Control Data Corporation

Network Operating System (NOS)

Version 2.1

Level 580/577

Feature Notes

Number 6

Copyright. Control Data Corporation, 1983.

SMD130412

Table of Contents

High Speed Terminal Support 3
Jeff Koniges, NOS Field Support

SHELL Processing 7
Paul Grade, NOS Field Support

SHELL Example Procedure Listings 13
Paul Grade, NOS Field Support

High Speed Buffered Input/Output Support 21
Jim Marchak, NOS Field Support

Remote Diagnostic Facility 27
Paul Grade, NOS Field Support

Network Features 33
Cheng Chow, Network Field Support

High Speed Terminal Support

Introduction

IAF high speed terminal support is primarily intended to increase the performance and usability of interactive applications which must send large amounts of data to IAF terminals at speeds of up to 9600 bits per second (BPS). While terminals running at high line speeds have been supported by previous NOS systems, the flow of data to these terminals was not always continuous thus decreasing the effective line speed. This was most apparent in applications which use the graphics mode of terminals such as the CD/2000 application.

High speed terminal support smooths out the flow through internal restructuring of some IAF and NOS components. There are beneficial side effects to this feature. Terminal input which has been typed ahead is processed more quickly. System overhead has been reduced for most terminal I/O, thus freeing system resources for useful work.

History

High Speed Terminal Support was first written for NOS 1 and distributed to selected CD/2000 users. The feature was standardized and included in NOS 1 at level 552. Changes were necessary to include the feature in NOS Version 2 due to the internal differences in the two NOS systems. It is now available in the 2.1 (level 580) release of NOS Version 2.

Usage

This feature is invoked automatically for all IAF users regardless of the terminal speed. The amount of data buffered in IAF differs, depending on the terminal speed. All users take equal advantage of the improved type ahead processing.

No program changes are necessary to gain some performance benefits. Programs may need to be retuned, however, to gain maximum benefits. Previously, large data buffers in user programs were a way to smooth output to a high speed terminal. These are no longer necessary and may induce a rollout for the job. There are some cases where programs may not execute exactly as before. These cases are noted later in this article.

Implementation

The major design change implemented by this feature is to prevent the mandatory rollout of jobs in order to perform terminal input and output. The original design in this area would cause a job to be rolled from central memory to mass storage whenever output was written to or input requested from a file assigned to an interactive terminal. The reasoning for this was based on the thought that terminals operating at 110 or 300 BPS are so much slower than the central computer that a job should not tie up central memory while waiting for the data to be sent to the terminal. Since the concept of type ahead was also not present, the system would roll out the job whenever input was requested in order to wait for the user to type something. The widespread use of high speed communications and the implementation of type ahead have required this design to be changed.

Since jobs were rolled for terminal I/O operations, the programs responsible for transferring the data between IAF and the user were the rollout and rollin programs 1RO and 1RI, plus the IAF helper 1TO. On output, a job was first rolled to mass storage, then the data was read from the rollout file and sent to IAF. 1RI was responsible for transferring input data to the job after rolling it in. Under NOS 2.1, a new PP program exists, 1MI, to handle interactive input and output. 1MI is named following the convention of the other device drivers, 1MS and 1MT. Code to process interactive input has been removed from 1RI. 1RO and 1TO are still involved in interactive output and are used when more data is to be written to the terminal than IAF can buffer. 1TO, however, no longer translates data to ASCII. This is now done entirely within IAF.

When a job posts a CIO request, CPUMTR (CPUCIO) assigns the request to 1MI if the file is assigned to a terminal. On input, 1MI checks the type ahead queue in IAF for this terminal. If data is available, it is transferred at this time to the job thus eliminating the rollout. If

data is not available for the job, 1MI issues a rollout request for the job and enters 1MI into the PP recall stack. When data is available, the job is rolled in and 1MI recalled to process the read request. Note that 1RI is no longer involved in terminal input.

On output, 1MI is again called to process the request. 1MI requests pots (data buffers within IAF) and transfers data from the job to IAF until the write is complete or the maximum amount of data is queued within IAF for this connection. This maximum is based on the network block size for this terminal which is assumed to be a function of the terminal speed.

When IAF cannot accept any more data, but the user buffer has not been emptied, 1MI checks to see if the job is active. If the job is in recall, 1MI requests the job be rolled out and lets 1RO continue the output operation. If the job is still active, that is, the write request has been issued without recall and the job has not entered recall for another reason, 1MI enters the PP recall stack to wait for IAF to get rid of some of the data. Once recalled, 1MI continues the output process. If, after coming back from recall, IAF has not disposed of any data, 1MI will roll the user job and let 1RO/1TO continue. Thus, if a job generates data faster than IAF, NOS or the Network can process it, the job will still be rolled to mass storage. On a typical write, data is sent to IAF, the job regains control and continues with this and subsequent job steps without waiting for data to be received by the terminal.

A change incidental to this feature is that two monitor functions, TSEM and TGPM, have been moved from MTR to CPUMTR.

List of Files Processing

List of files refers to an address within a user program containing pointers to FETs for files in use by a program. The address of the list is set by the program and has a default of RA+2. The list is processed by the operating system at job step completion to assure that any data in user buffers but not yet written to files is flushed. This is especially useful for abnormal job step termination. Volume 4 of the NOS Version 2 reference set contains the appropriate details needed to use a list of files.

