To CPI

NOS 2.3

FEATURE NOTES

SMD130712

CONTENTS

CHAPTER 1

INTRODUCTION

NOS 2.3 contains many new features and enhancements to the existing NOS 2 features. These new features and enhancements represent Control Data's continuous drive for excellence in this product. The spectrum of new features range from support of the Cyber Supermini and its new peripheral hardware to enhancements for programmers and end users of the computer system. The Tailored Release Process that was introduced at NOS 2.2 Level 605 has been expanded to support additional software product options, and, for the Cyber Supermini model 810-1, a fully configured system. Enhancements have also been made to the NOS Site manuals. The three manuals, Installation Handbook, Operator/Analyst Handbook, and System Maintenance Reference Manual, have been reorganized into four manuals which document the four different tasks associated with a NOS system. They are the Installation, Operations, Analysis, and Administration Handbooks.

The Central Software Support organization of Control Data has put together this document to describe the new features in NOS 2.3. Many of the following chapters contain specific information about the design of the new features and examples of their usage.

(This page left intentionally blank.)

CHAPTER 2

CYBER SUPERMINI OVERVIEW

The Cyber 180 Supermini series consists of the Model 810 and the Model 830. Operating system support for these mainframes is being introduced with the NOS 2.3 release.

The introduction of the tailored release of NOS at 2.2 L605 allows for a quick installation of the NOS operating system. An additional enhancement to this process includes procedures specifically designed for the Supermini customer. These procedures have been written to help in the initial setup of operating procedures such as user validation, permanent file backup, etc.

The following chapters cover the peripheral support and features added to NOS 2.3 for the Supermini system.

- 834 Disk Subsystem
- 639 Magnetic Tape Subsystem
- CC634B Console
- Printer Support Utility (PSU)
- Supermini Procedures

2.1 Model 810

The Model 810 is a compact, air-cooled, microcoded, single central processor with 2M bytes of central memory, 10 PPs and 8 external I/O channels. CIP Level 02 supports this mainframe.

Field upgradeable options include a processor performance upgrade to the equivalence of a Model 830, central memory expansion to 16M bytes, 10 additional PPs and 1 or 2 channel increments (for a maximum total of 16 channels.) Operating power may be 50, 60 or 400 Hertz with a battery backup option available to ride through up to a 3 second loss of power (at which point NOS will checkpoint and step the system).

The I/O channels are of two types: Cyber 12-bit channels and 16-bit Integrated Controller Interface (ICI) channels. The basic IOU configuration consists of 2 ICI channels and 6 Cyber channels. Each of the 2 channel increments add 3 Cyber channels and 1 ICI channel. The ICI channels interface to the low cost peripherals; the 834-11 disk subsystem and the 639-1 magnetic tape subsystem. The IOU also interfaces to the time of day/date clock and the RS232-C direct communication ports for the console and remote maintenance (two port multiplexor).

2.2 Model 830

The Model 830 has about 1.6 times the power of the Model 810 and, with the dual CPU option, the power increases to about 2.9 times that of the 810. The Model 830 has the same basic hardware features as the 810, but has an additional field upgradeable option of a second CPU.

2.3 Peripherals

The Cyber 180 Superminis have their own set of peripherals offering maximum performance and minimum space and power requirements at a low cost. These peripherals are:

- 834 Disk Subsystem
- 639 Magnetic Tape Subsystem
- 533/536 Line Printers supported through the PSU software
- CC634B Console
- 553/556 Remote Print Station supported through RBF.

2.4 Model 810-1

The Cyber 180 Model 810 has been specially packaged with selected peripherals to make up the 810-1 Hardware Package. This package includes:

- Cyber 180 Model 810 Central Processor with 10 PPs, 2M bytes CM, 6 Cyber channels and 2 ICI channels.
- CC634B Console
- 1 834-11 Disk Subsystem with 7255-1 disk adapter.
- 1 639-1 Magnetic Tape Subsystem with 7221-1 tape adapter.
- 1 65K NPU for communications.

(This page left intentionally blank.)

CHAPTER 3

834 DISK SUBSYSTEM

3.1 Introduction

The 834 Disk Subsystem is a disk subsystem which is functionally similar to the 844 and 885 disk subsystems (7155 controller). The formatted capacity for use by NOS is 70 percent that of a double density 844(41/44) disk or 260,480D PRUs. This is equivalent to 167 megabytes (6-bit bytes), 125 megabytes (8-bit bytes), or 16,700,000 60-bit words. The 834 disk supports full tracking only. 834 support was added at NOS 2.2 L605 and was available on the CYBER 170-815/825. At NOS 2.3 L617 the NOS utilities and drivers were extended to support the 834 subsystem on the CYBER 180-810/830. The 834 Disk Subsystem consists of three functional units:

- 834 Disk - An 8-inch drive with nonremovable storage media.
- Control Module (CM) - The intelligent element that drives up to four 834 disks and which can interface with up to two 7255 adapters.

. 7255 Adapter - The logical element between the channel and up to eight control modules which transforms control module functions to make them functionally similar to the 7155 controller protocol.

A mainframe can support a maximum physical configuration of eight 834 disks and eight control modules. One or two cabinets are added to one end of the mainframe with up to 4 834 disks and 4 control modules in each. Express Deadstart Dump (EDD) is able to dump 7255 adapter peripheral microcode (controlware) only. Control module controlware cannot be dumped.

3.1.1 Multimainframe Support

At NOS 2.3 L617 multimainframe via shared RMS support was added for the 834 Disk Subsystem. A second mainframe must be physically positioned contiguous with the 834 cabinet(s) of the first mainframe. The second machine cannot add additional 834 cabinets on its opposite side. Running in multimainframe mode therefore does not allow an increase in the number of 834 drives or control modules which can be supported. When shared RMS is used, the two mainframes share at the control module level. Sharing at the 7255 adapter or drive level is not possible. It is strongly advised that if 834 equipment is to be used in shared RMS mode, then the subsystem should be cabled with one control module per shared device to maximize throughput.

3.2 External Modifications

3.2.1 EQPDECK

Two new equipment mnemonics are defined as follows:

DD - 834 Disk
CM - Control Module

The unit number that NOS associates with an 834 disk is generated as follows:

```
unit number = (control module equipment number)*10B
              +(834 unit number)
```

For example, if the control module equipment number is equal to 7, and the 834 unit number is equal to 2, then the unit number associated with this 834 disk is 72.

The 834 disk is defined in the EQPDECK with the following EQ command:

```
EQord=DD,ST=st,UN=unl/.../unx,AP=ap.
```

Channels used to access the 834 device are specified by the CM EST entry. The EST entry for the control module is used to specify channels to be used to access drives, to specify peripheral microcode load/no load options for the control module, and when granting a customer engineer access to the control module for maintenance purposes (because more than one control module may be on a channel). The CW parameter on the EQ command specifies the load/no load option. If the load option is desired, cc=CM, and if the no load option is desired, cc=NCM. The control module is defined in the EQPDECK with the following EQ command:

```
EQord=CM,EQ=eq,CH=C1/C2,CW=cc.
```

An LBC EQPDECK entry identifies the type of peripheral microcode to be installed in the 7255 adapter. For 834 disks, this entry defines a 7255 adapter for these channels and determines whether or not this peripheral microcode is to be loaded. If the LBC entry is not specified, the system will examine the mnemonics of the device in the EQPDECK entry and cause the default version of peripheral microcode to be installed. Use the LBC entry to override the defaults with the following format:

```
LBC,ID,c1,c2,...,cn.   (7255 peripheral microcode loaded)
        or
LBC,NI,c1,c2,...,cn.   (no 7255 peripheral microcode
                        loaded)
```

Refer to the NOS 2 Analysis Handbook for a more detailed description of these EQPDECK commands.

3.2.2 IPRDECK

Two commands, ENABLE,SPINDOWN and DISABLE,SPINDOWN have been added to the IPRDECK. If spindown is enabled, a CHECKPOINT SYSTEM command will spin down all 834 devices. If spindown is disabled, a CHECKPOINT SYSTEM command will not affect 834 devices. Spindown is enabled by default.

For a more detailed description of these IPRDECK commands, refer to the NOS 2 Analyst Handbook.

3.2.3 DSD Commands

The SPINUP and SPINDOWN DSD commands have been added to enable a Customer Engineer to spin up and down an 834 drive from the console. In order to spin up an 834 drive, it must be powered on. SPINDOWN will also be used by an operator to spin down a drive in emergency situations. The only requirements are that the console be UNLOCKED and at least one channel must be active to the drive in order to use these commands. The format of the commands is:

```
SPINUP,ord.
SPINDOWN,ord.
```

Other DSD commands are available to control the 834 Disk Subsystem. These include the DOWN,ch and UP,ch commands, the ON,ord and OFF,ord commands, and the MOUNT,ord command. ON,ord and MOUNT,ord will automatically spin up 834 disk drives. The AUTO and MAINTENANCE commands will also spin up all 834 disk drives which are ON in the EST.

3.2.4 LOADBC

The LOADBC command, which allows the peripheral microcode to be loaded, has been enhanced to allow peripheral microcode to be loaded to a control module. The format of the command is:

```
LOADBC,EQ=ord,F=lfn.
```

The LOADBC command can also be used to load peripheral microcode to the 7255 adapter as follows:

```
LOADBC,C=ch,F=lfn1,D=lfn2.
```

Refer to the NOS 2 Analyst Handbook for further information about the LOADBC command.

(This page left intentionally blank.)

CHAPTER 4

639 MAGNETIC TAPE SUBSYSTEM

4.1 Introduction

The 639 magnetic tape debuts in NOS 2.3. This hardware subsystem performs the same magnetic tape processing as other tape hardware. It can be used as a NOS deadstart device and is supported in a configuration as the only tape equipment. In fact, the 639 is supported like other tape hardware in a manner that is transparent to the user. The 639 tape unit is recommended for use primarily as a permanent file backup device, not for high volume tape operations.

The 639 keystone tape drive is a 9-track unit with 1600 CPI (PE) and 6250 CPI (GE) recording densities. There is no 7-track support. It operates at 75 IPS in streaming mode, and 25 IPS in start/stop mode. The unit switches automatically between streaming and start/stop modes as required based on the host's ability to maintain the data rate.

4.2 Changes and Differences

The 639 magnetic tape is supported only on CYBER 180 models 810 and 830 mainframes. NOS issues an "INCORRECT ENTRY." error message during deadstart processing if the EQ entry is made on a mainframe model which does not support 639 hardware.

The 639 EQ entry formats are as follows:

The EQ entry for one unit is:

```
EQest=NT,ST=status,UN=un,EQ=eq,CH=ch,TF=IST.
```

The 639 EQ entry for two units is:

```
EQest1=NT,ST=status,UN=un1,EQ=eq,CH=ch1,TF=IST.
EQest2=NT,ST=status,UN=un2,EQ=eq,CH=ch2,TF=IST.
```

NT, NT-1 and MT are all valid and equivalent entries for the EQ parameter. Each 639 device must be connected to an individual Internal Channel Interface (ICI) channel. Up to two 639 drives per mainframe are supported. IST indicates the availability of 639 tape unit hardware feature.

639 has no GE read reverse. For 639 GE, the read reverse operation in 1MT overlays is replaced with backspace, or if the data is required, with a backspace, a forward read, and another backspace.

Previously, reel swapping for multivolume file sets among ATS and Federal Standard Channel (FSC) drives was only supported on devices connected to the same channel. Reel swapping can now be performed on any IST, ATS, and FSC drives, as long as they can handle the requested density.

The NS (noise size) parameter is no longer supported for PE and GE tape requests. If the NS parameter is specified on the LABEL, REQUEST, or ASSIGN control statements for tape formats other than I or SI, noise size requests are ignored and the "NOISE SIZE IGNORED FOR PE/GCR." message is issued.

The "NS" specification is added to the COPY and TCOPY commands to allow noise blocks on a tape to be copied or ignored.

The new order-dependent format for the copy command is:

```
COPY(lfn1,lfn2,x,c,tc,copycnt,bsize,charcnt,erlimit,
po,lfn3,nsc)
```

The new order-independent format is:

```
COPY(I=lfn1,O=lfn2,V=x,M=c,TC=tc,N=copycnt,BS=bsize,
CC=charcnt,EL=erlimit,PO=po,L=lfn3,NS=nsc)
```

Noise blocks are sensed only when processing S to S or L, L to L, or F to F tape copies. Any block containing fewer than nsc characters is considered noise and discarded. A maximum value of NS=41 is allowed. If NS=0 is specified, the default of 18 characters is used.

The new order-dependent format for the TCOPY command is:

```
TCOPY(lfn1,lfn2,format,tc,copycnt,charcnt,erlimit,po,
lfn3,nsc)
```

The new order-independent format is:

```
TCOPY(I=lfn1,O=lfn2,F=format,TC=tc,N=copycnt,CC=charcnt,
EL=erlimit,PO=po,L=lfn3,NS=nsc)
```

Noise blocks are sensed only when processing S format tape for E or B format conversion. Any block containing fewer than nsc characters is considered noise and discarded. A maximum value of NS=41 is allowed. If NS=0 is specified, the default of 18 characters is used.

When using COPY or TCOPY with noise size specified, a double buffering copy scheme is used rather than a single buffer scheme. This causes the copying to be slower.

4.3 Conclusion

The usability of IST tape units remains the same as ATS or MTS tape units. User CIO interface and labeling remain compatible. There is no impact on existing user interface to the tape subsystems except the NS parameter specifications.

(This page left intentionally blank.)

CHAPTER 5

CC634B CONSOLE

5.1 Overview

NOS has been enhanced to support the CDC 721-21 Viking terminal as a local primary console. This new console is CDC product number 18002-2 and consists of 2 parts; the 721-21 terminal (CC634B) with version 4.0 controlware, and the RS232 cable (AV117A). This chapter refers to the new console as the CC634B Console. A local console is one that is located within about 50 feet of the mainframe, connected via a direct cable connection, and a primary console is the one that deadstarts the system. The CC634B console is supported through the 2-port mux of a Cyber 180-810 or 830 on channel 15. For acceptable usability and performance of the CC634B console the baud rate must be set to 19.2K.

Both DSD and DIS have had major changes to their displays. This affects not only CC634B but also CC545 (the pre-NOS 2.3 standard console) console operators.

This chapter will cover the different aspects of the CC634B console and its support in the following order.

- Section 5.2 - General changes in the deadstart, OS and display areas.
- Section 5.3 - CC634B console keys.
- Section 5.4 - Display generator/CC634B communication overview.
- Section 5.5 - CC634B peripheral microcode.
- Section 5.6 and 5.7 - Appendix covering the internals of the System Console Driver (SCD) and the 2 port mux access protocol.

5.2 General Changes

5.2.1 CC634B Console vs. CC545 Console

Several differences between the CC634B and the CC545 consoles are:

- The CC634B console does not have the special power requirements that a CC545 console has, making it suitable for office environments.
- Each console has its own screen size. Reference Figure 3 for a description of the CC634B console screen format.
- The refresh rate of the first 27 lines for the CC634B console is about once per second as opposed to the 40-50 times per second on the CC545. The bottom 3 lines for the error message and the keyboard input are more responsive than the rest of the display.
- Cursor movement appears on the CC634B keyboard entry line.

- Only small characters are supported on the CC634B console.
- 026 is not currently supported on the CC634B console.
- The game programs; ADC, BAT, DOG, WRM, CHD and EYE can not be run on the CC634B console.

5.2.2 Deadstart

The CC634B console needs to be set up initially for use as a console. The initialization procedure can be found in appendix I of the CIP Handbook (pub #60457180, revision B). Once it is set up, future initializations are performed by pressing the RESET button on the console. The port option switch on the mainframe for port 0 must be set to "DS ENABLED" or "DS & PWR ENABLED" to allow deadstarting from that port.

Deadstart is initiated on a CC634B console by the following sequence:

```
CONTROL-G
System responds:  OPERATOR ACCESS ENABLED
CONTROL-R
```

CTI determines the type of console that is performing the deadstart and passes this information to the operating system.

5.2.3 Displays

Many changes have been made to the DIS and DSD displays at NOS 2.3 for both console types. These changes were implemented to improve usability of the displays and to accommodate the screen size of the CC634B console. Major changes to these displays include:

1. The DSD W,C display contains the channel status which was removed from the right screen header.

2. The DSD W,R display contains system resource information such as the next JSN, the JSN in the CPU, available resources and table addresses.

3. The DSD W,Q display contains the system queues which had formerly been found in the W-display.

4. The contents of the old DSD B-display have been split into 2 new displays: B,A and B,O. The B,O display (default B-display) contains the information primarily intended for use by an operator, such as the JSN and status message of each control point. The B,A display is intended for an analyst's use and contains all of the old B-display information except the messages.

5. The DSD E,F display contains family status information, in particular the indirect and direct access file masks (formerly in the E,C display) and the family job count and direct access files attached count (formerly in the E,M display).

6. The DIS B-display has been split into two displays: a new B-display and a J-display. The new B-display contains the breakpoint address and exchange package and can only be brought up on the left screen. Incrementing and decrementing the breakpoint address is accomplished with the + and - characters. The J-display contains the job status information and the current message and command buffers. The J-display is the DIS right screen default display.

7. When displaying data using the DIS T, U or V-displays on a CC634B console, the data will be checked for the presence of Y-coordinates outside the range of 7702 to 7062. Any Y-coordinates outside of this range will be changed to 7702, the sixth line of the display. This will prevent data from being written in the header or trailer areas.

8. The SUBSYST L-display and the MODVAL, PROFILE, FLAW and INITIALIZE K-displays have been reformatted to use 23 lines for their displays with the last two lines containing the error message line and the keyboard input line.

9. The K and L-display interface with a CPU program does not allow the CPU program to attract an operator's attention on a CC634B through intensification, inverse video, blinking, etc. A site written K-display may need to be changed if it was attempting to flash a message to the CC634B operator.

10. Site written K and L-displays may need to be rewritten for the CC634B console. These displays are now restricted to use the display portion between Y-coordinates 7714 and 7062. This will be enforced by DSD when run on a CC634B console. If a K or L-display generator tries to write outside this area, the Y-coordinate will be changed to 7714 when output to the display.

11. Another common problem in running K and L-displays on the CC634B console is long lines. The old DSL macro and other K and L-display macros blank fill the last word of a line. A long line typically fills 7 words and this causes the line to wrap around and blank out the beginning characters of the line on a CC634B. The new DSL macro solves this problem by zero filling a line. The CC634B controlware treats a zero as a skip character. So, recompilation using the new version of the DSL macro may be all that is required to get site written K and L-displays working on a CC634B console.

12. The MODVAL K-displays have been changed to include forward and backward paging on both left and right screens through the use of the +, -, (, and) keys.

5.2.4 CMR

The RCLP word (143B) has also been defined as SCDP. Bytes 2 and 3 of this word contain the first word address (FWA) of the SCDPT (SCD Parameter Table).

The SCDPT is a 4 CM word table that contains parameters required for SCD to support the CC634B console. This table is defined in the section on SCD.

The SCD CM buffer allocated immediately after the SCDPT is used to store non-priority output. The size of the buffer is determined by the COMSMSC symbol, SCDCL, with a default size of 1400B. This buffer is only allocated when using a CC634B console.

The EST entry for the system console has been modified to include, in bits 35-24, the terminal type of the console (0-CC545, 1-CC634B), and in bits 2-0, the port number for the console. The EQPDECK does not need a console entry as the EST entry is set up with information passed from CTI to NOS.

5.2.5 Common Deck Changes

Many changes have been made to the common decks. The changes include:

1. COMCDCM is a new common deck that contains the following macros used by K and L-displays. It has been added to PSSTEXT.

   - CSTATUS (console status) provides a user with program information regarding the type of console.
   - DSDOUT has been moved from COMCMAC and contains an optional parameter (I) to indicate that the specified address is an indirect address.
   - KDL (K-display line) is a new macro that replaces the DSL macro for K-displays. It generates a display line containing the X and Y-coordinates for the CC545 and also generates a remote block that contains CC634B console Y-coordinates to be used in the display buffer.
   - PAGT (create page table) can be used to create the page table which contains the FWA of each page of the K or L-display.
   - CONSOLE, DSDINP and DSL have also been moved from COMCMAC to COMCDCM.

2. COMCDCP is a new common deck that contains the following subroutines used by K and L-display generators.

   . PRK (preset K-display) is called to preset the display buffers.

   . PGD (page display) is called to process paging commands for K and L-displays.

3. COMCFLD has a new entry point called FLB (format L-display buffer). It performs the same function as FLD, but changes the Y-coordinate increment from 10D to 15D for the CC545 console. This increment change provides better spacing of the lines on the CC545 for displays in which the number of lines has been decreased to accommodate the CC634B console.

4. COMDSYS routine SYS has had its exit conditions changed to leave channel 10 active with the right screen and small characters selected.

5. COMCMAC has had macros CONSOLE, DSDINP, DSDOUT and DSL removed. These macros have been added to COMCDCM. All programs using these macros must be modified to call the new common deck if they are reassembled using a 2.3 OPL, or else use PSSTEXT.

5.2.6 LIBDECK Change

It is no longer necessary to specify (in the LIBDECK) the CM residency of DSD overlays which must be CM resident. All overlays in the range of 9CA - 9C9 are automatically made CM resident by SYSEDIT.

5.3 CC634B Console Keys

Paging control keys for DSD are:

+ Page left screen forward. If it is at the last page, it displays the first page.

- Page left screen backward one page or back to the first page depending on the current display. If it is at the first page, no change is made.

(Page right screen forward as above.

) Page right screen backward as above.

The HELP key presents a menu of the special keys for a CC634B console.

The Control-G and Control-R are used in the deadstart sequence. A Control-I causes SCD to reload the controlware augmentation package and reinitialize the CC634B console. A Control-I will not cause any harm to NOS. These are the only control keys in use for the CC634B console.

Keys that are deactivated on the CC634B console include the following. The bell will sound when these keys are entered.

:/;	CR/DEL	DATA
"/'	INSRT	STOP
}/{	DLETE	SETUP
|/\	CLEAR	BACK
]/[	LF/ESC	LAB
~/`	TERM/ANS	COPY
F6 thru F12	SHIFT LOCK	EDIT

The shift key only works for the shifted characters: *) (and +. This makes the following characters unavailable on a CC634B console.

! @ # $ % ^ & _ ? < >

The F1 through F5 keys control the screen display on the CC634B console. An operator can select which portion of the

hidden screen buffer will be displayed by pressing the appropriate function key. Reference Figure 3 for a detailed description of the CC634B console screens.

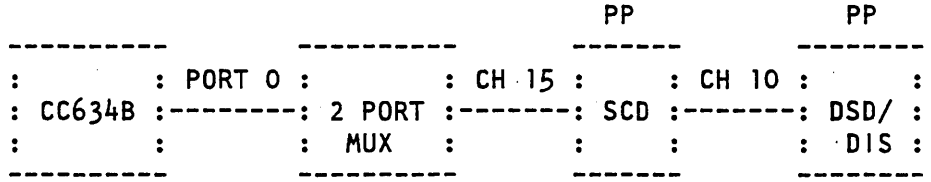
- F1 - Toggle upper or lower left screen display. *
- F2 - Select left screen display only. This is a 30 line by 64 character display left justified in 80 column mode.
- F3 - Select dual screen display. Two displays of 30 lines by 64 characters in 132 column mode with 4 blanks separating the displays.
- F4 - Select right screen display only. This is a 30 line by 64 character display in left justified 80 column mode.
- F5 - Toggle upper or lower right screen display. *

* The F1 and F5 keys provide a way to toggle between the upper and lower pages for the appropriate screen. This allows viewing of the larger CC545 console displays which have not been modified to conform to the CC634B console screen size, such as unmodified site written K and L displays.

If a printer is configured on the CC634B console, the PRINT key causes all data from the top of page to end of page to be printed as it appears on the screen.

5.4 Display Generator/CC634B Communication Overview

A general diagram of the communication between the CC634B, SCD and DSD/DIS is:



DSD/DIS pass octal function codes and display coded data to SCD over channel 10. This information is analyzed by SCD to determine if it is data or a function code. Data is converted by SCD to ASCII and sent on to the console via channel 15 and the 2 port mux. Octal function codes cause SCD to send the appropriate hex function code to the CC634B console controlware via channel 15 and the 2 port mux. The augmented controlware package in the CC634B console processes the hex function codes and data for the CC634B console.

Information coming from the CC634B console to DSD/DIS is handled in a similar fashion. The section on the CC634B controlware and the SCD appendix explain this communication in further detail.

5.5 CC634B Peripheral Microcode (Controlware)

An augmented CC634B console peripheral microcode (controlware) package has been developed for NOS 2.3. This enhanced controlware package will make the CC634B behave in a manner similar to the CC545 console. The controlware is driven by data sent via the two port mux by SCD, the driver for the CC634B console. SCD is a dedicated PP and is released in binary format on the CIP tape. The augmented controlware is part of the SCD record on the CIP tape.

The CC634B console is driven by the controlware which processes the functions and data stream received from the host in addition to operator keyboard entries. Host loadable controlware (i.e., the controlware augmentation package) modifies the CC634B resident and adds or enhances the capabilities defined in the following sections.

The data stream from SCD to the CC634B console consists of display character codes, display coordinate codes and function codes. The data stream from the CC634B console to SCD consists of keyboard character codes and X-on/X-off codes.

5.5.1 Character Codes

Keyboard Character Codes

Figure 1 identifies the character and display codes received from the CC545 keyboard along with the characters from a CC634B keyboard. Lower case CC634B characters are folded into upper case characters by the controlware.

CC634B Char.	Displ Code	CC545 Char.	CC634B Char.	Displ Code	CC545 Char.
A	01	A	0	33	0
B	02	B	1	34	1
C	03	C	2	35	2
D	04	D	3	36	3
E	05	E	4	37	4
F	06	F	5	40	5
G	07	G	6	41	6
H	10	H	7	42	7
I	11	I	8	43	8
J	12	J	9	44	9
K	13	K	DOWN	45	+
L	14	L	+	45	+
M	15	M	UP	46	-
N	16	N	-	46	-
O	17	O	F15	47	*
P	20	P	*	47	*
Q	21	Q	/	50	/
R	22	R	FWD	51	(
S	23	S	(	51	(
T	24	T	BKW	52	)
U	25	U	)	52	)
V	26	V	¦<-- (1)	53	LB
W	27	W	ERASE	53	LB
X	30	X	=	54	=
Y	31	Y	-->¦ (2)	55	RB
Z	32	Z	,	56	,
			.	57	.
			NEXT	60	CR
			<-- (3)	61	BS
			(SP)	62	SP

(1) This is the forward tab key.
(2) This is the backward tab key.
(3) This is the backspace key.

FIGURE 1

Display Character Codes

The character displayed on a CC634B console for each display code is identified below.

Char. Displ'd	Displ Code	Char. Displ'd	Displ Code
	00	0	33
A	01	1	34
B	02	2	35
C	03	3	36
D	04	4	37
E	05	5	40
F	06	6	41
G	07	7	42
H	10	8	43
I	11	9	44
J	12	+	45
K	13	-	46
L	14	*	47
M	15	/	50
N	16	(	51
O	17	)	52
P	20	Space	53
Q	21	=	54
R	22	Space	55
S	23	,	56
T	24	.	57
U	25		
V	26		
W	27		
X	30		
Y	31		
Z	32		

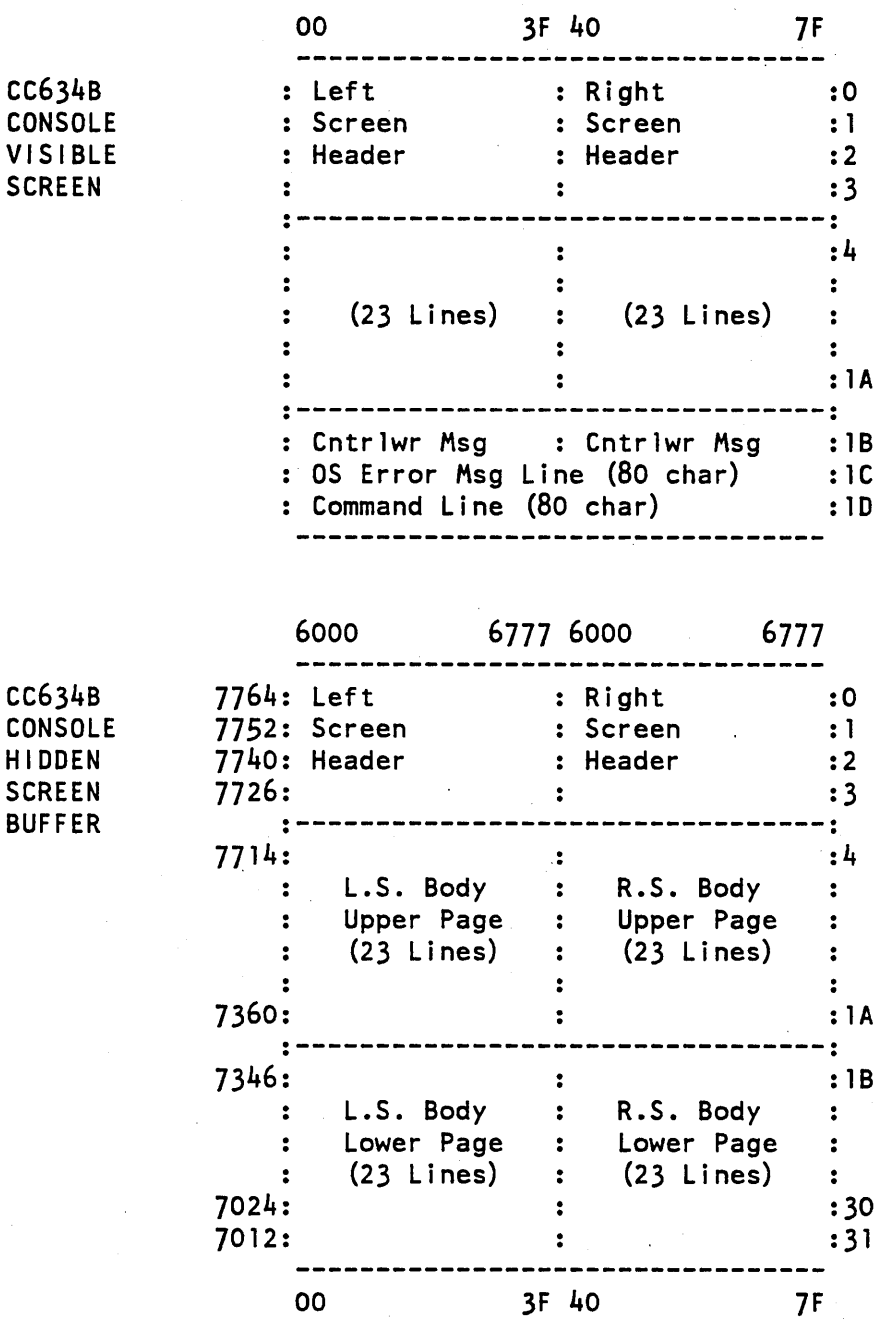
* A 00 code will result in the display character position being skipped.

