FTL REPORT R82-1

COMPUTER PROGRAM DEVELOPMENT SPECIFICATION FOR THE AIR TRAFFIC CONTROL SUBSYSTEM OF THE MAN-VEHICLE SYSTEMS RESEARCH FACILITY

by the

Flight Transportation Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts 02139

January 1982

T A B L E O F C O N T E N T S

T A B L E O F C O N T E N T S (Continue)

T A B L E O F C O N T E N T S (Continue)

T A B L E O F C O N T E N T S (Continue)

T A B L E O F C O N T E N T S (Continue)

1.0 SCOPE

1.1 Identification

This specification establishes the requirements for performance, design, test, and qualification of a computer program identified as ATCSIM Release 0.

1.2 Functional Summary

The Air Traffic Control (ATC) Subsystem of the Man-Vehicle System Research Facility (MVSRF) is a hardware/software complex which provides the MVSRF with the capability of simulating the multi-aircraft, multi-controller Air Traffic Control environment required to perform full-mission flight simulations. The ATCSIM CPCI is the software component of this complex.

The ATCSIM operates in three modes:

1) Standalone, that is, without the required participation of the rest of the MVSRF.

2) Single-cab mode, with either the conventional (727) or advanced cab.

3) Dual-cab mode, with both simulated cabs.

The ATCSIM is capable of simulating up to three simultaneous ATC radar positions and up to 40 aircraft (called "pseudo-aircraft" to differentiate them from the cab aircraft(s)). The ATC subsystem performs the following functions:

1) Simulates the movements of up to 40 pseudo-aircraft.

2) Simulates the navigational and surveillance equipment in the area.

3) When in the single or dual cab mode, it communicates with the host computer and the cab computer(s) to receive data concerning the cab positions as well as directives regarding the management and control of the ATC environment.

4) Drives up to three simulated ATC positions, each equipped with a calligraphic display scope and a keyboard. The human operator of each ATC position (called the air traffic controller) can monitor the aircraft movements on the calligraphic display and can control them through clearances communicated to the pseudo-pilots (see next item) via audio channels provided for this purpose. The keyboard can be used to enter aircraft clearances and to modify the information displayed on the radar screen.

5) Drives up to three pseudo-pilot positions, each equipped with a video terminal and a keyboard. The operator of each pseudo-pilot position (called the pseudo-pilot) will at any point be responsible for "piloting" up to 12 aircraft. He is responsible for responding to clearances received from the air traffic controller regarding any aircraft under his control. To accomplish this task, the pseudo-pilot will have complete status information on all the aircraft under his control displayed on the video screen at all times. To initiate the execution of the clearance received by the air traffic controller, a repertoire of piloting commands will be available to him. Those will have to be entered on the keyboard along with appropriate identification of the target aircraft.

2.0 APPLICABLE DOCUMENTS

1) MVSRF: ATC Subsystem Preliminary Design Document, FTL Report Rxx-x, Flight Transportation Laboratory, M.I.T., Cambridge, Ma, October 1981.

2) MVSRF ATC Subsystem: SIMCON Process Preliminary Design Document, FTL Report Rxx-x, Flight Transportation Laboratory, M.I.T., Cambridge, Ma, January 1982.

3) MVSRF ATC Subsystem: POSGEN Process Preliminary Design Document, FTL Report Rxx-x, Flight Transportation Laboratory, M.I.T., Cambridge, Ma, January 1982.

4) MVSRF ATC Subsystem: SECTOR Process Preliminary Design Document, FTL Report Rxx-x, Flight Transportation Laboratory, M.I.T., Cambridge, Ma, January 1982.

5) MVSRF ATC Subsystem: PPILOT Process Preliminary Design Document, FTL Report Rxx-x, Flight Transportation Laboratory, M.I.T., Cambridge, Ma, January 1982.

6) MVSRF: Host computer - ATC subsystem Interface Control Document, (in preparation).

7) MVSRF: ATC subsystem - Audio subsystem Interface Control Document (in preparation).

8) Heinz, V. Stochastic Simulation of Terminal Area Airspace, S.M. Thesis, M.I.T. Department of Aeronautics and Astronautics, Cambridge, Ma, September 1976.

9) Hoffman, W.C., Hollister, W.M., and Simpson, R.W., Functional Error Analysis for ATC Concepts Evaluation, DOT-TSC-212-72-1, Aerospace Systems Inc., Burlington, MA., May 1972.

3.0 REQUIREMENTS

The ATCSIM CPCI consists of a number Computer Program Components (CPC's), which execute in a concurrent fashion in a VAX-11/750 virtual memory single processor computer. These CPC's rely on the standard VMS Operating System supplied by the computer manufacturer for dispatching and communications. In VMS terminology, each CPC that competes for resources concurrently with other CPC's is termed a "Process"; thus, each ATCSIM CPC is a VMS Process.

The ATCSIM CPCI is made up of four types of CPC's:

1) The SIMCON process. This CPC performs initialization and overall simulation coordination functions, such as timing and host computer I/O. It is the first to be activated (from the ATC command console). Once activated, and given minimal information about the configuration of the system (e.g. how many radar controllers, how many pseudo-pilots, standalone, one-cab or two-cab configuration), this process initiates all other processes, and accepts subsequent command lines from either the ATC console or the main host processor. Reference 2.2 contains the Development Specifications for this CPC.

2) The POSGEN process. This CPC is responsible for the pseudo-aircraft, from creation to deletion, including flight maneuvers. It maintains the true, onboard estimated, and ground (radar) estimated position of these aircraft. Reference 2.3 contains the Development Specifications for this CPC.

3) The SECTOR process. This CPC is responsible for maintaining the simulated radar display, including processing the keystrokes that the air traffic controller may type at the attached keyboards, with immediate servicing of "local" requests (such as changing the display scale), and routing of the rest to the CPC that may satisfy them (such as sending a handoff message to another SECTOR process). There will be one such CPC per radar display position desired, i.e up to three can be active at any given time. Reference 2.4 contains the Development Specifications for this CPC.

4) The PPILOT process. This CPC is responsible for communications between the POSGEN process and the simulation actors which "control" the

pseudo-aircraft (pseudo-pilots). This includes displaying pseudo-aircraft data and messages from other parts of the CPCI, as well as accepting pseudo-pilot commands. There will be one such CPC per pseudo-pilot position, i.e. up to three can be active at any given time. Reference 2.5 contains the Development Specifications for this CPC.

3.1 Computer Program Definition

The hardware used by the ATC subsystem consists of:

1) One DEC VAX-11/750 processor, with one 67MB disk drive, console printer, tape drive, and 1M of real memory. This processor is dedicated to the ATC simulation task.

2) One Sanders System 7 graphics controller, driving three calligraphic display stations. Each station has an attached alphanumeric keyboard with lighted, programmable function keys. These screens are used to simulate ATC radar displays.

3) Three TI 940 alphanumeric terminals (132 columns, 24 lines) with multiple scroll region capability. These screens are used to assist the pseudo-pilots in flying the pseudo-aircraft.

When in the standalone mode, the ATC subsystem is a self-sufficient program and does not interact with any other MVSRF components.

When in the single or dual cab mode, the ATC subsystem communicates with the cab computer(s), the Host computer(1) , and the Audio subsystem software.

From the cab computers the ATC subsystem receives information on the "state" of the cab simulators. This will minimally consist of position, altitude, and identification for each active cab. In addition changes in the frequency selection on the voice channels will be sent to the ATC subsystem from the cab computers.

The experimenter will be able to control the execution of the ATC subsystem through commands entered on his console or sent to the ATC computer by the Host computer.

The ATC subsystem provides the Audio Subsystem with information regarding which micro/headphone position should be linked to which audio loop, and which voice disguiser parameters to be used. The audio subsystem provides the ATC subsystem with mike PTT button closure information. This information is used by the ATC subsystem to control (indirectly through the Audio

---

(1) The Host computer will be one of the cab computers but the ATCSIM to Host computer communications are functionally distinct from the ATCSIM to cab communications.

subsystem) the allocation of frequencies to audio loops as well as the allocation of frequencies and voice disguising parameters to pseudo-aircraft following their assignment to an air traffic controller and a pseudo-pilot.

3.1.1 System Capacities

In its maximum configuration the ATC subsystem software will consist of:

1) The SIMCON process

2) The POSGEN process

3) Three "copies" of the SECTOR process

4) Three "copies" of the PPILOT process

The capacities and design requirements in this section will, unless stated otherwise, pertain to this maximum configuration.

The ATC subsystem will occupy approximately 400K bytes of code. The memory requirements for data storage will be approximately 500K. This means that the entire program will be able to be in memory at any given time. Therefore, paging of code and data in and out of real memory will be avoided.

The periodic functions of the ATC subsystem(1) will require about 50% of the available CPU time on the VAX-11/750 computer. This estimate assumes that all 40 pseudo-aircraft are active in the simulation. The non-periodic functions will, on the average, consume another 15% of the CPU time. It is estimated therefore, that the average CPU utilization rate will be 65%. This of course does not preclude instances where saturation of the CPU will occur. In cases like that, the periodic functions will have priority access to the CPU so that the program timing will be unaffected.

The six displays (three for the ATC stations and three or the pseudo-pilot stations) will be the major output load for the ATC subsystem.

Preliminary calculations, using the available data for the SANDERS Graphic-7 system, show that the capacity of the graphics controller is approximately 10,000 one inch vectors at a

---

(1) See references 2.2-2.5

flicker-free rate of 30Hz. Based on this estimate, each ATC display will be capable of containing up to 20 aircraft This will allow each radar display to contain up to 20 aircraft with their alphanumeric tags, airway network display of up to 40 waypoints, as well as the required alphanumeric (tabular) data. Note that this adds up to a total of 60 aircraft displayed in all three displays. This allows for an ample amount of duplications (i.e. one aircraft displayed in more than one ATC screen). Such duplications will occur often, for example when handoffs are being made, etc.

Each pseudo-pilot display will be capable of displaying information on 12 pseudo-aircraft. It is expected that this number exceeds the maximum number of aircraft that can be handled simultaneously by even an "experienced" pseudo-pilot. Note that the simulation's capacity in terms of number of pseudo-aircraft is limited by the maximum number that can be handled by pseudo-pilots. In the current configuration up to 36 pseudo-aircraft can be active. The 9600 baud rate available for the pseudo-pilot displays will be more than adequate to produce almost instantaneous screen updates even if the majority of the information on the video screen has to be modified.

3.1.2 Interface Requirements

The interface requirements between ATCSIM and the Host computer are specified in the Host computer - ATC subsystem Interface Control Document. (reference 2.6); the interface requirements between ATCSIM and the Audio subsystem are specified in the ATC subsystem - Audio subsystem Interface Control Document (reference 2.7).

3.2 Detailed Functional Requirements

The following subsections describe in detail the functions performed by ATCSIM. Each function is described within the process it executes.

3.2.1 Simulation Control (SIMCON)

The SIMCON process is made up of 8 major functions: INITX, MBXDRV, CMDINT, XTIMER, SCRIPT, AUDIO, RECDAT, and TERMX. With the exception of INITX and TERMX which are only executed once for each run, all SIMCON functions are designed for "parallel" execution. Each of these is triggered by the occurrence of internal or external events. Typical such "events" are the expiration of a time interval, receiving a mailbox message, receiving input from the ATC console or the host computer, etc. All events are signaled by the VMS operating system through software interrupts called Asynchronous System Traps (AST's). When an AST is signaled VMS passes control to a routine designated to respond to the particular event that caused it. The AST routine can perform the required processing in its entirety, set a flag which can be polled later by other modules responsible for responding to the event, or perform partial processing and set a flag to signal that further processing is necessary.

The overall functional flowchart of SIMCON is shown in figure 3.2.1-1. The execution starts by a call to INITX. Subsequently the main loop of the program is executed repeatedly as required by the flags polled in module GETFLG. If no flags are set the program hibernates until some AST routine awakens it. Each of the three functions (SCRIPT, CMDINT, and AUDIO) are then called to process any pending functions. As each function is called it first checks its flag. If it is not set the function returns immediately Otherwise the flag is reset and the required processing is performed. Note that XTIMER and MBXDRV are not explicitly called since they are directly invoked by the AST mechanism.

3.2.1.1 Initialize Execution (INITX)

3.2.1.1.1 Inputs

This function accepts the following inputs:

1) The mode for the run (i.e standalone, single cab, dual cab).

2) The host computer identification (if the run is not standalone).

3) The name of the file containing the script to be used for the run.

SIMCON

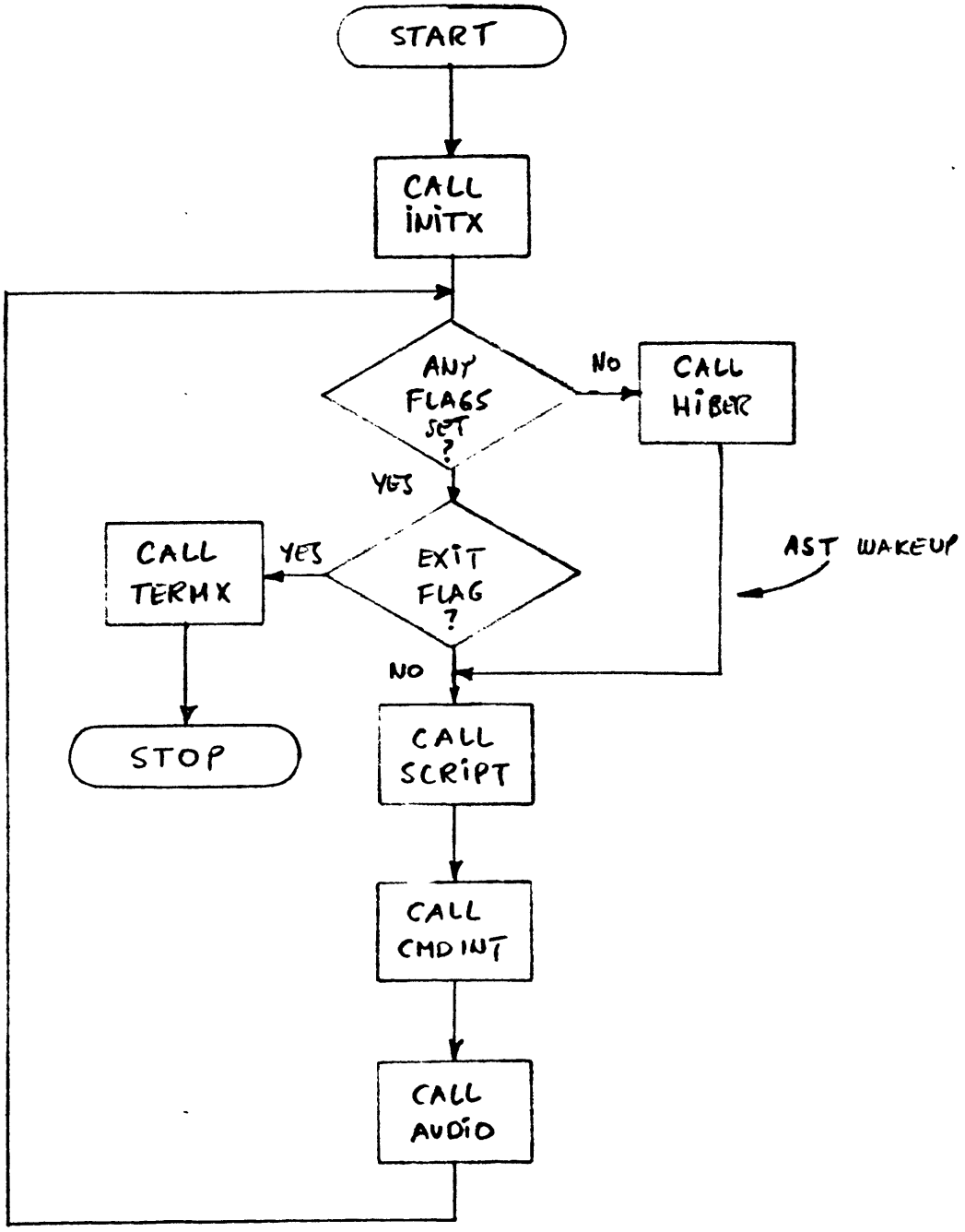

Figure 3.2.1 - 1

4) The number of ATC (and pseudo-pilot) stations to be used for the run. This is required only if fewer than 3 such stations will be used.

3.2.1.1.2 Processing

This function is invoked once in the beginning of the run. It takes as inputs the configuration selections made by the experimenter and keyboarded through the ATC console. Based on those, it initializes the ATC subsystem configuration. The overall flowchart for this function is shown in figure 3.2.1-2.

The first step in the initialization procedure is to set up the database for the ATC subsystem. Module CREGBL allocates the memory required by all the global datasets. PRCIM, STAIM, and DYNIN initialize the data in PROCESS, STATIC and DYNAM global sections (see section 3.5 for description of the data in each of these global sections). PRCIM and STAIM have subsidiary modules, each responsible for reading in a specific data table. These are listed in section 3.5. Run-independent data are fully initialized here, while only the data structures (lists, queues etc.) for run-specific data are initialized. Values for run-specific data are assigned as the configuration setup unfolds.

Other ATC processes required by the requested configuration are loaded next. Minimally the ATC subsystem requires the POSGEN process, one SECTOR process, and one PPILOT process. The maximum configuration for the current version will be the POSGEN process, three (3) SECTOR processes, and three (3) PPILOT processes. The required processes are created by module CREPRC.

I/O channels required for the internal communications of the ATC subsystem processes are setup by module CREMBX. Each process is assigned a mailbox for the reception of its messages.

Finally, module SCRPIM initializes the data in the SCRIPT global section and sets up the communications with the experimenter's console if the single or dual cab mode is specified. A connection with the host computer is now established and the ATC console can be disconnected. The connection protocol includes clock synchronization and other such coordination procedures. Before exiting, the INITX function sets the SCRIPT flag which will cause the SCRIPT function to be invoked immediately in order to setup the run specific data according to the script on which the run is going to be based. The program now enters the normal execution loop waiting for the message to start real-time operation of the ATC simulation. During this time other script directives can be entered and processed.

INITX

(SIMCON)

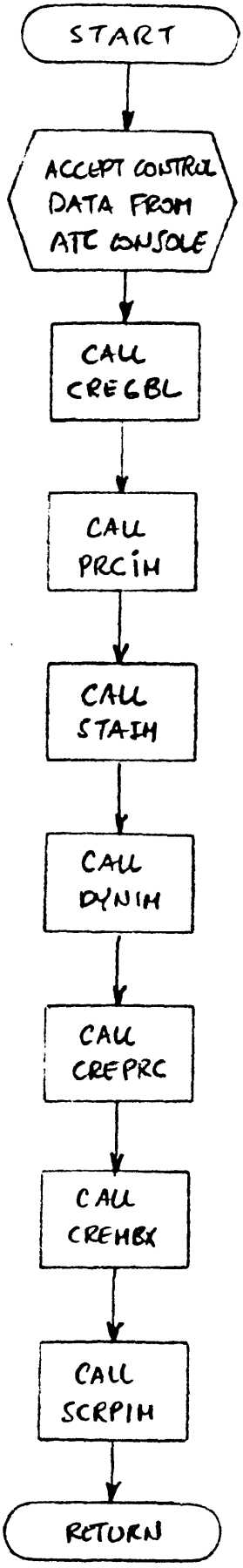

Figure 3.2.1-2

3.2.1.1.3 Outputs

The fully initialized ATC system is the output of this function.

3.2.1.2 Command Interpreter (CMDINT)

3.2.1.2.1 Inputs

This functions accepts and parses an alphanumeric input string placed in the command buffer. The list of possible commands includes:

1) Commands to start and end the real time execution.

2) Commands to activate and deactivate ATC sectors and pseudo-pilot stations.

3) Commands directing the generation of aircraft with specified characteristics at specific points in the simulated region.

4) Commands to modify the simulation data base.

5) Commands to send a message to one or more of the simulation operators (including the experimenter).

The exact syntax of the allowable commands will be specified in the ATC Subsystem User's Manual (to be provided upon completion of the coding).

3.2.1.2.2 Processing

The command interpreter parses and acts upon commands dispatched to it by SCRIPT (see section 3.2.1.4). In most cases the commands will be script directives indicating modifications to be made to the ATC environment. There is however a restricted set of commands (notably START and END) that relate to controlling the simulation execution environment.

Some of the script directives are acted upon directly by this function. Such directives are ones that require the setup of a new ATC sector or request changes in the average number of aircraft generated per hour at specific airports of fixes. Other

directives cause transfer of messages to other functions that are then responsible for the further processing of the directive. Such directives are the ones that force generation of specific aircraft at specific locations and times, for the purpose of creating an encounter or of causing other traffic complication to occur.

In the flowchart of figure 3.2.1-3, TIMRIM initializes the timing for the real time execution by causing an AST to be generated for every timer chain. In the current configuration there is only one such timer chain (see reference 2.2). ACTPRC initializes the process specific data of the process (i.e. SECTOR or PPILOT) that is to be activated. This module also generates the message to be sent to the process. The data initialized here include the sector name the frequency for the audio communications, etc. Finally, MBXSND is a general purpose module which sends the input alphanumeric string to the mailbox of the process whose identification it also accepts as an input. This module is used by all processes of the ATC subsystem for dispatching mailbox messages.

3.2.1.2.3 Outputs

The outputs from the CMDINT function will depend on the input command. Possible outputs are:

1) Initialization of the timer chains.

2) Setting of the exit flag which will cause the simulation to end.

3) Dispatch of mailbox messages to other ATC processes.

4) Dispatch of messages or "cues" to the experimenter.

5) Modification of a variety of data in the simulation database.

