branch OPOWFILECOMPL which outputs the final link after this OPOWFILEPASS closes the patch file and generates the completion message. A failure results in a call to OPOWFILEFAIL where error processing is performed.

Update Overwrite Function

3.21 The purpose of the update overwrite function (Fig. 12) is to update the checksum file on tape to reflect the overwrite currently residing in OWBUFFER. On planned changes, the affective checksum words must be changed on tape and in resident store. The checksums are calculated over each 4K block of resident store.

3.22 After locating the file on tape, the size of the OWBUFFER and the length of the ORGBLKs it contains are determined. CHKRSEG then locates the appropriate 4K block and sets up for the checksum. Two loops are then set up. The outer loop checks for the completion of the OWBUFFER and/or store segment. The inner loop starts at CSADRLP which keeps tract of the 4K boundaries. CSORGLD sets up the length of the ORGBLK. Checksums are calculated for each ORGBLK in each 4K segment. The checksums are calculated by exclusive ORing the data word and its checksum word. The results are then written back onto tape into the appropriate checksum word.

Output Overwrite Function

3.23 This function (Fig. 13) will either print the contents of OWBUFFER on the TTY, or append the overwrite in OWBUFFER to the end of the patch file on tape. The two messages which can initiate this function are: OP:OW;TTY or OP:OW;TAPE.

3.24 The tape option will cause the patch file to be opened and then searched for an overwrite of the same number as the one currently in OWBUFFER. When the number is found, OPOWFAIL will fail the function. When the number is not found, the tape is positioned to the end of the patch file and WRTOWRCRD adds the record from OWBUFFER to the end of the file.

3.25 The TTY option requires location of the proper overwrite number in OWBUFFER. The overwrite data is then printed on the TTY as a linked message with a loop which cycles all ORGBLKs (see INOWDATA for ORGBLK explanation). When all the data has been printed out, the function is terminated.

Activate, Remove, or Cancel Overwrite Function

3.26 The purpose of this function (Fig. 14) is to change the status of an overwrite. It can be activated, ACT:OW; canceled, CNL:OW; or removed, RMV:OW. The status of the action is recorded in the overwrite record header.

3.27 The tape is activated and the path file is searched by SEARCH for the current overwrite number. When it is not found, the request is failed by CHGOWFAIL. When the tape record is read into the I_OBUF; the status is checked for cancel, remove, or activate already in affect. When the old status is the same as the requested status, no further action is taken. When the status is different, the new status is inserted and the modified block is written back onto the overwrite file on tape.

G. Load Off-Line Main Store Function

3.28 The purpose of this function (Fig. 15) is to load a new generic and/or translation file into the off-line main store. It is initiated by the LOD:OMAS input message and the generic identification and whether or not a translation file is to be loaded is specified in the message parameters. A patch file will not be loaded and the system must be in the manual state.

3.29 Input message processing is accomplished by LODOMAS, which identifies the request and sets up the control buffer with the parameters specified by the message. When message processing is complete, an in-progress message is printed on the TTY.

3.30 The load function is started by a multiscan function entry from MSFC. Multiscan entry processing is identical to that previously described (paragraphs 3.02 through 3.04) with COPYGENMON selected as the monitor function. COPYGENMON controls processing by sequentially advancing through a number of routines which initiate various functions (eg, tape unit control delay times, error checks, etc).

3.31 Initial processing determines whether or not a translation file load is desired. When it is requested, tape activities are started to locate the proper file location. The generic file on tape is opened and INITOST is invoked to disable the write-protect controls. Another substate of COPYGENMON then clears the off-line store and sets up the tape unit for a read activity.

3.32 DMPMON initiates the read which reads the generic a block at a time from the tape into the input/output (I/O) buffer. When the tape completes the data read into the buffer, the buffer is unpacked to format the words into the proper length for the move to the main store.

3.33 COPYGENMON then sets up to copy the data from the I/O buffer to the proper store locations. A maximum of 297 words is moved at one time and the loop continues until all words have been copied. As each block is copied, it is read back and compared. Any noncompare will stop the function and print the LOD OMAS ERR message. This activity continues a block at a time until the file is complete. When the file is complete tape activities are halted, the LOD OMAS COMPL message is printed and the function is terminated.

H. Off-Line Register Dump Function

3.34 The purpose of the off-line register dump function (Fig. 16) is to dump the register contents of the off-line processor and print them out on the TTY. The function is initiated by the DMP:REGOFL input message. Input message processing is identical with that described for the overwrite entry functions (paragraphs 3.02 through 3.04) except that the register dump monitor ENTRDPRG is selected.

3.35 ENTRDPRG checks for any aborts and when appropriate the DMP REG OFL INH message is printed and the function terminated. The scan status is checked on each entry. On first scan, the off-line processor is placed in the maintenance stop state and conditions are set up to gate the registers to the maintenance channel. GTNXT 16 then gates the contents of all 16-bit registers onto the maintenance channel and they are transmitted to the REGBUF scratch area of the on-line processor.

GTNXT 20 will then perform the same function on the 20-bit registers. The TTY header message DMP REG OFL is then formatted and printed out.

3.36 On subsequent scans, the REGBUF scratch area is accessed via a loop and the register contents stored there are output as a linked message to the TTY. When the last link is printed, control is returned to MSFC.

I. ▶Simulated Bootstrap

3.37 A method is available to load off-line data via a simulated bootstrap. This is accomplished on the off-line 3A CC by the input command:

LOD:OMAS;BOOT[FULL]!

In addition to loading the generic and translation files, the patch file is also loaded. This method has the additional feature of verifying that the bootstrap was successful for the cartridge tape specified. A reload based on checksums or a complete reload (FULL specified in the input command) of memory is possible depending upon the input command. The bootstrapping process will attempt to use both tapes for the loading process just as it does during an on-line bootstrap, and to load only from the off-line tape, the on-line tape must be removed from the cartridge tape transport.◀

4. GLOSSARY

4.01 The following terms and acronyms are contained in this section.

Assembly Unit—A collection of code that is assembled or compiled as one entity. The assembly unit is the highest level of a modular program structure and may or may not contain functionally related subunits.

Base Level Loop—A major ESS software loop which normally includes all functions not performed at interrupt level.

CBLM—Common base level monitor.

Checksum—An error checking code wherein a series of check digits are computed based on the modulo (no carries) summing of the data digits. The value of the check digits are computed so that the modulo sum of the data digits plus the check digit is equal to zero.

Entry Point—A labeled location at which a CSECT or any logical block of code may be entered.

EOS—Extended Operating System.

ESS—Electronic Switching System.

I/O—Input/output.

MSFC—Multiscan Function Controller.

3A CC—3A Central Control.