FIGURE 2

5.5.2 Display Coordinate Codes

The CC634B console controlware augmentation supports two types of screens, a visible screen and a hidden screen buffer (reference Figure 3). All data is entered in one of the selected screen lines based upon the current X and Y-coordinate. Subsequent characters increment the X-coordinate automatically, one unit per character, across the line. At the right edge the X-coordinate is reset to the left edge and the Y-coordinate is unchanged.

The hidden screen buffer serves as an interface between SCD and the visible display console screen. This buffer, which is 20 lines larger than the CC634B console visible screen, provides a facility through which existing (unmodified) CC545 display screens can be supported. Each character to be displayed requires one byte for a character display code and one byte for a display attribute code. Therefore, the hidden screen buffer contains a total of 13,200 bytes ((2 bytes/character) (50 lines) (132 characters/line)). The Command Line (1D) and OS Error Msg Line (1C) on the visible screen are both 80 characters in length. The Cntrlwr Msg line (1B), which is 64 characters in length, appears at the bottom of both the left and right screen.



Note: The 4 digit octal numbers represent CC545 positioning.

FIGURE 3

APPENDIX TO CC634B CONSOLE

5.6 System Console Driver (SCD)

The general functions of SCD are:

- Receive function codes from channel 10.
- Receive display screen coordinates for subsequent output characters.
- Receive display coded data from channel 10, convert it to ASCII and send it to the console.
- Monitor and honor RDF and MDD requests for access to the two port mux.
- Send keyboard data when requested by the display generator.

5.6.1 CTI - SCD Initialization Interface

Upon determining that the primary console is a CC634B console, CTI performs the following steps before initiating NOS:

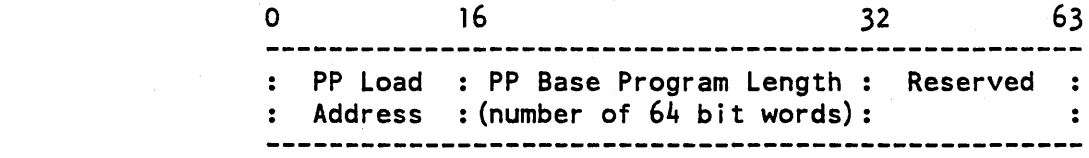
1. Store the CC634B console controlware augmentation package and SCD in the CM buffer reserved in EI at CTI's request. This controlware is read from the Common Disk Area on the system disk.

2. Identify the PP which contains the SCD program. This information is included with the response for the "Send Display Console Identification Data" function. The SCD PP must be other than 0, 1, 2, 10 or the PP on the deadstart channel. The PP into which SCD is loaded must be designated as logically "OFF" in the Hardware Descriptor Table (HDT) by CTI. The Fault Status Mask (FSM) bit will be set for this PP when CTI initializes the FSM register in the IOU.

3. Load a bootstrap program within the first 30B words of the SCD PP which enables it to load the SCD program from a CM buffer and then return itself to the deadstart load state.

4. Load the default parameters into words 30, 31, 32 and 33B of the SCD PP.

5. Activate the SCD bootstrap program and wait for SCD to enter its deadstart load state. The bootstrap reads the first CM word in the SCD package to obtain the information it requires for loading SCD. This CM word is located through the parameters in words 30-32 initialized above. The format of the first CM word is as follows.



The bootstrap loads the base program, starting at the load address, into itself before entering a deadstart load state.

6. Ensure that channel 10 does not have a "selected" CC545 by issuing a de-select function code.

The default parameters specified by CTI are as follows:

. SCD Words 30, 31 and 32 (48 bits)

These words contain the PP R-register value and the CM word offset from this value, which is used to locate the SCD package in central memory. This package contains the SCD image and the CC634B controlware augmentation.

- Word 30: CM word offset used to locate the first CM word of the SCD package.

- Word 31: The least significant 10 bits in this word are the most significant 10 bits of the R-register.

- Word 32: The least significant 12 bits in this word are the least significant 12 bits of the R-register.

. SCD Word 33 (16 bits)

This word contains the primary console definition in the following format:

```
48              63
------------------
:xxxxasst:ttcccppp:
------------------
```

```
xxxx = Reserved
a    = Active primary console flag
ss   = System state driving primary console
       1 = DSD
ttt  = Terminal type
       1 = CC634B
ccc  = OS/SCD Protocol
       0 = Controlware not required
       1 = CC545
ppp  = Port number of primary console
```

5.6.2 OS - SCD Initialization Interface

Since the SCD PP is left in the deadstart load state, it can be reclaimed as a pool PP in the same manner that other PPs are assigned as pool PPs. NOS ensures that the CC545 on channel 10 (if configured) is de-selected prior to activating SCD.

5.6.2.1 SCD Parameters Obtained Via I/O Channel

To function as the console driver, SCD must be initialized by NOS passing it the appropriate parameters. The initial set of parameters are passed via the output of 3 PP words prior to activating SCD through a DCN of its associated channel.

When SCD is activated, if parameter words 2 and 3 are zero, the default parameters passed by CTI are used by SCD to support the primary console.

- Parameter Word 1 (16 bits)

  This word contains the FWA-1 of SCD's initialization routine. The value is 77B.

- Parameter Words 2 and 3 (32 bits)

  These words contain the first word address of the SCD Parameter Table which is used by SCD to access additional parameters.

  48 63

  ```
  ------------
  : 1st word :  - Upper 12 SCDPT address bits
  ------------
  : 2nd word :  - Lower 12 SCDPT address bits
  ------------
  ```

  If these two words are zero when passed to SCD, the location of the table (SCDPT) can be defined by issuing the select SCD initialization function.

5.6.2.2 SCD Parameters Obtained Via Central Memory

The SCD Parameter Table (SCDPT) contains parameters required for SCD to support the CC634B console and it identifies SCD operational changes in the post deadstart timeframe. The SCDPT is built in central memory and located by SCD through the contents of deadstart parameter words 2 and 3 or by receipt of the select SCD initialization function.

The SCDPT contains four CM words. All parameter fields designated as "Reserved" are initialized to binary zeroes when the SCDPT is built by NOS. This table has the following format:

```
   59 56    47         31          15        0
   ---------------------------------------------
1  :        :SCDPT Sta-:Primary Con.:Reserved :
   :Reserved:tus Byte  :Definition  :for CDC  :
   ---------------------------------------------
2  :I/O Interface      :                      :
   :CM Address         :Reserved for CDC      :
   ---------------------------------------------
3  :                                          :
   :Reserved for CDC                          :
   ---------------------------------------------
4  :  :Rese-:Buffer    :                      :
   : 2:rved :Length    :SCD Buffer FWA        :
   ---------------------------------------------
```

WORD 1 (Console Definition Word)

- Bits 59-48: Reserved

- Bits 47-32: "sxxxxxxxxxxx1111"
  s = When set it denotes a change in one or more of the definition words used in this table to support the primary console.
  NOTE: The convention for using this bit is:
  - The display generator builds new SCDPT which includes changed parameters.
  - The change is denoted by the S bit being set.
  - The display generator issues the select SCD initialization function.
  - SCD clears the bit upon completing its initialization.

x-x = Reserved.
1111 = Length of SCDPT expressed in number of CM words, = 4.

. Bits 31-16: Primary Console Definition Byte
"xxxxasssttcccppp"
xxxx = Reserved
a = Active primary console flag
ss = 1 (Values 0, 2 and 3 are reserved)
ttt = Terminal type
0 = Not available for terminal assignment
1 = CC634B
2-7 = Reserved
ccc = OS/SCD protocol
0 = Controlware not required
1 = CC545
2-7 = Reserved
ppp = Port number for primary console

. Bits 15-0: Reserved.

WORD 2 (Primary Console Interface Definition)

. Bits 59-32: I/O Interface CM Address
. CM word address of zero = I/O Channel 10
. Non-zero CM word address = Undefined

. Bits 31-0: Reserved.

WORD 3 (Reserved)

WORD 4 (Supportive SCD Definition Word)

. Bits 59-57: Code ID for Supportive Word
0 = No entry in this word
1 = Reserved
2 = SCD's CM Buffer for Primary Console
3-7 = Reserved

SCD's CM Buffer For Primary Console Definition Word (Code ID = 2):
. Bits 56-48: Reserved

- Bits 47-32: Number of CM words in buffer (0 means no buffer available)
- Bits 31-0: CM FWA of buffer used by SCD while transferring non-priority display information to the primary console.

5.6.3 SCD - Channel 10 Interface

When SCD operates in its active console driver state its main loop consists of checking for the following:

- Is "Full" status present on channel 10?
- Is "Input Ready" status present in 2 port mux?
- Is priority output operation in progress?
- Does the SCD CM buffer (if defined) contain information for the CC634B console?

When SCD senses channel 10 "Full", it performs an input of one channel word (16 bits), which is either a function or data, and leaves the channel in an "Active" state. (The display generator function codes are defined in detail in section 5.6.5.)

When SCD senses "Input Ready" in the 2 port mux it inputs the character code and retains it until receipt of a select keyboard input (160400) function from the display generator. If SCD does not receive this function before additional keyboard characters are input by SCD, the succeeding characters will be discarded.

If SCD is processing a priority output operation, it suspends the transfer of any remaining information in its CM buffer to the CC634B console until the priority operation is complete. The priority output consists of SCD transmitting display information directly (bypassing its CM buffer) to the 2 port mux. This means that the display generator's output transfer rate is governed by the 2 port mux.

SCD stores non-priority output information from the display generator into the buffer concurrent with its removal of

information from the buffer for transmission to the 2 port mux. Expediting the display generator's output operation in this manner enables it to be more responsive in its processing of keyboard input. If "Full" status is not present when SCD senses for it, it continues processing the remainder of its control loop.

Two controlware functions are available from SCD to handle priority output. These functions are used by several of the screen and keyboard input function codes described later. While doing priority output, take caution in using screen manipulating functions (such as 010020, 010040, 010060 or 010100 functions) because the status of the SCD CM buffer may be unknown.

. Activate Priority Output Display Parameters

Receipt of this function directs the controlware to save the non-priority output display parameters and activate the priority parameters. These parameters control current attribute and position coordinate information which is used to support a priority output mode of operation. This means that all data and attribute information received during line selection is directed as follows:

DSD Y-Coord	SCD Y-Coord	Screen Area (Line #)	CC634B Screen	CC634B Y-Coord
7764-7726	00-03	Header (1-4)	Hidden	00-03
7714-7360	04-1A	Body Upper Page (1-23)	Hidden	04-1A
7346-7024	1B-30	Body Lower Pg (24-45)	Hidden	1B-30
7012	31	OS Error Msg	Visible	1C
7000	32	Command Line	Visible	1D

. Activate Non-Priority Output Display Parameters

Receipt of this function directs the controlware to save the priority output display parameters and activate the non-priority parameters. The activated parameters contain current attribute and position coordinate information which is used to support a non-priority role of operation. This means that all data and attribute information received during line selection is directed as follows:

DSD Y-Coord	SCD Y-Coord	Screen Area (Line #)	CC634B Screen	CC634B Y-Coord
7764-7726	00-03	Header (1-4)	Hidden	00-03
7714-7360	04-1A	Body Upper Page (1-23)	Hidden	04-1A
7346-7012	1B-31	Body Lower Pg (24-46)	Hidden	1B-31
7000	32	Command Line	N/A	N/A

All position information received by the controlware during selection of SCD Y-coordinate address 32 is discarded in this mode of operation. Data and attribute information received will be placed in the last valid line number.

5.6.3.1 Keyboard Input Operation

This operation is initiated by issuing the select keyboard input (160400) function. SCD responds with a "00" code to the display driver if no character was entered or with a non-zero code if a character was entered.

Receipt of this function denotes completion of the previous "16xxxx" function processing for the display generator. If SCD was processing the transfer of non-priority output from its CM buffer at the time this function was received, SCD does not respond prior to its buffer being emptied unless a non-zero keyboard code was received from the CC634B console. Therefore, when a "00" code is returned following a non-priority output, it signifies that the entire transfer to the hidden screen has been accomplished and there is no input character available.

5.6.3.2 Display Operation

Initiation of a display operation requires the display generator to issue one of the four "16xxxx" output functions. The SCD display operation is defined as the transfer, to the CC634B console, of all channel 10 information received between an output function and the next

"16xxxx" function. The channel 10 information consists of display coordinates, character codes "17xxxx" functions and "01xxxx" functions. If this information (excluding "17xxxx" functions) was received by SCD prior to receipt of a select output function, it would be discarded (i.e., processed as a no-op).

A typical display sequence for the display generator consists of the following:

- Select initialize hidden screen (010020)
- Transfer contents of first screen
- Select keyboard input (160400)
- Transfer contents of second screen
- Select transfer of hidden to visible (010060)
- Select keyboard input
- Repeat sequence.

5.6.4 Channel 10 Access Control

SCD recognizes information on the display channel only if it has not been requested to set the channel flag and terminate its I/O on the channel. The following protocol describes how to trigger SCD to change its operational state.

1. Initially the channel 10 channel flag is clear and the channel is inactive.
2. The display generator (DSD for example) reserves and activates the channel and then proceeds to perform I/O operations with SCD.
3. When the display generator wants SCD to leave the channel, it issues the "set channel 10 channel flag" function code (170040).
4. When SCD processes the function code, it sets the flag and suspends I/O operations on channel 10

until it senses the channel flag returning to a clear state.

5. The display generator (after the function was accepted) goes on to do whatever was desired.

6. When the display is wanted again, the display generator clears the channel flag to reconnect the driver.

The ability to switch SCD on and off the channel is needed to allow switching of generator programs and use of the display channel to load programs from a helper PP (such as when 1DL loads display overlays).

5.6.5 Channel 10 Function Codes

SCD analyzes each 16 bit word received from channel 10 to determine if it is function or data. SCD considers any channel word containing a non-zero value in the most significant four bits to be a function code, not data. Invalid bit positions are ignored by SCD.

Common deck, COMDMAC, contains symbol definitions for the function codes documented below and macros for use by system console display drivers.

All function codes from channel 10 are in octal.

5.6.5.1 Screen, Keyboard Input and Clear Function Codes

16 Bit Function Code	COMDMAC Symbol	12 Bit Function Code	Function Description
160000	SLSF	7000*2**4	Select left screen, 64 char
160100	SLPF	7004*2**4	Select left screen, 64 char, priority output
162000	SRSF	7100*2**4	Select right screen, 64 char.
162100	SRPF	7104*2**4	Select right screen, 64 char, priority output

160400	SKIF	7020*2**4	Select keyboard input
170020	SSIF	7401*2**4	Select SCD initialization
170040	SDCF	7402*2**4	Set channel 10 channel flag
010000	EELF	0400*2**4	Erase to end of line
010020	CHSF	0401*2**4	Initialize hidden screen and display parameters. When this function is issued, it applies to both the left and right screens.
010040	RTPF	0402*2**4	Initialize display page selection
010060	THVF	0403*2**4	Transfer hidden to visible screen. When this function is issued, it applies to both the left and right screens.
010100	THCF	0404*2**4	Transfer hidden to visible and initialize hidden screen and display parameters. When this function is issued, it applies to both the left and right screens.

. Select Left Screen (160000)

This function directs SCD to accept "01xxxx" functions and data for the left screen in the hidden screen buffer until a different "16xxxx" select function is received.

The CM buffer is used to store information from channel 10 at SCD's maximum input rate while it concurrently transmits to the 2 port mux. The transmission of information from the CM buffer is suspended during the time required for SCD to perform a priority output (160100 or 162100).

. Select Left Screen, Priority Output (160100)

This function directs SCD to accept "01xxxx" functions and data for the following screen areas until receipt of a different "16xxxx" select function:

- The first 49 lines of the left screen in the hidden screen buffer.

- The last two lines of the visible screen.

The CM buffer is not used to support execution of this function.

- Select Right Screen (162000)

  This function directs SCD to accept "01xxxx" functions and data for the right screen in the hidden screen buffer until a different "16xxxx" select function is received.

  The CM buffer is used to store information from channel 10 at SCD's maximum input rate while it concurrently transmits to the 2 port mux. The transmission of information from the CM buffer is suspended during the time required for SCD to perform a priority output (160100 or 162100).

- Select Right Screen, Priority Output (162100)

  This function directs SCD to accept "01xxxx" functions and data for the following screen areas until receipt of a different "16xxxx" select function.

  - The first 49 lines of the right screen in the hidden screen buffer.

  - The last two lines of the visible screen.

  The CM buffer is not used to support execution of this function.

- Select Keyboard Input (160400)

  This function directs SCD to return a keyboard keycode in a word output to channel 10 in the format defined below. This function must be followed by another 16xxxx function.

```
 48     56  57 58  63
---------------------
:000000000: d :cccccc:
---------------------
```

cccccc = Keyboard character in display code
d = Display refresh request

The "Display Refresh Request" bit indicates that SCD has detected one of the following conditions which requires that the host refresh the display.

- Loss and re-acquisition of "Carrier-On" status.
- A Control-I, F1, F2, F3, F4, F5 or HELP keyboard character was selected by the operator.

By convention, no response is returned for this function unless one of the following conditions is present:

- The code to be returned is non-zero (i.e., character was entered).
- The code to be returned is zero and the CM buffer is either not defined or empty (i.e., no character was entered and all previous output operations have been completed).

. Select SCD Initialization (170020)

This function directs SCD to accept the next two channel words for use in determining what parameters must be reinitialized. These words define the first word address of the SCD Parameter Table which is used by SCD to reinitialize itself.

```
48        63
------------
¦ 1st word ¦  - Upper 12 SCDPT address bits
------------
¦ 2nd word ¦  - Lower 12 SCDPT address bits
------------
```

Once the location of the SCDPT is known to SCD it examines the first word in the table to determine what must be reinitialized.

- Set Channel 10 Channel Flag (170040)

  This function directs SCD to set the channel flag for channel 10 and then to wait until the channel flag is cleared before continuing to transmit or receive data across channel 10.

- Erase to End of Line (010000)

  This function directs the controlware in the CC634B console to erase all characters from and including the current X-coordinate to the end of the selected line. A code of 20(16) is entered into the erased character positions and all attribute bytes associated with these characters are reinitialized.

  If this function is received immediately after receipt of a "00" display code, the controlware will also backspace one character position and erase from that point up to and including the last character position on that line. In addition, if that line was the Command Line, the cursor will be positioned at the character position from which the erase was initiated.

- Initialize Hidden Screen and Display Parameters (010020)

  Initialization consists of the controlware setting the display character fields to a 20(16) in the hidden screen buffer and setting the priority and non-priority display parameters to their default value.

- Initialize Display Page Selection (010040)

  Receipt of this function directs the controlware to initialize the page selection status for the currently selected left or right screen in the hidden screen buffer to its default state (i.e., display upper page).

- Transfer Hidden to Visible Screen (010060)

  This function directs the controlware to move the display characters and their associated attributes from the selected hidden screen (left, dual or right) to the

visible screen. When transferring from the hidden screen to lines in the visible screen with Y-coordinates of "03" and "1A", the controlware sets the underscore attribute for each character in these lines.

. Transfer Hidden to Visible and Initialize Hidden Screen and Display Parameters (010100)

This function directs the controlware to perform functions "transfer hidden screen to visible screen" and "initialize hidden screen and display parameters" in consecutive order.

5.6.5.2 Terminal Attribute Function Codes

Highlighted fields (blinking, intensified) may be displayed using the underscore, blink, reduced intensity and inverse video capabilities of the terminal. These attributes may be used singly or in combination.

16 Bit Function Code	COMDMAC Symbol	12 Bit Function Code	Function Description
010140	BIPF	0406*2**4	Start inverse and underscore fields one character position before next data byte
010160	EITF	0407*2**4	End inverse and underscore fields at next character position
010200	BUSF	0410*2**4	Start underscore field
010220	EUSF	0411*2**4	End underscore field
010240	BBLF	0412*2**4	Start blink field
010260	EBLF	0413*2**4	End blink field
010300	BRIF	0414*2**4	Start reduced intensity field
010320	ERIF	0415*2**4	End reduced intensity field
010340	BIVF	0416*2**4	Start inverse and underscore fields
010360	EIVF	0417*2**4	End inverse and underscore fields

These function codes direct SCD to insert the appropriate select/de-select hex function code for these attributes in the output data stream for the CC634B console. It is acceptable to issue an "end" function before a "start" function. It is also acceptable to issue these functions either immediately before or after a coordinate code. These functions should be viewed as a bracket (e.g. all screen areas written into between a "start" and "end" attribute function has that attribute selected).

5.6.5.3 State Verification Function Code

A special function code (expressed in octal values)

010400 - no-op (0420 * 2**4)

can be used by a display generator to verify that SCD is in the correct state to accept functions and data for display on a CC634B console. The channel status changing from "Full" to "Empty" after issuing this function denotes that SCD is in this state.

5.6.6 Channel 10 Data Stream To SCD

The data stream consists of display characters and X and Y-coordinates. This information is packed in the right most 12 bits of each channel word. Display codes in the range 00B to 57B (see Figure 2) comprise the total character set sent to SCD. The 00B code is treated as a "skip current character position" function and will not necessarily result in the same visual effect seen on a CC545.

These display codes are converted by SCD to their upper case ASCII equivalents prior to transmitting them to the CC634B console.

Coordinate addressing within the CC634B console hidden screen buffer is compatible with the addressing of a CC545 screen. That is, X-coordinates in the range of 6000B to 6777B are converted by SCD to their CC634B equivalent coordinate which is as follows:

Left Screen X-Coordinates		Right Screen X-Coordinates	
CC634B	CC545	CC634B	CC545
00	6000	40	6000
01	6010	41	6010
.	.	.	.
.	.	.	.
.	.	.	.
3F	6770	7F	6770

This corresponds to 64 character positions in each screen, using the standard eight-unit increment. Since the CC634B console does not have finer than character unit addressing, X-coordinates are forced into their left equivalent if they fall between character positions.

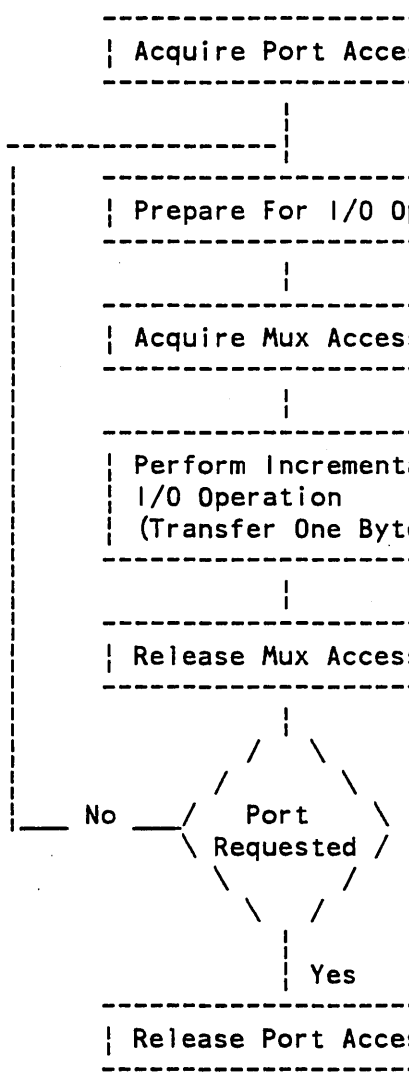
The Y-coordinates are in the range 7000B to 7777B. This corresponds to 51 lines using the standard 12B unit increment. Y-coordinates not corresponding to units of 12B are forced to the next (numerically) lower display line. SCD converts these Y-coordinates to their CC634B console equivalent coordinate which is as follows;

Left Screen Y-Coordinates		Right Screen Y-Coordinates	
CC634B	CC545	CC634B	CC545
0	7764	0	7764
1	7752	1	7752
.	.	.	.
.	.	.	.
.	.	.	.
31	7012	31	7012
32 *	7000	32 *	7000

* Command Line (Priority output only)

5.7 Two Port Mux Access Protocol

The diagram below illustrates how the following protocol conventions would be used when a PP program communicates through a port in the two port mux:



Common deck, COMPTMA, has been created which defines routines used to communicate with the two port mux. This deck should be referenced to obtain definitive information regarding use of the protocol defined below.

The following IOU flag and status bits are used to coordinate access to ports 0 and 1 on the two port mux:

- Channel 17 Channel Flag

- Channel 15 Channel Flag

- Four maintenance register status bits in the IOU Test-Mode register (A0):

  Bit 60 - Port 1 Requested
  Bit 61 - Port 0 Requested
  Bit 62 - Port 1 Reserved
  Bit 63 - Port 0 Reserved

The convention established to coordinate access to the two ports uses these flags and status bits in the following manner.

5.7.1 Acquire Port Access

- Test and set channel 17 flag (this instruction is used to prioritize multiple access requests).

- Set the "Port Reserved" status bit for the desired port.

  If this bit was already set, the port is currently in use and the requesting PP must wait for this bit to clear. To signal the using PP, the requesting PP sets the "Port Requested" status bit for the same port. The requesting PP then clears the channel 17 flag and loops in a sequence waiting for "Port Reserved" status to clear.

- Clear channel 17 flag.

The code sequence for acquiring port access consists of the following steps:

```
/acquire_port_access/
loop
  interlock 17
  read test_mode_register
  if not port_reserved then
    exit  /acquire_port_access/
  if not port_requested then
    set port_requested
    write test_mode_register
  clear_interlock 17
  delay
loopend
set port_reserved
write test_mode_register
clear_interlock 17
```

NOTE: Only those PP programs required to monitor "Port Requested" status (e.g. MDD) must relinquish control of their port when requested. If the requesting PP can be returned to the OS, it enters a three minute delay loop waiting for "Port Reserved" to clear before dropping from execution. Requesting PPs which cannot be returned to the OS enter an infinite loop waiting for port access.

5.7.2 Acquire Mux Access

- Test and set channel 17 flag (prioritize multiple requests).
- Test and set channel 15 flag.

  If already set, channel 15 is currently in use and the requesting PP must wait for this flag to clear. This requires the requesting PP to clear the channel 17 flag and attempt the sequence again.
- Clear channel 17 flag.

5.7.3 Release Mux Access

- Clear channel 15 flag.

5.7.4 Release Port Access

- Test and set channel 17 flag (prioritize multiple requests).
- Clear the "Port Reserved" and the "Port Requested" status bits for the port to be released.
- Clear channel 17 flag.

All programs using the two port mux must adopt the convention of acquiring and releasing their access in a manner which maximizes its availability for all users.

CHAPTER 6

PRINTER SUPPORT UTILITY (PSU)

6.1 Printer Support Utility

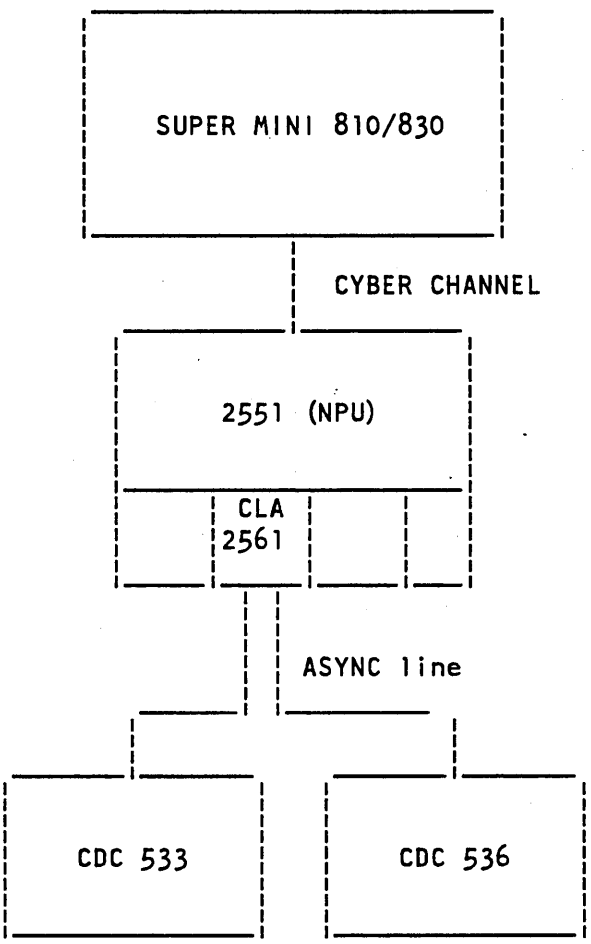
Printer Support Utility (PSU) is a NAM application that drives low cost line printers on CYBER 180-810/830 systems. PSU supports up to two CDC 533 or 536 printers connected via RS232 ASYNC ports to the 2551 Network Processing Unit.