Another use for this list of files involves terminal input and output. When a program reads from the terminal, NOS looks at the first FET from the list of files and flushes any data present in the buffer if flush conditions are met. Both the write and the read are processed at the same time. Using list of files processing, a write followed by a read, such as a prompt for input, will cause only one rollout for the job instead of two. This, of course, was before high speed terminal support. The double rollout has been eliminated by this new feature. There is no longer a need to use the list of files processing to reduce system overhead. A simple write followed by a read will be almost as

efficient for the system to process and much easier for the user to program.

1MI processes the list of files so that programs will execute as before but with the following exception. If data is available in the type ahead queue and the user job has not used auto recall on the read, 1MI will not process any list of files output. The solution for programs which depend on this feature is to insure auto recall is used on all reads from the terminal.

Control Byte Translation

IAF allows the content of output data being sent to a terminal to alter the characteristics of subsequent terminal I/O operations. This is called control byte processing. Certain control bytes may change the way IAF handles the next input request. A situation exists, again involving type ahead, where a program may execute differently as a result of high speed terminal support. Control bytes are now detected in job output by IAF instead of being detected by 1TO. They are detected after the data has been transferred to IAF and therefore after the write may be complete from the program's point of view. If a read request is issued by the program and type ahead data is available, the data may be delivered to the program before control bytes present in previous output have been interpreted and appropriate bits set in the terminal table. This results in data being delivered to the program in the wrong mode. A solution to this is to use the CSET macro instead of control bytes to change input translation mode. This request will be processed before any subsequent input requests will be processed by IAF, thereby insuring that data is delivered correctly. The CSET macro only changes between display code and 6/12 display code. Control bytes are required to use the 12 bit ASCII code set. Type ahead should be avoided, therefore, for transparent mode operations.

SHELL Processing

A SHELL program provides you with the capability to create a screening program for all commands. Possible uses of this feature are:

- Validating a user number to a site developed SHELL program which permits commands that execute only 'legal' applications.
- Writing a SHELL program that processes a non-NOS syntax for commands.

SHELL Command or Macro

The SHELL command (it is also available as a macro) permits you to set controls for a command monitoring program, i.e., a SHELL program. The SHELL command describes:

- the name of the SHELL program
- where the SHELL program is to be found (local, library, or SYSTEM)
- monitoring level (interactive end of job step, outside procedures, inside procedures)
- clear controls if SHELL program aborts

Descriptions of the SHELL control options are listed on page 8.

The name of the SHELL program and SHELL control bits are stored by CPM in word SHCN of your negative field length.

When the SHELL program is called, the current command is passed in CCDR (location RA+70B through RA+77B). If there is no current command (as when you're monitoring interactive end of job step), RA+70B will be zero.

The SHELL program may take any action whatever as a result of the command. In particular, it may issue the same or another NOS command using the EXCST macro. The SHELL program is not called for commands which the program itself originates through the EXCST macro.

Entering:

SHELL.

clears word SHCN, and thus disables SHELL screening. Of course, if a SHELL program is in effect, it may not permit you to enter the SHELL command, and thus may prevent you from clearing the SHELL screening.

Note that if you wish to monitor commands issued at interactive end of job step, your SHELL program must also read commands from the terminal. This is because 1AJ will load your SHELL program at interactive end of job step, and unless you READ from the terminal, 1AJ will keep loading your SHELL program.

SHELL Validation through MODVAL

You can also declare a SHELL monitoring program for a user name through the MODVAL subcommand:

SH=name of SHELL program

This will cause 1AJ to use the named SHELL program to screen commands as soon as the user enters the system. Where and when 1AJ searches for a SHELL program to process commands is indicated by the setting of SHELL control options.

The SHELL control options, set by the MODVAL subcommand SO=, are:

ABTM Clear SHELL processing if the screening program aborts. This should be set for all but the most restrictive SHELL programs, or there will be no way for a user to escape an incorrectly coded SHELL program.

The next SHELL control options inform 1AJ where to search for the SHELL program.

LFID Allows load of SHELL program from a local file.
GLID Allows load of SHELL program from a global library.
SLID Allows load of SHELL program from SYSTEM.

The remaining SHELL control options are to stipulate when 1AJ uses the SHELL program. You should consider how restrictive your SHELL program will be, along with what overhead you wish to incur. This is because if you set all these SHELL control options, the SHELL program will be loaded for every command 1AJ processes.

TCSM Commands are monitored directly from terminal input, i.e., the SHELL program is called at interactive end of job step (when the job runs out of commands).
BCSM Commands are monitored outside procedures. The SHELL program is loaded before each command is processed, unless the command is inside a procedure.
CCLM Commands are monitored inside procedures.

SHELL Program Example - restricting a user name to an application

As mentioned before, one use of a SHELL program could be to permit a user to access only certain applications. In other words, anyone using a designated user name could not, say, compile programs, save files, etc. Instead, all one could do is access controlled application(s).

This can be accomplished by the following.

. Write a menu procedure to provide the user with the application choices. The SHELL program can then be a simple COMPASS program which executes this menu procedure through the EXCST macro. The menu procedure would invoke other procedures to actually perform the chosen application.

  These procedures and the absolute of the COMPASS program can be placed in a global library.