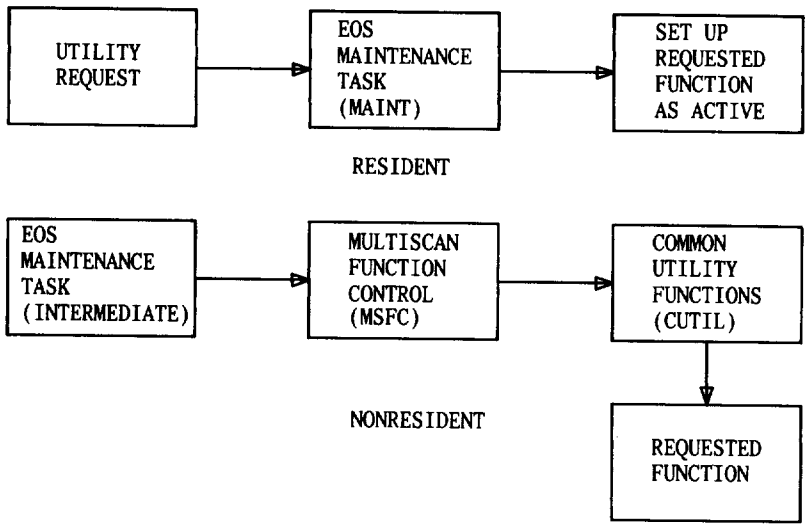


Fig. 1—Utility Functional Flow

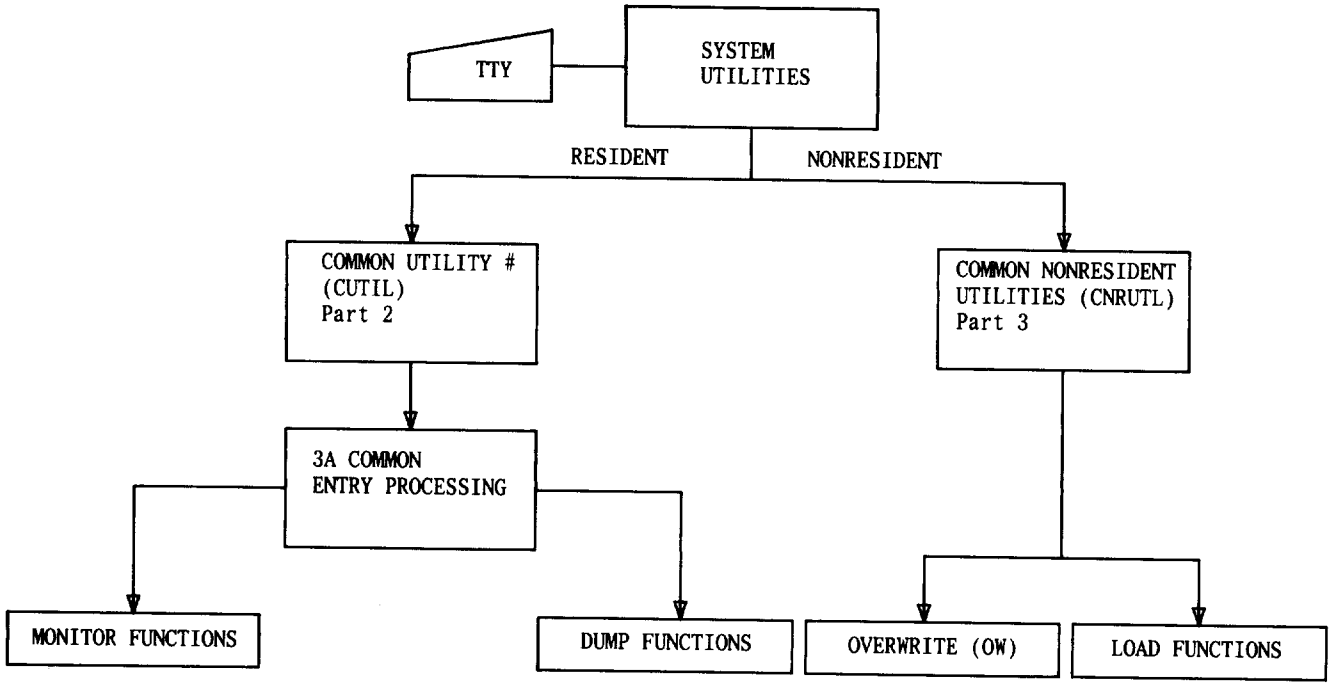
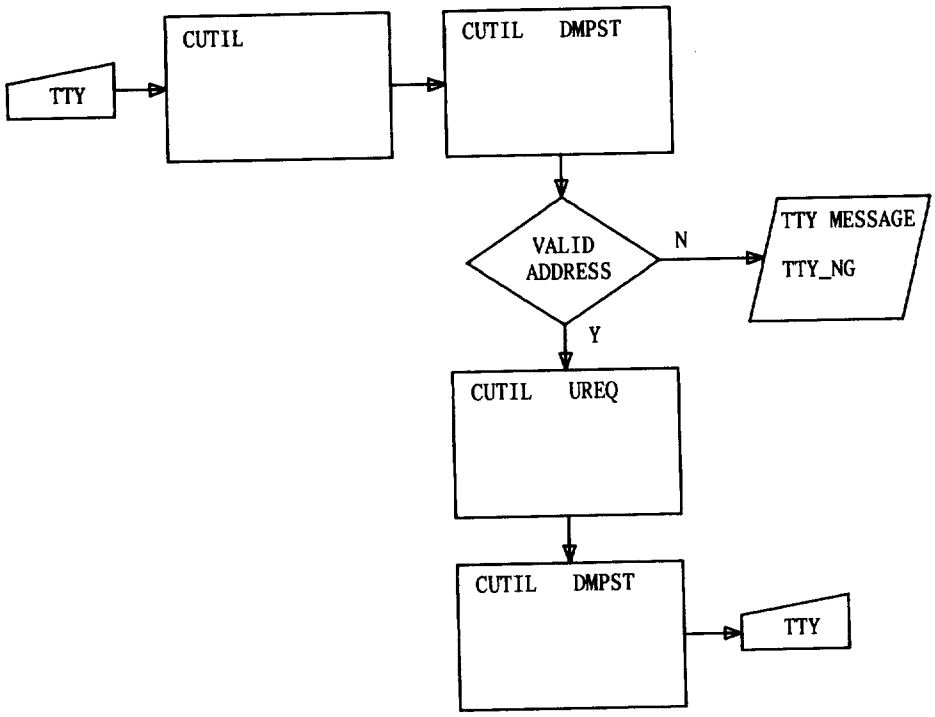