The printer is equipped with either a 96-character band or a 64-character band. The 96-character band printer prints the ASCII characters in upper and lower case. The 64-Character band printer prints in upper case only, and is slightly faster than the 96-character band printer.

If there is only one printer or there are two printers with the same character band, then the printers are ready to be used as soon as the system is installed. If there is one 96-character printer and one 64 charater printer, then the operator must enter the forms code command to define the 96-character printer. Users should specify FC=AS to select 96-character printer. Routing to a 64-character printer is the default for FC.

NOTE PSU is not included in the release material when NAM is ordered. PSU must be licensed and ordered separately.

6.2 Network Configuration for CDC 533/536 Printers



6.3 Installation Requirement

Since PSU is a Network application, there are several installation requirements to make PSU work. The following is a list of file and common deck changes that are required. On the release system, the SYSGEN utility will build a default system with all the required files and common deck changes. The build procedure is documented in the Installation Handbook. If there are any changes to the default system configuration, the changes will be in items of the following list.

1. NDL statements in NCF -

```
line1: LINE,PORT=port1,LTYPE=A2,TIPTYPE=ASYNC,LSPEED=9600.
device1: TERMDEV,TC=M33,AUTOCON,HN=hostnode,LK=YES,OC=YES,
         PA=0,PW=0.

line2: LINE,PORT=port2,LTYPE=A2,TIPTYPE=ASYNC,LSPEED=9600.
device2: TERMDEV,TC=M33,AUTOCON,HN=hostnode,LK=YES,OC=YES,
         PA=0,PW=0.
```

2. NDL statements in LCF -

```
device1: USER,MFAM=fam,MUSER=PRINT01,MAPPL=PSU.
device2: USER,MFAM=fam,MUSER=PRINT02,MAPPL=PSU.
```

3. Application definition in common deck COMTNAP -

   Application PSU is specified in COMTNAP using the same validation bit as RBF.

4. Job record and job template in NAMSTRT file -

   Procedure file and job template are needed in NAMSTRT.

5. SYSTEM Validation file -

   User names PRINT01 and PRINT02 are validated to use PSU.

6.4 Printer Support Utility Commands

To use the Printer Support Utility, the operator must communicate with PSU by using the K display.

To initiate the communication with PSU, enter K,jsn at the console, where jsn is the JSN for PSU. The JSN for PSU can be obtained from NAM status K-display. Enter K.command for the PSU commands. The details of the PSU commands are documented in the CYBER Supermini Operations Handbook. Following is a list of the PSU commands and a brief description.

1. K.FORM,pn,AS. - Define a printer
2. K.MAXIMUM,pn,nn. - Ignore lengthy listings
3. K.SELECT,pn,jsn. - Move a job up in sequence
4. K.RERUN,pn. - Reprint a file
5. K.STOP,pn. - Stop the printing
6. K.BKSPRU,pn,nn. - Backspace a file
7. K.CONTINU,pn. - Resume printing after an interruption.
8. K.REPEAT,pn,nn. - Make multiple copies
9. K.SKIPRU,pn,nn. - Skip sections of a file
10. K.DISABLE,pn,BANNER. - Eliminate banner pages
11. K.ENABLE,pn,BANNER. - Permit banner pages
12. K.END,pn. - Drop a file

CHAPTER 7

SUPERMINI PROCEDURES

7.1 Introduction

CDC has found that after NOS has been installed at a site, there is still work that must be done before people at the site can begin to use the system. Most notably, users must be validated and a strategy for permanent file, queued file, and dayfile maintenance must be set up.

The current NOS utilities are powerful and flexible, but complicated, and require some study before they can be implemented. To simplify this process, CDC has developed a set of procedures that use the more complex utilities with pre-determined default values. These procedures require a minimum of training to use, all use the same format, and require few parameters. They are automatically installed during NOS installation.

When a site elects to use the procedures, it accepts the assumptions made during the writing of the procedures. These assumptions are as follows:

1. The CHARGE command is not used by the site.

2. The site uses the system default service classes.

3. A few sets of validation directives (called user types) can define the majority of users.

4. Only the default family is used for the user validation.

5. The dayfile resides on the default family.

Later in this chapter we will discuss how to create or modify a validation user type and the procedures that are documented here.

7.2 Location of Procedure Code

The source code for these procedures resides on the NOS PL delivered with NOS 2.3. During installation of NOS, this source code is compiled. The system procedure named MS and two procedure files (ZZZPIFL and ZZPPIFL) are created. MS is placed on the deadstart file while the procedure files reside on user name SYSTEMX.

You can obtain a listing of the procedures and the default MODVAL input files from the NOS PL by using the following procedure:

```
ATTACH,nospl_name/UN=LIBRARY.
MODIFY,P=nospl_name,L=LIST,C=0,LO=DEMCATS,Z./*EDIT SUPERM
```

You can use this listing to determine if you want to use the default validations, modify the default validations, create new validation types, or modify any of the procedures to use parameters not used in the supplied versions.

7.3 Initiation of Procedures

The system procedure MS is used to initiate all the procedures documented here. The procedures designed to be initiated from the system display console are initiated using the following command:

```
X.MS(procname,p1,p2,...,pn)
```

where procname is the name of the procedure and p1, p2, etc., are the parameters to the procedure.

The procedures designed to be initiated from a terminal use the following command:

```
SCREEN,model
MS(procname)
```

where model is the terminal model type and procname is the name of the procedure. The procedure then prompts you for any parameters it requires.

7.4 Procedure Descriptions

The procedures provided by CDC are listed below. The procedures are grouped by functional area. For complete information on the use of the procedures and any parameters, refer to the CYBER Supermini Operations Handbook (publication number 60459850).

Unless otherwise mentioned, you must run these procedures from the system display console. In those cases where you may run or are required to run the procedure from a terminal, you must use a terminal supported by the NOS full screen utilities.

PROCEDURE	DESCRIPTION
MS(BLANKST)	Blank labels a set of tapes. You specify the number of tapes in the set. These tapes are used for dumps of the permanent files, queued files, and dayfiles.
MS(DUMPLOG)	Dumps the permanent files ACCOUNT, DAYFILE, ERRLOG, and MAINLOG (created by the MS(ENDLOG) procedure) to tape. Each tape is written with a density of GE. A parameter optionally allows the purging of the files after being dumped.
MS(ENDLOG)	Appends a copy of the current active system logs to permanent files with the names ACCOUNT, DAYFILE, ERRLOG, and MAINLOG, depending on which dayfile is being ended. A new system

dayfile is created to receive subsequent dayfile messages.

MS(FINDJSN) Prints all entries for the specified JSN found in the current active system dayfile. You can run this procedure from either the system console or from a terminal. When you run it from a terminal, the procedure scans the permanent file DAYFILE for entries rather than the active system dayfile.

MS(LOADLOG) Loads the permanent files ACCOUNT, DAYFILE, ERRLOG, and MAINLOG back onto disk from the tape created by the procedure MS(DUMPLOG). You must run this procedure from a terminal.

MS(PURGLOG) Purges the permanent files with the names ACCOUNT, DAYFILE, ERRLOG, and MAINLOG on user name SYSTEMX.

MS(DUMPALL) Dumps all permanent files to a set of tapes. The set of tapes must be previously blank labeled.

MS(DUMPMOD) Dumps permanent files modified since the last full dump of permanent files.

MS(DUMPOLD) Dumps permanent files that have not been read, executed, or written after the date specified as a parameter.

MS(LOADF) Loads permanent files from tapes created by the procedures MS(DUMPALL), MS(DUMPMOD), and MS(DUMPOLD). These tapes were blank labeled using the procedure MS(BLANKST) and were recorded with a density of GE.

MS(DUMPQUE) Dumps all queued files to a tape. After the queued files are dumped, they are purged from the system.

MS(LOADQUE) Loads the queued files onto the system from the tape created by the procedure MS(DUMPQUE).

MS(VALUSER) Creates, deletes, or modifies the validation for a single user (when parameters are specified) or accesses the file created by the procedure MS(WRITEIV) and submits the directive file to MODVAL.

MS(WRITEIV) Writes a validation input file used by the procedure MS(VALUSER). You must run this procedure from a terminal. WRITEIV is used to validate many users with one of the user types previously defined (either by CDC or the site).

7.5 Differences Between NOLIMIT and LIMITED User Types

A user, to which you have assigned the default NOLIMIT user type, can request and use all resources available to the system. The system administrator (user type SYSADM), the Control Data Central Hardware Support (user type CDCCHS), and the Control Data Central Software Support (user type CDCCSS) users are NOLIMIT users with additional special privileges.

A user, to which you have assigned the default LIMITED user type, is limited in the following ways:

- An individual direct access file is limited to 20,480 PRUs.
- An individual indirect access file is limited to 192 PRUs.
- The total mass storage allocatable per job is limited to 66,048 PRUs.
- The total size of all indirect access files is limited to 4096 PRUs.
- The maximum central memory available is limited to 203,700 octal words (67,520 decimal words).
- The maximum number of detached jobs is limited to 3.
- The maximum number of deferred batch jobs is limited to 8.

If you feel these limits need modification, refer to the following section for information on modifying these values.

7.6 Creating or Modifying User Validation Types

There are five user types supplied by CDC for use with these procedures. They are; CDCCHS for CDC hardware support personnel, CDCCSS for CDC software support personnel, SYSADM for the system administrator, NOLIMIT for a user with high resource requirements, and LIMITED for a user with low resource requirements. These five types are names of text records on ZZZPFIL. Each of these text records is a collection of MODVAL input directives. If you wish to create a new user validation type using an existing type as a base or modify the limits of any of the supplied types, use the following process:

1. Log into the system using a terminal that is supported by the NOS full screen utilities under the user name INSTALL.

2. Enter the following command to establish screen mode:

   SCREEN,model.

   where model is the terminal model supported by the NOS full screen utilities.

3. Attach the file containing the user validation type record with the following command:

   ATTACH,ZZZPIFL/M=W,UN=SYSTEMX.

4. Extract the text record of the user validation type you are using as a base (NOLIMIT in this example) with the following command:

   GTR,ZZPIFL,lfn.TEXT/NOLIMIT

   where lfn is a local file name of your choosing.

5. Begin editing the local file with the following command:

   FSE,lfn.

   During the editing session, if you make no change to the seven-character first line of the file, you will only modify the default values of the user validation type. If you change the first line of the file, you

will have defined a new user type with the type name specified in the first line of the file. If you have created a new user type, you must have an end-of-record mark as the last line of the file. In FSE you indicate an end-of-record with the following string placed left-justified after the last line of the file:

(EOR)

6. Exit the editing session when changes are complete by pressing the appropriate function key, or by going to the home line and entering the following command:

   Q

7. Edit your new or modified text record back into the file ZZZPIFL and generate a new file (NPL in this example) by entering the following command:

   LIBEDIT,ZZZPIFL,N=NPL,B=lfn,I=0.

   where lfn is the local file you edited in step 5.

8. Copy and verify the new ZZZPIFL onto the old procedure file with the following command:

   COPYEI,NPL,ZZZPIFL,V.

9. Return the new file ZZZPIFL with the following command:

   RETURN,ZZZPIFL.

You can now run the procedures with the new or modified validation available. You should save a copy of ZZZPIFL after it was modified because if you re-install NOS, the changes will revert to the default values supplied with the release. You would then have to repeat the above process to reestablish the modified values.

7.7 Modifying a Procedure File

If you wish to modify a procedure file rather than a user validation type, you would use the exact same process as outlined above, except that you would specify the following command in place of the GTR command in step 4:

```
GTR,ZZZPIFL,lfn.PROC/procname
```

CHAPTER 8

SCREEN MANAGEMENT ENHANCEMENTS

First of all, the term Screen Management needs some clarification. It is the generic name given to the collection of Control Data's full-screen products which were introduced at NOS 2.2. Screen Management encompasses the Full Screen Editor (FSE), Screen Management Facility Subsystem (SMF) and the Screen Format Products. FSE can be run in single user mode whereby each user has their own copy, or via SMF where many users utilize a single copy. The Screen Format Products provide full-screen input and output capabilities for NOS procedures and for COBOL5, FORTRAN5 and PASCAL 1.1 programs. NOS 2.3 adds support of PASCAL 1.1 for all features previously supported for COBOL5 and FORTRAN5. See the NOS 2 Screen Formatting Reference Manual, publication number 60460430, for further information. Even though the different entities of Screen Management may share some common routines, they are different utilities.

At NOS 2.2 L596, the Full Screen Editor included system defined support for five different terminals. They were:

CDC VIKING 721	DEC VT100
CDC VIKING 722	ZENITH Z19/H19

Also at L596, the only terminal which could utilize the Screen Format Products was the CDC VIKING 721.

At L602, three additional terminals were added to the list of supported terminals for FSE. They included:

LEAR SIEGLER ADM3A
LEAR SIEGLER ADM5
TEKTRONIX 4115

Support for all other terminals for use with FSE involved the site in making modifications to routine VIRTERM (See NOS 2.2 Feature Note - NOS FSE Support of Additional Terminals).

This leads us to the changes made at NOS 2.3 L617.

First, FSE now allows direct cursor positioning via the touch panel on the CDC 721, if a touch panel is installed.

Second, FSE will support type-ahead. This allows the user to send commands to FSE before it responds to previous commands, in case the system is slow in responding. Caution should be used since the terminal screen may not reflect the data on the file as it really exists and therefore this feature should only be used by an experienced analyst. This feature is activated by a parameter within the TDUIN file.

It is now possible to interface almost any terminal which uses the asynchronous communication protocol in character mode to work with all products within Screen Management. This is accomplished by describing your terminals' attributes to one of the screen formating utilities called TDU (Terminal Definition Utility). The following pages will further describe TDU and illustrate its use in adding support for the TELEVIDEO 920C terminal.

TDU is an interactive procedure which accepts the users' terminal definition file (an ASCII file called TDUIN) and produces a loadable capsule which is placed in a file called TERMLIB. TERMLIB can exist as a local file, a permanent file in the users' catalog, or, if you want other users to access it, a permanent file in the LIBRARY catalog. A default TERMLIB is provided as part of the NOS installation and contains capsules for the same terminals supported at NOS 2.2 L602. The site has the option to add all its terminal definitions to the file in the LIBRARY catalog or each user can build their own version and keep it in their own permanent file catalog. This enables a user to override what the system default is.

The NOS SCREEN command has been enhanced to search for a file named TERMLIB, first as a local file, then as a permanent file in the user catalog, and then as the installation file under user name LIBRARY. Once a terminal name match is found, the terminal definition capsule is copied from the appropiate TERMLIB to a local file called ZZZZTRM. This file will be accessed by all products under Screen Management. Because of this enhancement, the "SET SCREEN TERMINAL MODEL" command within FSE has been changed. The "SET SCREEN" can be used only to repaint the screen, not change terminal model. If this is needed, exit FSE, execute the NOS SCREEN command supplying the new terminal model, then re-enter FSE.

What follows are the steps in using TDU to define the TELEVIDEO 920C for NOS Screen Management.

Our first step is to place our terminal in ASCII mode since the TDUIN file must be an ASCII file. We perform this with the NOS command:

```
ASCII.
```

We retrieve from file TDUFILE under user name LIBRARY, the record called TDUIN. This is a skeleton of the file needed as input by TDU. It contains all the necessary definitions needed by Screen Management. Our job will be to fill in the blanks for each definition. Execute the following to obtain the TDUIN file.

```
ATTACH,TDUFILE/UN=LIBRARY.
GTR,TDUFILE,TDUIN.TDUIN
```

Now we use an editor to modify the TDUIN file. You will need the hardware reference manual describing your terminal's attributes in order to answer some of the questions. What follows is the TDUIN file for the TELEVIDEO 920C. Descriptions of each parameter, along with comments on what we have already talked about, are included within the file. Capital letters have been used in the file for documentation purposes only.

********** BEGIN THE TDUIN FILE FOR 920C **********

PLACE YOUR TERMINAL NAME IN THE LINE BELOW FOR DOCUMENTATION

TERMINAL DEFINITION FILE FOR T920C TERMINAL !

The terminal definition utility (TDU) allows user definition of most character mode asynchronous type terminals for use with all NOS full screen products. A detailed description of TDU can be found in the NOS Screen Formatting Reference Manual.

There should be a collection of system defined terminal definition files (as seperate records) on file TDUFILE, UN=LIBRARY that may assist you (and perhaps already define your terminal or one that is very similar to it). The record names and the terminals that they define should include:

TDUIN	TEMPLATE FILE
TDU721	CDC 721 (Viking X with Version 4 firmware)
TDU722	CDC 722
TDUVT10	DEC VT100
TDUT415	TEKTRONIX T4115
TDUZ19	ZENITH Z19/Z29
TDUADM3	LEAR SIEGLER ADM3A
TDUADM5	LEAR SIEGLER ADM5
TDUVKX3	CDC 721 (Viking X) with Version 3.0 firmware.
TDU721T	CDC 721 (Viking X) with type ahead, touch panel, and automatic tabbing (for screen formatting)
TDU722T	CDC 722 with type ahead
TDUVT1T	DEC VT100 with type ahead
TDUT41T	TEKTRONIX T4115 with type ahead
TDUZ19T	ZENITH Z19/Z29 with type ahead
TDUAD3T	LEAR SIEGLER ADM3A with type ahead
TDUAD5T	LEAR SIEGLER ADM5 with type ahead
TDUVK3T	CDC 721 (Viking X) with Version 3.0 firmware and type ahead.

A collection of terminal definition files for other terminals will also be made available through Central Software Support for a variety of popular terminals and micro computers.

This file (TDUIN) is the input file that you will fill with the specific terminal dependent data that you should find in the hardware reference manual for your terminal. When the sequences, capabilities and attributes of your terminal have been filled in you will then compile your terminal definition by using the system command TDU. This will produce a file named TERMLIB which contains an encapsulated copy of the information needed by NOS screen formatting products to utilize your terminal. The command SCREEN,xxxxxx (xxxxxx being the value you specified for the model_name statement) will then enable you to interact with all NOS full screen facilities.

Note that if your have removed this file from TDUFILE where it was one of a number of records that are each a terminal definition file, you will need to delete the first line (TDUIN) which is the record name and not part of the actual file.

A number of capabilities are required for your terminal to function in screen mode. These are a clear_page_stay or a clear_page_home, a cursor_home, and the ability to directly position the cursor on the screen. At least a subset (F1-F6) of the application keys and a CDC standard STOP function key (a good choice is CTL/T) should also be defined. An erase_end_of_line capability is not required but will provide considerably better performance for all full screen products.

Any line surrounded by quotation marks is a comment line and will be ignored when compiling your terminal capsule. This is a way in which you can add your own comments to this file as you proceed to fill in the requested information. Those lines that are not surrounded by quotation marks in this file are the input directives to TDU for which you will fill in the correct values for your terminal.

TDU allows you to define variables for commonly used character strings and recognizes ASCII mnemonics (such as rs, ack). Both your variables and the mnemonics can be used anywhere in this file.

Here are some examples to assist you in your definitions:

```
VARIABLES

set_line_mode    = ()          Empty sequence.
set_line_mode    = (rs ack)    ASCII mnemonics.
set_line_mode    = (14(8))     (8) indicates an octal
                               value.
```

set_line_mode	= (14(16))	(16) indicates a hexadecimal value.
set_line_mode	= (14)	Any nonsubscripted number is decimal.
blank_character	= (' ')	Blank character (see line drawing).
start_underline	= (rs '=')	ASCII mnemonic and character.
stop_underline	= (rs '''')	Use of apostrophe.

Another use of the TDU capability to define variables can be used to make default function key sequences for FSE (which are also defined in TDUIN) more readable.

Here is an example for a terminal with a set of six (F1-F6) keys:

```
VARIABLES FOR FULL SCREEN EDITOR FUNCTION KEY DEFINITIONS

   k1     = ('SK1/SM/L/ MARK/;SKS1/SMW/L/MRKCHR/')
   k2     = ('SK2/MMTP/L/ MOVE/;SKS2/CMTP/L/ COPY/')
   k3     = ('SK3/IBP/L/ INSB/;SKS3/DB/L/ DELB/')
   k4     = ('SK4/PF/L/ FIRST/;SKS4/VL/L/ LAST/')
   k5     = ('SK5/U/L/ UNDO/')
   k6     = ('SK6/Q/L/ QUIT/')

FOR THE T920C, WE WILL DEFINE 11 VARIABLES WHICH DEFINE THE
FUNCTION KEYS USED IN THE FSE FUNCTION KEY DEFINITION
SECTION.

    K1     = ('SK1/VNS/L/ FWD/')
    K1S    = ('SKS1/VPS/L/ BKW/')
    K2     = ('SK2/VN/L/LINEUP/')
    K2S    = ('SKS2/VP/L/LINEDN/')
    K3     = ('SK3/.I/L/ INSC/')
    K3S    = ('SKS3/.D/L/ DELC/')
    K4     = ('SK4/IBP1/L/ INSL/')
    K4S    = ('SKS4/D;PN/L/ DELL/')
    K5     = ('SK5/SM/L/ MARK/')
    K5S    = ('SKS5/U/L/ UNDO/')
    K6     = ('SK6/MMTP/L/ MOVE/')
    K6S    = ('SKS6/CMTP/L/ COPY/')
    K7     = ('SK7/H/L/ HELP/')
    K7S    = ('SK7S/SV00/L/ LEFT/')
    K8     = ('SK8/Q/L/ QUIT/')
    K8S    = ('SK8S/SV0&&C;.P+20/L/ RIGHT/')
    K9     = ('SK9/.E/L/ENDLIN/')
    K9S    = ('SK9S/.C/L/CENTER/')
```

```
K10    = ('SK10/.J/L/ JOIN/')
K10S   = ('SK10S/.S/L/SPLIT/')
K11    = ('SK11/V/L/MIDDLE/')
K11S   = ('SK11S/.F/L/ PARA/')
```

There are several basic types of statements that you will encounter in this file:

- VALUE STATEMENTS

```
model_name                value = 'myown'
has_protect               value = TRUE
```

Where VALUE is TRUE, FALSE, an alphabetic string or a number.

- TYPE STATEMENTS

```
cursor_pos_encoding       type  = ansi_cursor
char_past_last_position   type  = wrap_adjacent_next
```

Where TYPE is one of a predefined list of choices that will be listed preceding the statement.

- IN STATEMENTS

```
f1                        in    = (rs 71(16))
help                      in    = (rs 5C(16))
```

Where IN is the sequence that comes upline from the terminal when a specific function is performed or key is pressed.

- OUT STATEMENTS

```
cursor_pos_begin          out   = (stx)
bell_nak                  out   = (bel)
```

Where OUT is the sequence sent down line to the terminal to perform a certain function.

- INOUT STATEMENTS

```
erase_page_home           inout = (ff)
tab_forward               inout = (ht)
```

Where INOUT is the identical sequence sent up and down line for a certain function.

It should be noted that you may break any INOUT statement like

```
tab_forward             inout = (ht)
```

into a matched pair of statements like

```
tab_forward             in    = (ht)
tab_forward             out   = (ht)
```

You will need to do this if your terminal needs a different sequence downline to the terminal to perform a certain function than is sent upline when that function is performed. If in our example your terminal recognized ht (from the host) as a signal to perform a tab forward but sent vt (to the host) to indicate that the tab forward key had been pressed, then the single tab_forward inout = () statement would be split into:

```
tab_forward             in    = (vt)   [in from the
                                        terminal]
tab_forward             out   = (ht)   [out to the
                                        terminal]
```

Any statement that is IN or OUT only should be left as is.

The file from this point on is arranged by functional groups and contains comments for each directive that should assist you in filling in the correct sequences for your terminal.

MODEL NAME AND COMMUNICATION TYPE

model_name - A one to six character alphanumeric name for your terminal. Lower case letters are translated to upper case. The value specified here will be the name used on the SCREEN command.

ENTER YOUR MODEL NAME BELOW, AS IN T920C.

```
model_name          value = 'T920C'
```

Communication type is asynch as only asynchronous terminals are presently supported.

```
communications      type  = asynch
```

END OF INFORMATION SPECIFICATION

This defines the end of information sequence which is a zero byte.

THIS SHOULD NOT NEED CHANGING.

```
    end_of_information  in    = (0)
```

CURSOR POSITIONING INFORMATION

The way in which your terminal encodes cursor positioning will determine your choice for cursor_pos_encoding and cursor_pos_column_first. The general format for cursor positioning is:

```
   Let X    ---------------> represent the column coordinate.
   Let Y    ---------------> represent the row coordinate.
   Let a    ---------------> represent cursor_pos_begin.
   Let b    ---------------> represent cursor_pos_second.
   Let c    ---------------> represent cursor_pos_third.
   And Bias ---------------> is the integer value added to
                             the row or column for cursor
                             positioning.  You should be able
                             to find the value for bias in
                             the hardware reference manual
                             for your terminal (often
                             20(16)).
```

Then cursor_pos_encoding will be one of three type:

```
 ansi_cursor    ----> Those terminals which are ANSI standard
                      and use decimalized cursor coordinates.
                      Format is:
                           a (X + bias) b (Y + bias) c
                           a (Y + bias) b (X + bias) c
                      the order of X and Y for your terminal
                      determines the value for
                      cursor_pos_column_first.

 cdc721_cursor ----> The Control Data 721 (Viking X)
                      terminal.  Format is:
                           a    (X + bias)      (Y + bias)
                                (if X is less than 81)
                           a b  (X + bias -80)  (Y + bias)
                                (if X greater than 80)

 binary_cursor ----> Those terminals which use direct
                      coordinate positioning.
                      Format is:
                           a (X + bias) b (Y + bias) c
```

a (Y + bias) b (X + bias) c
the order of X and Y for your terminal determines the value for cursor_pos_column_first.

SINCE THE T920C USES DIRECT CURSOR ADDRESSING WE HAVE CHOSEN BINARY_CURSOR.

THE BIAS FOUND IN THE HARDWARE REFERENCE MANUAL WAS 32 DECIMAL OR 20(16).

cursor_pos_encoding bias = 32 type = binary_cursor

Cursor_pos_column_first has a value of TRUE if your terminal sends the X (or column) coordinate followed by the Y (or row) coordinate and has a value of FALSE if the reverse is true.

THE T920C ISSUES THE Y (OR ROW) FIRST.

THE FORMAT FOR CURSOR ADDRESSING IS AS FOLLOWS

ESC=YX(Y = 1-24, X = 1-80)

FOR EXAMPLE, ESC=%@ SAYS Y=6 AND X=33 WHICH PLACES THE CURSOR AT LINE 6 AND COLUMN 33.

cursor_pos_column_first value = FALSE

Cursor_pos_column_length and row_length apply only to ANSI type cursor position (these are zero for both other types) and are non-zero only if your terminal sends a fixed number of decimalized bytes for the column and row coordinates (as opposed to a variable number which is the usual case).

cursor_pos_column_length value = (0)
cursor_pos_row_length value = (0)

Cursor_pos_begin, second and third are the sequences sent before the first coordinate, in between coordinates and after the last coordinate when positioning the cursor (a b and c in the formats shown above). At least a cursor_pos_begin should be supplied for your terminal though second and third are often an empty sequence and can be left alone.

AS SEEN IN THE FORMULA ABOVE, THE ESC= PRECEEDS THE Y AND X COORDINATES. THERE ARE NO SEQUENCES BETWEEN THE Y AND X COORDINATES. I HAVE REPRESENTED THIS WITH 1B(16) 3D(16).

```
cursor_pos_begin        out   = (1B(16) 3D(16))
cursor_pos_second       out   = ()
cursor_pos_third        out   = ()
```

CURSOR MOVEMENT INFORMATION

Cursor_home, up, down, left and right are the sequences sent both downline to the terminal and via cursor keys, sent upline from the terminal, to move the cursor to the home position or a single column or row up, down, left, or right. Since this is both an upline and downline sequence the INOUT keyword is used.

THE FOLLOWING SEQUENCES WERE FOUND IN THE T920C REFERENCE MANUAL AND REPRESENT THE CURSOR KEYS

```
cursor_home             inout = 1E(16)
cursor_up               inout = 0B(16)
cursor_down             inout = 0A(16)
cursor_left             inout = 08(16)
cursor_right            inout = 0C(16)
```

CURSOR BEHAVIOR (for cursor movement keys)

Move_past_right, left, top and bottom describe what happens when the cursor on your terminal is urged to move past the right, left, top and bottom of the screen by a cursor movement key (not by cursor movement caused by character input or a seperate backspace key your terminal may have in addition to a cursor left key, these behaviors may be different from those for cursor positioning keys and will be defined in the next section). The possible types are:

```
wrap_adjacent_next ----> The cursor wraps to the other end
                         of the screen on the adjacent row
                         (next row cursor_right or previous
                         row for cursor_left).
wrap_same_next     ----> The cursor wraps to the other
                         end of the screen still in the
                         same row or column.
scroll_next        ----> The terminal scrolls.
stop_next          ----> The cursor stops
home_next          ----> The cursor homes.
```

CURSOR BEHAVIOR CAN BEST BE FOUND BY EXPERIMENTING WITH YOUR TERMINAL. TRY MOVING THE CURSOR (WITH THE CURSOR KEYS) FOR THE RESULTS.

```
    move_past_right          type  = WRAP_ADJACENT_NEXT
    move_past_left           type  = WRAP_ADJACENT_NEXT
    move_past_top            type  = WRAP_SAME_NEXT
    move_past_bottom         type  = WRAP_SAME_NEXT
```

CURSOR BEHAVIOR (for character keys)

Char_past_right, left and last_postion describe what happens when the cursor on your terminal is urged to move past the right, left and end of the screen by character input or a separate backspace key your terminal has in addition to (or in place of) a cursor left key. The possible behaviors are the same as those for cursor positioning keys.

```
 wrap_adjacent_next ----> The cursor wraps to the other end
                          of the screen on the adjacent row
                          (next row cursor_right or previous
                          row for cursor_left).
 wrap_same_next     ----> The cursor wraps to the other
                          end of the screen still in the
                          same row or column.
 scroll_next        ----> The terminal scrolls.
 stop_next          ----> The cursor stops
 home_next          ----> The cursor homes.
```

AGAIN, THE WAY TO FIND OUT WHAT TO USE IS BY EXPERIMENTATION

```
    char_past_right          type  = WRAP_ADJACENT_NEXT
    char_past_left           type  = WRAP_ADJACENT_NEXT
    char_past_last_position  type  = WRAP_ADJACENT_NEXT
```

TERMINAL ATTRIBUTES

These describe various attributes and capabilites of your terminal that should be either TRUE or FALSE.