3.2.1.3 Execution Timer (XTIMER)

3.2.1.3.1 Inputs

The inputs to this function are:

CMDINT
(SIMCON)

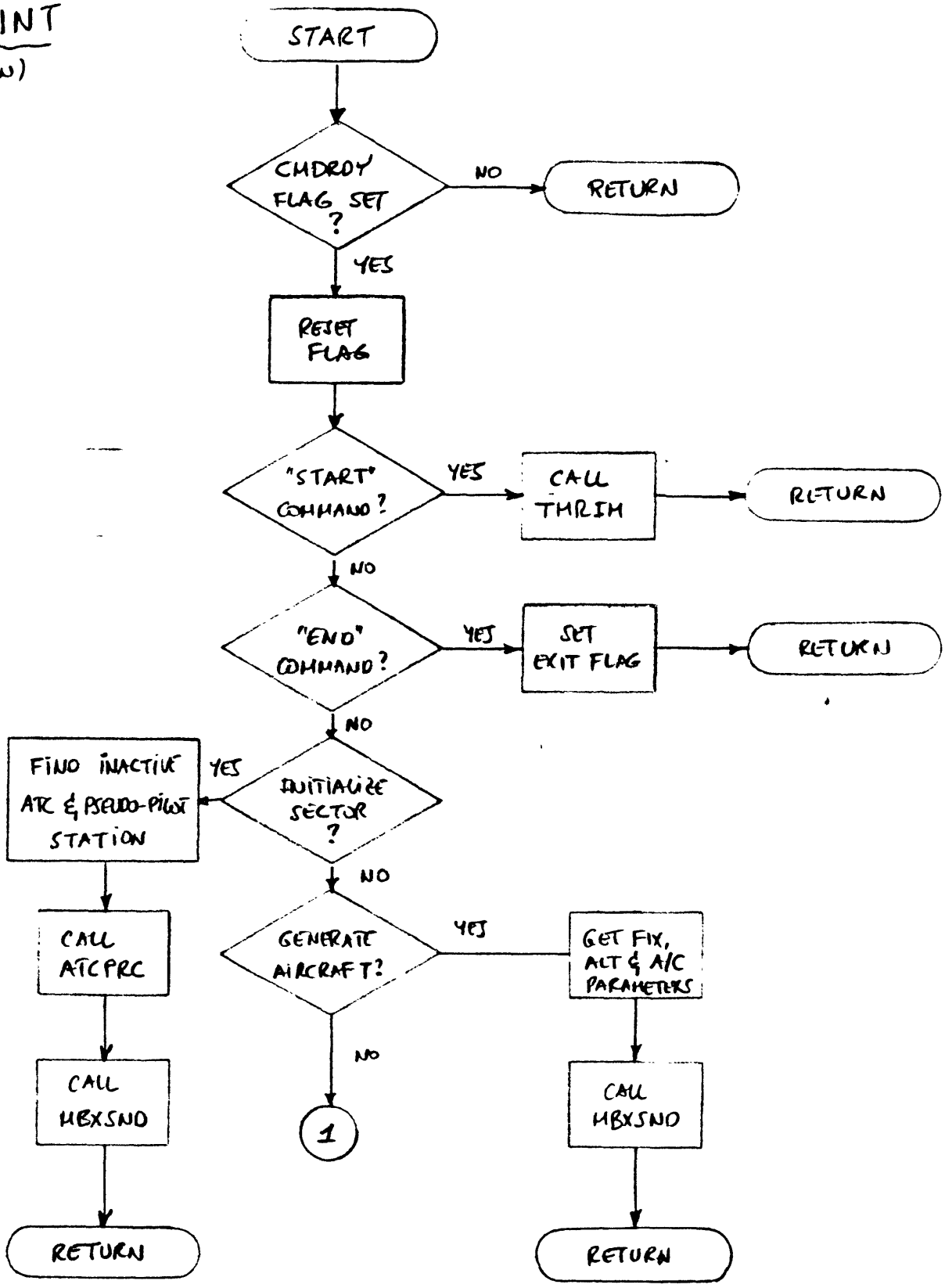

Figure 3.2.1-3 (continues)

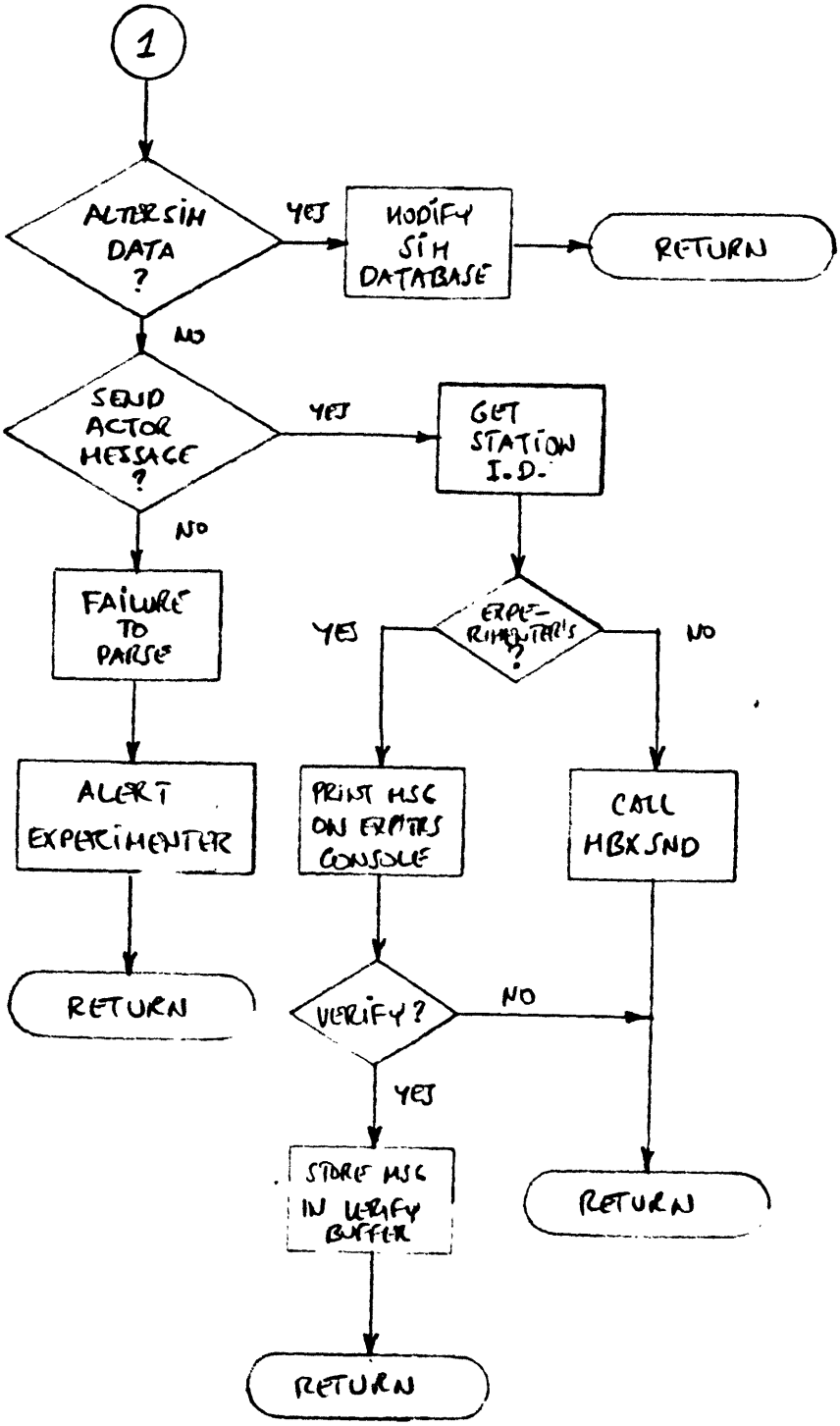

Figure 3-2-1-3 (concluded)

1) The identification number of the timer chain being processed.

2) The identification number of the process that was last invoked in this timer chain.

3) Supplementary data on the timer chain. These include the period of the timer chain (in milliseconds) and the sequence of processes that make up the chain.

3.2.1.3.2 Processing

The execution timer is responsible for invoking the periodic functions of the ATC simulation. The timing is performed by defining "timer chains" each composed of a list of ATC processes. Each timer chain also has a chain identification number and a specific time interval as its period. SIMCON associates a timer alert with each of the defined timer chains. This function is invoked either directly by a timer alert signaled through an AST, or indirectly by the MBXDRV. The flowchart for XTIMER is shown in figure 3.2.1-4.

In the first case the timer chain is being restarted and the process identification (see inputs) will be zero. The execution timer immediately requests a new timer alert to be signaled at the end of the next time interval appropriate to the timer chain that caused the alert. The process which is first in the timer chain in question is then informed through a mailbox message which will invoke the required function (or sequence of functions) within that process.

Upon completion of the requested functions, the invoked process sends a "task completed" message to SIMCON. MBXDRV (see section 3.2.1.6) responds by invoking XTIMER in order for the timer chain to be continued. This is the second method of invoking XTIMER. The process identification is now non-zero and XTIMER responds by invoking the next process in the timer chain or terminating the chain if there is no other process in the chain to be invoked. As always MBXSDN is used to transfer the mailbox messages.

The logic is set up to handle any number of periodic timer chains though the current design requires only one timer chain to be implemented.

XTIMER

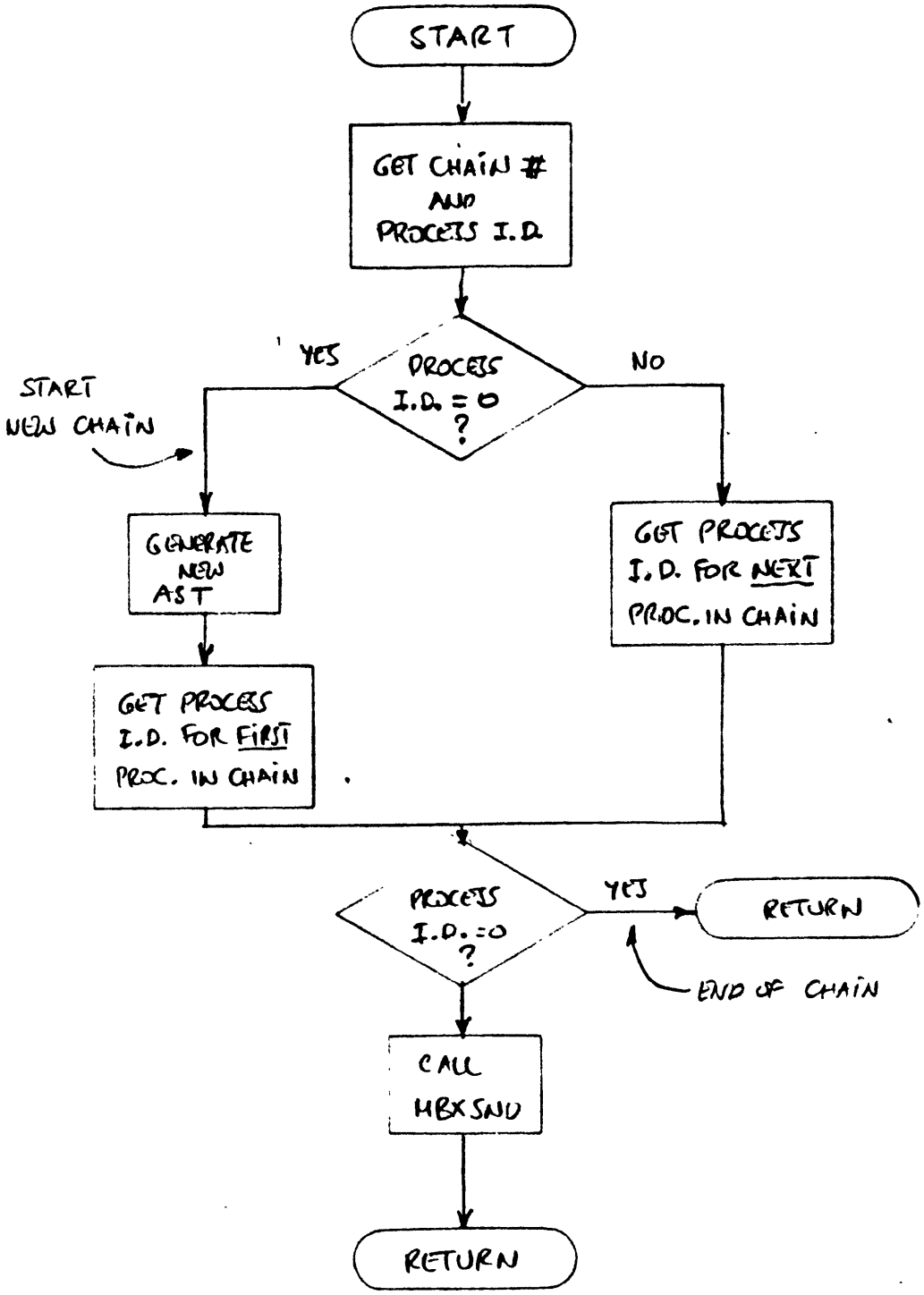

Figure 3.2.1 - 4

3.2.1.3.3 Outputs

The function has two outputs:

1) Queueing of a new timer AST.
2) Mailbox messages sent to various ATC processes.

3.2.1.4 Script Control (SCRIPT)

3.2.1.4.1 Inputs

This function takes as input an alphanumeric string. The string will be called "script directive" and is composed of one or more "trigger directives" followed by one or more "action directives" or commands. Each type of directive is separated from the next directive of the same type by a semicolon (";"). The last trigger directive is separated from the first action directive by a colon (":"). SCRIPT only processes trigger directives, treating action directives as data.

The exact syntax of the trigger directives will be specified in the ATC Subsystem User's Manual (to be provided upon completion of the coding).

3.2.1.4.2 Processing

SCRIPT is responsible for the management of script directives input from the host computer, the ATC console or the script file. Inputs from all sources are in the same format so that the origin of the script directive is of no significance to its processing.

Trigger directives define simulation events which fall in the following generic categories:

1) Events related to simulation time. Such events assume the truth value of relational expressions between the time specified in the event directive, (ET), and the current simulation time, (ST). The allowable forms for such expressions are:
    a) ET = ST
    b) ET > ST
    c) ET < ST
    d) ET <= ST
    e) ET >= ST

f) ET $\neq$ ST

2) Events related to aircraft position or altitude. The above 6 forms of relational expressions will be allowable for each of the three components (latitude, longitude, and altitude). In interpreting the relations "greater than" and "less than", the projection of the aircraft ground speed vector will be used to determine the positive direction along the latitude and longitude axes. Similarly, a discrepancy in any component that is within the limits of error of the navigational and/or surveillance equipment will be interpreted as satisfying the equality relationship.

3) Events related to push-to-talk button activation at a specified audio stations.

4) Events related to specific key depression at specific stations. 128 such events can be active at any time. In essence these allow any operator the define quick action keys in real time. They will be referred to as "manually activated discrete events".

A significant event is a specific instance of one of the above generic event types which, when it occurs will trigger some action directive.

SCRIPT is invoked when a new command has been received, when new entity (e.g aircraft or ATC sector) has been generated or activated or when a significant event has occurred. The flowchart of figure 3.2.1-5a depicts the three cases.

In the first case, the directive is passed to DIRPRS for parsing. In the second case, SCANSF is called to scan the script file and activate events related to the newly generated entity. Finally, if the function has been invoked by the occurrence of a significant event, the event is removed from the event list and the associated script directive (i.e. the unparsed portion of the original script directive) is sent to DIRPRS for further parsing.

The processing detail within DIRPSR is show in figure 3.2.1-5b. The parser identifies the first trigger directive of the input. If none is present the CMDINT is invoked by setting the CMDRDY flag. If one is found, DIRPRS creates a new entry in the significant event list, associates with it the remaining of the input line, and invokes the function most appropriate for monitoring the event. If the trigger directive is time related, the appropriate function is DIRPRS itself. This will result in the event to be entered in the list of time triggered events kept

SCRIPT

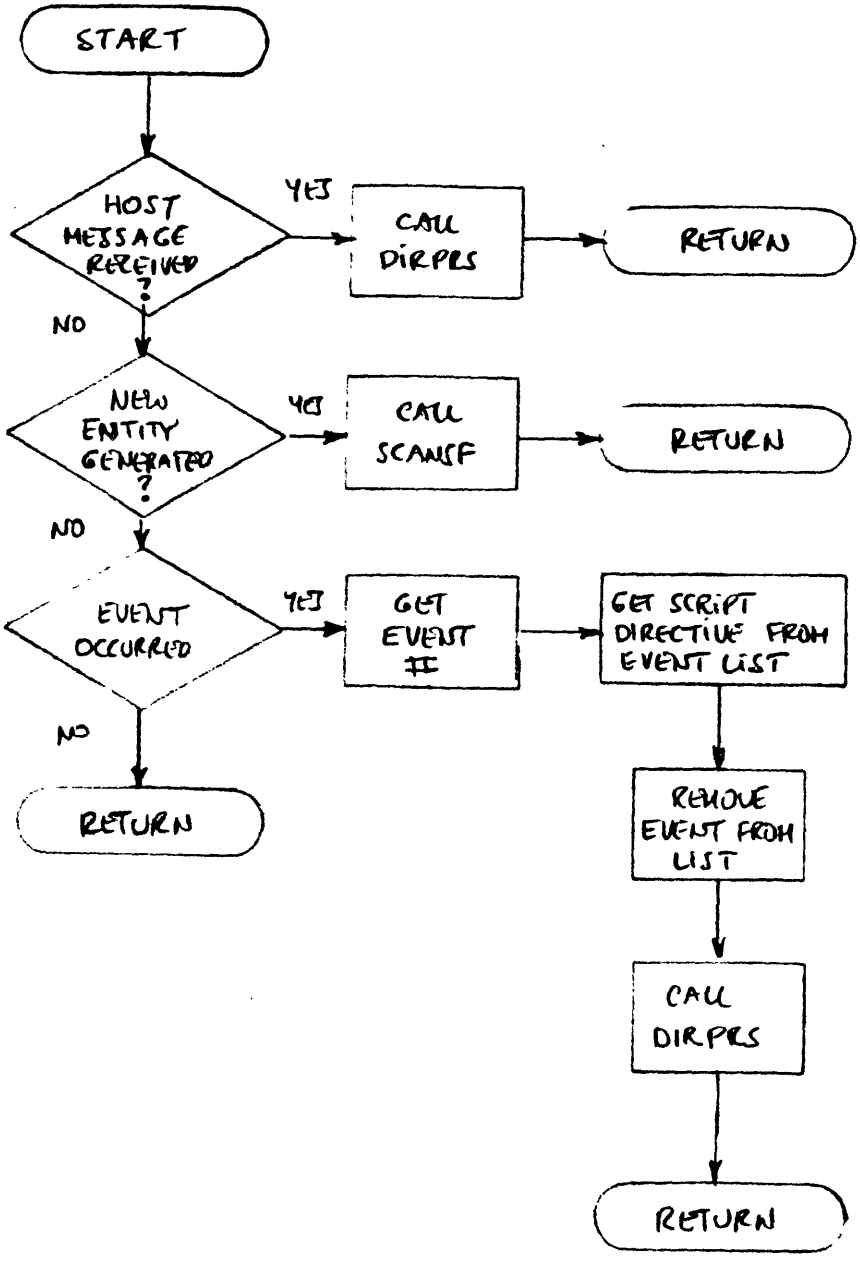

Figure 3.2.1-5a

DIRPRS

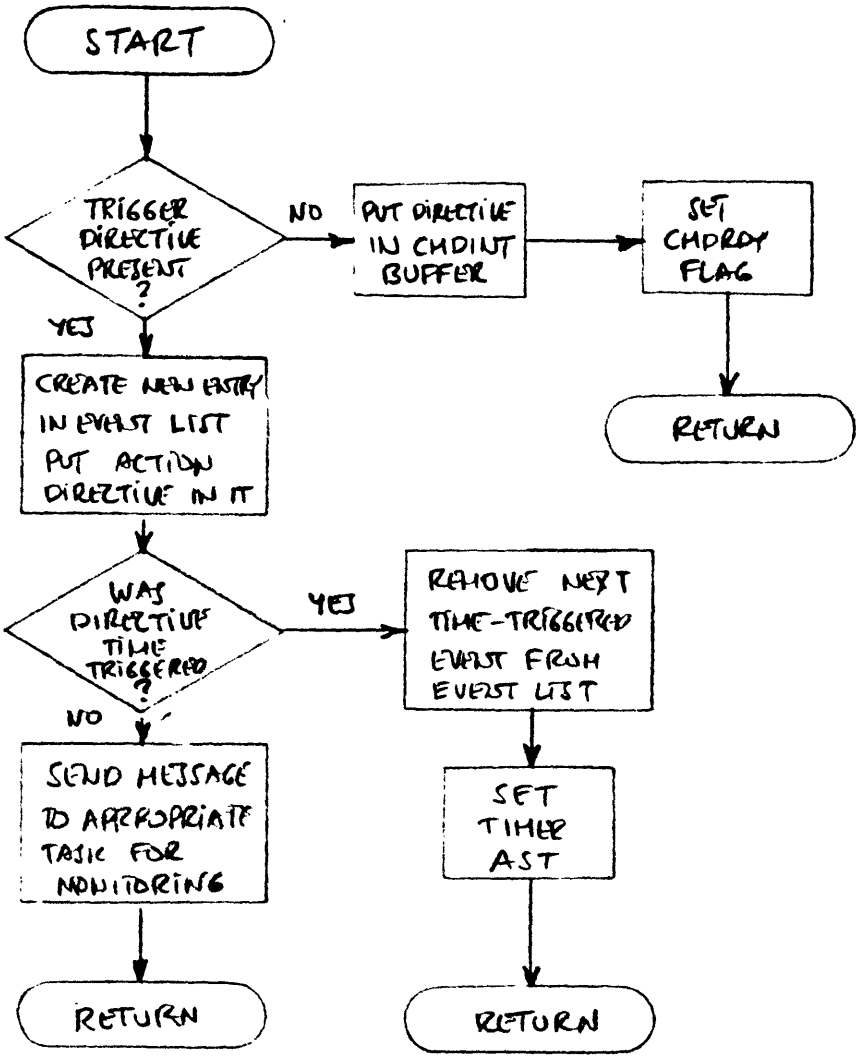

Figure 3.2.1-5b

by DIRPRS. In order to avoid having more than one time-triggered event "active" at any time, DIRPRS is required to check the reason it was invoked. If a time-triggered event has occurred (i.e. in some instances of case 3 above), DIRPRS activates the next (in terms of simulation time) event in the time-triggered event list. The activation is done by requesting a timer AST to e signalled at the appropriate moment.

