MTB - 052

Multics Technical Bulletin

TO: Distribution

FROM: R.W.Franklin

SUBJECT: NPS/355 as the front-end processor for Multics

DATE: February 27, 1974

This bulletin describes the effort currently underway to interface NPS/355 to Multics. It starts with background information on the reasons for the initiation of the feasibility study and continues up to the current status of the preliminary implementation.

CONTENTS

1.0 General NPS background

NPS/355 is the Network Processing Supervisor for the dn355. It is a software program which operates in the dn355 and which controls devices which may be connected to the dn355.

Generally speaking NPS/355 is a merging of the features provided in GRTS/355 (including its SR F/8 enhancements) and the features contained in the Special Product Message Switching System which was delivered to the State of New Hampshire in April 1972.

The purpose of NPS/355 is to support Series 6000 information networking. This requires that NPS/355 support all of the remote processing modes of multi-dimensional GCOS. NPS/355 is intended to be utilized both as a free-standing message switching system and as a direct replacement for the dn355 GRTS software in a GCOS environment.

A significant feature of NPS/355 is that it makes all processing dimensions available to all types of terminals (except where the site chooses not to allow certain terminals to access certain dimensions). Each site has complete flexibility to assign terminals to dimensions as long as terminal characteristics do not make the proposed connection physically impossible.

The specific remote processing dimensions supported by NPS are:

1. Remote Job Entry

2. Transaction Processing
3. Time-Sharing
4. Direct Program Access
5. Message Switching

2.0 Consideration of NPS as a front-end processor for Multics

With the preceding information available the obvious question arose. Was it feasible to also use this standard product offering as a replacement for the dn355 software currently used to front-end Multics? if so, then many advantages to Honeywell would automatically accrue. By the same token, the disadvantages would also have to be weighed and so an NPS feasibility study was undertaken. The Phase 0 goals of this feasibility study are:

-To determine mutual impact of merging Multics and NPS

-To evaluate the overall functional and economical advantages in comparison with the existing dn355 implementation

-To provide a preliminary design of changes in the NPS and in the Multics side and a plan for implementation.

Phase 0 is to be followed by a Phase 1 which is to provide a Multics environment using NPS as the dn355 software package.

Some other factors involved in considering NPS/355 as the front-end processor for Multics were:

-To have a common front-end processor for all 6XXX systems.

-to be able to maintain the Multics dn355 software in the standard methods utilized for other PCO maintained offerings. i.e. standard releases, changes, Quality Assurance, etc. through which NPS already cycles.

-Standardized Test and Diagnostic routines are offered on-line under an NPS/355 environment.

-An immediate community of knowledgeable programmers at PCO (approximately 20) already know NPS/355.

-A wealth of documentation, training manuals, etc. are already developed and published.

-Many courses have already been held and may hundreds of in-house and customer personnel have attended these.

-To attempt to maintain a common computer-to-computer interface between the dn355 and the 6XXX lines.

-Many capabilities are available under NPS which can be made available to the Multics user community. These include:

-Message Switching

-Sysout capabilities

-Remote Batch

-Remote Job Entry

-etc..

-NPS/355 is planned to front-end multiple 6XXX's and it is not infeasible for one NPS/355 to front-end a Multics 6180 and a GCOS 6080.

-Utilization of NPS/355 as a common front-end processor greatly simplifies the VMM implementation.

-Allows for an easy transition to a completely terminal independent interface to its host computers

-It is very modular and easy to change; especially in its device handlers. These are micro-op coded and it is thus easy to add new devices without affecting the host computer.

Below are some other factors which were also considered.

-Nps/355 requires at least a part of a disk drive. This adds some cost to a Multics configuration which may or may not be significant (a dn355 PSIC plus cables plus use of a drive). For the preliminary implementation which is currently underway, NPS/355 has been assigned one DS190B disk drive. It accesses this through a PSIC on the dn355 IOM which connects to the second side of a link adapter pair(PSA) on the Multics MPC.

-There may be some fairly extensive NPS/355 changes required to make available all of the current Multics terminal capabilities (see Section 7.0).

3.0 Plan of Attack

With the Phase 0 goals in mind the following plan was begun.

-Provide a preliminary implementation where NPS is connected to Multics in a running environment. This preliminary implementation involves changes to both NPS and Multics. It can be used to:

-demonstrate that the merging is/is not feasible

-provide a preliminary design of changes in both the NPS and Multics sides together with a plan for its final implementation.

-Not all Multics functions are necessarily contained in this preliminary implementation. Only those necessary to demonstrate the feasible of the merging are required.

-An evaluation of both the functional and economical advantages/disadvantages of NPS versus the current dn355 implementation was initiated.

4.0 General Rules followed in interfacing NPS/355 to Multics

The general rules used to connect NPS to Multics are as follows:

-All line control functions are performed in NPS (i.e. tty_ctl ceases to exist within the Multics ttydim).

-The GRTS/NPS interface is accepted as the computer to computer interface (see Appendix A).

-The current Multics ttydim flow, as shown in Figure 1 (with minor changes), is generally adopted as the Multics ttydim flow using NPS.

-The character-set/terminal-dependencies used in tty_read/tty_write remain the same for the initial implementation of NPS/Multics.

-No routines in the 6180, outside of the Multics ttydim routines, are to be changed.

-The changes required to the ttydim are transparent to the Multics user.

5.0 General Information

Figure 1 shows the Multics ttydim flow as it currently exists. This flow is based on the dn355 front-end processor acting as a GIOC simulator (i.e. all terminal dependent characteristics are, in effect, defined in the table flow of tty_ctl; all statuses are defined in GIOC format; all commands are defined in GIOC format; etc.,).

Appendix F is a brief writeup depicting how dn355 interrupt handling/mailbox processing works in the current Multics environment. The reader not familiar with how this works should read this appendix.

Figure 2a defines the current Multics mailbox formats while figure 2b defines the current NPS mailbox formats. It should be noted that figure 2b has been modified from basic NPS formats so that while it still adheres to NPS requirements it has been extended to contain required Multics fields. In a similar vein figures 3a and 3b reflect segment dn355_data$ as it currently exists under Multics and then as modified to be used with NPS.

Figure 4 shows the complete list of GRTS/NPS opcodes which may be sent from computer to computer. Only certain of these opcodes are utilized in connecting NPS to Multics. As needs/desires etc., dictate, then the other opcodes will be used.

Appendix B gives a brief explanation of the command codes (within a

mailbox) which are sent from computer to computer. As with the opcodes only certain of these commands codes are utilized in connecting NPS to Multics.

Appendix A is taken directly from the DIA EPS. It repeats some of the information shown in Figures 2a, 4 and Appendix B but is very valuable in showing how GRTS/NPS presents their interface. The sample sequence therein is well-worth traversing.

6.0 Changes required - Multics

Figure 5 is a list of Multics ttydim routines showing which ones have required modification to date and the extent of the modifications required.

As shown in Figure 5, routines dn355 and tty_inter required major changes. Appendix C is a general flowchart of tty_inter as modified to accept the GRTS/NPS computer to computer interface. It should be noted that the main control of tty_inter, the poll, the time-out, the queued status flow, and the basic external entries of write, write_abort, etc., remain basically the same and hence the metering remains basically the same.

As was mentioned earlier, tty_ctl ceases to exist and tty_inter is no longer table driven. The dn355 routine has been modified to use the new dn355_mailbox$ and dn355_data$ formats. In addition the cur_status, get_status, and stop_channel routines within dn355 have been deleted.

In addition to the format changes required to dn355_mailbox$ (figure 2) and dn355_data$ (figures 3) there have been some minor changes made to tty_buf$ and to the input and output formats. The latter 2 are described in Appendices D and E.

Routines tty_read and tty_write have been modified to work with the changed input and output formats in tty_buf$ and routine tty_free has been modified to get/free these new formats.

NPS/355 requires two handshakes during a connection to the 6000. One handshake lets each system know that the other is initialized and a later handshake lets NPS/355 know that the 6000 is ready to accept calls (ACALL) from NPS/355. For the purposes of Phase 0 implementation the initialization handshake is performed followed immediately by the ACALL handshake.

7.0 Changes required - NPS/355

The preceding was an overview of the changes made to the Multics ttydim to interface it to NPS. There are also changes required to NPS and I have broken these down into three categories.

-Addition of new device interface modules.

-Functions other than device interface functions which are not currently in NPS/355 but which must be performed there.

-Functions currently existing in Multics but not in NPS/355 and which are contrary to NPS/355 philosophy.

7.1 Category 1 changes to NPS/355

The addition of device handlers to NPS/355 to handle the identical devices that Multics currently handles.

7.1.1 ARDS

The ARDS graphic terminal must be given a device handler within NPS/355. This will be one of the most difficult device handlers to add to NPS/355.

7.1.2 IBM1050

The IBM 1050 terminal is not handled at the present time by NPS/355

7.1.3 MIT2741

The IBM 2741 is currently handled by NPS/355 but not the MIT2741. It will have to be added as a device handler.

7.1.4 TTY37

The TTY37 is currently handled by Multics and must be handled by NPS/355.

7.1.5 TTY38

The TTY38 is currently handled by Multics and must be handled by NPS/355.

7.2 Category 2 changes to NPS/355

7.2.1 There must be a correspondence between the logical line number assigned to an NPS/355 time-sharing line and the devx code used by Multics. A change is being put into NPS/355 (requested by the Wimmix project) which passes along the:

-line adaptor number

-dn355 number

-dn355 IOM number

-HSLA/LSLA indicator

-subchannel number

with the "Accept New Terminal" opcode. I am using this to build a logical/devx correspondence table within the 6180. This Wimmix requested change must be standardized into NPS/355.

7.2.2 NPS/355 currently uses one DCW for the transfer of input or output to/from the dn355/6XXX. I have requested (and had approved) an NPS/355 change to add the capability of scatter read/write utilizing LPW's. This must be implemented in NPS/355 at least for phase-1 processing of output to a terminal.

7.3 Category 3 changes to NPS/355

7.3.1 GRTS/NPS currently operates in a half duplex mode. In order to match the current Multics capability of type ahead (where the terminal is always in the input mode unless it is physically outputting) certain changes must be made to NPS/355. These changes revolve around aborting an input mode with an output request. In the same manner a Multics user is allowed to abort already initiated output under program control. This involves a change in NPS/355 similar to the one mentioned above.

7.3.2 The carriage return echo currently available in Multics is not available in NPS/355. It must be implemented there.

8.0 Current Status

-all of the changes required of the Multics ttydim routines (as shown in Figure 5) have been made, assembled, and debugging is now in progress.

-A running version of NPS, modified to:

- -use only 1 disk
- -use the DIA on channel 4 with the 6180 mailbox at 600
- -not require the ACALL
- -define the dial-in lines as time-sharing lines on receipt of a carriage return

has been established.

-the debugging version of Multics for the preliminary implementation has been frozen at version 21-0-2 to eliminate the problems encountered with using a continually changing base. A non-changing set of disk packs, tapes, etc., has been established for debugging and tape generation purposes.

-A simple simulation program has been written to debug without a dn355. Both environments, live and simulation, are used for debugging purposes.

-The 1st interrupts from NPS/355 have been received by the running modified Multics system. No true input or output to the terminal has

yet been achieved.

-No NPS/355 changes as defined in section 7.0 have been made.

APPENDIX A - DIA SOFTWARE

A.1 SCOPE

This appendix provides a convenient means of documenting the standard procedure of passing information between the 355 and 6000 systems. It is a documentation of current software and by no means is a specification of requirements.

A.2 COMMON MAILBOX REGION

All communication between the 355 and 6000 software shall be controlled via a 64 word "mailbox" region in 6000 memory. This block will be located in the first 256K of any extended memory system and will begin on a modulo 64 address. The starting address is specified in the 355 by switches (see 3.2.2.3) and in the 6000 by the startup deck. The 64 word Mailbox Region consists of 8 words of overhead control followed by 7, 8 word channel mailbox areas. These channel mailbox areas shall be assigned on an as-needed basis as areas to transfer control information. Figures A.2.1 and A.2.2 show the structure of the overhead and channel mailbox areas.

Both the 355 and 6000 read the entire channel mailbox areas during normal information transfer. However, in the general case, only one of the systems write into particular words. That system is indicated in the righthand margin.

Table A.2 shows the various operation codes (function) and I/O commands (nature/direction).