BE AWARE THAT YOUR TERMINAL MAY HAVE ATTRIBUTES LIKE INVERSE VIDEO, BLINKING, PROTECT MODE, UNDERLINE, ETC. BUT IF THE TERMINAL HARDWARE UTILIZES A CHARACTER POSITION TO STORE THE ATTRIBUTE (WHICH MEANS THAT A CHARACTER CANNOT EXIST IN THAT POSITION), THEN YOU MAY NOT USE YOUR TERMINAL ATTRIBUTES, SINCE SCREEN MANAGEMENT WILL NOT HANDLE THIS. THIS IS THE CASE WITH THE T920C.

Automatic_tabbing is TRUE if your terminal supports tabbing from one completed filled unprotected input field to the next without requiring that a tab key is pressed.

automatic_tabbing value = FALSE

Clears_when_change_size is TRUE if your terminal has more than one screen size and changing screen sizes causes the screen to be cleared.

THE T920C ONLY HAS ONE SCREEN SIZE.

clears_when_change_size value = FALSE

Function_key_leaves_mark is TRUE if pressing a function key on your terminal leaves a visible mark or character on the screen or if function keys for your terminal will be supported by an escape or control sequence that will require a character to complete. The full screen editor will then know to rewrite the line on the screen that has been overwritten by the mark or character(s).

THE T920C HAS 11 FUNCTION KEYS AND DOES NOT LEAVE A VISIBLE MARK OR CHARACTER ON THE SCREEN WHEN THEY ARE DEPRESSED.

function_key_leaves_mark value = FALSE

Has_hidden is TRUE if your terminal supports a hidden attribute that allows a field to be defined as input only such that typed characters are not displayed on the screen.

has_hidden value = FALSE

Has_protect is TRUE if the terminal hardware supports a protected field attribute so that users can only enter data within specified areas on the screen.

has_protect value = FALSE

Home_at_top is TRUE if the cursor goes to the top of the screen when the home key is pressed or FALSE if it goes to the bottom.

THE T920C HAS THE CURSOR HOME POSITION AT THE TOP OF THE SCREEN.

home_at_top value = TRUE

Multiple_sizes is true if your terminal has more than one screen size that can be set by a sequence sent downline to the terminal.

AN EXAMPLE OF THIS WOULD BE THE CDC721. THE T920C ONLY HAS ONE SCREEN SIZE.

multiple_sizes value = FALSE

Tabs_to_home is TRUE if when tabbing forward from the last unprotected field on the screen (or backward from the first) the cursor moves to the home position and will move to the field when the tab key is pressed again. Set FALSE if your terminal can tab directly from the last unprotected field to the first (and vice versa) or if your terminal does not support a protect attribute.

tabs_to_home value = TRUE

Tabs_to_tab_stops is TRUE if your terminal supports hardware tabbing to tab stops, FALSE otherwise.

THE T920C SUPPORTS TABBING FORWARD AND BACKWARD.

tabs_to_tab_stops value = TRUE

Tabs_to_unprotected is TRUE if your terminal supports tabbing from one unprotected field to the next (or previous). Set to FALSE if the terminal does not support protect or protected tabbing.

tabs_to_unprotected value = FALSE

Type_ahead is TRUE if you wish to run the full screen editor in type ahead mode, FALSE if you do not. This has no effect on screen formatting applications. Type ahead means that you do not have to wait for the system response to each carriage return (next key) but may continue to type. Care should be exercised not to abuse this capability since it is possible to produce a screen that does not reflect the actual file contents. If you fear this is the case do a clear page or a SET SCREEN (SS) command to tell FSE to repaint the screen. In addition typed ahead control t-s (STOP keys) can not presently be handled by FSE so you should avoid using procedures unless you are sure they will end and not loop continuously.

I CHOSE NOT TO ENABLE THIS FEATURE SINCE I DID NOT WANT INEXPERIENCED USERS TO GET CONFUSED BY TYPING AHEAD AND SEEING THE TERMINAL SCREEN IN A CONDITION WHICH IS CONTRARY TO WHAT THE ACTUAL FILE LOOKS LIKE.

type_ahead value = FALSE

SCREEN SIZES

These sequences are those necessary to set the terminal to a specific number of lines and columns if the terminal has more than one screen size that can be downline configured. If the terminal does have more than one size, specify them in ascending order (giving columns preference over lines) by duplicating the entire set_size rows = yy columns = xx out = (sequence) statement. A maximum of four sizes and a minimum of one are to be specified.

Rows is the integer number of rows (lines) on the screen for a specific screen size.

Columns is the integer number of columns (characters per line) for a specified screen size.

Out is the sequence to be sent to the terminal to set a screen size (it may be an empty sequence for a terminal with only one size but the rows and columns should still be entered).

THE T920C SUPPORTS ONLY ONE TERMINAL SIZE, 24 X 80.

set_size rows = 24 columns = 80 out = ()

SCREEN AND LINE MODE TRANSITION

Screen_init is the sequence that will be sent to the terminal when a SCREEN,TERMINAL_NAME command (or a SCREEN. command when a SCREEN,TERMINAL_NAME or LINE,TERMINAL_NAME identifying the terminal has previously been executed) is executed. This is useful for a terminal that requires a large amount of reconfiguration, some of which does not affect line mode dialogs and thus does not have to be done at each entrance to a full screen application (see set_screen_mode).

THERE IS NO TERMINAL PRE-CONDITIONING WHICH IS NECESSARY FOR THE T920C. THIS IS USEFUL FOR TERMINALS LIKE THE TEKTRONIX 4115 WHICH NEED A LARGE AMOUNT OF SETUP IN ORDER FOR THE TERMINAL TO FUNCTION WITH SCREEN MANAGEMENT.

screen_init out = ()

Line_init is the sequence that will be sent to the terminal when a LINE,TERMINAL_NAME command (or a LINE. command when a SCREEN,TERMINAL_NAME or LINE,TERMINAL_NAME identifying the terminal has previously been executed) is executed.

line_init out = ()

Set_screen_mode is the sequence that will be sent when the terminal enters the full screen editor or a screen formatting application. This is where page mode should be set, tabs perhaps cleared and so on to configure for running in screen mode.

WE SUPPLIED NO INFORMATION IN THE SCREEN_INIT DEFINITION SINCE THE T920C DOES NOT NEED ANY MAJOR PRE-CONDITIONING. HOWEVER, WE ARE CONCERNED ABOUT TABS WHICH MAY HAVE BEEN SET VIA SOME OTHER APPLICATION, ETC. THEREFORE WE WILL CLEAR ALL TABS WHICH ARE SET IN THE TERMINAL BY THE SEQUENCE

ESC 3

WE WILL DO THIS UPON ENTRY/EXIT OF ANY SCREEN APPLICATION NEW TABS WILL BE SETUP VIA YOUR FSEPROC FILE OR VIA FSE FUNCTION KEY DEFINITIONS WHICH ARE DESCRIBED LATER WITHIN THIS FILE.

set_screen_mode out = (1B(16) 33(16))

Set_line_mode is the sequence that will be sent when the terminal exits the full screen editor or a screen formatting application. This is where roll (or line) should be set and what was done by the set_screen_mode sequence reversed.

set_line_mode out = (1B(16) 33(16))

TERMINAL CAPABILITIES

These define what capabilities such as local insert and delete line or character your terminal provides.

Backspace allows you to define a key that sends a different (from the cursor left key) sequence upline from the terminal to move the cursor one character position to the left. This is of particular use if the behavior for the backspace key (which will be treated as a character movement key, not a

cursor movement key and hence is bound by the CHARACTER MOVEMENT BEHAVIOR descriptions) differs from the CURSOR MOVEMENT BEHAVIOR for the cursor_left key (as described in the CURSOR MOVEMENT BEHAVIOR section of this file). This is an input only sequence so the IN keyword is used here.

SINCE THE T920C USES THE SAME SEQUENCE FOR CURSOR LEFT KEY AND BACKSPACE KEY, NO DEFINITION IS NEEDED HERE.

```
backspace           in    = ()
```

Delete_char is the sequence for local delete character for your terminal. In order for this to function correctly, the key that does the local (that is on the screen) delete character must send a sequence upline to make the full screen product aware that the screen has changed. This is true for all terminal capabilities.

THE ESC W SEQUENCE IS USED BY THE T920C TO DELETE A CHARACTER.

```
delete_char         inout = (1B(16) 57(16))
```

Delete_line_bol and delete_line_stay are provided so that full screen applications are aware of the cursor position after a delete line function has been performed. If your terminal has a local delete line function then one (and only one) of delete_line_bol or delete_line_stay should be filled with the correct terminal sequence. Delete_line_bol if the cursor moves to the leftmost position when a line is deleted, delete_line_stay if the cursor stays in the column it was in when the delete line function was performed.

THE ESC R SEQUENCE IS USED BY THE T920C TO DELETE A LINE. THE CURSOR DOES NOT MOVE TO THE BEGINNING OF THE LINE AFTER THE DELETE TAKES PLACE.

```
delete_line_bol     inout = ()
delete_line_stay    inout = (1B(16) 52(16))
```

Erase_char is the sequence for an erase character function.

```
erase_char          inout = ()
```

Erase_end_of_line is the sequence for an erase from the current cursor position to the end of that line. This is not a required terminal capability but will provide much better performance for all full screen products.

THE ESC T SEQUENCE IS USED BY THE T920C TO ERASE FROM CURSOR TO END OF LINE.

```
erase_end_of_line    inout = (1B(16) 54(16))
```

Erase_field_bof is reserved for future use.

```
erase_field_bof      inout = ()
```

Erase_field_stay is reserved for future use.

```
erase_field_stay     inout = ()
```

Erase_line_bol and erase_line_stay are provided so that full screen applications are aware of the cursor position after an erase line function has been performed. If your terminal has a local erase line function then one (and only one) of erase_line_bol or erase_line_stay should be filled with the correct terminal sequence. Erase_line_bol if the cursor moves to the leftmost position when a line is erased, erase_line_stay if the cursor stays in the column it was in when the erase line function was performed.

THE T920C HAS THIS FEATURE IN THE FORM OF THE ABOVE DEFINED ERASE_END_OF_LINE. IF THE CURSOR IS POSITIONED AT THE FIRST COLUMN IT ESSENTIALLY PERFORMS AN ERASE LINE. SINCE WE ALREADY HAVE THIS DEFINED IN ONE PLACE IT IS UNNECESSARY AND ILLEGAL TO DEFINE IT AGAIN.

```
erase_line_bol       inout = ()
erase_line_stay      inout = ()
```

Erase_page_home and erase_page_stay are provided so that full screen applications are aware of the cursor position after an erase page function has been performed. If your terminal has a local erase page function (that sends a sequence upline) then one (and only one) of erase_page_home or erase_page_stay should be filled with the correct terminal sequence. Erase_page_home if the cursor moves to the home position when the screen is cleared, erase_page_stay if the cursor stays where it was when the erase page function was performed.

THE CTL/Z SEQUENCE IS USED BY THE T920C TO ERASE THE PAGE AND THE CURSOR ALWAYS GOES TO THE HOME POSITION.

```
erase_page_home      inout = 1A(16)
erase_page_stay      inout = ()
```

Insert_char is the sequence for local insert character for your terminal. In order for this to function correctly the key that does the local (that is on the screen) insert character must send a sequence upline to make the full screen product aware that the screen has changed. This is true for all terminal capabilities.

THE ESC Q SPACE SEQUENCE IS USED BY THE T920C TO INSERT A CHARACTER.

```
insert_char          inout = (1B(16) 51(16) 20(16))
```

Insert_line_bol and insert_line_stay are provided so that full screen applications are aware of the cursor position after an insert line function has been performed. If your terminal has a local insert line function (that sends a sequence upline) then one (and only one) of insert_line_bol or insert_line_stay should be filled with the correct terminal sequence. Insert_line_bol if the cursor moves to the leftmost position when a line is inserted, insert_line_stay if the cursor stays in the column it was in when the insert line function was performed.

THE ESC E SEQUENCE IS USED BY THE T920C TO INSERT A LINE AND THE CURSOR ALWAYS GOES BACK TO THE BEGINNING OF THE LINE AFTER THE INSERT.

```
insert_line_bol      inout = (1B(16) 45(16))
insert_line_stay     inout = ()
```

Erase_unprotected is reserved for future use.

```
erase_unprotected    inout = ()
```

Erase_end_of_page is reserved for future use.

```
erase_end_of_page    inout = ()
```

Erase_end_of_field is reserved for future use.

```
erase_end_of_field   inout = ()
```

Insert_mode_begin is the sequence to enter insert mode. Characters are inserted, shifting other characters right rather than overstriking them.

```
insert_mode_begin    inout = ()
```

Insert_mode_end is the sequence to exit insert mode. Characters will now overstrike rather than insert.

```
insert_mode_end      inout = ()
```

Insert_mode_toggle will switch between insert and overstike mode.

```
insert_mode_toggle   inout = ()
```

Tab_backward is the sequence sent (and received) when tabbing from a tab stop or unprotected field to the previous tab stop or unprotected field.

THE ESC I SEQUENCE IS USED BY THE T920C TERMINAL TO TAB BACKWARDS.

```
tab_backward         inout = (1B(16) 49(16))
```

Tab_clear is the sequence to clear the tab stop at the current cursor position.

THE ESC 2 SEQUENCE IS USED BY THE T920C TO CLEAR A TAB AT THE CURSOR POSITION.

```
tab_clear            inout = (1B(16) 32(16))
```

Tab_clear_all is the sequence to clear all tab stops.

THE ESC 3 SEQUENCE IS USED BY THE T920C TO CLEAR ALL TABS.

```
tab_clear_all        inout = (1B(16) 33(16))
```

Tab_forward is the sequence sent (and received) when tabbing from a tab stop or unprotected field to the next tab stop or unprotected field.

THE T920C USES THE CONTROL CODE 09(16) TO TAB FORWARD.

```
tab_forward          inout = 09(16)
```

Tab_set is the sequence to set a tab stop at the current cursor position.

THE ESC 1 SEQUENCE IS USED BY THE T920C TO SET A TAB WHERE THE CURSOR IS POSITIONED.

```
tab_set              inout = (1B(16) 31(16))
```

MISCELLANEOUS TERMINAL SEQUENCES

Bell_nak is the sequence to ring the bell on your terminal.

MOST TERMINALS USE THE 07 CONTROL CODE FOR THE BELL.

```
    bell_nak            out = 07(16)
```

Bell_ack is reserved for future use.

```
    bell_ack            out = ()
```

Display_begin is reserved for future use.

```
    display_begin       out = ()
```

Display_end is reserved for future use.

```
    display_end         out = ()
```

Field_scroll_down is reserved for future use.

```
    field_scroll_down   out = ()
```

Field_scroll_set is reserved for future use.

```
    field_scroll_set    out = ()
```

Field_scroll_up is reserved for future use.

```
    field_scroll_up     out = ()
```

Output_begin is the sequence that will be sent before each stream of output is sent downline to the terminal. This should include the sequence to disable protect if the terminal supports it as well as the sequence to exit insert mode if the terminal has an insert mode.

```
    output_begin        out = ()
```

Output_end is the sequence that will be sent after each stream of output (and therefore before the next request for input) is sent downline to the terminal. This should include the sequence to enable protect if the terminal supports protect.

```
    output_end          out = ()
```

Print_begin is reserved for future use.

```
print_begin          out = ()
```

Print_end is reserved for future use.

```
print_end            out = ()
```

Print_page is reserved for future use.

```
print_page           out = ()
```

Protect_all is the sequence that will set the protect bit for all character positions on the screen. For some terminals that have protect this will be an empty string (an example is a terminal that uses a clear screen to protected character positions sequence to accomplish this function).

```
protect_all          out = ()
```

Reset is reserved for future use.

```
reset                out = ()
```

Return is reserved for future use.

```
return               out = ()
```

PROGRAMMABLE FUNCTION KEY INPUT INFORMATION

All full screen products use programmable function keys so that a user can tell the full screen product what they want to do next. Programmable function keys in the full screen editor allow a frequently used command to be reduced to pressing the correct function key (or required sequence of keys) for the terminal in use.

This section allows you to define what input sequences will be sent upline by your terminal to be recognized as programmable function keys. These should be entered for at least F1 - F6 and should be defined for all of the keys if possible.

Procedures run in screen mode will require only F1 - F6 to execute correctly but local screen formatting application programs that use programmable function keys to drive menus or to terminate form type input may require that more programmable functions keys than just F1 - F6 be defined in this file.

Escape or control sequences such as ESC - 1 for F1 can be a good way to define programmable function keys but take care not to use sequences that conflict with terminal hardware sequences. These are input only sequences so the IN keyword is used here.

THE T920C HAS 11 FUNCTION KEYS. THE CONTROL SEQUENCES WERE GIVEN IN THE HARDWARE REFERENCE MANUAL AS FOLLOWS:

F1 SAYS FUNCTION KEY 1 (UNSHIFTED)

F1_S SAYS FUNCTION KEY 1 (SHIFTED)

```
    f1          in = (01(16) 40(16))
    f2          in = (01(16) 41(16))
    f3          in = (01(16) 42(16))
    f4          in = (01(16) 43(16))
    f5          in = (01(16) 44(16))
    f6          in = (01(16) 45(16))
    f7          in = (01(16) 46(16))
    f8          in = (01(16) 47(16))
    f9          in = (01(16) 48(16))
    f10         in = (01(16) 49(16))
    f11         in = (01(16) 4A(16))
    f12         in = ()
    f13         in = ()
    f14         in = ()
    f15         in = ()
    f16         in = ()
    f1_s        in = (01(16) 60(16))
    f2_s        in = (01(16) 61(16))
    f3_s        in = (01(16) 62(16))
    f4_s        in = (01(16) 63(16))
    f5_s        in = (01(16) 64(16))
    f6_s        in = (01(16) 65(16))
    f7_s        in = (01(16) 66(16))
    f8_s        in = (01(16) 67(16))
    f9_s        in = (01(16) 68(16))
    f10_s       in = (01(16) 69(16))
    f11_s       in = (01(16) 6A(16))
    f12_s       in = ()
    f13_s       in = ()
    f14_s       in = ()
    f15_s       in = ()
    f16_s       in = ()
```

CDC STANDARD FUNCTION KEY INPUT INFORMATION

All full screen products use what are called CDC standard function keys. These are keys that have the same meaning to a particular full screen product regardless of the terminal in use. Each of these keys also corresponds to a physical key on the CDC 721 (Viking X) terminal.

The next section allows you to define what input sequences the terminal you wish to use will send upline to be recognized as CDC standard function keys. This capability will make all full screen products more usable to the end user but is not required when using the Full Screen Editor or procedures in screen mode.

Local screen formatting applications that have been written to use CDC standard function keys (rather than programmable function keys described in the previous section) to drive menus or to terminate form type input may require that at least some CDC standard function keys be defined in this file.

Escape or control sequences such as ESC - F for Forward can be a good way to define CDC standard function keys but take care not to use sequences that conflict with terminal hardware sequences. These are input only sequences so the IN keyword is used here.

SINCE MOST OF THE ESCAPE AND CONTROL SEQUENCES ARE USED BY THE TERMINAL TO PERFORM ITS NORMAL FUNCTIONS, IT LEFT LITTLE TO USE FOR THIS TYPE OF FUNCTION KEY. I WILL AT LEAST DEFINE THE STOP (CTL/T) AND STOP_S (CTL/P) WHICH WILL ENABLE THE USER TO SEND THE STOP/ABORT SIGNAL TO THE FULL SCREEN MANAGEMENT APPLICATION.

```
back      in = ()
back_s    in = ()
help      in = ()
help_s    in = ()
stop      in = 14(16)
stop_s    in = 10(16)
down      in = ()
down_s    in = ()
up        in = ()
up_s      in = ()
fwd       in = ()
fwd_s     in = ()
bkw       in = ()
bkw_s     in = ()
edit      in = ()
edit_s    in = ()
```

```
data      in = ()
data_s    in = ()
```

TERMINAL VIDEO ATTRIBUTES

These attributes are used mainly by screen formatting applications to define various types of fields (though protect_begin and end as well as inverse_begin and end or alternate_begin and end where they are available are used by FSE).

Define the attributes sequences below as described in the hardware reference manual for your terminal. The only restriction is that attributes that require an actual character position on the screen can not be used. If your terminal has a protect mode that uses a video attribute such as alternate video (either bright or dim) then you will want to place these sequences in the protect_begin and protect_end statements. These sequences are output only hence the OUT keyword is used here.

REMEMBER THAT THE T920C USES A CHARACTER POSITION TO TURN ON/OFF AN ATTRIBUTE. THEREFORE WE CANNOT DEFINE ANY ATTRIBUTES.

Alt_begin is the sequence to cause subsequent characters sent downline to be displayed in an alternate intensity (which may be bright or dim on your terminal).

```
alt_begin          out = ()
```

Alt_end is the sequence to cause subsequent characters sent downline to be in normal intensity.

```
alt_end            out = ()
```

Blink_begin is the sequence to cause subsequent characters sent downline to be displayed with a blinking attribute.

```
blink_begin        out = ()
```

Blink_end is the sequence to cause subsequent characters sent downline not to be displayed with the blinking attribute.

```
blink_end          out = ()
```

Hidden_begin is the sequence to set the hidden attribute for subsequent characters so that data typed in this area can not be seen on the screen (also called a guarded attribute).

```
hidden_begin        out = ()
```

Hidden_end is the sequence to return to visible characters.

```
hidden_end          out = ()
```

Inverse_begin is the sequence to cause subsequent characters to be displayed in inverse video.

```
inverse_begin       out = ()
```

Inverse_end is the sequence to return to normal video.

```
inverse_end         out = ()
```

Protect_begin is the sequence to cause subsequent characters sent downline to the terminal to be protected, which means data can not be typed in these character positions on the screen.

```
protect_begin       out = ()
```

Protect_end is the sequence to return to unprotected mode.

```
protect_end         out = ()
```

Underline_begin is the sequence to cause subsequent characters sent downline to be displayed with an underline attribute.

```
underline_begin     out = ()
```

Underline_end is the sequence to cause subsequent characters sent downline to no longer be underlined.

```
underline_end       out = ()
```

LOGICAL ATTRIBUTE SPECIFICATIONS

Logical attributes are used mainly by screen formatting applications to define various types of fields. Procedures run in screen mode, for example, define all input variables for a procedure as logical type INPUT TEXT, which assures that they are underlined for those terminals that have that

capability or that any blanks in the variables are replaced with hypen characters on the screen to make them easily recognizable. You may define the logical attributes below as any combination of physical attributes by using the sequences to turn them on and off, or use any other displayable type function (except an attribute that will require an actual character position on the screen) that your terminal supports, such as RED_ON for error_begin and RED_OFF for error_end.

ERROR

```
error_begin          out = ()
error_end            out = ()
```

INPUT TEXT

If your terminal supports protect by use of a video attribute such as alternate intensity for unprotected areas of the screen, you should define input_text_begin and end accordingly so that screen formatting applications display the input fields correctly as unprotected areas.

```
input_text_begin     out = ()
input_text_end       out = ()
```

ITALIC

If your terminal supports an alternate character set then here is a place that you can make use of it with screen formatting applications.

```
italic_begin         out = ()
italic_end           out = ()
```

MESSAGE

Attributes displayed here will be used when printing help and error messages on the first line of the screen when a screen formatting application is running. Use any physical attributes that you wish but remember that if your terminal has a video attribute based protect capability this area should be protected data.

```
message_begin        out = ()
message_end          out = ()
```

OUTPUT TEXT

For output only data, so if your terminal has a video attribute based protect capability, this area should be protected data.

```
output_text_begin    out = ()
output_text_end      out = ()
```

TITLE

```
title_begin          out = ()
title_end            out = ()
```

LINE DRAWING CHARACTER SPECIFICATION

Line drawing character sets that your terminal supports should be specified here for use with the box drawing capability found in NOS screen formatting. There are three line weights, fine, medium, and bold, each with a sequence to enable and disable that weight and with eleven characters that represent the corners, edges and intersections for the corresponding line drawing character set.

If your terminal has the capability of actual line drawing, then place the sequences to turn the line drawing on and off in the ld_fine_begin, ld_fine_end and so on for up to three types of line drawing sets (you may specify the same sequences for all three or for any two if your terminal does not have three line drawing sets). If your terminal has no line drawing then the use of a hyphen character for a horizontal character, a colon or like character for a vertical line, and asterisks for all corners and intersections is suggested. In this case the ld_fine_begin, ld_fine_end sequences would be blank though you could use a terminal attribute such as alternate intensity.

Also for a bold line drawing character set you can define all characters as blanks (' ') and use inverse_on and inverse_off as the ld_bold_begin and ld_bold_end sequences.

SINCE WE CANNOT USE ANY OF THE T920C ATTRIBUTES I SUGGEST THAT WE USE THE - CHARACTER FOR DRAWING HORIZONTAL LINES, THE ¦ CHARACTER FOR DRAWING VERTICAL LINES AND THE * CHARACTER FOR INTERSECTION POINTS AND BOX CORNERS.

Fine Line Drawing Begin and End Sequences.

```
    ld_fine_begin           out = ()
    ld_fine_end             out = ()
```

Horizontal and Vertical Characters.

```
    ld_fine_horizontal      out = ('-')
    ld_fine_vertical        out = ('¦')
```

Box Corner Characters.

```
    ld_fine_upper_left      out = ('*')
    ld_fine_upper_right     out = ('*')
    ld_fine_lower_left      out = ('*')
    ld_fine_lower_right     out = ('*')
```

Intersection Characters.

```
    ld_fine_up_t            out = ('*')
    ld_fine_down_t          out = ('*')
    ld_fine_left_t          out = ('*')
    ld_fine_right_t         out = ('*')
    ld_fine_cross           out = ('*')
```

Medium Line Drawing Begin and End Sequences.

```
    ld_medium_begin         out = ()
    ld_medium_end           out = ()
```

Horizontal and Vertical Characters.

```
    ld_medium_horizontal      out = ('-')
    ld_medium_vertical        out = ('¦')
```

Box Corner Characters.

```
    ld_medium_upper_left      out = ('*')
    ld_medium_upper_right     out = ('*')
    ld_medium_lower_left      out = ('*')
    ld_medium_lower_right     out = ('*')
```

Intersection Characters.

```
    ld_medium_up_t            out = ('*')
    ld_medium_down_t          out = ('*')
    ld_medium_left_t          out = ('*')
    ld_medium_right_t         out = ('*')
    ld_medium_cross           out = ('*')
```

Bold Line Drawing Begin and End Sequences.

```
    ld_bold_begin           out = ()
    ld_bold_end             out = ()
```

Horizontal and Vertical Characters.

```
    ld_bold_horizontal      out = ('-')
    ld_bold_vertical        out = ('¦')
```

Box Corner Characters.

```
    ld_bold_upper_left      out = ('*')
    ld_bold_upper_right     out = ('*')
    ld_bold_lower_left      out = ('*')
    ld_bold_lower_right     out = ('*')
```

Intersection Characters.

```
    ld_bold_up_t            out = ('*')
    ld_bold_down_t          out = ('*')
    ld_bold_left_t          out = ('*')
    ld_bold_right_t         out = ('*')
    ld_bold_cross           out = ('*')
```

DEFAULT KEY DEFINITIONS FOR THE FULL SCREEN EDITOR

Here is where the default function key sequences that will be used by the full screen editor are defined. Using the variables defined earlier (see VARIABLES FOR FULL SCREEN EDITOR FUNCTION KEY DEFINITONS around line fifty) the six function keys our example terminal has are defined.

The keyword here is APPLICATION STRING (the three dots indicates a line continuation to TDU) and the name used is FSEKEYS which will be recognized by FSE. The out sequence is just the previously defined variable strings separated by semi-colons to make a correct FSE command. In addition to default function key sequences, here is a good place to put a SET TAB command if your terminal has predefined hardware tabs. Simply define a variable as was done with k1 through k6 as s1 = ('st 7 11 14 24 34 44 54 64') and include it in one of the out sequences below.

REMEMBER THAT WE PRE-DEFINED THE K1 THRU K11 KEYS AT THE BEGINNING OF THIS TDUIN FILE. NOW WE WILL JUST USE THEM WITHIN THE APPLICATION STRINGS.

```
application_string...
name=('FSEKEYS')...
out=(K1 ';' K2 ';' K3 ';' K4 ';' K5 ';' K6 ';' K7 ';' K8 ';'
    K9)
application_string...
name=('FSEKEYS')...
out=(K10 ';' K11 ';' K1S ';' K2S ';' K3S ';' K4S ';' K5S ';'
    K6S)
application_string...
name=('FSEKEYS')...
out=(K7S ';' K8S ';' K9S ';' K10S ';' K11S)
```

Now that you have completed your TDUIN file you need to execute the TDU command. It should compile this file and produce a local file called TERMLIB (or add the capsule for this terminal to a file called TERMLIB, such as the one from UN=LIBRARY, that is already local). Replace this file and then whenever the SCREEN,model_name command is executed you will see a local file called ZZZZTRM that will allow you to interact with all NOS full screen products.

ADD YOUR TERMINAL NAME BELOW FOR DOCUMENTATION.

END OF TERMINAL DEFINITION FILE FOR T920C TERMINAL.