  Procedure BLDSGL, listed below, contains the commands, procedures, and program to do this. It constructs the user library, STATSGL, on user name LIBRARY.

. Validate a user name for that SHELL program. Set the SHELL control options so that the SHELL program is loaded from a global library, and monitoring is done on commands from terminal input and batch.

  Also validate the user name for a user prologue which will make STATSGL a global library.

  Procedure CRESHUN, listed below, creates a user name with SHELL permissions.

An example of using CRESHUN is to create:

```
user name=STATS
password=STATS
charge,project=STATS,PACKAGE
user prologue=STATPRO
SHELL program=LODMENU
```

You would call CRESHUN from a DIS job:

```
BEGIN,CRESHUN,CRESHUN,
UNM=STATS,PWM=STATS,CNM=STATS,PJM=PACKAGE,UPM=STATPRO,
SHELLNM=LODMENU.
```

NOTE
Because of its length, you should put the above command in a procedure, and then execute the procedure.

. Finally, move the user prologue, STATPRO, to the user index associated with user name STATS. You will have to use DIS and the SUI command for this, as if you try to log in to STATS, 1AJ will attempt to use the SHELL program, and you subsequently will not be able to do anything.

Thus, users who enter the system through STATS will execute the user prologue STATPRO, which will make STATSGL a global library.

After execution of the user prologue, 1AJ will load the SHELL program LODMENU from the global library set. This is because the user name was validated for SHELL monitoring of timesharing commands (SO=TCSM) by the SHELL program LODMENU (SH=LODMENU) which is loaded from a global library (SO=GLID).

LODMENU then executes the menu procedure STATS which causes the user to get a menu of available applications displayed at his terminal.

The user can only enter menu choices - anything else is diagnosed as an error by the CCL menu processor. Upon entry of a valid choice, the menu procedure can invoke other procedures which actually execute the desired application. Commands in these procedures are not monitored by the SHELL program as the user name is not validated with SO=CCLM (monitor commands inside procedures).

If the user tries to abort a program, 1AJ will reload LODMENU, which as mentioned above, executes STATS, and the menu is again displayed to the user. This prevents the user from entering commands other than application choices.

This method is quite simple and powerful. All you have to do is write procedures to execute the applications, and the SHELL monitoring ensures the user name can only run those procedures.

SHELL Program Example - allowing a non-NOS syntax

In this example (listed on page 17), a simple SHELL program, SHELDEM, is used to permit commands which are longer than 7 characters. The SHELL command:

```
SHELL,SHELLB,L,T.
```

causes the local file SHELLB (L option) to screen commands whenever a timesharing command (T option) is entered.

Valid commands are: ASSEMBLER, DOCUMENT, and DOVEDANYA. Since SHELLB is loaded at interactive end of job step, there is no value passed in CCDR. SHELLB prompts the user with a time stamp and then accepts the next command inputted by the user. Further processing of the command is caused by execution of a procedure or NOS command associated with the command.

NOTE

For brevity, SHELDEM has been kept simple. It will not handle commands with leading blanks, $, lower case, or internal blanks.