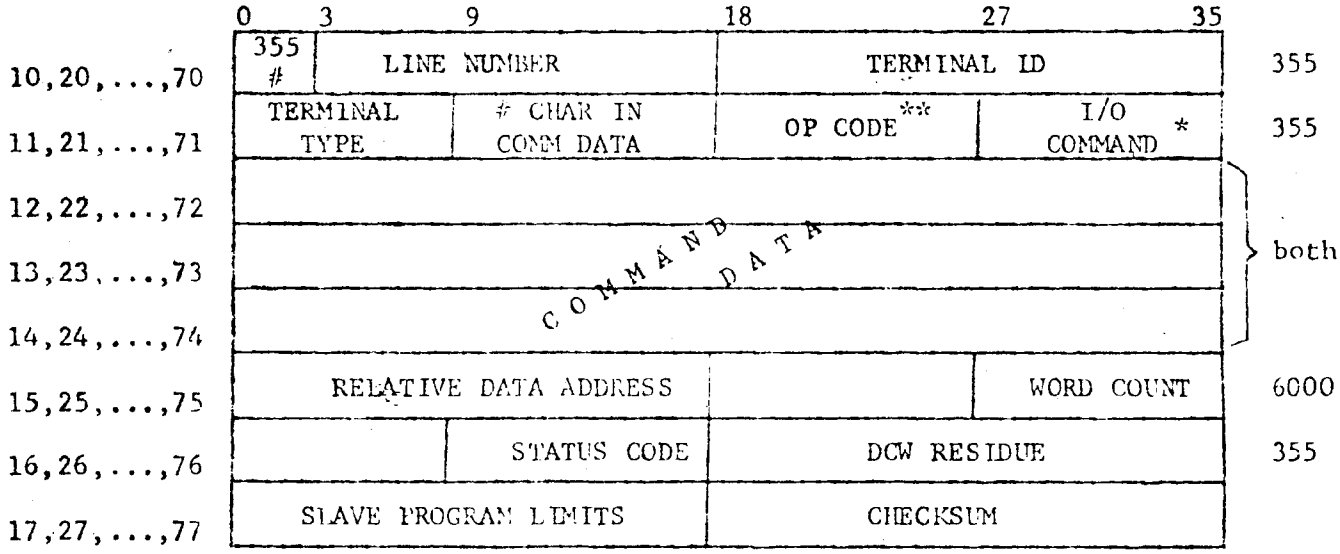
Figure A.2.1 - OVERHEAD AREA

ADDRESS	0	18	24	30	35	
0		0		CHANNEL #	COMMAND	6000
1		# SPECIAL INTERRUPTS				355
2	TERMINATE 1 2 3 4 5 6 7				1	Both
3						Neither
4						Neither
5	355 CONTROL SOURCE	# LINES CONFIGURED				6000
6	POINTER FOR LINE CONFIG.					6000
7	POINTER FOR ANOTHER SYSTEM					6000

Where:

Word	Bits	
0	18	Normally set to zero by 6000, inverted (see 3.4.2) by 355 if command illegal
	24-29	Channel mailbox # when command is 71
		0 - 6000 going down
		1-7 - normal communication
		8-14 - repeat communication (1-7) as 6000 computed checksum not the same as what supplied
		15 - 6000 just booted
	30-35	6000 command to 355
		71 - set ~~execute~~ interrupt cell
		72 - Bootload 355 (0-17 gives 6000 Address)
		75 - Dump 355 core
1	18-35	A modulo 256K count of 355 Special Interrupts
2	0-6	A "1" in any of these bits indicates that the 355 has sent a terminate for that mailbox area that has not been processed.
	35	A "1" means that the 355 is not in the process of updating bits 0-6
5	0-17	355 Control Source (a negative number indicates T&D)
	18-35	Number of 355 lines configured by Startup
6	0-17	Pointer to table describing lines (states, IDs, etc.)

Figure A.2.2 - CHANNEL MAILBOX AREA



Where:	Word (Mod 8)	Bits	
	0	0-2	DATANET 355 #
		3-17	Terminal Line Number
		18-35	Terminal ID
	1	0-8	Terminal Type 003 GE 115 004 TTY 005-010 760's 020 IBM-2741
		9-17	Number Characters in Command Data (Words 2-4)
		18-26	Op Code (See Table A.2.1)
		27-35	I/O Command (See Table A.2.1)
	2-4		Command Data
	5		Relative DCW for data transmission in 6000
	6	9-17	Status Code of peripheral
		18-35	355 DCW Residue
	7	0-17	Slave program limits (from Base Register)
		18-35	Checksum of 30 previous 9 bit characters (rest of mailbox area)

* loaded by only the 6000
** also loaded by 6000

TABLE A.2.1

OPERATION CODES - I/O COMMANDS

OPERATION CODES

From 6000 to 355

0	Terminal Accepted
1	Disconnect This Line
2	Disconnect All Lines
3	Accept No More Calls
4	Accept Calls
5	Input Accepted
6	Terminate Input (Reason)
7	Accept Output
10	Accept Final Output
11	Output Not Available
12	Accept Direct Output
13	Accept Direct Output & Wait for Input
14	Accept Direct Output & Receive Paper Tape
15	Accept Direct Paper Tape
16	Reject Request - Temporary
17	Reject Request - Permanent
20	Terminal Rejected - Reason
21	Disconnect Accepted
22	6000 Initialize Complete
23	Terminate Direct Paper Tape
24	Mass Store Job Accepted
25	Accept Mass Store Space
26	Mass Store Space Denied
27	Mass Store Space OK, Continue
30	Accept Free Link
31	Accept Free Links
32	Accept Links in Use
33	Input List Finished
34	T&D Initialization
35	Break Acknowledged
36	Accept Terminal Characteristics
37	Send USERID and Password
40	Abort Network Processing System (GRTS)
41	Send Configuration Data

TABLE A.2.1 OPERATION CODES - I/O COMMANDS (continued)

From 355 to 6000

100	Accept New Terminal
101	Disconnected Line
102	Accept Input
103	Accept End of Current Job
104	Send Initial Output
105	Send Output
106	Send Status
107	Abort
110	Backspace Output
111	Connect to Slave
112	Accept Direct Input
113	Break Condition
114	Connect to Slave with No Wait
115	Accept Mass Store Job
116	Accept Configuration Info and Send MS Space
117	Restart Successful
120	Restart Unsuccessful
121	Link(s) Released
122	Old Mass Store Space (No Input List)
123	Old Mass Store Space (Input List)
124	Accept Mass Store Input List
125	Terminal Characteristics Accepted

From 355 to 355

200	Mass Store Done
201	Mass Store Disconnects
210	Mass Store Restart
211	Mass Store Write Input List
212	Mass Store Read Pack Label
300	ICA Detected Error

I/O COMMANDS

(all put in mailbox by 6000)		Command Executed by
01	Read Control Data	6000
02	Read Text	6000
03	Write Control Data	355
04	Write Text	355
05	Automatic Call Unit	355
06	Reject Mailbox	355
07	Send Terminate Interrupt	355
08	Delayed Write Text	355

A.3 STANDARD RESPONSES - CONVENTIONS

To simplify any step by step discussion of various communications between the 355 and 6000, several standard responses are described below. These conventions involve the use of interrupts, checksum words, and correlating devices and responses. Any detailed discussion will reference these conventions.

1. When the 355 wishes to initiate communication with the 6000, it sends a special interrupt to request a mailbox area (1-7) to do the actual communication.

2. The 355 ends a communication by setting a terminate bit (word 2) and issuing a terminate interrupt. It does this by clearing the word in the 6000 (see 3.4.1.12) and replacing that word with another that has both the identical bits set, and the new one corresponding to the channel mailbox it is finished with.

3. The DIA sets a unique interrupt for each of the seven channel mailboxes so the 355 will never read the overhead area (first 8 words). The 6000 on the other hand, receives only one level for both special and terminate interrupts. It responds to an interrupt in the following manner:

   - It checks the number of special interrupts received (word 1) with the number of specials serviced (internal word). If the former is greater, it processes each by finding a free mailbox area, placing that mailbox area number and a set interrupt cell command (71) in word 0, placing a Read Control Data command (01) in word 1 of that channel mailbox area, and issuing a connect to the DIA. It does this up to the limit of the available channel mailbox areas.

   - If it runs out of channel mailbox areas, or the interrupt was not a special, the 6000 then processes all terminates (word 2) by doing a Read Clear and processing all terminates recorded. (Note: If the 6000 has earlier processed multiple terminates, or if the 355 is currently updating this word, the 6000 will not find any terminate bits. It shall simply ignore this interrupt. This doesn't cause any loss of interrupts because they have already been serviced or another interrupt will occur shortly and it will be serviced with that one.)

4. After the line number is supplied by the 355, individual responses are recognized by this attribute.

A.3 STANDARD RESPONSES - CONVENTIONS (continued)

5. The first thing the 355 or 6000 does when it receives a channel mailbox area that contains new information is to copy it into storage, calculate the checksum for the first seven and one half words, and compare that with the one supplied. If the 355 finds a disagreement between them, it rereads the channel mailbox area and rechecks some prespecified number of times before aborting. If the 6000 finds a disagreement, it requests rewriting by the 355 (increase channel mailbox number by 7 and reconnect) and rechecks some prespecified number of times before aborting.

6. Whenever either system wants to change an entry in a channel mailbox area, it places the change in its copy, recalculates the checksum, copies the eight words into that particular mailbox and terminates (or connects).

7. The 6000 writes all the I/O commands in the channel mailbox. Both the 355 and 6000 start their services by reading this command. If the 355 finds a 6000 command (see table), it realizes this was in response to its Special Interrupt, adds the necessary channel mailbox information, and terminates. If the 6000 finds a 355 command, it simply releases the mailbox area. In the other two cases, the 355 and 6000 perform the service requested and respectively issue their terminate or connect.

A.4 SAMPLE SEQUENCE

As a typical example of an I/O sequence, the following is the sequence of transactions between the 355 and the 6000 which occurs to input a userid from a time sharing terminal after it has dialed in. This example assumes no checksum error (see Convention 5) or terminal rejection.

a. After the 355 has determined that a new terminal has called in (and needs a banner message), it sends a Special Interrupt to the 6000. (See Convention 1.)

b. The 6000 responds with a RCD command (See Convention 3.)

c. The 355 reads the RCD command and, realizing that this is an operation for the 6000, puts (see Convention 6) the 355 #, the Line Number, Terminal ID, Terminal Type, # of Characters (0), Accept New Terminal Op code (100), and new checksum in the channel mailbox assigned. It then sends a Terminate Interrupt to the 6000. (See Convention 2.)

d. The 6000 reads the RCD command and Accept New Terminal Op code. This causes it to process the information supplied. If the terminal is accepted, it writes a Write Control Data command (03), a Terminal Accepted Op code (000), and new checksum in the channel mailbox. It then sends a connect to the DIA.

e. The 355 sees the WCD and, realizing that this is a 355 operation, sends the information just added to the 355 program and a Terminate Interrupt to the 6000.

f. The 6000 reads the WCD command and, realizing that this is a completed (355) operation, releases the mailbox area.

g. The 355 shortly is informed by the 355 program that a "connect to slave prog." is needed and again sends a Special Terminate to the 6000.

h. The 6000 again responds with a RCD.

i. The 355 reads the RCD command and changes the Op code to Connect to Slave (111), the # of characters to 3, and puts a 6000 Program name in word 2. It then sends a Terminate.

j. The 6000 reads the Program name (and the rest of the mailbox area), notifies the 6000 Program, and releases the mailbox.

A.4 SAMPLE SEQUENCE (continued)

k. The 6000 is shortly informed by the 6000 Slave Program where the banner message is located. It finds a mailbox area, and restores it with the following changes: the I/O command is Write Text (04) the Op code is Accept Direct Output (012), the # of Characters is 0, and the bounds of the Slave Program and the relative address and tally of the banner message is supplied. It then sends a connect to the DIA.

l. The 355 copies the mailbox area, starts processing the information, and sends a Terminate Interrupt.

m. The 6000 releases the mailbox.

n. The 355 is then informed by the 355 program that a userid message is needed and sends a third Special interrupt.

o. The 6000 again responds with a RCD.

p. The 355 reads the RCD command and restores the mailbox with the information read except it leaves the RCD command and changes the Op code to Send Output (105). It then sends a Terminate Interrupt to the 6000.

q. The 6000 reads the RCD, changes the I/O command to Write Text, the Op code to Accept Direct Output and Wait for Input (013), and the relative address and tally to point at the userid message. It then sends a connect to DIA.

r. The 355 reads the WTX, waits for input op code, control information for response and output message as pointed to by tally word, outputs the message, accepts input and stores it in the 6000 as per the control information, and sends a terminate interrupt to the 6000.

A.5 SUMMARY

As demonstrated in the preceeding example several observations can be made about 355/6000 communications.

1. 6000 dominates - 355 must do "may I?" step before initiating any service.

2. Mailbox areas are released while either system is processing the information supplied in the last step.

3. A channel mailbox area consists of eight words that describe a device and control the I/O transactions. The words are massaged by either one or the other system, but the basic characteristics uniquely identifying the device are retained during each step.

4. The I/O command is supplied by the 6000 and tells each system whether to ignore this mailbox (service not for them) or to look at the op code to determine what to do.

5. The Op code describes the service.

6. The Command Data supplies addition, short control information (program names, terminate, reasons, etc.).

APPENDIX - B

Within the channel mailbox is a command field and an op-code field. The command field is defined to contain one of the following ...

1	RCD	Read Control Data
2	RTX	Read Text
3	WCD	Write Control Data
4	WTX	Write Text
5	ACU	Automatic Call Unit
6	RJT	Reject Mailbox
7	STI	Send Terminate Interrupt
8	DWT	Delayed Write Text

The first four of these (RCD, RTX, WCD, WTX) are the commands used most extensively by NPS and are defined more fully below.

The op-code field is used to further define the command. For some commands, within certain contexts, an op-code field may not be required. The dialog between NPS and the 6000 is really a handshaking where the commands define in general what the mailbox contains and the op-code defines specifically the function being performed.