********** END OF TDUIN FILE FOR T920C **********

Now that we have the TDUIN file built, we can execute the TDU procedure to compile our TDUIN file, fix any errors, and try it out with FSE and Screen Formatting. The TDU procedure follows.

```
.PROC,TDU*I"Terminal Definition Utility",
 I'Terminal definition file (TDUIN)'=(*F,*N=TDUIN),
 L'Error listing file (OUTPUT)'=(*F,*N=OUTPUT),
 LIB'Library file (TERMLIB)'=(*F,*N=TERMLIB).
.HELP.
TDU - Terminal Definition Utility.

The TDU procedure compiles into capsule format a user-
defined terminal definition file, inserting the new
capsule into a local user library.
.HELP,I.
The name of the user-defined terminal definition file
( default is TDUIN ).
.HELP,L.
The name of the listing file ( default is list ).
.HELP,LIB.
```

```
The name of the library ( default is TERMLIB ).
.ENDHELP.
.IFE(.NOT.FILE(I,AS),NOINPUT)
REVERT.  NO INPUT FILE I.
.ENDIF(NOINPUT)
REWIND,ZZZZZTL,ZZZZZTB,ZZZZZTA,LIB.
RETURN,ZZZZZTC.
TDUEX,I,ZZZZZTC,L.
IF(.NOT.FILE(ZZZZZTC,AS),TDUERRS)
REVERT.  COMPILATION FAILED.
ENDIF(TDUERRS)
COMPASS,#I=ZZZZZTC,#L=ZZZZZTL,B=ZZZZZTB.
LOAD,ZZZZZTB.
NOGO,ZZZZZTA.
.IF(FILE(LIB,AS),REPLACE)
  ULIB,R,ZZZZZTA,LIB.
.ELSE(REPLACE)
  ULIB,C,ZZZZZTA,LIB.
.ENDIF(REPLACE)
RETURN,ZZZZZTB,ZZZZZTA,ZZZZZTC.
REWIND,LIB.
REVERT.  I --> LIB.
EXIT.
REVERT.  COMPILATION FAILED.
```

To execute TDU, execute:

```
BEGIN,,TDU.
```

When complete, try the screen command to identify your new terminal. If this succeeds, try FSE. If there are any errors, fix them and rerun TDU. Repeat this process until there are no errors and the screen management products work for your terminal.

A set of TDUIN files will be maintained by Central Software Support. At present the following terminal definitions are available.

CDC 721	DEC VT100
CDC 722	TEKTRONIX 4115
CDC 752	TELEVIDEO 920C
LEAR SIEGLER ADM3A	ZENITH Z19/H19
LEAR SIEGLER ADM5	

If you would like to donate a TDUIN file for a particular terminal, we would be happy to be the focal point for distribution. Contact us at the address below either to donate a particular TDUIN file or if you need one.

Control Data Corporation
Central Software Support
ARH213
4201 North Lexington
St. Paul, Mn. 55112
Toll Free (from within U.S.) - (800)-328-9567
Or (612)-482-3074
Or CDC Controlnet 235-3074

(This page left intentionally blank.)

CHAPTER 9

NP/QTF, PTF

9.1 NP/QTF, PTF

The applications PTF (Permanent file Transfer Facility) and QTF (Queue file Transfer Facility) have been enhanced to interface to the NAM subsystem. This will allow these applications to transfer files on NAM supported networks as well as RHF (Loosely Coupled) networks. These networks may coexist and the user interface is identical.

This chapter addresses the NAM interface to these applications. Although the names of these applications are the same as those that use RHF (Loosely Coupled Network), they are a seperate product.

9.2 General Overview

The PTF and QTF applications each contain two parts, the "initiator" and the "server". During file transfer, the initiator executes in the mainframe where the transfer request started and the server executes in the other mainframe. In a configuration with both NAM and RHF there will be two variants of each application. There is a QTF

server/initiator pair for RHF and a QTF server/initiator pair for NAM. There is a PTF server for both NAM and RHF but in order to maintain a single user interface, the PTF initiator (MFLINK) will dynamically determine which network to use.

9.3 Compatibility

9.3.1 User Compatibility

This version of PTF and QTF is end user upward and downward compatible with earlier versions. This means the user's job or procedure will function properly without any changes.

9.3.1.1 LISTLID Utility

The LISTLID utility will allow the end user to obtain a listing of the LID (Logical Identifier) table. LIDs with the NLIST attribute cannot be listed by the end user. The format of the LISTLID command is:

LISTLID(p1,p2,...pn)

where p1 through pn are the parameters as follows:

LID=xxx	Print the attribute of PIDs in which LID xxx exists, is enabled and listing is allowed. If the LID keyword is omitted, the default is to list all listable LIDs.
L=lfn	Output is written on file lfn. If the L parameter is omitted, the default is file OUTPUT.
PID=yyy	List the attributes of LIDs under PID yyy. If yyy is omitted, all LIDs are listed.

The following lists various formats of the LISTLID command and the type of the output received.

LISTLID.

List all LIDs in the LID table.

LISTLID,LID=xxx.

List all xxx LIDs in the LID table.

LISTLID,LID=xxx,PID=yyy.

List LID xxx found under PID yyy.

LISTLID,LID=xxx,PID.

List all xxx LID entries.

LISTLID,PID.

List all PIDs and all LIDs in the LID table.

LISTLID,PID=yyy.

List all LIDs found under PID yyy.

9.3.2 System Compatibility

9.3.2.1 IPRDECK Changes

Extensive changes have been made in the way that the system manages LIDs. At previous levels, the LIDs were defined in the IPRDECK. LIDs are now defined using a LID configuration file for each mainframe. The format of this file is described below. All LID entries in the IPRDECK must be deleted.

9.3.2.2 CMRDECK Changes

Space must be allocated for the LID table. This was done at previous levels by using the "LIDT=n" entry in the CMRDECK. The value of "n" was the number of LIDs defined for this mainframe. Space is now allocated by using the "LDT=nnnn" CMRDECK entry. Where "nnnn" is the number of words to allocate for the LID table. The value of "nnnn" is calculated by the formula:

nnnn=(3 + LID)*PID + 1

PID total number of PIDs in all networks

LID number of LIDs per PID

The old "LIDT" entry must be deleted from existing CMRDECKs.

9.3.3 Operator Compatibility

9.3.3.1 QTF Initiation

QTF will be brought up automatically by the NAM subsystem. In case of QTF failure the operator may bring up QTF through a DSD command. Since there are two versions of the QTF application, operators will have to specify which version to execute when they begin QTF. The format of the QTF command is:

X.QTF(p1)

p1 "RHF" or "NAM" depending on which interface is desired.

9.3.3.2 LIDOU Utility

During system operation LIDs and LID attributes may be added or deleted from the LID table by using the LIDOU utility. Modified LID tables are retained across all recovery deadstarts (levels 1, 2 and 3). Level 0 deadstarts will return to the LID configuration defined in the configuration file. The following is a list of the LIDOU directives:

END	terminate LIDOU
OUT	route a listing of the LID table
HELP	list available directives
BACK	back to original display
+	page L display forward
-	page L display backward
pid,PA=xxz	set (all applicable) attributes
pid,NT=yyy,NA=z	enable/disable network status
pid,MF=aaaaaaa	set mainframe description
pid,LD=bbb,LA=ccccz	add or set attributes for LID
pid,LD=bbb,DELETE	delete LID

Where:

pid	physical identifier
bbb	LID
xx	V - validate or N - nolist
z	E - enable or D - disable
yyy	RHF, NAM, SSF or ALL
aaaaaaa	1 to 7 character mainframe description

cccc S-store and forward, B-loopback, V-validate or N-nolist

9.3.3.3 RHF K Display

The RHF ID display has been removed. The information that was obtained from this display is now available from the LIDOU utility. The other RHF displays are unchanged.

9.4 LID Configuration File

The LID configuration file is an indirect access file stored under user name SYSTEMX (user index 377777B). The file name must be of the form "LIDCMid" where id is the mainframe machine ID. The LIDCMid file defines the PID/LID relationship and the attributes associated with them. The LID configuration file directives are similar to the RHF configuration file directives. These directives are documented in the NOS Analysis Handbook.

A new CPU program "CLDT" is called by CMS at deadstart to generate the LID table from the LIDCMid file. The operator may also enter X.CLDT at the console to build/rebuild the LID table if a change has been made to the LIDCMid file. This program needs system origin privileges and can only be executed when NAM and RHF are idle.

9.5 Format

The first line of the configuration file is the 1 to 7 character name of this configuration file. The remainder of the file describes the PIDs and their associated LIDs.

PID Definition

NPID, PID=pid, ENABLED=yes/no, MFTYPE=b,
AT=NVALID/VALID/NLIST/LIST, NETDIS=SSF/RHF/NAM.

PID — A unique 3 alphanumeric character physical identifier of the mainframe. The host PID must be of the form "Mid", where "id" is the manframe machine id. This is a required parameter.

ENABLED — Indicates whether the mainframe identified by this PID is available. Default is YES if not specified.

MFTYPE — Any 1 to 7 character string indicating the mainframe type. You may use any string meaningful to your site. This is a required parameter.

AT — Defines attributes for this PID. Specify either NVALID or VALID and either NLIST or LIST. Default attributes are NVALID and LIST.

VALID - indicates that USER command pre-validation is required. Use this when validation is the same on all mainframes.

NVALID - indicates that USER command pre-validation is not required.

LIST - indicates that this PID will be available for display to the end user through the LISTLID utility.

NLIST - indicates that this PID will not display with LISTLID.

NETDIS — Indicates which network accesses to the mainframe indicated by the PID are disabled. Default is that all accesses are enabled. NETDIS may not be specified for the host PID. Valid accesses are:

SSF - Scope Station Facility.

RHF - Remote Host Facility.

NAM - Network Access Methods.

LID Definition

NLID, LID=lid, ENABLED=yes/no,
AT=LOOPB/STOREF/LIST/NLIST/VALID/NVALID.

LID — A 3 character logical identifier for the mainframe identified by the last PID definition. The LID may be the same as the last PID. This parameter is required.

ENABLED — Indicates whether the mainframe identified by lid is available. Default is YES if not specified.

AT — Indicates the attributes associated with this LID. The attribute of LOOPB is valid only for LIDs defined for the Host mainframe. Either STOREF of LOOPB may be specified but not both.

LOOPB - indicates loop back capability for RHF testing or for accessing other files from different user/familes when secondary user commands are disabled.

STOREF - indicates that the queued file store and forward capability exists for this mainframe. Queued files may pass through this machine and into the network.

VALID - indicates that USER command pre-validation is required and can only be specified if the STOREF option is specified.

NVALID - USER command pre-validation is not required.

LIST - indicates that this LID should be available to the end user through the LISTLID utility.

NLIST - indicates that this LID should not be available to the end user through the LISTLID utility.

Comments may be placed anywhere in the file after the first line. A comment is any line which begins with an asterisk in the first column.

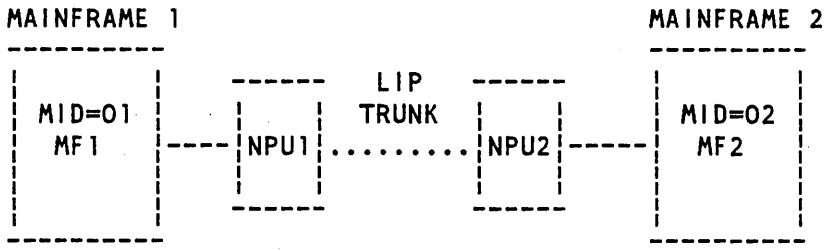
9.6 Examples

The following examples show two simple networks, one for 2 mainframes and the other for 3.

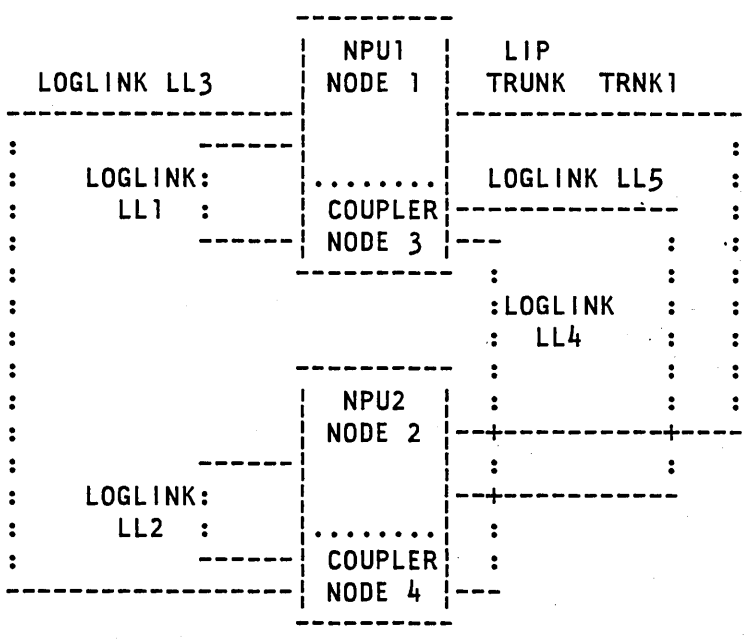
9.6.1 Example 1

The first is a two mainframe configuration allowing queued and permanent file transfers between each machine.

NETWORK CONFIGURATION



NPU CONFIGURATION



9.6.1.1 CMRDECK Entries

The following entries should be added to the CMRDECK for each mainframe.

9.6.1.1.1 Mainframe 1

```
MID=01.
LDT=15.
```

9.6.1.1.2 Mainframe 2

```
MID=02.
LDT=15.
```

9.6.1.2 LIDCMid Files

The following files need to be created and saved as indirect access files on user name SYSTEMX (UI=377777B).

9.6.1.2.1 Mainframe 1

```
LIDCM01
*
* Define the PID for M01 and force
* validation.
*
NPID, PID=M01, MFTYPE=MF1, AT=VALID, NETDIS=RHF/SSF.
NLID, LID=M01.
*
* Define M02 as a linked mainframe.
*
NPID, PID=M02, MFTYPE=MF2, NETDIS=RHF/SSF.
NLID, LID=M02.
```

9.6.1.2.2 Mainframe 2

```
LIDCM02
*
* Define the PID for M02 and force
* validation.
*
NPID, PID=M02, MFTYPE=MF2, AT=VALID, NETDIS=RHF/SSF.
NLID, LID=M02.
*
* Define M01 as a linked mainframe.
*
NPID, PID=M01, MFTYPE=MF1, NETDIS=RHF/SSF.
NLID, LID=M01.
```

9.6.1.3 NDL

The following NDL is an example to highlight the entries needed for PTF and QTF. These entries should be added to the site NDL as needed. NDL is fully documented in the NDL REFERENCE MANUAL (60480000).

```
EXAMP1: NFILE.

* Define the trunk connecting NPU1 to NPU2.

TRNK1: TRUNK, N1=NPU1, N2=NPU2, P1=1, P2=1.

* Define NPU1

NPU1: NPU, NODE=1, VARIANT=xxx.

* xxx is the CCP variant for NPU1

SUPLINK, LLNAME=LL1.
CPLR1: COUPLER, NODE=3, HNAME=M01, LOC=PRIMARY.

* Define the logical links for NPU1.

LL1: LOGLINK, NCNAME=NPU1.
LL4: LOGLINK, NCNAME=CPLR2.
LL5: LOGLINK, NCNAME=NPU2.
```

```
* Define the lines and terminals for NPU1 here.

* Define NPU2

NPU2: NPU, NODE=2, VARIANT=yyy.

* yyy is the CCP variant for NPU2

SUPLINK, LLNAME=LL2.
CPLR2: COUPLER, NODE=4, HNAME=MO2, LOC=PRIMARY.

* Define the logical links for NPU2

LL2: LOGLINK, NCNAME=NPU2.
LL3: LOGLINK, NCNAME=NPU1.

* Define the lines and terminals for NPU2 here.

* End of network division.

* Start the local divisions for each mainframe.

MO1LOC: LFILE.

* Define applications.

* put existing applications here

** NP/QTF, PTF applications.
QTF  :APPL, MXCOPYS=4, PRU, NETXFR.
QTFS :APPL, MXCOPYS=4, RS, PRU, NETXFR.
PTF  :APPL, MXCOPYS=4, PRU, NETXFR.
PTFS :APPL, MXCOPYS=4, RS, PRU, NETXFR.

* INCALL/OUTCALL STATEMENTS FOR QTF/PTF

INCALL, UNAME=NETOPS, ANAME=QTFS, DBL=7, ABL=7.
OUTCALL, NAME1=QTFS, PID=MO2, SNODE=3, DNODE=4, DBL=7,
ABL=7.

INCALL, UNAME=NETOPS, ANAME=PTFS, DBL=7, ABL=7.
OUTCALL, NAME1=PTFS, PID=MO2, SNODE=3, DNODE=4, DBL=7,
ABL=7.

* User statements for MO1 next.

* End of MO1 local file.
```

```
MO2LOC: LFILE.

* Define applications.

* put existing applications here

** NP/QTF, PTF applications.
QTF  :APPL, MXCOPYS=4, PRU, NETXFR.
QTFS :APPL, MXCOPYS=4, RS, PRU, NETXFR.
PTF  :APPL, MXCOPYS=4, PRU, NETXFR.
PTFS :APPL, MXCOPYS=4, RS, PRU, NETXFR.

* INCALL/OUTCALL STATEMENTS FOR QTF/PTF

INCALL, UNAME=NETOPS, ANAME=QTFS, DBL=7, ABL=7.
OUTCALL, NAME1=QTFS, PID=MO1, SNODE=4, DNODE=3,
DBL=7,ABL=7.

INCALL, UNAME=NETOPS, ANAME=PTFS, DBL=7, ABL=7.
OUTCALL, NAME1=PTFS, PID=MO1, SNODE=4, DNODE=3,
DBL=7,ABL=7.

* User statements for MO2 next.

* End of MO2 local file.

* End of NDL
END.
```

After executing NDLP, there will be three local files, EXAMP1, MO1LOC and MO2LOC. EXAMP1 is the network configuration file and should be placed on both mainframes. MO1LOC and MO2LOC are the local configuration file and should be placed on their respective mainframes. Be sure to use the file names referenced in the NAMSTRT file for these file names. The default permanent file names are NCFFILE for the network file and LCFFILE for the local file.

9.6.1.4 Example Jobs

Suppose a user is logged into MO1 and needs a file that resides on MO2. The commands needed to transfer the file are:

```
MFLINK,LFN,ST=M02.
*USER,usernam,passwor.  for M02
*GET(PFN) or ATTACH whichever is appropriate
*(CR)
```

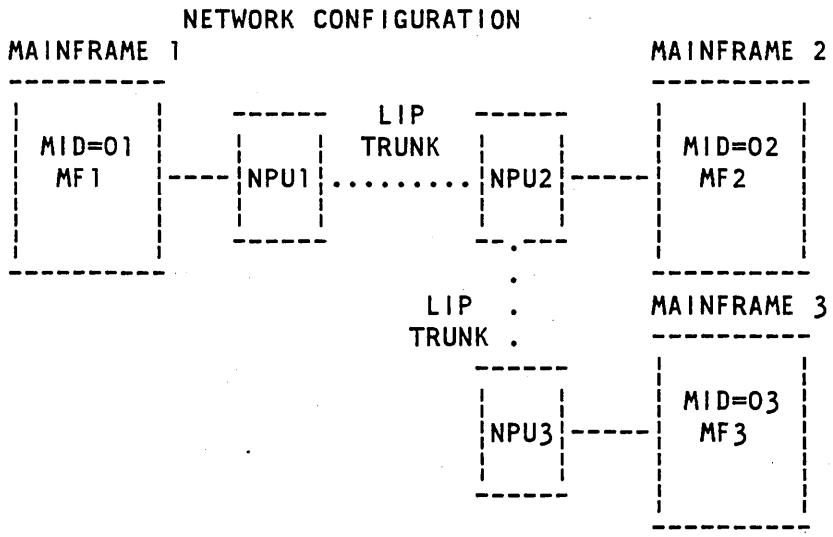
MFLINK prompts the user with the *. When the file transfer is complete, the local file LFN will contain the permanent file PFN.

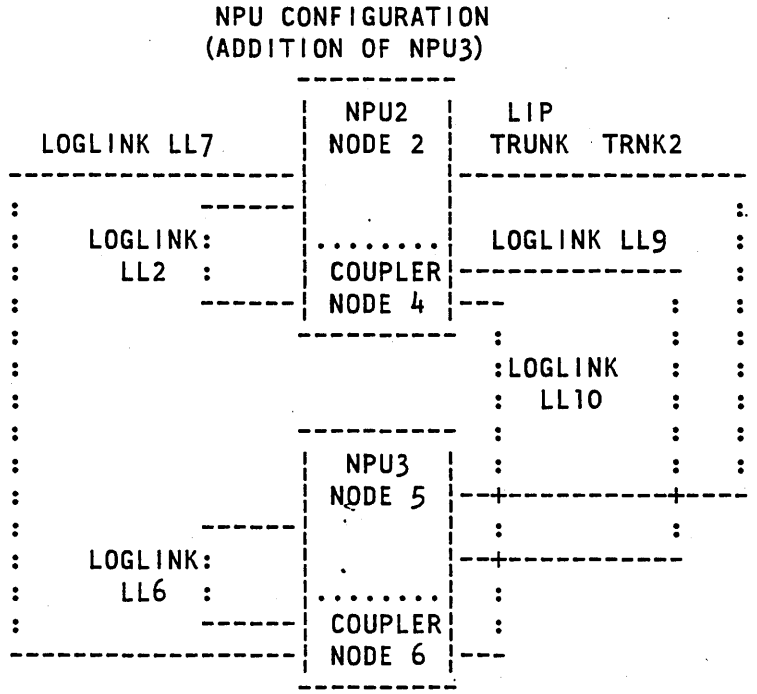
Again assume that the user is logged into M01 and now wants to execute a job on M02. The following job will perform a catlist on M02 and return the output to M01.

```
JOB,ST=M02.
USER,usernam,passwor.
CATLIST.
```

9.6.2 Example 2

The second example adds a third mainframe and NPU to the network. There is no direct connection between M01 and M03 so M02 will have to act as a store and forward link between these two machines. Note that permanent file transfers can not be performed directly through a store and forward link. This type of transfer may be done by sending a job from M01 to M02 that does a permanent file transfer from M03 to M02 then from M02 to M01.





9.6.2.1 CMRDECK Entries

The following entries should be added to the CMRDECK for each mainframe.

9.6.2.1.1 Mainframe 1

```
MID=01.
LDT=21.
```

9.6.2.1.2 Mainframe 2

```
MID=02.
LDT=23.
```

9.6.2.1.3 Mainframe 3

```
MID=03.
LDT=15.
```

9.6.2.2 LIDCMid Files

The following files need to be created and saved as indirect access files on user name SYSTEMX (UI=377777B).

9.6.2.2.1 Mainframe 1

```
LIDCM01
*
* Define the PID for M01 and force
* validation.
*
NPID, PID=M01, MFTYPE=MF1, AT=VALID, NETDIS=RHF/SSF.
NLID, LID=M01.
*
* Define M02 as a linked mainframe.
*
NPID, PID=M02, MFTYPE=MF2, NETDIS=RHF/SSF.
NLID, LID=M02.
NLID, LID=M03, AT=S.
```

9.6.2.2.2 Mainframe 2

```
LIDCM02
*
* Define the PID for M02 and force
* validation.
*
NPID, PID=M02, MFTYPE=MF2, AT=VALID, NETDIS=RHF/SSF.
NLID, LID=M02.
*
* Define the store and forward LIDs
*
NLID, LID=M01, AT=S.
NLID, LID=M03, AT=S.
*
* Define M01 as a linked machine.
*
NPID, PID=M01, MFTYPE=MF1, NETDIS=RHF/SSF.
NLID, LID=M01.
*
* Define M03 as a linked machine.
*
NPID, PID=M03, MFTYPE=MF3, NETDIS=RHF/SSF.
NLID, LID=M03.
```

9.6.2.2.3 Mainframe 3

```
LIDCM03
*
* Define the PID for M03 and force
* validation.
*
NPID, PID=M03, MFTYPE=MF3, AT=VALID, NETDIS=RHF/SSF.
NLID, LID=M03.
*
* Define M02 as a linked mainframe.
*
NPID, PID=M02, MFTYPE=MF2, NETDIS=RHF/SSF.
NLID, LID=M02.
NLID, LID=M01, AT=S.
```

9.6.2.3 NDL

The following NDL is an example to highlight the entries needed for PTF and QTF. These entries should be added to the site NDL as needed.

```
EXAMP2: NFILE.

* Define the trunks connecting NPU1 to NPU2
* and connecting NPU2 to NPU3.

TRNK1: TRUNK, N1=NPU1, N2=NPU2, P1=1, P2=1.
TRNK2: TRUNK, N1=NPU2, N2=NPU3, P1=2, P2=1.

* Define NPU1

NPU1: NPU, NODE=1, VARIANT=xxx.

* xxx is the CCP variant for NPU1

SUPLINK, LLNAME=LL1.
CPLR1: COUPLER, NODE=3, HNAME=M01, LOC=PRIMARY.

* Define the logical links for NPU1

LL1: LOGLINK, NCNAME=NPU1.
LL4: LOGLINK, NCNAME=CPLR2.
LL5: LOGLINK, NCNAME=NPU2.
```

```
* Define the lines and terminals for NPU1 here

* Define NPU2

NPU2: NPU, NODE=2, VARIANT=yyy.

* yyy is the CCP variant for NPU2

SUPLINK, LLNAME=LL2.
CPLR2: COUPLER, NODE=4, HNAME=MO2, LOC=PRIMARY.

* Define the logical links for NPU2

LL2: LOGLINK, NCNAME=NPU2.
LL3: LOGLINK, NCNAME=NPU1.
LL9: LOGLINK, NCNAME=NPU3.
LL10:LOGLINK, NCNAME=CPLR3.

* Define the lines and terminals for NPU2 here

* Define NPU3

NPU3: NPU, NODE=5, VARIANT=zzz.

* zzz is the CCP variant for NPU3

SUPLINK, LLNAME=LL6.
CPLR3: COUPLER, NODE=6, HNAME=MO3, LOC=PRIMARY.

* Define the logical links for NPU3

LL7: LOGLINK, NCNAME=NPU2.
LL6: LOGLINK, NCNAME=NPU3.

* Define the lines and terminals for NPU3 here

* End of network division.

* Start the local divisions for each mainframe.

MO1LOC: LFILE.

* Define applications.

* put existing application here

** NP/QTF, PTF applications.
QTF  :APPL, MXCOPYS=4, PRU, NETXFR.
QTFS :APPL, MXCOPYS=4, RS, PRU, NETXFR.
```

```
PTF  :APPL, MXCOPYS=4, PRU, NETXFR.
PTFS :APPL, MXCOPYS=4, RS, PRU, NETXFR.

* INCALL/OUTCALL STATEMENTS FOR QTF/PTF

INCALL, UNAME=NETOPS, ANAME=QTFS, DBL=7, ABL=7.
OUTCALL, NAME1=QTFS, PID=MO2, SNODE=3, DNODE=4, DBL=7,
ABL=7.

INCALL, UNAME=NETOPS, ANAME=PTFS, DBL=7, ABL=7.
OUTCALL, NAME1=PTFS, PID=MO2, SNODE=3, DNODE=4, DBL=7,
ABL=7.

* User statements for MO1 next.

* End of MO1 local file.

MO2LOC: LFILE.

* Define applications.

* put existing application here

** NP/QTF, PTF applications.
QTF  :APPL, MXCOPYS=4, PRU, NETXFR.
QTFS :APPL, MXCOPYS=4, RS, PRU, NETXFR.
PTF  :APPL, MXCOPYS=4, PRU, NETXFR.
PTFS :APPL, MXCOPYS=4, RS, PRU, NETXFR.

* INCALL/OUTCALL STATEMENTS FOR QTF/PTF

INCALL, UNAME=NETOPS, ANAME=QTFS, DBL=7, ABL=7.
OUTCALL, NAME1=QTFS, PID=MO1, SNODE=4, DNODE=3,
DBL=7,ABL=7.
OUTCALL, NAME1=QTFS, PID=MO3, SNODE=4, DNODE=6,
DBL=7,ABL=7.

INCALL, UNAME=NETOPS, ANAME=PTFS, DBL=7, ABL=7.
OUTCALL, NAME1=PTFS, PID=MO1, SNODE=4, DNODE=3,
DBL=7,ABL=7.
OUTCALL, NAME1=PTFS, PID=MO3, SNODE=4, DNODE=6,
DBL=7,ABL=7.

* User statements for MO2 next.

* End of MO2 local file.
```

```
M03LOC: LFILE.

* Define applications.

* put existing application here

** NP/QTF, PTF applications.
QTF  :APPL, MXCOPYS=4, PRU, NETXFR.
QTFS :APPL, MXCOPYS=4, RS, PRU, NETXFR.
PTF  :APPL, MXCOPYS=4, PRU, NETXFR.
PTFS :APPL, MXCOPYS=4, RS, PRU, NETXFR.

* INCALL/OUTCALL STATEMENTS FOR QTF/PTF

INCALL, UNAME=NETOPS, ANAME=QTFS, DBL=7, ABL=7.
OUTCALL, NAME1=QTFS, PID=M02, SNODE=6, DNODE=4,
DBL=7,ABL=7.

INCALL, UNAME=NETOPS, ANAME=PTFS, DBL=7, ABL=7.
OUTCALL, NAME1=PTFS, PID=M02, SNODE=6, DNODE=4,
DBL=7,ABL=7.

* User statements for M03 next.

* End of M03 local file.

* End of NDL
END.
```

After executing NDLP, there will be four local files, EXAMP2, M01LOC, M02LOC and M03LOC. EXAMP2 is the network configuration file and should be placed on both mainframes. M01LOC, M02LOC and M03LOC are the local configuration file and should be placed on their respective mainframes. Be sure to use the file names referenced in the NAMSTRT file for these file names. The default permanent file names are NCFFILE for the network file and LCFFILE for the local file.