The processing detail of SCANSF is shown in figure 3.2.1-5c. It scans the script file, identifies script directives that relate to the new entity and calls DIRPRS to process each such directive.

3.2.1.4.3 Outputs

This function dispatches action directives to the command interpreter, and sends messages to other processes required to monitor simulation events.

3.2.1.5 Audio Control (AUDIO)

3.2.1.5.1 Inputs

This function receives the following inputs:

1) Messages dispatched to it by MBXDRV.
2) PTT closure information from the AUDIO subsystem.

Messages from MBXDRV can be one of the following:

1) Sector activation or deactivation message.
2) Pseudo-pilot activation or deactivation message.
3) Cab frequency selection change.
4) Pseudo-aircraft handoff acceptance message.
5) Pseudo-aircraft deactivation message

3.2.1.5.2 Processing

The Audio Control task (AUDIO) is responsible for communications to and from the Audio Subsystem. The ATC subsystem obtains frequency selector position information from the host computer, indicating the frequency selected on each cab radio. The frequencies in use by each simulated ATC position at any time are stored as part of the ATC database, initialized and modified by the script. Using this information, the AUDIO task

SCANSF

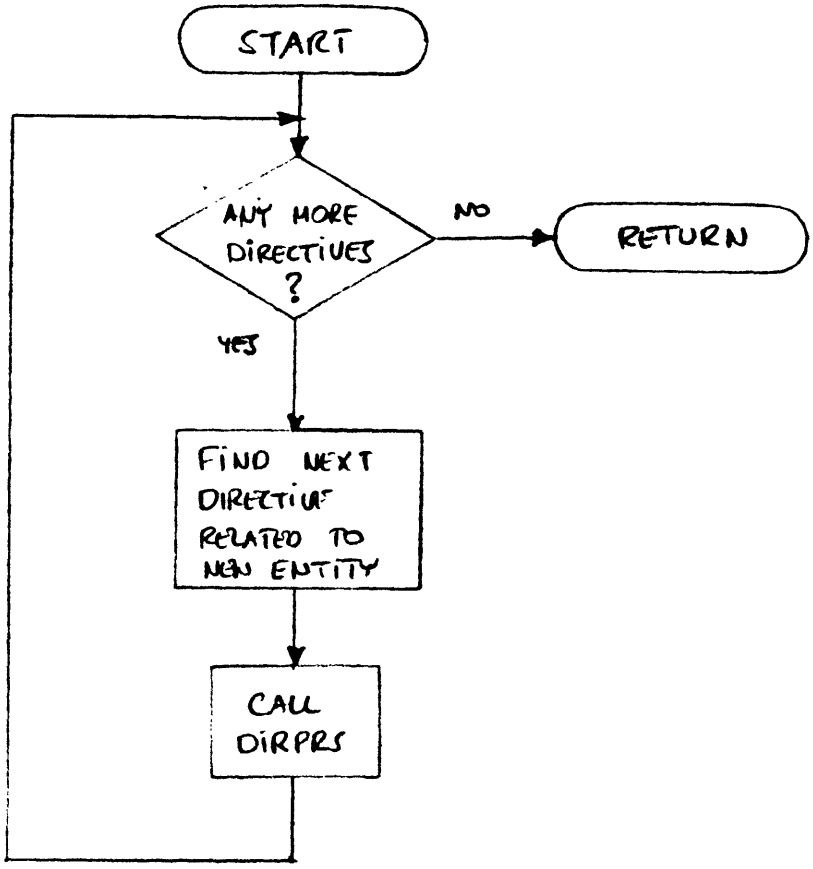

Figure 3.2.1-5c

periodically updates the Audio System with information regarding which loop an audio station should be connected to (including no loop at all, for unused frequencies), and which voice disguiser parameters to use on each PTT button on each station.

In turn, the Audio System provides the ATC subsystem with PTT closure information, which is logged by ATC, and also used to generate script-driving events.

The AUDIO task therefore performs the following functions (shown in figure 3.2.1-6):

1) Upon sector activation, call ATLOOP to determine the next available audio loop and attach the ATC controller and pseudo-pilot audio stations to that loop (with the appropriate disguiser parameters, if so specified by the script). Also call ATTPTT to attach PTT switches to all pseudo-aircraft which may be in the newly activated sector. Finally call SWCABF to attach the cab to the new audio loop if the frequencies match. The last operation is executed as if the cab had switched radio frequency.

2) Upon acceptance of an aircraft by a controller (and therefore inclusion on a pseudo-pilot's control authority), call ATTPTT to determine which voice disguiser parameters to use, and attach them to the appropriate PTT box button (the position of the button corresponds to the position of that pseudo-aircraft's data in the pseudo-pilot's screen). The same function is performed for every aircraft on a pseudo-pilot's control authority upon pseudo-pilot initialization.

3) Upon receiving a cab frequency selection change signal, call SWCABF to update the internal table, and determine what to do with that audio station: connection to an audio loop (if the frequency matches), or to none (if the frequency is not an ATC frequency: the ATC/Audio System does <u>not</u> simulate "fixed audio", such as VOR identification codes).

4) Upon deactivation of a pseudo-aircraft (handoff to another sector or actual deletion), call DETPTT to disable the corresponding PTT switch, and mark the voice disguiser parameters used by that pseudo-aircraft as available for further use. The same function is performed for every aircraft on a pseudo-pilot's control authority upon pseudo-pilot termination.

AUDIO

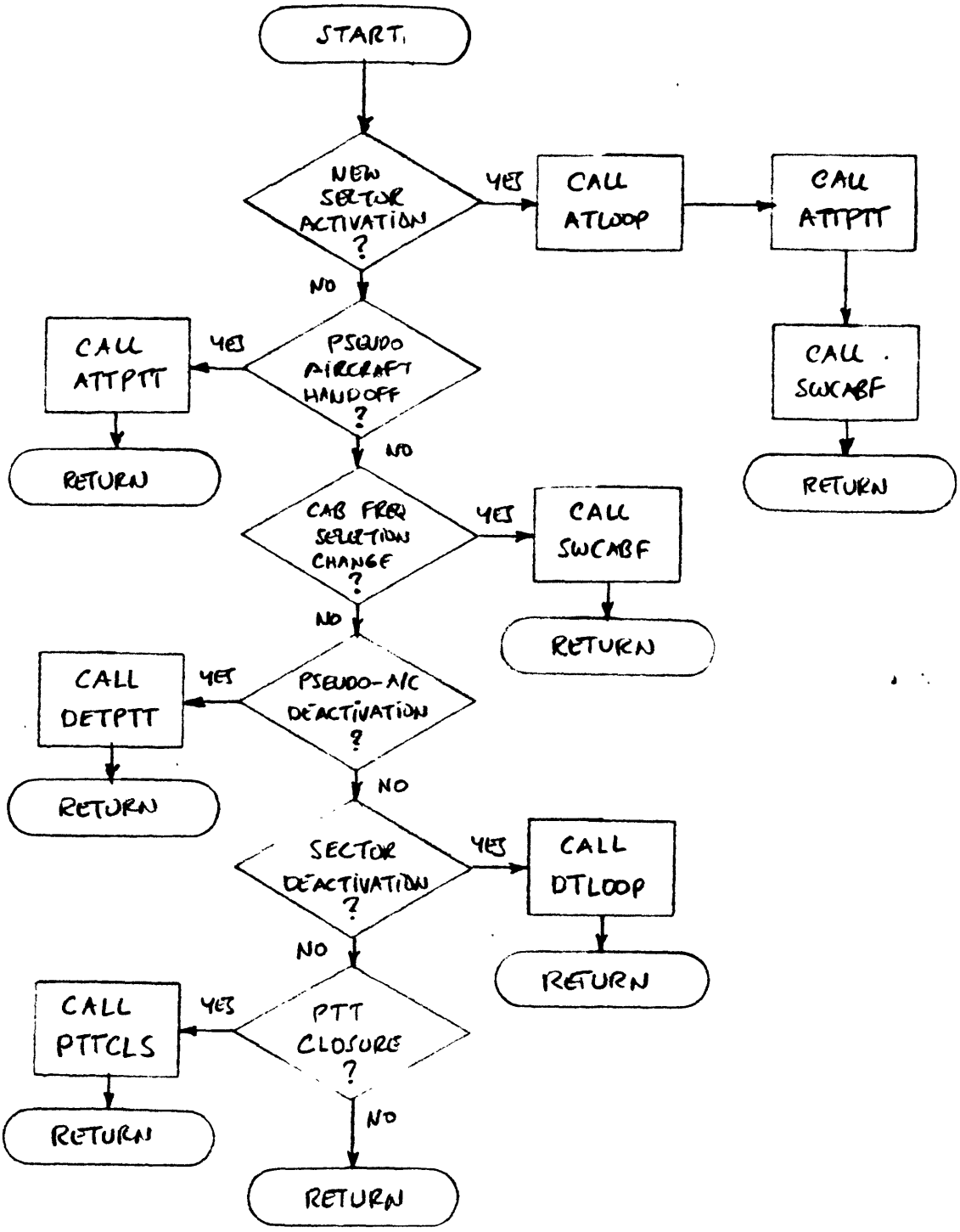

Figure 3.2.1-6

5) Upon deactivation of a sector, call DTLOOP detach all audio stations from that loop, and mark the audio loop as available.

6) Upon receiving a PTT closure signal, call PTTCLS to log the event and if closure of the PTT in question is being monitored transmit the event to the script dispatcher (SCRIPT task in the SIMCON process).

3.2.1.5.3 Outputs

This function provides the Audio subsystem of the MVSRF with the current associations between audio loops and audio stations and signals PPT closures to the script control function.

3.2.1.6 Mailbox Driver (MBXDRV)

3.2.1.6.1 Inputs

The mailbox driver receives messages from other ATC processes. The message format minimally contains the message identification, sending process identification and the message length in the first 8 bytes transmitted (2 bytes are allocated for each item and 2 bytes are reserved for future use). The format of the remainder of the message will depend on the message type.

3.2.1.6.2 Processing

The mailbox driver is responsible for reading mailbox messages received by SIMCON and invoking the function which is responsible for its further processing. With the exception of XTIMER which is invoked directly to insure quick response, the invoking mechanism is to insert the message in an appropriately setup buffer and set the flag which eventually trigger the proper function. The mailbox driver centralizes mailbox message reception and dispatching. The following types of messages are recognized as shown in figure 3.2.1-7:

1) Task completed message. This message includes the timer chain index (although it is not currently needed) for purposes of identification. The execution timer is invoked to process this message.

MBXDRV

(SIMCON)

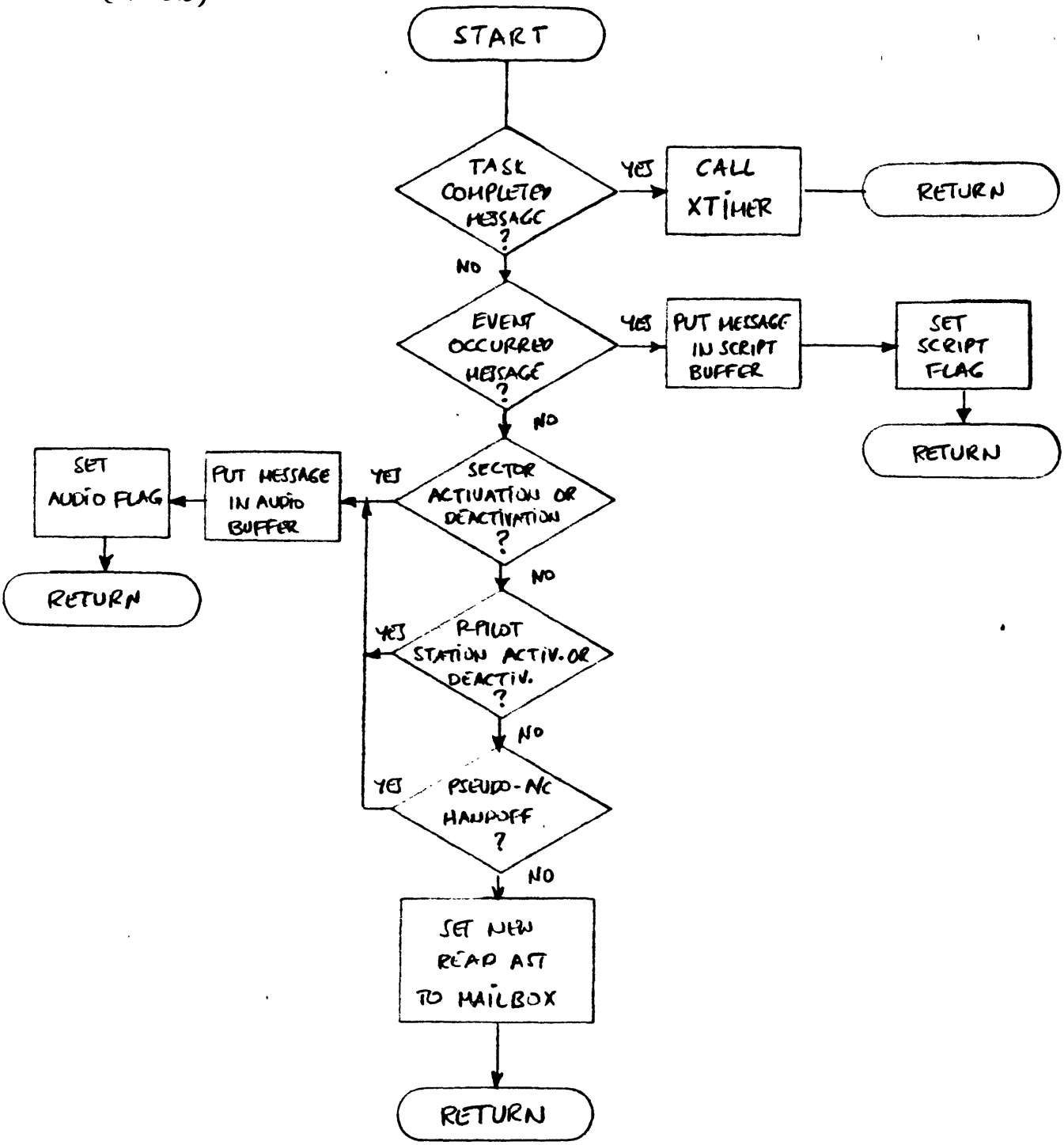

Figure 3.2.1-7

2) Event occurred message. This message includes the index to the event that occurred as it is stored in the event list. Script control is invoked for processing of this message.

3) Sector activation or deactivation message. This message includes the index of the SECTOR process. The AUDIO function is invoked to process this message.

4) Pseudo-pilot activation or deactivation message. This message includes the index of the SECTOR process. The AUDIO function is invoked to process this message.

5) Pseudo-aircraft handoff message. This message includes the identification of the pseudo-aircraft that in the handoff process. The AUDIO function is invoked to process this message.

3.2.1.6.3 Outputs

MBXDRV sets appropriate flags when necessary and puts the received messages in the input buffers of various functions that will be invoked for their further processing.

3.2.1.7 Data Recording (RECDAT)

3.2.1.7.1 Inputs

This function takes the following inputs:

1) Input strings from all communications channels (ATC to Host, ATC to ATC stations, and ATC to PPILOT stations).

2) Latitude, longitude, altitude, and velocity for all aircraft in the system. Actual as well as ground measured and airborne measured values for all the above items are recorded.

3) ID's of sector and pseudo-pilot having control authority for all aircraft in the system.

4) Disguiser parameters for all aircraft in the system.

5) PTT closure events.

6) Wind direction and velocity.

7) Cab frequency selection.

8) ID's, frequencies, disguiser parameters, and audio loop associations for all active sectors.

3.2.1.7.2 Processing

This function records the required data on magnetic tape. Each entry is accompanied by the simulation time (time logging). The function is invoked periodically, at the end of the main simulation timer loop. Recording of items 2 and 6 above is done. Subsequently this functions records any entries in the data recording buffer. Entries in this buffer have been created by other functions. I/O drivers create entries every time an input string is received on the communications channel for which they are responsible. The CMDPRC function of the SECTOR process creates an entry after a handoff acceptance has been successfully processed. The AUDIO function of the SIMCON process creates entries when disguiser parameters are assigned to pseudo-aircraft, when PTT closures are detected, when cab frequency changes are detected, and when audio loops are assigned to sector and pseudo-pilot audio stations.

3.2.1.7.3 Outputs

The items listed in section 3.2.1.7.1 as inputs are output on magnetic tape accompanied by the simulation time associated with them. The exact format for the outputs is specified in the Host to ATC subsystem Interface Control Document (reference 2.6).

3.2.1.8 Terminate Execution (TERMX)

3.2.1.8.1 Inputs

NONE

3.2.1.8.2 Processing

At the end of the run this task is activated and basically performs all the functions that are needed for a smooth completion of the program execution. The task's major function is to insure that all ATC processes are stopped and unloaded successfully. In addition it frees all unwanted resources that the program was using (e.g. mailboxes, I/O channels, etc.), closes open data sets, and stores all the data that are needed for further analysis or for continuation of the current experiment.

3.2.1.8.3 Outputs

NONE

3.2.2 Position Generator (POSGEN)

POSGEN is made up of 11 tasks: INITX, MBXDRV, CABIO, ACGEN, COMAND, DYNAM, NAVUPD, SURVEY, ACSTAT, WINDS, and CNFRMC. INITX is only executed once in the beginning of each run. MBXDRV is invoked whenever a mailbox message sent by another ATC process. CABIO manages the transfer of data to and from the cab(s) and executes at the AST level. All other tasks are invoked by the mailbox driver following a message from the XTIMER task of SIMCON requesting the pseudo-aircraft position update to be performed. The top level flowchart for this process is shown in figure 3.2.2-1. The mathematical formulas used in the various POSGEN functions are described in great detail in references 2.8 and 2.9.

3.2.2.1 Initialize Execution (INITX)

3.2.2.1.1 Inputs

This function uses as inputs the process specific data for POSGEN as they have been prepared by the PRCIM module of SIMCON.

3.2.2.1.2 Processing

This function is invoked once in the beginning of the run. Its flowchart is shown in figure 3.2.2-2. It is responsible for mapping to the global data sets (MAPGBL), assigning I/O channels to the mailboxes (ASNMBX), initializing communications with the cab(s) (CABIM), initializing local data, and for setting the appropriate AST routines so that the process is ready to go into the normal execution phase. ASGMBX and CABIM are responsible for initializing the AST delivery for the mailbox messages and the communications with the cabs respectively. Prior to exiting INITX calls MBXSND to send a ready message to SIMCON.

3.2.2.1.3 Outputs

The output of this function is the ready message sent to SIMCON.

POSGEN

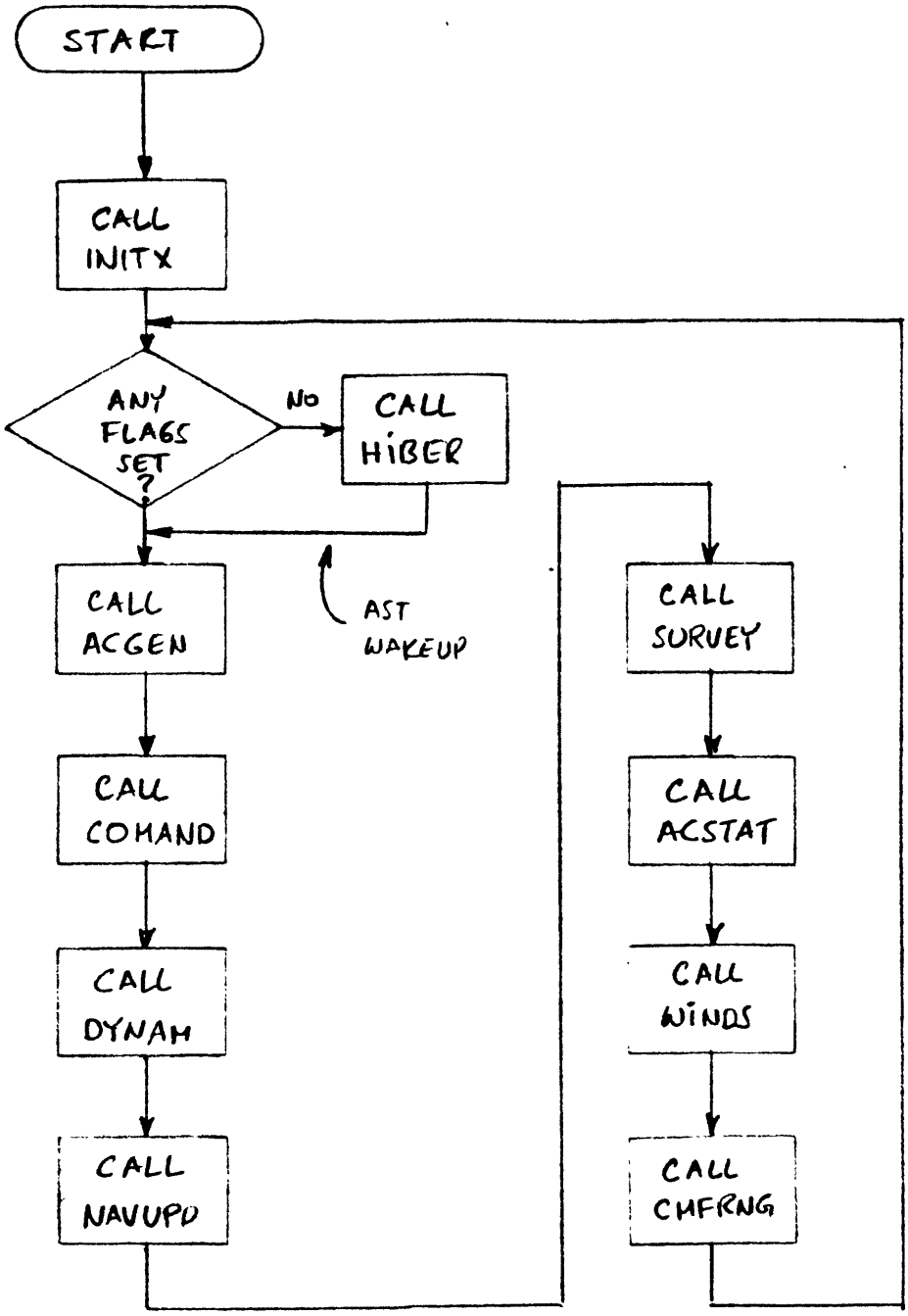

Figure 3.3.2-1

INITX

(POSGEN)

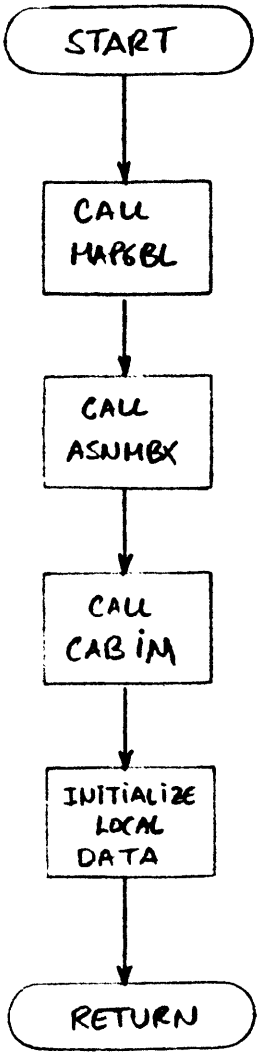

Figure 3.2.2-2

3.2.2.2 Mailbox Driver (MBXDRV)

3.2.2.2.1 Inputs

The mailbox driver receives messages from other ATC processes. The message format minimally contains the message identification, sending process identification and the message length in the first 8 bytes transmitted (2 bytes are allocated for each item and 2 bytes are reserved for future use). The format of the remainder of the message will depend on the message type.