SHELL Example Procedure Listings

This section lists the procedures mentioned in the SHELL Processing section.

```
.PROC,BLDSGL.
.* PGG  09-JAN-83  BUILD USER LIBRARY STATSGL.
.IF(OT.NE.SYO)REVERT,ABORT.  MUST BE SYOT TO ACCESS LIBRARY USER INDEX
.*
.* ASSEMBLE COMPASS ROUTINE TO INVOKE MENU PROCEDURE STATS.
.*  MAKE THIS ROUTINE AN ABSOLUTE ON THE FILE ABS.
.*  USE LIBEDIT TO BUILD A USER LIBRARY CONTAINING ABS AND THE
.*   STATS PROCEDURES ON TO THE FILE STATSGL/UN=LIBRARY.
.*
UNLOAD,ABS,LGO,STATSGL.
SUI,377776.
COMPASS,I=LODMENU,L=0,B=LGO,S=SYSTEXT,S=PSSTEXT.
LOAD,LGO.
NOGO,ABS.
PURGE,STATSGL/NA.
DEFINE,STATSGL/M=R,CT=PU.
LIBEDIT,P=0,U,N=STATSGL,B=ABS,Z.+*FILE,STATSM
UNLOAD,ABS,LGO,LODMENU,STATSGL,STATSM.
SUI,0.
REVERT,NOLIST.
.DATA,LODMENU.  COMPASS ROUTINE TO INVOKE MENU PROCEDURE STATS
           IDENT  LODMENU
           ENTRY  LODMENU
           ENTRY  RFL=
           SYSCOM B1
           LDSET  EPT=LODMENU/RFL=
 LODMENU   SB1    1
           EXCST  STATS
 STATS     DIS    ,*STATS.*
           USE    //
 RFL=      BSS    0            RUNNING FIELD LENGTH
           USE    0
           END    LODMENU
.DATA,STATSM.  PROCEDURES FOR STATS ROUTINES
.PROC,STATS*M
"Statistical Procedures"
,CHOICE=
(1"Cosap"
,2"SPSS"
,3"Minitab"
,4"Bye"
).
.HELP
Execute Statistical Procedures from a terminal.
.HELP,1
Cosap - Conversationally Orientated Statistical Analysis Package.
.HELP,2
SPSS - Statistical Package for the Social Sciences.
.HELP,3
Minitab - worksheet data manipulator.
.HELP,4
Logout your terminal.
.ENDHELP
#.EX.STATS#_CHOICE.
.EOR
```

```
.PROC,STATS1*I.
.*                Commands to be executed for choice 1 in STATS.
.EOR
.PROC,STATS2*I.
.*                Commands to be executed for choice 2 in STATS.
.EOR
.PROC,STATS3*I.
.*                Commands to be executed for choice 3 in STATS.
.EOR
.PROC,STATS4*I.
.*                Commands to be executed to logout the user.
.* The following will NOT work on a 63-character set system.
.*  Note also that the colon is a 6-bit zero character.
ASSIGN,TT,OUTPUT.
NOTE,OUTPUT.+:D
```

```
.PROC,CRESHUN*I
,UNM"User Name"=(*F)
,PWM"Password"=(*F)
,CNM"Charge Number"=(*S10(ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789))
,PJM"Project Number"=(*S20(ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789))
,UPM"User Prologue"=(*F)
,SHELLNM"Name of SHELL Program"=(*F)
,UI"Optional Forced User Index"=(*N=,*S6(01234567))
.
.HELP
 CRESHUN  pgg  09-Jan-83  Create user name with SHELL permissions.

Execute MODVAL to create a user name with a SHELL program.
SHELL load options of

 GLID   global library load
 TCSM   monitor timesharing commands
 BCSM   monitor batch job commands

are validated.
.ENDHELP
.*
.IF(OT.NE.SYO)REVERT,ABORT.  MUST BE SYOT TO EXECUTE
MODVAL,OP=U,I=MODIN,FA.
SUI,O.
.DATA,MODIN.  MODVAL INPUT DIRECTIVES
/UNM,PW=PWM,SC=63
AW=NUL
AW=CLPF,AW=CSPF
AP=NUL
AP=IAF,AP=TVF
IS=BATCH
CN=CNM,PN=PJM,UP=UPM
SH=SHELLNM
SO=NUL
SO=GLID,SO=TCSM,SO=BCSM
.IF($UI$.NE.$$)FUI=UI_B
```

```
.PROC,STATPRO.
.* PGG  09-JAN-83  USER PROLOGUE TO SET UP STATSGL AS GLOBAL LIBRARY.
$ATTACH,STATSGL/UN=LIBRARY.
$LIBRARY,STATSGL.
$REVERT,NOLIST.
```

```
.PROC,SHELLD*I.
.HELP
 PGG  21-Dec-82  Try out SHELL program, SHELDEM.  The binary is SHELLB.
  It is a local file, and because it recognizes few commands, it only
  screens timesharing commands, not commands inside or outside
  of procedures.
.ENDHELP
.IF,FILE(OPL,.NOT.AS).ATTACH,OPL/UN=LIBRARY.
MODIFY,Q,CB=SHELLB,Z./*REWIND,SHELLB/*CREATE,SHELDEM/*EDIT,SHELDEM
SHELL,SHELLB,L,T.
.DATA,SHELDEM
SHELDEM
          IDENT  SHELDEM,FWA
          ABS
          ENTRY  SHELDEM
          SYSCOM B1
          ORG    111B
 FWA      BSS    0

*CALL COMCCMD
*CALL COMCMAC
*CALL COMSTCM     DEFINE TERMINAL CONTROL CODES

 IBUFL    EQU    101B          TERMINAL INPUT BUFFER LENGTH
 OBUFL    EQU    101B          TERMINAL OUTPUT BUFFER LENGTH

 I        BSS    0             TERMINAL INPUT
 ZZZZZII  FILEC  IBUF,IBUFL,DTY=2RTT
 O        BSS    0             TERMINAL OUTPUT
 ZZZZZOO  FILEC  OBUF,OBUFL,DTY=2RTT

 SHELDEM  SB1    1             PRESET B1=1
 SHD1     RJ     GCI           GET COMMAND INPUT
          SA1    CCDR
          ZR     X1,SHD3       IF NO COMMAND
          BX6    X1
          SA6    CMDSA         ENSURE COMMAND WILL BE MATCHED
          SA5    CMDS-1        FWA OF COMMAND TABLE-1
SHD2      SA5    A5+B1
          IX3    X5-X1
          NZ     X3,SHD2       IF NOT MATCHED
          SX4    A5-CMDSA
          ZR     X4,SHD3       IF COMMAND NOT IN TABLE
          RETURN I,R           RETURN TERMINAL INPUT FILE
          RETURN O,R           RETURN TERMINAL OUTPUT FILE
          EXCST  PRCT-CMDS+A5  EXECUTE CORRESPONDING PROCEDURE

SHD3      WRITEW O,SHDA,SHDAL  UNKNOWN COMMAND
          EQ     SHD1          LOOP FOR ANOTHER COMMAND

SHDA      DIS    ,* SHELDEM - UNKNOWN COMMAND.*
SHDAL     EQU    *-SHDA
```

```
CMDS     DATA   L*ASSEMBLER*   VALID COMMANDS
         DATA   L*DOCUMENT*
         DATA   L*DOVEDANYA*  CLEAR SHELL PROCESSING
CMDSA    BSS    1             RESERVE FOR INPUT COMMAND

PRCT     DATA   L*CMPSPRC.*  PROCEDURES (MUST BE SAME ORDER AS CMDS)
         DATA   L*DOCPROC.*
         DATA   L*SHELL.*    CLEAR SHELL PROCESSING.


GCI      SUBR                GET COMMAND INPUT
         SA1    GCIB         CHECK IF CALLED BEFORE
         NZ     X1,GCI1
         SX6    B1           FLAG CALLED BEFORE
         SA6    A1
         MX6    1            SET FLUSH BIT ON OUTPUT
         LX6    36+1
         SA1    O+1
         BX6    X6+X1
         SA6    A1
         SETLOF GCIC         SET LIST OF FILES
         RETURN I,R          RETURN INPUT FILE
         RETURN O,R          RETURN OUTPUT FILE
         REQUEST I,U,NMSG    ASSIGN INPUT TO TERMINAL
         REQUEST O,U,NMSG    ASSIGN OUTPUT TO TERMINAL
         WRITE  O,*          PRESET WRITE CODE FOR IMPLICIT TT OUTPUT
GCI1     PROMPT OFF          TURN OFF IAF *?*
         CLOCK  GCIA         OUTPUT TIME PROMPT
         WRITEW O,GCIA,GCIAL
         READ   I            GET COMMAND
         READO  I
         ZR     X1,GCI2      IF NOT (CR)
         SX6    O
GCI2     SA6    CCDR
         EQ     GCIX

GCIA     BSS    1            TIME STAMP FOR INPUT PROMPT
         VFD    12/0L>,48/0L"EB"  END OF BLOCK TERMINAL CONTROL
GCIAL    EQU    *-GCIA       LENGTH OF TIME STAMP STRING
GCIB     CON    0            ZERO IF FIRST TIME IN GCI
GCIC     VFD    12/,18/GCID,30/   LIST OF FILES POINTER
GCID     VFD    1/,41/,18/GCIDL   LIST OF FILES
         VFD    42/0LZZZZZ00,18/0
         VFD    60/
GCIDL    EQU    *-GCID

IBUF     BSS    IBUFL        TERMINAL INPUT BUFFER
OBUF     BSS    OBUFL        TERMINAL OUTPUT BUFFER

*CALL COMCCIO
*CALL COMCCPM
*CALL COMCLFM
*C^LL COMCRDO
*CALL COMCSYS
*CALL COMCWTW
```

```
            END     SHELDEM
```