9.6.2.4 Example Jobs

Suppose a user is logged into M01 and needs a file that resides on M03. The following job will transfer a file from M03 to M01:

```
JOB,ST=M02.
USER,usernam,passwor.  for M02
MFLINK(DUMMY,ST=M03)
*USER,usernam,passwor.  for M03
*GET(PFN) or ATTACH whichever is appropriate
MFLINK(DUMMY,ST=M01)
*USER,usernam,passwor.  for M01
*PURGE(PFN/NA)
*SAVE(PFN) or DEFINE whichever is appropriate
```

The * is required and must be the first character of the MFLINK commands.

Again assume that the user is logged into M01 and now wants to execute a job on M03. The following job will perform a catlist on M03 and return the output to M01.

```
JOB,ST=M03.
USER,usernam,passwor.  for M03
CATLIST.
```

(This page left intentionally blank.)

CHAPTER 10

NETWORK PRODUCTS X.25 ENHANCEMENTS

10.1 Network Products X.25 Enhancements for NOS 2.3

The following enhancements to Network Products' support of X.25 public data networks are included in the NOS 2.3 release.

1. PAD parameters may be set at network configuration time.

2. Features have been added to the X.25 terminal interface to increase its similarity to the asynchronous terminal interface, including keyboard input mode support and partial packet handling.

3. The network operator may dynamically manage Switched Virtual Circuits (SVCs).

4. 1980 CCITT standard is supported, including fast select facility, D-bit confirmation, one-way logical channel outgoing facility, and additional facility codes.

5. CDSN, UNINET, and C120 are supported as public data networks under Network Products.

6. Applications are allowed to supply their own OUTCALL block on application to application (A-A) connections, thus facilitating establishment of such connections to foreign hosts over X.25 networks.

7. Improved link control features have been added.

10.2 Description of New X.25 Features

10.2.1 Setting PAD Parameters via NDL

X.3 PAD parameters can be set by specifying them on the TERMDEV or TERMINAL statement in the Network Definition Language (NDL) configuration file.

If the site does not specify any PAD parameters, the Communications Control Program (CCP) will set the recommended parameters it set at the previous release. If the site specifies any PAD parameters, the site is responsible for specifying all references that must be set, including the separator between CCITT and National parameters (i.e. CCP will not set any by default).

CCP will send the parameters to the PAD when it receives a call request packet.

10.2.2 Terminal Interface Enhancements

Two features in the 2.3 release help to make terminals connected via the X.25 Terminal Interface Program (TIP) function in a manner more compatible with terminals connected via the ASYNC TIP.

1. Keyboard or block input mode. At the previous release, the X.25 TIP supported only block mode input. Now both block and keyboard input modes are supported. The TIP sets the data forwarding signal

PAD parameter, if necessary, according to the selected end-of-line, end-of-block, or transparent delimiter character.

As with the ASYNC TIP, the default is keyboard input mode unless changed in the NDL or by the terminal user or program.

2. More (M) bit not required. CCP has relaxed its requirement that the M bit be set in a partial packet, thus supporting data forwarding on idle timer. CCP now checks the last character in the packet as well as the M bit. If the last character is not an end-of-line (or end-of-block or transparent delimiter, as appropriate), the packet is treated as if the M bit were set.

10.2.3 SVC Management and Accounting

New parameters on the TERMINAL and TERMDEV NDL statements allow the site to specify minimum and maximum numbers of switched virtual circuits to be used for terminal (PAD) and for A-A connections. The network operator is then able to status, enable, and disable SVCs via host operator (HOP) / NPU operator (NOP) commands.

Two additional fields in the line statistics message can help the site determine if the parameters need to be adjusted. These fields contain the number of PAD and A-A call requests that have been rejected due to insufficient enabled circuits.

10.2.4 1980 CCITT Standards

Features of the 1980 CCITT standards which were not already supported by Network Products have been added.

1. D-bit confirmation. If a received data packet has the D-bit set, CCP will send out the receive ready (RR) packet for it even if the receiving window is not closed. (Formerly CCP sent RR packets only

when the receiving window was closed.) CCP does not generate data packets with the D-bit set.

2. One-way logical channels. A new parameter on the LINE statement in the NDL allows the site to restrict outgoing calls, for A-A connections, to certain logical channels. If this parameter is included, CCP will initiate calls only on a particular range of channels.

3. Fast select. To support the fast select facility, Network Products now handles 128 octets of call user data in outgoing and incoming A-A calls. The call user data can be supplied on the NDL OUTCALL statement or by the application.

4. Additional facilities. The INCALL statement has new keywords for reverse charge acceptance and flow control negotiation. The site can also use the FACn parameters to specify other facility codes, requiring no action, which should be allowed to appear on an incoming A-A call request.

10.2.5 Additional X.25 Network Types Supported

New legal values for the PSN parameter on the X.25 LINE statement are

CDSN (Cybernet's Control Data Shared Network),
UNINET (of United Telecom),
and C120 (for NOS to CYBER 120 terminal access or file transfers over a direct X.25 trunk).

CCP uses the declared PSN to send the correct clear, reset, and restart packets and to set up the default X.3 PAD parameters.

10.2.6 Application Supplied OUTCALL Parameters

A privileged network application is permitted to supply its own OUTCALL block parameters in the application connection request. Thus the application can have more control over

and correspondingly more flexibility with its outgoing call requests, and reduce its dependency on the NDL.

The Network Validation Facility (NVF) merges the information from the application with the addressing information in the NDL to choose a matching, accessible connection path.

10.2.7 Improved X.25 Link Control

1. CCP will detect link level failures, and will attempt to reset the link if it continues to receive no response to polls. If that is unsuccessful, it will notify the network operator.

2. CCP clears the virtual circuit if two minutes elapse without a host connection, to bring X.25 disconnect processing into conformity with that for all other dial-up lines.

(This page left intentionally blank.)

CHAPTER 11

3270 TIP STANDARDIZATION

With the NOS 2.3 release the 3270 TIP is supported as a standard product. It is essentially the same TIP that has previously been sold by Professional Services under Product Number 720-001B. If you used the Professional Services TIP then you MUST rebuild your NDL before you can use the standardized version.

There are a few minor variations between the two TIPS. The standardized TIP supports WAIT.. / REPEAT.. input regulation and the Secure Login / Trusted Path feature.

The 3270 TIP supports up to 32 multi-dropped clusters of up to 32 devices on each dedicated or dial-up line. There can be a maximum of 32 devices per controller but no more than 8 line printers associated with any owning console. The 3270 TIP does not support the contention protocol, sometimes refered to as the dial-feature.

3270 terminals cannot be auto-recognized. You must declare a TIPTYPE of 3270 and a Terminal Class of 3270 in the NDL. Line type may be S1, S2 or S3. There is no STIP and the only Character Set supported is EBCDIC.

NOTE The 3270 TIP is not included in the release material when CCP is ordered. The 3270 TIP must be licensed and ordered separately.

(This page left intentionally blank.)

CHAPTER 12

MISCELLANEOUS NETWORK PRODUCTS USABILITY ENHANCEMENTS

12.1 Miscellaneous Network Products Usability Enhancements

AUTOLOG Parameter Removed

The AUTOLOG parameter is no longer an option of the DEVICE statement in the Network Definition Language, and will be flagged as illegal. Previously, CCP used the AUTOLOG flag in the connection request to signal NVF to read the auto-login information in the LCF (Local Configuration File). NVF now gets all pertinent auto-login data from the LCF for each connection request.

NAM K-Display

The NAM K-Display includes a self-documenting HELP command, and comprises both left and right screens.

Immediate Disconnect

Dial-up lines may be disconnected immediately, or timed out after two minutes as with the current default. This option can be selected using the IMDISC (IMmediate DISConnect) parameter on the LINE statement in the Network Definition Language.

Terminal Name on Host Availability Display

The Host Availability Display now includes entries showing the node number of the NPU to which the terminal is connected, and the symbolic name of the terminal as defined in the Network Configuration File via the Network Definition Language.

Single Terminal Command for Host Disconnect/Connect

A single command is available which allows a user at a terminal to terminate an existing connection to one host, then establish a new connection to another. It can be issued in either of two formats: %HC<cr> or %HC=hostname<cr> where % is the TIP control character, and <cr> is a carriage return. The shorter version establishes the connection with the currently selected host.

Changing Service Class of RBF Job

A new parameter on the RBF CHANGE statement (SC=xx, where xx is the mnemonic for the selected service class) may be specified to change the service class of a job in the input or the execution queue. If the SC parameter is used, then the queue type, the PRI parameter, and the REP parameter are not allowed on the CHANGE command.

Generating SAM-P Cassette on an NPU

The System Autostart Module Dump routine (SAM-D), used in generating SAM-P cassettes for remote NPU loading, may now be generated using a local NPU equipped with a cassette controller and tape unit.

QTRM Application Wait Time

An application using the QTRM (Queued Terminal Record Manager) interface may specify a wait time of up to 4095 seconds.

Application Auto-Restart, Multiple Copies

A network application can now be restarted when a connection request has been made to that application. Another new feature is that more than one copy of a given application can be connected to the network at the same time. See the RS and MXCOPYS parameters for the APPL statement in the Network Definition Language Reference Manual for details.

Ignore-Parity Option, Transparent Input Delimiter

For transparent input on asynchronous or X.25 type lines, it is sometimes desireable to ignore the parity bit when checking for the transparent input delimiter character. A new option on the TIP command for parity select (PA=I) causes the TIP to ignore the parity bit, so that only the low-order 7 bits are examined when checking for the transparent input delimiter character. No-parity (PA=N) differs from Ignore-parity in that No-parity causes the TIP to examine all 8 bits of input data for the delimiter character.

(This page left intentionally blank.)

CHAPTER 13

FAULT TOLERANT MAINFRAME ERROR PROCESSING

13.1 Introduction

When the C170-7xx product line was initially introduced, a philosophy was established within the operating system for handling processor related mainframe errors. This philosophy was based on the premise that the majority of errors would be 'hard' failures. On the basis of field experience, it has been discovered that a significant percentage of mainframe related interrupts are associated with transient/intermittent errors. In addition, current C170-8xx field experience exhibits an even greater percentage of these errors. In this environment, every error, hard or intermittent, which interrupts the processor will result in termination of the operating system.

Experience has also indicated that valuable resources such as PPs, disk space, and CPU time are being consumed for logging, analyzing, and reporting recovered processor errors.

The fault tolerant philosophy that this feature provides enables CDC systems to provide an operational environment that minimizes unscheduled interrupts as well as lessening the overhead involved in logging, analyzing, and reporting corrected and non-fatal uncorrected errors. This fault

tolerant philosophy is only used on C170-8xx (excluding the 865/875) and C180-8xx mainframes.

Throughout this chapter the term C180 refers to both C170-8xx (excluding 865/875) and C180-8xx mainframes.

13.2 Terminology

13.2.1 Detected Uncorrected Error (DUE)

These shall include but not be limited to the following malfunctions.

- Parity error(s) on transmissions to/from central memory from/to the processor(s).
- Non-correctable central memory data parity errors (SECDED) on central memory accesses.
- Other model-dependent conditions.

13.2.2 Process Not Damaged (PND)

Uncorrectable errors were detected at a point in the execution of an instruction whereby no process was yet damaged (i.e., the 'point of no return' with this instruction has not been passed).

13.3 Technical Solution

13.3.1 Corrected Processor Errors

A new central memory word, CECL, has been defined in CMR to hold the corrected processor and single bit CM and LCME (CYBER 176) error counters. Each time a corrected processor error is encountered, the corresponding processor error counter in this word will be incremented.

NOS will log the first 'n' corrected processor errors encountered each hour to the Binary Maintenance Log (BML). The actual number is controlled by the assembly constant, CPTL, in COMSMSC. The released value is 20B per processor. At the top of each hour the total number of corrected processor errors will also be logged in the BML before clearing the error counters.

13.3.2 Uncorrected Processor Errors

Recovery from uncorrected processor errors can be performed by the hardware, Environment Interface (EI) or NOS/VE, and NOS.

The hardware will automatically retry a processor instruction once that failed because of a DUE/PND error.

When NOS is running in 'stand alone' state the Environment Interface (EI) will retry a processor instruction that failed because of a DUE/PND error after the hardware retry attempt has failed. NOS/VE is responsible for this when NOS is running in dual state. A maximum number of eight attempts will be made by EI or NOS/VE.

Uncorrected processor errors are logged in a similar fashion as corrected processor errors. Word CECL is updated, and at the top of each hour the count of uncorrected processor errors will be logged to the BML. The assembly constant, UPTL, in COMSMSC is the threshold limit for reporting uncorrected processor errors. Also, the console operator will be notified if more than a total of 10B (released value for UPTL) uncorrected processor errors occur per hour.

If all hardware and EI or NOS/VE retry attempts have failed, or the error was not a DUE/PND error, NOS will regain control of the user job by the setting of an error mode 20B. At this point, the job cannot continue, but will be rerun, if possible. Before rerunning the job, NOS will check the integrity of memory associated with the job to ensure there are no double bit memory errors. Since rewriting a word will clear a transient double-bit error, the job's field length and any assigned Unified Extended Memory (UEM) is first zeroed out and then the entire field length including negative field length (NFL) and assigned UEM is re-read to verify there are no double bit memory errors.

To prevent the job from being rerun indefinitely, NOS will allow the job to be rerun only a predetermined number of times. This is controlled by the assembly constant, HRTL, which is defined in COMSMSC. The released value is 2. If this job rerun limit is exceeded, the job will then be terminated. Interactive jobs will not be rerun, but will only have the current job step terminated.

13.4 Design Overview

13.4.1 Error Processing

During deadstart, routine SET will now always build the Environment Interface Control Block (EICB) table for all C180 environments. Previously this table was only reserved in CMR when NOS/VE was enabled. Word 0 (DSCM) of this table is used by EI (or NOS/VE) to pass information concerning error states in the hardware to NOS.

MTR can call 1MB for maintenance register processing and/or 'top of the hour' processing if the inhibit 1MB call flag has not been set. 1MB processes all maintenance register errors on the C180 hardware. Maintenance register processing of single bit SECDED errors on mainframes with status/control registers is processed by MTR. If an error of this type has been detected, the respective counter in CECL is incremented.

Processing of memory and processor errors in 1MB for C180 mainframes will only checkpoint and step the system when absolute word zero is set non-zero (as when CPUMTR modes out). The appropriate counter in word CECL is updated, but no logging is done to the BML, and the maintenance registers are left unaltered. All other conditions are counted in CMR word CECL, logged to the BML, and cleared.

Since the hardware maintenance registers report uncorrected error status for each DUE, it is necessary for EI to inform NOS when this status is associated with a retry attempt. The following error codes are returned by EI in word DSCM of the EICB. The values marked by asterisks (***) indicate the error code returned when the retry count is exhausted.

- 41B = Cyber 170 State DUE
- 42B = Cyber 170 State DUE/PND (***)
- 44B = Cyber 180 State DUE
- 45B = Cyber 180 State DUE/PND (***)
- 61B = Cyber 170 State DUE/PND
- 65B = Cyber 180 State DUE/PND

A DUE or a DUE/PND with retry count exhausted will be considered as uncorrected processor errors by NOS and counted (in byte 0 of CECL), and logged as such. A total of the DUE/PND errors in which the retry was successful is kept in byte 1 and 2 (depending upon which CPU had the error) of CECL. 1MB conforms to the following rules concerning word CECL.

- Word CECL is written to the BML, if any of the five bytes are non-zero, at 'top of the hour' processing by 1MB. CECL is then zeroed out.
- All five bytes of CECL can only be incremented to 7777B.
- Logging of the corrected processor errors continues until the corresponding byte for that CPU exceeds the value CPTL (Corrected Processor Threshold Limit).

. On each uncorrected processor error per hour in which the count in byte 0 of CECL exceeds UPTL, the operator is notified via the 'ERRLOG ALERT' message. The message 'UNCORRECTED PROCESSOR ERROR THRESHOLD EXCEEDED' is placed in the system ERRLOG. The operator should inform the Customer Engineer and the Hardware Performance Analyzer (HPA) should be used to process the BML.

13.4.2 Job Processing

The Environment Interface (EI) will set error exit mode 20B whenever DUE or DUE/PND with retry unsuccessful has occurred. All other Cyber 170 state valid condition bits in the Monitor Condition Register (MCR) and User Condition Register (UCR) when set, will cause an error exit 67B. After the exchange to monitor mode, CPUMTR will set the PEET error code in the Control Point Area (CPA), if either of the above exit modes is present, and call 1AJ for job advancement.

1AJ/3AL was modified so that when processing a Parity Error (PEET) error flag set on a user job on a C180, CPUMTR function VFLM is called to zero out the job's field length and read/test the job's entire field length including its UEM field length. If no double-bit error is encountered, byte HRSS in the system sector is incremented and, if the result is not more than HRTL, the job is rerun (if not in 'no-rerun' status) using the new job termination code, HRJT (COMSMSC).

If the value in HRSS exceeds HRTL, then the new job abort code, HRIE (COMSDSP), is set into the JASS system sector byte and the job is rerun. 1AJ/3AA (begin job) was changed to detect this new job abort code and will abort the job with the dayfile message, HARDWARE RERUN THRESHOLD EXCEEDED.

Also, each time 1AJ reruns a job due to a recovered hardware error, no output will be queued for the job, and the following account dayfile message will be issued.

. AERR, HW.

However, if a double-bit error is encountered by CPUMTR when reading the job's entire field length, 1AJ will freeze the

job's field length, unconditionally terminate the job with the PEET error code still set, and drop from the PP. 1MB will soon be checkpointing and stepping the system for a deadstart, since CPUMTR received a mode error (i.e., word 0 of CMR is non-zero).

In addition, 1AJ now issues the message, PROCESSOR STATE ERROR, when processsing an exit mode 67B.

13.5 User Impact

13.5.1 Data Corruption

The user can regain control from a processor detected error through REPRIEVE (error class 001) or EREXIT processing but there is a chance that the data in the exchange package or within the job's field length has been corrupted. In this situation the user's program must be written in such a way that the corrupted data is not propagated into a data base or file.

13.5.2 Rerun Processing

The system automatically attempts to rerun a job if it has been aborted because of a processor error. User jobs which should not be automatically rerun should issue the NORERUN command at the appropriate point of execution.

13.6 Message Changes

13.6.1 System Dayfile

In addition to the dayfile messages already mentioned, the following have been expanded to contain the CPU number to which they are referring.

- CPUn POWER FAILURE
- CPUn FATAL ERROR
- CPUn SHUTDOWN IMMINENT

Where 'n' is either 0 or 1.

13.6.2 Binary Maintenance Log

The following changes were made to the uncorrected processor BML messages (message ID's of 232B, 233B, and 234B) that are currently issued by 1MB.

- Symptom codes 110B, 111B, and 112B are no longer used. These are replaced by the codes returned by EI or NOS/VE (biased by 100B) in the respective fields in word DSCM of the EICB.
- The retry field (bits 42-47 of word 3 of the BML message) is used to indicate how many software retry attempts EI or NOS/VE initiated. This infomation is in word DSCM of the EICB. 1MB was modified to set the retry field to a value of 77B if the number of retries is greater than 77B.
- The recovered/not recovered flag (bit 36 of word 3 of the BML message) is set for symptom codes 161B and 164B. This flag is clear for symptom codes 141B, 142B, 144B, and 145B.

- The CPU element number (0 or 1) is set into bits 18-23 of word 3 of these BML messages to identify which processor is at fault.

A new BML message is issued by 1MB at the top of the hour whenever any of the following events have occurred during the past hour.

- Single bit CM error
- Single bit LCME error (CYBER 176)
- Corrected processor error
- Uncorrected processor error

Word 3 of this BML message is a copy of word CECL from CMR at the end of the hour when this message is issued to the BML.

(This page left intentionally blank.)

CHAPTER 14

UEM CHECKPOINT

14.1 Introduction

Prior to NOS 2.3, the DSD command, CHECK POINT SYSTEM, saved all system critical information on mass storage except that which was resident in Unified Extended Memory (UEM).

UEM may contain rollout files, user file data, and alternate system libraries. Because UEM is part of central memory, and may be destroyed subsequent to a system checkpoint, it must be included in the system checkpoint file to be restored by a level 1 or 2 deadstart.

Since checkpointing UEM can result in much more time required to checkpoint the system, and more mass storage space to save the checkpoint file, the checkpoint processor was modified to execute the system checkpoint expeditiously and save the checkpoint file only on specified devices.

To decrease checkpoint execution time, the checkpoint processor now terminates or rolls out all control point jobs concurrently. A new EQPDECK entry, SCKP, specifies the devices on which the checkpoint file is to reside.

Checkpointing of UEM involves reading all non-user UEM tracks and writing to the checkpoint file only the sectors

containing system data. The UEM checkpoint data is written to the end of the system checkpoint file. User data in UEM is not checkpointed since it is assumed that this portion of UEM was saved when the user jobs were rolled out.

If both external extended memory and UEM are defined (as possible on 865/875), UEM will not be checkpointed since system data is stored in external extended memory, and the user data in UEM has been previously rolled out.

The new EQPDECK entry, SCKP, is documented in the NOS Analysis Handbook. All new and changed error messages are documented in Appendix A of the NOS Operations Handbook.

14.2 Design Overview

14.2.1 Level 0 Deadstart

The EQPDECK entry SCKP is processed during a level 0 deadstart by routine SET to define the devices on which the system checkpoint file will reside. It will be ignored on all other levels of deadstart. If no SCKP entries are successfully processed, a default device will be defined.

SET defines checkpoint devices by setting byte 3 of the DULL MST word to 3777B. The upper bit in this byte serves as a flag to indicate that a valid checkpoint file exists on this device.

REC calls ICK to preallocate the checkpoint file on each checkpoint device. ICK will first clear the upper bit of the checkpoint file pointer in the SLA (sector of local areas) for all mass storage devices and then determine the number of sectors to reserve for the maximum size of the checkpoint file by the following formula.

(CMR + total UEM - user UEM + 1/2*(RPL+RCL+PLD))/100B

If the sectors can be successfully reserved via the RTCM function, ICK will store the first track number in the checkpoint file pointer (upper bit cleared) in DULL of the MST, and save this pointer in the sector of local areas.

Otherwise, an appropriate message will be displayed and the deadstart will be aborted.

Both SET and DSD mass storage status displays have been modified to include the checkpoint file device status. A 'G' in the status field on these displays indicates that the device has been defined as a checkpoint file device.

14.2.2 System Checkpoint

The system checkpoint involves first rolling out user jobs and terminating (or rolling) subsystems, then checkpointing mass storage devices, and finally writing the checkpoint file. All the above processes are done by PP routine 1CK.

1CK has been changed to terminate selected subsystems without waiting for each to terminate before terminating the next. Each subsystem will be given a maximum interval in which to complete clean-up and termination. If not complete within that time, the subsystem will be rolled out to complete termination at deadstart time. The maximum interval is the assembly constant, MXTT, which is defined in COMSMSC. Its released value is 10D seconds.

After initiating the rollout of non-subsystem jobs, subsystems are rolled/terminated. Before checkpointing mass storage devices, all subsytems and jobs must have completed termination or have been rolled out.

1CK will then checkpoint all mass storage devices before writing the system checkpoint file to the specified mass storage device(s). The STBM monitor function is used to set the 'valid checkpoint flag' in DULL of the MST. Then all CM and UEM resident system information is written to the device. This will continue until either no devices remain or time has run out (abnormal environmental condition short warning).

14.2.3 Level 1 or 2 Deadstart

A new overlay, 4DK, has been added to RMS to provide for UEM recovery processing.

On level 1 or 2 recovery deadstarts, 4DK will search the EST for the first checkpoint device that has the upper bit set in the checkpoint file pointer byte in word DULL of the MST and will restore UEM. If all devices have been processed without successfully restoring UEM, RMS will hang with an appropriate error message.

After UEM has been recovered, 4DK will restore the UEM MST and TRT from the UEM label track on the checkpoint file.

CHAPTER 15

ALTERNATE PP/CRITICAL ERROR LOG

15.1 Background

Previous to NOS 2.3, the CTI (Common Test and Initialization) portion of CIP handed off data concerning the hardware configuration and environment to NOS by leaving tables and flags set in PP zero. This was done so that when the PP program SET was loaded into that PP, it could interpret that data and preset the operating system accordingly. Since both SET and the CIP tables resided in a single PP, the size of the CIP tables (and the amount of information in them) was limited.

With the release of CIP level 2, CTI has been enhanced to generate a great deal more data than it previously handed off to NOS. This enhancement to CTI causes it to dump maintenance registers to the Critical Error Log (CEL) if a hardware error is discovered during the deadstart process. A flag is set to alert the operating system that the data in the CEL must be logged to the Binary Maintenance Log (BML). Since not enough space was available in PP zero for SET, the CEL, and CIP tables, a new interface between CIP and NOS needed to be developed.

15.2 The Solution

CIP level 2 provides a PP, other than zero, which holds the handoff data and provides this data to other PPs. Although the previous data is still passed thru PP zero, this will be eliminated at some time in the future. The Critical Error Log is located on the deadstart disk. An I/O channel is used to pass commands to and data from the alternate PP.

SET locates the alternate PP and requests from it any data needed to preset the operating system. If this PP resides in a PP critical to deadstart (such as 1 or 2) it is moved to some other available PP. Finally, if Critical Error Log information is available, it is written to the BML by REC, when deadstart has almost completed.

With this solution, the format of the System Attribute Block in low core is changed. The CESAB macro in CETEXT is expanded to return either the new or old format. Also, NOS will not allow turning off PPs at deadstart time, and the PPU= CMRDECK entry is removed to prevent this. Failing PPs may still be turned off by using CIP.

The Alternate PP appears to be in the deadstart state, so the operating system can load PP resident into it at any time. Additional information is planned to be passed thru the alternate PP in the future.

CHAPTER 16

380-170 (NAD) CODE CONVERSION

16.1 Introduction

The Remote Host Facility (RHF) communicating on the Loosely Coupled Network (LCN) allows users to transfer files between homogeneous and heterogeneous mainframes. File transfers between heterogeneous mainframes may require character conversion to/from a local mainframe character set from/to a common network character set. NOS V2 and NOS/BE currently do this code conversion in the 170 CPU at the application level. Doing the code conversion in the 170 CPU is a burden on the CPU and has a direct impact on response time for other users. To alleviate the burden on the 170 CPU a decision was made to put the code conversion in the 170 NAD.

16.2 Advantages

- Off-loads the 170 code conversion from the mainframe to the NAD.

- Performance transfer rate equivalent to 170-730 CPU code conversion.
- CPU utilization is cut drastically in a code conversion transfer.
- Binary transfers when code conversion are not being performed are activated in the NAD.
- Configuration option to select CPU or NAD code conversion.
- Runs on existing NADs.
- Selectable at NAD level.

16.3 Constraints

- Requires 96K bytes of NAD memory (380-32 memory option for the 380-170 NAD).
- Supported with NOS/BE 1.5 L604 and NOS 2.3.
- Supported with QMOD controlware (MG401).
- Supported with CML 161.
- Not selectable at user level.

16.4 User Application Interface Changes

- The NETUXFR interface has a new input parameter.
- The NETXFR reply codes have been modified.
- The RHF file transfer FET has been modified.

See the RHF Access Method Reference Manual documentation changes section for further information.

16.5 Enabling Code Conversion In The NAD

The code conversion feature is enabled in the NAD by NDR (RHF PP driver) executing an Initialize NAD function on the local NAD. The Initialize NAD function is issued any time a local NAD changes from an off-state to an on-state in RHF's local NAD Table (all NAD's are in an off-state when RHF first comes up).

The number of paths and the number of buffers are given to RHF in the RHF configuration file (RCFILE), which is read by RHF at start-up time. To enter the convert mode parameters the RCFGEN LNAD macro has two new parameters (CMPATHS and CMBUFFS). The LNAD macro format:

```
laddr  LNAD   CH=ch,MAXNDRS=n,DEDICATE=status,CMPATHS=paths,
              CMBUFFS=buffers
```

Parameter	Description
laddr	Symbolic address referenced in a preceding PATH statement. This parameter is required and must begin in column one or two.
ch	Channel (octal) that NAD is on.
n	Maximum number of NAD drivers (NDRs) that may be assigned at one time to this NAD. (1 to 3) Default is 3.
status	Indicates whether the driver will always hold the NAD channel reservation between consecutive blocks of one I/O request. Values may be YES or NO. Default is YES. YES should always be specified unless some Non-CDC driver requires high performance access to the NAD channel.

paths	Maximum number of convert mode paths (0 to 63). Default is 0. If the number of convert mode paths to allow equals zero RHF assumes that code conversion should not be enabled in the corresponding local NAD.
buffers	Maximum number of convert mode buffers (0 to 63). Default is 0.

The number of convert mode paths is directly related to the number of convert mode buffers, and the number of convert mode buffers is directly related to the number of buffers in the NAD.

The minimum number of type 1 buffers that the NAD requires for normal activity is two. The minimum number of buffers required for code conversion to run is two. Buffers reserved by the NAD for code conversion cannot be used by the NAD for normal activities and vice-versa. Each path reserved for code conversion must have 1.5 code conversion buffers reserved (rounded up).

The software relies on the NAD to determine the actual number of paths and buffers reserved for code conversion. The algorithm used by the NAD is:

- Buffers Reserved = the smaller of (buffers requested) or (total buffers -2)

- Paths Reserved = the smaller of (paths requested) or (buffers reserved * 2/3)

The code conversion parameters should be adjusted according to the average number of concurrently active connections doing code conversion and the average size of the files being converted. The following table gives some suggested values for the code conversion parameters. In this table the term "average connection count" implies four converting connections. The term "average file size" is a file approximately 500 PRUs long.