In general the following statements may be made about the RTX, WTX, RCD, and WCD commands.

1. The 6000 places all 4 of the commands into the channel mailboxes. The DN355 merely uses the command which is in the channel mailbox.
2. The commands which deal with transfer of data other than mailbox information always are the RTX or WTX commands.
3. The commands which deal with transfer of only mailbox information are the RCD and the WCD commands.
4. The R commands designate flow from the DN355 to the 6000 while the W commands designate flow from the 6000 to the DN355.

RCD

a. The 6000 sends this command in response to a special interrupt from the DN355. It means "here is a free channel mailbox".

Specifically, when the 6000 receives a special interrupt from the DN355 it ...

1. finds a free channel mailbox
2. places a RCD command code into the channel mailbox
3. issues a connect to the DN355 on the correct channel

b. When the 6000 receives a terminate interrupt and the command code in that channel mailbox is RCD it means that the DN355 has just put an op-code into that mailbox and it is an op-code from the DN355 to the 6000.

RTX

a. The 6000 sends this command (i.e. places it into a channel mailbox and issues a connect) to tell the DN355 that it has placed an op-code into this channel mailbox which requires a data transfer from the DN355 to the 6000. At this time the DCW is sent so that the DN355 may use it.

b. When the 6000 receives a terminate interrupt and the command code in that channel mailbox is RTX it means that the DN355 has just completed processing the RTX (same one) command that the 6000 had sent (e.g. input accepted) and the 6000 may now free the channel mailbox.

WCD

a. The 6000 sends this command to tell the DN355 that it has placed an op-code into this channel mailbox for the DN355 to process. i.e. it is an op-code sent from the 6000 to the DN355 via a connect.

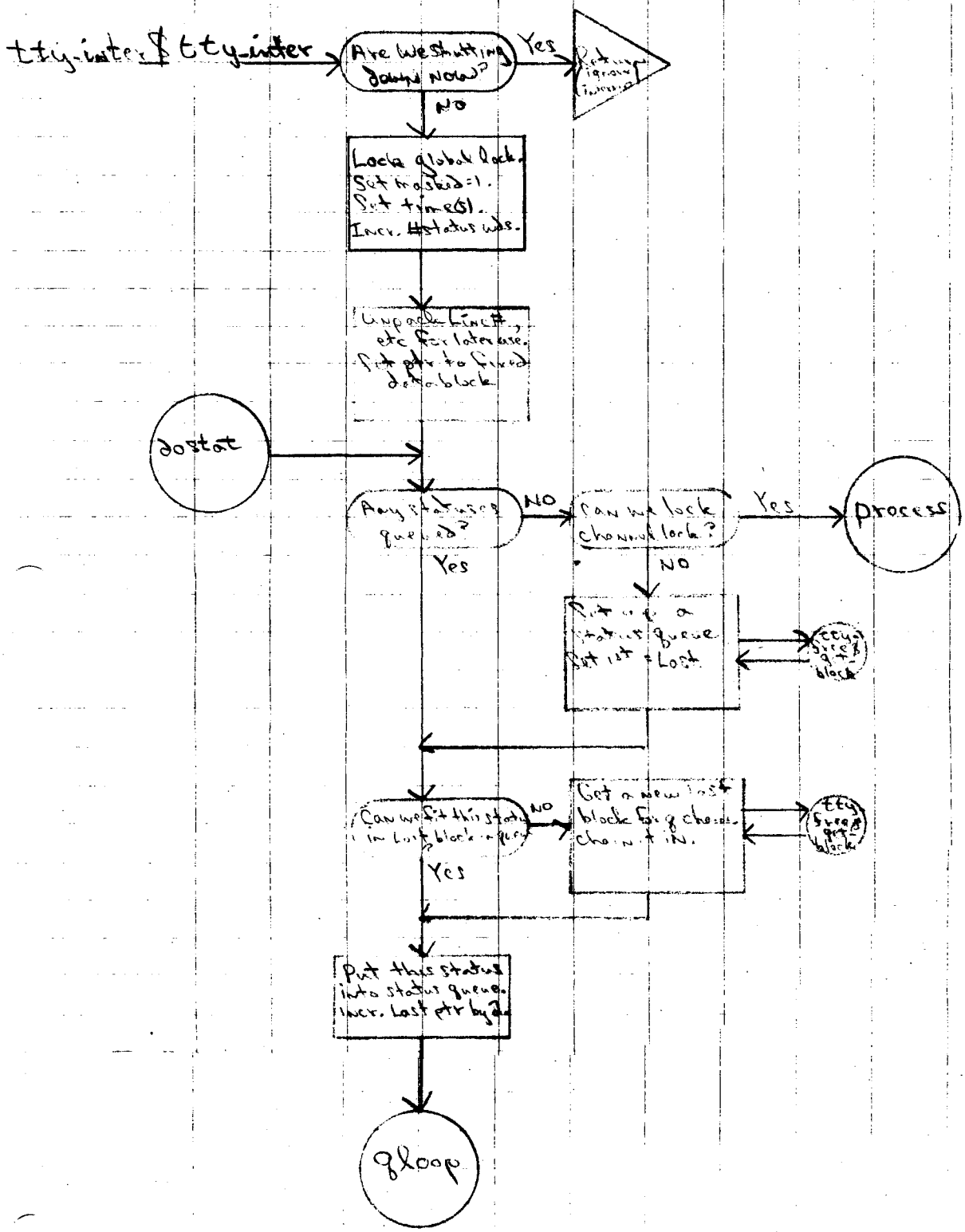
b. When the 6000 receives a terminate interrupt and the command code in that channel mailbox is WCD it means that the DN355 has just completed processing the WCD op code the 6000 had just sent (same one) and the channel mailbox may be freed now.

WTX

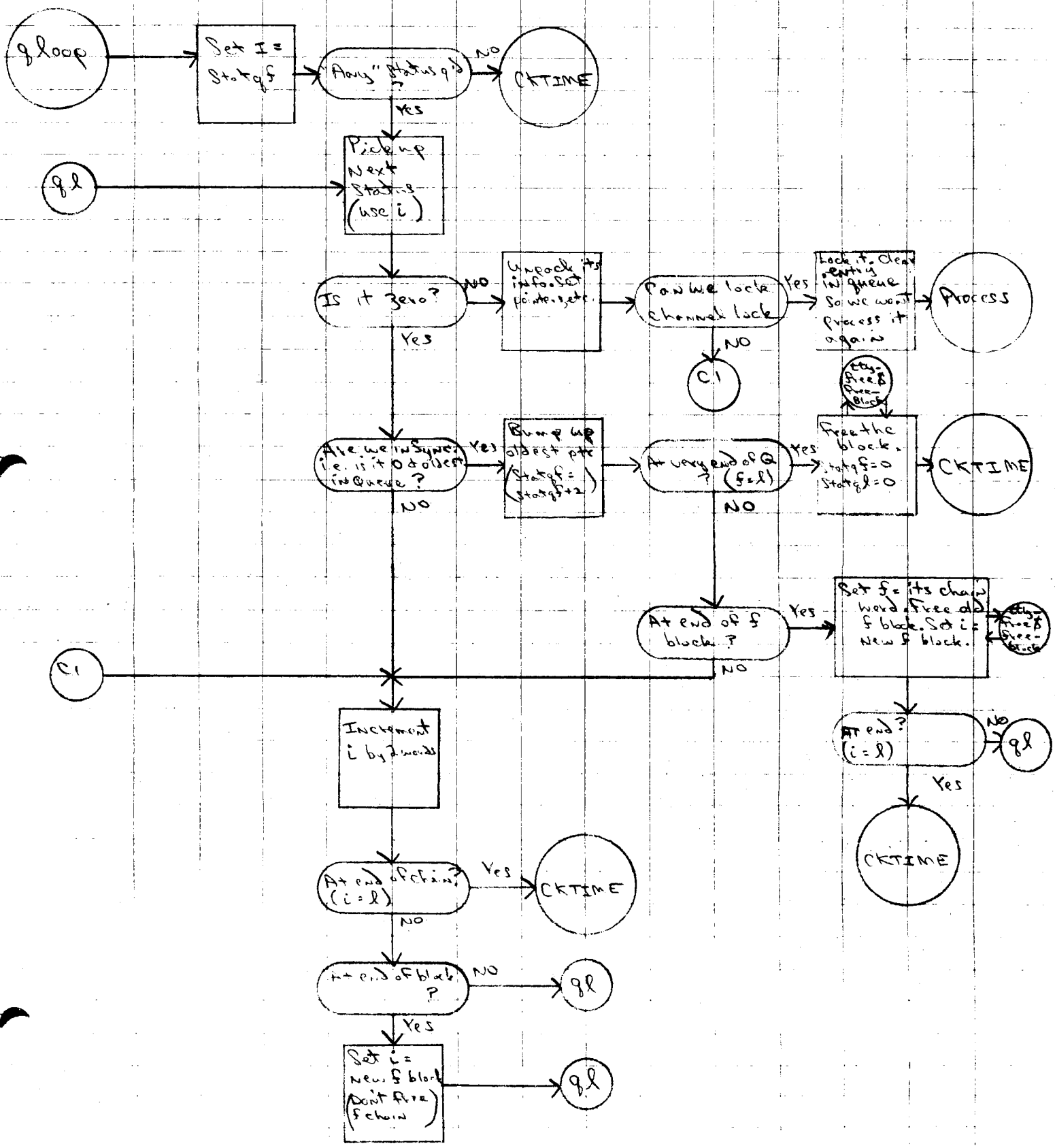
a. The 6000 uses this command to tell the DN355 that it has placed an op-code into this channel mailbox which requires a data transfer from the 6000 to the DN355. At this time the DCW is sent so that the DN355 may use it.

b. When the 6000 receives a terminate interrupt and the command code in that channel is WTX it means that the DN355 has just completed processing the WTX op-code the 6000 had just sent (same one) and the channel mailbox may now be freed. The data may be released now as the DN355 has moved the data to its memory.

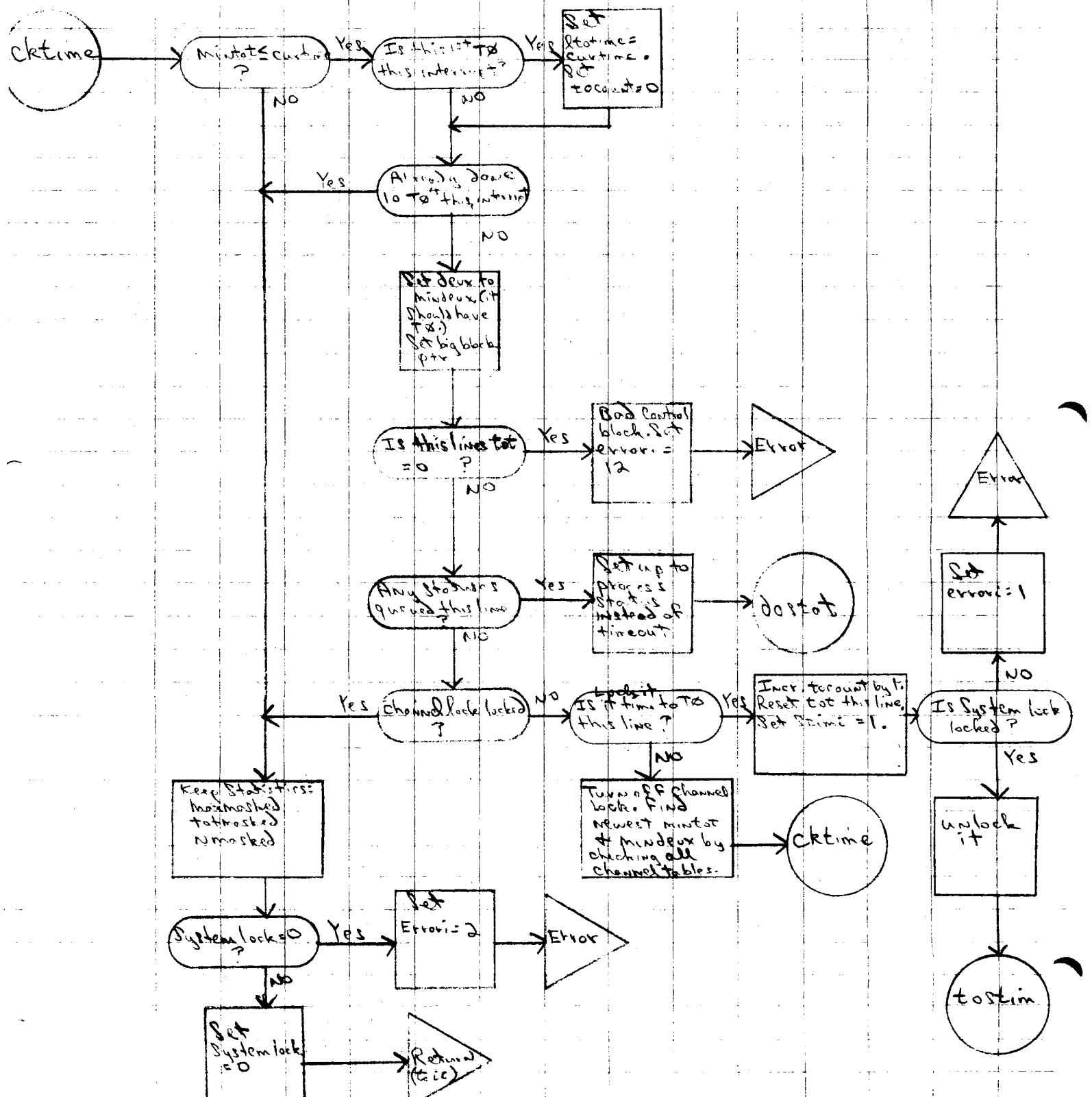
Entered Only from dn355$inter → free_mbx because of a status received from II