High Speed Buffered Input/Output Support

NOS 2.1 supports a new high speed buffered I/O option that is available with all CYBER 170 series mainframes equipped with Extended Semiconductor Memory (ESM); the option is not currently available on models 170-825, 835, and 855. This document provides a description of the feature and discusses the potential benefits of its installation.

Overview

The high speed I/O option consists of 7155-401 controller(s), 885-4x parallel head fixed module disk drives (FMDs), and ESM. Data is transferred directly between disk drives and ESM through the low speed ESM ports. Each disk drive has 4 heads which read/write in parallel thus providing a transfer rate of 4 times the rate of a single head FMD. Data is transferred between ESM and the user's CIO buffer via new routines in the operating system. The buffers serve as a cache for data most recently read or written; data is retained in these buffers on a most recently accessed basis so that records accessed very frequently are immediately available to the requestor without the need to perform physical disk I/O.

Drive positioning time and capacity is identical to that of single head 885 disk models.

Hardware Requirements

The high speed buffer I/O option is available with all CYBER/170 (except models 815, 825, 835, and 855) machines. Peripheral equipment required includes one or more 7155-401 disk controllers, 885-4x parallel head disk drives, and ESM. Each drive can be accessed through one or two controllers and each controller can drive up to 8 disk units. The 7155-401 controller can also be used to drive single head 885 and 844

devices as can the 7155, however, simultaneous use of low and high speed devices on the same controller can severely degrade perfromance of the high speed devices.

ESM requirements vary with workload and peripheral configuration.

885-4x drives cannot be configured to operate as shared devices however multi-machine access to ESM via the high speed ports is permitted and does not impact performance of buffered devices. Conventional mass storage devices (885, 844) can coexist with high performance equipment and the conventional devices can be shared between mainframes. 885-4x spindles can be confirued as 1, 2, or 3 spindle logical devices as can the single head 885 models.

Software Requirements

A software option is available with NOS 2.1 to support the high speed disks described above. This option includes new feature code in CPUMTR to handle data transfers between program buffers and ESM; this code also manages the ESM buffers (cache). A new PP program, 1HP, controls physical I/O between ESM and disk under direction of CPUMTR.

Software Overview

The software modules which support high speed devices consists of a pool of buffers in ESM, CPUMTR segments, and PP routine 1HP. Each buffer is 4000B words long and the buffer pool pool is spread over all high speed devices. Buffers are not permanently associated with any particular device but are obtained from the pool on an as needed bases. Buffer management is performed on a least recently used basis such that the least recently used buffer is reassigned as needed. Routines in CPUMTR, collectively called BUFIO, perform all buffer pool management, transfers of data between the requestor's buffer and ESM buffers, and issue requests to 1HP to initiate data transfers between ESM and disk. BUFIO is part of the CPUCIO package in CPUMTR.