	64K BYTE NAD	96K BYTE NAD	128K BYTE NAD
Less than average connections, average file size	NOT	2 paths 3 buffers	2 paths 3 buffers
Less than average connections, large file sizes	RECO	2 paths 5 buffers	2 paths 5 buffers
Average connections, Average file sizes:	MMEO	4 paths 6 buffers	5 paths 8 buffers
Greater than average connections, average file sizes:	DEDN	6 paths 9 buffers	7 paths 11 buffers
Greater than average connections, large file sizes		5 paths 9 buffers	6 paths 13 buffers

The local NAD Table entries, which are created by RCFGEN and maintained in RHF's field length, are modified to contain convert mode initialization parameters and current convert mode status. The LNAD entry format:

59 47	41	35	29	23	17	11	0
A:B:C:D:E:F:G:H:O:NB:NP:NAP:MAX NDR's				LNAD ORD	CHANNEL	EST ORD	O:
Reserved	Queue Count	Connect Count		Assgnd NDRs	NAD Q Address		

A = Dedicated Channel

B = NAD Status Queued

C = EST off

D = Status taken

E = NAD EST previously off

F = NAD controlware not loaded

G = NAD configuration error

H = NAD conversion on

NB = Number of Type 1 buffers to reserve for code conversion

NP = Number of Paths to be allowed concurrently in code conversion mode

NAP = Number of available convert mode paths

CHAPTER 17

UCOPY FOR CONTROL DATA CONNECT

17.1 Background

RMF, the Remote Micro Facility, has been available since 1982 for interfacing micros with CYBERs. RMF is available for a variety of micros, either from CDC or from customers. By itself, RMF provides simple terminal capabilities to a micro. When used with the CYBER resident program UCOPY, it allows files to be transferred, with error correction, between a micro and a CYBER.

Control Data has another micro product, called Control Data CONNECT, which is similar to RMF. It is more user friendly than RMF, and uses a slightly different file transfer protocol. Control Data CONNECT (herein referred to as CONNECT) was originally a CYBERNET product.

UCOPY is the command and the name of the CYBER program used by RMF and CONNECT to do error correcting file transfers. Even though RMF and CONNECT use the same UCOPY command, the UCOPY used by RMF is distinct (and incompatible) with the UCOPY used by CONNECT (i.e. there are two separate programs with the same name). Both versions of UCOPY may be run on the same machine. UCOPY for RMF usually resides on the system (deadstart file). UCOPY for CONNECT resides under user name LIBRARY, and is automatically acquired when a file transfer is initiated.

To summarize, RMF and CONNECT refer to programs which run on a micro. UCOPY is a program which runs on a CYBER. There are two different versions of UCOPY - one which supports RMF and one which supports CONNECT.

17.2 NOS 2.3

With the release of NOS 2.3, CONNECT replaces RMF as the standard CDC product. RMF will remain available and supported, but will have no new features added.

UCOPY is released as a standard part of NOS 2.3 and supports only CONNECT. UCOPY for RMF is available as it has been in the past and still requires a separate license (product number T200-002). If you already have the RMF version of UCOPY, it will still run. It will not conflict with the version of UCOPY released with NOS 2.3 because the latter is installed as a permanent file on user name LIBRARY, rather than being installed on the deadstart file.

CONNECT, the micro program, must be ordered separately. Currently, CONNECT is available only as an imbedded function in the Micro IPF Version 2 application. The communications section of Micro IPF uses a module of CONNECT to communicate with a CYBER host, using UCOPY as released with NOS V2.3. Currently Micro IPF Version 2 is available for the CDC 110, IBM* PC, Zenith* Z-150, and the Zenith* Z-100. Stand alone versions of Control Data CONNECT will be available in the near future. Separate Software Availability Bulletins (SABs) will be distributed for them.

UCOPY for CONNECT is released in binary only. It is installed as public direct access file UCOPYV2 under user name LIBRARY. The procedure file containing procedures used by Control Data CONNECT is installed on public indirect access file RMUGET under user name LIBRARY. These procedures are used to see if a file is local, create a permanent file, submit a file directly from the micro, etc. This UCOPY requires a NOS 2.1, level 580 or later system.

The differences in the protocol used by CONNECT and RMF are:

1. RMF turns the CYBER prompt off, and an end of block is indicated by an ASCII ETB (27 octal) followed by three Cyclic Redundancy Check (CRC) characters and a carriage return.

CONNECT leaves the CYBER prompt on. An end of block sent from CYBER to micro is indicated by an ETB and three CRC characters, followed by a carriage return, line feed, question mark, and a space. A block sent from micro to CYBER is terminated in the same manner as with RMF.

The advantage of CONNECT (over RMF) is that it can be sure of working without typeahead support (e.g. through the Remote Diagnostic Facility (RDF) using the two port mux) as it waits for the question mark prompt from the CYBER. RMF Version 2 relies on typeahead.

If your version of RMF ignores all characters after the CRC characters, this difference will not cause any problems.

2. RMF uses an ASCII EOT (04) as the end-of-transmission block indicator.

   CONNECT uses an ASCII BELL (07).

   Some public data networks may discard an EOT, but will pass a BELL. This incompatibility will cause an RMF used with a Control Data CONNECT version of UCOPY to abort after the last data block has been sent (on the end-of-transmission block).

3. RMF retries a block transfer three times before aborting.

   CONNECT retries 25 times.

4. There is an additional parameter on CONNECT's UCOPY command - micro type. This parameter is not used in the release of UCOPY at NOS 2.3. In the future this parameter will be used to distinguish between an RMF and a CONNECT protocol transfer.

5. Only UCOPY is needed on the CYBER to support RMF.

   CONNECT also requires a procedure file, RMUGET, which resides on user name LIBRARY.

6. RMF provides only minimal terminal support. CONNECT provides terminal emulation. This allows the use of full screen mode, including the NOS Full Screen Editor (FSE).

Thus, for most versions, the only change needed to make RMF work with the version of UCOPY released with NOS 2.3 is to have it recognize (and send) an ASCII BELL as the end-of-transmission block indicator.

The user interface differences on the micro end are far more significant. Here the two bear little resemblance. CONNECT is more user friendly than RMF:

1. CONNECT makes sure a file exists before trying to transfer it from CYBER to micro. RMF will attempt to transfer an empty file.

2. CONNECT attempts to GET or ATTACH the file to be sent to the micro if the file is not local. RMF requires the file to be local.

3. When transferring from micro to CYBER, the file may be made local or permanent. If the transfer will destroy an already existing file, the user is warned and given an opportunity to change their mind. RMF creates only local files, and destroys any existing local file of the same name without warning.

4. CONNECT uses ESC X to get back to the menu, rather than the CTRL-Q of RMF (which conflicts with CTRL-S CTRL-Q flow control).

5. CONNECT allows you to SUBMIT a file from the micro directly to the CYBER input queue.

6. CONNECT provides an automatic directives capability similar to that of RMF, but easier to use. Instead of the complex command syntax used by RMF, the user simply enters the commands as they would do normally. This allows a user to build a directive file to transfer one or more commonly used files with one command. It does not provide RMF's capability of auto-dialing or of sending interactive commands to the host.

17.3 Future Plans

In a future release of NOS, a new version of UCOPY, called PFTF, will be released which supports both CONNECT and RMF, as well as adding support for XMODEM (Christensen) protocol. PFTF will have the entry points UCOPY and XMODEM. It will be released as a standard part of NOS Version 2, and will replace all previous versions of UCOPY. PFTF may reside on the deadstart file, or may be run as a global library. PFTF will be completely upward compatible with existing RMF and CONNECT software.

Control Data CONNECT will continue to be enhanced to add additional functionality and user interface improvements. It will also be supported on additional microcomputers as appropriate for CYBER customers.

* IBM is a registered trademark of International Business Machines, Inc.

* ZENITH is a registered trademark of ZENITH corportion.

(This page left intentionally blank.)

CHAPTER 18

HUMAN INTERFACE IMPROVEMENTS

18.1 Description

This feature changes operating system generated error messages to be more accurate and user friendly. The major emphasis is to change user error messages and corresponding documentation. Two of the options for the ENABLE and DISABLE IPRDECK and DSD commands have also been changed for consistency. The majority of changes include the following:

- The use of COMMAND instead of CONTROL CARD or STATEMENT.

  e.g. "ROUTE COMMAND ERROR."

- The more general term EXTENDED MEMORY replaces ECS.

  e.g. "EXTENDED MEMORY NOT AVAILABLE."

- INCORRECT or UNKNOWN replaces ILLEGAL and INVALID when appropriate.

  e.g. "INCORRECT MACHINE ID."
  "UNKNOWN DEVICE TYPE-- LFN= filename."

- MONITOR REQUEST replaces RA+1.

  e.g. "MONITOR REQUEST CALL WITH INCORRECT FUNCTION CODE."

- USER replaces ACCOUNT and USER NAME replaces ACCOUNT/USER NUMBER.

  e.g. "USER NAME REQUIRED FOR *TT* OPTION."

18.2 Compatibility

Many error messages have been reworded. Also, the following options for the ENABLE and DISABLE IPRDECK entries and DSD commands have been changed:

- SECONDARY USER COMMANDS replaces SECONDARY USER CARDS.
- USER EXTENDED MEMORY replaces USER ECS.

18.3 Affected Documents

Pub. No.	Title
60459380	Network Terminal Users Instant
60459300	NOS Version 2 Analysis Handbook
60459310	NOS Version 2 Operations Handbook
60459370	NOS Version 2 System Programmer's Instant
60459390	NOS Version 2 Diagnostic Index
60459500	TAF Version 1 Reference Manual
60459680	NOS Version 2 Reference Set, Volume 3, System Commands
60459690	NOS Version 2 Reference Set, Volume 4, Program Interface

CHAPTER 19

NOS PROCEDURE AND FLOW CONTROL COMMAND ENHANCEMENTS

Section 4, NOS Procedures, of the NOS Reference Set Volume 3 contains major revisions to clarify procedure concepts and usage. Also, the title of section 6 has been changed from Execution Control Commands to Flow Control Commands.

19.1 Search User Name LIBRARY for NOS Procedures

CCL has been enhanced to automatically search user name LIBRARY for the file containing the procedure if the file is not local or in the user's permanent file catalog. The system first searches for an indirect access file, then for a direct access file.

19.2 New Symbolic Names

The following symbolic names have been added for Flow Control Commands:

PL Page length. Default=60 lines, min=16, max=255.
PS Same as PL.

PW Page width. Default=136 characters, min=40, max=255.

PD Page density. Default=6 lines per inch, 6 or 8 allowed.

MON	Value=1	Symbolic names with fixed values that can be compared with the symbol WEEKDAY.
TUE	Value=2	
WED	Value=3	
THU	Value=4	
FRI	Value=5	
SAT	Value=6	
SUN	Value=7	

FAMILY Current family name, set by the system.

SLE An error flag (EF) value for a service limit error.

SC2 Symbolic name designating the SCOPE 2 Operating System. It can be compared with the host operating system (SYS) value within an expression.

The page setting symbols were actually implemented in NOS 2.2 L596, but not documented, Their use was described in the NOS 2.2 L596 feature notes, SMD 130581.

19.3 Compatibility

The numerical value returned for the symbolic name WEEKDAY has been changed to 1 through 7 for Monday through Sunday. It was 0 through 6 for Saturday through Friday.

TJE has been deleted as a possible error flag (EF) value since NOS error processing never sets EF to TJE. However, an error will now occur if TJE is referenced in a NOS procedure.

19.4 Affected Document

Pub. No.	Title
60459690	NOS Version 2 Reference Set, Volume 3, System Commands

CHAPTER 20

BLOCK COMMAND

20.1 Description

The BLOCK command enables users to add one or more lines of block letters to a file. Each block letter is 10 columns by 10 lines. The large block letters provide easier identification of output listings. Users can add additional banner pages providing information such as PHONE, NAME, SEND TO, HOLD FOR, etc. This information can help provide more positive identification and distribution instructions.

BLOCK command parameters specify whether or not to rewind the file before writing the block character lines and the carriage control character to be inserted before the first line. The default is no rewind and top of form. The file will be overwritten by the block character lines if it is rewound.

Each block character line can be either a string of up to ten characters, or one of the following special values: DATE(current date), TIME(current time), USER(current user name), UJN(user job name), or JSN(job sequence name). Any character can be used to delimit the block character lines. Consecutive delimiters generate blank lines. Following is an example BLOCK command, using * as a delimiter:

```
BLOCK,BLOCKF.*MYJOB*DATE
```

The string MYJOB will be centered on the first block character line of file BLOCKF. The current date will occupy eight block character (yy/mm/dd) positions on the second block character line.

Block character example:

```
 MM       MM  YY       YY     JJJJJJJJ   00000000   BBBBBBBBB
 MMMM   MMMM   YY     YY      JJJJJJJJ  0000000000  BBBBBBBBBB
 MM MMMM MM     YY   YY           JJ     00      000 BB       BB
 MM  MM  MM      YYYY            JJ     00     0 00 BB       BB
 MM  MM  MM       YY             JJ     00    0  00 BBBBBBBBB
 MM      MM       YY             JJ     00  0    00 BBBBBBBBB
 MM      MM       YY       JJ    JJ     00 0     00 BB       BB
 MM      MM       YY       JJ    JJ     000      00 BB       BB
 MM      MM       YY       JJJJJJJ      0000000000 BBBBBBBBBB
 MM      MM       YY        JJJJJ        00000000  BBBBBBBBB
```

20.2 Common Decks

Two sizes of block characters are defined in common deck COMTBAN, 10 columns by 10 lines and 12 columns by 16 lines. Common deck COMCBAN calls COMTBAN and generates the block characters. COMCBAN selects the small characters if the symbol SMCH$ is defined. SMCH$ is defined in deck NOTE for the BLOCK command. PP programs can only use the large characters (12 columns by 16 lines).

20.3 Affected Document

Pub. No.	Title
60459680	NOS Version 2 Reference Set, Volume 3, System Commands

CHAPTER 21

CLASS COMMAND ENHANCEMENTS

21.1 Description

Previously, the CLASS command could change the service class of your current job only. A JSN parameter has been added to allow you to change the service class (sc) of any job you have submitted for execution. You cannot change the class of an interactive job connected to another terminal. Format:

```
CLASS,SC=sc,JSN=jsn.  or  CLASS,sc,,,,jsn.
```

The following message will be issued to both the user's dayfile and the account dayfile indicating the initial service class of the job:

```
ABSC,sc.
```

Also, the available service classes/priority display is modified to include input files. Example:

```
/class
                AVAILABLE SERVICE CLASSES

                  ---RELATIVE PRIORITY---
 CLASS  INPUT FILES   EXECUTING JOBS   OUTPUT FILES
 DI     *             ******           ***********
 IO     *             *******          ***********
 TS     ***********   ***********      ***********  CURRENT

 ENTER CLASS: di
 CLASS COMPLETE
/
```

21.2 New Error and Dayfile Messages

1. Inclusion of the OT, L, or OP parameters with the JSN parameter will issue:

   SC ONLY PARAMETER VALID WITH JSN.

2. Excluding the SC parameter with the JSN parameter will issue:

   SERVICE CLASS REQUIRED WITH JSN.

3. Entering "CLASS,SC=sc,JSN=jsn.", when the number of jobs with the specified class (sc) is already at the service limit will issue:

   SERVICE CLASS FULL.

4. A successful "CLASS,SC=sc,JSN=jsn." command for a job other than your current job will cause the following message to be issued to the dayfile of the job whose class was changed:

   CLASS CHANGED EXTERNALLY.

5. Specifying an undefined service class will issue:

   UNDEFINED SERVICE CLASS.

6. Specifying a service class that is incorrect for the user or the origin type of the job specified by the JSN parameter will issue:

   INCORRECT SERVICE CLASS.

7. Specifying a JSN that is not in the system or does not belong to the calling job will issue:

   JSN NOT FOUND.

8. Specifying a JSN for an on-line job connected to another terminal will issue:

   CANNOT CHANGE CLASS OF ON-LINE JOB.

21.3 GETUSC Macro Change

The GETUSC macro is changed to return the lower bound priority for input files. This is returned in bits 35-24 of the return word. This field in the returned parameter word was previously zero.

21.4 QAC Call Changes

The following changes have been made to the prefix portion of the QAC parameter block:

Error code = 20. Incorrect service class. The service class selected on an ALTER function is not valid for the user or not valid for the origin type of the specified job.

Error code = 21. Service class full. The service class selected on an ALTER function is at the service limit.

Error code = 22. QAC cannot alter the service class of an online job.

The following changes have been made to the alter portion of the QAC parameter block:

Bit 19 of the alter flags is used to indicate a service class change.

Bits 6-11 of word 15B are used for the requested service class.

21.5 Affected Documents

Pub. No.	Title
60459680	NOS Version 2 Reference Set, Volume 3, System Commands
60459690	NOS Version 2 Reference Set, Volume 4, Program Interface
60459840	NOS Version 2 Administration Handbook

CHAPTER 22

EFFECT COMMAND

22.1 Description

The EFFECT command enables or disables the output format effectors supplied by IAF. Format effectors cause predefined formatting operations such as space 2 lines before output, position to start of current line before output, space 1 line after output, etc. They do not appear on the output device and are not included in page width calculations. The allowable format effectors are documented in the Network Access Method Reference Manual, pub. no. 60499500.

The default for this command is EFFECT,ON (IAF supplies the effectors). EFFECT,OFF enables user supplied format effectors. The first character of every output line is processed as the format effector under EFFECT,OFF. This command has no effect on multiplexor terminals.

Control bytes 0001, 0002, 0003, 0005, 0013, 0014, and 0015 should not be used with EFFECT,OFF because they include format specifications to be supplied by IAF. Consequently, these control bytes conflict with EFFECT,OFF and can cause unpredictable results.

22.2 Affected Documents

Pub. No.	Title
60459680	NOS Version 2 Reference Set, Volume 3, System Commands
60459690	NOS Version 2 Reference Set, Volume 4, Program Interface

CHAPTER 23

ERRMSG COMMAND

23.1 Description

The ERRMSG command enables or disables the display of error messages generated by commands executed within a NOS procedure. ERRMSG,OFF disables the display of error messages at the terminal. The messages are still listed in the dayfile. ERRMSG,ON is the default (error messages are displayed). This command has no effect when issued from a batch job.

The ERRMSG command must be executed from within a procedure. The error message off status remains in effect until an ERRMSG,ON command is issued, or until your job returns to interactive command status. The error message off status is not affected by procedure nesting levels such as REVERT or other BEGIN commands.

23.2 Affected Documents

Pub. No.	Title
60459370	NOS Version 2 System Programmer's Instant
60459670	NOS Version 2 Reference Set, Volume 2, Guide to System Usage
60459680	NOS Version 2 Reference Set, Volume 3, System Commands
60459690	NOS Version 2 Reference Set, Volume 4, Program Interface

CHAPTER 24

FCOPY COMMAND ENHANCEMENTS

24.1 Description

FCOPY is enhanced to read and write stranger format (F=S) tapes containing 8-bit characters, either ASCII or EBCDIC. This is specified by the new code set options ASCFL and EBCFL for the PC and NC parameters. Data on the tape is organized into blocks of fixed length records (lines). Parameters FL and LB have been added to allow the user to specify the record and maximum block lengths.

The code set ASCII88 has also been added for other files. It specifies the full 8-bit ASCII code in 8-bit bytes.

Parameters PL and NL have also been added to support additional line terminators. The line terminators that can be specified are zero byte (default for most code sets), carriage return, form feed, line feed, unit separator (default for the ASCII88 code set), record separator, carriage return and line feed, line feed and carraige return, or an octal character value.

24.2 Compatibility

The NOS 2.3 release of FCOPY is fully upward compatible with the NOS 2.2 release.

24.3 Affected Document

Pub. No.	Title
60459680	NOS Version 2 Reference Set, Volume 3, System Commands

CHAPTER 25

GET REPRIEVE INFORMATION

25.1 Description

This feature allows a user program to get the existing reprieve conditions. This enables secondary applications (those called by other applications) to establish their own reprieve conditions. The new user callable function RRI (134B) is added to CPM. The new macro GETRI calls RRI to return the control point type, extended reprieve block or exit address, and error mask from the contents of control point word EECW. GETRI returns a parameter value of zero if a reprieve condition is not set. If a reprieve condition is in effect, GETRI returns its type.

25.2 Example Application

Secondary applications often need to establish their own reprieve conditions in order to perform appropriate error processing. However, any reprieve conditions which may have been set by the calling application are overwritten when the secondary application establishes its own reprieve conditions. The reprieve conditions for the secondary application may be inappropriate for the calling application and/or reprieve processing code may be overlayed by the caller when control is returned. Therefore, the calling application must save its reprieve conditions and disable reprieve before calling the secondary application, and restore its reprieve conditions when control is returned. The example below illustrates this use of GETRI.

GETRI is used by the Information Processing Family (IPF) and other applications. The IPF initial processing state, including RECOVR and the extended reprieve parameter block, is overlaid by secondary states within the application field length. Prior to NOS 2.3, IPF had to determine the reprieve conditions of the previous state by using the absolute address of the reprieve stack within RECOVR.

25.3 Affected Documents

Pub. No.	Title
60459370	NOS Version 2 System Programmer's Instant
60459690	NOS Version 2 Reference Set, Volume 4, Program Interface

25.4 GETRI Example

```
          IDENT  GETRIEX
          SST                 FOR PPTEXT/NOSTEXT SYMBOLS
          SYSCOM B1
 RRC      SUBR
*         RRC RESTORES REPRIEVE CONDITION FROM GETRI INFO.
*         IT ASSUMES A REPRIEVE IS NOT IN PROGRESS.
          SB1    1            B1=1
          SA1    RPVINFO      OLD REPRIEVE STATUS FROM GETRI
          ZR     X1,RRCX      IF NO CONDITION SET
          MX0    5
          BX6    -X0*X1       MASK OUT FLAG BITS
          NG     X1,RRC2      IF REPRIEVE TYPE IS *EREXIT*
          LX1    1
          MX0    0
          NG     X1,RRC1      IF NORMAL REPRIEVE
*         X6=EXTENDED REPRIEVE PARAMETER BLOCK ADDRESS
          SA3    X6           EXTENDED REPRIEVE PROCESSING
          MX0    12
          LX0    24
          BX7    X0*X3        X7=PARAMETER BLOCK LENGTH
          SX2    B1+B1
          BX7    X7+X2        STORE *SETUP* FUNCTION CODE
          SA7    A3           IN PARAMETER BLOCK
          SX2    B1           SET EXTENDED REPRIEVE BIT
          LX2    18
          BX6    X6+X2        FOR RA+1 RPV CALL
 RRC1     SX1    3RRPV        FORM RA+1 REQUEST
          LX1    42
          SX2    B1           SET RECALL BIT
          LX2    40
          BX1    X1+X2
          BX6    X6+X1        RA+1 RPV CALL FORMAT
          RJ     =XSYS=       POST RPV REQUEST
          ZR     X0,RRCX      IF NORMAL REPRIEVE
          BX7    X3           1ST WORD OF PARAMETER BLOCK
*         RESTORE OLD LENGTH & FUNCTION IN PARAMETER BLOCK
          SA7    A3
          EQ     RRCX
 RRC2     SX1    X1           ISOLATE REPRIEVE ADDRESS
          EREXIT X1           ISSUE ERROR EXIT TYPE REPRIEVE
          EQ     RRCX
 RPVINFO  BSS    1            GETRI REPRIEVE INFO
```

```
          .
          .
GETRIEX   SB1    1           B1=1
          GETRI  RPVINFO
          SA1    RPVINFO
          ZR     X1,APPCALL  IF NO REPRIEVE CONDITION SET
          NG     X1,APPCALL  IF *EREXIT* ENABLED
          LX1    1
          PL     X1,LBL1     IF NOT NORMAL REPRIEVE
          SYSTEM RPV,R       CLEAR NORMAL REPRIEVE
          EQ     APPCALL
LBL1      LX1    1
          PL     X1,APPCALL  IF NOT EXTENDED REPRIEVE
*         CLEAR EXTENDED REPRIEVE
          SYSTEM RPV,R,CLRPV,1
          .
APPCALL   LOAD & CALL APPLICATION
          .
          RJ     RRC         RESTORE REPRIEVE CONDITION
          .
*         PARAMETER BLOCK TO CLEAR EXT REPRIEVE
CLRPV     VFD    48/25,12/2
          BSSZ   25
          END    GETRIEX
```

CHAPTER 26

RECLAIM COMMAND ENHANCEMENTS

The RECLAIM utility released at NOS 2.2 L602 enables NOS users to easily provide magnetic tape or disk backup for permanent files, and maintain greater control over the size of their permanent file disk space. It is not intended for system operation permanent file backup. Operations should use PFDUMP/PFLOAD, which preserves file history information such as file creation date, last date modified, and last date accessed. RECLAIM sets these values to the date and time of loading.

The NOS 2.3 release includes enhancements to allow a wider range of file manipulation capabilities. These enhancements are designed to address the needs of the systems oriented user, without affecting the current capabilities directed at the business and production oriented user. The systems oriented user tends to have more need for file transferring capabilities and less for strictly archival/backup capabilities.

26.1 Current Capability Summary

RECLAIM minimizes the amount of user activity needed to dump or load files. For example, by specifying a single dump option on a RECLAIM control command, all files in a user's catalog can be dumped to magnetic tape. The dumped files can then be reloaded to the same or a different user name as permanent files or as local files. All files are reloaded

with the same permissions, modes, categories, etc., as they had when dumped. The creation, access and modification dates, however, will reflect the load date.

Information about dumped files (e.g., when dumped, on what tape) is maintained in a direct access database in a user's catalog. A single database may contain dump information about files for more than one user, if two or more users agree to maintain a common database.

RECLAIM can be used by both batch and interactive users.

26.2 New Features

The following file manipulation capabilites are supported by RECLAIM, in addition to existing permanent file archival capabilities:

- A new RECLAIM parameter (NV) specifies that a dump is to be written at end-of-information (EOI), regardless of what precedes the EOI. At least one valid RECLAIM dump must precede the EOI if NV is omitted.

- Dump of local files as well as permanent files. This is provided by the new option:

  FN=filename

  The FN option functions exactly the same as the PF option with one exception; when used with a DUMP directive RECLAIM first checks for a local file by the specified name. If one is found, the local file is dumped. A dummy PFC constructed for this purpose will indicate a private file with no permits. The TY option can be used to specify whether local files are to be dumped, and hence loaded, as direct access or indirect access files. The default is direct access (D).

- Dump to and load from a local or permanent mass storage file. This is provided by the new options CF and CN for the COMPACT directive, and DF and DN for the DUMP directive.

- Optionally load over an existing permanent file via the new RP option for the COPY and LOAD directives.

- Dump or load files without maintaining a RECLAIM database, via the RECLAIM parameter DB=0. The database has one four word record for each file dumped and one four word record for each reel dumped.

- Exception processing - select all files except those matching selection criteria, via the EX option used with several directives.

- Files are now dumped in the sequence specified by the user via the PF or FN option. If a file is listed more than once, multiple copies will be dumped in the order listed. If the PF and FN options are omitted, files are dumped in the sequence in which they are found in the user's permanent file catalog.

- Diagnostic information indicates when a file specified in selection criteria was not found and processed.

- The default for the maximum number of files processed (NF option) is now unlimited for all directives with one exception; the limit is still 4095 for the dump directive.

- Dump tape format remains compatible with PFDUMP.

- Elimination of unnecessary dump and load tape repositioning and manipulation. For example, a file load from a multi-reel dump will only process those reels actually containing the files to be loaded. This requires specification of a RECLAIM database via the DB parameter. The multi-reel load improvement is implemented using a new tape keyword on the FILINFO macro. NOS 2.2 dumps can not make use of this feature since the database does not contain the required reel information. Also, there is no capability to convert old databases. However, dumps to new reels of an old dump set will take advantage of this feature.

26.3 Compatibility

RECLAIM now dumps files in the sequence specified by the user. Formerly, the files were always dumped in the sequence in which they were found in the user's permanent file catalog.

26.4 Affected Document

Pub. No.	Title
60459680	NOS Version 2 Reference Set, Volume 3, System Commands

CHAPTER 27

REDO COMMAND

27.1 Description

The NOS 2.2 R command is now available as REDO or R. It allows you to modify and reenter a previously entered command without retyping the entire command. The system prints the characters OLD: , followed by the command to be modified. On the following line, the system prints the characters MOD: , as a prompt for you to enter modifications to the line. When you enter a carriage return, the system prints the characters NEW: , followed by the modified command and executes the modified command.

27.2 Affected Document

Pub. No.	Title
60459680	NOS Version 2 Reference Set, Volume 3, System Commands

(This page left intentionally blank.)

CHAPTER 28

SHOW COMMAND

28.1 Description

The SHOW command calls an interactive procedure to display a screen formatting panel. SHOW is used for testing; it enables a user to display a panel without writing a program that calls the panel. Format:

```
SHOW,panelname.
```

Panelname specifies the name of a compiled panel in user library PANELIB or in a global library set. Since the SHOW command is an interactive procedure, you can get help information by entering SHOW? .

28.2 Affected Document

Pub. No.	Title
60459680	NOS Version 2 Reference Set, Volume 3, System Commands
60460430	NOS Version 2 Screen Formatting Reference Manual

(This page left intentionally blank.)

CHAPTER 29

COBOL VERSION 5.3 ENHANCEMENT

29.1 New Feature

The external data feature provides another method of sharing data between COBOL programs. Specification of the external clause in a record description entry in Working Storage makes the contents of the item described by the entry available to all COBOL subprograms which describe the same item with an external clause.

29.2 Incompatibilities

This feature does not introduce any incompatibilities. The word "EXTERNAL" is already reserved as part of the external file feature.

(This page left intentionally blank.)