Fig. 2—System Utility Structure



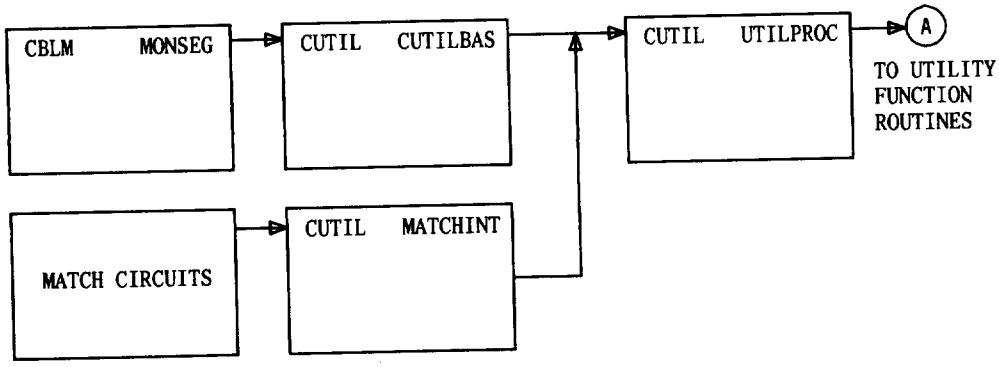


Fig. 3—Utility Common Processing

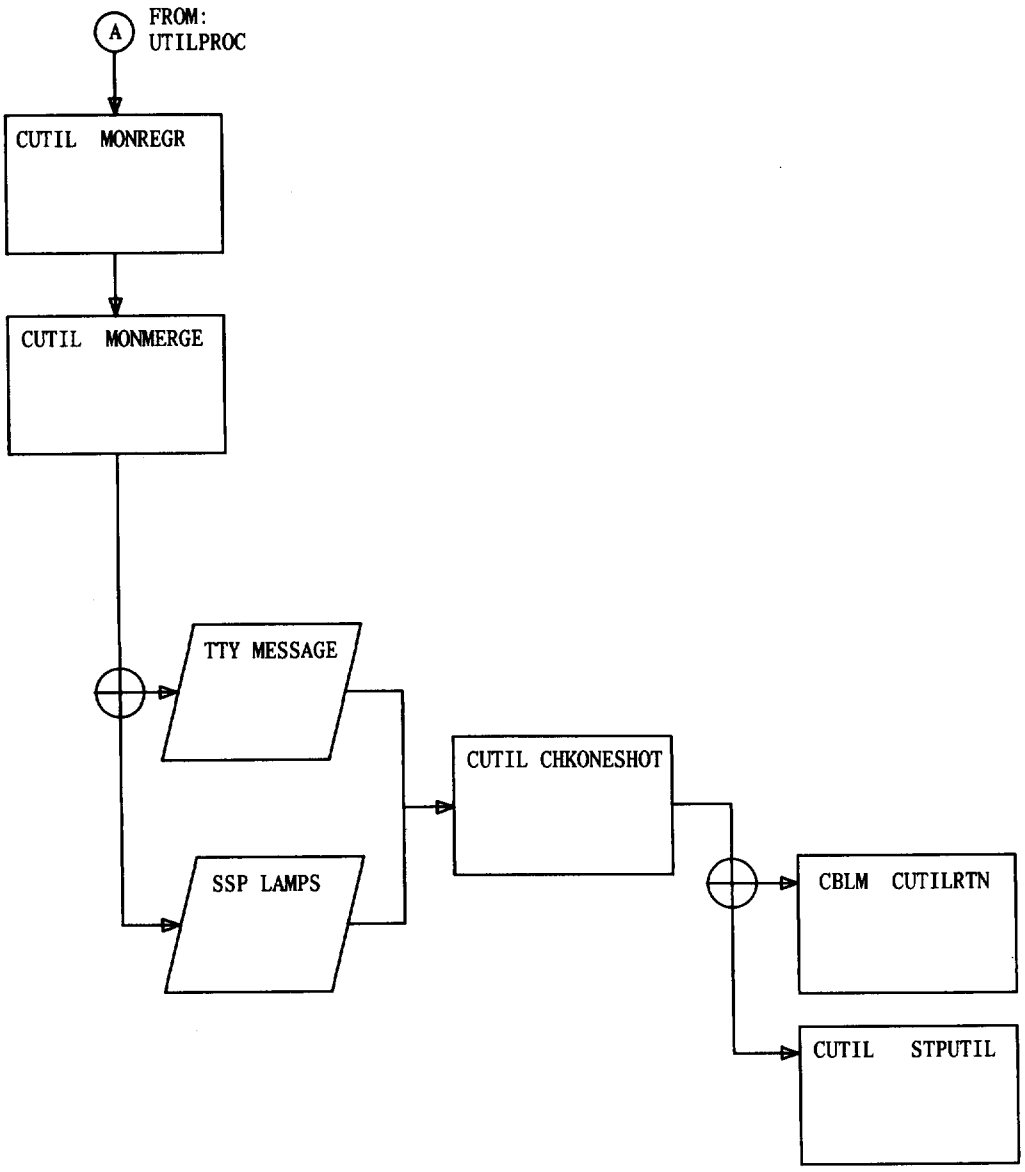


Fig. 4—Monitor and Load Functions

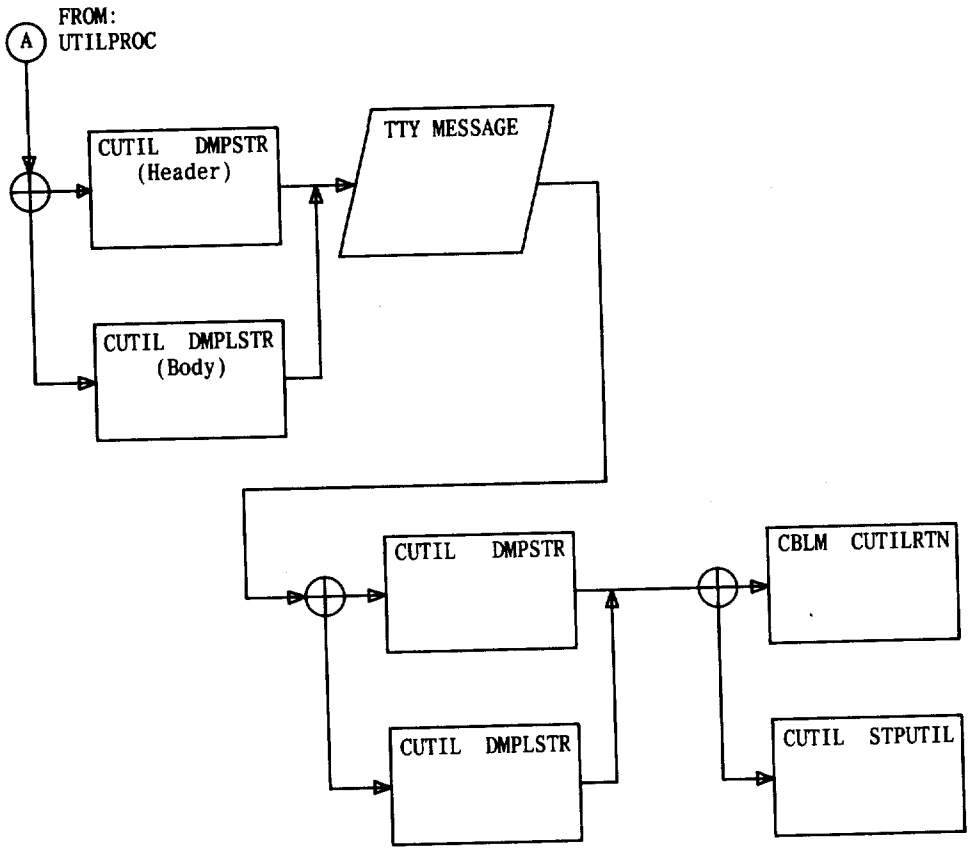