3.2.2.2.2 Processing

The Mailbox Driver manages the messages that are sent to POSGEN from other ATC subsystem processes. Messages request one of the following (see figure 3.2.2-3):

1) Pseudo-aircraft position update. This request is sent by the XTIMER function of the SIMCON process. The mailbox driver responds by setting the POSUPD flag waking the process and requesting a new AST to be signalled upon reception of a subsequent mailbox message.

2) Insert new event in the significant event list. This request is sent by the SCRIPT task of the SIMCON process. The ACSTAT function is invoked by this message by setting the NEWEV flag and putting the message in the ACSTAT buffer. Signalling the occurrence of a significant event is left up to various POSGEN tasks.

3) Insert new command in the pseudo-aircraft command queue. This request is sent by the CMDPRC task of the PPILOT process. The COMAND function is invoked by this message by setting the NEWCOM flag and putting the message in the COMAND buffer.

4) Pseudo-aircraft generation directive. This request is sent by the CMDINT function of SIMCON. ACGEN is invoked by setting the ACGEN flag and putting the message in the ACGEN buffer.

5) Execution termination. This message is sent by the CMDINT task of the SIMCON process. The EXIT flag is set and the execution termination task will be

MBXDRV

(POSGEN)

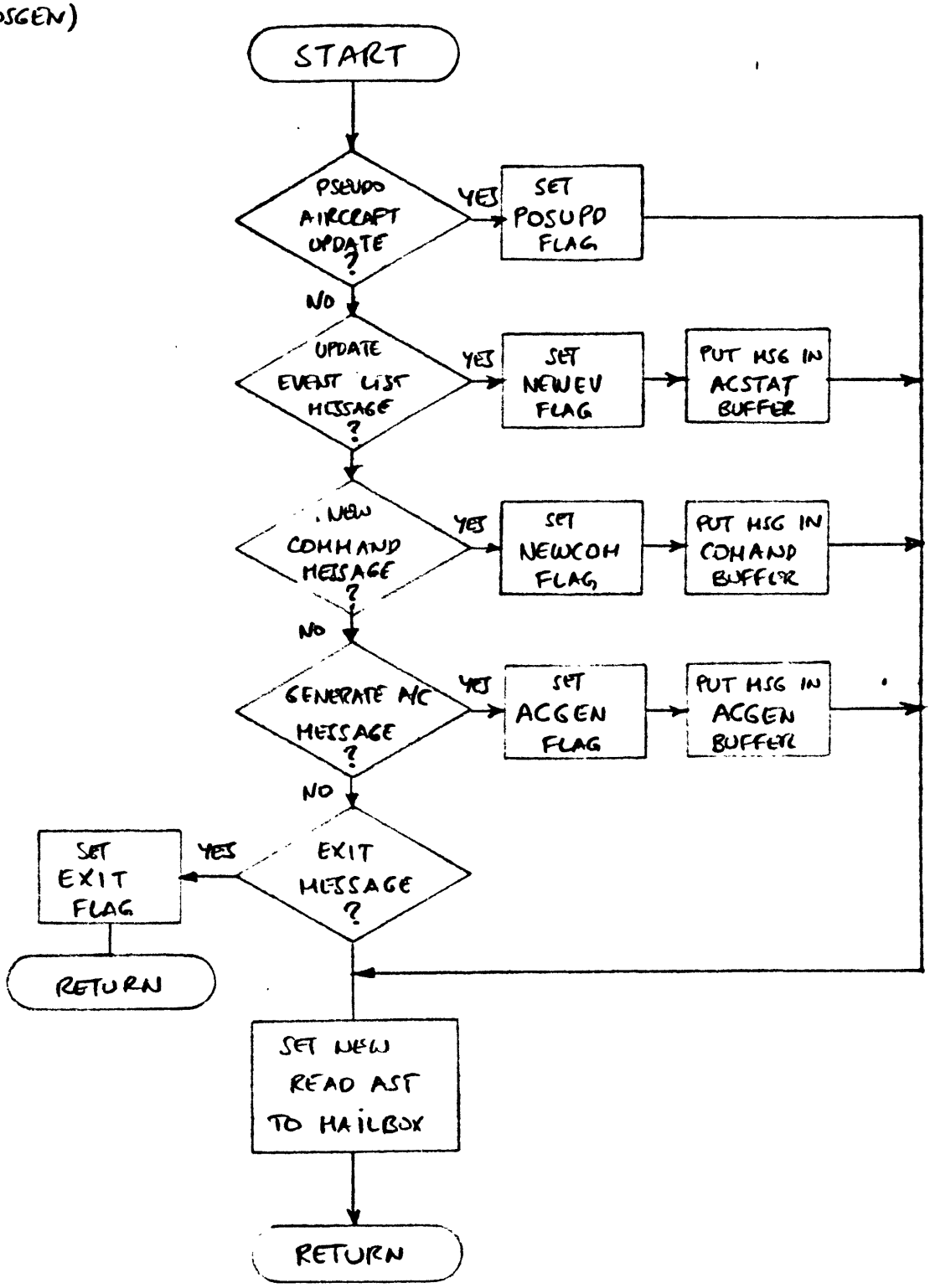

Figure 3.2.2-3

invoked when all other pending tasks have completed execution. When this request is received no further mailbox messages are read.

3.2.2.2.3 Outputs

MBXDRV sets appropriate flags when necessary and puts the received messages in the input buffers of various functions that will be invoked for their further processing.

3.2.2.3 Cab I/O Interface (CABIO)

3.2.2.3.1 Inputs

The inputs to this function are data sent by the two cabs. The data format and exact communication protocol and timing will be specified in the Host Computer - ATC subsystem Interface Control Document (reference 2.6). The data minimally required by POSGEN will consist of:

1) Cab identification (Airline name, Flight number, etc)

2) true longitude, latitude and altimeter reading for each cab.

3) Cab radio frequency selections.

4) Cab IDENT button depression information.

Data item 1 will be required when each cab first enters the simulation. Data item 2 will be required at regular intervals of four (4) or ten (10) seconds depending on the radar which is tracking the aircraft. Data items 3 will be required whenever a frequency change occurs in the cab. Data item 4 is required whenever the IDENT button on the transponder is depressed by the cab pilot.

3.2.2.3.2 Processing

This function manages the data flow to and from the cab(s). The function operates at the AST level. Depending on the inputs the following functions will be performed (see figure 3.2.2-4):

CABIO

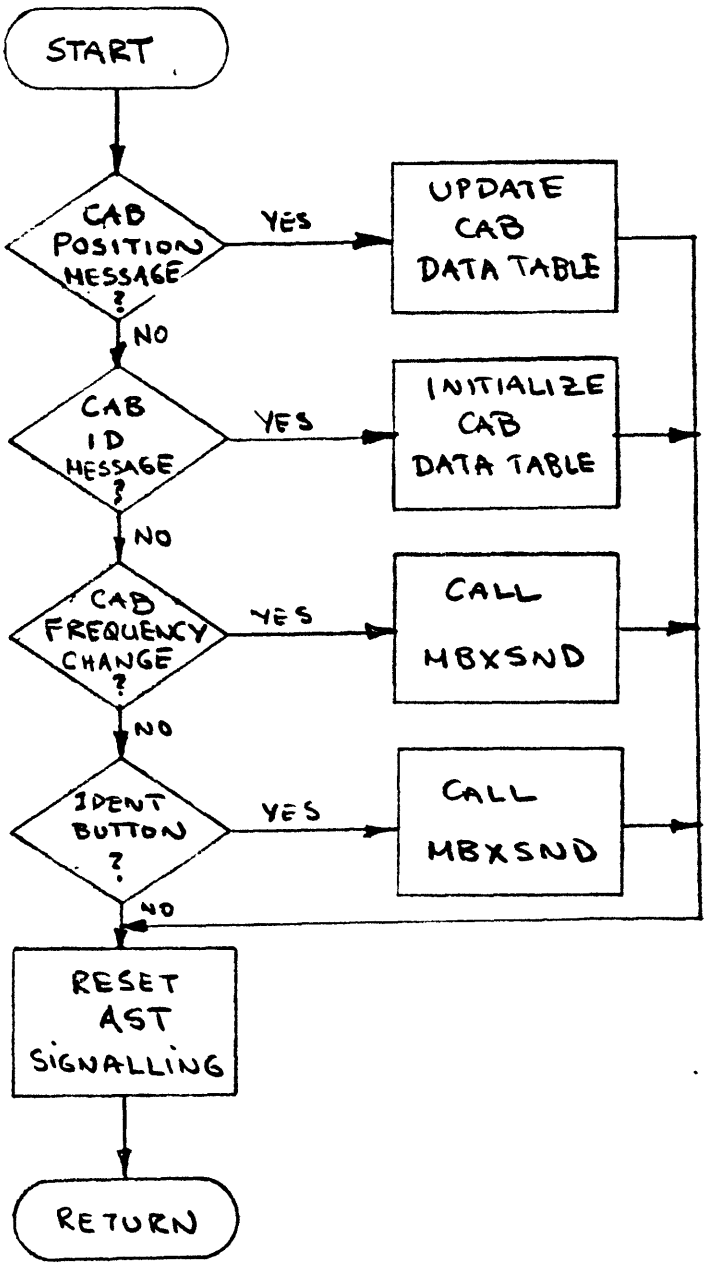

Figure 3.2.2-4

1) When the periodic position information is received from either cab, CABIO updates the appropriate entries in the aircraft data tables maintained by POSGEN.

2) When cab identification information is received, CABIO initializes the aircraft data table.

3) When frequency selection changes or IDENT depression messages are received, CABIO sends the information to SIMCON or SECTOR respectively via a mailbox message.

3.2.2.3.3 Outputs

The outputs of this function are:

1) Updated aircraft data tables for both cabs.
2) Messages to SIMCON concerning the Audio control function.
3) Messages to SECTOR concerning IDENT button depression.

3.2.2.4 Aircraft Generator (ACGEN)

3.2.2.4.1 Inputs

This function accepts as inputs:

1) The ACGEN buffer.
2) The data describing the generation process.

3.2.2.4.2 Processing

The aircraft generation task manages the generation of new pseudo-aircraft that will participate in the simulation. Two methods of generating aircraft will be available (see figure 3.2.2-5):

1) Random generation: This method will generate new pseudo-aircraft at exponentially distributed time intervals with some prespecified mean. The location of the aircraft upon their generation will also be randomly selected from a prespecified set of

ACGEN

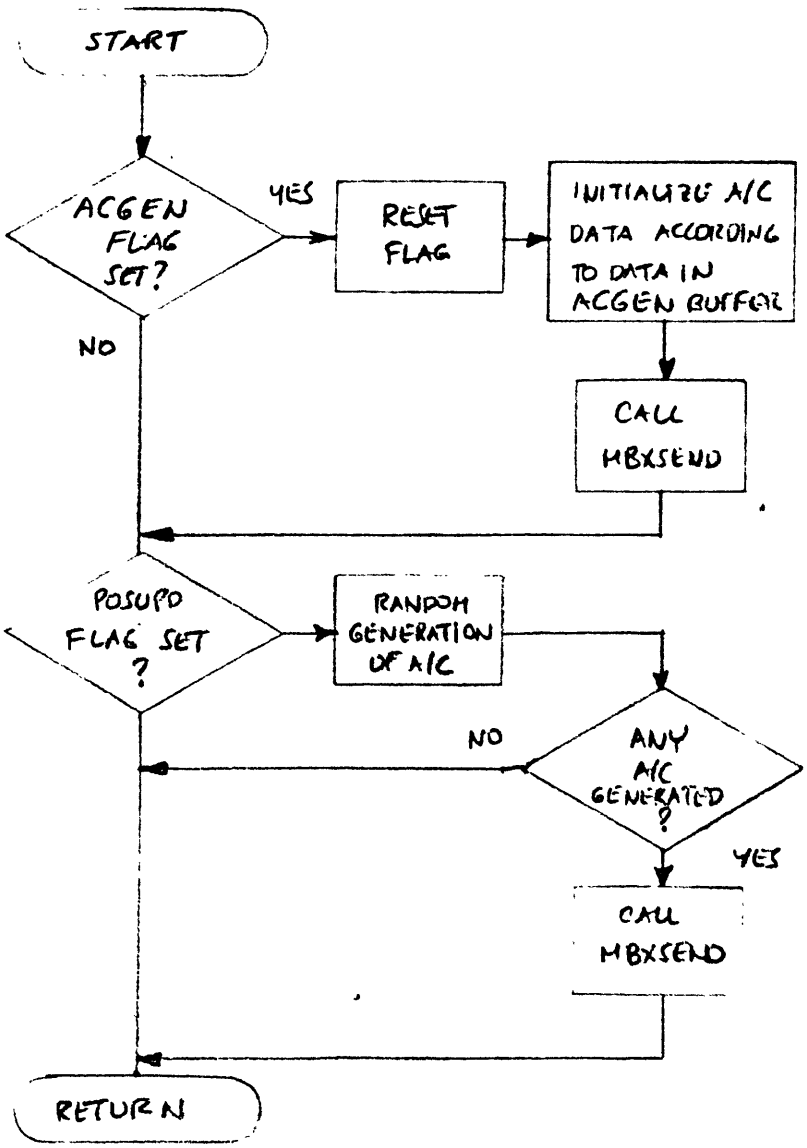

Figure 3.2.2-5

"source" points according to some discrete probability distribution. The same random procedure will be used to generate the aircraft types, flight ID's as well as other pertinent information. The procedure will be independent for each simulated ATC sector. Flight plans for such aircraft will be generated by first randomly selecting an exit fix among prespecified exit fixes appropriate for the sector and then assigning a route from the source point to the exit fix to the aircraft. If the exit fix is at a boundary with a sector that is also simulated, a new exit fix appropriate for that sector will be generated and the process will continue until the aircraft crosses to a sector that is not simulated.

2) Deterministic generation: This method will generate aircraft according to script directives. All or a portion of the pertinent data (aircraft type flight ID , etc.) can be specified by the script. As all script directives, the generation of any or all aircraft can be triggered by either reaching a specific simulation time or by the occurrence of some other significant event in the simulation. Script driven aircraft generation allows better control of the simulation environment provided by the ATC subsystem.

Independent of how the aircraft was generated SIMCON is informed through a mailbox message.

Since all newly generated aircraft "emerge" from inactive ATC sectors, it is important that, when they are handed off to the air traffic controller of an active sector, they are properly spaced from other traffic in the area. The pseudo-aircraft generator is responsible for providing the required separations thus simulating the actions of the air traffic controller responsible for the sector from which the newly generated aircraft are presumed to emerge. Pseudo-aircraft are therefore not immediately activated upon generation. Instead their status is set to DORMANT and the earliest time the aircraft can be at the source point without violating separation requirements with other traffic in the area is determined. This time is called the activation time of the aircraft. The actual activation of the pseudo-aircraft is then left up to the pseudo-aircraft status update task.

3.2.2.4.3 Outputs

New entries in the aircraft data tables are created and messages signalling aircraft generation to SIMCON are sent.

3.2.2.5 Command Queue Management (COMAND)

3.2.2.5.1 Inputs

The primary input of this function is the command queue for each pseudo-aircraft. When new commands are sent by PPILOT the COMAND input buffer contains data to be inserted in the command queue.

3.2.2.5.2 Processing

This task generates the commanded state for all pseudo-aircraft which is used as input to the dynamics update task. Piloting commands are received by POSGEN through mailbox messages by PPILOT processes. The overall flowchart for this function is shown in figure 3.2.2-6.

When a command has been received from a pseudo-pilot, INSERT will insert the command in the command queue. The command is inserted in its proper sequence depending on the activation time. The commands sent by the pseudo-pilot are assumed to be immediate.

During a normal update cycle, COMAND first checks if any of the as yet inactive commands need to be activated (ACTCOM). Command activation is done either when their activation time has been reached, or when the aircraft has reached a point in space. For example, intercepting a VOR radial is activated when the aircraft is at the proper distance from the radial. The proper distance, of course, will depend on the intercept angle. Subsequently the commanded values for the aircraft are generated based on the aircraft's current state (GENCOM). All values are generated based on the noisy readings of the onboard instruments.

3.2.2.5.3 Outputs

The following commanded values are generated:

COMAND

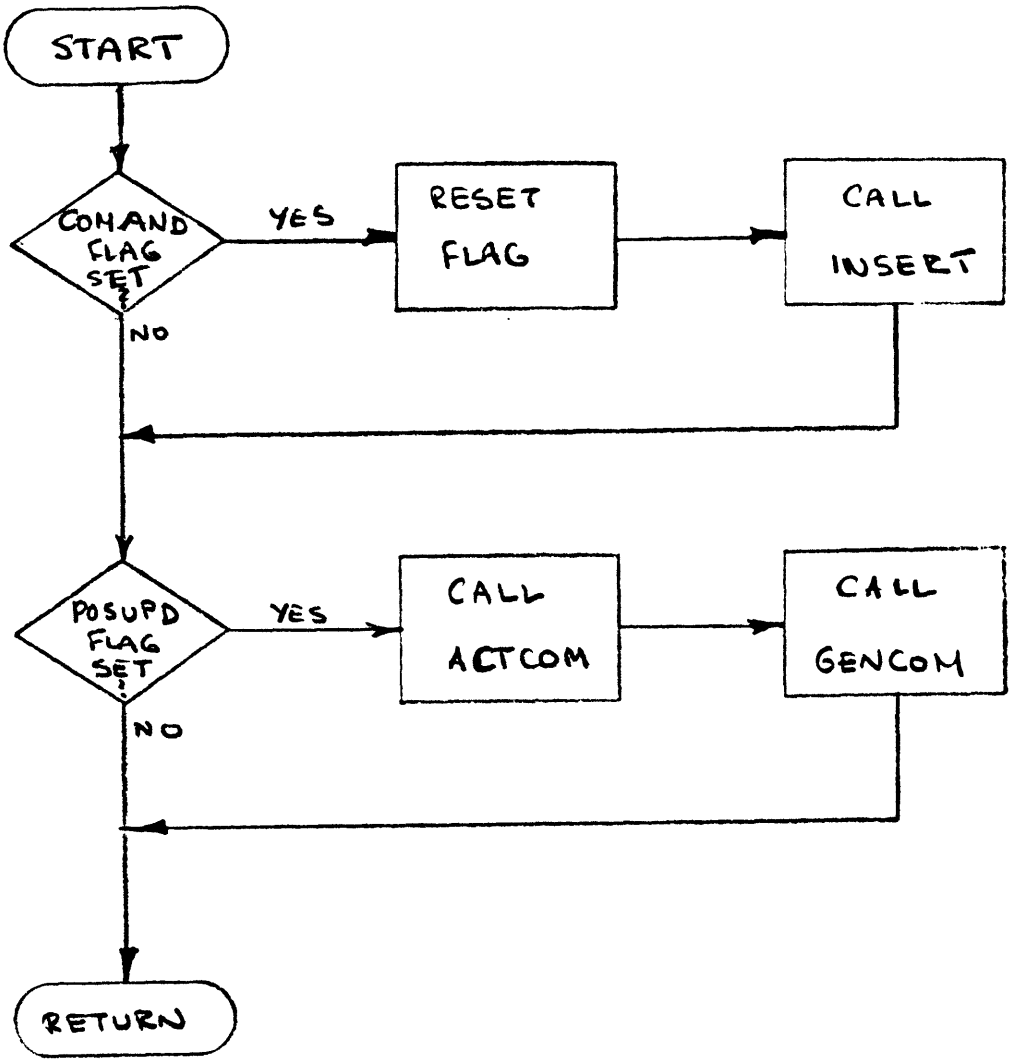

Figure 3.2.2-6

1) Bank angle
2) Vertical speed
3) Longitudinal acceleration

3.2.2.6 Aircraft Dynamic Update (DYNAM)

3.2.2.6.1 Inputs

This function takes as inputs:

1) The commanded bank angle
2) The commanded vertical speed
3) The commanded longitudinal acceleration
4) The current aircraft state, including altitude, latitude and longitude, airspeed and ground speed, heading and ground track angle, vertical speed, longitudinal and lateral accelerations, bank angle, rate of turn, etc.