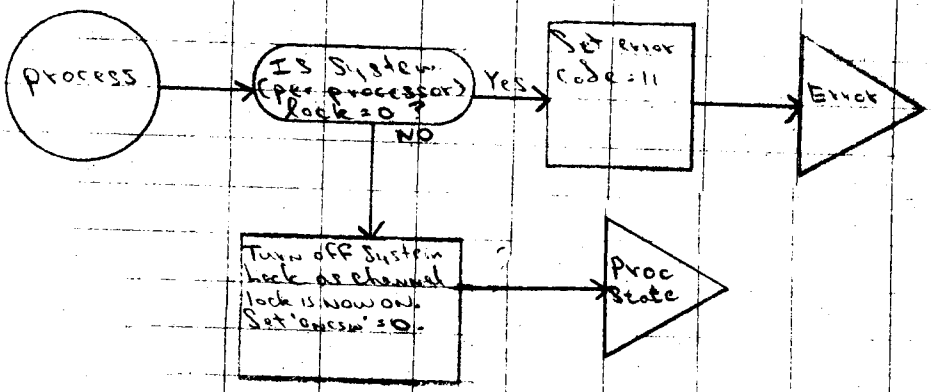
Entered here ONLY when in tty-inter to process a status.
i.e. entry to tty-inter via $write/write-abort/test-state/
or prescan never come into here.

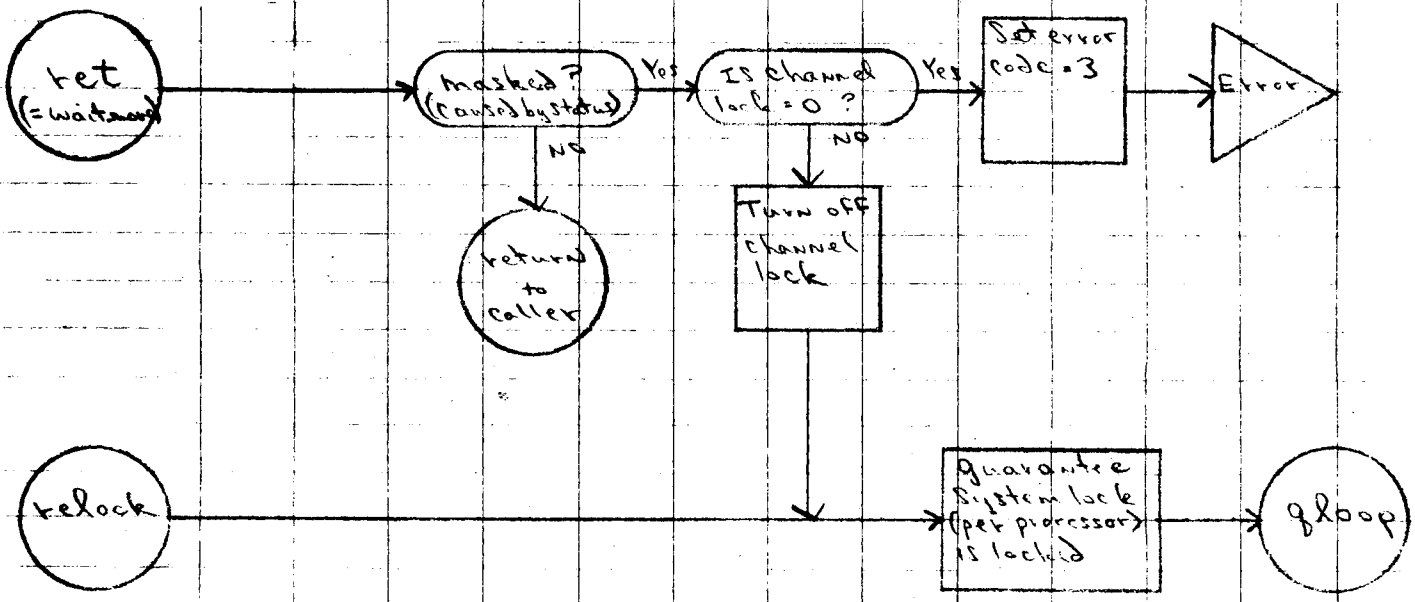


Entered Only when tty_inter has been entered via its $inter entry point and only then after all statuses have been processed.

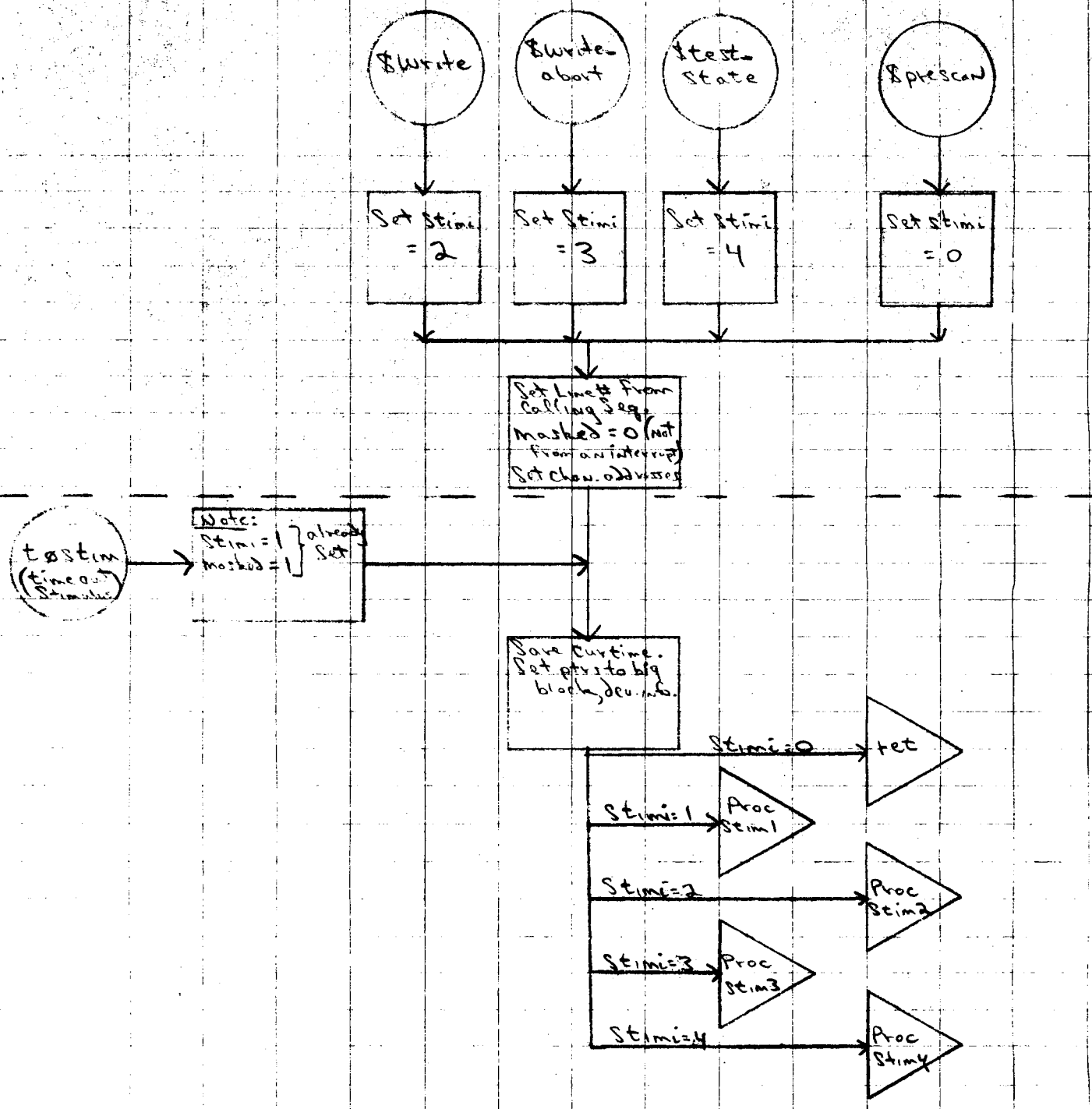


Entered here only when processing a status. i.e. entry to tty-inter was via its $inter entry point.



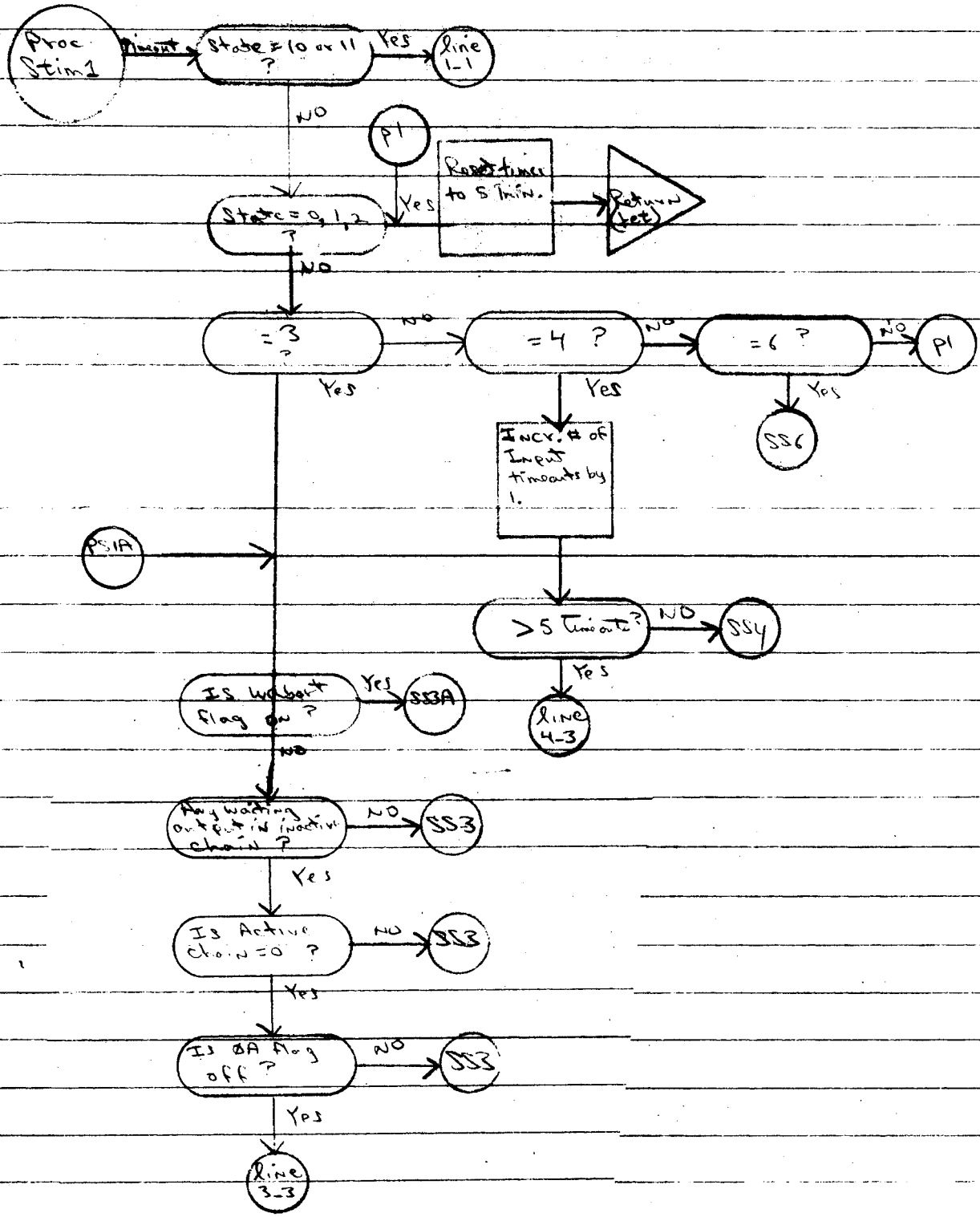


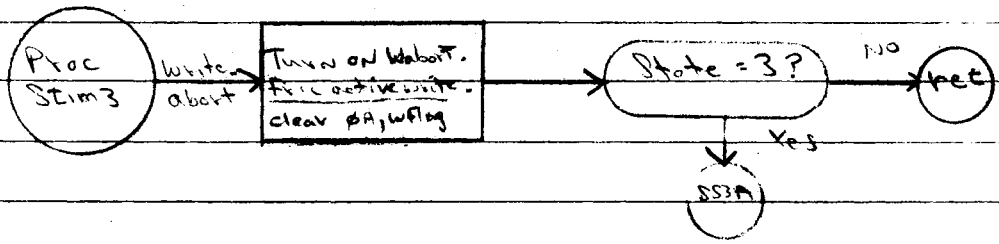
Entered here ONLY from external to tty-inter - (i.e. not a status entry)
(per processor lock not set, masked = 0)