Fig. 5—Dump Function Processing

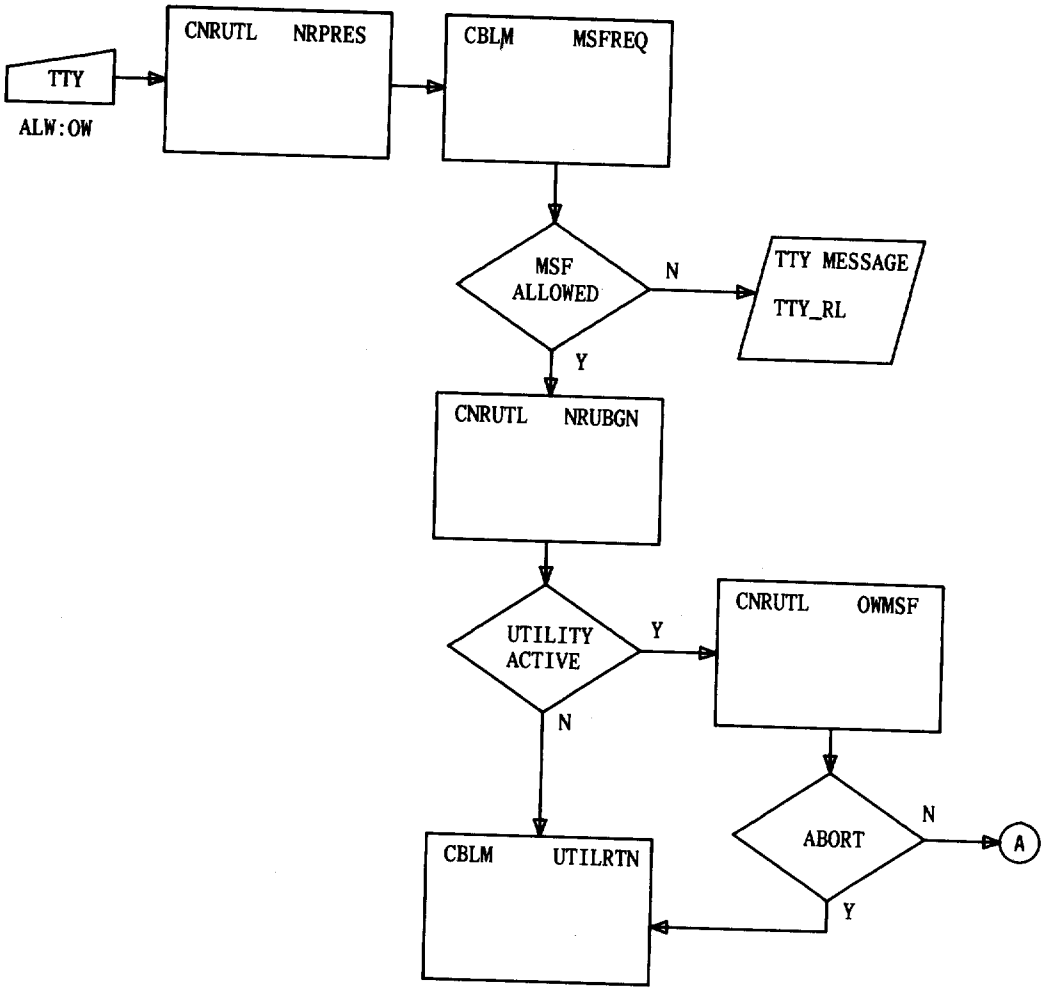


Fig. 6—Nonresident Utility Overwrite Function

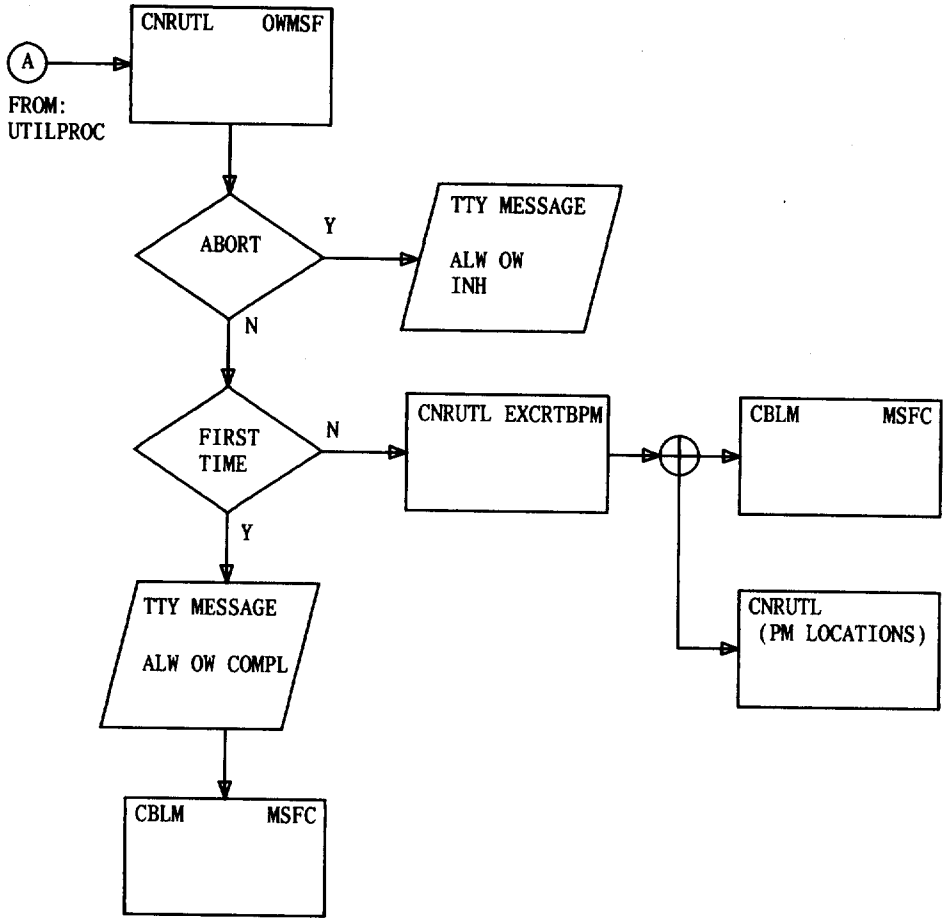


Fig. 7—Multiscan Function Processing

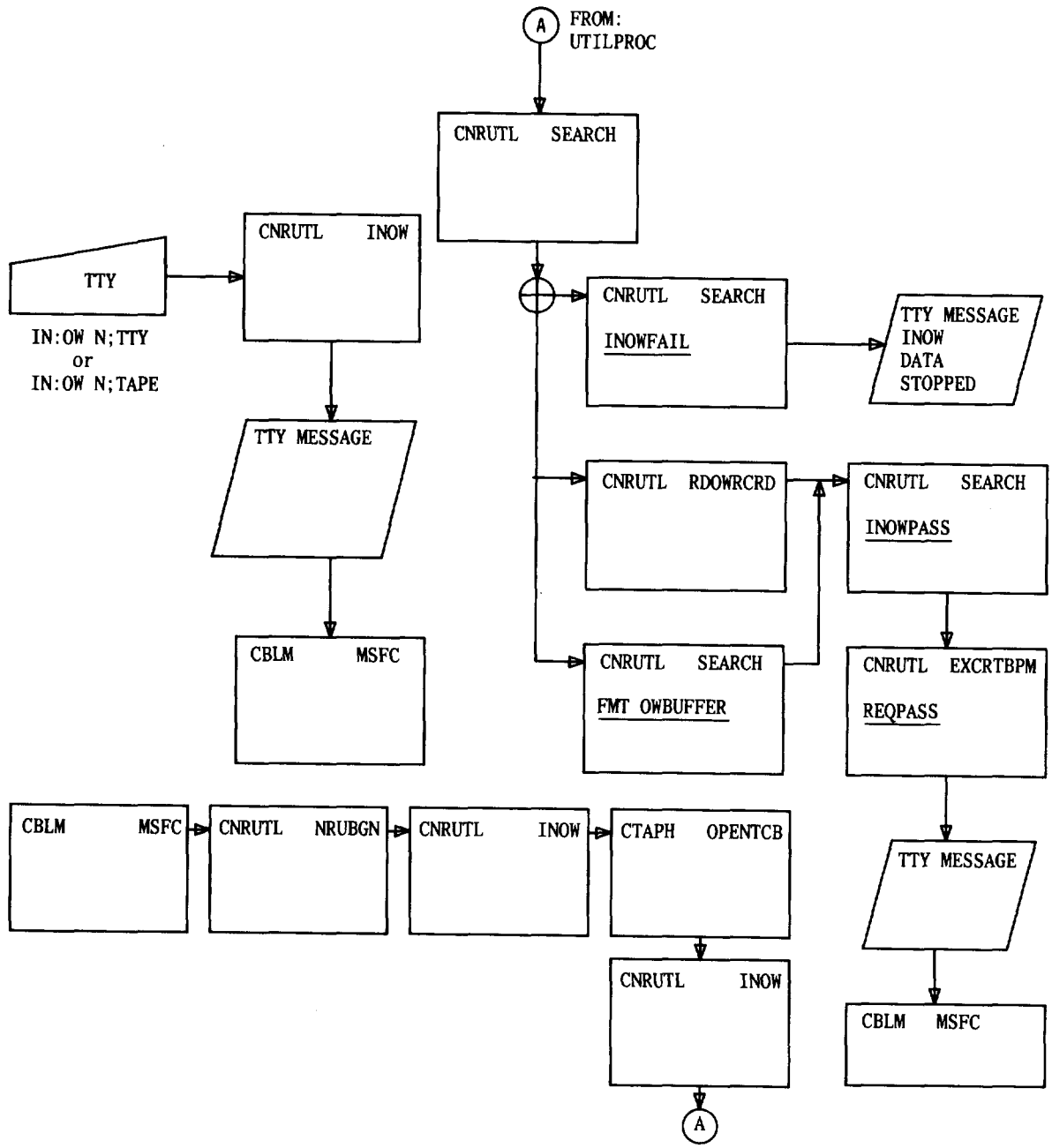