3.2.2.6.2 Processing

This task updates the true aircraft state based on the above commanded values. The update is performed through a series of calls to the following modules (see figure 3.2.2-7):

1) ALTCOM. This module updates the aircraft's altitude and altitude rate.

2) ENRGRC. This module restricts the achievable values for longitudinal acceleration based on the rate of climb or descent.

3) LATACC. This module computes the aircraft's lateral acceleration based on the bank angle.

4) SPDTRK. This module computes the aircraft's ground speed and track angle.

5) VELHGD. This module updates the aircraft's velocity and heading.

DYNAM
(POSGEN)

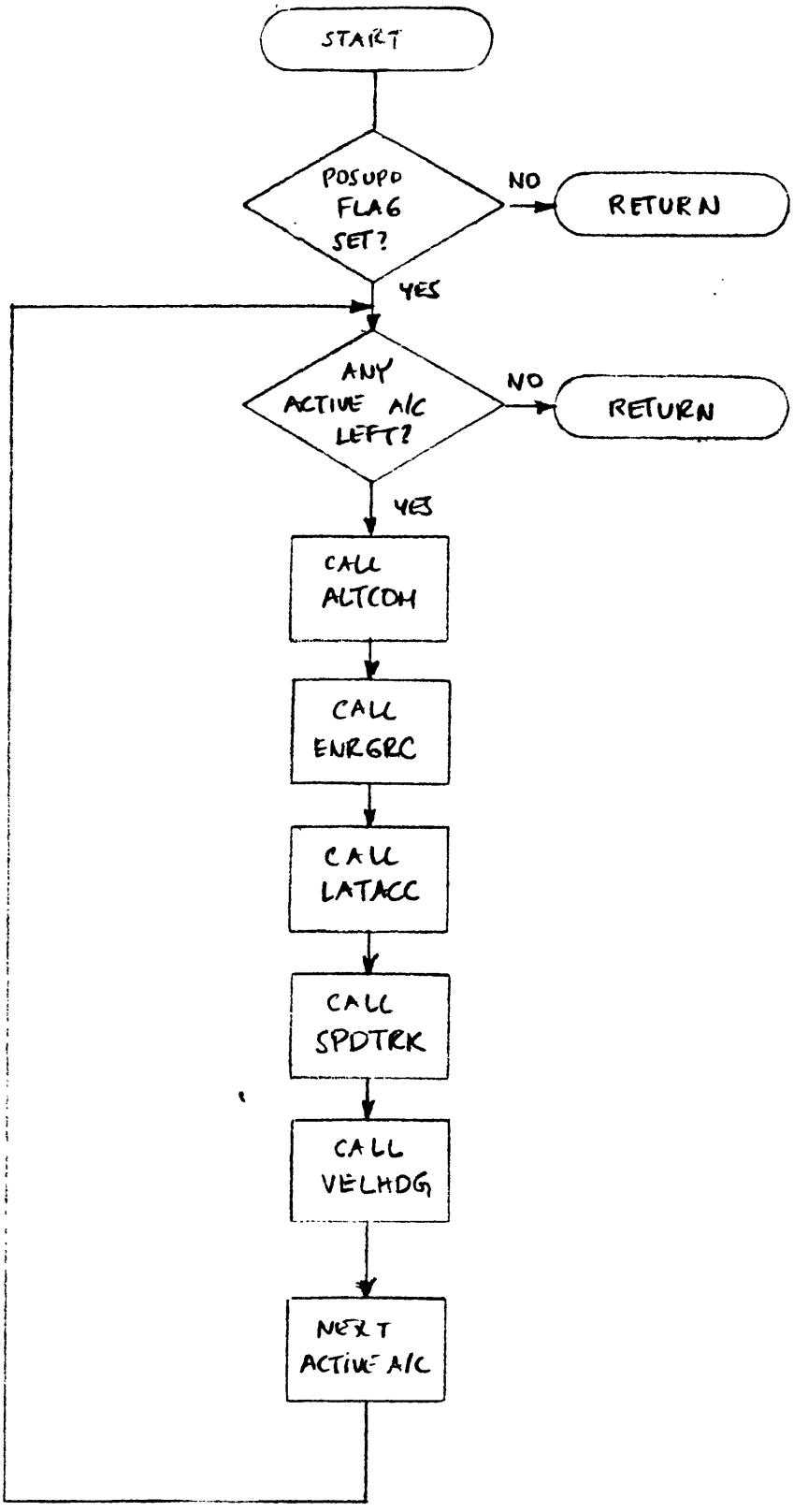

Figure 3.2.2-7

3.2.2.6.3 Outputs

Output from this function is the updated aircraft state.

3.2.2.7 Navigation Equipment Update (NAVUPD)

3.2.2.7.1 Inputs

The input to this function is the true state of all pseudo-aircraft.

3.2.2.7.2 Processing

NAVUPD updates the readings of the aircraft navigational equipment. The functions performed are:

1) Determination of the indicated altitude.

2) Determination of the indicated climb or descent rate.

3) Determination of the indicated and corrected airspeeds.

4) Determination of the indicated Mach number.

5) Determination of the indicated heading.

6) Determination of the radial and range from two VORTAC's.

Equipment errors are modelled. For VOR and DME readings the ground equipment errors are also modelled. The readings generated by this task are used by the command queue management task for pseudo-aircraft navigation.

3.2.2.7.3 Outputs

The output of this function is the indicated state of the aircraft.

3.2.2.8 Surveillance Equipment Update (SURVEY)

3.2.2.8.1 Inputs

SURVEY takes as input the position and altitude of the aircraft as well as the position and characteristics of the radars in the simulated area.

3.2.2.8.2 Processing

This task updates the aircraft positions as measured by the surveillance equipment. Measured positions of for cabs are also generated by this task. It will be generally assumed that a beacon radar system is available for surveillance with aircraft being equipped with altitude encoding (mode C) transponder. The two prevalent radar beacon systems (terminal area and enroute) used today for air traffic control are being modelled. The difference between them is in their range, (60 NM for terminal area and 200 NM for enroute are typical values), their relative accuracy and the sampling rate (4 seconds for terminal area radar versus 10 seconds for enroute radar). Secondary surveillance radars as well as the surveillance function of the new Discrete Address Beacon System (DABS) can also be modelled.

Detailed algorithms for aircraft tracking will not be developed at this stage. Such algorithms may be developed later if their need arises.

3.2.2.8.3 Outputs

This function generates the tracked positions of all the aircraft in the simulation.

3.2.2.9 Aircraft Status Update (ACSTAT)

3.2.2.9.1 Inputs

The current status and position of all the aircraft in the simulation are input to ACSTAT.

3.2.2.9.2 Processing

ACSTAT manages the changes in the status of pseudo-aircraft as they move through the simulated airspace. It is also responsible for signalling to the SCRIPT task of the SIMCON process significant events that relate to aircraft passage through some part of the simulated airspace. The following functions are performed (see figure 3.2.2-8):

1) Activation of dormant aircraft when the time for their activation is reached. Upon activation of a pseudo-aircraft, simulate handoff initiation procedure to the appropriate air traffic controller.(1) Finally, send message to SIMCON since activation of a new aircraft is a permanent significant event.

2) Update aircraft passage information through selected waypoint in the simulated region. For each aircraft the expected time of passage through the next waypoint in its flight plan is generally updated. This information can be used for automatic flight plan selection of randomly generated aircraft.

3) Simulate handoff acceptance of pseudo-aircraft handed off from an active to an inactive ATC sector. The status of such aircraft is not changed but a deactivation time is set. This is done so that the target remains on the radar displays for some time after the handoff.

4) Deactivate pseudo-aircraft when their deactivation time has been reached. At this time check if the pseudo-aircraft is handed off to a sector that will be activated in the future. If so, assign the aircraft to that sector to insure that it is reactivated when the sector is activated, and set their activation time to infinity. If not, delete the aircraft.

5) Reactivate deactivated pseudo-aircraft if the time for their activation has been reached. The INITSC task of the SECTOR process to which the aircraft has been assigned upon their deactivation is responsible for setting the activation time. The position in which the aircraft are reactivated is determined based on their position at the time of their

---

(1) See section 6.3 of this document.

ICSTAT

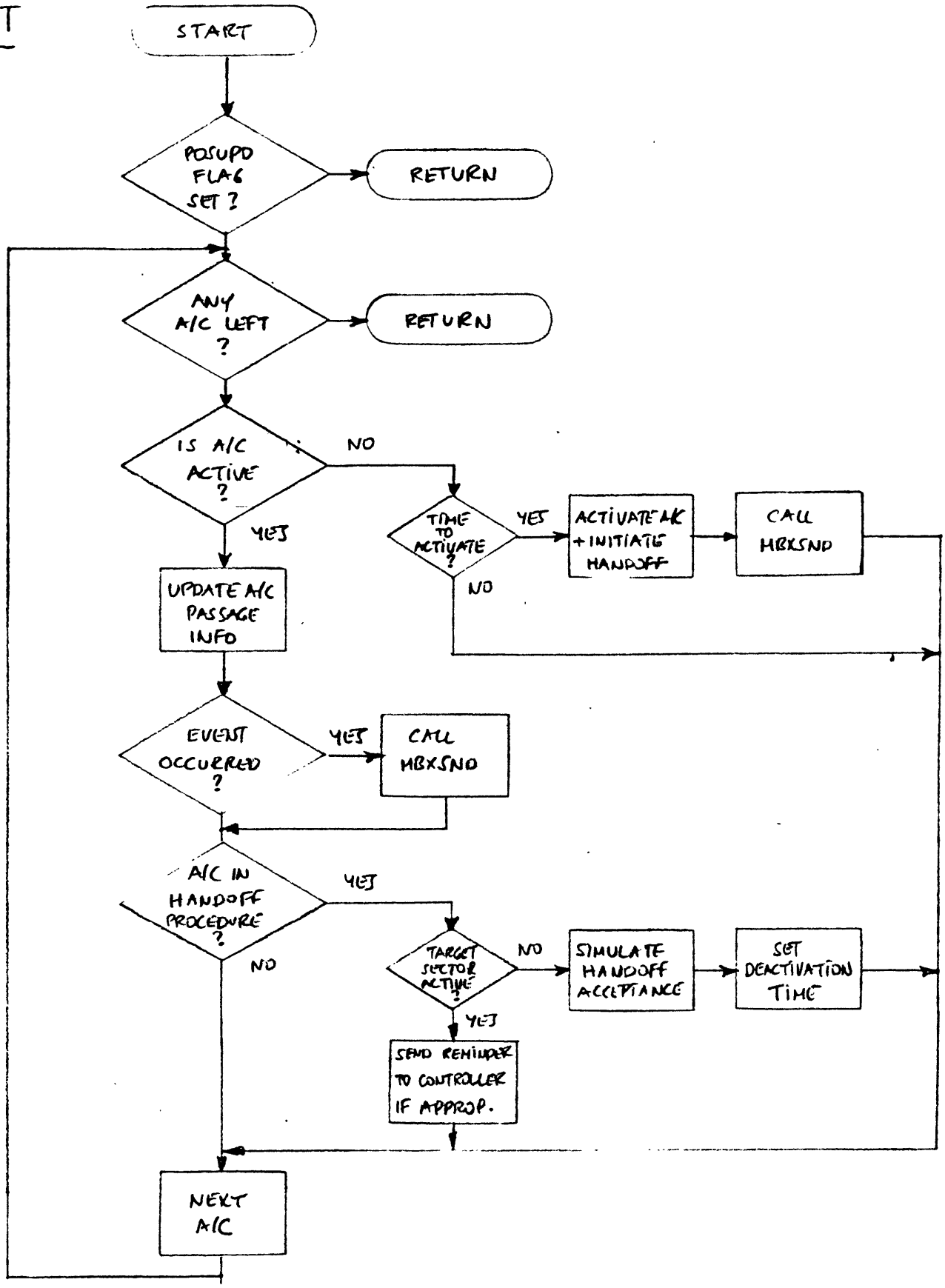

Figure 3.2.2-8

deactivation, and on their flight plan. Finally, send message to SIMCON since activation of a new aircraft is a permanent significant event.

6) Scan the significant event list relating to aircraft passage through particular points in the simulated airspace and send appropriate message to the SIMCON process for each ones that has occurred after deleting them from the list.

3.2.2.9.3 Outputs

The outputs of this function are:

1) The new aircraft status.
2) Signalling of significant events to SIMCON.

3.2.2.10 Weather Update (WINDS)

3.2.2.10.1 Inputs

The inputs to this function are:

1) Probability distributions of wind direction and velocity.
2) Weather reports.

3.2.2.10.2 Processing

This task is responsible for updating weather data. It generates wind direction and velocity, each according to prespecified probability distribution. It also maintains lists of weather reports for various regions. These are dispatched to the air traffic controllers when weather data is requested.

3.2.2.10.3 Outputs

The outputs are:

1) Instantaneous values of wind direction and velocity.
2) Dispatching of weather reports to SECTOR processes.

3.2.2.11 Conformance Monitor (CNFRMC)

3.2.2.11.1 Inputs

The inputs to the conformance monitor are:

1) The current aircraft position altitude and ground speed as measured from the surveillance radar.

2) The flight plans and clearances for the aircraft.

3.2.2.11.2 Processing

This task is responsible for monitoring aircraft and insuring that they are within conformance limits of their flight plans and altitude clearances. Alerts will be generated for two reasons.

1) An aircraft is not on the route specified by the flight plan. This alert will be generated when lateral deviation from the route exceeds some limit. Along the route deviations will not be monitored unless a 3 dimensional flight plan (i.e with time specified at all waypoints) is followed by the advanced cab. Since such alerts are not used in today's ATC system, the air traffic controller will not be informed of such deviations unless the conformance monitor is put in this mode by the experimenter. The experimenter will be able to change the conformance monitoring mode at any time during the run.

2) An aircraft is not at the proper altitude and the altitude return provided by its transponder has not changed for some specified time. Again this alert will not be sent to the controller unless the conformance monitor is in the proper mode.

A third function performed by the conformance monitor relates to the information displayed on the air traffic controller's radar display. Whenever the altitude clearance for an aircraft is not the same (within say 50 feet) of the altitude return provided by the aircraft transponder, both altitudes will be displayed on the aircraft's data block. Otherwise the altitude clearance will only be displayed with a "C" next to it

to indicate that the aircraft is currently conforming to its clearance. Note that this is similar but not exactly the same as the second type of alert above.

3.2.2.11.3 Outputs

This function generates conformance alerts and dispatches them to the air traffic controller whenever aircraft have significantly deviated from their flight plan.

3.2.3 ATC Sector (SECTOR)

The SECTOR process is made up of 5 major tasks: INITX, CMDPRC, MBXDRV, INITSC, and DSPDRV. With the exception of INITX which is executed only once for each run, all SECTOR functions are designed for "parallel" execution. Each is triggered by the occurrence of internal or external events. The process will respond to the following asynchronous events:

1) Reception of a mailbox message. This causes the MBXDRV to be invoked.

2) Reception of a character from the controller station keyboard. This causes the CMDPRC to be invoked.

The overall flowchart for the SECTOR process is shown in figure 3.2.3-1.

3.2.3.1 Initialize Execution (INITX)

3.2.3.1.1 Inputs

The input to this function is the ATC station number with which the process is going to be associated.

3.2.3.1.2 Processing

This function is invoked once in the beginning of the run. Its flowchart is shown in figure 3.2.3-2. It is responsible for mapping to the global data sets (MAPGBL), assigning I/O channels to the mailboxes (ASNMBX), initializing communications with the ATC station (DSPIM), initializing local data, and for setting the appropriate AST routines so that the process is ready to go into the normal execution phase. ASGMBX and DSPIM are responsible for initializing the AST delivery for the mailbox messages and the communications with the ATC station respectively. Prior to exiting INITX calls MBXSND to send a ready message to SIMCON.

SECTOR

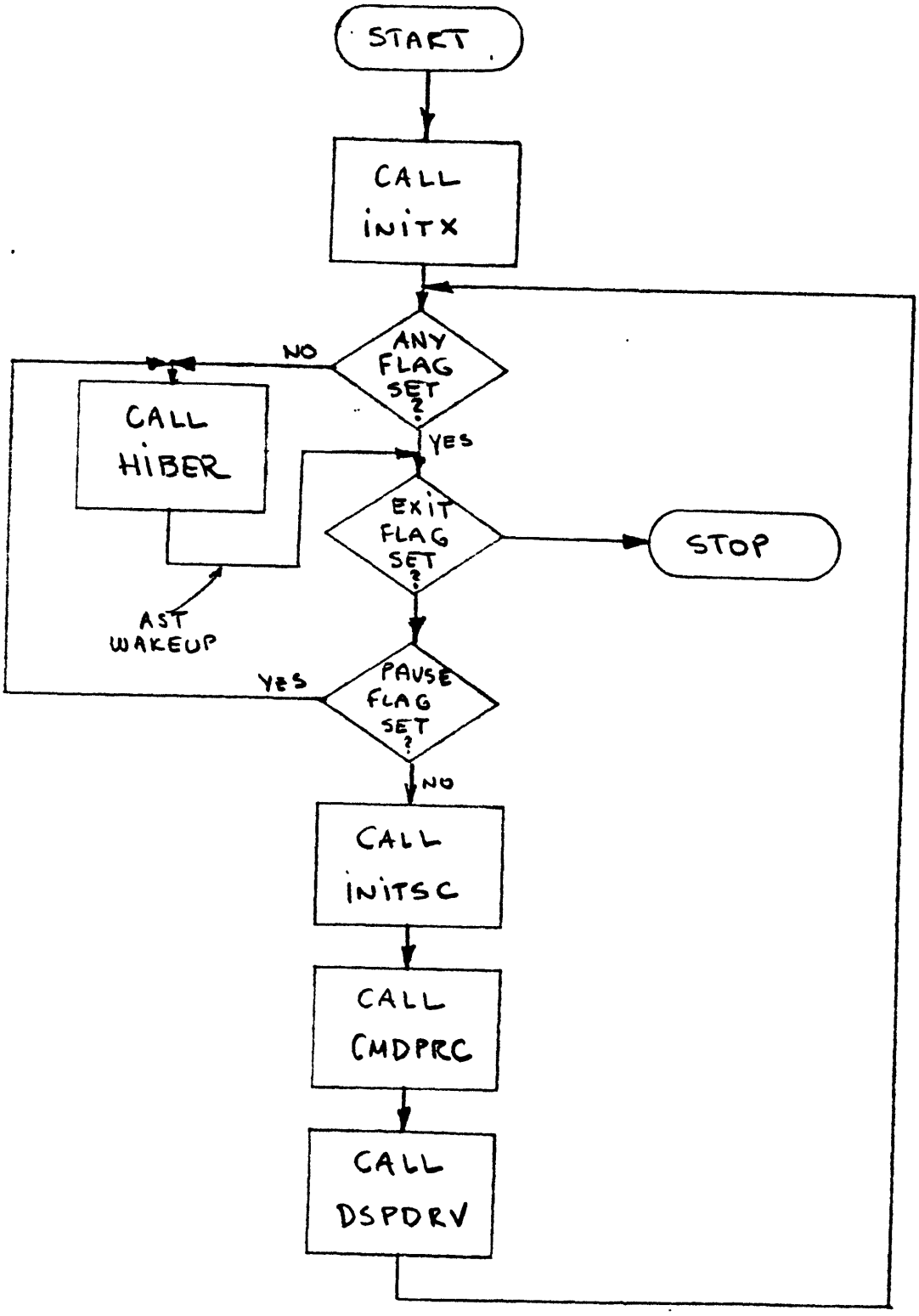

Figure 3.2.3-1

INITX
(SECTOR)

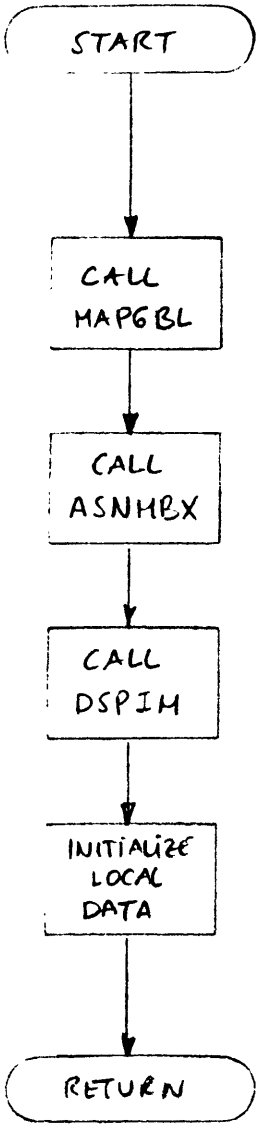

Figure 3.2.3 -2

3.2.3.1.3 Outputs

The ready message sent to SIMCON is the output of this function.

3.2.3.2 Command Processor (CMDPRC)

3.2.3.2.1 Inputs

This function accepts inputs from the ATC station keyboard.

3.2.3.2.2 Processing

The command processor has two distinct subtasks: the Command Editor and the Command interpreter. The Command Editor (shown in figure 3.2.3-3a) is responsible for the character by character processing of the controller inputs. The Command Interpreter (shown in figure 3.2.3-3b) is responsible for parsing and acting upon commands after the character signalling the end of the command line (i.e. the carriage return) has been detected.

The character by character processing will increase somewhat the data transfer load on the VAX-11/750 to Sanders display controller I/O interface. Even so however the transfer rates will be well below the capacity of the hardware. The advantage of this approach is that a variety of editing functions may be performed on the input string thus making the air traffic controller's input task considerably easier. In addition special function keys can be utilized to provide the controller with a set of "quick action" commands which again should prove easier to use. Both these features will be useful when simulation participants playing the role of air traffic controllers are not experienced typists. More importantly the editor will be an indispensable tool for the air traffic controller when the facility is upgraded and is capable of simulating the silent world of the digital data link era.