CPUCIO performs all initial processing on I/O requests; this processing is performed in monitor mode, regardless of mass storage device type. If the request is an I/O operation on a buffered device, the BUFIO routines are called. BUFIO (which is embedded in CPUCIO) runs in program mode (at CPU priority 76) and the interface between CPUCIO program mode and CPUMTR monitor mode is similar to that used for communication with the PPs. CPUCIO has its own 8 word communication block in the PP communication area. When CPUCIO (and BUFIO) are called, the requesting control point number and the name CIO are inserted into the input register in the communication block. If BUFIO is already busy when a new request is processed, the request is queued by CPUMTR. Since the interface to CPUCIO (BUFIO) is similar to that of the PPs, this package is referred to as a psuedo-PP.

For read operations, BUFIO determines if the desired logical sector(s) are already resident in ESM; this is done through a hash table. If the data is available, BUFIO transfers the requested data from ESM to the requestor's buffer and completes the request. The ESM buffer is placed at the top of the accessed list to indicate that it was the most recently accessed. If the requested data is not available in ESM, BUFIO posts a request to 1HP to perform a disk to ESM transfer. There is a 1HP request queue associated with each high speed mass storage device. BUFIO will process other outstanding requests while waiting for the disk-ESM transfer to complete. If there are no outstanding requests, BUFIO will exit. BUFIO is restarted via monitor function by 1HP when the disk-ESM transfer is complete, and the original request processing is reinitiated.

The driver, 1HP, resides permanently in a PP throughout system operation whenever high performance devices are present. There is one active copy of 1HP for each channel that has high performance devices configured. 1HP(s) are automatically initiated during the deadstart process if high speed disks are present. There is a request queue associated with each 1HP through which BUFIO issues requests for 1HP to initiate transfers between ESM and disk. Each 1HP controls all of the disks on the channel to which it is assigned. 1HP performs positioning functions and starts I/O transfers. Once a transfer is initiated, it can complete without further intervention from 1HP thus allowing overlap of positioning and transfer operations on different devices.

Each ESM buffer is 4000B words long which is sufficient to hold 10B physical sectors. Each physical sector is 400B words long however logical sectors are still 100B. The blocking-deblocking between logical and physical sectors is performed by BUFIO. For sequential read operations the buffer manager streams data by issuing read ahead requests to 1HP. On write operations, buffers are flushed to disk whenever a buffer is filled (4000 words written), a file is closed, or a positioning operation is performed, or the requesting job is rolled out. Data flushed to disk is still retainined in ESM buffers according to the buffer aging process. PP I/O to buffered devices is performed via the PIOM monitor function.

User Impact

The presence of high speed buffered devices is transparent to system users except for perception of improved performance. Buffered devices can be used for all types of files: permanent, temporary, rollout, system, etc. However, the system deadstart file cannot be INSTALLed on a buffered device, thus deadstart from a buffered device is not possible.

Files residing on buffered devices have the same logical file structure as all NOS mass storage files. No special programming consideration is required of the user as all buffer management and blocking-deblocking is handled automatically by the system.

Performance Impact

The high speed buffer I/O option significantly improves system throughput, especially where I/O bandwidth is the primary limiting factor and CPU utilization is moderate to low. This is likely to be the case under heavy interactive loads or with large I/O bound job mixes. Relative performance improvement depends also on the speed of the CPU, with faster CPU machines having the largest performance improvement potential.

The data streaming potential makes buffered devices very well suited for swapping and system residence. The buffer management technique that is used will tend to keep system overlays in buffers depending on frequency of use; this has the effect of providing a dynamic alternate high speed system device residency based on frequency of use.

Actual performance improvements gained from installation of high speed buffered devices is very configuration and load dependent.

Operational Consdierations

No impact on system operations staff is anticipated.

A new CMRDECK entry is supported which is used to define buffered devices.

```
EQnn=DB-n,status,controller,unit,chan1,chan2,flawdeck,hash.
```

Where nn, status, controller, unit, chan1, chan2, flawdeck are the same as for other mass storage devices.

hash defines the size of the hash table used by BUFIO. it must be an exact power of 2 in the range 40B to 400B.

A new extended memory definition is also provided.

```
XM=mid,buffer space,user EM space,EM.
```

where

mid is the machine ID for the current machine.

buffer space is the amount of space/1000B to be allocated to the buffer pool for buffered devices.

user EM space is the amount of space/1000B to be allocated for user extended memory.

EM	is optional on non-800 series machines. It must be present on models 865 and 875 that have both unified extended memory and external extended memory configured.

In multi-mainframe configurations, the PRESET machine should have the CMRDECK entry XM for each machine in the cluster.

Consult the NOS Installation Handbook for detailed information.

Remote Diagnostic Facility

The Remote Diagnostic Facility (RDF) debuts in NOS 2.1. RDF provides a single user functional equivalent of CCP, NAM, and IAF. The hardware employed for RDF is the two port multiplexer, part of all Cyber 800 series mainframes. An interactive terminal or dialup modem can be connected to one of the ports on the two port multiplexer, and Control Data Engineers can then execute maintenance diagnostic software from a remote location.

At the same time, RDF provides security enhancements to the online maintenance environment.

RDF Package

The RDF package includes the following.