Fig. 8—Input Overwrite Function

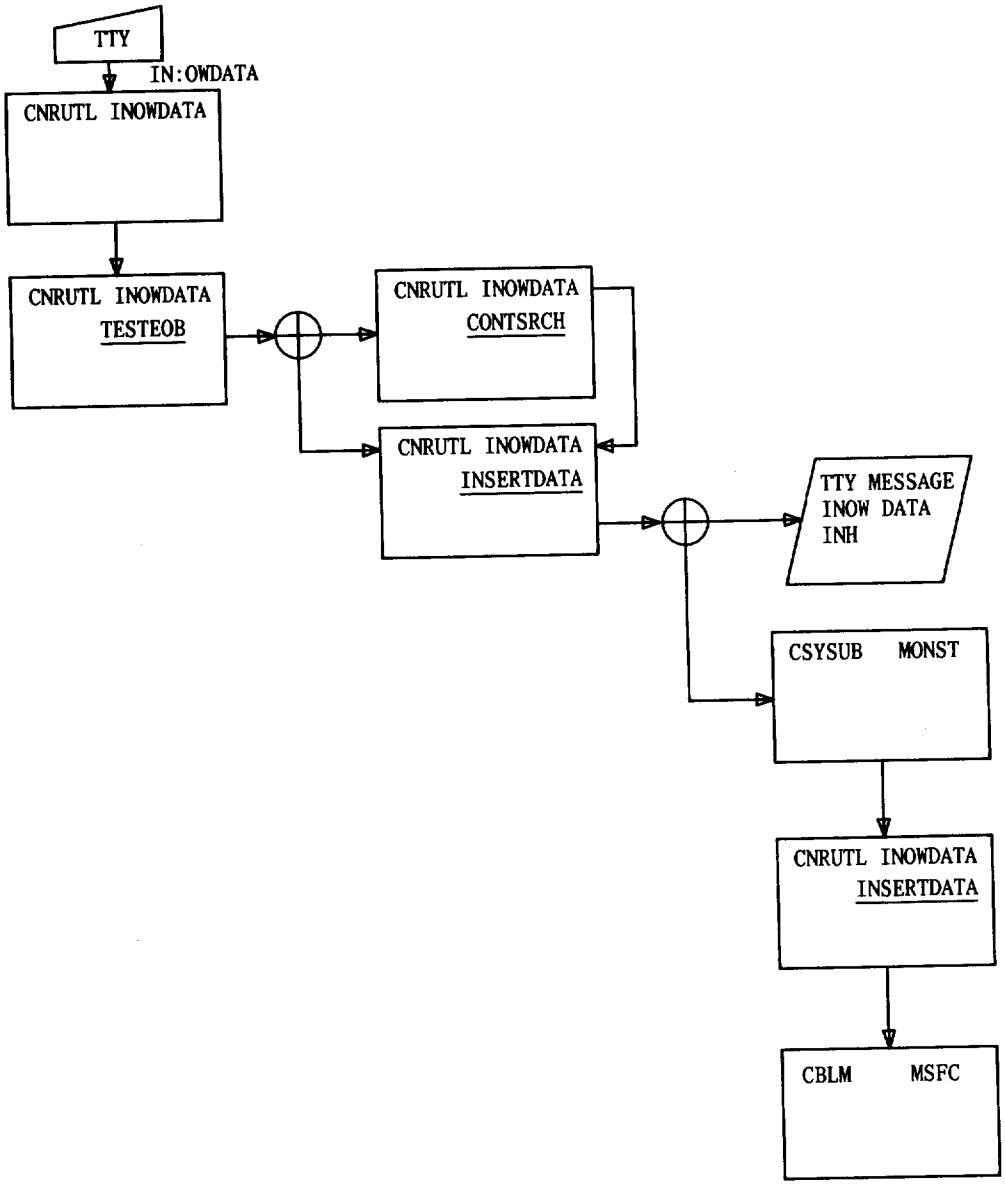


Fig. 9—Input Overwrite Data Function

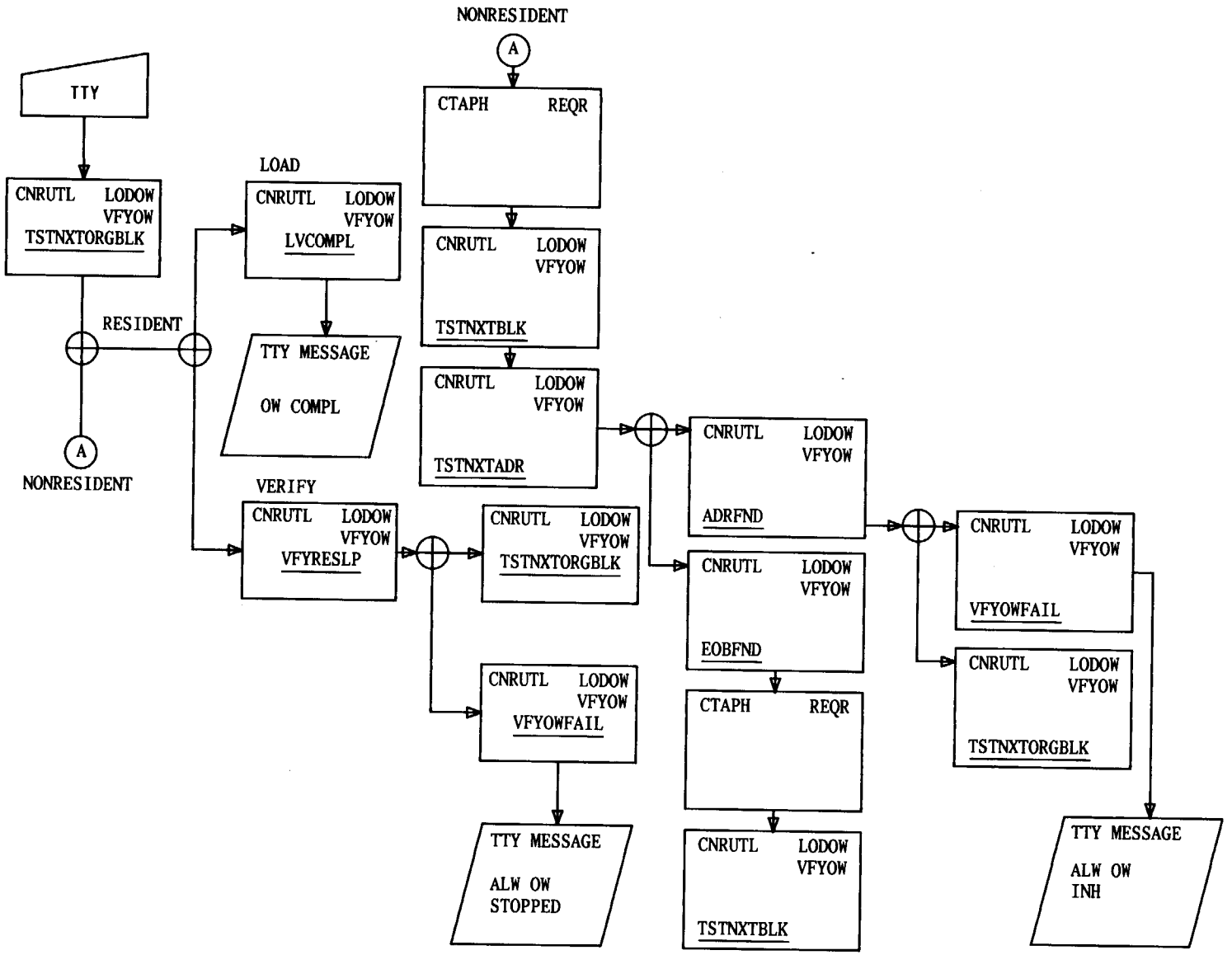


Fig. 10—Load or Verify Overwrite Function

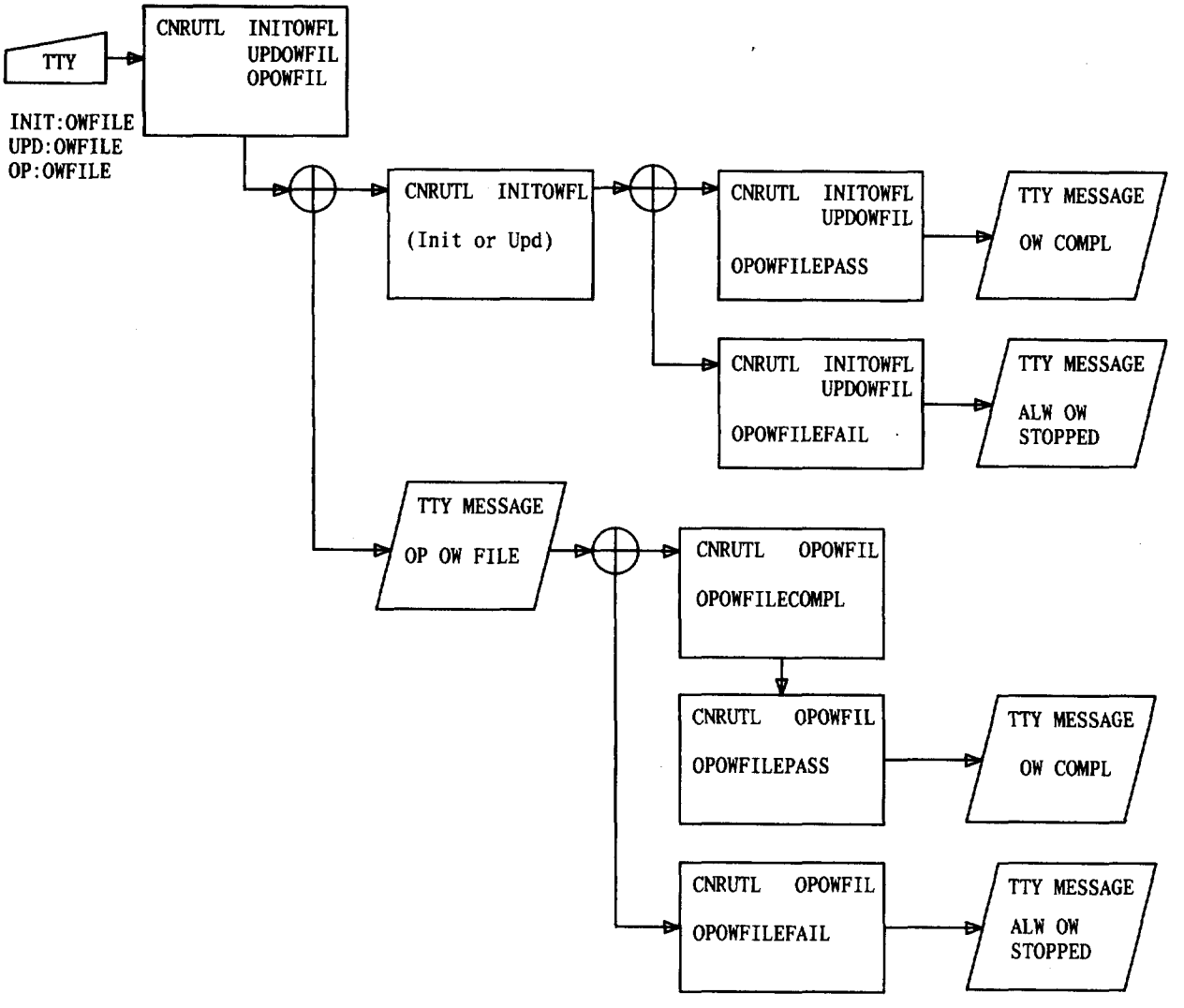