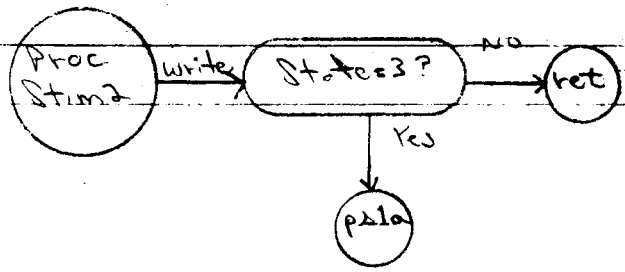


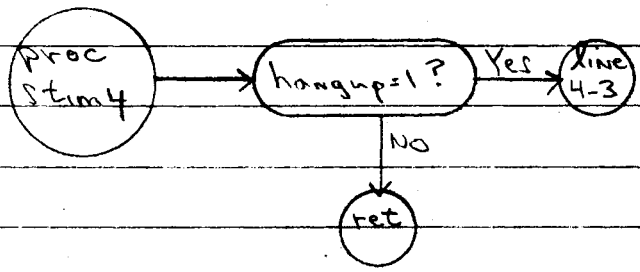
Entered here from cktime routine after tty-enter $ inter had been entered to process a status.
(per processor lock is locked, masked = 1)

Timeout Stimulus



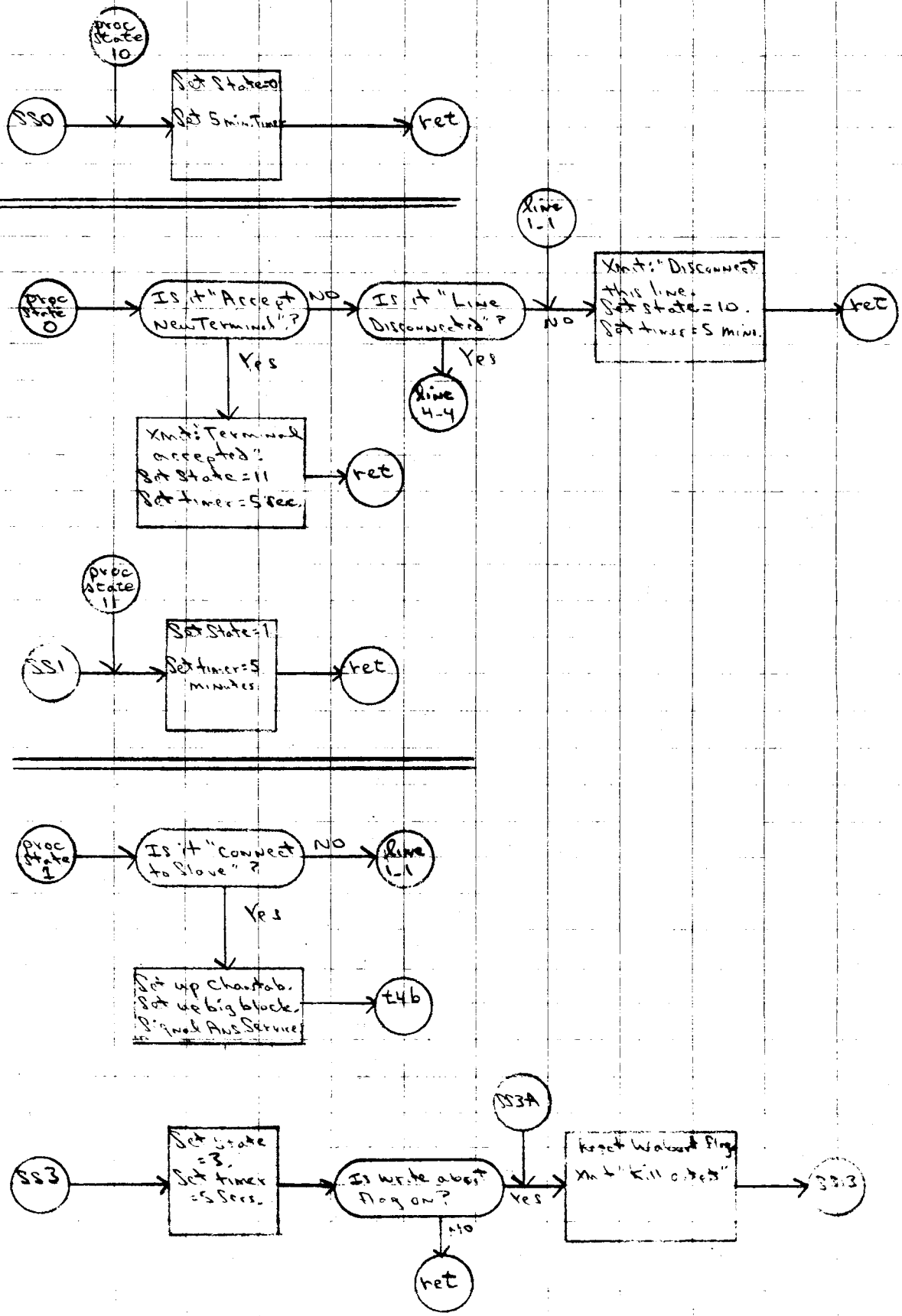

Proc Stim3
write abort
Turn on Wabort.
Fric active write.
clear #A, wFlag
State = 3?
NO
ret
Yes
SS3A

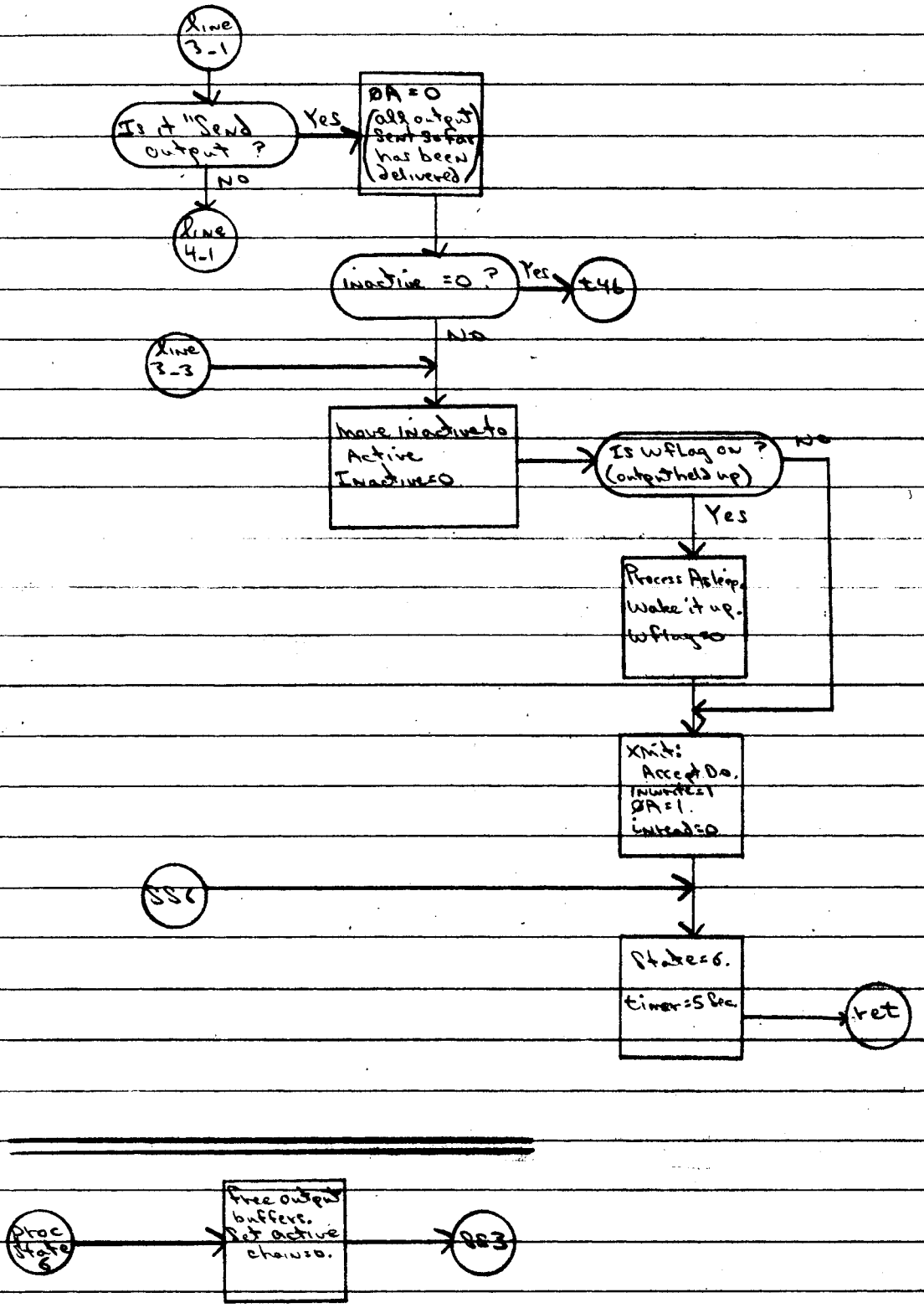

Proc Stim2
write
State=3?
NO
ret
Yes
p2la

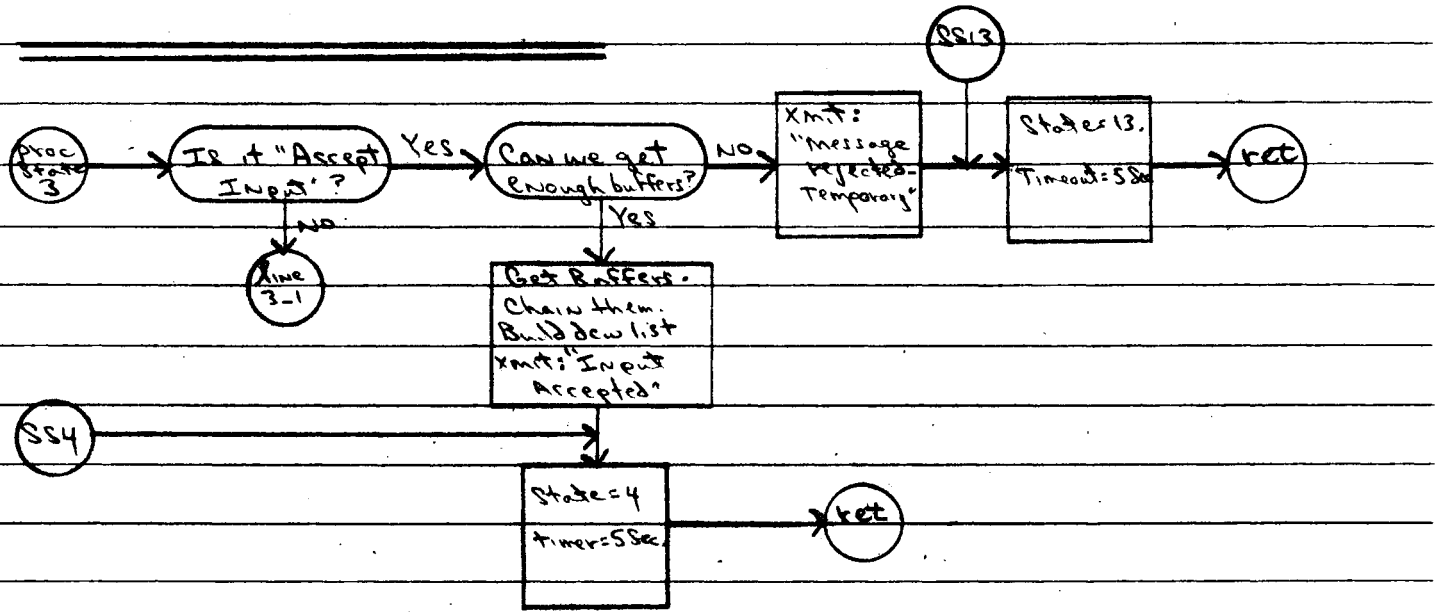

proc Stim4
hangup=1?
Yes
line 4-3
NO
ret

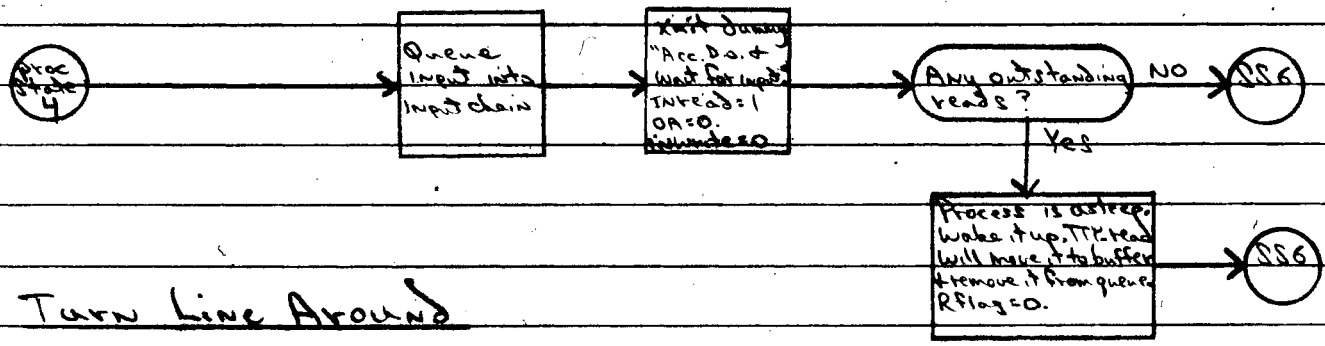
Goto Vect (state)