CHAPTER 30

DATA CATALOGUE VERSION 2.0 ENHANCEMENTS

30.1 New Features

Data Catalogue can now read the source code for DMS-170 databases, and generate the $UPDATE directives to add the information to the catalogue. Source for the schemas, subschemas, and for the master directory can be processed. FILE control statements which are used in creation of the schema are also processed.

Reports generated by Data Catalogue have been updated to reflect the DMS-170 usage of items in the catalogue.

30.2 Incompatibilities

The format of the catalogue has changed in Version 2.0, and existing catalogues are not directly readable. There is a technique for converting to the new format of the catalogue, and this is discussed under "Migration", below.

30.3 Migration of an Existing Catalogue

The migration of an existing catalogue is accomplished in four steps:

1. Use $COPY HIERARCHIES of the existing Data Catalogue to convert the contents of the existing catalogue into directives for $UPDATE. Call this file TARGET.

2. Use the new Data Catalogue to initialize a new catalogue.

3. Use $UTILITY MIGRATE of the new Data Catalogue to convert the contents of the TARGET file into the format and content required for the $UPDATE of the new version of Data Catalogue. Call this file NEWTARG.

4. Use $UPDATE of the new Data Catalogue to read NEWTARG and create a new catalogue with all the information from the old catalogue.

CHAPTER 31

FORTRAN 5, CHANGES TO GETPARM

31.1 New Features

NOS 2.3 introduces some changes in GETPARM, the routine called to access user parameters from the command line (e.g., LGO,P1,P2). The changes have two main effects:

1. GETPARM can now handle continuations of command lines.

2. GETPARM can now handle values delimited by dollar signs.

Details of how to use these features are included below, as well as some examples and cautions.

31.2 Continuation Lines with GETPARM

GETPARM has been changed to interpret a period which immediately follows the command terminator to mean the command line is continued on the next command line. For simplicity, this special terminator will be referred to as two periods, although it could also be a right parenthesis

followed by a period. The two periods are syntactically seen as a comma. For example,

```
LGO,P1,P2,P3..
P4,P5,P6.
```

is then seen (through the eyes of GETPARM) as:

```
LGO,P1,P2,P3,P4,P5,P6.
```

Similarly,

```
LGO,P1,P2,P3,..
P4,P5=..
P6,P7.
```

is interpreted as:

```
LGO,P1,P2,P3,,P4,P5=,P6,P7.
```

Also remember that the terminator of a command line is not simply the last non-blank character. The following example shows a mistake of this kind:

```
LGO,P1,P2)P3,P4..
```

In this example, there is no continuation of the command line. A right parenthesis was used to make the terminator more obvious.

There is no limit to the number of continuation lines that GETPARM can read.

If command line continuations are used interactively, GETPARM will read each continuation line from the terminal, using '..' as a prompt.

There are two known anomalies of the continuation processing, and they are both addressed by PSR FL5A668. The symptoms are listed below so you can be aware of this behavior.

1. When interactively entering continuation lines, and at the '..' prompt, a valid continuation line must be entered. If a carriage return only is entered at this point, the job step is aborted with:

FORMAT ERROR ON CONTROL CARD

2. Be careful that there are some parameters on the command line before the two periods. For example,

   LGO..

   gets aborted with the message

   FORMAT ERROR ON CONTROL CARD

31.3 Delimited Parameter Values

GETPARM will now also properly accept parameters which have embedded special characters. Values of this sort must be delimited by dollar signs ('$'). For example,

LGO,P1,P2=$ABC.DEF$,P3=$XYZ$$123$.

is now correctly interpreted as having the following values:

P2 - ABC.DEF
P3 - XYZ$123

31.4 Caveats...

This new interpretation of continuations of command lines means that some command lines will now be interpreted differently by GETPARM than before. The following are of the cases which could cause problems or confusion.

1. Do not enter a terminator at the end of a command line entered at the Loader interactive prompt: 'LDR>?'. The Loader unconditionally appends a period to the end of the line for you. If you enter one, you may unknowingly make it look as if there should be a continuation of the command line.

2. Do not enter a terminator at the end of a command line entered with the OS directive of Query Update. The OS directive unconditionally appends a period to the command line entered (the OS directive issues a command line to the operating system).

When in doubt, use DAYFILE to see what command lines were actually passed to the operating system (and from there to GETPARM). This can help spot the other cases of redundant periods being added by intermediate routines.

CHAPTER 32

IAF ABORT PROCESSING ENHANCEMENTS

Enhancements have been made to IAF to assure that registers used by system macros, the ITA and ITO requests, and the exchange package are retained in the dump after abnormal termination. The following information is saved before abort processing takes place:

1. The name of the calling routine.
2. The contents of registers X1, X2, and A1 used by the system macros.
3. The last ITA and ITO requests issued.
4. The exchange package.

These features have been added to make dump analysis and debugging easier.

Included in these changes, sense switch 2 has been added to mean "DO NOT ATTEMPT A RELOAD AFTER AN ABORT" when the switch is set to 'on'. Termination processing will check to see if switch 2 is set and, if so, IAF will terminate on the abort, active users will be logged off (with the messages "IAF TERMINATED" and "ABORT") and the dump and dayfile will print immediately. If switch 2 is not set, the message "RECOVERY COMPLETE" is displayed and the recovery flag is set.

(This page left intentionally blank.)

CHAPTER 33

OPERATOR NOTIFICATION OF INSUFFICIENT RESOURCES

Prior to NOS 2.3, a job requiring an 'OFF' equipment to satisfy its resource demands was rolled out until such equipment became available. Resources already assigned to the job remained assigned to it until it finished with them thus depleting the resources available to other jobs and raising the possibility of a scheduling deadlock.

This feature has been added to inform the system operator whenever a job's resource demands require the use of one or more 'OFF' nonallocatable equipment and allow the operator to identify the job needing the equipment and intervene.

NOS will inform the operator via the equipment preview (E,P) display when a job requests a temporarily unavailable equipment and allow the operator to terminate the job with the appropriate diagnostic message or to allow the job to remain waiting for the resource to become available.

The DSD E,P display has been changed so that the blank before the packname/vsn is changed to '*' when the job requires an 'OFF' equipment. The following is an example of the changed E,P display:

```
E,P.  RESOURCE REQUESTS.

JSN  EQ   PN/VSN   USERNAM RING LAB        STATUS        LEVEL

ABCD MT   VSNnnn   USER123 IN             MT060         LVL2
                                      RING CONFLICT
AAAN PE   *TEST1   USER456  --  YES                      LVL0
```

(This page left intentionally blank.)

CHAPTER 34

INSTALLATION DEFINED SYSTEM EVENTS

Installations may now provide user callable, system functions without requiring site modifications to NOS. This feature has been added to provide an adequate inter-control point scheduling interface for installation provided pseudo subsystems.

An installation may define up to eight system events. When a user program (non SYOT) enters one of the reserved system events, an installation designed pseudo-subsystem, which is ordinarily rolled out, is then scheduled.

An example of an installation provided, user callable system function implemented with a pseudo subsystem is a utility called to attach a permanent file when the permanent file may be archived on tape. If the permanent file is not disk resident, the utility enters the system event which then requests the pseudo subsystem to retrieve the permanent file. The user called utility waits for an archive retrieval completion by attempting attaches of the permanent file on disk. The pseudo subsystem searches the tape library directory for the file, copies the requested file from the tape library to disk, and makes the file permanent.

The DSD rollout (R) display has been changed to display the new installation defined event codes (10 through 17) in the status field. The mnemonic for installation defined events is 'IE'.

(This page left intentionally blank.)

CHAPTER 35

INDIRECT ACCESS PERMANENT FILE PROCESSING ENHANCEMENT

Prior to NOS 2.3, when performing a copy operation (GET, OLD, SAVE, REPLACE, or APPEND) on an indirect access file, PFM performed the copy by repeatedly reading the next 8 sectors from one file into a buffer in the PP and then writing them to the other file. On a non-buffered device this procedure led to losing one disk revolution every eight sectors; on a buffered device, a transfer rate of less than 20% of the possible full speed of transfer was maintained. PFM remained in the PP for the entire operation, and it may have kept the same disk channel reserved for the entire operation.

While this procedure was adequate for small files, there may have been a serious impact on total system performance when large indirect access files were used.

For NOS 2.3, if a file to be transferred is over a certain threshold value (defined in COMSPFM and has a released value of 6 PRUs), PFM calls a CPU program (CPUPFM) which performs the transfer via CIO calls. CPUPFM will not be called if the caller is a subsystem or if the caller is a DMP= program. The entry point to CPUPFM is CPF (Copy Permanent File).

PFM communicates with CPUPFM through SPCW in the control point area, through the DMP= parameter block (DMPN) in NFL, and through four special files. The SPCW request to CPUPFM has the following format:

```
18/3LCPF,2/1,1/1,21/0,18/FC
```

where FC is the CPUPFM function code
FC = 0 copy from permanent file to local file
FC = 1 copy from local file to permanent file

CPUPFM returns the following status in SPCW on termination:

```
24/0,4/1,8/EC,24/0
```

where 1 indicates this was CPUPFM not RESEX, and EC is the error code of the error encountered (if any).

PFM writes the following parameter block into NFL when calling CPUPFM:

```
36/0,24/LF
36/0,24/APLF
60/PFID
60/PFID
60/PFID
60/PFID
```

where LF = length of the local or permanent file; APLF = length of the original file (for APPEND) - if non-zero this file is copied before copying the local file; PFID = permanent file id (device number, track, sector and entry ordinal of the PFC; formatted as a PFM special request block) of the PFC entry for the original file on an APPEND.

PFM creates four files when calling CPUPFM:

```
PFM*LFN (local file FNT)
PFM*PFN (permanent file FNT)
PFM*APF (APPEND original file name)
PFM*ILK (catalog track interlock FNT)
```

CPUPFM calls a new common deck COMCCPF to do the actual copy.

CHAPTER 36

ACPD/CPD ENHANCEMENTS

Performance tools enhancements in this area include:

1) The ability to select an ACPD report interval by CPD sample file record count.
2) Changes in the collection and reporting of secondary rollout activity.
3) Changes to the CMR statistical data area.
4) Deletion of certain account dayfile messages.

An IC parameter has been added to the ACPD command to allow the specification of the number of CPD sample file records per report interval. This allows the selection of report intervals of less than one minute. Using both the IC and IN parameters on the ACPD command generates an error and results in the message "IN AND IC PARAMETER CONFLICT".

The system now collects the number of rollouts to secondary rollout devices and the number of sectors rolled out to secondary devices. These accumulators and the total rollouts and sectors rolled out accumulators reside in the new statistical data area. All of the accumulators will be updated by the RTCM CPUMTR function.

All of the items in the statistical data area (SDAL) in CMR which really are statistical data have been moved to the extended statistical data area which has been renamed the statistical data area. The tag SDXP has been removed and

replaced by the tag SDAP as the pointer to this area. New tags have been defined for those words in the old statistical data area which contain non-statistical information. The remaining words have been freed up for use by the site or applications. The following is the format of the new statistical data area:

WORD	TAG	BITS	DESCRIPTION
0	JS0S	59-36	EJT scans
		35-0	Scheduled jobs
1	JS1S	59-48	Reserved
		47-24	Jobs preempted
		23-0	Jobs scheduled
2	JS2S	59-48	Reserved
		47-24	Jobs scheduled without service constraints
		23-0	EJT scans with insufficient CM to schedule job
3	JS3S	59-48	Reserved
		47-24	EJT scans with insufficient EM to schedule job
		23-0	EJT scans with no control point available to schedule job
4	JS4S	59-48	Reserved
		47-24	Rollouts for resource limits
		23-0	Count of time slices
5	MTRS	59-48	Reserved
		47-36	Count of clock updates missed
		35-24	Worst case MTR MXN time
		23-12	Worst case MTR cycle time
		11-0	Current MTR cycle time
6	CMMS	59-0	CM storage moves
7	EMMS	59-0	EM storage moves

10	CBNS	59-0	No. of times CPUMTR communication buffer was not available
11	PRXS	59-0	PP priority exchanges
12	ROTS	59-0	Number of rollouts
13	SOTS	59-0	Rollouts to secondary rollout devices
14	ROSS	59-0	Sectors rolled out
15	SOSS	59-0	Sectors rolled out to secondary rollout devices

Note that the count of missed clock updates which was maintained in the MTR input register is now in the statistical data area (word 5).

The following account dayfile messages will no longer be issued by SFM each time one of the system dayfiles is accessed: SDCA, SDCI, SDCM, SDMR, SDMS, SDNF, SDPX, SDSF, SDTO and SDTS. The data reported in these messages is available from both ACPD and PROBE.

(This page left intentionally blank.)

CHAPTER 37

DISK ERROR TRANSPARENCY

NOS maintains an error log (ERRLOG) which contains both recovered and unrecovered error messages for hardware errors. Originally this information was intended for the system operator and as input to HPA. With its implementation, the binary maintenance log (BML) was to contain the hardware error messages which were to be input to HPA. The ERRLOG was to continue to contain messages for system operator notification.

Many messages being issued to the ERRLOG that should be logged to the BML, could create the mistaken impression of serious disk problems. Also, since all retries are logged, the ERRLOG might become very large requiring excessive disk space and affect the performance of HPA which reads the entire ERRLOG.

Disk error transparency as a feature, addresses the migration of error messages of 844, 885-11/12, FSC and 834 disks from the ERRLOG to the BML. The following changes in manner of reporting normal and reserve errors has been implemented:

1. Error retries will not be logged.
2. Recovered errors will only be logged in the BML.
3. Unrecovered errors will be logged in the BML, the job dayfile, the system dayfile and the ERRLOG.

If the number of errors for a device exceeds defined limits within a one hour period (either recovered or unrecovered errors) an ERRLOG ALERT message will be posted in the A,OPERATOR display. The operator will be directed to look in the ERRLOG for information on the potentially failing device.

The limits are defined in COMSMSP and can be set individually for each type of mass storage device. The released values for all device types are: for unrecoverd errors, one and for recovered errors, fifty.

CHAPTER 38

FLAW COMMAND ENHANCEMENTS

To relieve the analyst from having to convert the physical address to the logical address before using either FLAW or the APRDECK to flaw tracks on a disk, the flaw utilities have been changed to allow specification of both physical and logical disk addresses. The following new commands replace the RTK, STK and TTK commands used in the FLAW utility and in the APRDECK:

SLF - Set Logical track Flaw
CLF - Clear Logical track Flaw
SPF - Set Physical track Flaw
CPF - Clear Physical track Flaw
CAF - Clear All Flaws

The following applies to the physical flawing commands:

1. They will work with any mass storage device.
2. Use of address ranges for extended memory is still valid.
3. Specification of a particular cylinder, track and sector for rotating mass storage devices is valid.

A table with all of the relevant information needed when flawing tracks can be found in the Analysis Handbook (pub. 60459300) either under the section on the FLAW utility, or the section on the APRDECK.

(This page left intentionally blank.)

CHAPTER 39

ALTERNATE CATLIST SECURITY

39.1 Feature Overview

This feature introduces a user-specified file permission which defines whether alternate users may obtain information about the file using CATLIST. This feature enables the file owner to keep file names and other information private, even though the files are accessible.

The CHANGE, DEFINE, and SAVE commands and macros are modified to process a new parameter, AC, which specifies the file's alternate CATLIST permission. AC=Y/AC=N specify that the file may/may not be CATLISTed by alternate users, respectively. A file will not appear in a CATLIST of an alternate user name if alternate CATLIST permission is not granted for the file. Alternative CATLIST permission may be specified for public, semi-private, and private files. Files that were defined before the feature is installed will be non-CATLISTable by default.

The CATLIST command is modified to display the alternate CATLIST permission for each file when a full list (LO=F) is requested.

The PF utilities display the alternate CATLIST permission for each file when LO=T is requested.

39.2 MACRO Modifications

CHANGE, DEFINE, and SAVE Macro Modifications

The alternate CATLIST parameter is added. Other parameters remain unchanged.

ac Alternate CATLIST permission. Specifies whether alternate users may obtain information about the file using CATLIST.

Mnemonic	m	Value	Description
ACNO	N	1	no.
ACYS	Y	2	yes.

CATLIST Macro Modifications

The CATLIST macro will return the alternate CATLIST permission in the permanent file catalog entry buffer.

39.3 Other Changes

PF Utilities Modifications

The PF utilities PFATC, PFCAT, PFCOPY, PFDUMP, and PFLOAD will display the alternate CATLIST attribute for each file listed when the LO=T option is specified.

Permanent File Catalog Entry Modifications

The alternate CATLIST permission field, FCAP, has been added to the PFC.

Word	Tag	Bits	Description
6	FCAP	47	Alternate CATLIST Permission. 0 = File not CATLISTable by alternate users. 1 = File CATLISTable by alternate users.

PFM Call Block (FET) Modifications

The alternate CATLIST parameter, CFAP, has been added to the PFM FET.

Word	Tag	Bits	Description
17	CFAP	47-46	Alternate CATLIST Parameter. 00 = No change. 01 = Clear alternate CATLIST permitted. 10 = Set alternate CATLIST permitted.

(This page left intentionally blank.)

CHAPTER 40

TERMINAL I/O AT LOGOFF

40.1 Feature Overview

This feature introduces a new set of account file messages that display the combined total of terminal characters input and output at the end of each account block and at the end of a job. A message indicating the combined input/output characters is also displayed on a user's terminal at logout.

IAF maintains a combined count of characters input and characters output. This accumulator will have a maximum value of 16777215. This accumulator will be displayed on the user's terminal at logoff and in the user dayfile and the account file at the end of each account block, and at end of job. If the combined accumulator overflows, an overflow message will be displayed in the account file. The current UDCI, UDCO, UECI and UECO messages will continue to exist, but the UECI and UECO messages are no longer issued at the time that IAF detaches or terminates a job. Instead, these messages are output at the end of job as is done for all other resource usage accumulators.

40.2 Characters Transferred Terminal Message at Logout

The following message is displayed on the terminal just after the SRU message.

CHARACTERS=xxxxx.xxxxCHS.

xxxxx.xxx is the combined input/output character count divided by 1000.

40.3 New Account File and User Dayfile Messages

Three new account file messages have been added:

UDCT, xxxxxx.xxxKCHS.

This message is issued to the account file and to the user dayfile at the end of an account block and will display the total characters input and output by the job up to the current time minus any overflows of the combined input/output accumulator which may have occurred.

UECT, xxxxxx.xxxKCHS.

This message is issued to the account dayfile and to the user dayfile at end of job and will display the total characters input and output for the entire job minus any overflows of the combined input/output accumulator which may have occurred.

UCCT, 16777.215KCHS.

This message is issued to the account file whenever the combined input/output accumulator overflows. When this message is issued the value 16777215 will be subtracted from the total input/output accumulator. Note that the combined input/output total is not cleared because of the way this count is maintained in the IAF terminal table. This message is not issued to the user dayfile because the job is typically under IAF control when this overflow occurs.

40.4 Changes to Existing Account and User Dayfile Messages

The current UDCI, UDCO, UECI, and UECO messages have been retained but have a slightly different meaning. Currently, the UECI and UECO messages are issued whenever an online interactive job is either detached or terminated by IAF. The character counts are not recovered if a detached interactive job is later recovered by a user. This means that the UDCI and UDCO messages represent the character counts only for the time since login or the last recovery and not for the entire account block which can span several detaches and recoveries. If the account block ends while the job is not online, the messages are not displayed at all. Likewise, the UECI and UECO messages do not represent the entire job, but only the time from login or recovery to detach or termination.

This feature causes the input and output accumulators to be retained on a termination or detach and to be recovered on detached job recovery. Therefore, the UDCI and UDCO messages will represent the total input and output counts for the entire account block (minus any overflows which may have occurred) whether the job is online or not. The UECI and UECO messages will be issued at the end of job, consistent with other resource accumulators, and will represent the input and output counts for the entire job minus any overflows which may have occurred.

(This page left intentionally blank.)

CHAPTER 41

DEFAULT CHARGE PROCESSING

41.1 Feature Overview

This feature provides for a more consistent method of processing default charge information, as well as making the CHARGE/PROJECT number available to the System Account File and Negative Field Length.

41.2 USER and CHARGE Command Processing

The syntax of both commands will remain the same; however, their functions have been altered as follows:

1. USER command processing will continue to look forward to the next command in the command stream for a CHARGE command as it does now. The following occurs when:

    A. Charge is required:

        a. If a CHARGE command is not found, the default charge/project is validated. If found valid, the system, project, and/or user

prologue(s) are initiated. (For interactive users, the file INPUT is read if a default charge and/or project does not exist, or if the default charge is found invalid.)

Whenever the charge and/or project is found invalid, the job will be aborted.

b. If a CHARGE command is found in the command stream, CHARGE processing is initiated. In this case, CHARGE processing will initiate the system, project, and/or user prologue(s) after validating the specified charge and project.

B. Charge is not required.

a. If a CHARGE command is not found in the command stream, the default charge will not be examined and processing will immediately initiate the system and/or user prologue(s) if they exist.

b. If a CHARGE command is found, the charge/project will be validated. If found valid, CHARGE processing will initiate the system, project and/or user prologue(s). If found invalid, the system and/or user prologue(s) will be initiated and the illegal CHARGE command will be reissued to the CHARGE processor, causing the job to abort.

2. The CHARGE(*) command will cause the default charge and project to be validated if a charge and/or project are present in the user's validation file entry. If both the charge and project fields are blank and charge is not required, it will be treated as a no-op and processing will continue. If the default charge and/or project are found invalid, and charge is required, the user will be aborted. If charge is not required, the user will not be aborted, and processing will continue with the next command after issuing the "INVALID DEFAULT CHARGE" message.

3. A new account file message (ABIC) will be issued after every primary USER account file message (ABUN). This new message specifies the default charge/project fields associated with the user. If both fields are

zero, no message is issued. This message is issued regardless of whether or not the user has "charge required" set. This is issued before the prologue file is initiated.

Batch Processing

ABUN, username, familyname, terminalname.
ABIC, charge, project.

Terminal Processing

ABAP, C1, username, familyname, terminalname.
ABAP, C2, application
ABIC, charge, project, terminalname.

41.3 Negative Field Length (NFL) Additions

The charge/project a user is currently running under is saved in a three word block in the NFL. The tags for this block are CHGN and PRJN respectively and are located between word SCVN and word SHCN. The charge/project is no longer written in the input file system sector.

41.4 GETCN Macro

GETCN is a user callable macro which issues the Get Charge/Project Number function call.

The GETCN macro returns a four word block containing the current charge/ project number combination beginning at the specified address. Word four, bit 59, is clear if this is a nonvalidated charge/project number. This bit will be set if this is a validated charge/project.

Macro format:

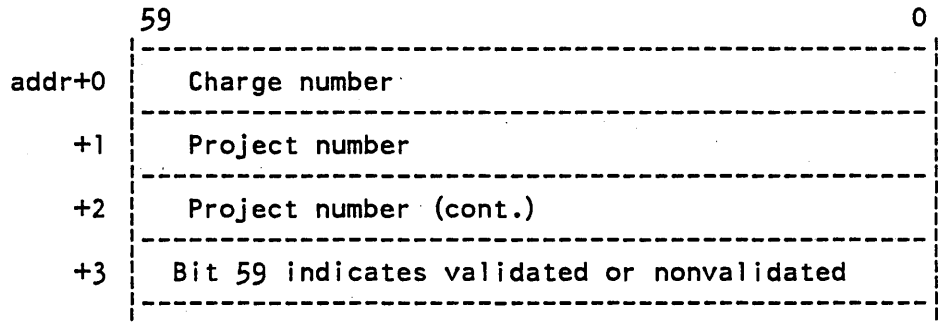
```
------------

Location        Operation       Variable
--------        ---------       --------

                GETCN           addr
```

addr Address of the reply block.

The following information is returned to the four word reply block beginning at location addr:



CHAPTER 42

PERMANENT FILE CATALOG - CHARGE AND PROJECT NUMBER

42.1 Feature Overview

Previously, there was no record of the charge/project number under which a file was created. This feature stores the charge/project numbers in the Permanent File Catalog (PFC). This feature places the charge and project numbers that a user is executing under into a permanent file's catalog entry when that file is defined or saved and when the parameter CP is used in the CHARGE command.

The system's CATLIST routine has been changed so that when a user requests a full CATLIST (LO=F), the charge and project number is placed on the fourth line of its description.

An additional line has been added to the reports given by PFCAT. This line contains the charge/project numbers.

42.2 CHANGE Command Changes

The CHANGE command has a new parameter, CP. This parameter is used to notify the PFM change function to replace the

current charge and project numbers in the catalog entry with the charge and project numbers the user is currently executing under. The CP parameter has also been added to the system macro CHANGE. The format of the CHANGE command follows:

```
CHANGE,nfn=ofn,...,nfn=ofn/CP
```

42.3 Format of Permanent File Catalog (PFC) Entry

The format of the 16 Word PFC has been changed. The words 12B - 14B, which were unused, now contain the charge number (12B) and the project number (13B - 14B).

Word 12B	60/charge number
Word 13B	60/project number (first 10 characters)
Word 14B	60/project number (second 10 characters)

42.4 CHANGE Macro

The CHANGE macro has a new parameter value for the Special Request (SR) field, the value is CP. This value is used to notify the PFM change function to replace the current charge and project numbers in the PFC entry with the charge and project numbers the user is currently executing under.

NOTE: The new parameter CP and the existing parameter CE cannot both appear on the CHANGE command.

An example of the use of the CP parameter in a macro call follows:

```
CHANGE FET,MYFILE,,,,PU,,,,CP
```

42.5 CATLIST Output Format Changes

The CATLIST command output now contains the current charge and project number of the file, when LO=F is specified. The command and macro CATLIST will not return the charge/project numbers for an alternate CATLIST unless the caller has SSJ= privileges.

42.6 Permanent File Utilities Output Format Changes

The PFATC, PFCAT, PFCOPY, PFDUMP, and PFLOAD routines output now has the charge and project numbers of the file.

(This page left intentionally blank.)

CHAPTER 43

PASSWORD RANDOMIZATION

43.1 Feature Overview

This feature introduces a user validation permission which defines whether the user's login password will be randomized. This feature enables an installation to guarantee that the user's password is changed to a new value when the PASSWOR command is executed.

The number of random characters to be appended to the end of passwords is defined in COMSACC and has the label PWRC. The PWRC constant can be a number between 2 and 5, and has a released value of 2.

The MODVAL command is modified to process a new user validation parameter, COPR (override password randomization), which specifies whether password randomization is required.

The PASSWOR command is modified to test if password randomization is required (COPR=0). COPR is bit 23 of the access control word (AAWC), and is set by default. If password randomization is required, the user is permitted to specify all but the last PWRC characters of the new

password. The system will provide PWRC random alphanumeric characters, which must be entered as the last characters of the new password. The length of the new password, including the random characters, must be greater than or equal to the minimum required password length and less than or equal to seven characters.

43.2 MODVAL Command Modification

A new MODVAL conversion option, CV=D, is used when converting a pre-NOS 2.3 user validation file to NOS 2.3. The option is only valid on a creation run and will validate all newly created users for "override" password randomization.

43.3 LIMITS Command Modification

The following message will appear in the LIMITS output as a valid user permission if COPR is set:

DEFINE NON-RANDOMIZED LOGIN PASSWORD

43.4 PASSWOR Modification

The PASSWOR command may only be entered from an interactive job if password randomization is required, and may be used to change either the interactive or batch password. If password randomization is not required, the batch password may only be changed from a batch, remote batch, or system origin job, and the interactive password may only be changed from an interactive job. Users will not be permitted to enter a password expiration date without specifying the old and new passwords, regardless of whether randomization is required.

CHAPTER 44

RESTRICT USER T0 DEFAULT CHARGE

44.1 Feature Overview

This feature provides the capability to guarantee that given user names will execute specified protected project prologues.

A new access permission, CNRD (charge not restricted to default), has been defined in the user validation file. When CNRD is set (default condition), the user may use any valid charge/project number when issuing a CHARGE control command. When CNRD is clear, the user may only use the default charge/project number defined in the user validation file when issuing a CHARGE control command. If CNRD is clear and no default charge/project is defined, the user may use any valid charge/project number. CNRD is bit 22 of the access control word (AACW).

The associated project prologue is not executed when a secondary USER command is entered. Therefore, users will not be permitted to enter a secondary USER command which is restricted to default charge.

If the primary user is restricted to default charge and the secondary user is not, the secondary USER command will be permitted and will be restricted to the primary user's default charge.

44.2 MODVAL

44.2.1 LIMITS Command Output

The following message will appear in the LIMITS command output as a valid user permission if CNRD is set:

NOT RESTRICTED TO DEFAULT CHARGE/PROJECT NUMBER

44.2.2 MODVAL Conversion Option CV=C.

This option is used when converting a pre-NOS 2.3 user validation file to NOS 2.3. The option is only valid on a creation run (OP=C) from source and will validate all users for "charge not restricted to default" (CNRD is set).

44.3 CHARGE

The CHARGE control command will abort and issue the message:

CHARGE RESTRICTED TO DEFAULT

to the system and user dayfiles if the user is not validated for "charge not restricted to default" (CNRD) and requests a charge/project number other than the default.

44.4 USER

The USER control command will abort and issue the message:

INCORRECT USER COMMAND

to the system and user dayfiles if a secondary USER command is entered which is not validated for "charge not restricted to default."

(This page left intentionally blank.)

(This page left intentionally blank.)

CHAPTER 45

SECURITY VIOLATION TRACKING

Previously, if a user's security count was zero or infinite, the SISC message was not issued to the accounting dayfile. With this feature, the SISC message will be issued, regardless of the security count value.

CHAPTER 46

/CHARGE FOR SUBMIT

This feature implements a /CHARGE formatting directive for use with the SUBMIT command.

/CHARGE inserts a CHARGE command identical to the one currently in effect for the submitting job. If no charge is in effect, nothing is added to the submit file during reformatting.