Fig. 11—Initialize, Output, or Update Overwrite File

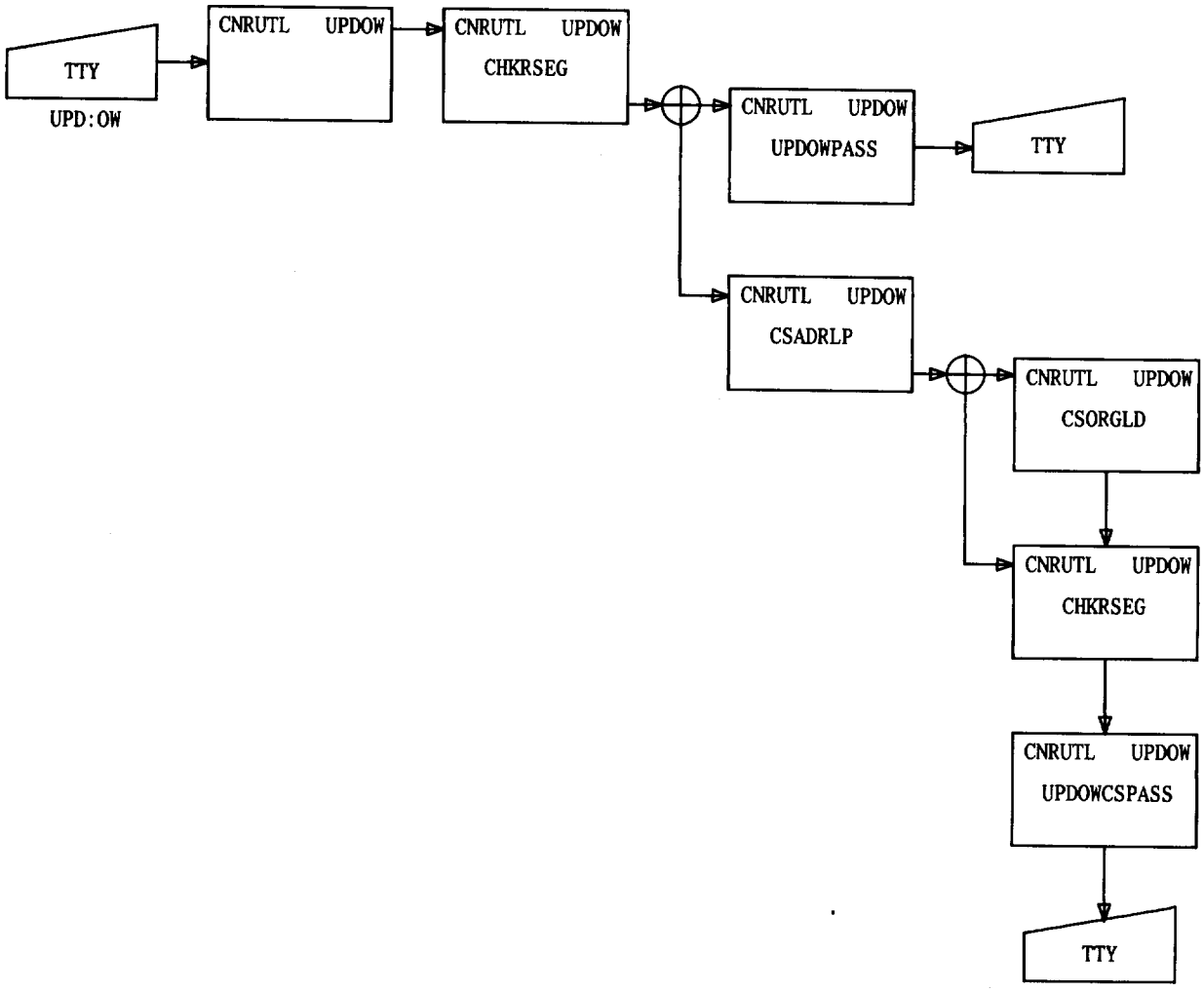


Fig. 12—Update Overwrite Function

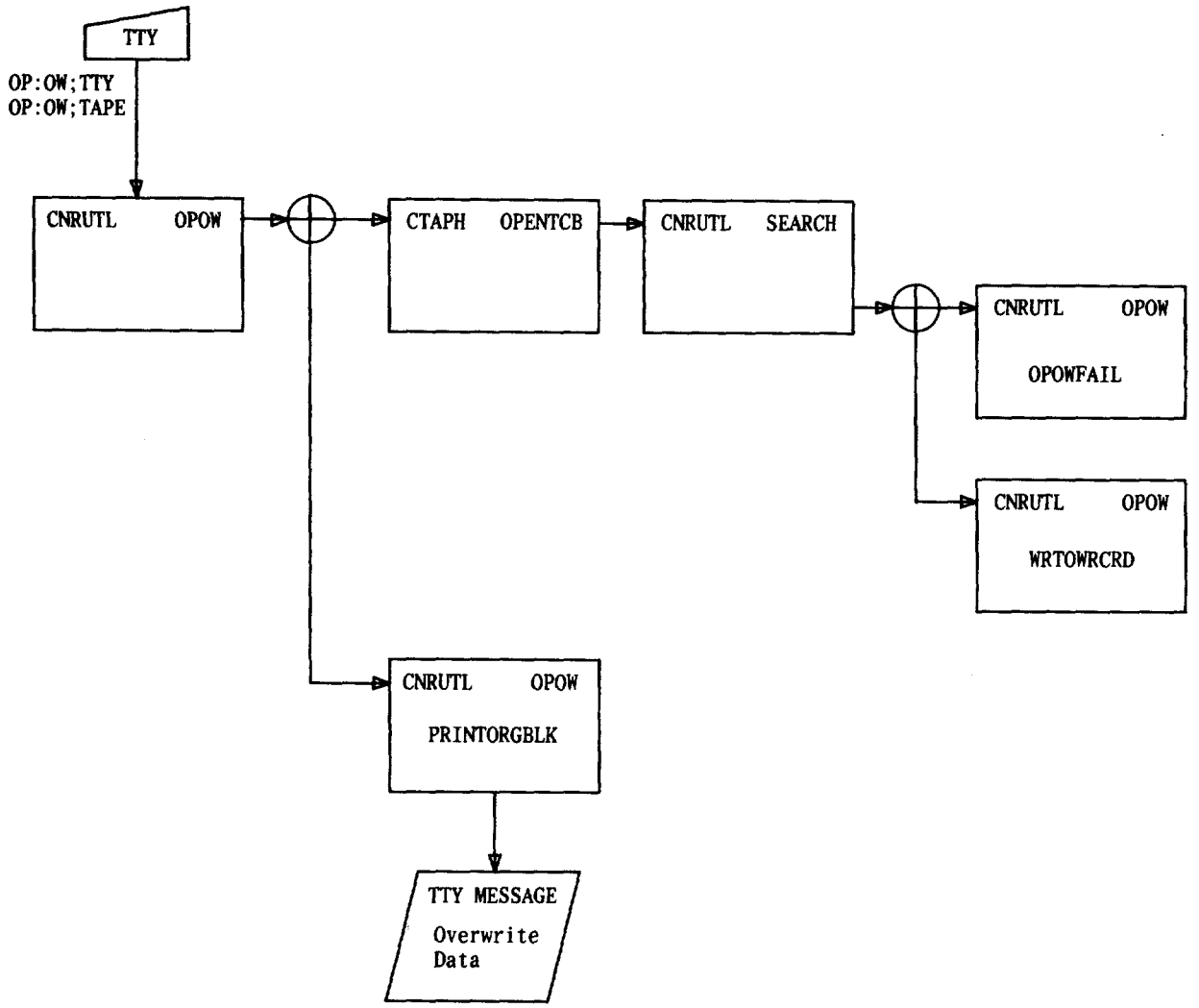


Fig. 13—Output Overwrite Function

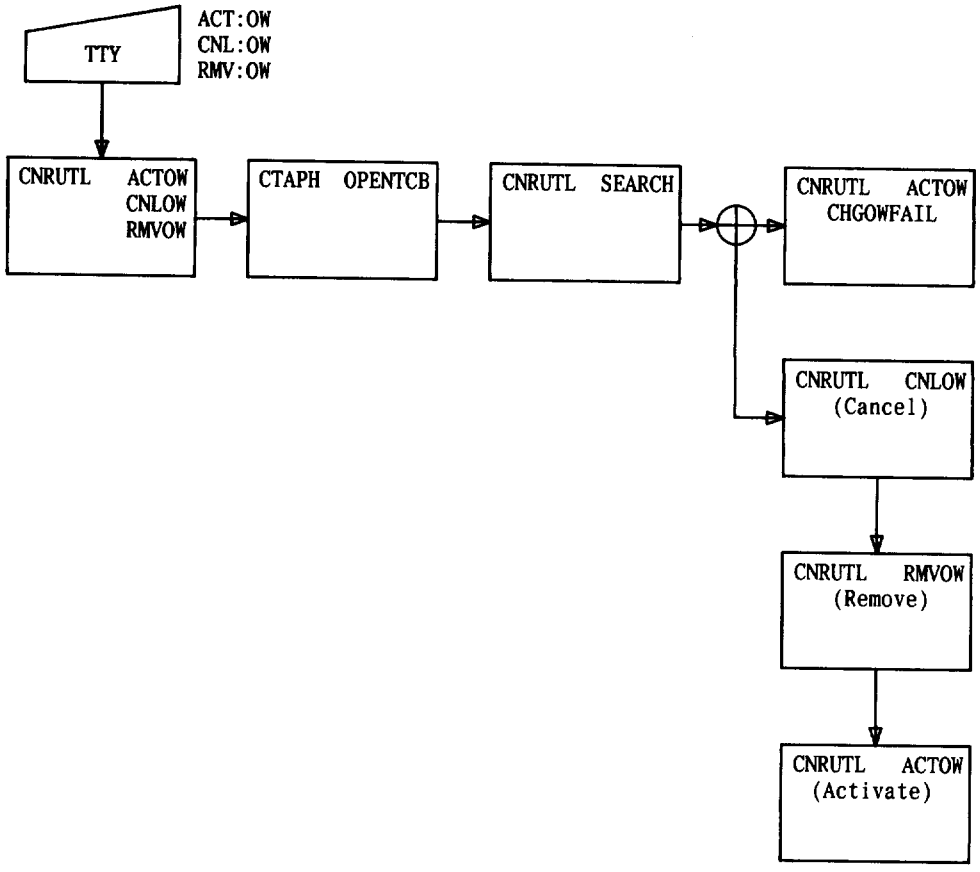