The Command Editor distinguishes three different types of characters:

1) Self-insert. These are normal input characters that are inserted by the command editor in the input buffer.

COMMAND EDITOR
(SECTOR)

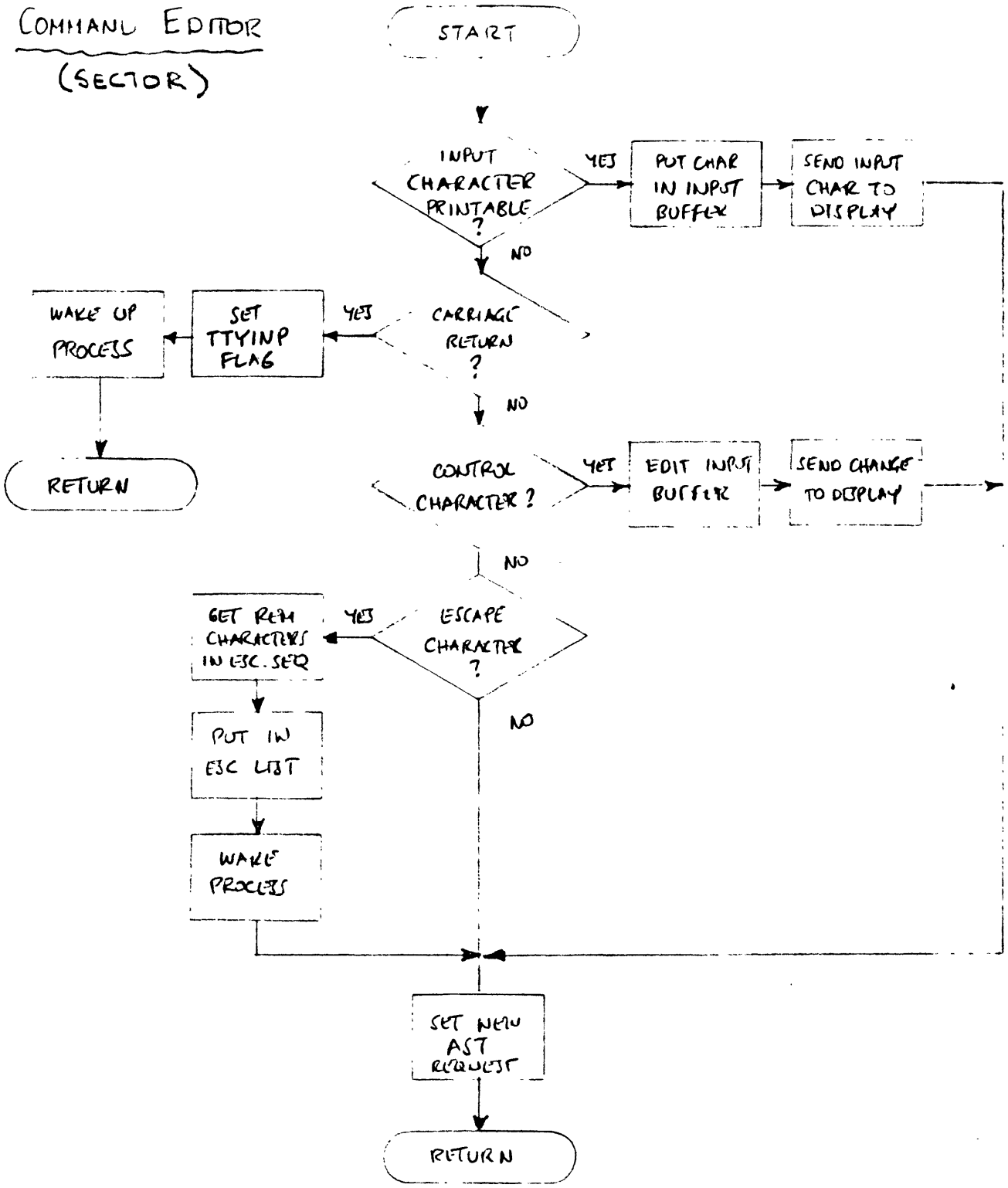

Figure 3.2.3-3a

COMMAND INTERPRETER
(SECTOR)

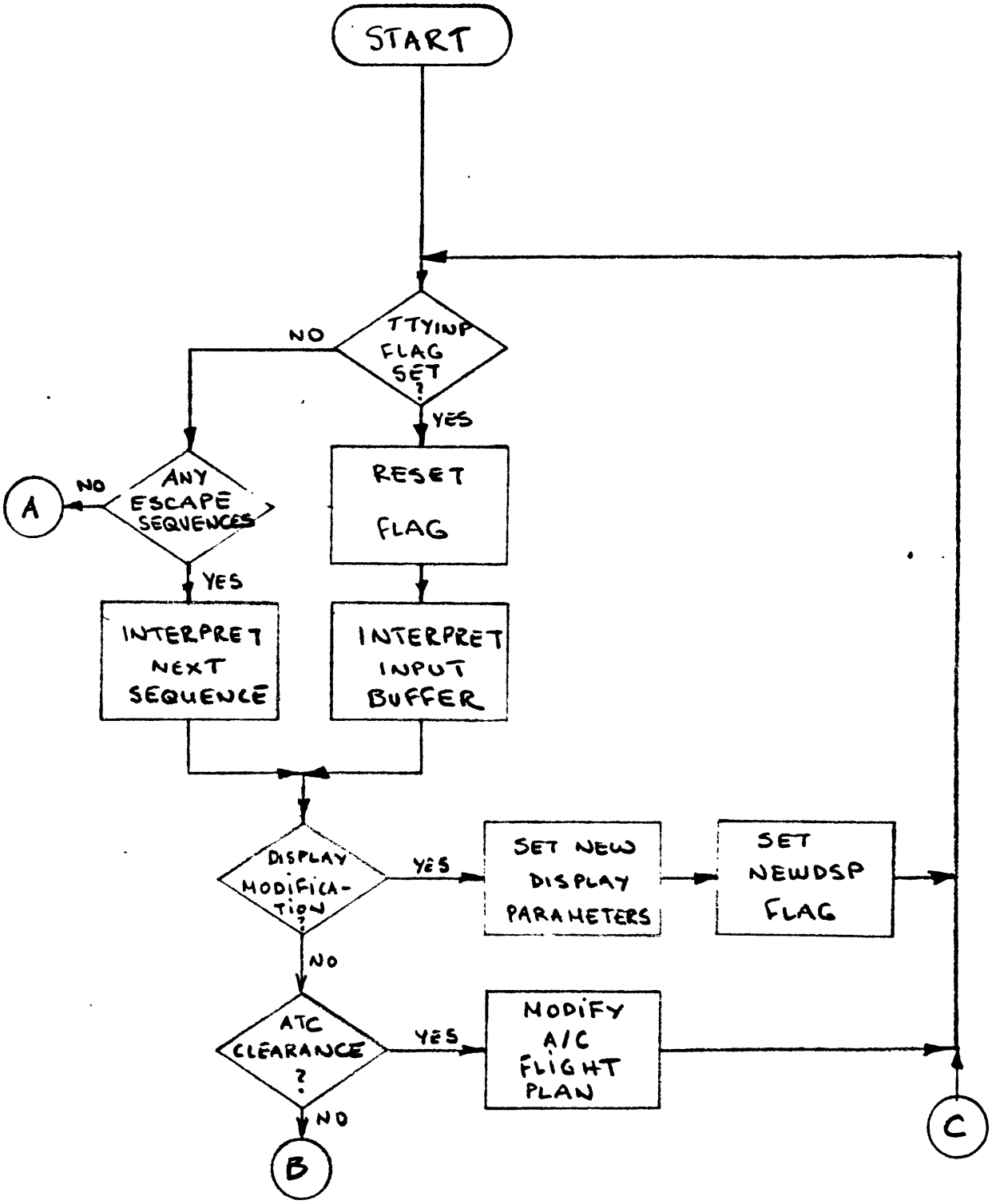

Figure 3.2.3-3b (Continues)

COMMAND INTERPRETER
(SECTOR)

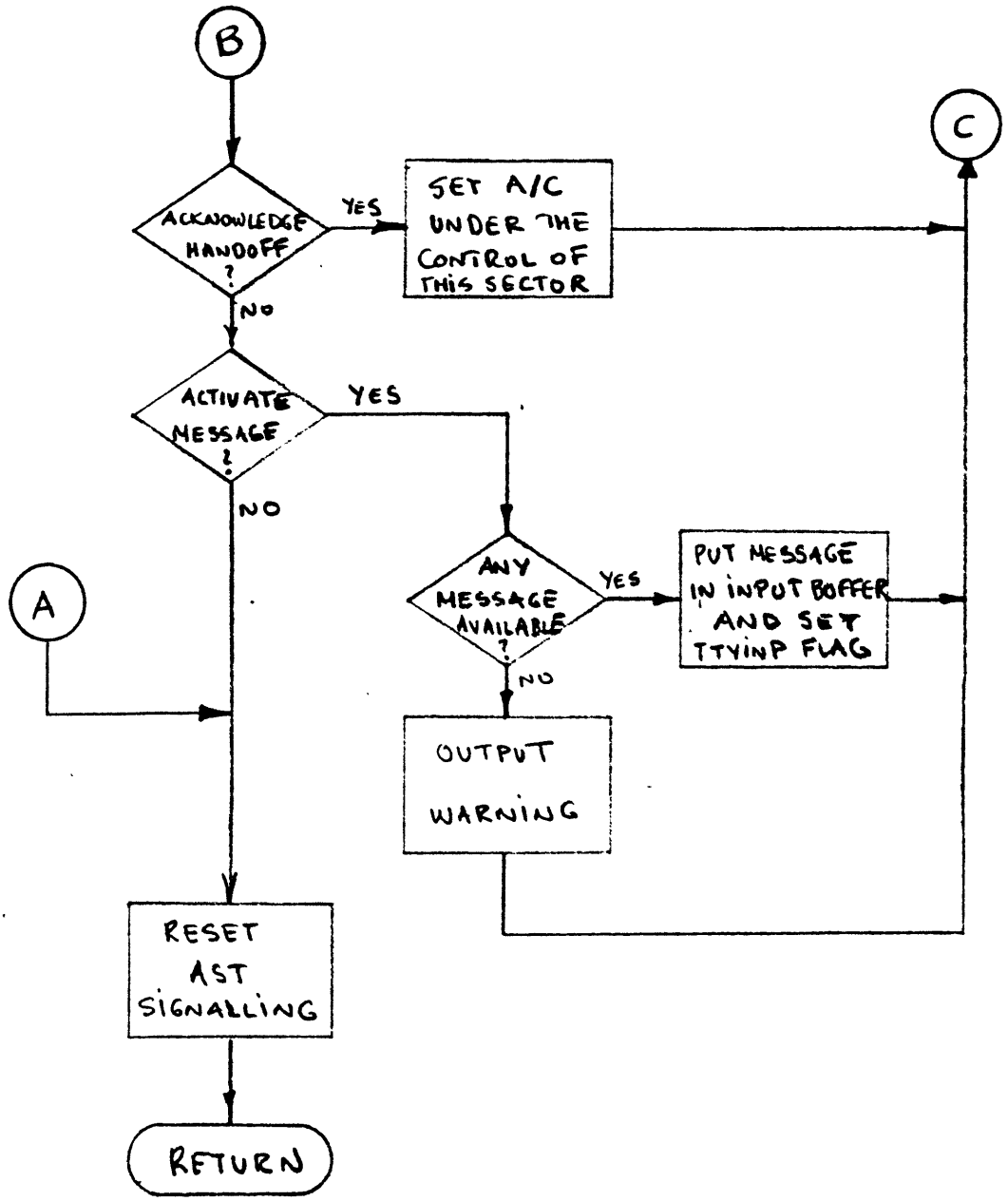

Figure 3.2.3-36 (Concludes)

2) Control Characters. These are non-printable characters that are interpreted as editing commands. Typical editing commands will affect the active "cursor position" allowing insertion and/or deletion of characters in the middle of the input string. Two control characters of particular importance are the carriage return (octal code 015) and the ¬Z (octal code 032). The first signals the end of the command line and cause the transfer of the input buffer to the Command Interpreter for further processing, while the second will cause the flushing of the input buffer effectively cancelling the command.

3) Quick action characters. These are really character sequences of three or more characters that start with a control character, typically the escape character (octal code 033). They are treated like one character however since they are generated by the depression of a single key. These character sequences will be sent to the Command Interpreter immediately independent of the input buffer editing function. This allows quick action commands to be entered and executed when needed even if the air traffic controller is in the midst of entering a normal command.

The Command Editor executes at the AST level. When the carriage return character is detected, the Command Editor wakes the SECTOR process and sets the appropriate flag signalling that input is available for the Command Interpreter which executes in the normal level. While the Command Interpreter is processing the command line, there is no active read request queued to the controller keyboard. Input characters, if any is available during that time, are however queued and each one will cause an AST to be signalled when a new read request is queued to the keyboard. The latter is done by the Command Interpreter when processing of the input line is complete.

3.2.3.2.3 Outputs

A variety of outputs are possible from this function depending on the inputs received.

3.2.3.3 Mailbox Driver (MBXDRV)

3.2.3.3.1 Inputs

The mailbox driver receives messages from other ATC processes. The message format minimally contains the message identification, sending process identification and the message length in the first 8 bytes transmitted (2 bytes are allocated for each item and 2 bytes are reserved for future use). The format of the remainder of the message will depend on the message type.

3.2.3.3.2 Processing

The Mailbox Driver manages the messages that are sent to SECTOR from other ATC subsystem processes. Messages request one of the following (see figure 3.2.2-4):

1) ATC display update. This request is sent by the XTIMER function of the SIMCON process. The mailbox driver responds by setting the NEWDSP flag waking the process and requesting a new AST to be signalled upon reception of a subsequent mailbox message.

2) Cab IDENT button depression signal. The mailbox driver sets the IDENT flag for the corresponding cab. This will be reflected in the aircraft tag displayed on the radar screen next time the DSPDRV is invoked.

3) Script message. The mailbox driver puts the message in the appropriate buffer and sets the NEWMSG flag so that the message is processed by the display driver.

4) Sector activation message. The mailbox driver sets the NEWSCT flag and resets the PAUSE flag to invoke the INITSC function.

5) Sector deactivation message. The mailbox driver sets the PAUSE flag to suspend execution of the SECTOR process.

6) Execution termination. This message is sent by the CMDINT task of the SIMCON process. The EXIT flag is set and the execution will terminate all other pending tasks have completed execution. When this

MBXDRU
(SECTOR)

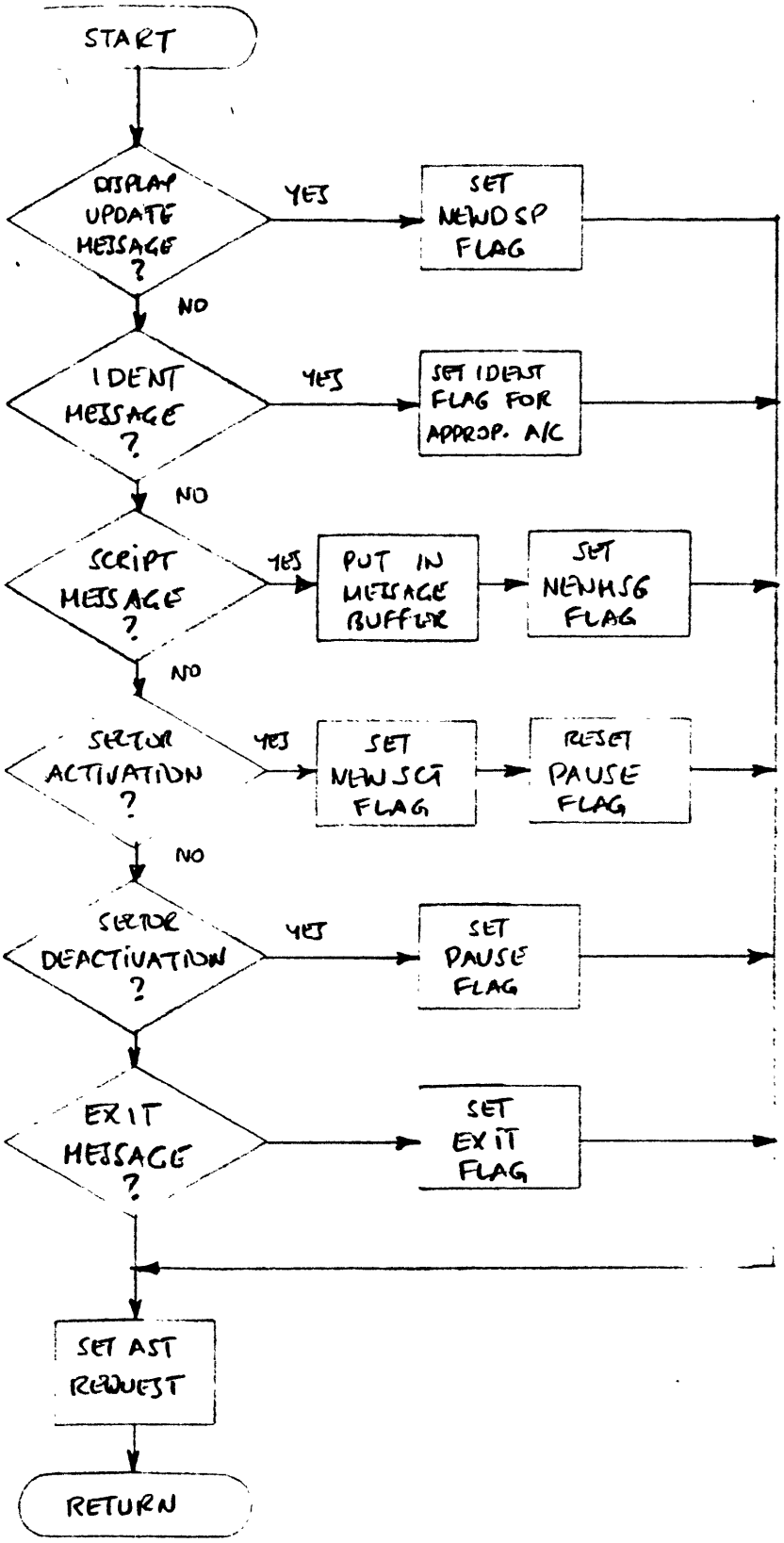

Figure 3.2.3-4

request is received no further mailbox messages are read.

3.2.3.3.3 Outputs

MBXDRV sets appropriate flags when necessary and puts the received messages in the input buffers of various functions that will be invoked for their further processing.

3.2.3.4 Sector Initialization (INITSC)

3.2.3.4.1 Inputs

The inputs to this function are:

1) The process specific data prepared by the CMDINT function of SIMCON.

2) The sector data read in from the sector data file (see section 3.5.1).

3.2.3.4.2 Processing

This function is invoked following a reception of a mailbox message sent by SIMCON requesting the initialization of a new sector. The top level data, including the sector's name and frequency, necessary for the initialization of the sector have already been set by SIMCON in the global data base.

INITSC is responsible for loading all remaining information necessary for the ATC position. This includes sector boundaries, entry and exit fixes, traffic rates associated with entry fixes (if appropriate), location and frequencies of navigation aids associated with the sector, waypoints within the sector. To insure that the newly initialized sector does not require extended period of "warm up" before it reaches a steady state in terms of traffic flow, this function will also be capable of reconstructing a "snapshot" of traffic within the sector taken prior to the initialization of the real time run as part of the script generation procedure. It will be assumed that all the information required by INITSC will have been preprocessed so that the initialization of a new sector will require as little time as possible.

When all sector data is initialized, the list of deactivated pseudo-aircraft is scanned and the activation time for all those assigned to the sector is set to the current simulation time. This will cause the reactivation of those aircraft by the ACSTAT function of the POSGEN process.

Finally the NEWDSP flag is set and a mailbox message is sent to the SIMCON process to advise that the sector has been successfully initialized.

The flowchart for this function is shown in figure 3.2.3-5.

3.2.3.4.3 Outputs

INITSC loads the data base for the new sector, and causes the display to be reset according to the new data.

3.2.3.5 Display Driver (DSPDRV)

3.2.3.5.1 Inputs

The display driver uses the following inputs:

1) The display parameters including the latitude and longitude for the center of the display, the display range, number of past tracker positions to be displayed for each aircraft, number of minutes for projecting future position for each aircraft.

2) The current radar measured position altitude and ground speed for each aircraft.

3) The flight plan and ATC clearances for each aircraft.

4) The waypoint locations and airway structure within the displayed range.

5) Alphanumeric data to be displayed on the radar screen.