RDFEX

The unconfiugred deadstart tape released with NOS 2.1 contains a copy of RDFEX which supports terminals connected to the two port multiplexer.

IAFEX also supports terminals connected to the two port multiplexer, but is only available if your site has a license for IAF. If your site has a license for IAF, you can substitute IAFEX wherever you see RDFEX in this article.

1TM

The PP driver for the two port multiplexer is 1TM. When RDFEX is

initiated, it checks the equipment status table for a two port multiplexer with ON status.

If there is one, and RDF is enabled, 1TM is loaded into a PP so RDFEX can communicate through the two port multiplexer. If no two port multiplexer entry is found, the equipment is OFF, or RDF is not enabled, then 1TM is not loaded.

The two port multiplexer is configured like any other equipment, i.e., via the EQ entry in the CMRDECK. Its format is documented in the NOS 2.1 Installation Handbook.

RMSHELL

RMSHELL, released with NOS 2.1, is a program which screens commands. Such a screening program is known as a SHELL program (see page 7) The RMSHELL program only executes commands which are part of a table inside of RMSHELL. If RMSHELL is active, and a user enters a command which is not in RMSHELL's table, RMSHELL issues the error message:

```
UNAUTHORIZED COMMAND.
```

and does not execute the command. RMSHELL considers only maintenance commands as authorized commands.

Examples of maintenance commands are:

```
CATLIST   HPA   LABEL   MALET
```

Unauthorized commands include:

```
FTN5   PFDUMP   SYSEDIT
```

A user name can be validated for a SHELL program so that the SHELL program screens every command as soon as the user enters the system. SHELL validation is described on page 8.

Thus, the user name for the Control Data Engineers could be validated for the SHELL program RMSHELL, thereby limiting the engineers to only maintenance usage under that user name.

Operator Communication

Two commands and a macro permit you to communicate with an operator through the A,OPERATOR display. If you have either CSOJ or CMNT (explained below) validation, you can issue the OPMSG command or macro, as in:

OPMSG.PROBLEM SOLVED

This results in the intensified SEE A,OPERATOR message to appear in the upper right corner of the operator console. The operator can use the DSD command A,OPERATOR. to see your message. The display associates your message with your job sequence name (JSN). The operator can then reply through the DSD command CFO. His response appears in your user dayfile.

While OPMSG can be used anytime from any terminal, there is another method to 'talk'. This is accomplished by entering the sequence:

<esc>MS=message

from a logged-in terminal connected to the two port multiplexer. The notation '<esc>' means to press the ESCAPE key.

The effect of this command is similar to OPMSG, except the A,OPERATOR. display indicates your message as originating from a terminal. This allows the operator to send you a message with the DSD command DIAL. The operator's reply is output directly to your terminal.

Validation Changes

CMNT (maintenance user) is a new bit in the access word. User names validated with this permission can access maintenance functions of the system. Every time such a user enters the system, the ERRORLOG dayfile contains the message:

MAINTENANCE ACCESS BY UN=username.

Some of the maintenance functions validated users can perform are:

- executing MALET
- issuing the OPMSG command or macro
- accessing the ERRORLOG and Binary Maintenance Log dayfiles, through the ELD and MAINLOG commands

NOTE
Equipment usage messages which formerly appeared in just the ACCOUNT dayfile are now also written to the ERRORLOG dayfile.

RDF Usage

This section provides an example of using RDF in a setting where NAM or

CCP is not available.

Declare the two port multiplexer

Indicate the equipment as ON in your CMRDECK. Ensure a terminal is connected to the proper port, with the correct baud rate.

Declare mode of RDF

Permit RDF to run through the DSD command:

```
ENABLE,RDF.
```

NOTE
This also sets engineering mode at the operator console.

Have RDF execute in the following possible modes.

privileged While in privileged mode, the SHELL program is not loaded to screen commands. This applies only to users with CMNT validation. The setting of privileged mode has no bearing on SHELL processing for users without CMNT validation.

This mode is enabled by the DSD command:

```
ENABLE,PRIVILEGED RDF.
```

It is disabled (the SHELL program is loaded to screen commands) via the DSD command:

```
DISABLE,PRIVILEGED RDF.
```

The default is for privileged mode to be disabled.

resident If resident mode is disabled (the default), RDF becomes inactive (the PP driver 1TM drops) if the terminal on the two port multiplexer doesn't do anything for 15 minutes. Resident mode is disabled through the DSD command:

```
DISABLE,RESIDENT RDF.
```

If resident mode is enabled, RDF stays active, regardless of terminal activity. Resident mode is enabled by the DSD command:

```
ENABLE,RESIDENT RDF.
```

Initiate RDF or IAF

If your site does not have a license for IAFEX, enter the DSD command:

```
RDF.
```

to initiate the RDF subsystem.

If your site has a license for IAFEX, enter the DSD command:

```
IAF.
```

to initiate the IAF subsystem.

NOTE

Your site may use a different procedure to initiate the RDF or IAF subsystem. Substitute the name of the procedure for the RDF or IAF command mentioned above.

Login

Press the RETURN key a few times on the terminal connected to the two port multiplexer. This should cause the terminal to display the login banner.