Fig. 14—Activate, Cancel, or Remove Overwrite

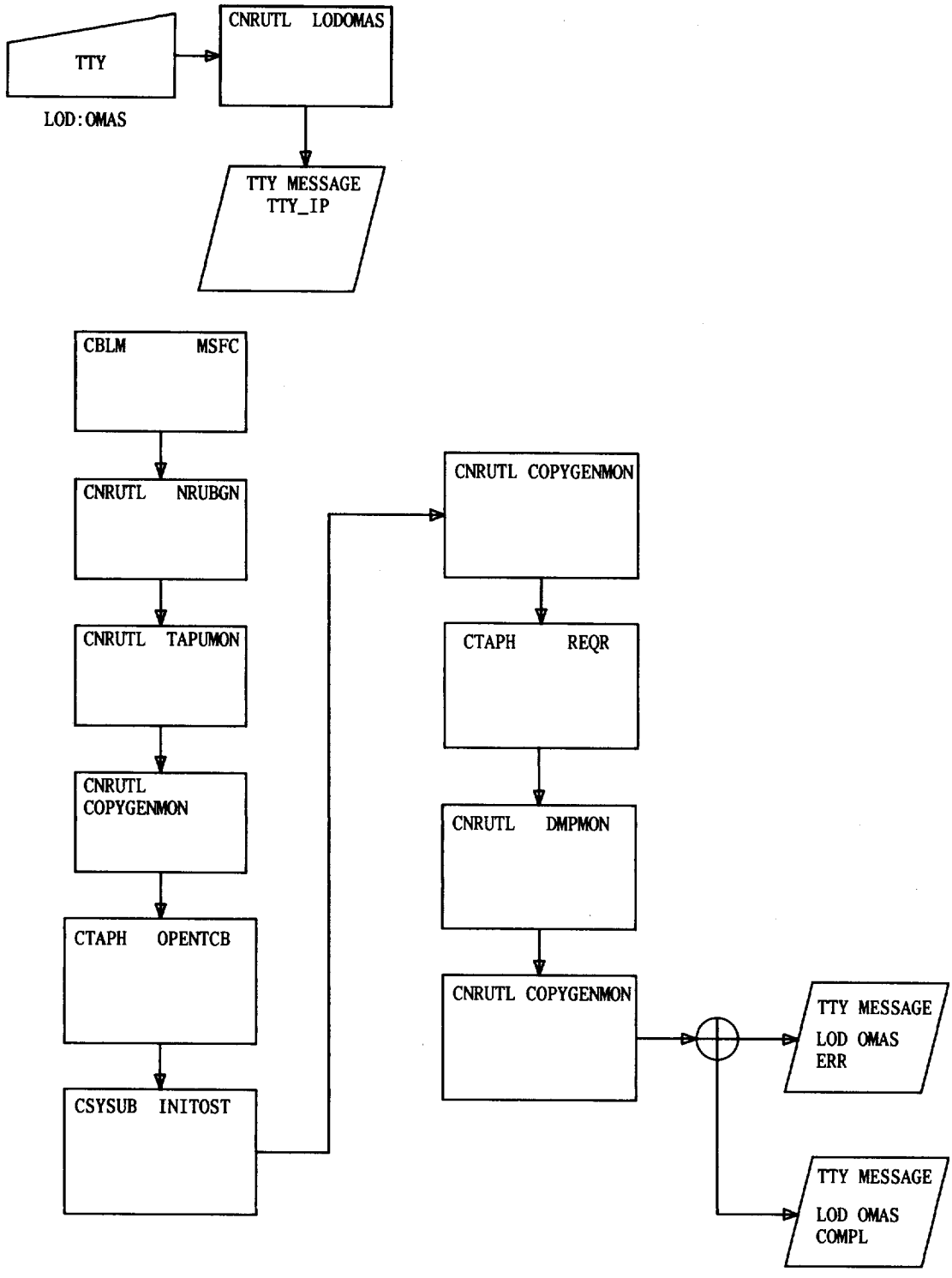


Fig. 15—Load Off-Line Main Store

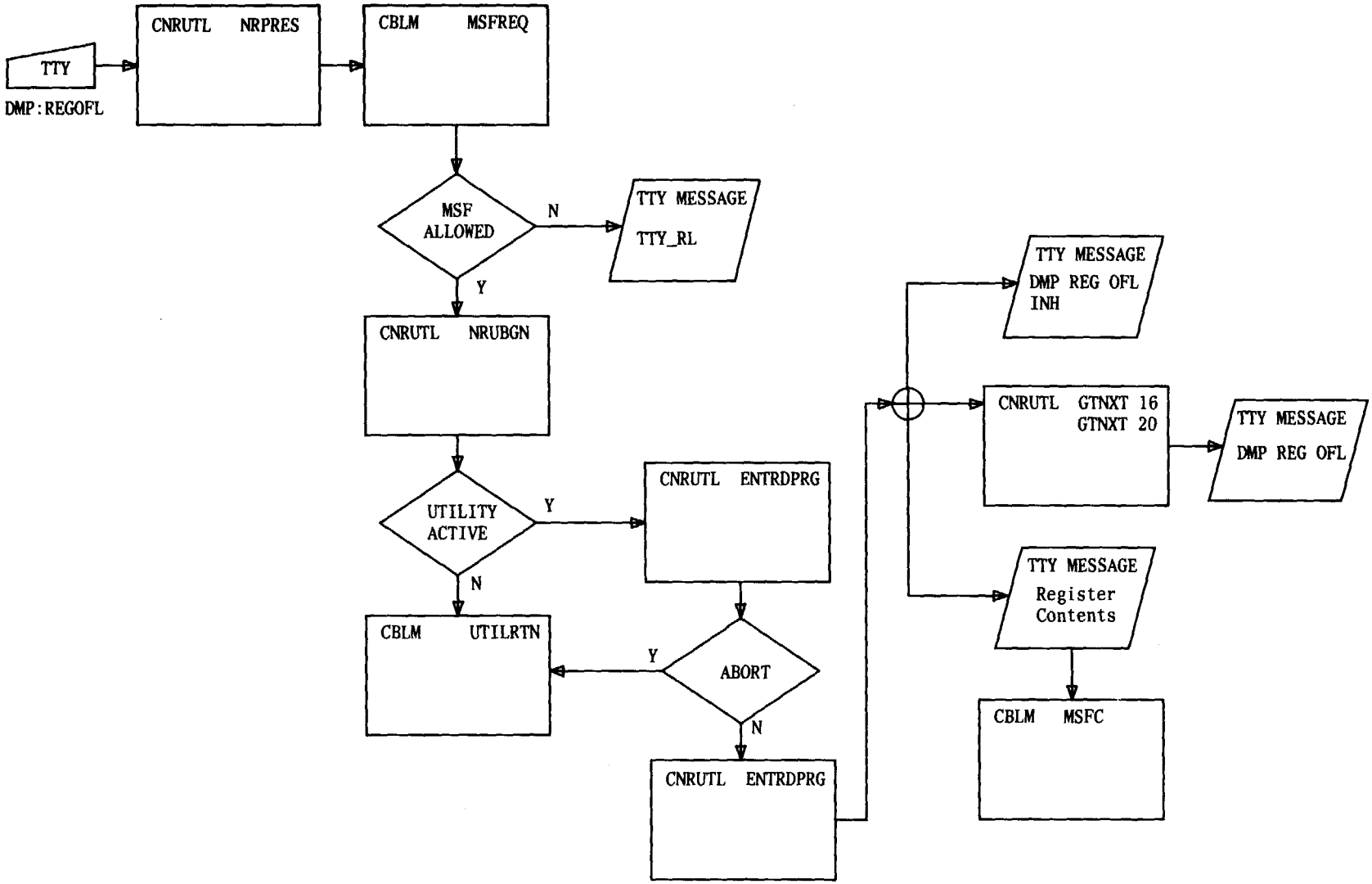


Fig. 16—Off-Line Register Dump

TABLE A

TTY FUNCTION ENTRY POINTS

TTY FUNCTION	ENTRY POINT
LOD:ST	LODST
MON:ST	MONST
LOD:REG	LODREG
MON:REG	MONREG
LOD:INDIR	LODINDIR
MON:INDIR	MONINDIR
DMP:ST	DMPST
DMP:OFLST	DMPOFLST
DMP:INITQ	DMPINITQ
SET:MATCH	SETMATCH
SET:INDIR	SETINDIR
STOP:UTIL	STPUTIL