INITSC

(SECTOR)

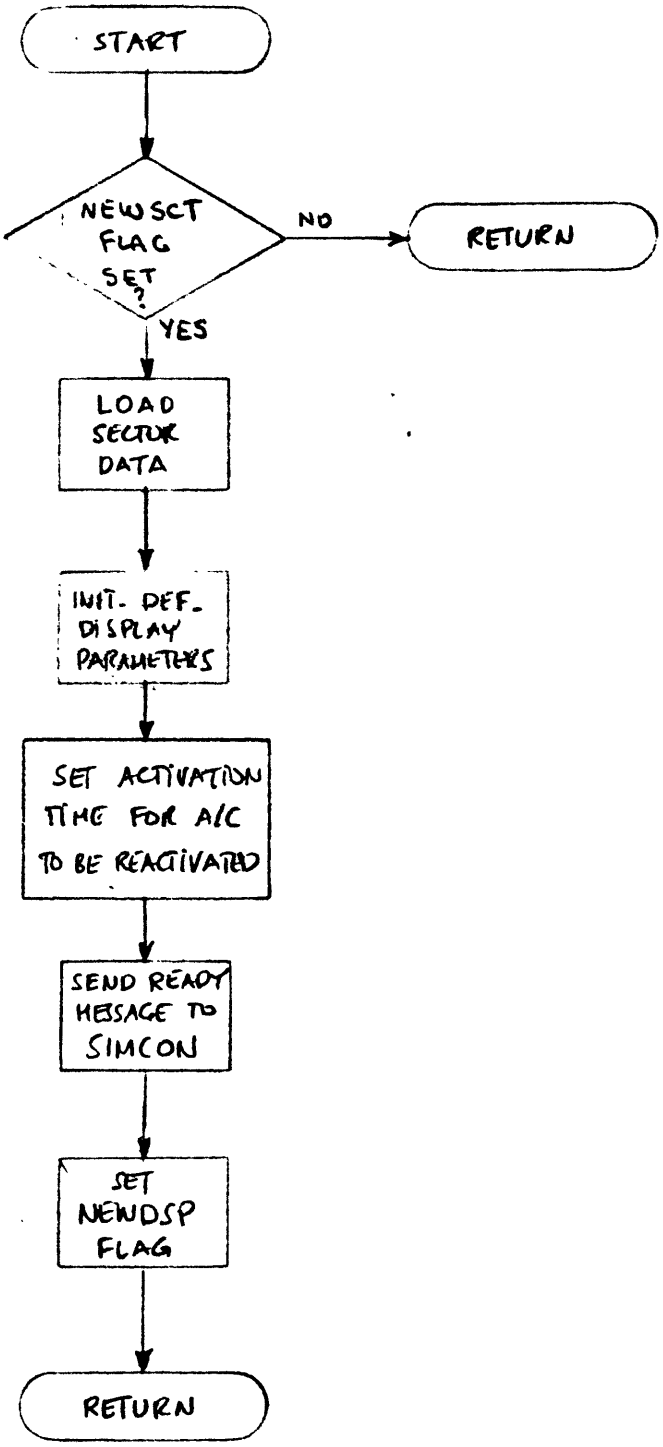

Figure 3.2.3-5

3.2.3.5.2 Processing

The display driver is responsible for updating the information on the controller's display screen. It is composed of three basic subfunctions (see figure 3.2.3-6): INITD, RADARD, and ALPHAD.

INITD is responsible for initializing the display for each sector. It generates all the parts of the display that will not change at all during the life of the sector.

RADARD is responsible for updating the aircraft positions as seen by the surveillance system, and displaying the waypoints and airways in the sector. The position of airways and waypoints change only when the display range or the center of the displayed area changes.

ALPHAD manages the alphanumeric (tabular) data that appears on the screen.

3.2.3.5.3 Outputs

The information displayed on the air traffic controller's screen falls into the following 3 categories:

1) Radar information.
2) Maps.
3) Tabular data.

Radar information includes:

a) The tracked position of the aircraft. Special symbols (TBD) are used to indicate the aircraft position. The actual symbol used for some aircraft will depend on whether the aircraft is uncontrolled, controlled by some other controller or controlled by the controller associated with the display.

b) Aircraft data block. Each aircraft symbol will be accompanied by a data block (possibly empty). The first line of the data block contains the aircraft ID or the transponder code (if the ID is not available), or is blank (if neither of the above is available). The aircraft ID will generally not be available for uncontrolled traffic. The transponder code will not be available for uncontrolled traffic which is not equipped with a transponder. The second line of the data block will contain altitude information. The last altitude clearance will

DSPDRV

(SECTOR)

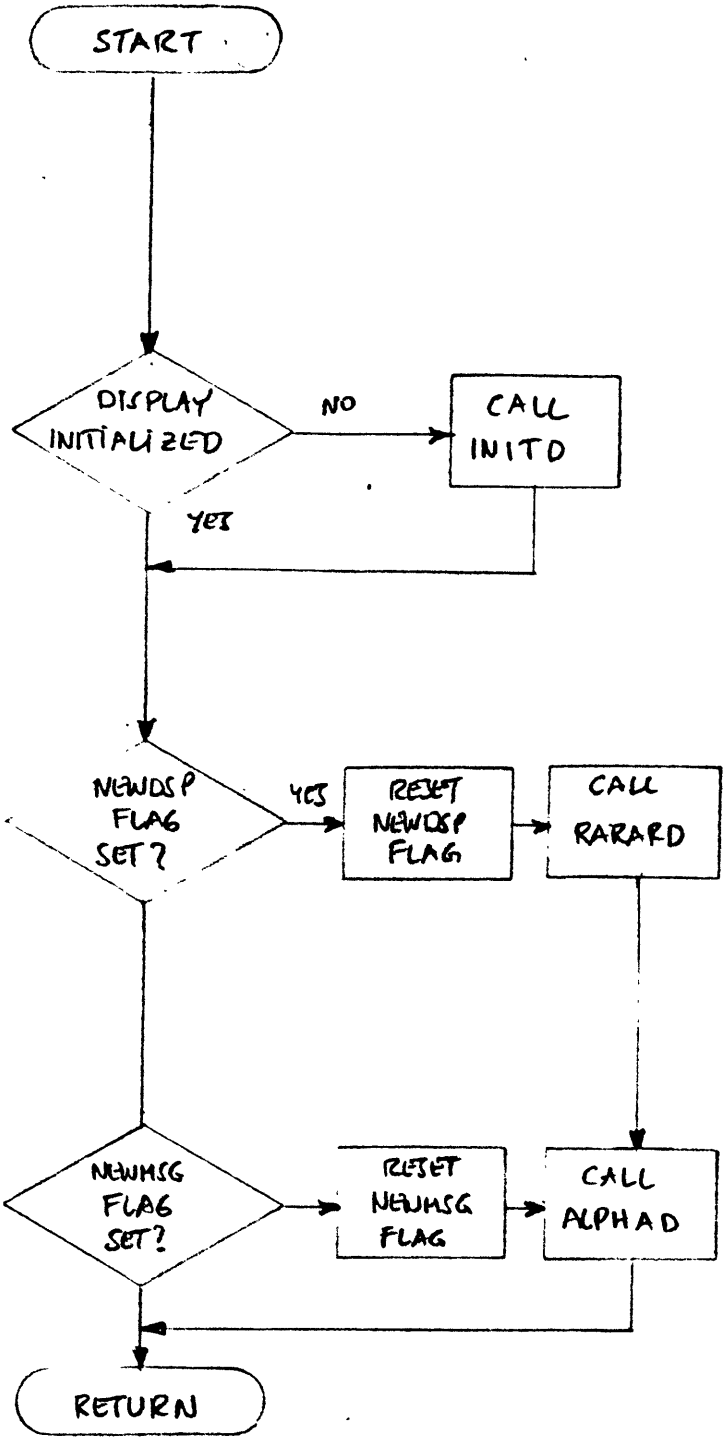

Figure 3.2.3-6

always be displayed (if available). If the current altitude of the aircraft is known and it is not the same as the altitude clearance it will be displayed also. Information on the current aircraft altitude will be available if the aircraft is equipped with mode-C transponder, or by verbal pilot report. In the latter case of course the controller will have to manually enter the actual aircraft altitude. The third line of the data block displays the aircraft ground speed and, if available, any pertinent information on the aircraft status (e.g. when the aircraft is being handed off from one sector to the other).

c) Track history. The position of each aircraft during the last 4 sweeps of the radar will be displayed.

d) Projected position. The future position of each aircraft will be projected based on its current ground speed. The interval used for the projection will be selectable.

Three types of maps can be displayed depending on the ATC position being simulated on the station.

a) Airport runway and taxiway maps will be displayed for ground control sectors. These will include the airport layout(as would be drawn in an architectural drawing) as well as names of all the displayed runways and taxiways.

b) Jetroute maps will be displayed for sectors whose control area does not extend to altitudes below 18,000 feet (FL 180).

c) Airway maps will be displayed for sectors whose control area is below 18,000 feet. Both jetroute and airway maps will include position of navaids and waypoints in the displayed area and their published identifiers (names), the jetroute or airway structure, and lines indicating the sector boundaries as well as the names of all adjacent sectors.

Tabular data will be displayed at the bottom of the screen and will include:

a) Preview area, where controller inputs will be echoed.

b) ATC message area, where script commands and other ATC messages will be displayed.

c) ATC data area, where optional ATC data will be displayed upon controller request.

3.2.4 Pseudo-pilot (PPILOT)

The PPILOT process is made up of 5 major tasks: INITX, CMDPRC, MBXDRV, INITPP, and DSPDRV (see figure 3.2.4-1). With the exception of INITX which is executed only once for each run, all PPILOT tasks are designed for "parallel" execution. Each of these tasks is triggered by the occurrence of internal or external events. The process will respond to the following asynchronous events:

1) Reception of a mailbox message. This causes the MBXDRV task to be invoked.

2) Reception of a character from the pseudo-pilot station keyboard. This causes the CMDPRC to be invoked.

3.2.4.1 Initialize Execution (INITX)

3.2.4.1.1 Inputs

The input to this function is the pseudo-pilot station number with which the process is going to be associated.

3.2.4.1.2 Processing

This function is invoked once in the beginning of the run. Its flowchart is shown in figure 3.2.4-2. It is responsible for mapping to the global data sets (MAPGBL), assigning I/O channels to the mailboxes (ASNMBX), initializing communications with the pseudo-pilot station (DSPIM), initializing local data, and for setting the appropriate AST routines so that the process is ready to go into the normal execution phase. ASGMBX and DSPIM are responsible for initializing the AST delivery for the mailbox messages and the communications with the pseudo-pilot station respectively. Prior to exiting INITX calls MBXSND to send a ready message to SIMCON.

PPILOT

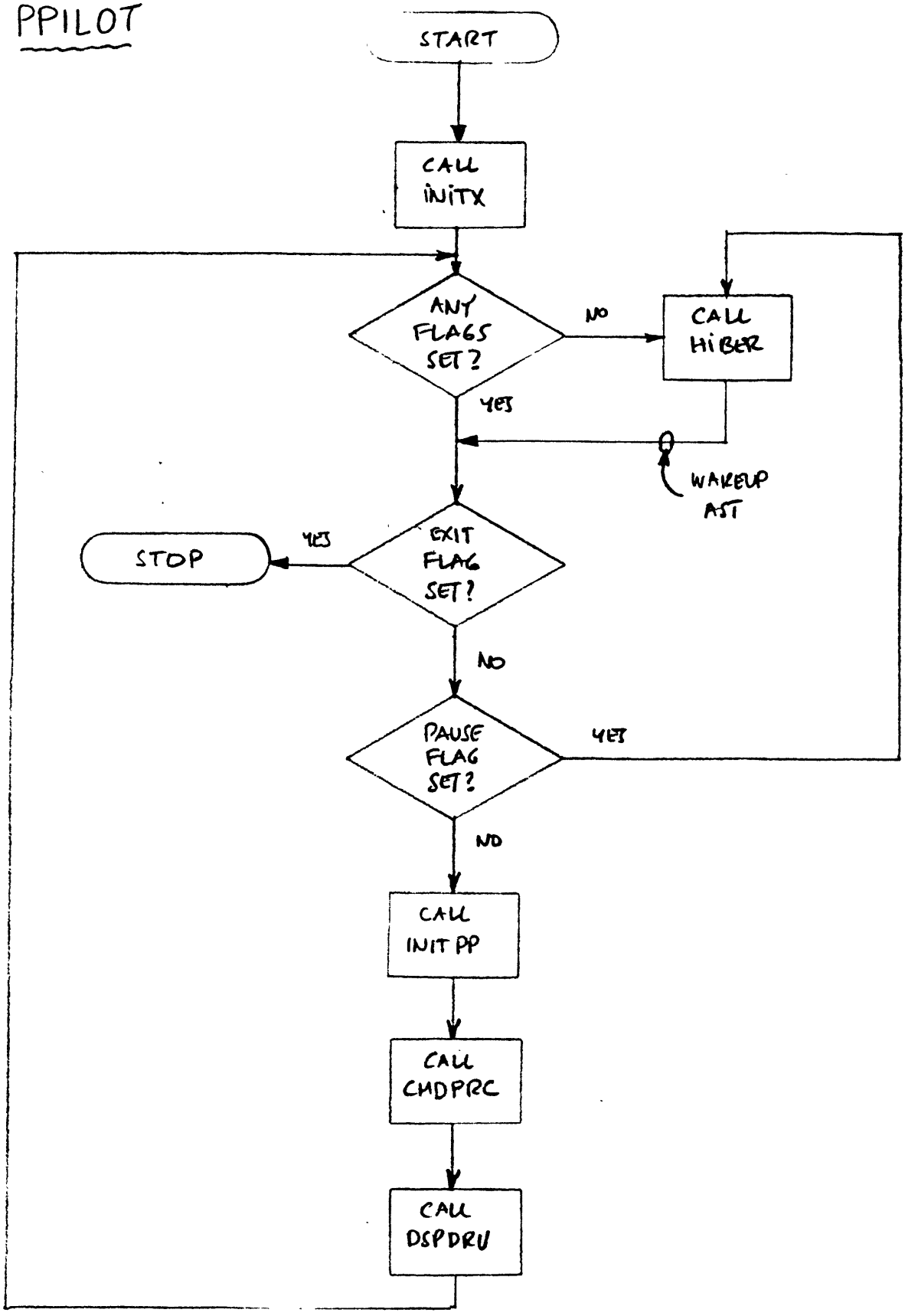

Figure 3.2.4-1

INITX
(PPILOT)

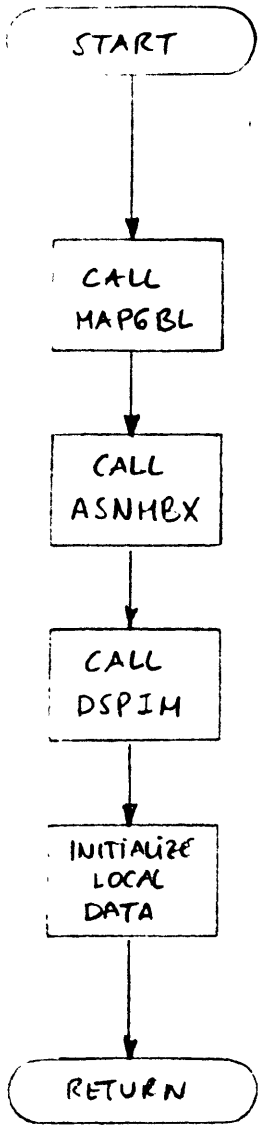

Figure 3.2.4 -2

3.2.4.1.3 Outputs

The ready message sent to SIMCON is the output of this function.

3.2.4.2 Command Processor (CMDPRC)

3.2.4.2.1 Inputs

This function accepts inputs from the pseudo-pilot station keyboard.

3.2.4.2.2 Processing

The command processor has two distinct subtasks: the Command Editor and the Command interpreter. The Command Editor (see figure 3.2.4-3a) is responsible for the character by character processing of the pseudo-pilot inputs. The Command Interpreter (see figure 3.2.4-3b) is responsible for parsing and acting upon commands after the character signalling the end of the command line (i.e. the carriage return) has been detected.

The character by character processing will allow a variety of editing functions to be performed on the input string thus making the pseudo-pilot's input task considerably easier. Special function keys will be utilized to provide the pseudo-pilot with a set of "quick action" commands. Twelve function keys (one for each aircraft) will be reserved to allow the pseudo-pilot quick aircraft identification for the purpose of entering piloting commands without having to enter the aircraft ID. Both these features will be useful when simulation participants playing the role of pseudo-pilots are not experienced typists.

The Command Editor distinguishes three different types of characters:

1) Self-insert. These are normal input characters that are inserted by the command editor in the input buffer.

2) Control Characters. These are non-printable characters that are interpreted as editing commands. Typical editing commands will affect the active "cursor position" allowing insertion and/or deletion

COMMAND EDITOR
(PPILOT)

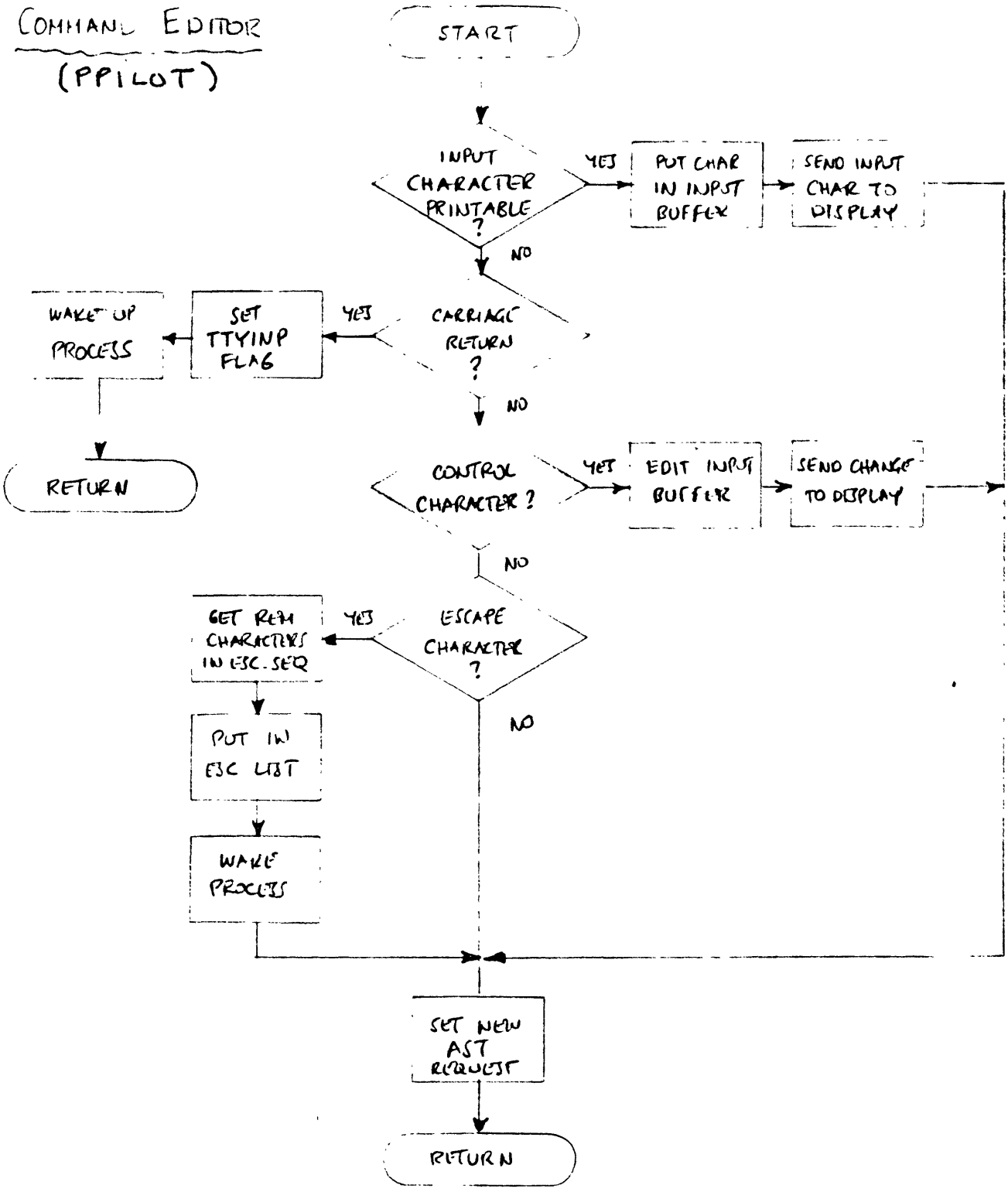

Figure 3.2.4 - 3a

COMMAND INTERPRETER
(PPILOT)

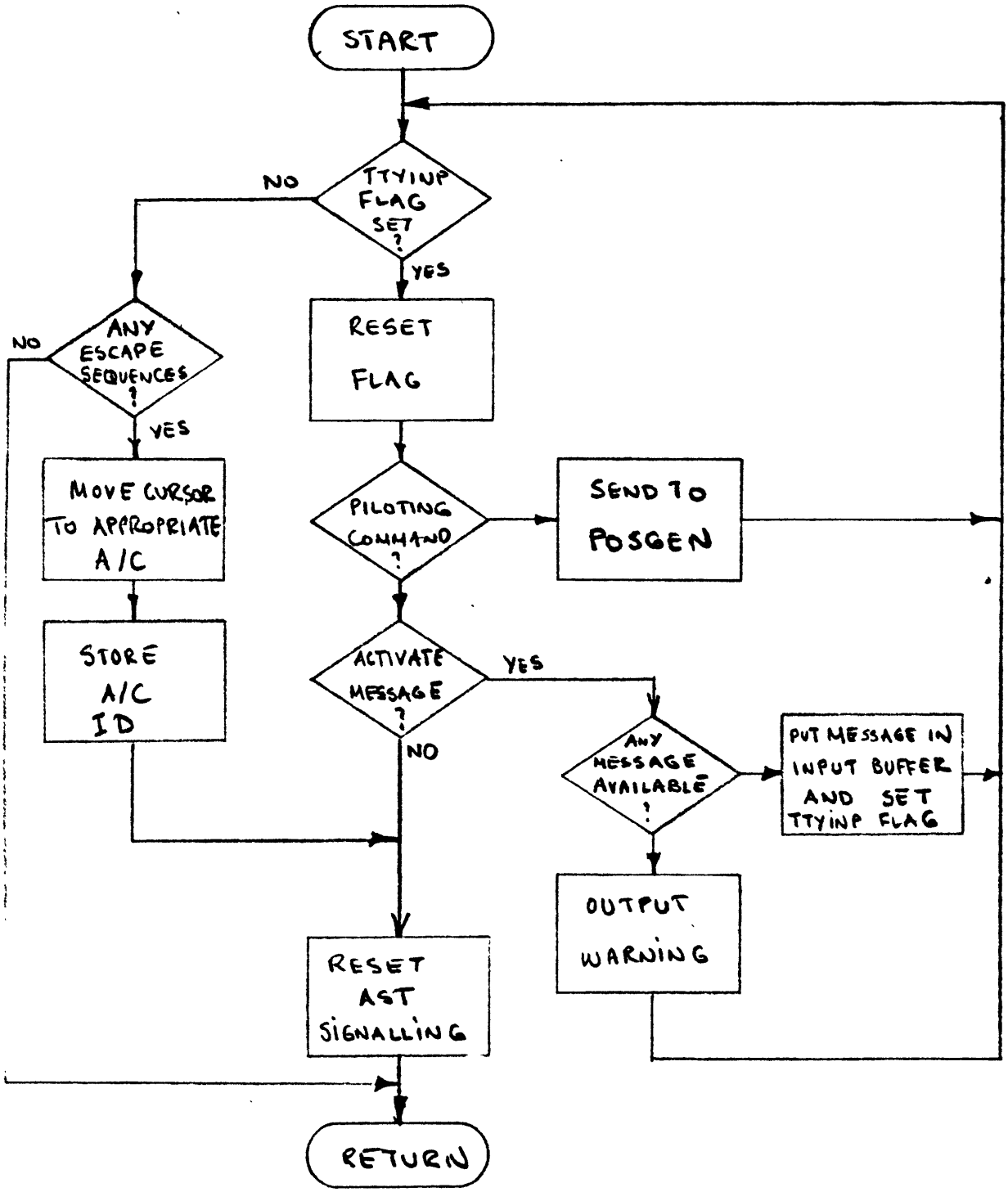

Figure 3.2.4-3b

of characters in the middle of the input string. Two control characters of particular importance are the carriage return (octal code 015) and the ¬Z (octal code 032). The first signal the end of the command line and cause the transfer of the input buffer to the Command Interpreter for further processing, while the second will cause the flushing of the input buffer effectively cancelling the command.