Documentation describing terminal usage through RDF can be found in section 3 of the Concurrent Maintenance Library Reference Manual (60455980H). Usage is similar to IAF/NAM, except for some details. A summary of important details is:

correcting input	backspace or CTRL/H key
deleting input	ESCAPE
interrupt execution	I or BREAK key
terminate execution	S key or enter STOP
exit TEXT mode	CTRL/C or BREAK key
typeahead	not available

Network Features

I. NETWORK ACCESS METHOD (NAM)

Release 255x Channel

When the EST entry for a channel connected to a 255x is turned off, NAM releases the equipment so it may be assigned to another program such as a MALET program.

Improved Network Management

Network Products now provide a more usable operator interface for the start-up and operation of the network. Specific capabilities of this feature are:

- A new utility, NAMI, which with a single operator interface starts up NAM and any desired jobs, including supervisory applications, TVF, and a new network dump and trace collection program, COLLECT.
- A NAM K-display which permits the host operator to communicate with NAM and the network supervisor programs NS, CS, and NVF.

Improved Network Supervision

The Network Products loading, configuration and connection protocols are changed to begin the suitable evolution for a multi-host environment. As a result, supervision functions in the host are consolidated so that NS has only dump/load responsibility; CS handles only configuration; and

NAM and NVF manage the connection functions.

Downline Termdef

Application programs are able to set multiple terminal definition parameters in such a manner that if all of the characteristics cannot be set, none will be. An application program may also request the values of any terminal definition parameter. The terminal user may also request a display of a limited set of his terminal characteristics.

II. COMMUNICATION CONTROL PROGRAM (CCP)

Block Mode Terminal Support

This feature permits Network products to better support block mode terminals, cassette devices, and intelligent terminals. Included with this feature is support of X-ON, X-OFF flow control.

- Block Mode Input

  With the new blockmode input feature, the terminal user can place the terminal in block mode, and the user is allowed to change the default end-of-line and end-of-block characters. The input block is forwarded after an end-of-block character is recognized. Also CCP does not send any output until end-of-block character is entered.

- Output Block Size

  Instead of text processing each individual logical line of output network block, the new feature will process the entire output block to create a single transmission block. The maximum transmission block size for each terminal is defined by the XBZ parameter in NDL, Network blocks larger than XBZ will be fragmented into multiple transmissions. Network blocks smaller than XBZ are combined into a single transmission block of a size less than or equal to XBZ. The TIP command AL is changed to abort block (AB). A TIP command entered when output is interrupted will be effective after the current output block has been transmitted.

- Input Flow Control

  A terminal user may use the TIP command [CT] IC=Y to enable the input flow control. If input control is enabled, the Async TIP will attempt to stop input from the terminal when either the upline block limit (UBL) is reached or the number of available CCP buffers reaches its threshold. In this case, a DC3 is sent to stop the input. When the situation is clear, a DC1 will be sent to start

the input. Input flow control is effective for all ASYNC terminals except 2741's.

. Output Flow Control

A terminal user may use TIP command [CT] OC=Y to enable output flow control. If output flow control is enabled, receipt of any DC3 from the terminal will cause the Async TIP to suspend output. A DC1 from the terminal causes the TIP to resume the output at the character following the last character transmitted when the DC3 was received. Output control is effective for all ASYNC terminals except 2741's.

. Upline Blocking and Queueing Algorithms

The upline blocking algorithm uses UBZ instead of PW of 150 as blocking factor. Terminals which cannot send less than 100 character blocks should set UBZ greater or equal to the minimum upline transmission block size. (The terminal user can enter [CT] BF=2 if blocksize is between 100 and 200).

The upline queueing algorithm requires that UBL be set greater than or equal to maximum number of network blocks a terminal may generate before it pauses to receive output.

Synchronous Auto Recognition

Network Products now provide the capability to auto-recognize HASP, BISYNC, and MODE 4 terminals on synchronous lines configured for auto-recognition.

Improved NPU Performance Statistics

The statistics reported by the NPU for lines, trunks, and performance are modified to provide more accurate and more useful information. Only one form of NPU statistics message is generated. The messages can be turned ON of OFF by CCP build option.

Dual Coupler Support

CCP is able to communicate over one or two channel couplers. If configured with 2 couplers, it is able to communicate over both simultaneously.

56KB Trunk and HASP Support

Network Products are able to support 56KB communication lines connecting a 255X with another 255X (trunk) or with a HASP multi-leaving workstation. Special CLA's required are 2563-2 (trunk) and 2560-21/31 (HASP).

X.25 LAP-B Support

CCP now supports only the LAP-B variant of the HDLC protocol.

BREAK Key as User Break 1

Users may use TIP command [CT] BR=Y to select the BREAK key as user-break-1 when output is in progress or as cancel character when input is in progress.

ASYNC Auto Recognition

Async auto recognition recognizes line speeds up to 2400 bps.

Re-auto Recognition

The command to change code set (AC) also allows the user to change line speed. The network expects 2 inputs from the user (2 CR's).

Full ASCII Mode Support - TIP Command [CT] FA

CCP now supports full ASCII mode for input. In full ASCII mode, all 128 ASCII characters may be input and sent to the Host, except the following characters:

- NUL and DEL
- DCS when inputing from paper tape

. [LF] following [EL] or [EB]

. DC1 and DC3 when OC is selected

Multiple Transparent Messages

With TIP command [CT] XL specified, the terminal user will be able to enter multiple transparent messages.

end