State	go to	Comment
= 0	procstate0	idle
= 1	procstate1	initial handshake
= 2	procstate2	error
= 3	procstate3	Main Cycle
= 4	procstate4	input line coming in
= 5	procstate5	error
= 6	procstate6	output just started
= 7	procstate7	error
= 8	procstate8	error
= 9	procstate9	error
= 10	procstate10	idle at this terminate
= 11	procstate11	initial handshake - at terminate
=	procstate12	error
= 13	procstate13	main cycle at terminate

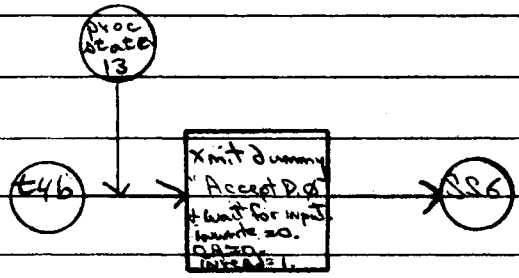

proc state 10
SS0
Set State=0
Set 5 min. Timer
ret
line 1-1
proc state 0
Is it "Accept NewTerminal"?
NO
Is it "Line Disconnected"?
NO
Xmit: "Disconnect this line.
Set state=10.
Set timer=5 mins.
ret
Yes
Yes
line 4-4
Xmit: Terminal accepted.
Set State=11
Set timer=5 sec.
ret
proc state 11
SS1
Set State=1
Set timer=5 minutes.
ret
proc state 1
Is it "connect to Slave"?
NO
line 1-1
Yes
Set up Chantab.
Set up big block.
Find ANS Service
t46
SS3
Set State =3.
Set timer =5 secs.
Is write about flag on?
SS3A
Yes
Reset Wakeup flag
Xmit "Kill others"
SS3
No
ret

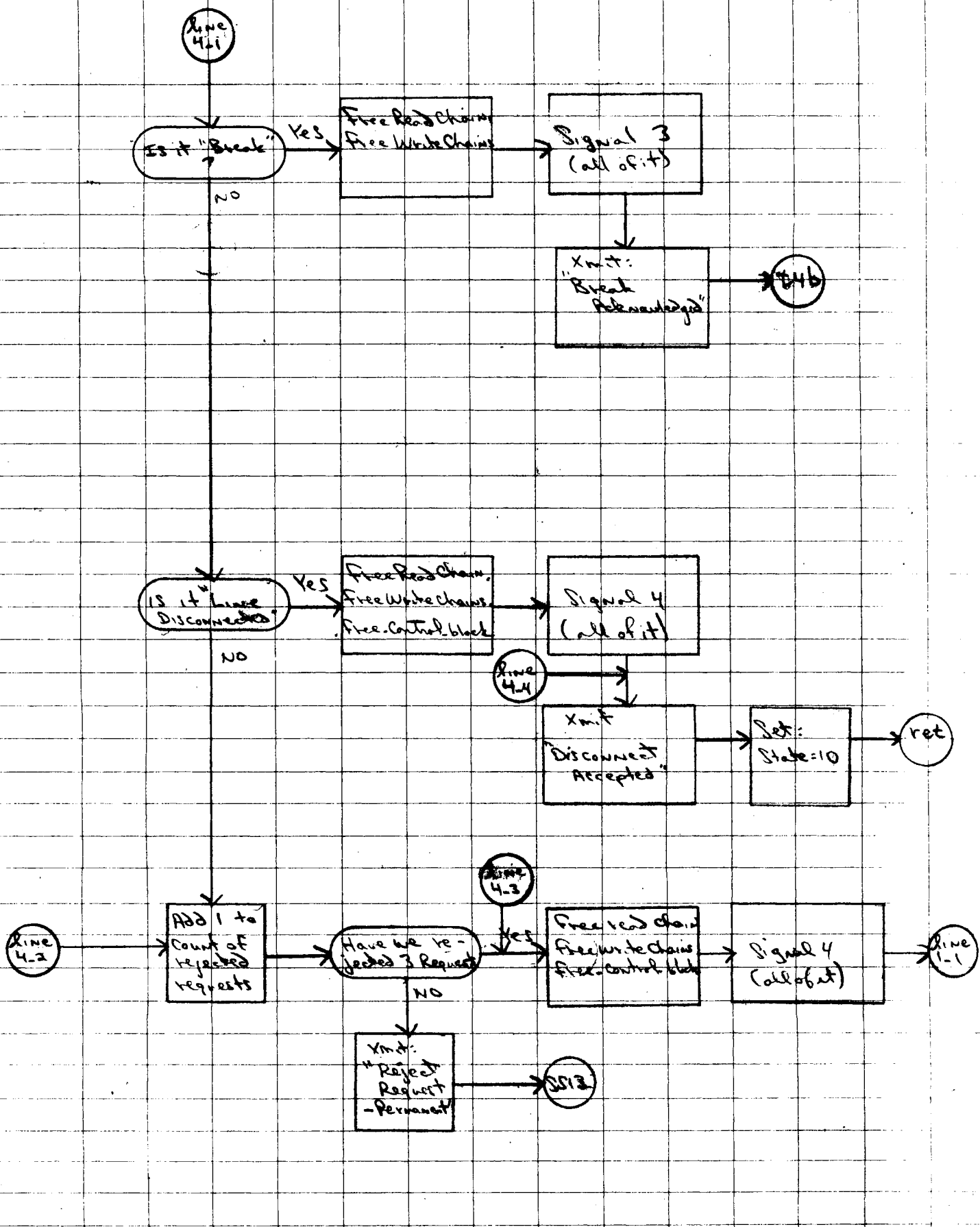

Line 3-1
Is it "Send output ?
Yes
DA = 0 (all output sent so far has been delivered)
NO
Line 4-1
inactive = 0 ?
Yes
t46
NO
Line 3-3
Move inactive to Active
Inactive = 0
Is wflag on ? (output thrd up)
No
Yes
Process Asleep
Wake it up.
wflag = 0
Xmit:
Accept Da.
INWRITE = 1
DA = 1.
inread = 0
SSC
State = 6.
timer = 5 sec.
ret
Proc State 6
Free output buffers.
Set active chain = 0.
883





Turn Line Around



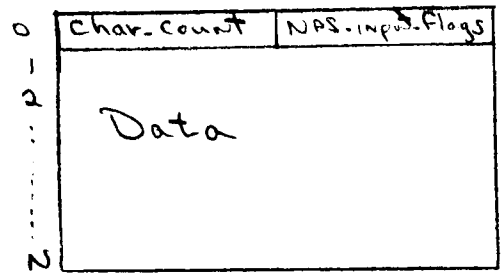

Line 4.1
Is it "Break"
Yes
Free Read Chains
Free Write Chains
Signal 3 (all of it)
NO
Xmit: "Break Acknowledged"
tt4b
Is it "Line Disconnected"
Yes
Free Read Chains
Free Write Chains
Free_Control_block
Signal 4 (all of it)
NO
Line 4.4
Xmit "Disconnect Accepted"
Set: State=10
ret
Line 4.3
Line 4.2
Add 1 to Count of rejected requests
Have we rejected 3 Requests
Yes
Free read chain
Free write chains
Free control block
Signal 4 (all of it)
Line 1.1
NO
Xmit: "Reject Request - Permanent"
tt13

APPENDIX - D

INPUT_DATA_FORMAT

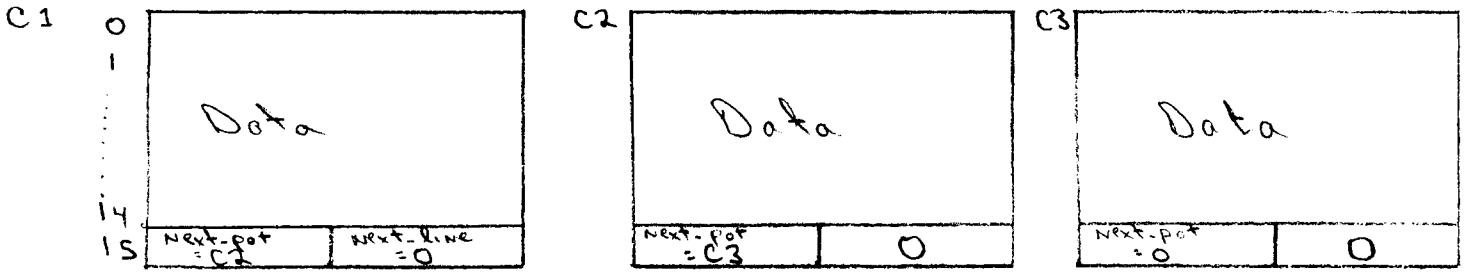
There are two formats used in processing input. One format is used to allow NPS to transmit input in the current GERTS format to a fixed area within the 6180 via one DCW. The other format is used to queue this input into Multics 16 word buffers which are in the free-store area within tty_buf.

The NPS buffer format is as shown below. Note that it is one contiguous area with the first word reserved for the character count and certain NPS flags. At the time the DCW pointing to this area is given to NPS, the maximum word count is also sent. NPS may or may not use the entire area depending on the size of the input message it is transmitting to Multics.



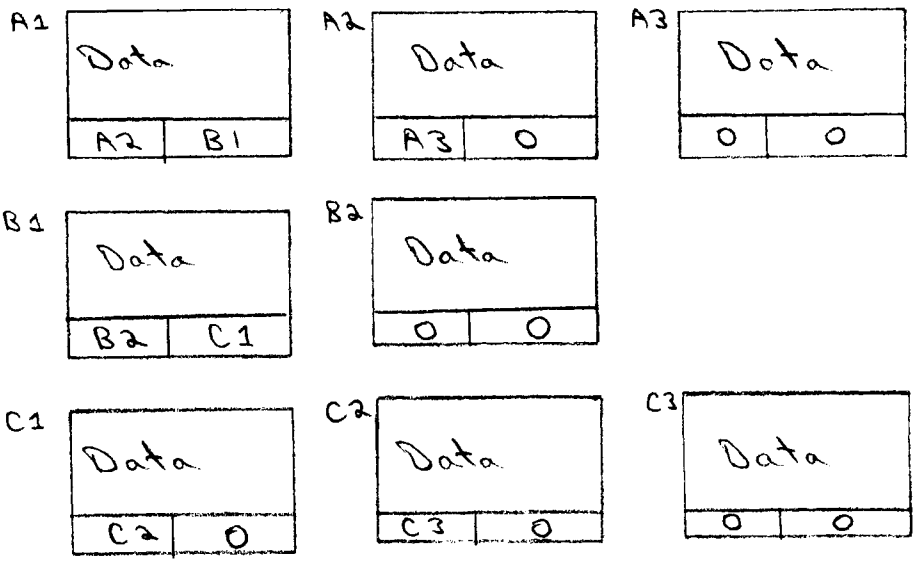
For the purposes of the feasibility study, either one or two of these fixed input buffers will be utilized. In an actual implementation some number greater than this would be used. Since this fixed area is only used for a very short duration of time it is not necessary to have many of them. As a matter of fact the rotating queue concept proposed by R. Snyder would probably be used in the final implementation although its usage/allocation would be controlled by the 6180 and not by the DN355 as

he initially suggested. NPS/Multics require use of this fixed buffer only for the duration of time from when the "Input Accepted" message is sent (via a channel mailbox) to NPS until the terminate on that channel mailbox is received from NPS (i.e. NPS has written over the input line). If there are no free fixed input buffers at the time Multics receives the "Accept Input" message from NPS then Multics will respond with a "Message rejected - temporarily" operation code. This causes NPS to re-submit the request after a short delay. At the terminate interrupt processing of the "Input Accepted" message the contents of the fixed buffer are moved to Multics 16 word buffers as shown below.



This chain of pots, representing one line, is then chained into the chain of lines, if any, as shown below.