3) Quick action characters. These are really character sequences of three or more characters that start with a control character, typically the escape character (octal code 033). They are treated like one character however since they are generated by the depression of a single key. These character sequences will be sent to the Command Interpreter immediately independent of the input buffer editing function. Quick action commands can thus be entered and executed when needed, even if the pseudo-pilot is in the midst of entering a normal command.

The Command Editor executes at the AST level. When the carriage return character is detected, the Command Editor wakes the PPILOT process and sets the appropriate flag signalling that input is available for the Command Interpreter which executes in the normal level. While the Command Interpreter is processing the command line, there is no active read request queued to the pseudo-pilot keyboard. Input characters, if any is available during that time, are however queued and each one will cause an AST to be signalled when a new read request is queued to the keyboard. The latter is done by the Command Interpreter when processing of the input line is complete.

3.2.4.2.3 Outputs

A variety of outputs can be produced by this function depending on the inputs.

3.2.4.3 Mailbox Driver (MBXDRV)

3.2.4.3.1 Inputs

The mailbox driver receives messages from other ATC processes. The message format minimally contains the message identification, sending process identification and the message length in the first 8 bytes transmitted (2 bytes are allocated for each item and 2 bytes are reserved for future use). The format of the remainder of the message will depend on the message type.

3.2.4.3.2 Processing

The Mailbox Driver manages the messages that are sent to PPILOT from other ATC subsystem processes. Messages request one of the following (see figure 3.2.4-4):

1) Pseudo-pilot display update. This request is sent by the XTIMER function of the SIMCON process. The mailbox driver responds by setting the NEWDSP flag waking the process and requesting a new AST to be signalled upon reception of a subsequent mailbox message.

2) Script message. The mailbox driver puts the message in the appropriate buffer and sets the NEWMSG flag so that the message is processed by the display driver.

3) Pseudo-pilot activation message. The mailbox driver sets the NEWSCT flag and resets the PAUSE flag to invoke the INITPP function.

4) Pseudo-pilot deactivation message. The mailbox driver sets the PAUSE flag to suspend execution of the PPILOT process.

5) Execution termination. This message is sent by the CMDINT task of the SIMCON process. The EXIT flag is set and the execution will terminate all other pending tasks have completed execution. When this request is received no further mailbox messages are read.

MBXDRV
(PPILOT)

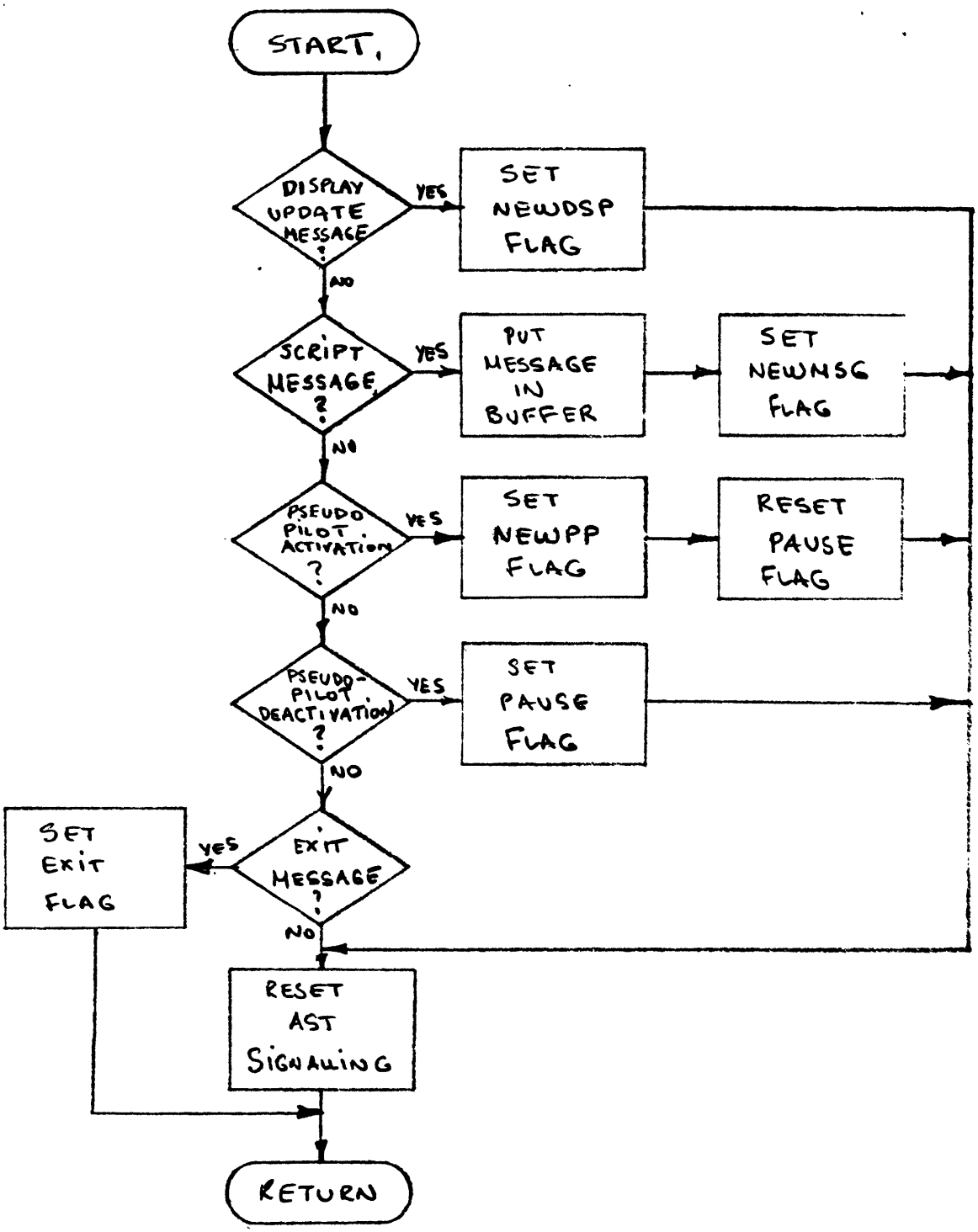

Figure 3.2.4-4

3.2.4.3.3 Outputs

MBXDRV sets appropriate flags when necessary and puts the received messages in the input buffers of various functions that will be invoked for their further processing.

3.2.4.4 Pseudo-pilot Initialization (INITSC)

3.2.4.4.1 Inputs

The inputs to this function are:

1) The PPILOT process specific data prepared by the CMDINT function of SIMCON.

2) The aircraft under the control of the associated ATC sector.

3.2.4.4.2 Processing

This function is invoked following a reception of a mailbox message sent by SIMCON requesting the association of the pseudo-pilot to aircraft in a newly initialized ATC sector. The top level data, including the sector's name and frequency, necessary for the initialization, have already been set by SIMCON in the global data base.

INITPP is responsible for determining which aircraft are controlled by the ATC sector in question, and making the association of these aircraft to the PPILOT process. SCREEN data for these aircraft are also initialized by this function. Finally a mailbox message is sent to the SIMCON process to advise that the process has been successfully initialized.

3.2.4.4.3 Outputs

The pseudo-pilot display database is initialized to reflect the new aircraft under the pseudo-pilot's control authority.

3.2.4.5 Display Driver (DSPDRV)

3.2.4.5.1 Inputs

The display driver takes as inputs the state of all the aircraft under the pseudo-pilot's control authority. It also processes script messages sent to the pseudo-pilot.

3.2.4.5.2 Processing

The display driver is responsible for updating the information on the pseudo-pilot's display screen. It is composed of two subfunctions: INITD, and ALPHAD. INITD is responsible for initializing the display for each pseudo-pilot. It generates all the parts of the display that will not change at all. This includes setting up scrolling regions within the screen. each of which will contain data for one aircraft. ALPHAD is invoked by the SIMCON process through the normal timer loop and is responsible for updating the state information for all the aircraft controlled by the process.

3.2.4.5.3 Outputs

The pseudo-pilot display has 24 lines, each containing 80 characters. The display will be divided into 4 sections:

1) Pseudo-aircraft data area (lines 1-12). This area is further subdivided into 12 "aircraft slots", one for each aircraft under the pseudo-pilot's control authority. In each aircraft slot data pertaining to a single aircraft are displayed. These include:
   a) Aircraft identification,
   b) indicated altitude and altitude clearance,
   c) indicated and corrected airspeed,
   d) indicated heading,
   e) bearing and range from the next waypoint along the aircraft flight plan.

2) ATC message area (lines 14-19). This area will display script messages (usually aircraft commands) addressed to the pseudo-pilot. Up to 6 messages will be visible at any given time. If more messages are available, they can be displayed by scrolling the ATC message area.

3) Preview area (line 21). This area will be used for echoing the pseudo-pilot inputs.

4) Computer response area (lines 22-23). This area will be reserved for messages from the computer regarding the input line. The messages will usually be due to use of illegal or undefined commands, or to illegal command syntax.

3.3 Special Requirements

3.3.1 Expandability

The ATCSIM CPCI is going to be part of an experimental facility. It is therefore expected that a variety of additional capabilities will be needed in the future. Some have been anticpated (e.g. digital data link to the advanced cockpit, addition of more air traffic control and pseudo-pilot stations, etc.). Others, however, are sure to arise depending on the future use patterns of the facility for research and experimentation. It has been a major goal from the very beginning of the software design to maintain the flexibility required in order to allow the system to grow.

To insure correct and easily identifiable program logic, a top-down approach to software development was adopted. Composite design techniques were used to produce a system consisting of highly independent functional parts (in the adopted terminology CPC's functions and CPM's) interconnected through clearly defined interfaces and performing clearly defined functions. This will allow modifying the existing functions, adding functions to a particular CPC, or adding new CPC's, to be done with minimal changes in other software components. As an example, advanced terminal area functions, such as runway scheduling and flight plan generation, could be added in the future as a totally new CPC.

Current memory and CPU availbility on the VAX-11/750 computer will allow a 50% increase in the data and computational requirements before the capabilities of the ATC computer are seriously stressed. On the other hand, the system in its current configuration will be operating very close to the capacity of the Graphic-7 display controller. Further upgrading of the ATC display capabilities will almost certainly require a second Graphic-7 controller.

3.3.2 Portability

Coding of the ATCSM CPCI will be done in standard ANSI77 FORTRAN. This fact should greatly enhance the portability of the software. It is, however, unrealistic to expect any large software system, and especially one designed for real-time operation, to be totally independent of the hardware environment on which it was designed to operate. The most important hardware dependencies of the ATCSIM CPCI are listed in the following section.

In order to maintain the maximum hardware independence, the code which is not transportable will be clearly identified and, whenever possible, isolated. Identification will be in the form of comments in the source code itself. An appropriate note will also be included in the documentation of each hardware dependent module. Isolating hardware dependent code will be achieved by making callable modules out of such code. This will be done whenever the same or similar hardware dependent operations are executed by various software components. The queued I/O operations, used by almost all software components, are a good candidate for isolation.

The final result will be that, even though the CPCI will not be transferable immediatelly to other hardware environments, the required changes will be concentrated in relatively few parts of the code and will be easily identified.

3.3.3 Machine/Hadware Dependencies

The following is a list of hardware dependencies for the ATCSIM CPCI:

1) The software driving the ATC displays will be specific to the SANDERS Graphic-7 display controller.

2) The pseudo-pilot displays will be dependent on the characteristics of the TI-940 video terminal insofar as the cursor positioning and setting of other display characteristics require control sequences that are not part of the existing ANSI standard for video terminals.

3) All queued I/O operations will be dependent on the VAX/VMS operating system.

4) The timing of the simulation will be dependent on the VAX/VMS operating system.

5) The management of shared (global) data will be dependent on the VAX/VMS operating system.

3.4 Human Performance

There are no strict requirements for the responce time or any other aspect of the system operators' performance. Air traffic controller and pseudo-pilot positions can be filled by personnel that has no prior experience with controlling traffic or navigating an aircraft. The level of competency for the operators will, by and large, be reflected by the conversation carried over the audio loops rather than by the operation of the hardware available to them. Generally, the air traffic controllers and the speudo-pilots should be familiar with the available commands and the capabilities provided to them by the system. Most importantly however, they should be familiar with the ATC terminology and the "jargon" used by pilots and controllers over the audio communications channels.

3.5 Data Base Requirements

This section describes the database required for the ATC subsystem. With few exceptions all major data required by the ATC subsystem will reside in global sections which are shared by all ATC subsystem processes. This allows processes to share data without imposing extensive I/O requirements among processes. There is one major requirement imposed by the use of global sections. The data in each global section have to be contiguous and the section should be page aligned. The data is made contiguous by assigning all tables (arrays) in each section to a COMMON (in the FORTRAN meaning of the word) block. By page alignment we mean that the starting address of each common block has to be on a page boundary. In the VAX/VMS operating system a page is a memory unit consisting of 512 bytes. The starting address of all global sections has to be a multiple of 512 (i.e. its last 9 binary digits have to be zero). Each such COMMON block can be page aligned by the VAX Linker during the linking of each ATC subsystem process. Command procedures that will be developed to link each ATC subsystem process will also include the required syntax to achieve the alignment of the global sections.

The global data is divided into groups each containing related information:

1) Process Data contain information required by SIMCON to control and monitor the operation of various process functions as well as data specific to each process that SIMCON must also have access to. The former include process ID, process name, execution priorities (current and default), status and privilege flags, CPU usage statistics, etc. The latter vary depending on the process in question. POSGEN maintains information on the number of aircraft in the system, the current simulation time, random number seeds used by various functions, etc. SECTOR specific data include the sector name, pointers to data defining the sector boundaries, sector audio frequency, etc. PPILOT specific data include the pointer to the sector controlling the pseudo-aircraft represented by the process, etc.

2) Dynamic data consist of aircraft datasets and radar datasets. Radar datasets contain surveillance information provided by each radar in the simulated area. The data include azimuth and range information, as well as tracking information (when a tracker is incorporated in the surveillance model. Aircraft datasets contain pertinent information on

active aircraft in the simulation including: (i) Aircraft ID data such as aircraft type, flight number, airline name, and other such data typically found in the flight strip as well as pointers to the SECTOR and PPILOT processes controlling the aircraft, and pointers to the aircraft's audio disguiser parameters, (ii) Aircraft state data such as position, altitude, airspeed, and similar information describing the aircraft's instantaneous state, (iii) Aircraft command data describing the clearances that the aircraft received from the air traffic controller and the commands the pseudo-pilot has initiated, Aircraft scheduling and flight plan data describe the scheduled time to take-off or land at an airport (if the approach and departure control sectors are active, and the IFR flight plan clearance as received by the aircraft with possible clearance updates that have been made enroute.

3) Static data consist of reference information which describes the simulation environment and remains unchanged throughout the run. This group includes data describing the simulation airspace (control centers, sector boundaries, names, and frequencies, location of airports, navigational aids, and radars, the characteristics of the pseudo-aircraft simulated, traffic levels at various sectors, etc.). Static data is very extensive and by necessity cannot all reside in memory at any given time. Sector data, for example, are read in at sector initialization time and remain in memory while the sector is active. Similar procedures are used for navigational aids and other large datasets.

4) Screen data consist of data especially formatted for display on the PPILOT video screen or alphanumeric data displayed on the air traffic controller's radar screen. This group contains some information found elsewhere in the simulation database. Here however the information is in a format suitable for output. Even though the procedure requires additional computer memory it is adopted since it results in considerable savings in CPU time.

5) Script data include information necessary to activate and control the script. The list of significant events is shared by all ATC processes. Each however has a separate list pointing to the events in the list (if any) it is responsible for monitoring.

Global data are automatically saved at the end of the run by the VMS operating system.

3.5.1 Sources and Types of Data Base Inputs

The following is a list of files that contain data required by the ATC subsystem:

1) PROCESS.DAT — This file contains raw data pertaining to all processes in the ATC subsystem. The data in this file are processed by module PRCIM and its subordinate modules into table PROCESS. This is a fixed length, formatted, readonly file and occupies approximately one block of disk storage.

2) STATIC.DAT — This file contains raw data controlling the processing of the static tables of the ATC subsystem. Its contents are processed by module STAIM into table INCSTA. This is a fixed length, formatted readonly file and occupies one block of disk storage.

3) DYNAMIC.DAT — This file contains raw data controlling the processing of the dynamic tables of the ATC subsystem. Its contents are processed by module DYNIM into table INCDYN. This is a fixed length, formatted readonly file and occupies one block of disk storage.

4) SCRIPT.DAT — This file contains raw data defining the script which will control the ATC subsystem execution. Its contents are processed by SCRIPT and its subordinate modules during the real time run into the event list tables of the process appropriate to monitor each event. Each process in the ATC subsystem has its own event list table. This is a random access read and write file. Its length will vary from one run to the next. The file will be constructed prior to the real time run by the script generation software and can be augmented during the run by script directives entered in real time by the experimenter.

5) NODES.DAT — This file contains raw data defining all the published waypoints in the airspace of interest. Ultimately this file will contain all the published waypoints in the continental

US. This is a formidable task that should be accomplished over a period of several years. The best approach (other than using existing waypoint tables, if such are available) will be to start with a relatively small region (e.g. the state of California) and adding regions as they are needed for specific experiments. This is a fixed length readonly file and will ultimately be very large. At some point off line software should be developed to prepare a smaller file from the master which will contain only waypoints in the region of interest for each particular experiment. The contents of this file are processed by module NODEIM of POSGEN and SECTOR into their respective NODES tables, during the real time run.

6) ATTRIB.DAT — This file contains raw data pertaining to the attributes of the various aircraft types that are used in the simulation. This is a fixed length, formatted, readonly file. It occupies 2 blocks of disk storage. Its contents are processed by module ATTRIM into table ATTRIB.

7) WIND.DAT — This file contains raw data pertaining to the weather conditions in various simulated regions in the simulation. This is a fixed length, formatted, readonly file. It occupies 2 blocks of disk storage. Its contents are processed by module WINDIM into table WIND.

8) ALNAME.DAT — This file contains raw data pertaining to the attributes of the various airlines that are used in the simulation. This is a fixed length, formatted, readonly file. It occupies 2 blocks of disk storage. Its contents are processed by module ALNMIM into table ALNAME.

9) ACDESC.DAT — This file contains raw data associating aircraft makes (e.g. B747) to aircraft types in terms of performance. This is a fixed length, formatted, readonly file. It occupies 2 blocks of disk storage. Its contents are processed by module ACDIM into table ACDESC.

10) DYNAM.DAT — This file contains raw data pertaining to the performance of the various aircraft types that are used in the simulation. This is a fixed length, formatted, readonly file. It occupies 2 blocks of disk storage. Its contents are processed by module ACDYIM into table DYNAM.

11) SOURCE.DAT This file contains raw data describing the sources that generate aircraft in each sector. This data is used only if pseudo-aircraft are generated stochastically. The sources are airports generating arrivals or departures if the sector is a terminal area. For enroute sectors the sources are dummy points and generate through traffic. This is a fixed record length, formatted, readonly file. Its length will depend on the number of sectors that will be simulated in a single run. Sources appropriate to each sector are activated upon sector initialization. This is done by module SRCIM. Active sources are put in table SOURCE.

12) FIXES.DAT This file contains raw data describing the fixes at which new pseudo-aircraft are generated. The data in this file are processed by SRCIM along with source data and are stored in table FIXES.

13) NAVDIM.DAT This file contains raw data describing the navigational aids (VORTAC's) in the simulated regions. This is a fixed length, formatted, readonly file. The comments for the NODES.DAT (see item 5 above) apply to this file also. The data in this file are processed by module NAVDIM into table NAVAID.

14) RADAR.DAT This file contains raw data describing surveillance radars in the simulated regions. This is a fixed length, formatted, readonly file. The comments for the NODES.DAT (see item 5 above) apply to this file also. The data in this file are processed by module SSDIM into table RADAR.

15) ACSEP.DAT This file contains raw data describing the required separations between aircraft in the simulation. This is a fixed length, formatted, readonly file. The data in this file are processed by module ACSRIM into tables AIRAIR, ARRARR, ARRDEP, DEPARR and DEPDEP.

13) SECTOR.DAT This file contains raw data describing the various sectors that are going to be simulated during a run. This is a fixed length, formatted, readonly file. The comments for the NODES.DAT (see item 5 above) apply to this file also. The data in this file are processed by function INITSC of the SECTOR process.

3.5.2 Internal Tables and Parameters

The detailed variable and parameter definitions and their storage allocation are included in the program listings which have been supplied.

3.6 Externally Developed Software

The ATCSIM CPCI requires only the standard VAX/VMS software provided by the vendor with the VAX-11/750 computer.

4.0 QUALITY ASSURANCE PROVISIONS

TBD