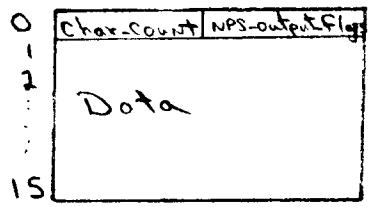
fblock = A1
lblock = C1



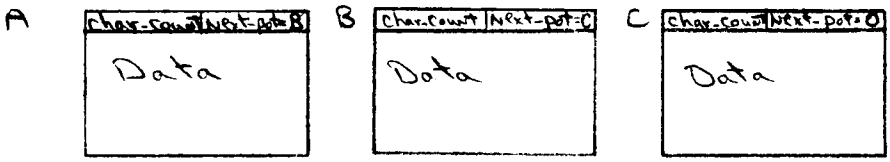
APPENDIX - E

OUTPUT DATA FORMAT There are two formats used in processing output. One format is used for the inactive chain to queue output in the current Multics method (although the buffer format itself has been slightly changed). This format is generated by routine tty_write. The other format is used for the active chain and it is in the standard NPS/GERTS format. For purposes of the feasibility study this format is defined to be 16 words in length (one dynamic memory pot from tty_buf) where the first word contains a character count and NPS output flags. It is generated by tty_inter during the swap from the inactive chain to the active chain.

The active chain format is shown below. Note that it is one pot in length. It is generated by pulling the oldest inactive pot out of the inactive chain and re-formatting it to become the active chain.



The format of the inactive chain is shown below. As output is generated it is chained onto the end of the inactive chain.



At a swap-write function (done when the active chain has been emptied), a maximum of one pot is pulled off the front of the inactive chain and, with reformatting, becomes the active chain. This maximum of one pot restriction will be removed when NPS incorporates scatter read capabilities.

Key variables, maintained on a per-channel basis within the users big block, are defined below.

1. Wafblock = relative pointer to first word of first (and only) block (within tty_buf) of active chain.

2. Wafchar = relative pointer to first character (within Wafblock) to be output.

3. Walblock = same as number 1

4. Walchar = relative pointer to last character +1 (within Walblock) to be output. Not really applicable in a one pot active chain.

5. Wacc = total number of characters output via the active chain.

6. Wiafblock = relative pointer to first word of first block (within tty_buf) of the inactive chain.

7. Wialblock = relative pointer to first word of last block of the inactive chain.

8. Wiafchar = relative pointer to first character to be output from first block.

9. Wialchar = relative pointer to last character +1 to be output from last block.

10. Wiacc = Number of characters in inactive chain.

APPENDIX - F

MULTICS DATANET-355 INTERRUPT HANDLING (OVERVIEW)

There are two types of mailboxes on MULTICS: one 8-word main mailbox and fifteen 8-word channel mailboxes allocated per Datanet-355. Whenever the DN-355 interrupts the 6180, control is given to routine dn355$inter by the interrupt interceptor (II).

The DN-355 will interrupt the 6180 whenever it:

a. Wishes to send one or more terminate statuses. It indicates this by setting on the appropriate bit (one per channel mailbox) in word 2 of the main mailbox. (This designates which channel mailbox has terminated.)

b. Wishes to send information to the 6180 but needs a channel mailbox to do so. It indicates this by adding 1 to a count field (for each mailbox it needs) in word 1 of the main mailbox. This is the Special interrupt count.

Therefore, when an interrupt occurs in the 6180 it may be for either or both of the above reasons; i.e., none, 1, or more terminate bits may be on and the special interrupt count may be incremented by none, 1, or more.

When the DN-355 interrupts the 6180, routine dn355 must determine why the interrupt occurred and then take the appropriate action; e.g.:

If the special interrupt count has been incremented then a free channel mailbox must be found and given to the DN-355 (via a notification through the main mailbox which includes a connect).

If a terminate with no status has occurred then the appropriate channel mailbox must be freed, and

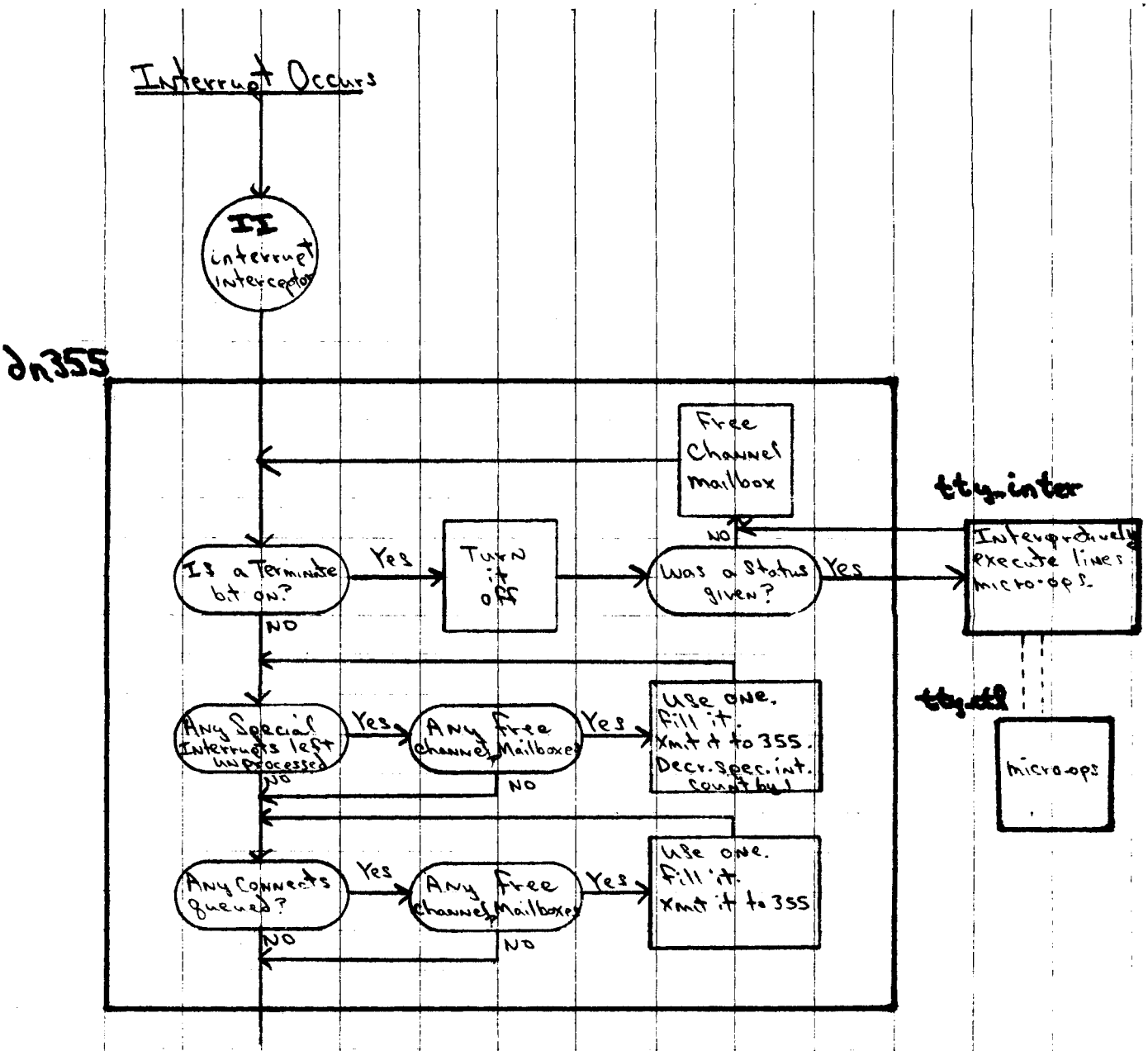
If a terminate with status has occurred then the appropriate channel mailbox must be freed and the status must be analyzed dependent on what that particular line is doing at that time.

At this point a brief explanation of micro ops is beneficial in order to illustrate the continuing flow of interrupt handling of the DN-355.

A set of micro ops (macros) have been defined in segment tty_macros. These macros (called IOC language) are used to define the line discipline for all tty terminals. The line discipline is written in this language and resides in segment tty_ctl.ioc. This segment is interpretively executed by a portion of routine tty_inter. Each terminal has a control table which has a pointer to some position in segment tty_ctl. This position defines where that terminal is in its line control and what it will do next. When tty_inter is given control for that line it will use the pointer to interpretively execute the macro in segment tty_ctl to which it is pointing.

In case C above, (a terminate with a status) the channel mailbox is freed after first preserving the status. Control is now given to routing tty_inter to process this received status. Routine tty_inter will take that line's current pointer to some position in the micro ops (in segment tty_ctl) and interpretively execute it. It will interpretively execute as many micro ops as it can for this line and return control to routine dn355 when it becomes blocked.

The following chart pictorially depicts the above flow..


Interrupt Occurs
II
interrupt interceptor
dn355
Free Channel mailbox
tty.inter
NO
Is a Terminate bit on?
Yes
Turn it off
Was a Status given?
Yes
Interpretively execute lines micro-ops.
NO
tty.ctl
Any Special Interrupts left unprocessed
Yes
Any Free channel Mailboxes
Yes
Use one. Fill it. Xmit it to 355. Decr. Spec. int. count by 1
NO
NO
micro-ops
Any Connects queued?
Yes
Any Free channel Mailboxes
Yes
Use one. Fill it. Xmit it to 355
NO
NO

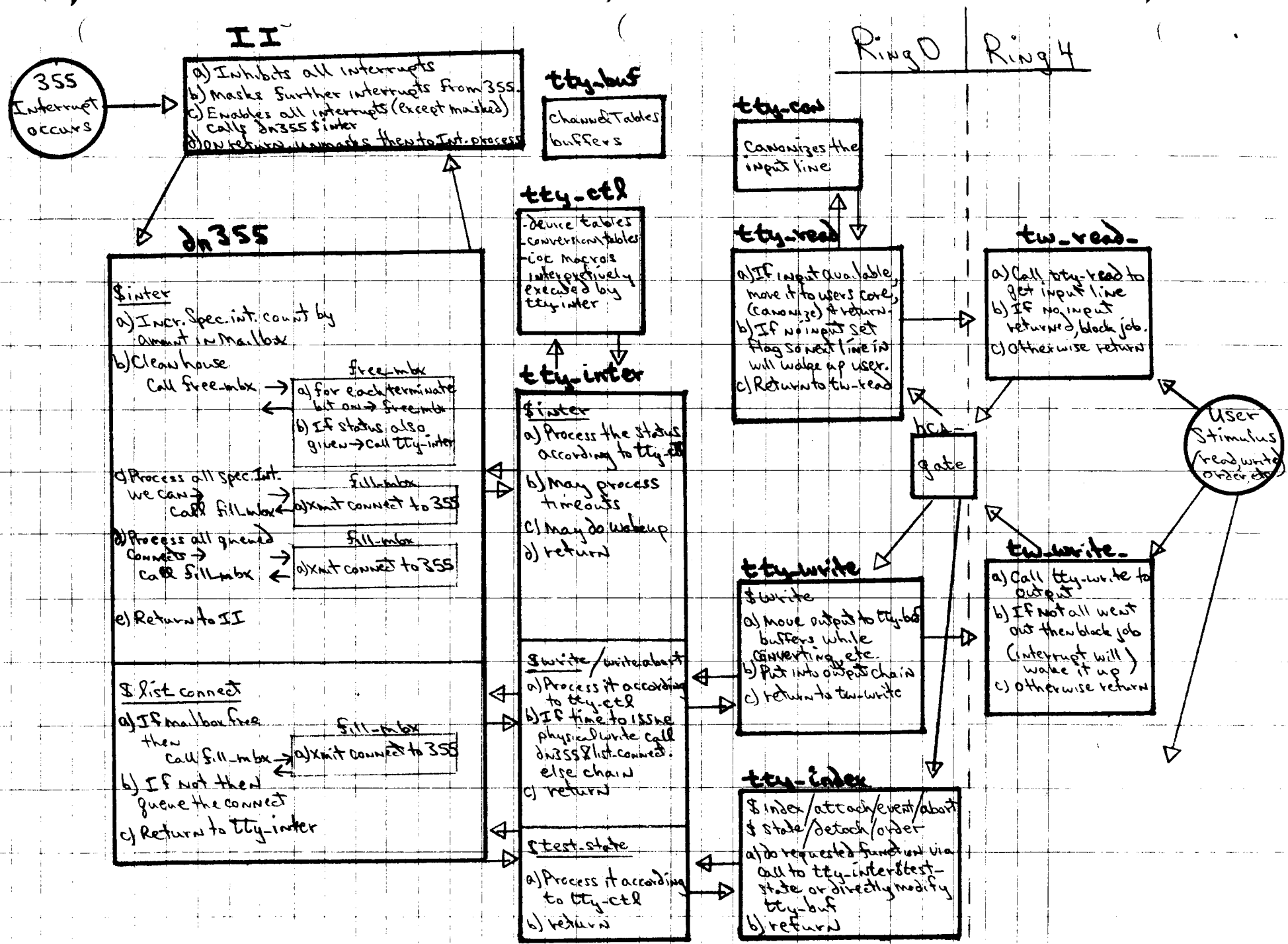

II
355 Interrupt occurs
a) Inhibits all interrupts
b) Masks further interrupts from 355.
c) Enables all interrupts (except masked)
d) Calls dn355 $inter
e) On return unmasks then to Int. process
tty_buf
Channel Tables
buffers
tty_ctl
-device tables
-conversion tables
-ioc macros
interpretively executed by tty_inter
dn355
$inter
a) Incr. Spec. int. count by amount in mailbox
b) Clean house
Call free_mbx
free_mbx
a) for each terminate bit on → free mbx
b) If status also given → call tty_inter
c) Process all Spec. Int. we can
Call fill_mbx
fill_mbx
a) Xmit connect to 355
d) Process all queued Connects
Call fill_mbx
fill_mbx
a) Xmit connect to 355
e) Return to II
$ list connect
a) If mailbox free then
Call fill_mbx
fill_mbx
a) Xmit connect to 355
b) If not then queue the connect
c) Return to tty_inter
tty_inter
$inter
a) Process the status according to tty_ctl
b) May process timeouts
c) May do wakeup
d) return
$write / writeabort
a) Process it according to tty_ctl
b) If time to issue physical write call dn355 & list_connect. else chain
c) return
$test_state
a) Process it according to tty_ctl
b) return
Ring 0 | Ring 4
tty_con
Canonizes the input line
tty_read
a) If input available, move it to users core, (canonize) & return
b) If no input set flag so next line in will wake up user.
c) Return to tw_read
tw_read_
a) Call tty_read to get input line
b) If no input returned, block job.
c) Otherwise return
hcs_ gate
User Stimulus (read, write, order, etc)
tty_write
$write
a) Move output to tty_buf buffers while converting etc.
b) Put into output chain
c) return to tw_write
tw_write_
a) Call tty_write to output
b) If not all went out then block job (interrupt will wake it up)
c) Otherwise return
tty_index
$ index / attach / event / abort
$ state / detach / order
a) do requested function via call to tty_inter & test_state or directly modify tty_buf
b) return

Figure 1

DN355_Mailbox -Multics

format of dn355_mailbox

datanet_mbx

Word	Fields		
ico-pcw 0		mbx_no	seventy-one
1	special_interrupts		
2	term_inpt_mpx_wd		
3	lock		
4		last_special_count	
5		queued_specials	
6		mailbox_waits	
7		mailbox_wait_time	

format of channel mailboxes

chan_mbx

Word	Fields						
0	dn355_no	is_rs	is_rsa	la_no	slot_no		opcode
1–2	lpw						
3–4	dcw						
5–6	status						
7	ccw				checksum		

above entry repeated 15 times

Figure 2a

DN355_mailbox - NPS Format

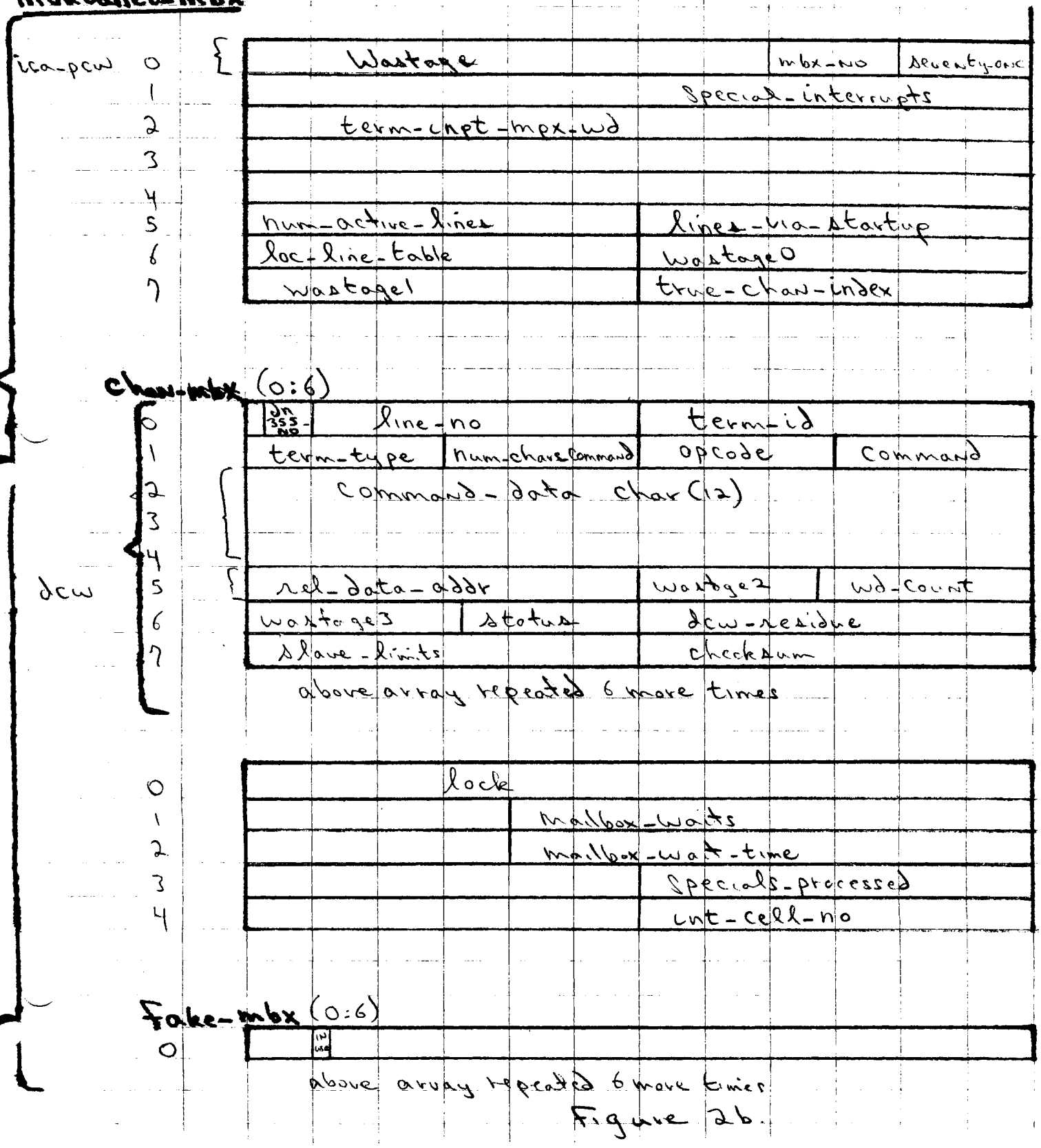


Figure 2b.

DN355_Data -Multics

datanet_info

0		No_of_355s fixed bin
1		dequed_status fixed bin(24)
2		dequeued_cnets fixed bin(24)
3	Fill	
7	above entry repeated 4 times	
deux_tab. 10	355 no | subchan | lano | slot_no	
407	above 18 bit entry repeated 511 times = 256 words	
per_datanet 410-411	mbx_pt = ITS pointer to mailboxes	
412		port_no
413		int_cell_no
414		config_card_corresp
415		com_deux
416		lsla_idx
423	above entry repeated 5 times	
424		lsla_idx
426	above entry repeated 2 times	
427		count (of items in q)
430		cur_ptr (oldest item in q)
431		next_ptr (next item in q)
432		count (of args to list_connect)
433		deux
434		cew
435		lpw
436	dcw	
437		
connect_q 440	above 6 word entry repeated 127 times	
...	above per_datanet entry repeated 2 times	

Figure 3a

dn355_data$ - NPS

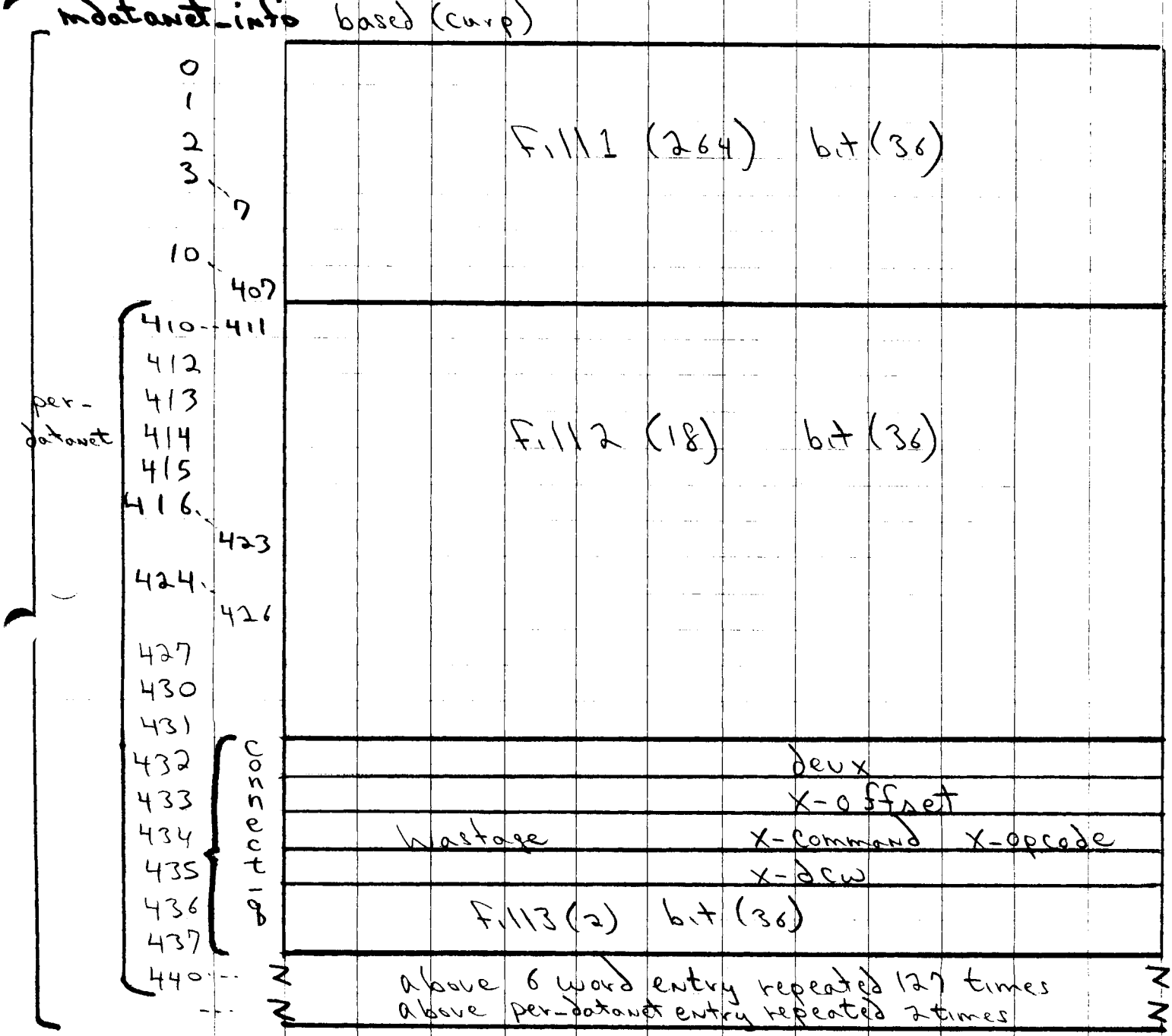


Figure 36

MESSAGES

DN355 To 6XXX

Code	Reason
100	Accept New Terminal
101	Disconnected Line
102	Accept Input
103	Accept end of Current Job
104	Send Initial Output
105	Send Output
106	Send Status
107	Abort
110	Backspace Output
111	Connect to Slave
112	Accept direct input
113	Break
114	Connect to Slave with no Wait
115	Accept Mass Store Job
116	Accept Config Info & Send NS Space
117	Restart Successful
120	Restart Unsuccessful
121	Link(s) Released
122	Old NS Space (No Input List)
123	Old NS Space (with Input List)
124	Accept Mass Store Input List
125	Terminal Characteristics

MESSAGES

6XXX TO DN 355

Code	Reason
0	Terminal Accepted
1	Disconnect this Line
2	Disconnect All Lines
3	NCALL
4	ACALL
5	Input Accepted
6	Terminate Input (Reason)
7	Accept Output
10	Accept final Output
11	Output Not Available
12	Accept Direct Output
13	Accept Direct Output + Wait for Input
14	Accept Direct Output + Receive Paper tape
15	Accept Direct paper tape
16	Reject Request - Temporary
17	Reject Request - Permanent
20	Terminal Rejected - Reason
21	Disconnect Accepted
22	6000 Initialize Complete
23	Terminate Direct Paper Tape
24	Mass Store Job Accepted
25	Accept Mass Store Space
26	Mass Store Space Denied
27	Mass Store Space OK, Continue
30	Accept Free Link
31	Accept Free Links
32	Accept Links in use
33	Input List finished
34	T & D initialization
35	Break Acknowledged
36	Accept Terminal Characteristics
37	Send Userid + password
40	Abort 355

Figure 4-2

Figure 5

	Routine	Requires mods?		Changes made are:
1	dn355.pl1	Yes		Major
2	dn355_init.pl1	NO		N.A.
3	dn355_util.alm	Yes		Minor
4	tty_con.pl1	NO		N.A
5	tty_ctl.ioc	NO	(deleted)	N.A
6	tty_free.pl1	Yes		Medium
7	tty_index.pl1	Yes		Minor
8	tty_init.pl1	NO		N.A
9	tty_inter.pl1	Yes		Major
10	tty_read.pl1	Yes		Medium
11	tty_read_tv.alm	NO		N.A
12	tty_write.pl1	Yes		Medium
13	tty_write_tv.alm	NO		N.A
14	tty_unlock.pl1	NO		N.A
	tty_buf$	Yes		
	dn355_mailbox$	Yes		
	tty_ctl$	NO		
	dn355_data$	Yes		